TEC | Tecnológico de Costa Rica

FroSigPro: Un marco matemático-computacional para la solución de problemas de optimización aplicado a procesamiento de señales utilizando la norma de Frobenius

Código # 1440037

Departamento Académico Responsable

Matemática

Investigador Responsable

Dr. Juan Pablo Soto Quirós

Otros Investigadores

M.Sc. Jeffry Chavarría Molina

M.Sc. Juan José Fallas Monge

Informe Final

Enero 2019 – Junio 2021

1. Grado de Cumplimiento

Objetivo General: Generar un marco matemático-computacional para los problemas de minimización en el área de procesamiento de señales cuya formulación involucre la norma de Frobenius, lo cual permitirá derivar generalizaciones de los problemas ya conocidos y mejorar las soluciones existentes

Objetivo Específico	Productos	Actividades	% de Logro	Publicación de Referencia
OE 1: Realizar un análisis científico bibliográfico de los problemas de optimización más relevantes en el procesamiento de señales relacionados con la norma de Frobenius.	Una monografía con los conceptos matemáticos y computacionales asociados a cada uno de los problemas estudiados.	**A 1.1:** Búsqueda bibliográfica en revista científicas. **A 1.2:** Clasificación de los problemas por tipo de soluciones: analíticos y aproximados. **A 1.3:** Clasificación de los problemas por aplicación: en imágenes, en sonidos, en datos numéricos, entre otros. **A 1.4:** Análisis de los conceptos matemáticos y computacionales de cada problema.	100 %	Anexos 1, 2, 3, 5.
OE 2: Establecer similitudes y diferencias entre los problemas de optimización más relevantes en el procesamiento de señales relacionados con la norma de Frobenius para el desarrollo del marco-matemático.	Un modelo matemático que involucre a todos los problemas de optimización más relevantes en el procesamiento de señales relacionados con la norma de Frobenius.	**A 2.1:** Análisis de los conceptos matemáticos que relacionan a cada uno de los problemas y sus respectivas soluciones. **A 2.2:** Búsqueda de conexiones entre cada uno de los problemas estudiados y el modelo presentado en la ecuación (1). **A 2.3:** Creación de un diagrama que relacione cada uno de los problemas estudiados.	100 %	Anexos 13
OE 3: Desarrollar una librería de toolbox de los algoritmos asociados a cada uno de los problemas de optimización más relevantes en el procesamiento de señales relacionados con la norma de Frobenius.	Una librería de toolbox, y su respectivo manual, con cada uno de los algoritmos que dan solución a los problemas estudiados.	**A 3.1:** Implementación de cada uno de los algoritmos que dan solución a los problemas estudiados, utilizando algunos de los softwares de cálculo numérico **A 3.2:** Validación de cada algoritmo implementado usando los ejemplos presentados en la literatura consultada y los datos proporcionados por el Dr, Torokthi.	100%	Anexos 13 y 14

Objetivo Específico	Productos	Actividades	% de Logro	Publicación de Referencia
OE 4: Elaborar un nuevo conjunto de problemas de optimización, basados en el modelo presentado en la ecuación (1), que generalicen y mejoren varios de los problemas de optimización estudiados anteriormente.	Un conjunto de nuevos problemas de optimización con sus respectivos algoritmos, los cuales se adapten al modelo general presentado en la ecuación (1).	**A 4.1:** Análisis matemático y computacional de cada uno de cada uno de los problemas de optimización estudiados en la literatura, para encontrar posibles mejoras, ya sea incrementando la precisión o mejorando rendimiento de los métodos computacionales. **A 4.2:** Elaboración de los nuevos modelos matemáticos basados en el modelo presentado en la ecuación (1). **A 4.3:** Desarrollo de la teoría matemática que fundamente las ventajas de los nuevos modelos matemáticos desarrollados.	100%	Anexos 4, 5, 6, 7, 8, 9, 10, 11, 12
OE 5: Crear una librería de toolbox de los algoritmos asociados a cada uno de los nuevos problemas de optimización desarrollados.	Una librería de toolbox, y su respectivo manual, con cada uno de los algoritmos implementados que dan solución a los nuevos problemas de optimización.	**A 5.1:** Implementación de cada uno de los algoritmos que dan solución a los nuevos problemas de optimización propuestos, utilizando algunos de los softwares de cálculo numérico. **A 5.2:** Validación los algoritmos asociados a cada uno de los nuevos problemas de optimización desarrollados, utilizando los ejemplos presentados en la literatura consultada y los datos proporcionados por el Dr, Torokthi.	100%	Anexos 13 y 14

2. Plan de Difusión

Publicaciones Científicas				
Título	**Estado**	**Indexación**	**Evento/Revista**	**Comité Científico**
Descomposición en valores singulares de una matriz: un repaso por los fundamentos teóricos y sus aplicaciones en procesamiento de imágenes (Ver Anexo 1)	Publicado	Scopus	Revista de Investigación Operacional	Si
A general class of arbitrary order iterative methods for computing generalized inverses (Ver Anexo 2)	Publicado	Scopus / Web of Science (Q1)	Applied Mathematics and Computation	Si
Iterative processes with arbitrary order of convergence for approximating generalized inverses (Ver Anexo 3)	Publicado	-	Proceedings of the XXVI Congreso de Ecuaciones Diferenciales y Aplicaciones. XVI Congreso de Matemática Aplicada	Si
A Fast Algorithm for Image Deconvolution Based on a Rank Constrained Inverse Matrix Approximation Problem (Ver Anexo 4)	Publicado	Scopus	Proceedings of Sixth International Congress on Information and Communication Technology	Si
Effective implementation to reduce the execution time of a low-rank matrix approximation problem (Ver Anexo 5)	Publicado	Scopus / Web of Science (Q1)	Journal of Computational and Applied Mathematics	Si
Error analysis of the generalized low-rank matrix approximation (Ver Anexo 6)	Publicado	Scopus / Web of Science (Q3)	The Electronic Journal of Linear Algebra	Si
Data Compression: Multi-Term Approach (Ver Anexo 7)	Publicado	-	arXiv	No
Second Degree Model for Multi-Compression and Recovery of Distributed Signals (Ver Anexo 8)	Publicado	-	arXiv	No
Extended Principal Component Analysis (Ver Anexo 9)	Publicado	-	arXiv	No
Matrix approximation by a sum of matrix products (Ver Anexo 10)	Sometido a Publicación	Scopus	International Journal of Applied and Computational Mathematics	Si

Título	Estado	Indexación	Evento/Revista	Comité Científico
Fast Multiple Rank-Constrained Matrix Approximation (Ver Anexo 11)	Sometido a Publicación	Scopus	SeMA Journal	Si
Multinomial Karhunen-Loeve Transform (Ver Anexo 12)	Sometido a Publicación	Scopus / Web of Science (Q1)	IEEE Transactions on Signal Processing	Si
FroImPro: A Mathematical-Computational Framework Based on Frobenius Norm for a Set of Image Processing Problems (Anexo 13 y 14)	Sometido a Publicación	Scopus	Ingeniería e Investigación	Si

3. Participación Estudiantil

Este proyecto no contó con participación estudiantil.

4. Ejecución Presupuestaria

Presupuesto 2019			
Rubro	**Monto Aprobado**	**Monto Ejecutado**	**Monto Pendiente**
Útiles y materiales de oficina y cómputo	₡ 50 000	₡ 0	₡ 50 000
Productos de papel, cartón e impresos	₡ 750 000	₡ 473 180,63	₡ 276 819,37
Total	₡ 800 000	₡ 473 180,63	₡ 326 819,37

Presupuesto 2020			
Rubro	**Monto Aprobado**	**Monto Ejecutado**	**Monto Pendiente**
Útiles y materiales de oficina y cómputo	₡ 50 000	₡ 0	₡ 50 000
Productos de papel, cartón e impresos	₡ 750 000	₡ 549 000	₡ 201 000
Total	₡ 800 000	₡ 549 000	₡ 251 000

Las razones por las cuales no se ejecutaron al 100% del presupuesto se explica en la sección de Limitaciones y problemas encontrados.

5. Limitaciones o Problemas Encontrados

- En el I semestre del 2019 (del 29 de enero del 2019 al 14 de marzo del 2019) se tuvo la limitante que el investigador Jeffry Chavarría Molina fue intervenido quirúrgicamente, y eso generó una incapacidad de 7 semanas, para la recuperación posquirúrgica. En ese periodo los investigadores Juan Pablo Soto y Juan José Fallas asumieron las labores correspondientes al proyecto. Posteriormente, el investigador Juan José Fallas también tuvo un periodo de incapacidad de 22 días (una primera incapacidad del 29 mayo al 12 de junio 2019, la cual luego se extendió por una semana más), lo cual también fue consecuencia de una cirugía. En ese periodo de incapacidad los investigadores Jeffry Chavarría y Juan Pablo Soto asumieron las labores del proyecto. A pesar de ambas limitaciones, en el año 2019 se logró avanzar satisfactoriamente con los objetivos del proyecto. Como se detalla en el cuadro de avance, los primeros tres objetivos específicos fueron trabajados en un alto porcentaje. El grupo de investigadores estima que en el año 2019 se logró avanzar un 85% de los objetivos asignados al año 2019, a pesar de los periodos de ausencia de alguno de los responsables, por motivos de fuerza mayor.

- Para el año 2020, específicamente durante el I semestre, las complicaciones se incrementaron por el recargo de labores generadas como consecuencia del COVID-19. Los tres investigadores, además del tiempo destinado para investigación, atendemos 3 cursos como parte de nuestra jornada laboral. Siendo, en algunos casos, cursos únicos que implican una planificación completa de todas las actividades inherentes al diseño de los cursos (por ejemplo, el diseño de guías semanales de trabajo asincrónico), preparación de clases, diseño y aplicación de evaluaciones virtuales, etc. Adicionalmente, durante varias semanas (en el mes de abril) los profesores atendimos jornadas de capacitación (sobre plataformas como Schoology o el TEC-Digital, para la aplicación de pruebas en líneas), dadas las circunstancias que nos obligaron a modificar radicalmente la forma en que atendíamos los tres cursos. Por lo tanto, el avance en el proyecto desde el mes de marzo del 2020 hasta julio del 2020 fue poco.

- Después de julio del 2020, cuando ya existió más claridad con el manejo de las lecciones en modalidad remota, se pudo avanzar más en el proyecto. La ampliación que nos brindó la VIE por 6 meses más (Enero-Junio 2020) permitió terminar el proyecto, sin embargo, tuvimos el problema del desarrollo de los artículos científicos finales. Por esa razón, se solicitó entregar el informe final de este proyecto para diciembre del 2021.

- En diciembre del 2021, el coordinador del proyecto, Pablo Soto, fungió como director interino de la escuela de Matemática. Eso impidió que se finalizará el informe final en diciembre del 2021, debido a la falta de unos detalles del último artículo científico que se estaba desarrollando (Ver Anexos 13 y 14). Dicho informe y sus respectivas publicaciones fueron finalizando en el mes de Enero del 2022.

- Se presentó problemas en la ejecución total del presupuesto, ya que el coordinador del proyecto, Pablo Soto, no tenía manejo claro del manejo de dicho dinero. Pensaba que el dinero por los 2 años se podía usar en cualquier comento de esos 2 años. Entonces, con el rubro relacionado con útiles y materiales de oficina y cómputo, pensaba que se podían gastar todos los 100 mil colones en el 2020, lo cual no fue así. Por otra parte, en el 2020 no se gastaron los 50 mil colones asignados a ese rubro debido a la pandemia. Ese dinero se iba a utilizar en 2 pizarras, pero al estar en modalidad remota, entonces no se pudo realizar dicha compra.

- Con respecto a los productos de papel, cartón e impresos, se hizo la compra de todos los libros y artículos posibles relacionados con el tema de investigación, el cual los miembros del proyecto utilizaron. Ya no había necesidad de gastar en más libros, por eso quedó un dinero sobrante.

6. Anexos

A continuación, se indica el detalle de los anexos, los cuales sustituyen al informe general denominado Documento 1 (Ver más detalles en la sección Plan de Difusión).:

- **Anexo 1:** Descomposición en valores singulares de una matriz: un repaso por los fundamentos teóricos y sus aplicaciones en procesamiento de imágenes
- **Anexo 2:** A general class of arbitrary order iterative methods for computing generalized inverses
- **Anexo 3:** Iterative processes with arbitrary order of convergence for approximating generalized inverses
- **Anexo 4:** A Fast Algorithm for Image Deconvolution Based on a Rank Constrained Inverse Matrix Approximation Problem
- **Anexo 5:** Effective implementation to reduce the execution time of a low-rank matrix approximation problem
- **Anexo 6:** Error analysis of the generalized low-rank matrix approximation
- **Anexo 7:** Data Compression: Multi-Term Approach
- **Anexo 8:** Second Degree Model for Multi-Compression and Recovery of Distributed Signals
- **Anexo 9:** Extended Principal Component Analysis
- **Anexo 10:** Matrix approximation by a sum of matrix products
- **Anexo 11:** Fast Multiple Rank-Constrained Matrix Approximation
- **Anexo 12:** Multinomial Karhunen-Loeve Transform
- **Anexo 13:** FroImPro: A Mathematical-Computational Framework Based on Frobenius Norm for a Set of Image Processing Problems
- **Anexo 14:** Toolbox de aplicaciones de la norma de Frobenius en procesamiento de imágenes.

Anexo 1

DESCOMPOSICIÓN EN VALORES SINGULARES DE UNA MATRIZ: UN REPASO POR LOS FUNDAMENTOS TEÓRICOS Y SUS APLICACIONES EN EL PROCESAMIENTO DE IMÁGENES

Juan José Fallas-Monge,* Jeffry Chavarría-Molina, Pablo Soto-Quiros
Instituto Tecnológico de Costa Rica, Costa Rica.

ABSTRACT

This review paper systematizes a detailed, didactical and own construction of the singular value decomposition (SVD). The theory is complemented with relevant applications to image processing. Also, the algorithms of those applications and numerical examples are shown. The way how the document is organized allows it to be useful for students and researchers with interest in the SVD and its applications to image processing.

KEYWORDS: SVD, singular values, image processing, Linear Algebra.

MSC: 15-01, 15A18, 15A23

RESUMEN

El presente artículo, de tipo *review*, sistematiza una construcción propia, detallada y didáctica de la descomposición en valores singulares de una matriz (SVD, por su siglas en inglés). A su vez, complementa el desarrollo teórico con aplicaciones relevantes de la SVD en el procesamiento de imágenes, junto con los algoritmos respectivos y ejemplos numéricos. La forma clara en que se estructura el documento le permite ser útil para aquellos estudiantes e investigadores dedicados al estudio de la SVD y sus aplicaciones en procesamiento de imágenes.

PALABRAS CLAVE: SVD, valores singulares, procesamiento de imágenes, Álgebra Lineal.

1. INTRODUCCIÓN

La descomposición en valores singulares de una matriz (SVD, por sus siglas en inglés) es un tipo de factorización que generaliza para cualquier matriz rectangular el concepto de valores propios, mediante una extensión de la descomposición polar ([16, 13]). Esta estrategia es similar a la diagonalización de matrices, la descomposición QR o la descomposición LU, sin embargo, tiene la ventaja que existe para

*jfallas@itcr.ac.cr, jchavarria@tec.ac.cr, jusoto@tec.ac.cr

cualquier matriz, sin imponer limitaciones sobre su dimensión. Como se mostrará posteriormente, la SVD es ampliamente utilizada en diversas aplicaciones, algunas de las cuales se abordarán con detalle en este documento. Parte de su fortaleza recae en que los valores singulares permiten organizar, en cierto sentido, la información almacenada en una matriz. Los valores singulares de mayor magnitud se asocian a subespacios propios que condensan la información más significativa de los datos. Esto permite reducir la dimensionalidad de problemas concretos como la compresión de imágenes, en donde es posible controlar la calidad de la compresión, mediante la magnitud de dichos valores.

El inicio del estudio de la SVD fue en 1873 mediante los trabajos desarrollados por Eugenio Beltrami ([4]) y Camille Jordan ([19]). Sobre esto, el matemático G. W. Stewart resalta que Beltrami y Jordan son los progenitores de la SVD, ya que Beltrami fue el primero en llevar a cabo la publicación de dicha descomposición y Jordan por la completitud y la elegancia de la representación ([25]). Curiosamente, el origen de la SVD precedió al uso de las matrices, ya que los resultados se desarrollaron en términos de determinantes, formas bilineales y formas cuadráticas ([25, 22]).

La SVD tiene diversas aplicaciones en Matemática tales como calcular la inversa de Moore-Penrose, determinar el rango de una matriz, calcular el espacio nulo y rango de una transformación lineal ([15, 16, 13]) e, inclusive, en Estadística ha sido utilizada como una generalización del método de Análisis de Componentes Principales ([18]). Sin embargo, en los últimos años este concepto también se ha extendido al campo de la ingeniería. La búsqueda y ordenamiento de documentos relevantes dentro de una determinada colección (como lo hacen los motores de búsqueda en Internet), así como en el desarrollo de sistemas de control de múltiple entrada y múltiple salida, con el fin de mejorar las ondas de transmisión y recepción en antenas para dispositivos inalámbricos, son ejemplos de ello ([27, 28]). En el procesamiento de imágenes también ha ido en aumento y esto es relevante porque afecta positivamente áreas como la medicina, telecomunicaciones, control de procesos industriales y al entretenimiento ([23]). Algunos ejemplos concretos son la compresión de imágenes digitales a través del rango matricial ([22, 8]), el modelado de fondo de videos ([30]), la eliminación de ruido de una imagen ([1, 30]) y el reconocimiento facial ([29]).

El presente artículo, de tipo *review*, muestra una revisión meticulosa y autónoma de la descomposición en valores singulares de una matriz, con particular énfasis en algunas aplicaciones en el procesamiento de imágenes. La construcción teórica realizada es propia de los autores, no en el sentido de la originalidad de los resultados y teoremas que se usan (todos ellos son ampliamente conocidos en Álgebra Lineal), si no en la forma didáctica en que se organizan y detallan en el documento para que cualquier lector con conocimientos básicos de Álgebra Lineal pueda asimilar la deducción, sin necesidad de estar consultando recurrentemente otros artículos o libros. Dichas explicaciones se complementan con ejemplos numéricos que ayudan al lector para una mejor comprensión de los resultados. Muchas otras publicaciones relevantes tratan la SVD (ver [25] y [22], por ejemplo), sin embargo, se enfocan en el teorema en sí y su demostración, dejando de lado los conceptos y resultados periféricos necesarios para el proceso. Por ello, el artículo da una visión teórica más completa y global de la SVD, que potencia al lector a entender cómo esta descomposición funciona en aplicaciones concretas.

Por otra parte, el ejemplo clásico en la literatura sobre la SVD en el área del procesamiento de imágenes

consiste en la compresión de imágenes, por lo que el presente artículo muestra detalladamente otras aplicaciones actuales y que no son frecuentemente referenciadas al mostrar ejemplos aplicados de esta descomposición. Por ello, se mostrará cómo la SVD puede ser utilizada en el modelado de fondo de imágenes para la detección de movimiento en videos, en la eliminación de ruido de una imagen y en el reconocimiento facial.

Cabe aclarar que los autores del presente documento no elaboraron ni propusieron los algoritmos y aplicaciones mencionados en la Sección 4. Los autores solo se dedicaron a realizar la reproducción de ellos, y realizar una presentación y organización original de estos métodos.

Finalmente, el artículo está organizado de la siguiente manera. En la sección 2 se presentan los conceptos previos necesarios para formalizar la SVD. En la sección 3 se formaliza y ejemplifica la descomposición. En la sección 4 se sintetiza un conjunto de aplicaciones de la SVD en al área del procesamiento de imágenes. Finalmente, en la sección 5 se presentan las conclusiones.

2. PRELIMINARES

En esta sección se resumen los conceptos y resultados necesarios para formalizar la SVD. Algunos de ellos se justificarán en el texto, otros son bastante conocidos y pueden ser fácilmente encontrados en la literatura.

Dados $m, n \in \mathbb{N}$, se denotará con $\mathrm{Mat}(\mathbb{C}, m, n)$ el conjunto de las matrices de entradas complejas de m filas y n columnas. Por su parte, $\mathrm{Mat}(\mathbb{C}, n)$ denota el conjunto de las matrices cuadradas de orden n de entradas complejas. Se escribirá $A_{m\times n}$ o A_n, si $A \in \mathrm{Mat}(\mathbb{C}, m, n)$ o $A \in \mathrm{Mat}(\mathbb{C}, n)$, respectivamente. A la matriz $\Sigma \in \mathrm{Mat}(\mathbb{C}, m, n)$ tal que $\sigma_{ij} = 0$, cuando $i \neq j$, se le denomina *matriz diagonal generalizada*. En el caso que Σ sea cuadrada, simplemente se le llama *matriz diagonal*.

Si $U \in \mathrm{Mat}(\mathbb{C}, m, n)$, la *transpuesta conjugada* de U se denotará U^*, y satisface que $u^*_{ij} = \overline{u_{ji}}$, $\forall i, j$. Si U solo tiene entradas reales, entonces $U^* = U^{\mathbf{T}}$. En caso que $U \in \mathrm{Mat}(\mathbb{C}, n)$, se le llama *hermitiana* o *autoadjunta* si $U = U^*$. Además, U es *unitaria* si $U^* \cdot U = I_n = U \cdot U^*$, esto es, si U^{-1} existe y $U^{-1} = U^*$. Es claro que las matrices cuadradas AA^* y A^*A son hermitianas, para toda $A \in \mathrm{Mat}(\mathbb{C}, m, n)$.

Dada $A \in \mathrm{Mat}(\mathbb{C}, m, n)$, una *representación de A por bloques* con p bloques-fila y q bloques-columna tiene la forma:

$$A = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1q} \\ \vdots & & & \vdots \\ A_{p1} & A_{p2} & \dots & A_{pq} \end{pmatrix},$$

tal que para i fijo $(1 \leq i \leq p)$ todas las submatrices A_{ij} $(1 \leq j \leq q)$ tienen que tener la misma cantidad de filas (no así con el número de columnas). Similarmente, para j fijo todas las submatrices A_{ij} tienen que tener la misma cantidad de columnas (no así con el número de filas). Evidentemente, la representación por bloques de una matriz no es única. Si las matrices $A \in \mathrm{Mat}(\mathbb{C}, m, n)$ y $B \in \mathrm{Mat}(\mathbb{C}, n, s)$ se organizan por bloques de manera que A tiene p bloques-fila y q bloques-columna,

y B tiene q bloques-fila y t bloques-columna, entonces el producto AB está dado por $(AB)_{ij} = \sum_{k=1}^{q} A_{ik}B_{kj},\ 1 \leq i \leq p,\ 1 \leq j \leq t$, siempre que el producto de bloques $A_{ik}B_{kj}$ esté bien definido para todo k.

Si $A \in \mathrm{Mat}(\mathbb{C}, m, n)$, el *rango columna* de A es el número máximo de columnas que son linealmente independientes. Similarmente, el *rango fila* de A es el número máximo de filas que son linealmente independientes. En [20] se demuestra que dichos valores son iguales y se le denomina el *rango de A*. Es común utilizar las notaciones $\mathrm{r}(A)$ y $\mathrm{rg}(A)$ para denotar a este valor. Es claro que $0 \leq \mathrm{r}(A) \leq \mathrm{mín}\{m, n\}$, y en caso que $\mathrm{r}(A) = \mathrm{mín}\{m, n\}$, entonces se dice que A es de *rango completo*. Algunas de las propiedades bastante conocidas sobre el rango de una matriz son:

- Si A es una matriz cuadrada, A es invertible si y solo si es de rango completo.
- Si $B \in \mathrm{Mat}(\mathbb{C}, m)$ es una matriz no singular, $\mathrm{r}(BA) = \mathrm{r}(A)$, para todo $A \in \mathrm{Mat}(\mathbb{C}, m, n)$.
- Si $\Sigma \in \mathrm{Mat}(\mathbb{C}, m, n)$ es una matriz diagonal generalizada, entonces su rango corresponde al número de entradas no nulas.

Para $A \in \mathrm{Mat}(\mathbb{C}, n)$, a un vector $v \in \mathbb{C}^n$, $v \neq \mathbf{0}$, se le llama *vector propio* de A si existe un escalar λ (llamado *valor propio*) tal que $Av = \lambda v$ o, equivalentemente, $(\lambda I_n - A)\, v = \mathbf{0}$. Esta definición implica analizar la ecuación matricial $(\lambda I_n - A)\, X = \mathbf{0}$, que tendrá soluciones no triviales si y solo si $\mathrm{r}(\lambda I_n - A) < n$. Por ende, λ es valor propio de A si y solo si $|\lambda I_n - A| = 0$. A la expresión $P(\lambda) = |\lambda I_n - A|$ se le denomina *polinomio característico* de A y a $|\lambda I_n - A| = 0$ la *ecuación característica*. Los valores propios de A son las raíces de su polinomio característico.

En caso que exista una matriz Q no singular tal que $A = Q\Sigma Q^{-1}$ (o, equivalentemente, $Q^{-1}AQ = \Sigma$), donde Σ es una matriz diagonal, entonces se dice que A es *diagonalizable*. Esto es, A es diagonalizable si es semejante a una matriz diagonal. En general, una condición necesaria y suficiente para que una matriz $A \in \mathrm{Mat}(\mathbb{C}, n)$ sea diagonalizable es que su polinomio característico tenga solo raíces reales y tal que para cada valor propio λ, de multiplicidad k, $k \geq 2$, deben existir k vectores propios linealmente independientes. Si esto sucede, entonces las columnas de Q están formadas por tales vectores propios y Σ contiene en su diagonal a los valores propios de A, ordenados de manera respectiva como los vectores propios correspondientes fueron colocados como columnas de Q. La diagonalización de una matriz cuadrada, en caso que sea posible, da una factorización de ella y esto es útil, por ejemplo, para la rotación de secciones cónicas y para el cálculo de A^n, $n \in \mathbb{N}$. No obstante, la diagonalización de matrices tiene tres desventajas significativas:

- La matriz tiene que ser cuadrada. En muchas aplicaciones la matriz de datos no es cuadrada, y por ende la diagonalización deja de ser una herramienta útil.
- Los vectores propios de A usualmente no son ortogonales. Para el caso que A sea simétrica y de entradas reales siempre es posible diagonalizarla ortogonalmente (u ortonormalmente, en caso que se necesiten vectores propios unitarios), sin embargo, para cada valor propio de A de multiplicidad k, $k \geq 2$, es necesario seleccionar k vectores propios ortogonales del subespacio

propio respectivo. Esta selección no siempre es inmediata, y puede que se tengan que realizar cálculo adicionales, como el proceso de ortogonalización de Gram-Schmidt.

- Usualmente no hay "suficientes" vectores propios. Esto pasa en caso que el subespacio propio asociado a un valor propio de multiplicidad k, $k \geq 2$, tenga un dimensión menor que k. Por ende, no es posible construir todas las columnas de la matriz Q que define la diagonalización, con la característica de ser linealmente independientes.

Como se verá luego, la descomposición en valores singulares de una matriz es una alternativa más eficiente a la diagonalización de matrices, dado que evita las tres limitaciones citados arriba.

Algunos de los teoremas que permiten justificar ciertas características de la SVD se demuestran utilizando las propiedades de un producto interno (conocido también como *producto escalar*). Por ello es necesario recordar que si V es un espacio vectorial sobre $\mathbb{C}$, un *producto interno* en V es una función $\langle \cdot, \cdot \rangle : V \times V \to F$ que satisface para todo $u, v, w \in V$ y para todo $\alpha \in \mathbb{C}$ lo siguiente: $\langle u, v \rangle = \overline{\langle v, u \rangle}$, $\langle \alpha u, v \rangle = \alpha \langle u, v \rangle$, $\langle u + v, w \rangle = \langle u, w \rangle + \langle v, w \rangle$, $\langle u, u \rangle \geq 0$ y $\langle u, u \rangle = 0 \iff u = \mathbf{0}$. De la definición se deduce fácilmente que: $\langle x, y + z \rangle = \langle x, y \rangle + \langle x, z \rangle$ y $\langle x, \alpha y \rangle = \overline{\alpha} \langle x, y \rangle$. La forma general de un producto interno en $\mathbb{C}^n$ se conoce como la forma hermitiana y es tal que $\langle u, v \rangle = v^* M u = \overline{u^* M v}$, para todo $u, v \in \mathbb{C}^n$, donde M es cualquier matriz hermitiana definida positiva[1] y v^* es el conjugado transpuesto de v. Es común seleccionar $M = I_n$ y, por ende, usualmente se esribe: $\langle u, v \rangle = v^* u$.

Si $S = \{u_1, \ldots, u_n\}$ es un *conjunto ortogonal* de vectores no nulos de un espacio vectorial V sobre $\mathbb{C}$ de dimensión n, entonces para $i \in \{1, \ldots, n\}$ se tiene que:

$$\mathbf{0} = \sum_{k=1}^{n} c_k \cdot u_k \Rightarrow u_i^* \cdot \mathbf{0} = c_i \cdot (u_i^* \cdot u_i) \Rightarrow 0 = c_i \cdot \langle u_i, u_i \rangle \Rightarrow c_i = 0$$

Por lo tanto, todo subconjunto ortogonal finito de un espacio vectorial de dimensión finita es linealmente independiente. De hecho, como la cantidad de vectores no nulos linealmente independientes en S es n, que coincide con la dimensión de V, entonces S es una base de dicho espacio vectorial, denominada *base ortogonal* de V y si, además, los vectores de S son unitarios, entonces S es una *base ortonormal* de V.

Se concluye esta sección con la prueba de cuatro resultados fundamentales para establecer la SVD de una matriz. En resumen:

- Todo valor propio de una matriz hermitiana es real.
- Si $A \in \text{Mat}(\mathbb{C}, m, n)$, los valores propios de las matrices AA^* y A^*A son reales no negativos.
- Los vectores propios asociados a valores propios distintos de una matriz hermitiana generan subespacios propios ortogonales.
- Las columnas de una matriz unitaria de orden n forman una base ortonormal de $\mathbb{C}^n$.

[1] La matriz M hermitiana se dice definida positiva si $u^* M u > 0$, para todo $u \in \mathbb{C}^n$ no nulo. Otra forma equivalente de definirla es que todos los autovalores (valores propios) de M sean positivos.

Teorema 1 *(Valor propio de una matriz hermitiana)*
Sea $U \in \mathrm{Mat}(\mathbb{C}, n)$ hermitiana. Si λ es un valor propio de U, entonces $\lambda \in \mathbb{R}$.

Prueba: Sea $v \in \mathbb{C}^n$ un vector propio de U ($v \neq \mathbf{0}$, por definición) asociado a λ. Note que:

$$\begin{aligned} \overline{\lambda}\ \langle v, v\rangle &= \langle v, \lambda v\rangle\ =\ \langle v, Uv\rangle\ = (Uv)^* v = v^* U^* v \\ &= v^* U v =\ \langle Uv, v\rangle\ =\ \langle \lambda v, v\rangle\ = \lambda\ \langle v, v\rangle \end{aligned}$$

Así, $(\overline{\lambda} - \lambda) \cdot \langle v, v\rangle\ = 0$. Como $v \neq \mathbf{0}$, entonces $\overline{\lambda} = \lambda$ y de ahí se tiene el resultado.

♣

Teorema 2
Sea $A \in \mathrm{Mat}(\mathbb{C}, m, n)$. Los valores propios de A^*A y de AA^* son no negativos.

Prueba: Sea λ valor propio de A^*A (el otro caso es similar) y v un vector propio asociado a λ. Como A^*A es hermitiana, en virtud del teorema 1 se sabe que $\lambda \in \mathbb{R}$. Luego,

$$\begin{aligned} \lambda \cdot \langle v, v\rangle &= \langle v, \lambda v\rangle\ =\ \langle v, A^*Av\rangle\ = (A^*Av)^* v = v^* A^* A v \\ &= (Av)^* Av =\ \langle Av, Av\rangle\ \geq 0 \end{aligned}$$

Así, $\lambda \cdot \langle v, v\rangle\ \geq 0$, de donde $\lambda \geq 0$.

♣

Teorema 3
Sea $U \in \mathrm{Mat}(\mathbb{C}, n)$ hermitiana. Si λ_1 y λ_2 son autovalores de U, tales que $\lambda_1 \neq \lambda_2$, y v_1 y v_2 autovectores asociados a λ_1 y λ_2, respectivamente, entonces $v_1 \perp v_2$.

Prueba: En efecto:

$$\begin{aligned} \lambda_1\ \langle v_1, v_2\rangle &= \langle \lambda_1 v_1, v_2\rangle\ =\ \langle Uv_1, v_2\rangle\ = v_2^* U v_1 = v_2^* U^* v_1 \\ &= (Uv_2)^* v_1 =\ \langle v_1, Uv_2\rangle\ =\ \langle v_1, \lambda_2 v_2\rangle\ = \overline{\lambda_2}\ \langle v_1, v_2\rangle \\ &= \lambda_2\ \langle v_1, v_2\rangle \end{aligned}$$

Así, $(\lambda_1 - \lambda_2) \cdot \langle v_1, v_2\rangle\ = 0$, de donde $v_1 \perp v_2$.

♣

Teorema 4
Sea $U \in \mathrm{Mat}(\mathbb{C}, n)$. U es unitaria si y solo si sus vectores columnas forman un conjunto ortonormal con el producto escalar complejo usual.

Prueba: Denote $U = (u_1\ u_2\ \ldots\ u_n)$, donde cada u_i, $1 \leq i \leq n$, corresponde a la i-ésima columna de C. Note que:

$$U^*U = \begin{pmatrix} u_1^* \\ u_2^* \\ \vdots \\ u_n^* \end{pmatrix} (u_1\ u_2\ \ldots\ u_n) = \begin{pmatrix} u_1^*u_1 & u_1^*u_2 & \ldots & u_1^*u_n \\ u_2^*u_1 & u_2^*u_2 & \ldots & u_2^*u_n \\ \vdots & & & \vdots \\ u_n^*u_1 & u_n^*u_2 & \ldots & u_n^*u_n \end{pmatrix}$$

$$= \begin{pmatrix} \langle u_1, u_1\rangle & \langle u_2, u_1\rangle & \ldots & \langle u_n, u_1\rangle \\ \langle u_1, u_2\rangle & \langle u_2, u_2\rangle & \ldots & \langle u_n, u_2\rangle \\ \vdots & & & \vdots \\ \langle u_1, u_n\rangle & \langle u_2, u_n\rangle & \ldots & \langle u_n, u_n\rangle \end{pmatrix}$$

De lo anterior se concluye que: U unitaria $\iff$ $U^*U = I_n$ $\iff$ $\big[\ \langle u_i, u_j\rangle = 0$ para $i \neq j$ y $\langle u_i, u_i\rangle = \|u_i\|^2 = 1$ para $i \in \{1, \ldots, n\}\big]$. Esto finaliza la prueba. ♣

En virtud del teorema 4, las columnas de una matriz unitaria $U \in \mathrm{Mat}(\mathbb{C}, n)$ forman una base ortonormal de $\mathbb{C}^n$ con respecto al producto escalar usual y, además, se deduce que toda matriz unitaria es de rango completo.

3. LA DESCOMPOSICIÓN EN VALORES SINGULARES

En esta sección se procederá a formalizar y ejemplificar la SVD. El teorema 5 establece la existencia de la descomposición para una matriz cualquiera. Para la prueba, suponga sin pérdida de generalidad que $n \leq m$ (adaptaciones mínimas permiten establecer la prueba para el caso que $n \geq m$).

Teorema 5 *(Existencia de la SVD)*
Sean $m, n \in \mathbb{N}$ y $A \in \mathrm{Mat}(\mathbb{C}, m, n)$ de rango r. Existen matrices unitarias $U_{m\times m}$ y $V_{n\times n}$ tales que $A = U\Sigma V^*$, donde $\Sigma_{m\times n}$ es una matriz diagonal generalizada de entradas reales no negativas. Dicha factorización se le llama descomposición en valores singulares y si A solo tiene entradas reales, entonces $A = U\Sigma V^{\mathbf{T}}$.

Prueba: Primero se tiene que $r = \mathrm{r}(A) = \mathrm{r}(A^*A)$ (ver [6] para la prueba de este hecho). Luego, en virtud del teorema 2 los valores propios de A^*A son no negativos. Denote dichos valores propios $\sigma_1^2 \geq \sigma_2^2 \geq \ldots \geq \sigma_r^2 > \sigma_{r+1}^2 = \sigma_{r+1}^2 = \ldots = \sigma_n^2 = 0$ y los respectivos vectores propios $v_1, \ldots, v_n$ *normalizados*. Así,

$$A^*Av_i = \sigma_i^2 v_i, \quad \text{para } i = 1, \ldots, n \tag{3.1}$$

Del teorema (3) se concluye que $\{v_1, \ldots, v_n\}$ es un conjunto ortogonal (en realidad, ortonormal, por la normalidad pedida sobre los v_i). Considere las matrices $V_r = (v_1\ v_2\ \ldots\ v_r)$, $V_{n-r} = (v_{r+1}\ v_{r+2}\ \ldots\ v_n)$ y $\Sigma_r = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$. De (3.1) note que:

$$A^*AV_r = V_r\Sigma_r^2 \tag{3.2}$$

Además, como $\{v_1, \ldots, v_n\}$ es ortonormal, entonces $V_r^* V_r = I$, por lo que

$$A^* A V_r = V_r \Sigma_r^2 \Rightarrow V_r^* A^* A V_r = V_r^* V_r \Sigma_r^2 \Rightarrow V_r^* A^* A V_r = \Sigma_r^2$$

Luego,

$$V_r^* A^* A V_r = \Sigma_r^2 \Rightarrow \Sigma_r^{-1} V_r^* A^* A V_r \Sigma_r^{-1} = I \tag{3.3}$$

Ahora, denote $U_r = A V_r \Sigma_r^{-1}$, de donde

$$U_r^* = \Sigma_r^{-1} V_r^* A^* \tag{3.4}$$

De (3.3), entonces se nota que $U_r^* U_r = I$ y por ende U_r es una matriz unitaria de tamaño $m \times r$. Considere ahora otra matriz unitaria U_{m-r} de tamaño $m \times (m-r)$ tal que U_{m-r} sea ortogonal a U_r. Esto es, que cada vector-columna de U_{m-r} sea ortogonal a todo vector-columna de U_r, de donde $U_{m-r}^* U_r = \mathbf{0}$. La existencia y forma para construir a U_{m-r}, la garantiza el proceso de ortogonalización de Grand-Schmidt. Ahora, defina $U = (U_r \;\; U_{m-r})$ y $V = (V_r \;\; V_{n-r})$. Así,

$$U^* A V = (U_r \;\; U_{m-r})^* A (V_r \;\; V_{n-r}) = \begin{pmatrix} U_r^* \\ U_{m-r}^* \end{pmatrix} A (V_r \;\; V_{n-r}) \tag{3.5}$$

$$= \begin{pmatrix} U_r^* A \\ U_{m-r}^* A \end{pmatrix} (V_r \;\; V_{n-r}) = \begin{pmatrix} U_r^* A V_r & U_r^* A V_{n-r} \\ U_{m-r}^* A V_r & U_{m-r}^* A V_{n-r} \end{pmatrix} \tag{3.6}$$

Como $A v_i = \mathbf{0}$, para $i = r+1, \ldots, n$, entonces $A V_{n-r} = \mathbf{0}$. Además, de las ecuaciones (3.2) y (3.4) se tiene que $U_r^* A V_r = \Sigma_r^{-1} V_r^* A^* A V_r = \Sigma_r^{-1} V_r^* V_r \Sigma_r^2 = \Sigma_r$. Por otra parte, como $U_r = A V_r \Sigma_r^{-1}$, entonces $U_r \Sigma_r = A V_r$, por lo tanto de la construcción de U_{m-r} se obtiene: $U_{m-r}^* A V_r = U_{m-r}^* U_r \Sigma_r = \mathbf{0} \cdot \Sigma_r = \mathbf{0}$. Regresando a la ecuación (3.5) se observa que:

$$U^* A V = \begin{pmatrix} U_r^* A V_r & U_r^* A V_{n-r} \\ U_{m-r}^* A V_r & U_{m-r}^* A V_{n-r} \end{pmatrix} = \begin{pmatrix} \Sigma_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$$

Si a esta última matriz se le denomina Σ, entonces

$$U^* A V = \Sigma \Rightarrow U U^* A V V^* = U \Sigma V^* \Rightarrow A = U \Sigma V^*,$$

donde $U_{m \times m}$ y $V_{n \times n}$ son matrices unitarias[2], tal y como se quería probar.

♣

A partir de la SVD de A, como U y V^* son de rango completo (por ser unitarias y por ende invertibles), entonces $\mathrm{r}(A) = \mathrm{r}(U \Sigma V^*) = \mathrm{r}(\Sigma)$. Así, $\mathrm{r}(A) = r$ es el número de entradas no nulas de la matriz Σ. Por su parte, la matriz A no tiene por qué ser de rango completo.

En la prueba del teorema (5) se observa que los vectores propios en V_{n-r} están asociados a los valores propios nulos de $A^* A$. Por ende, para la construcción de las últimas $n - r$ columnas de V basta

[2]Como $U_r^* U_r = I$, $U_{m-r}^* U_{m-r} = I$ y $U_{m-r}^* U_r = \mathbf{0}$, entonces fácilmente se ve que $U^* U = I$ y, por lo tanto, U es unitaria. Similar pasa con la matriz V.

seleccionar $n-r$ vectores ortonormales del núcleo de[3] A, para ello se resuelve el sistema $Av = \mathbf{0}$. Así, $v_1, \ldots, v_r, v_{r+1}, \ldots, v_n$ forman una base ortonormal para $\mathbb{C}^n$. A estos vectores se les suele llamar *vectores singulares por la derecha*.

Adicionalmente, en la prueba se parte del análisis de los valores y vectores propios de la matriz A^*A. Sin embargo, una construcción similar pudo haberse realizado a partir de la matriz AA^*. Note que:

$$A = U\Sigma V^* \quad \Rightarrow \quad A^* = V\Sigma^* U^* \Rightarrow AA^* = U\Sigma V^* V\Sigma^* U^*$$

$$\Rightarrow \quad AA^* = U\Sigma\Sigma^* U^* \Rightarrow AA^*U = U\Sigma\Sigma^*$$

La igualdad $AA^*U = U\Sigma\Sigma^*$ da una diagonalización de AA^*. En efecto, si se denota u_j a las columnas de U, entonces se tiene que $AA^*u_j = \sigma_j^2 u_j$, para $j = 1, \ldots, m$. Por lo tanto, las columnas de U corresponden a vectores propios de AA^*, asociados a los valores propios σ_j^2 de AA^*, que a su vez corresponden a los valores propios de A^*A. En resumen, los valores propios de las matrices AA^* y A^*A coinciden, y corresponden a los valores singulares de la matriz A. Por lo tanto, para determinar los valores singulares de una matriz A conviene trabajar con A^*A o AA^*, según cuál sea de menor tamaño. Esto con el objetivo de tomar el polinomio característico de menor grado.

Si el proceso se comienza con la matriz AA^*, entonces las últimas $m-r$ columnas de U se toman del núcleo de A^*, esto es, se determinan $m-r$ vectores ortonormales a partir del sistema $A^*u = \mathbf{0}$. De esta manera $u_1, \ldots, u_r, u_{r+1}, \ldots, u_m$ forman una base ortonormal para $\mathbb{C}^m$. A estos vectores se les suele llamar *vectores singulares por la izquierda*. Antes de establecer algunos ejemplos, tome en cuenta que:

- Para construir la matriz V si ya se tiene la matriz U, basta aplicar el hecho que

  $$A = U\Sigma V^* \Rightarrow A^* = V\Sigma^* U^* \Rightarrow A^*U = V\Sigma^*$$

  Así, $A^*u_j = \sigma_j v_j$. Por lo tanto, para los $\sigma_j \neq 0$, $1 \leq j \leq \text{mín}\{m, n\}$, se tiene:

  $$v_j = \frac{1}{\sigma_j} A^* u_j \tag{3.7}$$

  La ortogonalidad de los vectores u_j implica que los vectores v_j construidos en (3.7) sean ortogonales también. En efecto, si $i \neq j$, entonces:

  $$\langle v_j, v_i \rangle = v_i^* v_j = \left(\tfrac{1}{\sigma_i} A^* u_i\right)^* \tfrac{1}{\sigma_j} A^* u_j = \tfrac{1}{\sigma_i \sigma_j} u_i^* A A^* u_j = \tfrac{1}{\sigma_i \sigma_j} u_i^* \sigma_j^2 u_j = \tfrac{\sigma_j}{\sigma_i} \langle u_j, u_i \rangle = 0$$

  Por lo tanto, solamente falta normalizarlos. Si ya se tienen todos los v_j, listo; si no, se busca una base ortonormal del $\text{Ker}(A)$ y se agregan para formar a V.

- Si se tiene la matriz V, entonces la matriz U puede ser construida de la siguiente manera. Como $A = U\Sigma V^*$, entonces $AV = U\Sigma$. Luego, $Av_j = \sigma_j u_j$, donde $1 \leq j \leq \text{mín}\{m, n\}$. Para los $\sigma_j \neq 0$ se obtiene:

  $$u_j = \frac{1}{\sigma_j} A v_j$$

[3]En realidad, se deben tomar del núcleo de A^*A, sin embargo, $\text{Ker}(A) \subset \text{Ker}(A^*A)$. Por ello es suficiente con construir una base ortonormal para el $\text{Ker}(A)$.

Similar a como sucede en el caso anterior, la ortogonalidad de los vectores v_j implica la ortogonalidad de los vectores u_j, los cuales deben ser normalizados. Si ya se tienen todos los u_j, listo; si no, se busca una base ortonormal del $\mathrm{Ker}(A^*)$ y se agregan para formar a U.

3.1. Reducción de la SVD

En muchas aplicaciones es común utilizar una versión simplificada de la SVD. Utilizando las notaciones empleadas en el teorema 5 se tiene que:

$$A = U\Sigma V^* = (U_r \;\; U_{m-r}) \cdot \begin{pmatrix} \Sigma_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \cdot (V_r \;\; V_{n-r})^* \tag{3.8}$$

$$= (U_r\Sigma_r \;\; \mathbf{0}) \cdot \begin{pmatrix} V_r^* \\ \\ V_{n-r}^* \end{pmatrix} = U_r\Sigma_r V_r^* \tag{3.9}$$

En síntesis, dada $A \in \mathrm{Mat}(\mathbb{C}, m, n)$, la *descomposición reducida* en valores singulares de A está dada por $A = U_r\Sigma_r V_r^*$, donde $r = \mathrm{r}(A)$, que corresponde al número de valores singulares no nulos de A, que a su vez son los valores propios no nulos de las matrices AA^* y A^*A. Las matrices U_r, Σ_r y V_r^* son como se definieron previamente. Se advierte que dicha reducción favorece porque utiliza matrices de menor tamaño, sin embargo, se pierden propiedades sobre ellas. Por ejemplo, las matrices U y V en la descomposición completa son unitarias, pero U_r y V_r no necesariamente lo son (ni siquiera tienen que ser cuadradas).

Si se visualiza a U_r como una matriz de 1 bloque-fila y 1 bloque-columna (es decir, un único bloque, la matriz total) y a Σ_r como de 1 bloque-fila y r bloques-columna, entonces:

$$U_r\Sigma_r = \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1r} \\ u_{21} & u_{22} & \dots & u_{2r} \\ \vdots & & \dots & \vdots \\ u_{m1} & u_{m2} & \dots & u_{mr} \end{pmatrix}_{1\times 1} \left(\begin{array}{c|c|c|c} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \dots & \sigma_r \end{array}\right)_{1\times r}$$

$$= \left(\begin{array}{c|c|c|c} \sigma_1 u_{11} & \sigma_2 u_{12} & \dots & \sigma_r u_{1r} \\ \sigma_1 u_{21} & \sigma_2 u_{22} & \dots & \sigma_r u_{2r} \\ \vdots & & & \vdots \\ \sigma_1 u_{m1} & \sigma_2 u_{m2} & \dots & \sigma_r u_{mr} \end{array}\right)_{1\times r} = (\sigma_1 u_1 \mid \sigma_2 u_2 \mid \dots \mid \sigma_r u_r)_{1\times r}$$

Ahora, como $V_r = (v_1 \;\; v_2 \;\; \dots \;\; v_r)$, entonces V_r^* puede ser visualizada como una matriz por bloques de r bloques-fila y 1 bloque-columna. Es decir, $V_r^* = \left(\begin{array}{c} v_1^* \\ \hline v_2^* \\ \hline \vdots \\ \hline v_r^* \end{array}\right)_{r\times 1}$.

Por ende,

$$\begin{aligned} U_r\Sigma_r V_r^* &= (\sigma_1 u_1 \mid \sigma_2 u_2 \mid \ldots \mid \sigma_r u_r)_{1\times r} \cdot \left(\begin{array}{c} v_1^* \\ \hline v_2^* \\ \hline \vdots \\ \hline v_r^* \end{array}\right)_{r\times 1} \\ &= \sigma_1 u_1 v_1^* + \sigma_2 u_2 v_2^* + \ldots + \sigma_r u_r v_r^* \end{aligned}$$

A la igualdad $U_r\Sigma_r V_r^* = \sigma_1 u_1 v_1^* + \sigma_2 u_2 v_2^* + \ldots + \sigma_r u_r v_r^*$ se le conoce como la SVD reducida y es fácilmente aplicable en contextos como la compresión de imágenes, tal y como se mostrará en la siguiente sección.

4. APLICACIONES DE LA SVD

En esta sección se sintetizan algunas de las aplicaciones de la SVD. Los experimentos numéricos desarrollados en esta sección se realizaron en una computadora del Instituto Tecnológico de Costa Rica. La computadora contiene un procesador Intel(R) Core(TM) i7-4712MQ CPU 2.30GHz, memoria 8GB, utilizando el programa de cálculo numérico MATLAB.

4.1. Compresión de imágenes

La combinación lineal

$$U_r\Sigma_r V_r^* = \sigma_1 u_1 v_1^* + \sigma_2 u_2 v_2^* + \ldots + \sigma_\alpha u_\alpha v_\alpha^* + \ldots + \sigma_r u_r v_r^*, \tag{4.1}$$

con $1 \leq \alpha \leq r$, muestra cómo la SVD reducida de una matriz puede ser vista como una técnica de compresión de imágenes. Dado que $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$, entonces dicha combinación agrupa de manera ordenada la información que se extrajo de la matriz de datos (que representa a la imagen tratada). Por lo tanto, la matriz $\sigma_1 u_1 v_1^*$ es la que contiene la información más significativa, en caso que se desee reconstruir la imagen original. Si $\sigma_1 > \sigma_2$, entonces la matriz $\sigma_2 u_2 v_2^*$ es la segunda en contener información de mayor calidad para realizar la reconstrucción, y así sucesivamente. En la Figura 1 se utiliza la imagen boat.512.tiff (ver [26]), que es de tamaño 512×512 pixeles, lo cual genera una matriz[4] A de 262144 entradas. Con respecto a la Figura 1, en (a) se muestra la imagen original y, posteriormente, se muestra la reconstrucción de la imagen usando diferentes truncamientos para la combinación lineal. Por ejemplo, para el caso $\alpha = 10$ se realiza el producto matricial $U_{10}\Sigma_{10}V_{10}^*$. Para efectos de almacenamiento, note que U_{10} es una matriz de $512 \cdot 10 = 5120$ entradas, Σ_{10} puede almacenarse como un vector de 10 entradas y V_{10}^* también tiene $10 \cdot 512 = 5120$ entradas. Es decir, en total 10250 entradas. Evidentemente, la compresión con $\alpha = 10$ genera mucha pérdida de información con respecto a la imagen original. De manera similar, en (f) se muestra la reconstrucción realizada con $\alpha = 100$, la cual es de mucho más calidad. En este caso se estarían almacenando 102500 entradas, que es menos de la mitad del número de entradas que representan a la matriz original. Evidentemente,

[4]Para el lector que no se encuentra familiarizado con la representación computacional de una imagen puede consultar alguna referencia afín. En particular puede ver [14].

entre más alto sea el valor de α (que está acotado por $\mathrm{r}(A)$), mayor calidad tendrá la reconstrucción, pero baja el nivel de comprensión.

A modo de ensayo, las matrices generadas con $\alpha = 10$ y $\alpha = 100$ fueron almacenadas por separado en archivos de texto. Lo mismo se realizó con la matriz A, la cual generó un archivo de tamaño 2.25 MB. La Tabla I muestra la información obtenida. Nótese que para el caso de $\alpha = 100$, los tres archivos almacenados en conjunto tienen un tamaño de 951 KB, que representa aproximadamente el 41 % del tamaño del archivo generado con la matriz A. En este ejercicio la comprensión resultó bastante beneficiosa con $\alpha = 100$. Una buena estrategia para seleccionar un valor adecuado para α es realizar el gráfico de las magnitudes de los valores singulares de A. La Figura 2 muestra, para el ejemplo, el comportamiento de los primeros 100 valores no nulos. De la gráfica se nota que σ_{50} y σ_{100} son pequeños en magnitud, comparados con los primeros valores singulares de A. Por ello, es razonable que haber pasado de $\alpha = 50$ a $\alpha = 100$ no generara una mejoría tan significativa en la reconstrucción de la imagen, como sí sucedió al pasar de $\alpha = 10$ a $\alpha = 50$.

Matriz	U_{10}, diag(S_{10}), V_{10}	U_{100}, diag(S_{100}), V_{100}
Tamaño del archivo	48.4 KB, 111 bytes, 48.2 KB	475 KB, 1011 bytes, 475 KB

Tabla 1: Tamaño de los archivos de textos generados con $\alpha = 10$ y $\alpha = 100$.

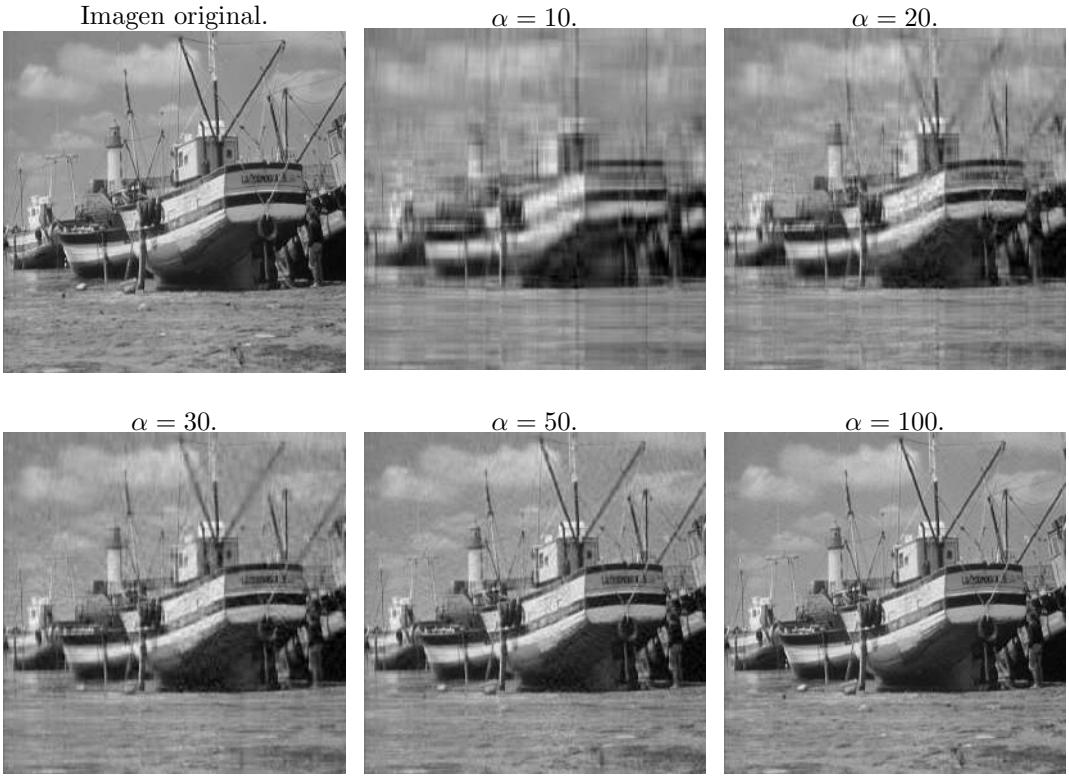


Figura 1: Reconstrucción de una imagen usando diferentes valores para α en la SVD reducida.

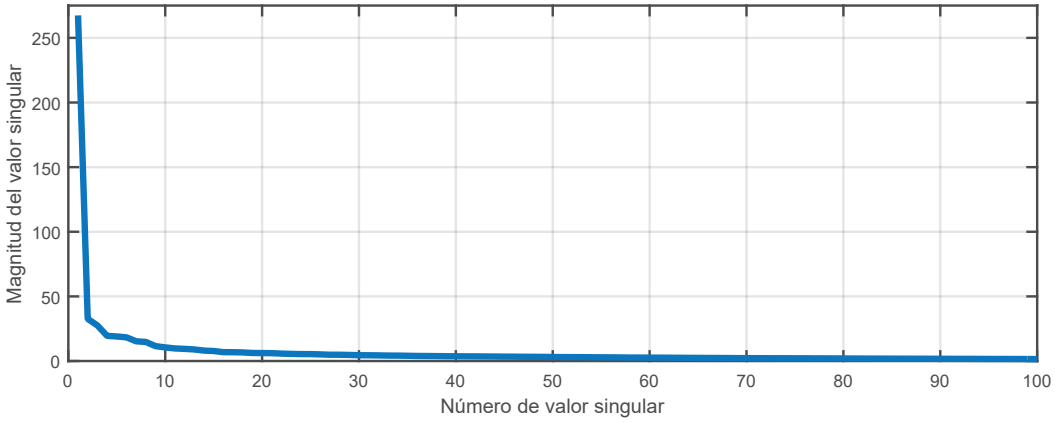


Figura 2: Gráfica de los valores singulares de la matriz A asociada a la imagen boat.512.tiff.

4.2. Cálculo de la pseudoinversa de una matriz

En muchas aplicaciones surgen sistemas de ecuaciones lineales de la forma $AX = B$, en los que la matriz de coeficientes es cuadrada y no singular, los cuales admiten la solución única $X = A^{-1}B$. Sin embargo, también es común que surjan sistemas lineales en los que la matriz A es singular. Para estos casos se puede recurrir a un concepto más general que la matriz inversa (que involucra incluso matrices no cuadradas), que permite generar matrices que se relacionan con las soluciones del sistema.

Dada $A \in \mathrm{Mat}(\mathbb{C}, m, n)$, una *inversa generalizada* de A es cualquier matriz de tamaño $n \times m$, denotada A^-, que satisface que $AA^-A = A$. Si la matriz A es no singular, entonces es claro que A^{-1} satisface la condición anterior. Adicionalmente, si A^- es cualquier otra inversa generalizada para la matriz no singular A, entonces $AA^-A = A$, de donde $A^- = A^{-1}AA^-AA^{-1} = A^{-1}AA^{-1} = A^{-1}$. En resumen, para matrices no singulares la inversa generalizada siempre existe, es única y corresponde a la matriz A^{-1}. En caso que A sea singular, la inversa generalizada siempre existe, pero no necesariamente es única (ver [15] para más detalles).

Este criterio se puede aplicar al sistema de ecuaciones lineales consistente $AX = B$, donde A no tiene que ser cuadrada ni invertible. Note que si A^- es una inversa generalizada de A, entonces $X' = A^-B$ es una solución del sistema $AX = B$. En efecto, si X_0 es cualquier solución de dicho sistema, entonces $AX_0 = B$, y así:

$$AX' = A(A^-B) = AA^-B = AA^-AX_0 = AX_0 = B$$

El concepto general presentado para una inversa generalizada es tan amplio (por la no unicidad), que puede ser poco aplicable. Por ello, existe un caso particular de inversa generalizada denominada *inversa de Moore-Penrose* o *pseudoinversa*, la cual siempre existe para cualquier matriz A y es única (ver [3]). Se define de la siguiente manera: dada una matriz $A \in \mathrm{Mat}(\mathbb{C}, m, n)$, la inversa de Moore-Penrose de A se denota $A^\dagger$ (de tamaño $n \times m$) y es la única matriz que satisface las condiciones $AA^\dagger A = A$ y $A^\dagger AA^\dagger = A^\dagger$ y que, además, las matrices $AA^\dagger$ y $A^\dagger A$ sean hermitianas (esto es, $(AA^\dagger)^* = AA^\dagger$ y $(A^\dagger A)^* = A^\dagger A$).

A partir de la SVD de la matriz $A \in \text{Mat}(\mathbb{C}, m, n)$ (refiérase al teorema 5), fácilmente se puede establecer una fórmula que permite calcular la inversa de Moore-Penrose. En efecto, si $A = U\Sigma V^*$, entonces $A^\dagger = V\Sigma^\dagger U^*$, donde $\Sigma^\dagger = \text{diag}(\sigma_1^{-1}, \ldots, \sigma_r^{-1}, 0, \ldots, 0) = \begin{pmatrix} \Sigma_r^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \in \text{Mat}(\mathbb{C}, m, n)$. Dada la unicidad de la pseudoinversa de Moore-Penrose, basta verificar que dicha matriz satisface la definición correspondiente. En efecto, como U y V son unitarias, entonces:

- $AA^\dagger A = U\Sigma V^* V\Sigma^\dagger U^* U\Sigma V^* = U\Sigma\Sigma^\dagger\Sigma V^* = U\Sigma V^* = A$, dado que:

$$\Sigma\Sigma^\dagger\Sigma = \begin{pmatrix} \Sigma_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \Sigma_r^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \Sigma_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} = \begin{pmatrix} I_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \Sigma_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} = \begin{pmatrix} \Sigma_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} = \Sigma$$

  Siguiendo un razonamiento similar se tiene $\Sigma^\dagger\Sigma\Sigma^\dagger = \Sigma^\dagger$ y por ende

$$A^\dagger AA^\dagger = V\Sigma^\dagger U^* U\Sigma V^* V\Sigma^\dagger U^* = V\Sigma^\dagger U^* = A^\dagger$$

- Es claro que $\left(\Sigma\Sigma^\dagger\right)^* = \Sigma\Sigma^\dagger$, por lo tanto:

$$\begin{aligned} (AA^\dagger)^* &= (U\Sigma V^* V\Sigma^\dagger U^*)^* = (U\Sigma\Sigma^\dagger U^*)^* = U\left(\Sigma\Sigma^\dagger\right)^* U^* \\ &= U\Sigma\Sigma^\dagger U^* = U\Sigma V^* V\Sigma^\dagger U^* = AA^\dagger \end{aligned}$$

  Análogamente, como $\left(\Sigma^\dagger\Sigma\right)^* = \Sigma^\dagger\Sigma$, entonces se comprueba que $(A^\dagger A)^* = A^\dagger A$.

Con fines ilustrativos, a continuación se muestra el cálculo de $A^\dagger$ para dos de las matrices mostradas en los ejemplos de la sección 3.

$$A = \begin{pmatrix} 0 & 2 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \Rightarrow A^\dagger = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 \end{pmatrix}$$

$$A = \begin{pmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{pmatrix} \Rightarrow A^\dagger = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{3\sqrt{2}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{3\sqrt{2}} & \frac{-1}{\sqrt{3}} \\ 0 & \frac{2\sqrt{2}}{3} & \frac{1}{\sqrt{3}} \end{pmatrix} \begin{pmatrix} \frac{1}{5} & 0 \\ 0 & \frac{1}{3} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} \frac{7}{45} & \frac{2}{45} \\ \frac{2}{45} & \frac{7}{45} \\ \frac{2}{9} & \frac{-2}{9} \end{pmatrix}$$

Es importante aclarar que este método es muy preciso, pero poco eficiente en términos del rendimiento computacional, particularmente cuando se aplica a matrices grandes. Sin embargo, para acelerar el rendimiento en el cálculo de la pseudoinversa existen varios métodos iterativos, algunos de los cuales pueden consultarse en [5], [21] y [2].

Finalmente, el concepto de pseudoinversa se puede utilizar en el procesamiento de imágenes para la eliminación de ruido. Por ejemplo, considere la imagen en escala de grises de Einstein que se muestra en la Figura 3(a), la cual se representa numéricamente como una matriz X de dimensión 685×172. La imagen con ruido de la Figura 3(b) representa a la matriz Y construida como $Y = X + G$, donde G es una matriz aleatoria generada con una distribución normal. Para eliminar el ruido de la imagen se

usa un *filtro*, que corresponde a una matriz F de dimensión 685×685 y dada por $F = XY^{\dagger}$ (consultar [7] para más detalles), la cual se pre-multiplica a la matriz Y. La imagen 3(c) muestra el resultado del producto matricial $FY = XY^{\dagger}Y$.

Imagen original. Imagen con ruido. Imagen procesada.

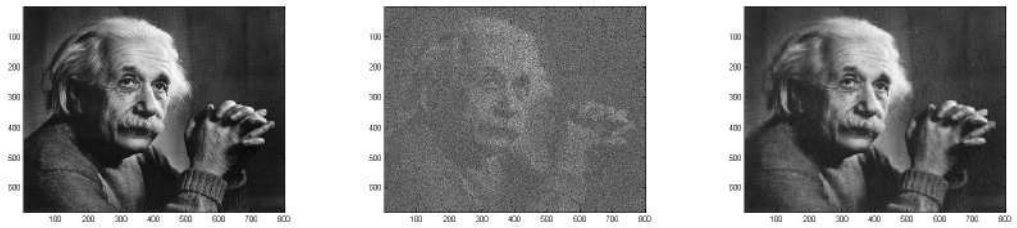


Figura 3: Ejemplo de aplicación de la pseudoinversa en procesamiento de imágenes.

4.3. Go decomposition (GoDec)

Dada una matriz A de rango r, en [9] y [17] se prueba cómo la SVD se puede utilizar para determinar una aproximación L_k de A de *rango reducido*, esto es, que su rango sea a lo sumo k, donde $k < r$. Esta aproximación se relaciona con la SVD reducida que se mostró en la ecuación (4.1). En efecto, si $A = U\Sigma V^*$ y $\mathrm{r}(A) = r$, de (4.1) se tiene que $A = U_r\Sigma_r V_r^* = \sigma_1 u_1 v_1^* + \sigma_2 u_2 v_2^* + \ldots + \sigma_r u_r v_r^*$. Luego, la mejor aproximación de A de rango *a lo sumo* k, está dada por:

$$L_k = \sigma_1 u_1 v_1^* + \sigma_2 u_2 v_2^* + \ldots + \sigma_k u_k v_k^* = U_k \Sigma_k V_k^*, \quad k < r$$

Así, $\min\limits_{\mathrm{r}(Y)\leq k} \|A - Y\|_{fr}^2 = \|A - L_k\|_{fr}^2$, donde $\|\cdot\|_{fr}$ denota a la norma de Frobenius[5]. Este resultado se conoce como el teorema de Eckart-Young y, además, la matriz L_k es única si y solo si $\sigma_k > \sigma_{k+1}$ [11].

El algoritmo iterativo *GoDec*, presentado en [30], emplea la SVD para determinar dos matrices L y S tales que $A \approx S + L$, donde L es de rango reducido y S es una matriz dispersa o rala (la mayor parte de sus entradas son ceros). Así, este algoritmo da una buena aproximación a la solución del problema:

$$\min_{\mathrm{r}(L)\leq k;\ \mathrm{card}(S)\leq c_0} \|A - L - S\|_{fr}^2,$$

donde card(S) denota la cantidad mínima de elementos no nulos en S. Esto es, se busca determinar una matriz L de rango a lo sumo k, $k < r$, y una matriz dispersa S con cardinalidad a lo sumo c_0 (parámetro de entrada en el algoritmo). Se suele escribir $A = L + S + G$, donde G es una matriz de ruido que modela el error de aproximación. En cada iteración del Algoritmo 1 se definen matrices L_t y S_t tales que $\mathrm{r}(L_t) \leq k$ y $\mathrm{card}(S_t) \leq c_0$. Las sucesiones $\{L_t\}$ y $\{S_t\}$ convergen linealmente a un mínimo local del problema de optimización [30]. La función $\mathcal{P}_{c_0}$ en dicho algoritmo (ver línea 6) recibe una matriz densa A' y retorna otra matriz dispersa de cardinalidad a lo sumo c_0, del mismo tamaño de

[5]Dada $A \in \mathrm{Mat}(\mathbb{C}, m, n)$, la norma de Frobenius de A se denota $\|A\|_{fr}$ y se define como $\|A\|_{fr} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}(a_{ij})^2}$. Como se puede notar, la norma de Frobenius de una matriz es similar a la norma euclidiana de un vector en $\mathbb{R}^n$.

A', la cual se construye manteniendo de A' las c_0 entradas de mayor magnitud (en valor absoluto) y las demás entradas iguales a cero.

Algoritmo 1 GoDec Clásico

Entrada: k, c_0, $A_{m\times n}$, ε.

Salida: L, S.

1: $L_0 := A$, $S_0 := \mathbf{0}_{m\times n}$, $t := 0$
2: **mientras** Verdadero **hacer**
3: $t := t + 1$.
4: $[U, \Sigma, V^*] = \text{svd}(A - S_{t-1})$.
5: $L_t := U_k \Sigma_k V_k^*$.
6: $S_t = \mathcal{P}_{c_0}(A - L_t)$.
7: $E_t := \|A - L_t - S_t\|_{fr}^2 / \|A\|_{fr}^2$.
8: **si** $|E_t - E_{t-1}| < \varepsilon$ **entonces**
9: **salir.**
10: **fin si**
11: **fin mientras**

Para ilustrar el Algoritmo 1, considere la matriz $A_{5\times 4}$, de rango 4, definida por:

$$A = \begin{pmatrix} 19 & 10 & 8 & 11 \\ -15 & 7 & 4 & -13 \\ -5 & -8 & 17 & 2 \\ 21 & 11 & -6 & 23 \\ 22 & -12 & 9 & 1 \end{pmatrix},$$

junto con los parámetros $k = 2$, $c_0 = 8$, $\varepsilon = 0.0001$, y las matrices $L_0 = A$ y $S_0 = \mathbf{0}_{5\times 4}$. Luego de 8 iteraciones se obtiene:

$$L_8 = \begin{pmatrix} 18.81311 & -1.66568 & 2.11459 & 11.26995 \\ -15.23693 & -2.94393 & 1.085 & -12.65785 \\ 1.19351 & -8.04379 & 5.30723 & -5.81271 \\ 20.94854 & 10.94608 & -5.98738 & 23.07557 \\ 22.04528 & -11.94952 & 8.99313 & 4.98487 \end{pmatrix}$$

y

$$S_8 = \begin{pmatrix} 0 & 11.66568 & 5.88541 & 0 \\ 0 & 9.94393 & 2.915 & 0 \\ -6.19351 & 0 & 11.69277 & 7.81271 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -3.98487 \end{pmatrix},$$

de tal manera que $\frac{\|A-L_8-S_8\|_{fr}^2}{\|A\|_{fr}^2} \approx 8.9948e^{-05}$. La Figura 4 muestra, para este ejemplo, el comportamiento del error a lo largo de las 8 iteraciones.

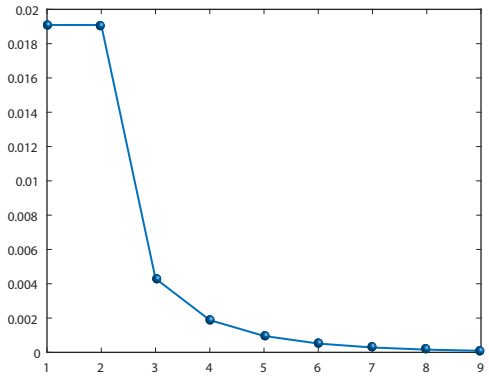


Figura 4: Comportamiento del error al aplicar el Algoritmo *GoDec* a la matriz A del ejemplo.

Expresar una matriz A como $A \approx S + L$, donde L es de rango reducido y S una matriz dispersa, tiene aplicación en el procesamiento de imágenes. En concreto, un video $\mathcal{V}$ (en el que se muestren objetos en movimiento) se puede descomponer en dos videos, de manera que uno de ellos muestre únicamente el fondo del video original (eliminando los objetos en movimiento) y el otro muestre solamente los objetos en movimiento. Para modelar este problema suponga que $\mathcal{V}$ está compuesto por K *frames*, todos del mismo tamaño y tal que el fondo sobre el que se mueven los objetos es invariante. Cada *frame* se modela por una matriz de tamaño $m \times n$, de tal manera que la matriz A sobre la cual itera el algoritmo *GoDec* es de tamaño $mn \times K$. La $i-$ésima columna de A corresponde a la matriz asociada al $i-$ésimo *frame*, la cual es reordenada como una matriz columna de tamaño $mn \times 1$. Así, cada columna de A corresponde a un *frame* de $\mathcal{V}$. Al aplicar el Algoritmo 1 a A se determinan las matrices L y S, de manera que para cada *frame* L almacena la información asociada al fondo y S contiene la información relativa a los objetos en movimiento. Por ende, la $i-$ésima columna de L (reescrita como matriz) corresponde al fondo del $i-$ésimo *frame* (pixeles que se mantienen invariantes), y la $i-$ésima columna de S (reescrita como matriz) corresponde a los pixeles de los objetos en movimiento del $i-$ésimo *frame*.

En la Figura 5 se observan parte de los resultados de la aplicación del algoritmo *GoDec* a un video de 60 *frames* (ver [24]), cada uno de tamaño 540×960, el cual consta de una plazoleta con personas en movimiento. En particular, se muestran los resultados asociados a los *frames* 1, 30 y 60, para visualizar cómo el algoritmo extrajo el fondo e identificó en cada *frame* los objetos en movimiento. Las imágenes (b), (e) y (h) muestran algunas personas como parte del fondo, dado que se mantienen estáticas a lo largo del video, o con movimientos muy leves. En caso que esto último ocurra, los objetos se observan como ligeras sombras en las imágenes que modelan el movimiento en cada uno de los *frames*. En la Figura 6 se resaltan dos ejemplos concretos en donde este fenómeno sucede.

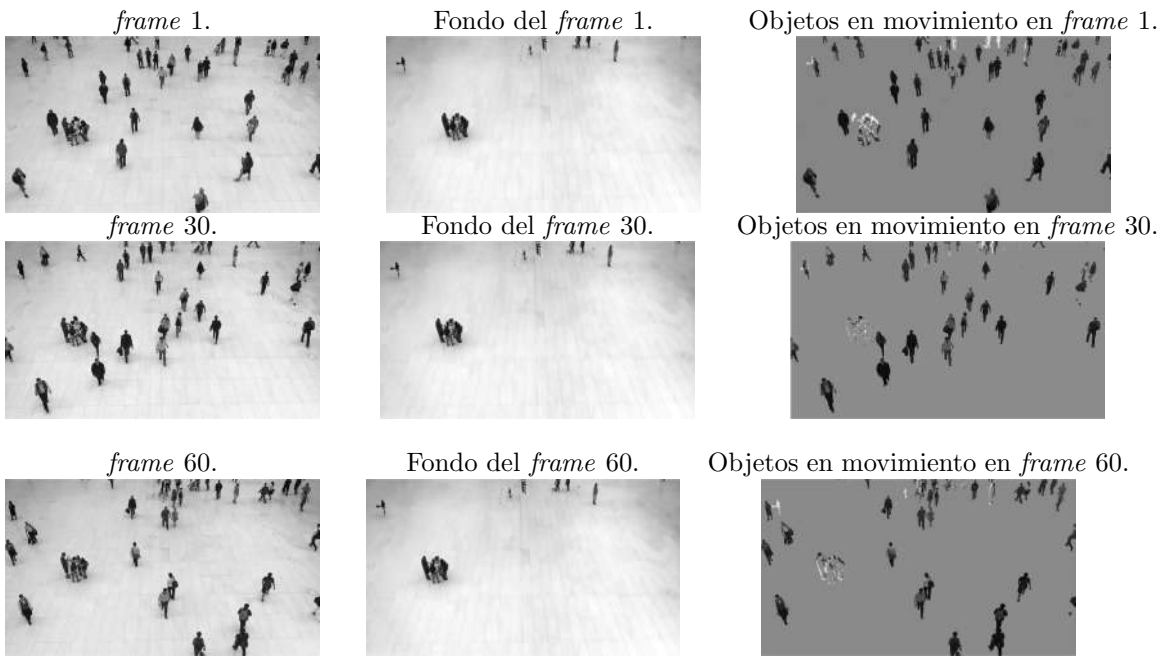


Figura 5: Procesamiento de un vídeo de 60 *frames* con el algoritmo *GoDec*.

Los parámetros utilizados para el procesamiento del video fueron[6]: $k = 2$, $\varepsilon = 10^{-9}$ y $c_0 = \lfloor 0.07 \cdot m \cdot n \cdot K \rfloor = 2177280$. Esto es, c_0 se definió como un porcentaje de la cantidad total de pixeles en el video (un 7 %, para el ejemplo). Tomando en cuenta que c_0 controla la cardinalidad de S, la matriz que modela los objetos en movimiento, entonces es razonable definir a c_0 en función de la cantidad total de pixeles. El valor 0.07 se ajustó empíricamente, luego de varias ejecuciones del algoritmo. Si el video procesado tiene muchos objetos en movimiento (u objetos en movimiento que ocupan mucha área en cada uno de los *frames*), entonces dicha constante de proporcionalidad debe ser mayor. Por su parte, la selección de $k = 2$ se hizo de esa manera (un valor bajo) porque el fondo de $\mathcal{V}$ permanece relativamente invariante a lo largo de los 60 *frames* (igual pudo haberse seleccionado $k = 1$ y los resultados obtenidos son similares). En caso que el fondo varíe a lo largo del video, entonces el valor de k debe incrementarse acorde con el número de variaciones significativas que se detecten. En resumen, tanto para la selección de c_0 como de k, debe mediar un análisis previo del video y, posteriormente, realizar varias ejecuciones del algoritmo, que en conjunto permitan ajustar los parámetros y así lograr la separación requerida entre el fondo y los objetos en movimiento.

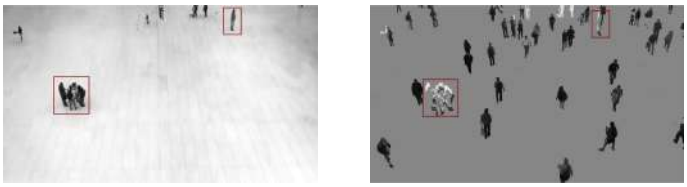


Figura 6: Objetos con poco movimiento en $\mathcal{V}$ que quedan como parte del fondo al aplicar *GoDec*.

[6]La notación $\lfloor \cdot \rfloor$ significa la función parte entera. Así, $\lfloor x \rfloor$ corresponde al mayor número entero menor o igual que x.

4.4. Algoritmo K-SVD

El algoritmo K-SVD, propuesto en [1], utiliza la descomposición en valores singulares para encontrar una representación esparcida de una matriz. Este algoritmo puede ser utilizado para la reconstrucción de imágenes con un cierto porcentaje de pixeles perdidos, tal y como se mostrará al final de la sección.

Formalmente, dada $Y \in \mathbb{R}^{m\times n}$, el algoritmo K-SVD permite aproximar matrices $D \in \mathbb{R}^{m\times t}$ y $X \in \mathbb{R}^{t\times n}$, $1 \leq t \leq \text{mín}\{m, n\}$, que minimicen la expresión $\|Y - DX\|_{fr}^2$, sujeto a las condiciones $\text{Card}(x_i) \leq c_0$, para $i = 1, \ldots, n$, donde $x_i \in \mathbb{R}^n$ representa la i-ésima columna de X y $\text{Card}(x_i)$ *representa la cantidad de entradas de* x_i *diferentes de cero*. En este caso, la constante c_0 es un número entero positivo que se conoce como constante de esparcidad y corresponde a un parámetro de entrada para el algoritmo. En términos prácticos, este problema de optimización busca realizar una representación esparcida de los datos de entrada, determinando un cierto número de patrones elementales (denominados *átomos*) que combinados entre sí permitan realizar una reconstrucción. Estos átomos corresponden a vectores que son organizados como columnas en la matriz D, a la cual se le denomina un *diccionario*. En otras palabras, se entenderá como diccionario a la matriz que permite encontrar la representación dispersa de un conjunto de datos, mediante una combinación lineal de sus columnas.

Este algoritmo corresponde a un método iterativo que alterna entre la escasa codificación de los datos basada en el diccionario actual D y un proceso de actualización de los átomos del diccionario para que haya un mejor ajuste. En este caso, la matriz D es el diccionario de la matriz esparcida X, que permite realizar una aproximación de la matriz Y. La actualización de las columnas de D se combina con una actualización de las representaciones esparcidas de las columnas de X, acelerando así la convergencia. Las matrices D y X se obtienen de manera iterativa, mediante el Algoritmo 2.

El Algoritmo 2, en el paso 3, plantea aproximar cada columna de X, sujeto a una condición de esparcidad. Este problema de optimización ha sido ampliamente estudiado y para su abordaje se utilizan algoritmos de *compressed sensing*. En particular para el ejemplo de aplicación del algoritmo K-SVD, que se mostrará posteriormente, en el paso 3 se utilizó el algoritmo *orthogonal matching pursuit* (OMP), que puede ser consultado en el capítulo 8 de [10]. Por otra parte, en el paso 8 del Algoritmo 2 tome en cuenta que:

- R_I es la matriz definida a partir de R tomando las columnas asociadas a los índices en I. Por ende, el tamaño de R_I es $m \times \text{Card}(I)$.
- d_j es la $j-$ésima columna de D (de dimensión $m \times 1$).
- $x_j(I)$ es el vector construido con las entradas no nulas de x_j (usando para ello los índices almacenados en I). Por lo tanto, los vectores I y $x_j(I)$ tienen tamaño $1 \times \text{Card}(I)$.

Finalmente, en el paso 11 se realiza la actualización de las entradas no nulas de la $j-$ésima columna de X a partir del vector $\sigma_1^2 v_1^T$. Esto es, de manera ordenada (acorde con los índices en I) se van actualizando las entradas no nulas de x_j, con las entradas de $\sigma_1^2 v_1^T$.

Una aplicación del algoritmo K-SVD consiste en la recuperación de pixeles perdidos en una imagen.

Algoritmo 2 Algoritmo K-SVD

Entrada: $Y \in \mathbb{R}^{m\times n}$, $D^{(0)} \in \mathbb{R}^{m\times t}$, $c_0 \in \{1, 2, ..., n\}$, $iter \in \mathbb{N}^*$.

Salida: $D^{(iter)} \in \mathbb{R}^{m\times t}$ y $X^{(iter)} \in \mathbb{R}^{t\times n}$.

1: $D = D^{(0)}$.
2: **para** $k = 1 : iter$ **hacer**
3: $\forall i = 1, ..., n$ determine $x_i = \underset{x}{\text{argmín}} \|y_i - Dx\|_2^2$, tal que $\text{Card}(x) \leq c_0$.
4: Defina $X = (x_1 \ \dots \ x_n)$.
5: Calcule $R = Y - DX$.
6: **para** $j = 1 : t$ **hacer**
7: Calcule I, el vector de índices asociados a las entradas de x_j no nulas.
8: Calcule $\widetilde{R} = R_I + d_j x_j(I)$.
9: Determine $\widetilde{R} = U\Sigma V^T$ (SVD de $\widetilde{R}$).
10: Sustituya $d_j = u_1$, donde u_1 es la primera columna de U.
11: Sustituya $x_j(I) = \sigma_1^2 v_1^T$, donde σ_1 es el primer valor singular de $\widetilde{R}$ y v_1 es la primera columna de V.
12: **fin para**
13: Defina $D^{(k)} = (d_1 \ \dots \ d_t)$.
14: **fin para**

Para el ejemplo que se presentará se utilizó una base de 55 imágenes (cada una de tamaño 112×88 y tomadas de [12]), formada por los rostros de 11 personas. Cada imagen se seccionó en bloques de tamaño 8×8 (obteniendo 154 bloques para cada imagen y 8470 bloques en total) y cada bloque se organizó como un vector columna de tamaño 64×1. Si se denota con $y_1, y_2, ..., y_{8470}$ tales vectores columna, entonces se puede definir la matriz $Y = [y_1, y_2, ..., y_{8470}]$ de tamaño $m \times n = 64 \times 8470$. La Figura 7 muestra el proceso descrito para definir a la matriz Y.

Al aplicar el algoritmo K-SVD a la matriz Y con los parámetros $c_0 = 10$, $iter = 40$, $t = 150$ y $D^{(0)}$ una matriz generada aleatoriamente usando una distribución normal, se obtuvieron matrices $D^{(40)}$ y X^{40} tales que $Y \approx D^{(40)}X^{(40)}$. La matriz $D^{(40)}$ corresponde al diccionario generado con el Algoritmo 2, que permite reconstruir una imagen con pixeles perdidos, que puede haber sido generada con alguna de las 55 imágenes originales o a partir de otra imagen con características similares. En nuestro caso se utilizó una imagen (ver Figura 8) que está en la base de datos, pero diferente a las 55 imágenes usadas para construir la matriz Y.

Para reconstruir la imagen con pixeles perdidos, que también es de tamaño 112×88, se divide la imagen en 154 bloques de tamaño 8×8 y se realiza la reconstrucción por bloques. Cada bloque se transforma en un vector z_j de tamaño 64×1 y para cada $j = 1, 2, ..., 154$ se realizan los siguientes pasos:

- El vector z_j puede tener algunas entradas iguales a 0, las cuales representan los pixeles perdidos. Utilizando esta información se define la *matriz diezmada* asociada a z_j, denotada D_{z_j}, de la

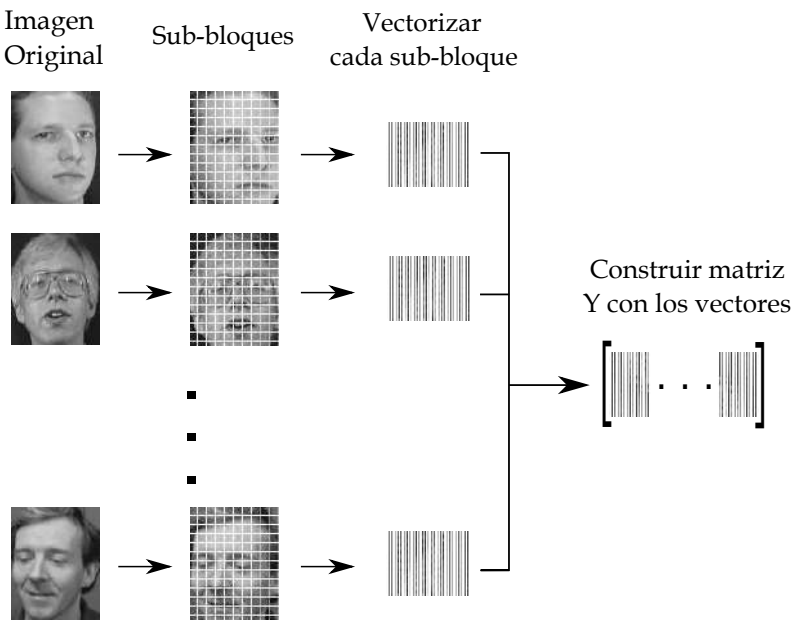


Figura 7: Representación matricial de las imágenes.

siguiente manera:

$$D_{z_j}(i,:) = \begin{cases} D^{(40)}(i,:) & \text{si } z_j(i) \neq 0 \\ \mathbf{0} & \text{si } z_j(i) = 0 \end{cases},$$

donde $D_{z_j}(i,:)$ y $D^{(40)}(i,:)$ representan la fila i de las matrices D_{z_j} y $D^{(40)}$, respectivamente, y $\mathbf{0}$ es el vector fila nulo. Esto es, la fila i de D_{z_j} se toma igual a la fila i de $D^{(40)}$, en caso que la $i-$ésima entrada del vector z_j no sea cero. Caso contrario, a la fila i de D_{z_j} se le asigna todas sus entradas iguales a cero.

- Utilizando un algoritmo de *compressed sensing* se determina x_{z_j}, tal que

$$x_{z_j} = \underset{x}{\text{argmín}} \|z_j - D_{z_j} x\|_2^2,$$

sujeto a $\text{Card}(x_{z_j}) \leq c_0$. En este caso nuevamente se utilizó el algoritmo OMP (orthogonal matching pursuit).

- Finalmente, para reconstruir el $j-$ésimo bloque se realiza la multiplicación $D^{(40)} \cdot x_{z_j}$. El resultado de este producto se convierte en un bloque de tamaño 8×8, y corresponde a la reconstrucción del $j-$ésimo bloque de la fotografía.

La Figura 8 muestra los resultados obtenidos en la reconstrucción de la fotografía seleccionada.

75% de pixeles perdidos

Reconstrucción con el algoritmo K-SVD

Figura 8: Reconstrucción de una imagen con 75 % de pixeles perdidos mediante el algoritmo K-SVD.

4.5. Reconocimiento facial

La última aplicación que se mostrará en el artículo se basa en [29], la cual utiliza la descomposición en valores singulares para realizar un reconocimiento facial en una base de imágenes. Para ello suponga que se tienen N imágenes de rostros, cada una de tamaño $m \times n$ pixeles. Sea $S = (f_1\ f_2\ \dots\ f_N)$ la matriz de tamaño $mn \times N$, donde f_i representa la $i-$ésima imagen redimensionada como columna de tamaño $mn \times 1$. Por su parte, el rostro promedio, $\overline{f}$, se define como $\overline{f} = \frac{1}{N}\sum_{i=1}^{N} f_i$, que para efectos de la matriz S representa una columna promedio. A partir de S y $\overline{f}$, se define la matriz $A = (a_1\ a_2\ \dots\ a_N)$, donde $a_i = f_i - \overline{f}$, para todo $i = 1, \dots, N$.

Si $r = \mathrm{r}(A)$, entonces la SVD de la matriz A, dada por $A = U\Sigma V^T$, genera matrices unitarias U y V de la forma $U = (u_1\ u_2\ \dots\ u_r\ u_{r+1}\ \dots\ u_{mn})$ y $V = (v_1\ v_2\ \dots\ v_r\ v_{r+1}\ \dots\ v_N)$. Se sabe que las columnas de $U_r = (u_1\ u_2\ \dots\ u_r)$ forman una base ortonormal del espacio generado por las columnas de A. En este caso a cada vector u_i, con $i \leq r$, se le denomina *rostro-base* del *espacio de rostros* formado por el conjunto inicial de imágenes de entrenamiento. Dado un nuevo rostro f, de tamaño $m \times n$, también redimensionado como vector columna de tamaño $mn \times 1$, es posible determinar las coordenadas para $f - \overline{f}$ asociadas a la base generada por las columnas de U_r. Esto es, se busca el vector de coordenadas $w = (w_1, w_2, \dots, w_r)^T$, con $w_i \in \mathbb{R}$ para todo i, tal que $f - \overline{f} = w_1u_1 + w_2u_2 + \cdots + w_ru_r$. En efecto,

$$\begin{aligned} & f - \overline{f} = w_1u_1 + w_2u_2 + \cdots + w_ru_r \\ \iff \quad & f - \overline{f} = (u_1\ u_2\ \dots\ u_r) \cdot (w_1, w_2, \dots, w_r)^T \\ \iff \quad & (u_1\ u_2\ \dots\ u_r)^T \left(f - \overline{f}\right) = (w_1, w_2, \dots, w_r)^T \end{aligned}$$

Por ende, basta tomar $w = (u_1\ u_2\ \dots\ u_r)^T \left(f - \overline{f}\right)$. Similarmente, para cada una de los rostros de entrenamiento, f_i, se tiene que sus coordenadas en el espacio de rostros están dadas por los vectores $x_i = (u_1\ u_2\ \dots\ u_r)^T \cdot (f_i - \overline{f})$, para $i = 1, \dots, r$. Por lo tanto, para determinar si f corresponde a alguna de las imágenes del conjunto de rostros de entrenamiento (o alguna variante de alguno de ellas), se procede a comparar el vector $\mathbf{w}$ con los vectores de coordenadas x_i. Si $x_j = \underset{1 \leq i \leq r}{\text{argmín}} \|w - x_i\|_2$, entonces el rostro f será identificado como el rostro f_j si $\|w - x_j\|_2 < \varepsilon_0$, donde ε_0 es una tolerancia previamente definida. En caso contrario, f debe ser reportado como un rostro desconocido. El Algoritmo 3 muestra

los pasos a seguir para la implementación.

Algoritmo 3 Reconocimiento facial

Entrada: $\{f_1, f_2, \ldots, f_N\}$, f, ε_0.

Salida: El rostro identificado o un mensaje indicando que el rostro es desconocido.

1: Construya $S = (f_1\ f_2\ \ldots\ f_N)$.
2: Calcule $\overline{f} = \frac{1}{N}\sum_{i=1}^{N} f_i$.
3: Construya $A = (a_1\ a_2\ \ldots\ a_N)$, donde $a_i = f_i - \overline{f}$.
4: Calcule $A = U\Sigma V^T$.
5: Defina $U_r = (u_1\ u_2\ \ldots\ u_r)$, donde $r = \mathrm{r}(A)$.
6: Para $i = 1, \ldots, N$ calcule $x_i = (u_1\ u_2\ \ldots\ u_r)^T \cdot (f_i - \overline{f})$
7: Calcule $w = (u_1\ u_2\ \ldots\ u_r)^T \cdot (f - \overline{f})$
8: Calcule $\varepsilon = \min\limits_{1\leq i\leq r} \|w - x_i\|_2 = \|w - x_j\|_2$, para algún $j \in \{1, 2, \ldots, N\}$
9: **si** $\varepsilon < \varepsilon_0$ **entonces**
10: **retornar** El rostro corresponde a la imagen f_j.
11: **si no**
12: **retornar** El rostro es desconocido.
13: **fin si**

Para fines ilustrativos, el Algoritmo 3 se aplicó a un conjunto de 500 imágenes de rostros extraídas de [12], las cuales fueron reescalas para que quedaran del mismo tamaño (156×111). Adicionalmente, dichas imágenes fueron convertidas a escala de gris. La base de datos en [12] incluye, para cada persona, 15 imágenes diferentes, con variantes en la posición y la expresión facial. La matriz S fue construida utilizando las primeras 10 imágenes de las 50 personas que se muestran en la Figura 9. Las cinco imágenes restantes de cada persona se utilizaron para realizar el experimento de reconocimiento facial.

Figura 9: Imágenes de rostros utilizadas en el experimento.

La Figura 10 muestra dos ejemplos particulares de los resultados obtenidos al aplicar el Algoritmo 3. En cada ejemplo, en la parte superior se muestran cinco imágenes que no fueron incluidas en la matriz S. Por su parte, en la parte inferior se muestran las imágenes con las que el Algoritmo 3 realizó la identificación. Las flechas utilizadas indican a cuál imagen detectó el algoritmo, para cada caso.

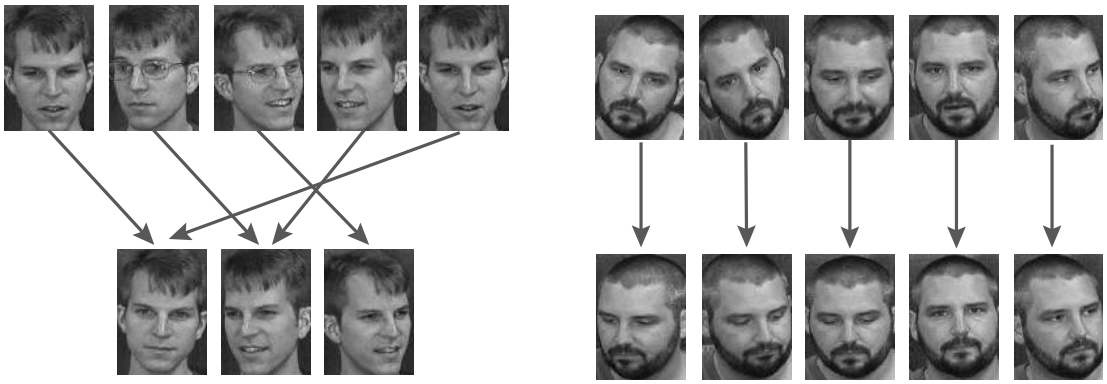


Figura 10: Resultados obtenidos al aplicar el algortimo de reconocimiento facial.

5. CONCLUSIONES

Este artículo presentó una revisión completa y autónoma de la descomposición en valores singulares de una matriz. En primera instancia se desarrolló con detalle los aspectos teóricos que permitieron justificar la existencia de la SVD y, posteriormente, se encausó al lector hacia aplicaciones concretas de la descomposición. En particular, la sección 3.1 mostró cómo la SVD logra condensar la información de una matriz en los subespacios asociados a los valores singulares de mayor magnitud, dando esto sentido a la aplicación de la compresión de imágenes (sección 4.1). Además, se presentaron otras aplicaciones de la SVD en el área del procesamiento de imágenes, abarcando los temas de modelado de fondo de imágenes para la detección de movimiento en videos (sección 4.3), eliminación de ruido de una imagen (secciones 4.2 y 4.4) y reconocimiento facial (sección 4.5).

Por la forma en que está estructurado, el artículo permite a un lector con conocimiento básico en Álgebra Lineal una mejor comprensión de este tipo de factorización. Por ello, puede ser considerado de interés como lectura complementaria en cualquier curso de Álgebra Lineal en el que se desee mostrar aplicaciones de esta área de la Matemática.

Reconocimientos:

Este artículo se realizó como parte del proyecto *FroSigPro* (código #1440037, periodo 2019-2020) inscrito en la Vicerrectoría de Investigación del Instituto Tecnológico de Costa Rica.

RECEIVED: JULY, 2020.

REVISED: NOVEMBER, 2020.

REFERENCIAS

[1] AHARON, M., ELAD, M. y BRUCKSTEIN, A. (2006): K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation, **IEEE Transactions on Signal Processing**, 54(11):4311–4322.

[2] ARTIDIELLO, S., CORDERO, A., and VASSILEVA, J. R. T. M. (2020): Generalized inverses estimations by means of iterative methods with memory, **Mathematics**, 8(1):1–13.

[3] BARATA, J.C. y HUSSEIN, M. S. (2011): The Moore–Penrose pseudoinverse: A tutorial review of the theory, **Brazilian Journal of Physics**, 42(1-2):146–165.

[4] BELTRAMI, E. (1873): Sulle funzioni bilineari, **Giornale di Matematiche ad Uso degli Studenti Delle Universita**, 11:98–106.

[5] BEN-ISRAEL, A. (1965): An iterative method for computing the generalized inverse of an arbitrary matrix, **Mathematics of Computation**, 19(91):452–455.

[6] BERNSTEIN, D.S. (2009): **Matrix Mathematics: Theory, Facts, and Formulas (Second Edition),** Princeton. Princeton University Press. New Jersey.

[7] CHUNG, J. y CHUNG, M. (2013): Computing optimal low-rank matrix approximations for image processing, In **2013 Asilomar Conference on Signals, Systems and Computers**, pages 670–674.

[8] DATTA, B.N. (2010): **Numerical Linear Algebra and Applications,** SIAM. Philadelphia.

[9] ECKART, C. y YOUNG, G. (1936): The approximation of one matrix by another of lower rank, **Psychometrika**, 1(3):211–218.

[10] ELDAR, Y.C. y KUTYNIOK, G. (2012): **Compressed Sensing: Theory and Applications,** Cambridge University Press. Cambridge.

[11] FRIEDLAND, S. y TOROKHTI, A. (2007): Generalized rank-constrained matrix approximations, **SIAM Journal on Matrix Analysis and Applications**, 29(2):656–659.

[12] GEORGIA TECH, Georgia tech face database, Disponible en `http://sipi.usc.edu/database/`. Consultado: 14-2,2020.

[13] GOLUB, G.H. y VAN LOAN, C.F. (2013): **Matrix Computations,** Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press. Baltimore.

[14] GONZALEZ, R., WOOD, R. y EDDINS, S. (2010): **Digital image processing using MATLAB,** Mc Graw Hill Education. India.

[15] HARVILLE, D.A. (2008): **Matrix Algebra From a Statistician's Perspective,** Springer. New York.

[16] HORN, R.A. y JOHNSON, C.R. (1990): **Matrix Analysis,** Cambridge University Press. Cambridge.

[17] JOHNSON, R. (1963): On a theorem stated by Eckart and Young, **Psychometrika**, 28(3):259–263.

[18] JOLLIFE, I.T. (2006): **Principal Component Analysis,** Springer. New York.

[19] JORDAN, C. (1874): Mémoire sur lesformes bilinéaires, **J. Math. Pures Appl., Deuxieme Série**, 19:35–54.

[20] LANG, S. (1987): **Linear Algebra,** Springer Undergraduate Texts in Mathematics and Technology. Springer. New York.

[21] LI, W. y LI, Z. (2010): A family of iterative methods for computing the approximate inverse of a square matrix and inner inverse of a non-square matrix, **Applied Mathematics and Computation**, 215(9):3433–3442.

[22] MARTÍNEZ-FERNÁNDEZ, J. J. (2005): La descomposición en valores singulares (svd) y algunas de sus aplicaciones, **La Gaceta de la RSME**, 8(3):796–810.

[23] PRATT, W.K. (2013): **Introduction to Digital Image Processing,** CRC Press.

[24] RYZE, M. (2018): Shopping, people, commerce, mall, many, crowd, walking free stock video footage youtube, Disponible en `https://www.youtube.com/watch?v=WvhYuDvH17I`. Consultado: 14-2,2020.

[25] STEWART, G. W. (1993): On the early history of the singular value decomposition, **SIAM Review**, 35(4):551–566.

[26] UNIVERSITY OF SOUTHERN CALIFORNIA, Signal and image processing institute, Disponible en `http://sipi.usc.edu/database/`. Consultado: 14-2,2020.

[27] WILLINK, T. J. (2008): Efficient adaptive SVD algorithm for MIMO applications, **IEEE Transactions on Signal Processing**, 56(2):615–622.

[28] WU, C. y TSAI, P. (2019): An svd processor based on Golub–Reinsch algorithm for MIMO precoding with adjustable precision, **IEEE Transactions on Circuits and Systems I: Regular Papers**, 66(7):2572–2583.

[29] ZENG, G. (2007): Facial recognition with singular value decomposition, In Elleithy, K., editor, **Advances and Innovations in Systems, Computing Sciences and Software Engineering**, pages 145–148, Dordrecht. Springer. Netherlands.

[30] ZHOU, T. y TAO, D. (2011): Godec: Randomized low-rank & sparse matrix decomposition in noisy case, **Proc. 28th ICML**, pages 33–40.

Anexo 2

Applied Mathematics and Computation 409 (2021) 126381

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

A general class of arbitrary order iterative methods for computing generalized inverses☆

Alicia Cordero[a], Pablo Soto-Quiros[b], Juan R. Torregrosa[a,*]

[a] Institute for Multidisciplinary Mathematics, Universitat Politècnica de València, Camino de Vera s/n, 46022 València, Spain
[b] Escuela de Matemática, Instituto Tecnológico de Costa Rica, Cartago, 30101, Costa Rica

ARTICLE INFO

Article history:
Received 13 April 2020
Revised 24 March 2021
Accepted 15 May 2021

Keywords:
Matrix equations
Inverse matrix
Iterative method
Order of convergence
Dependence on initial estimations

ABSTRACT

A family of iterative schemes for approximating the inverse and generalized inverse of a complex matrix is designed, having arbitrary order of convergence p. For each p, a class of iterative schemes appears, for which we analyze those elements able to converge with very far initial estimations. This class generalizes many known iterative methods which are obtained for particular values of the parameters. The order of convergence is stated in each case, depending on the first non-zero parameter. For different examples, the accessibility of some schemes, that is, the set of initial estimations leading to convergence, is analyzed in order to select those with wider sets. This wideness is related with the value of the first non-zero value of the parameters defining the method. Later on, some numerical examples (academic and also from signal processing) are provided to confirm the theoretical results and to show the feasibility and effectiveness of the new methods.

1. Introduction

Computing the inverse matrix of large-scale nonsingular complex matrices is essential in some engineering applications and it has binding computational requirements. In a wide variety of topics, such as signal and image processing [1–4], cryptography [5,6], control system analysis [7,8], and others, the inverse or different generalized inverses must be computed in order to solve the involved issues.

There are several procedures to address this problem, divided into two classes: the direct schemes such as Gaussian elimination with partial pivoting, which require big amount of calculations and memory for large scale problems, and the iterative schemes like the Schulz-type iterations, in which an approximation of the inverse matrix is obtained up to the desired precision.

In the last decade, many iterative schemes of different orders have been designed for approximating the inverse or some generalized inverse (Moore-Penrose inverse, Drazin inverse, etc.) of a complex matrix A. In this manuscript, we focus on constructing a new class of iterative methods, free of inverse operators and with arbitrary order of convergence, for finding the inverse of a non-singular complex matrix. We also analyze the proposed schemes for computing the Moore-Penrose

☆ This research was supported in part by PGC2018-095896-B-C22 (MCIU/AEI/FEDER, UE) and in part by VIE from *Instituto Tecnológico de Costa Rica* (Research #1440037).

* Corresponding author.
E-mail addresses: acordero@mat.upv.es (A. Cordero), jusoto@tec.ac.cr (P. Soto-Quiros), jrtorre@mat.upv.es (J.R. Torregrosa).

https://doi.org/10.1016/j.amc.2021.126381

inverse of complex rectangular matrices. The designed family depends on several real parameters, which by taking particular values provide us numerous known methods constructed by other authors.

Recently, some authors have presented a unified approach of the computation of different generalized inverses by means of the calculation of outer inverses $A^{(2)}_{T,S}$ for appropriate choice of T and S (see, for example, [9,10]).

Perhaps, the most common scheme to compute the inverse A^{-1} of a non-singular complex matrix A is the Schulz's method given in 1933, with iterative expression

$$X_{k+1} = X_k(2I - AX_k), \quad k = 0, 1, \ldots \tag{1}$$

where I is the identity matrix with the same size of A. Schulz in [11] demonstrated the convergence of sequence $\{X_k\}_{k\geq 0}$, obtained from (1), to the inverse A^{-1} is guaranteed if the eigenvalues of matrix $I - AX_0$ are lower than 1. Taking into account that the residuals $E_k = I - AX_k$, $k = 0, 1, \ldots$ satisfy $\|E_{k+1}\| \leq \|E_k\|^2$, expression (1) has quadratic convergence. In general, in the Schulz-type methods it is common to use as initial approach $X_0 = \alpha A^*$ or $X_0 = \alpha A$, where $0 < \alpha < 2/\rho(A^*A)$, where A^* is the conjugate transpose of A and $\rho(\cdot)$ the spectral radius. In this paper, we use n the case of inverses and also in generalized inverses, the initial estimation $X_0 = \beta A^*/\|A\|^2$. We will also analyze the set of values of the parameter β guaranteing the convergence.

Li et al. in [12] investigated the family of iterative methods

$$X_{k+1} = X_k\left(\nu I - \frac{\nu(\nu-1)}{2}AX_k + \frac{\nu(\nu-1)(\nu-2)}{3!}(AX_k)^2 - \ldots + (-1)^{\nu-1}(AX_k)^{\nu-1}\right), \quad \nu = 2, 3, \ldots$$

with $X_0 = \alpha A^*$. They proved the convergence of ν-order of $\{X_k\}_{k\geq 0}$ to the inverse of matrix A. This class was used by Chen et al. in [13] and by Li et al. in [14] for approximating the Moore-Penrose inverse.

Soleymani et al. in [15] also proposed a fourth-order iterative scheme for calculating the inverse and the Moore-Penrose inverse, with iterative expression

$$X_{k+1} = \frac{1}{2}X_k(9I - AX_k(16I - AX_k(14I - AX_k(6I - AX_k)))), \quad k = 0, 1, \ldots$$

On the other hand, Stanimirović et al. in [16] proposed the following scheme of order eleven for computing the generalized outer inverse $A^{(2)}_{T,S}$

$$X_{k+1} = X_k(I + (R_k + R_k^2)(I + (R_k^2 + R_k^4)(I + R_k^4))), \quad k = 0, 1, \ldots$$

being $R_k = I - AX_k$, $k = 0, 1, \ldots$.

Kaur et al. in [17], by using also the hyperpower iterative method, designed the following scheme of order five for obtaining the weighted Moore-Penrose inverse

$$X_{k+1} = X_k(5I - 10AX_k + 10(AX_k)^2 - 5(AX_k)^3 + (AX_k)^4), \quad k = 0, 1, \ldots$$

These papers are some of the manuscripts that have been published to approximate the inverse of a non-singular matrix or some of the generalized inverses of arbitrary matrices. In this paper, we design a parametric family of iterative methods with arbitrary order of convergence that contains many of the methods constructed up to date. For each fixed value of the order of convergence, we still have a class of iterative methods depending on several parameters. We are able to select the most stable members of these classes in terms of the wideness of the set of initial estimations converging to the solution (defined as basin of convergence). The best members are chosen to be compared, both dynamical and numerically, with some known methods.

The rest of this manuscript is organized as follows. Section 2 is devoted to the construction of the proposed class of iterative schemes, proving its convergence to the inverse of a nonsingular complex matrix, with arbitrary order of convergence. In Section 3, it is proven that the same family of iterative methods is able to converge to the Moore-Penrose inverse of a complex matrix of size $m \times n$. Some particular cases of this class are found in Section 4, corresponding to existing methods proposed by different authors. Section 5 is devoted to study the dependence on the initial estimations of the methods of the same class, with a fixed value of p, and also the effect of the kind of initial estimation in the wideness of the set of converging initial estimations. A wide range of numerical test are also found in Section 6, checking the robustness and applicability of the proposed methods, compared with other known ones on different kinds of matrices. Finally, some conclusions are stated.

2. A class of iterative schemes for matrix inversion

In this section, we are going to present a parametric family of iterative schemes for obtaining approximated inverses and to prove their order of convergence. Firstly, we define the following polynomial matrix.

Definition 1. Let $B \in \mathbb{C}^{m\times m}$ be a complex square matrix and $p > 0$ a positive integer number. We define the polynomial matrix $G_p(B)$ as

$$G_p(B) = \sum_{j=1}^{p}(-1)^{j-1}C_p^j B^{j-1} = C_p^1 I - C_p^2 B + C_p^3 B^2 + \ldots + (-1)^{p-1}C_p^p B^{p-1},$$

where C_p^j is the combinatorial number $C_p^j = \binom{p}{j} = \frac{p!}{j!(p-j)!}$.

The following result is a technical lemma whose proof is straightforward by using mathematical induction with respect to p.

Lemma 1. *Let $p > 0$ be a positive integer and $B \in \mathbb{C}^{m\times m}$. Then $(I-B)^p = I - BG_p(B)$.*

Let $A \in \mathbb{C}^{m\times m}$ be a nonsingular matrix and $p > 1$ a positive integer. Let $\{\alpha_1, \alpha_2, \ldots, \alpha_p\}$ be a set of real parameters such that $\alpha_i \in [0,1]$, for $i = 1, 2, \ldots, p-1, \alpha_p \in]0,1]$ and $\sum_{i=1}^{p} \alpha_i = 1$.

We assume a sequence of complex matrices $\{X_0, X_1, \ldots, X_n, \ldots\}$, of size $m \times m$, satisfying following two conditions:

(a) $\|I - AX_0\| = \gamma_0 < 1$,

(b) $I - AX_{k+1} = \sum_{i=1}^{p} \alpha_i (I - AX_k)^i$.

We consider the family of iterative schemes described as

$$X_{k+1} = X_k \sum_{i=1}^{p} \alpha_i G_i(AX_k), \quad k = 0, 1, \ldots \tag{2}$$

For each positive integer p, $p > 1$, we have a different class of iterative methods, whose order of convergence depends on the value of parameters α_i, $i = 1, 2, \ldots, p$.

In the following results, we prove the convergence of these schemes to the inverse of A.

Proposition 1. *Let $A \in \mathbb{C}^{m\times m}$ be a nonsingular matrix and $p > 1$ a positive integer. Let us consider the sequence of complex matrices constructed as*

$$X_{k+1} = X_k \sum_{i=1}^{p} \alpha_i G_i(AX_k), \quad k = 0, 1, \ldots,$$

where $\alpha_i \in [0,1]$, for $i = 1, 2, \ldots, p-1$, $\alpha_p \in]0,1]$ and $\sum_{i=1}^{p} \alpha_i = 1$. Then, condition

$$I - AX_{k+1} = \sum_{i=1}^{p} \alpha_i (I - AX_k)^i,$$

is equivalent to

$$X_{k+1} = X_k \sum_{i=1}^{p} \alpha_i \left(\sum_{j=1}^{i} (-1)^{j-1} C_i^j (AX_k)^{j-1} \right). \tag{3}$$

Proof. From Lemma 1 we obtain:

$$\begin{aligned} I - AX_{k+1} = \sum_{i=1}^{p} \alpha_i (I - AX_k)^i &\Leftrightarrow I - AX_{k+1} = \sum_{i=1}^{p} \alpha_i (I - AX_k G_i(AX_k)) \\ &\Leftrightarrow I - AX_{k+1} = I - \sum_{i=1}^{p} \alpha_i AX_k G_i(AX_k) \\ &\Leftrightarrow AX_{k+1} = AX_n \sum_{i=1}^{p} \alpha_i G_i(AX_k) \\ &\Leftrightarrow X_{k+1} = X_k \sum_{i=1}^{p} \alpha_i G_i(AX_k) \end{aligned}$$

Therefore, the result is proven. □

By mathematical induction it is also easy to prove the following result.

Proposition 2. *Let us consider sequence $\{X_k\}_{k\geq 0}$ obtained from expression (2). If $\|I - AX_0\| < 1$, then*

$$\|I - AX_k\| < 1, \quad k = 1, 2, \ldots$$

Therefore, the main result regarding matrix inversion can be stated.

Theorem 1. *Let $A \in \mathbb{C}^{m\times m}$ be a nonsingular matrix and an initial guess $X_0 \in \mathbb{C}^{m\times m}$. Let $\alpha_1, \ldots, \alpha_p$ be nonnegative real numbers such that $\alpha_i \in [0,1]$, $\alpha_p \neq 0$ and $\sum_{i=1}^{p} \alpha_i = 1$. If $\|I - AX_0\| < 1$, then sequence $\{X_k\}_{k\geq 0}$, obtained by (2), converges to A^{-1} with convergence order q for any $p > 1$, where $q = \min_{i=1,2,\ldots,p} \{i \mid \alpha_i \neq 0\}$.*

Proof. By using previous condition (b), we have

$$\|I - AX_{k+1}\| = \left\| \sum_{i=1}^{p} \alpha_i (I - AX_k)^i \right\| \leq \sum_{i=1}^{p} \alpha_i \|I - AX_k\|^i = \sum_{i=q}^{p} \alpha_i \|I - AX_k\|^i,$$

but, from Proposition 2 we can assure

$$\|I - AX_{k+1}\| \leq \sum_{i=q}^{p} \alpha_i \|I - AX_k\|^q = \|I - AX_k\|^q.$$

Thus,

$$\|I - AX_{k+1}\| \leq \|I - AX_k\|^q. \tag{4}$$

By using Equation (4) and mathematical induction, we prove

$$\|I - AX_{k+1}\| \leq \|I - AX_0\|^{q^{k+1}}.$$

Therefore, as $k \to \infty$:

$$\|I - AX_{k+1}\| \to 0 \Rightarrow I - AX_{k+1} \to \mathbf{0} \Rightarrow \{X_n\}_{k\geq 0} \to A^{-1},$$

with order q. □

In the next section, we extend family (2) for approximating the Moore-Penrose inverse of any complex rectangular matrix.

3. A class of iterative schemes for computing Moore-Penrose inverse

Let A be a $m \times n$ complex matrix. The Moore-Penrose inverse of A (also called pseudoinverse), denoted by $A^{\dagger}$, is the unique $n \times m$ matrix X satisfying

$$AXA = A, \quad XAX = X, \quad (AX)^* = AX, \quad (XA)^* = XA.$$

This generalized inverse plays an important role in several fields, such as eigenvalue problems and the linear least square problems [18]. It can be obtained, explicitly, from the singular value decomposition of A but, with a high computational cost. Therefore, it is interesting to have efficient iterative methods to approximate this matrix. In this section, we prove how family (2) allows us to compute the pseudoinverse with the same order of convergence that in the previous section, where the inverse of a square matrix was calculated. First, we establish the following technical result, that is proven by mathematical induction (similar to Lemma 2.1 of [13]), although other authors state similar results in the context of outer inverses (see, for example, [9]).

Lemma 2. *Let us consider $X_0 = \alpha A^*$, where $\alpha \in \mathbb{R}$, and sequence $\{X_k\}_{k\geq 0}$ generated by family (2). For any $k \geq 0$, it is held that*

$$(X_k A)^* = X_k A, \quad (AX_k)^* = AX_k, \quad X_k A A^{\dagger} = X_k, \quad A^{\dagger} A X_k = X_k. \tag{5}$$

Proof. For $n = 0$, the first two equations can readily be verified, and we only give a verification to the last two equations:

$$X_0 A A^{\dagger} = \alpha A^* A A^{\dagger} = \alpha A^* (A A^{\dagger})^* = \alpha A^* (A^{\dagger})^* A^* = \alpha (A A^{\dagger} A)^* = \alpha A^* = X_0,$$

and

$$A^{\dagger} A X_0 = (A^{\dagger} A)^* (\alpha A^*) = \alpha A^* (A^{\dagger})^* A^* = \alpha (A A^{\dagger} A)^* = \alpha A^* = X_0.$$

Let us assume now that the conclusion holds for k. We now show that it continues to be held for $k+1$. Using the iterative procedure in Equation (2), we have

$$\begin{aligned}(X_{k+1}A)^* &= \left(\left[X_n \sum_{i=1}^{p} \alpha_i \left(\sum_{j=1}^{i} (-1)^{j-1} C_i^j (AX_k)^{j-1} \right) \right] A \right)^* \\ &= \left(\sum_{i=1}^{p} \alpha_i \left(\sum_{j=1}^{i} (-1)^{j-1} C_i^j (X_k A)^j \right) \right)^*\end{aligned}$$

$$= \sum_{i=1}^{p} \alpha_i \left(\sum_{j=1}^{i} (-1)^{j-1} C_i^j (X_k A)^j \right)$$
$$= \left[X_k \sum_{i=1}^{p} \alpha_i \left(\sum_{j=1}^{i} (-1)^{j-1} C_i^j (A X_k)^{j-1} \right) \right] A = X_{k+1} A,$$

where the fourth equality uses that $(X_k A)^* = X_{k+1} A$. Thus, the first equality of (5) holds for $k+1$. Second equality can be proven in a similar way.

For the third equality of (5), we use the assumption that $X_k A A^\dagger = X_k$ and the iterative procedure in (2):

$$X_{k+1} A A^\dagger = \left[X_n \sum_{i=1}^{p} \alpha_i \left(\sum_{j=1}^{i} (-1)^{j-1} C_i^j (A X_k)^{j-1} \right) \right] A A^\dagger$$
$$= \sum_{i=1}^{p} \alpha_i \left(\sum_{j=1}^{i} (-1)^{j-1} C_i^j (X_k A)^{j-1} X_k A A^\dagger \right)$$
$$= X_n \sum_{i=1}^{p} \alpha_i \left(\sum_{j=1}^{i} (-1)^{j-1} C_i^j (A X_k)^{j-1} \right) = X_{k+1}.$$

Hence the third equality of (5) holds for $k+1$. The fourth equality can similarly be proven, and the desired result follows. □

Now, some technical results are presented.

Lemma 3 ([13]). *Let $A \in \mathbb{C}^{m\times n}$. Let $X_0 = \alpha A^*$ be, where $\alpha < \frac{1}{\sigma_1^2}$ and σ_1 is the largest singular value of A. Then $\|(X_0 - A^\dagger)A\| < 1$.*

The following result is straightforward by using Lemma 2.

Lemma 4. *Let $A \in \mathbb{C}^{m\times n}$ and $\{X_k\}_{k\geq 0}$ be the sequence generated by (2). Let us consider $E_k = X_k - A^\dagger$. Then,*

1. $\|X_k - A^\dagger\| \leq \|E_k A\| \|A^\dagger\|$,
2. $(I - A^\dagger A) E_k A = \mathbf{0}$.

Lemma 5. *Let $A \in \mathbb{C}^{m\times n}$ and $\{X_k\}_{k\geq 0}$ be the sequence generated by (2). We denote $E_k = X_k - A^\dagger$. Then*

$$E_{k+1} A = \sum_{i=1}^{p} \alpha_i (-1)^{i-1} (E_k A)^i, \quad k = 0, 1, \ldots \tag{6}$$

Proof. By using the fact that $\sum_{i=1}^{p} \alpha_i = 1$, we have

$$E_{k+1} A = \left[X_k \sum_{i=1}^{p} \alpha_i G_i(A X_k) \right] A - A^\dagger A$$
$$= \left[X_k \sum_{i=1}^{p} \alpha_i \left(\sum_{j=1}^{i} (-1)^{j-1} C_i^j (A X_k)^{j-1} \right) \right] A - A^\dagger A$$
$$= \sum_{i=1}^{p} \alpha_i \left(\sum_{j=1}^{i} (-1)^{j-1} C_i^j (X_k A)^j \right) - A^\dagger A$$
$$= \sum_{i=1}^{p} \alpha_i \left(-I + \sum_{j=1}^{i} (-1)^{j-1} C_i^j (X_k A)^j \right) + I - A^\dagger A$$
$$= -\sum_{i=1}^{p} \alpha_i (I - X_k A)^i + I - A^\dagger A.$$

Now, taking into account that $X_k A = A^\dagger A + E_k A$, $(I - A^\dagger A)^q = I - A^\dagger A$ and $(I - A^\dagger A) E_k A = \mathbf{0}$ (from Lemma 4), we get

$$-\sum_{i=1}^{p} \alpha_i (I - X_k A)^i + I - A^\dagger A = -\sum_{i=1}^{p} \alpha_i \left(I - A^\dagger A - E_k A \right)^i + I - A^\dagger A$$

$$
\begin{aligned}
&= -\left(\sum_{i=1}^{p}\alpha_i\Big[(I-A^\dagger A)^i - C_i^1(I-A^\dagger A)^{i-1}E_kA + C_i^2(I-A^\dagger A)^{i-2}(E_kA)^2 + \ldots\right.\\
&\quad \left.+(-1)^{i-1}C_i^{i-1}(I-A^\dagger A)(E_kA)^{i-1} + (-1)^iC_i^i(E_kA)^i\Big]\right) + I - A^\dagger A\\
&= -\left(\sum_{i=1}^{p}\alpha_i\Big[(I-A^\dagger A) - C_i^1(I-A^\dagger A)E_kA + C_i^2(I-A^\dagger A)(E_kA)^2 + \ldots\right.\\
&\quad \left.+(-1)^{i-1}C_i^{i-1}(I-A^\dagger A)(E_kA)^{i-1} + (-1)^iC_i^i(E_kA)^i\Big]\right) + I - A^\dagger A\\
&= -\left(\sum_{i=1}^{p}\alpha_i\Big[(I-A^\dagger A) + (-1)^iC_i^i(E_kA)^i\Big]\right) + I - A^\dagger A\\
&= \sum_{i=1}^{p}\alpha_i(-1)^{i-1}(E_kA)^i.
\end{aligned}
$$

□

Finally, we can state the following result.

Theorem 2. *Let $A \in \mathbb{C}^{m\times n}$ and $q = \min_{i=1,2,\ldots,p}\{i \mid \alpha_i \neq 0\}$. Then, sequence $\{X_k\}_{k\geq 0}$ generated by (2) converges to the Moore-Penrose inverse $A^\dagger$ with qth-order provided that $X_0 = \alpha A^*$, where $\alpha < \frac{1}{\sigma_1^2}$ is a constant and σ_1 is the largest singular value of A.*

Proof. Therefore,

$$\|X_{k+1} - A^\dagger\| \leq \|E_kA\|^q\|A^\dagger\|. \tag{7}$$

Now, from (7) and by applying mathematical induction, we prove

$$\|X_{k+1} - A^\dagger\| \leq \|E_0A\|^{q^{k+1}}\|A^\dagger\|.$$

Therefore, as $n \longrightarrow \infty$:

$$\|X_{k+1} - A^\dagger\| \to 0 \Rightarrow X_{k+1} - A^\dagger \to \mathbf{0} \Rightarrow \{X_k\}_{k\geq 0} \to A^\dagger,$$

with order q. □

4. Proposed family and previous known methods

The proposed class of iterative schemes in (2) is a generalization of other methods constructed with different techniques. Some of them are described below.

1. For any $p > 1$, if $\alpha_1 = \cdots = \alpha_{p-1} = 0$ and $\alpha_p = 1$, then we obtain the method proposed by Li and Li. (see Eq. (2.3) in [12] for inverse case and Eq. (2.1) in [13] for pseudoinverse one). Recall that method proposed by Li and Li generalizes the Newton-Schultz ($p = 2$) and Chebyshev method ($p = 3$).
2. On the other hand, if $\alpha_i = 0$ for $i = 1, 2, \ldots, 8$, $\alpha_9 = \alpha_{12} = 1/8$ and $\alpha_{10} = \alpha_{11} = 3/8$, then we get the method proposed by Soleymani and Stanimirovic (see Eq. (12) in [19]).
3. Also expression (2) gives us the method proposed by Toutounian and Soleymani (see Eq. (18) in [15]), if $\alpha_4 = 1/2$ and $\alpha_5 = 1/2$ and $\alpha_1 = \alpha_2 = \alpha_3 = 0$.
4. When the only not null parameter is $\alpha_7 = 1$, then we obtain method proposed by Soleymani (see Eq. (18) in [20]).
5. In a similar way, if the only parameter different from zero is $\alpha_6 = 1$, then the method proposed by Soleymani, Stanimirovic and Zaka (see Eq. (2.4) in [21]) is obtained.
6. When $\alpha_i = 0$ for $i = 1, 2, \ldots 7$, $\alpha_8 = \alpha_{10} = 1/4$ and $\alpha_9 = 2/4$, the resulting scheme is that proposed by Soleymani in Eq. (9) in [22].
7. The method proposed by Soleymani et al in [23], Eq. (10), appears if $\alpha_1 = \cdots = \alpha_8 = 0$, $\alpha_9 = 7/9$ and $\alpha_{10} = 2/9$.
8. The scheme proposed by Razavi, Kerayechian, Gachpazan and Shateyi, (see Eq. (16) in [24]) is obtained if we choose $\alpha_1 = \ldots = \alpha_9 = 0$, $\alpha_{10} = \alpha_{12} = 1/4$ and $\alpha_{11} = 1/2$ in Eq. (2).
9. When the first eight paramenters are null, $\alpha_9 = 343/729$, $\alpha_{10} = 294/729$, $\alpha_{11} = 84/729$ and $\alpha_{12} = 8/729$ then we get method proposed by Al-Fhaid et al in [25], Eq. (5).
10. If $\alpha_7 = 9/16$, $\alpha_8 = 6/16$, $\alpha_9 = 1/16$ and the rest of parameters are zero, then the scheme proposed by Soleymani is found (see Eq. (3.1) in [26]).
11. When $p = 12$ and the only parameters different from zero are $\alpha_9 = \alpha_{12} = 1/8$ and $\alpha_9 = \alpha_{10} = 3/8$, therefore the method proposed by Liu and Cai. see Eq. (4) in [27]) is obtained.
12. If $\alpha_2 = 0$, $\alpha_1 = 1 - \alpha$ and $\alpha_3 = \alpha$, where $\alpha \in (0, 1]$, then we find the method proposed by Srivastava and Gupta in [28]), Eq. (6).

5. Dependence on initial estimations

A key aspect of an iterative method is its order of convergence and also the set of initial guesses that converge to the estimated solution. When a class of iterative schemes is defined, all with the same order of convergence, a good behavior in terms of wider basins of convergence can make the difference. In this section, we use some graphical tools from complex discrete dynamics in order to deduce which scheme should be used within a class of iterative procedures with the same order of convergence.

Several known methods are used to this purpose, that are included in our proposed class: some of them are the extension for matrix equations of classical methods; the rest have been developed by other researchers in the last years and have been deduced as elements of our proposed class in Section 4.

So, the methods used are second-order Newton-Schulz (NS) scheme, with iterative expression (see [11])

$$X_{k+1} = 2X_k - X_kAX_k, \ \ k \geq 0,$$

Chebyshev's scheme (CH), with third-order of convergence, deduced by Amat et al. in [29],

$$X_{k+1} = X_k(3I - AX_k(3I - AX_k)), \ \ k \geq 0.$$

On the other hand, seventh-order scheme (SS) defined by Soleymani and Stanimirovich in [19] is also considered,

$$X_{k+1} = -1/8X_k(-7I + 9Y_k - 5Y_k^2 + Y_k^3)(12I - 42Y_k + 103Y_k^2 - 156Y_k^3 + 157Y_k^4 - 104Y_k^5 + 43Y_k^6 - 10Y_k^7 + Y_k^8), \ \ k \geq 0,$$

where $Y_k = AX_k$.

It is also interesting to compare the results with the fourth-order procedure by Toutonian and Soleymani (TS) [15], whose iterative expression is

$$X_{k+1} = (1/2)X_k(9I - Y_k(16I - Y_k(14I - Y_k(6I - Y_k)))), \ \ k \geq 0,$$

where $Y_k = AX_k$. Also, third-order Homeier's method (HM) is expressed by Li et al. as (see [30])

$$X_{k+1} = X_k(I + (1/2)(I - AX_k)(I + (2I - AX_k)^2)), \ \ k \geq 0,$$

and also Midpoint scheme (MP) has third-order of convergence and the iterative expression (see [30]) provided by Li et al,

$$X_{k+1} = (I + 1/4(I - X_kA)(3I - X_kA)^2)X_k, \ \ k \geq 0.$$

Let us remark that the last two methods do not belong to the family of proposed methods.

To observe which is the performance of proposed and existing methods by using different initial estimations, we use the dynamical planes. A dynamical plane is obtained by iterating a method on a wide set of initial estimations. To generate it, we used a mesh of 400×400 points as values of parameter $\beta \in \mathbb{C}$, used to define the initial estimation of the process X_0. We paint in orange the values of β whose related starting points converge to the inverse (or pseudoinverse) of matrix A, with a tolerance of 10^{-3}. Those points whose orbit (set of consecutive iterates) achieve the maximum of 80 iterations are painted in black.

Firstly, let us find an estimation of the inverse of Toeplitz matrix

$$A_1 = \begin{pmatrix} 1 & 1 & 1 \\ -1 & 1 & 1 \\ 0 & -1 & 1 \end{pmatrix}.$$

We firstly employ the initial estimation $X_0 = \beta I$, being I the identity matrix of the size of the matrix to be inverted. In order to satisfy condition $\|AX_0 - I\| < 1$, it is necessary that $|\beta - 1| < 1$. In Fig. 1, it can be observed that the set of converging initial values is narrow. However, it is wider in HM and MP cases than in NS or CH ones. In this case, we have worked with a the mesh in $[0, 1] \times [-0.5, 0.5]$ corresponding to real and imaginary part of a complex value of β.

Now, let us change the initial estimation. If we consider $X_0 = \beta \dfrac{A_1^T}{\|A_1\|^2}$, then the set of converging initial values is wider than in previous case (see Fig. 2). Nevertheless, the widest ones correspond again to HM and MP methods. This is the reason why we have constructed the mesh in $[-1, 4] \times [-2.5, 2.5]$.

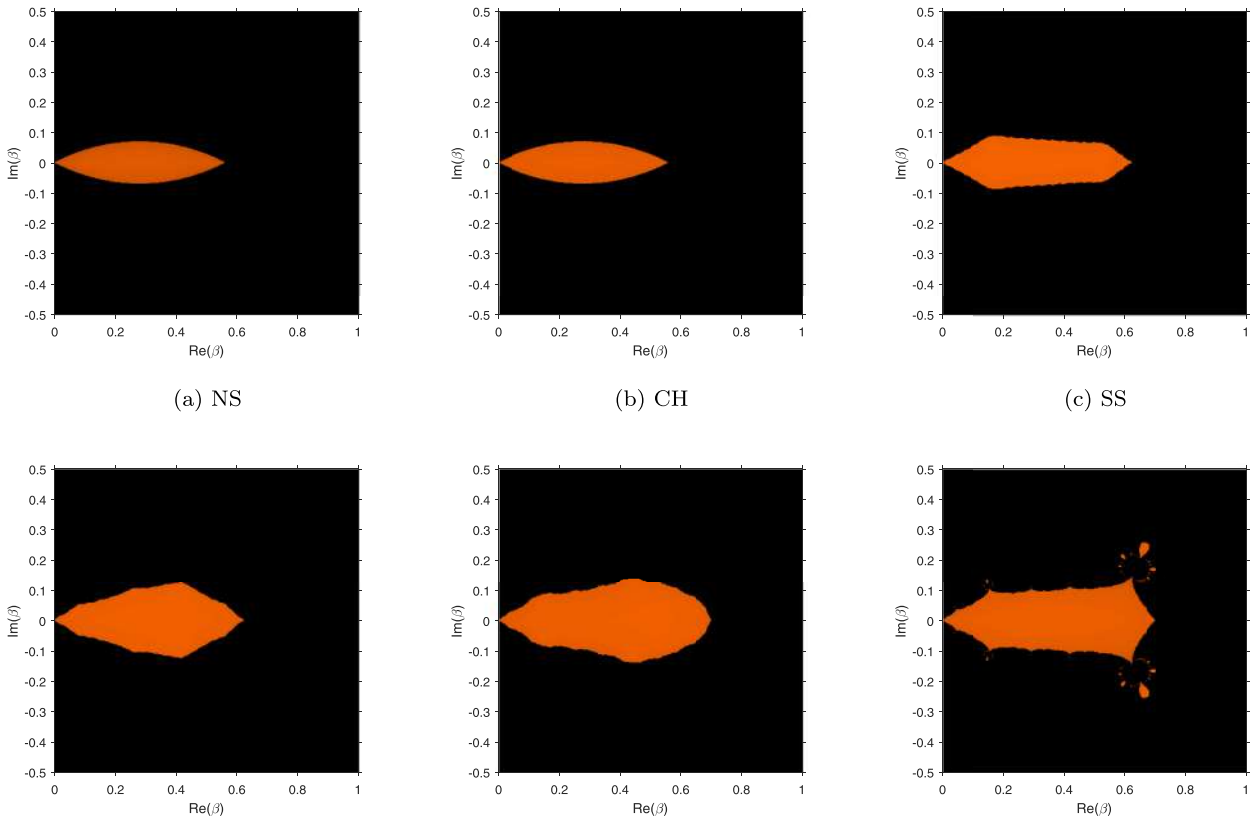
Let us observe the case $p = 2$ in expression (2), where, fixing $\alpha_2 = 1 - \alpha_1$, a class of first-order (at least) iterative schemes including Newton-Schulz's method (for $\alpha_1 = 0$) is obtained,

$$X_{k+1} = X_k(\alpha_1 + (1 - \alpha_1)(2I - AX_k)).$$

In Fig. 3, it can be observed that the basin of convergence (set of values of β that, taken as initial estimation, allows the process to converge) is wider for higher values of $|\beta|$. However, the color of these basins are brighter for lower values of $|\beta|$; the reason is that the order of convergence is only linear as $\alpha_1 \neq 0$ and it is slower for higher values of β.

Let us consider now a class of methods defined by the general expression of the family (2) with $p = 3$, $\alpha_1 = 0$ and $\alpha_2 + \alpha_3 = 1$,

$$X_{k+1} = X_k\big(\alpha_2(2I - AX_k) + (1 - \alpha_2)(3I - 3(AX_k) + (AX_k)^2\big).$$



(a) NS (b) CH (c) SS

(d) TS (e) HM (f) MP

Fig. 1. Dynamical planes corresponding to $X_0 = \beta I$ of different methods on Toeplitz matrix A_1.

The members of this class have, at least, second order of convergence if $\alpha_2 \neq 0$ and third-order (the corresponding iterative scheme is Chebyshev's one) if $\alpha_2 = 0$. As the initial estimation taken is $X_0 = \beta \dfrac{A_1^T}{\|A_1\|^2}$, in Fig. 4, we notice that also the wideness of the basin of attraction of the inverse matrix is bigger for higher absolute values of β but this amplitude is lower than in the case of the members of the family for $p = 2$, for the same value of β. That is, the order is higher but the set of converging values of β is lower.

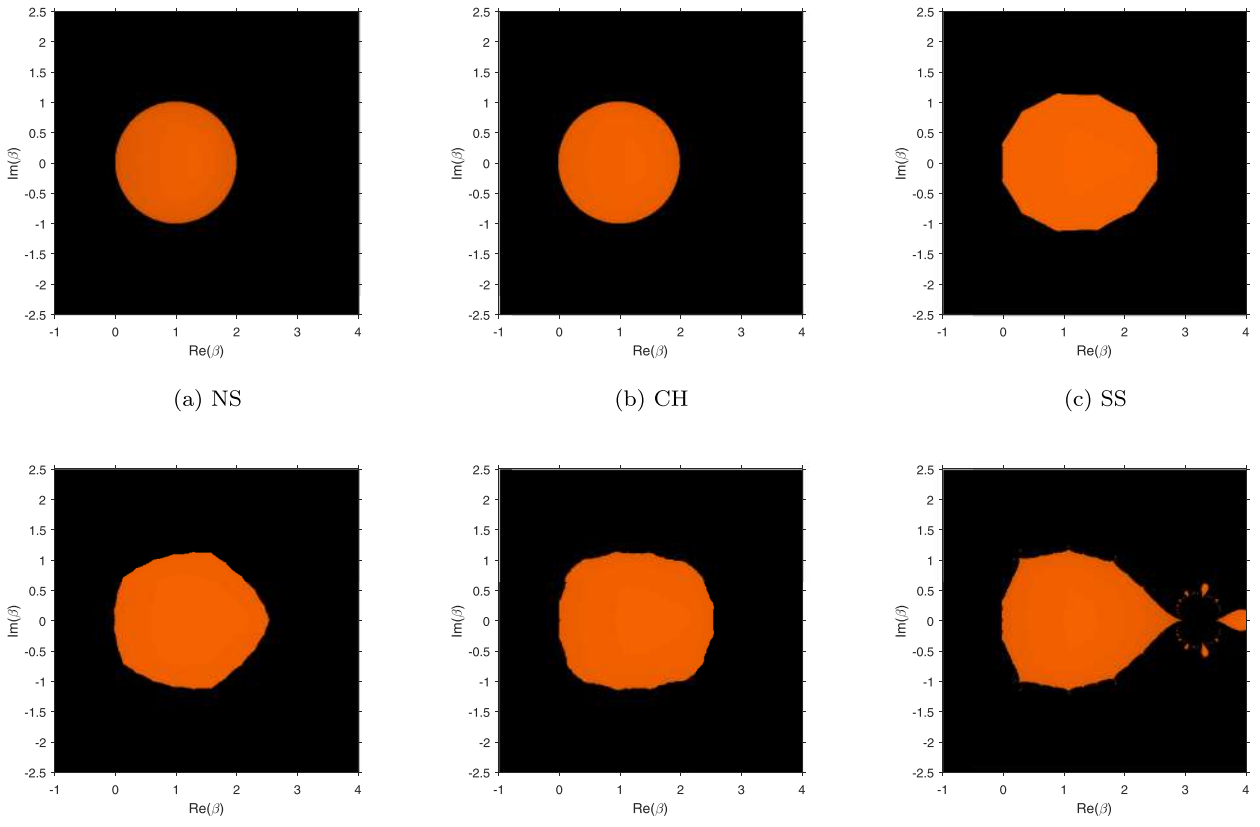
Now, let us to compare the performance of the methods with other kind of matrices by using command `gallery('leslie',3)` in Matlab, a 3×3 matrix appearing in population models,

$$A_2 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

By using this matrix and $X_0 = \beta \dfrac{A_2^T}{\|A_2\|^2}$, the set of converging initial values is painted in orange color for Newton-Schulz's method and also several second-order schemes of family corresponding to $p = 3$ and $\alpha_1 = 0$. Specifically, for $\alpha_2 = 0.3$, $\alpha_2 = 0.5$ $\alpha_2 = 0.7$ and $\alpha_2 = 0.8$ and $\alpha_3 = 1 - \alpha_2$. In Fig. 5 it is observed that the basin of attraction increases its wideness as well as parameter grows, being $\alpha_2 \leq 0.7$. Moreover, the set of converging values of β is connected meanwhile $\alpha_2 \leq 0.7$ and disconnected in other cases.

If the matrix is ill-conditioned, the role of the convergence and dependence on initial estimations is even more important. In Fig. 6, the inverse of a Hilbert matrix of size 3×3,

$$A_3 = \begin{pmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{pmatrix},$$



(a) NS (b) CH (c) SS

(d) TS (e) HM (f) MP

Fig. 2. Dynamical planes corresponding to $X_0 = \beta \frac{A_1^T}{\|A_1\|^2}$ on Toeplitz matrix A_1.

is estimated with $X_0 = \beta \frac{A_3^T}{\|A_3\|^2}$. Let us notice that the color of the basins of convergence is darker than in previous cases. This means that the convergence is slower, due to stability problems. However, each scheme has the same wideness and shape than in previous cases, were the conditioning of the involved matrices were good.

Now, pseudoinverse calculation is made for the low-rank matrix

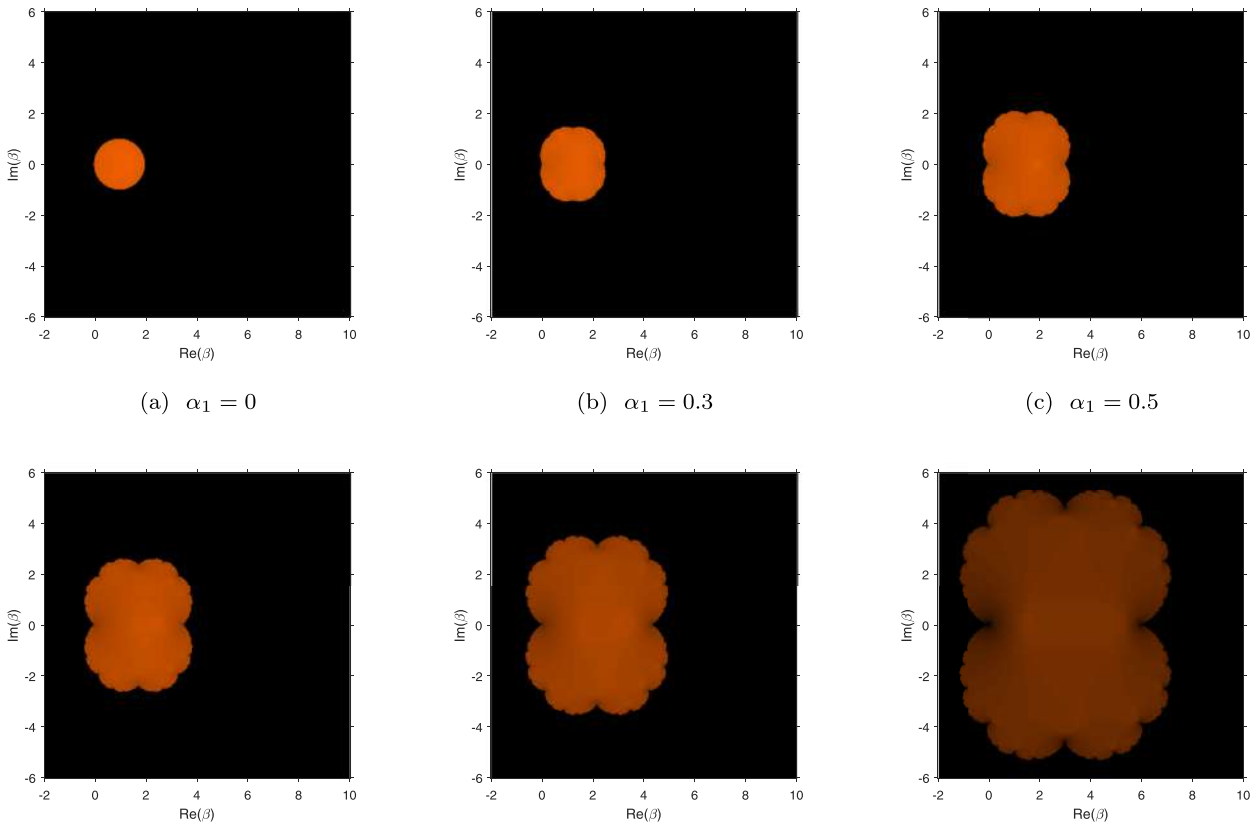
$$A_4 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & 1 & 0 & 1 \\ 2 & 4 & 6 & 8 \end{pmatrix}.$$

In Fig. 7, it can be observed that the performance of the iterative schemes for estimating the pseudoinverse matrix is the same as in previous non-singular square matrices, showing the best behavior the method $\alpha_2 = 0.7$, in terms of the size of the set of converging initial estimations. In case of first-order class of iterative methods, it has been also observed that the performance of the procedures (as far as it depends on the initial estimation) is the same as in the inverse calculations.

6. Numerical performance

In this section, we check the performance of the proposed schemes, in comparison with other known ones, on small and large–scale matrices. These came from academical problems and also from applied signal processing problems. Among the academical problems, we use the Hilbert matrix as example of ill-conditioned matrix. These numerical tests have been made with Matlab R2018b, by using double precision arithmetics. The convergence is checked by means of the stopping criterium of the residual, $\|AX_k - I\| < 10^{-6}$ and a maximum of 200 iterations. In all cases, the initial estimation taken is $X_0 = \beta \frac{A^T}{\|A\|^2}$, being A the matrix whose inverse we are estimating, as we have seen in the previous section that this estimation, with values of β close to 0.7, gives the best results.

In Tables 1, 2, 3, elements $\alpha_1 = 0$, $\alpha_1 = 0.6$ and $\alpha_1 = 0.8$ for the class $p = 2$ (all of them with $\alpha_2 = 1 - \alpha_1$) are compared with the members of class $p = 3$ corresponding to $\alpha_1 = 0$ and $\alpha_2 = 0$, $\alpha_2 = 0.6$ and $\alpha_2 = 0.8$, where $\alpha_3 = 1 - \alpha_2$. The



(a) $\alpha_1 = 0$ (b) $\alpha_1 = 0.3$ (c) $\alpha_1 = 0.5$

(d) $\alpha_1 = 0.6$ (e) $\alpha_1 = 0.7$ (f) $\alpha_1 = 0.8$

Fig. 3. Dynamical planes corresponding to first-order class with $\alpha_1 \in [0, 1[$ and $X_0 = \beta \frac{A_1^T}{\|A_1\|^2}$ on matrix A_1.

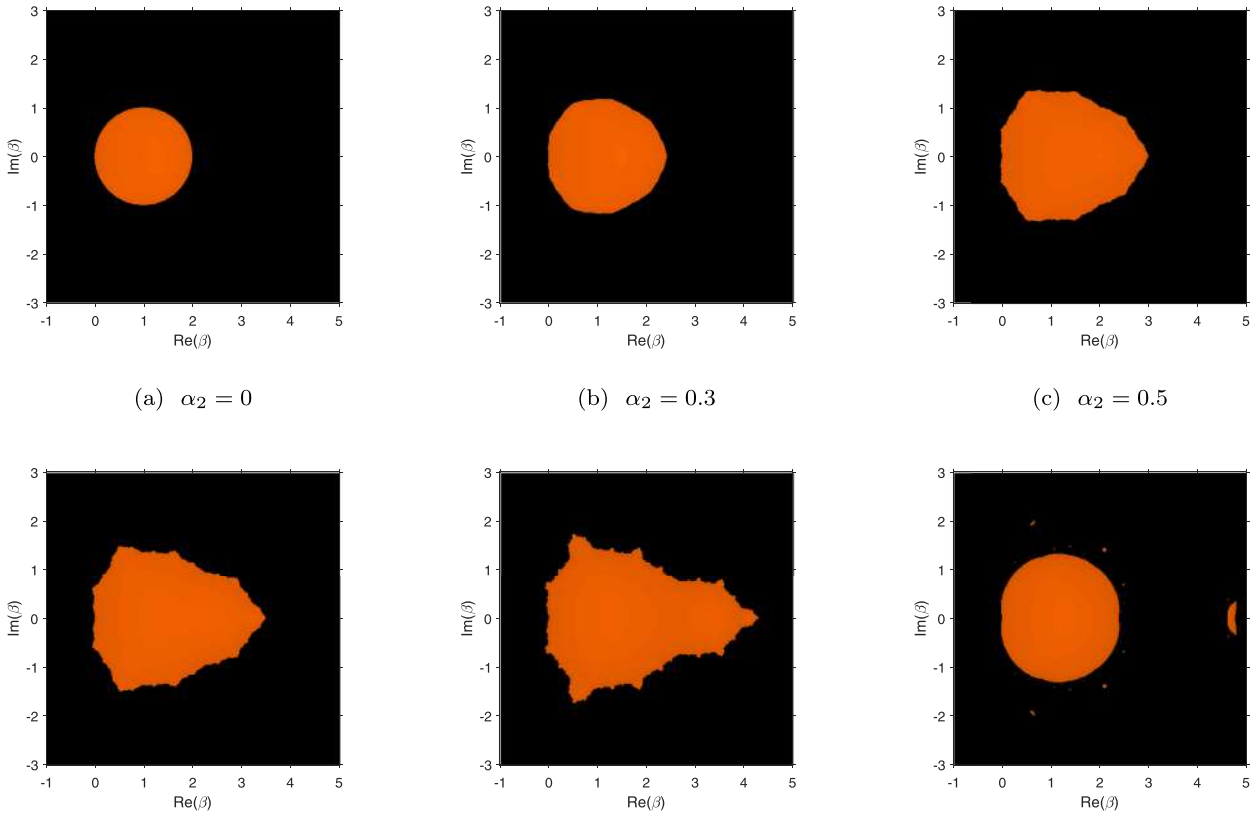
Table 1
Numerical results for Toeplitz matrix A_1.

β	$p=2$						$p=3$, $\alpha_1=0$					
	$\alpha_1=0$		$\alpha_1=0.6$		$\alpha_1=0.8$		$\alpha_2=0$		$\alpha_2=0.6$		$\alpha_2=0.8$	
	it	res	it	res	it	res	it	res	it	res	it	res
1	5	2.3e-10	28	9.3e-7	63	9.3e-7	3	7.5e-9	4	1.2e-7	5	2.5e-12
1.5	5	2.3e-10	26	6.8e-7	58	8.6e-7	3	7.5e-9	4	3.7e-10	4	2.2e-7
2	nc	-	25	6.9e-7	58	9.9e-7	nc	-	4	1.2e-7	6	1.5e-9
2.5	nc	-	24	8.1e-7	59	8.8e-7	nc	-	3	2.1e-8	nc	-
3	nc	-	28	9.4e-7	59	9.1e-7	nc	-	5	4.8e-8	nc	-
3.5	nc	-	nc	-	59	8.8e-7	nc	-	nc	-	nc	-
4	nc	-	nc	-	58	9.9e-7	nc	-	nc	-	nc	-
4.5	nc	-	nc	-	59	8.5e-7	nc	-	nc	-	nc	-
5	nc	-	nc	-	59	8.8e-7	nc	-	nc	-	nc	-
5.5	nc	-	nc	-	nc	-	nc	-	nc	-	nc	-
6	nc	-	nc	-	nc	-	nc	-	nc	-	nc	-

comparison is made through the number of iterations needed to converge (it) and the residual $\|AX_k - I\|$, denoted by (res). If the method does not converge (typically giving "NaN"), it is denoted by "nc" in the column of iterations; if the scheme simplify needs more than 200 iterations to converge, it is denoted by > 200.

Table 1 shows the results obtained for matrix A_1 introduced in the previous section. In Table 2 the numerical results correspond to a Leslie matrix of size 100×100, and in Table 3 the results generated by a Hilbert matrix of size 5×5 are shown.

From the results appearing in Table 1, it can be deduced that, among second-order schemes, the best convergence results are those corresponding to the method of class $p = 2$ with $\alpha_1 = 0$ and $\alpha_2 = 0.8$. It converges for a much wider set of initial estimations, although the number of iterations needed is high. In terms of the number of iterations needed to converge,



(a) $\alpha_2 = 0$ (b) $\alpha_2 = 0.3$ (c) $\alpha_2 = 0.5$

(d) $\alpha_2 = 0.6$ (e) $\alpha_2 = 0.7$ (f) $\alpha_2 = 0.8$

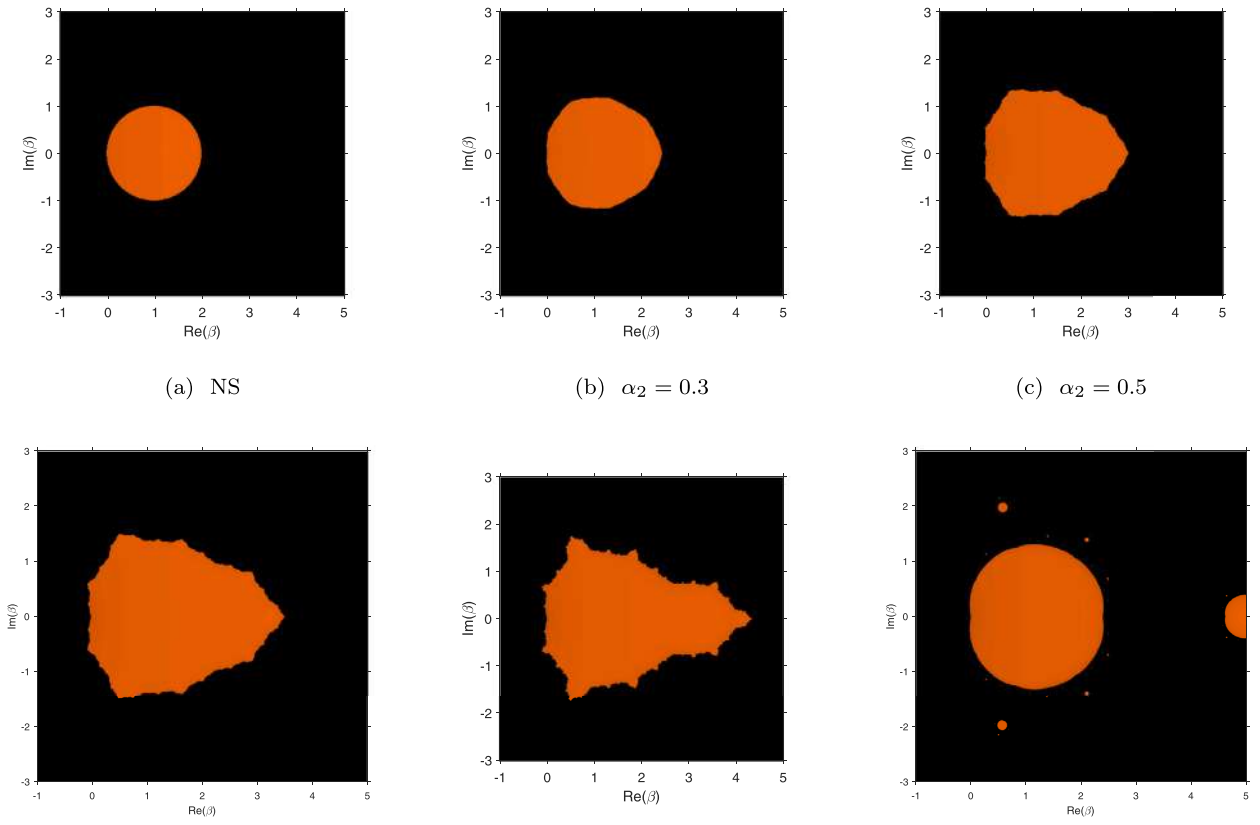
Fig. 4. Dynamical planes corresponding to second-order class ($p = 3$) with $\alpha_1 = 0$, $\alpha_2 \in [0, 1[$ and $X_0 = \beta \frac{A_1^T}{\|A_1\|^2}$ on matrix A_1.

Table 2
Numerical results for a Leslie matrix of size 100×100.

β	$p=2$						$p=3,\ \alpha_1=0$					
	$\alpha_1=0$		$\alpha_1=0.6$		$\alpha_1=0.8$		$\alpha_2=0$		$\alpha_2=0.6$		$\alpha_2=0.8$	
	it	res	it	res	it	res	it	res	it	res	it	res
1	18	6.9e-12	55	8.5e-7	113	9.2e-7	11	2.9e-8	14	6.4e-7	16	2.4e-10
1.5	17	4.2e-9	54	7.7e-7	111	8.8e-7	11	4.8e-12	14	2.4e-9	15	1.4e-7
2	53	3.7e-11	53	8.3e-7	107	9.6e-7	33	8.0e-11	14	1.3e-11	15	1.4e-9
2.5	nc	-	52	9.8e-7	108	9.2e-7	nc	-	13	3.9e-7	nc	-
3	nc	-	52	7.5e-7	107	9.2e-7	nc	-	13	3.7e-8	nc	-
3.5	nc	-	nc	-	106	9.5e-7	nc	-	26	1.1e-12	nc	-
4	nc	-	nc	-	106	8.1e-7	nc	-	nc	-	nc	-
4.5	nc	-	nc	-	105	8.7e-7	nc	-	nc	-	nc	-
5	nc	-	nc	-	104	9.6e-7	nc	-	nc	-	14	1.2e-12
5.5	nc	-	nc	-	104	8.5e-7	nc	-	nc	-	nc	-
6	nc	-	nc	-	> 200	-	nc	-	nc	-	nc	-

the best results are those provided by the element of family $p = 3$ with $\alpha_1 = 0$, $\alpha_2 = 0.6$ (and $\alpha_3 = 1 - \alpha_2$); the process converges in a very reduced number of iterations, with lower residual error than classical Newton-Schulz method. Regarding the results obtained by using classical Chebyshev's scheme, we remark that it does not improve the data from the best of proposed schemes, in sipe of having third-order of convergence.

In Table 2, we notice that for large–scale (100×100) Leslie matrix, the numerical results obtained by $p = 2$, $\alpha_1 = 0$ and $\alpha_2 = 0.6$ show convergence to the inverse matrix even when parameter β of the initial estimation is not close to zero. However, in these cases the number of iterations is very high. Regarding the lowest number of iterations needed to converge, the best method corresponds to $p = 3$, $\alpha_1 = 0$ and $\alpha_2 = 0.6$ as it holds low number of iterations and high value of β.



(a) NS (b) $\alpha_2 = 0.3$ (c) $\alpha_2 = 0.5$

(d) $\alpha_2 = 0.6$ (e) $\alpha_2 = 0.7$ (f) $\alpha_2 = 0.8$

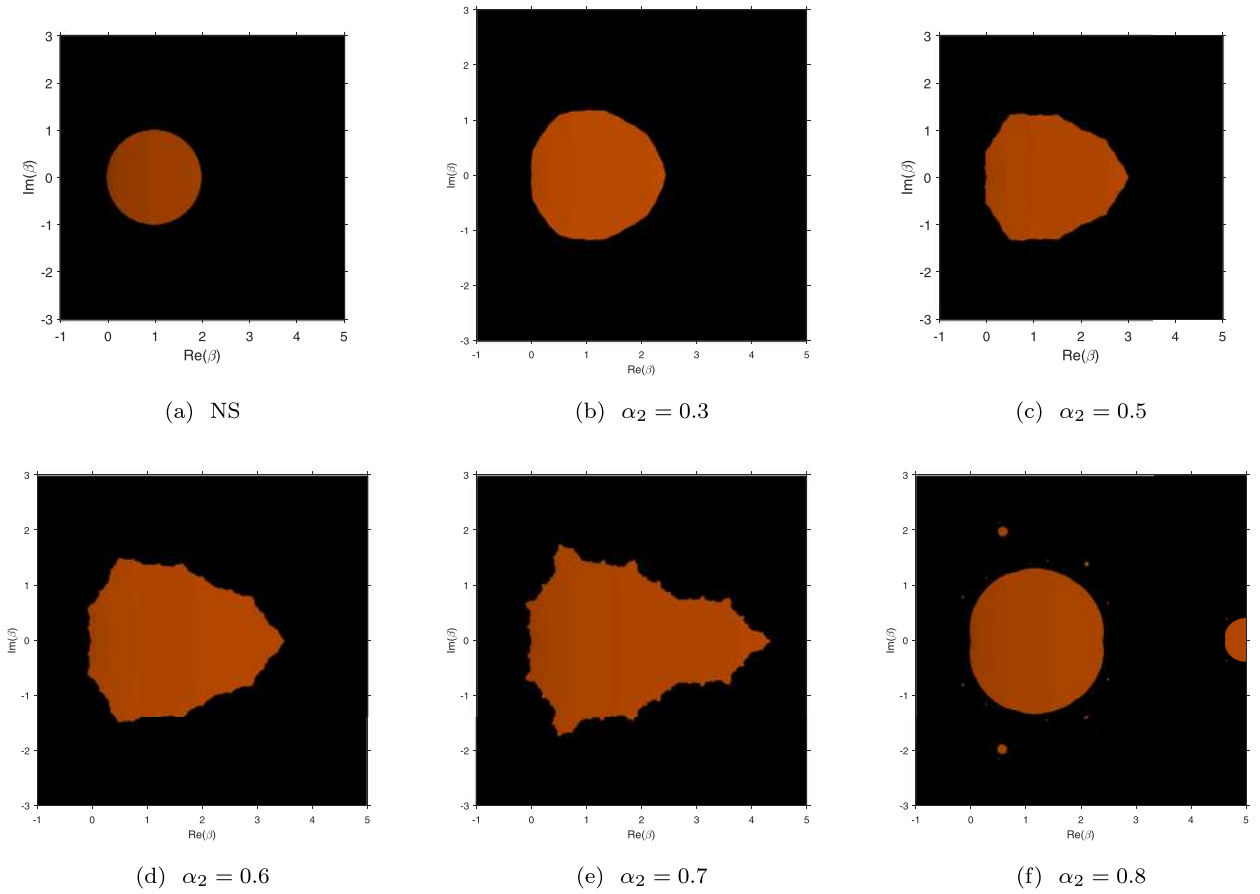
Fig. 5. Dynamical planes corresponding to some second-order methods with $X_0 = \beta \frac{A_2^T}{\|A_2\|^2}$ on Leslie matrix A_2.

Table 3
Numerical results for a Hilbert matrix of size 5×5.

β	$p=2$						$p=3$, $\alpha_1=0$					
	$\alpha_1=0$		$\alpha_1=0.2$		$\alpha_1=0.4$		$\alpha_2=0$		$\alpha_2=0.6$		$\alpha_2=0.8$	
	it	res	it	res	it	res	it	res	it	res	it	res
1	42	3.9e-9	54	5.7e-7	72	4.3e-7	27	2.3e-11	34	9.5e-11	37	1.07e-7
1.5	41	4.9e-7	53	9.3e-7	71	4.8e-7	26	5.1e-8	33	1.5e-7	37	9.8e-11
2	56	5.7e-8	53	4.2e-7	70	6.9e-7	nc	-	33	2.3e-9	36	3.9e-7
2.5	nc	-	nc	-	70	4.5e-7	nc	-	33	4.8e-11	nc	-
3	nc	-	nc	-	nc	-	nc	-	33	1.3e-11	nc	-
3.5	nc	-	nc	-	nc	-	nc	-	32	2.1e-8	nc	-
4	nc	-	nc	-	nc	-	nc	-	nc	-	nc	-
4.5	nc	-	nc	-	nc	-	nc	-	nc	-	nc	-
5	nc	-	nc	-	nc	-	nc	-	nc	-	nc	-
5.5	nc	-	nc	-	nc	-	nc	-	nc	-	nc	-
6	nc	-	nc	-	nc	-	nc	-	nc	-	nc	-

Table 3 corresponds to a test on a 5×5 Hilbert matrix. It is clear that the numer of iterations is high due to the bad conditioning of the matrix. Nevertheless, the performance is, in general similar to previous cases.

Now, we compare our proposed method of family $p = 3$, $\alpha_1 = \alpha_2 = 0$ and $\alpha_3 = 0.7$ (denoted by $PM3_{0.7}$ with other known ones, introduced in the previous section and denoted by SS, TS, HM and MP. In Table 4, we present the results obtained for these schemes on a Toeplitz matrix of size 100×100, by using the initial estimation $X_0 = \beta \frac{A^*}{\|A\|^2}$, tolerance 10^{-6} and a maximum of 200 iterations. We notice that our scheme gets convergence to the inverse matrix in a very similar execution time (e-t) than other methods with higher order of convergence and moreover, it converges with initial estimations with bigger values of β than the rest of procedures.



(a) NS (b) $\alpha_2 = 0.3$ (c) $\alpha_2 = 0.5$

(d) $\alpha_2 = 0.6$ (e) $\alpha_2 = 0.7$ (f) $\alpha_2 = 0.8$

Fig. 6. Dynamical planes corresponding to some second-order methods with $X_0 = \beta \frac{A_3^T}{\|A_3\|^2}$ on Hilbert matrix A_3.

Table 4
Numerical results for a Toeplitz matrix of size 100×100.

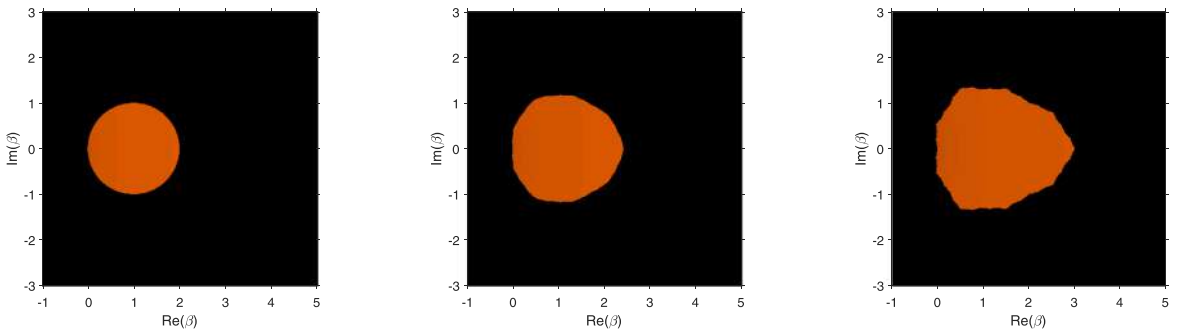
β	SS			$PM3_{0.7}$			TS			HM			MP		
	it	res	e-t	it	res	e-t	it	res	e-t	it	res	e-t	it	res	e-t
0.1	4	3.8e-14	0.028	10	3.1e-10	0.025	6	4.8e-15	0.044	7	1.4e-15	0.027	7	6.9e-12	0.022
0.5	3	4.4e-14	0.029	8	6.4e-10	0.026	4	4.2e-7	0.026	5	3.6e-8	0.028	6	1.1e-15	0.022
0.9	3	4.1e-14	0.031	7	2.6e-8	0.028	4	6.2e-12	0.034	5	1.7e-13	0.027	5	1.3e-10	0.031
1	3	3.9e-14	0.037	7	4.8e-9	0.032	4	3.9e-13	0.029	5	8.7e-15	0.038	5	1.1e-11	0.031
1.5	3	3.5e-14	0.029	7	1.1e-12	0.026	4	7.3e-15	0.032	4	1.8e-7	0.022	5	9.9e-16	0.026
2	2	2.7e-8	0.020	6	2.1e-8	0.028	3	8.6e-7	0.024	4	1.3e-9	0.027	4	5.6e-8	0.022
2.5	3	6.4e-14	0.024	6	3.3e-10	0.024	4	1.9e-10	0.027	5	1.1e-9	0.017	6	7.8e-15	0.028
3	nc	-	-	6	1.5e-11	0.027	nc	-	-	nc	-	-	nc	-	-
3.5	nc	-	-	6	1.4e-11	0.023	nc	-	-	nc	-	-	nc	-	-
4	nc	-	-	7	1.5e-11	0.028	nc	-	-	nc	-	-	nc	-	-
4.5	nc	-	-	nc	-	-	nc	-	-	nc	-	-	nc	-	-

The rest of the section is devoted to several test matrices for which we approximate its pseudoinverse. Table 5 includes the results of pseudoinverse calculation made for the low-rank matrix A_4 defined in Section 4. In this case, Chebyshev's method perfoms better than the most of methods under analysis, as it needs a very low number of iterations to converge to the psudo-inverse, although $p = 2$ can converge even with values of $\beta = 6$ or higher.

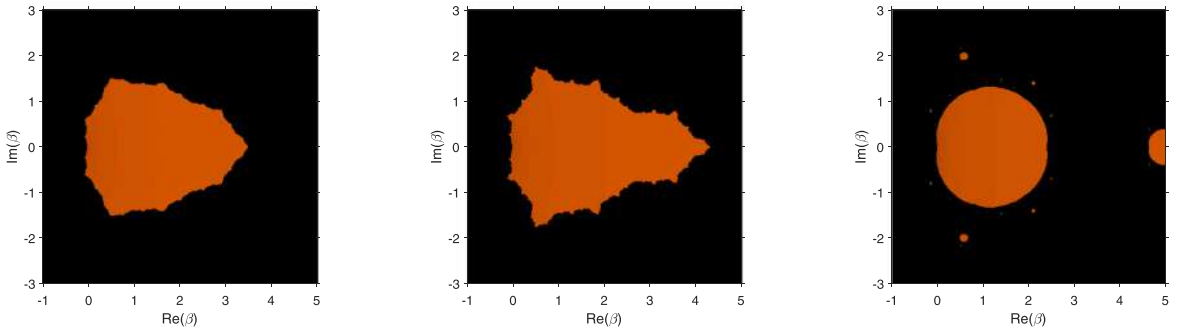
A similar performance is observed in Table 6, where we use a random matrix of size 300×200.

Finally, we present the last example corresponding to a random matrix with no full rank. Let us consider matrix

$$A = \begin{pmatrix} B & C \\ D & E \end{pmatrix} \in \mathbb{R}^{303\times 204},$$



(a) $\alpha_1 = 0$ (b) $\alpha_1 = 0.3$ (c) $\alpha_1 = 0.5$



(d) $\alpha_1 = 0.6$ (e) $\alpha_1 = 0.7$ (f) $\alpha_1 = 0.8$

Fig. 7. Dynamical planes corresponding to second-order class with $\alpha_1 \in [0, 1[$ and $X_0 = \beta \frac{A_1^T}{\|A_1\|^2}$ on matrix A_4.

Table 5
Numerical results for the estimation of the pseudo-inverse of A_4 with $X_0 = \beta \frac{A_4^T}{\|A_4\|^2}$.

β	$p=2$						$p=3,\ \alpha_1=0$					
	$\alpha_1=0$		$\alpha_1=0.6$		$\alpha_1=0.8$		$\alpha_2=0$		$\alpha_2=0.6$		$\alpha_2=0.8$	
	it	res	it	res	it	res	it	res	it	res	it	res
1	13	4.9e-10	43	6.9e-7	86	8.2e-7	11	7.6e-10	12	1.2e-10	9	2.9e-15
1.5	12	1.0e-7	42	6.2e-7	83	9.7e-7	11	2.0e-13	11	8.7e-8	8	3.5e-8
2	12	4.6e-10	41	6.7e-7	82	8.5e-7	10	1.3e-8	11	7.2e-10	8	1.1e-10
2.5	nc	-	40	7.9e-7	81	8.1e-7	10	3.5e-10	nc	-	nc	-
3	nc	-	39	9.9e-7	80	8.0e-7	10	1.0e-11	nc	-	nc	-
3.5	nc	-	39	7.9e-7	79	9.3e-7	10	6.2e-10	nc	-	nc	-
4	nc	-	nc	-	78	8.8e-7	nc	-	nc	-	nc	-
4.5	nc	-	nc	-	77	9.5e-7	nc	-	nc	-	nc	-
5	nc	-	nc	-	77	8.3e-7	nc	-	nc	-	nc	-
5.5	nc	-	nc	-	76	9.2e-7	nc	-	nc	-	nc	-
6	nc	-	nc	-	76	8.3e-7	nc	-	nc	-	nc	-

where B is a random matrix of size 300×200, $B = rand(300, 200)$, C is a matrix of zeros, $C = zeros(300, 4)$, D is a matrix of ones, $D = ones(3, 200)$ and E is the matrix

$$E = \begin{pmatrix} 1 & 1 & 0 & -1 \\ 2 & 0 & 0 & 0 \\ 3 & 1 & 0 & -1 \end{pmatrix}.$$

In this case, we use as stopping criterium $\|X_{k+1} - X_k\| < 10^{-6}$ and the last value of $\|X_{k+1} - X_k\|$ appears in the column called 'res'.

The results corresponding to this example are shown in Table 7. When this table is observed, we notice that the best global results are from MP and our proposed method $PM3_{0.7}$. Both schemes unify very wide areas of converging initial estimations with low execution times.

Table 6
Numerical results for the estimation of the pseudoinverse of a random matrix of size 300×200 with $X_0 = \beta \frac{A^T}{\|A\|^2}$.

β	$p=2$						$p=3,\ \alpha_1=0$					
	$\alpha_1=0$		$\alpha_1=0.6$		$\alpha_1=0.8$		$\alpha_2=0$		$\alpha_2=0.6$		$\alpha_2=0.8$	
	it	res	it	res	it	res	it	res	it	res	it	res
1	19	5.4e-8	56	6.7e-7	109	9.7e-7	13	4.9e-15	16	1.2e-9	18	5.7e-14
1.5	19	1.3e-11	55	6.0e-7	107	9.3e-7	12	4.3e-8	16	4.2e-13	17	3.8e-10
2	18	5.4e-8	54	6.5e-7	106	8.1e-7	12	1.5e-10	15	2.1e-8	17	6.8e-13
2.5	nc	-	53	7.7e-7	104	9.7e-7	nc	-	15	6.1e-10	nc	-
3	nc	-	52	9.7e-7	103	9.7e-7	nc	-	15	2.1e-11	nc	-
3.5	nc	-	52	7.7e-7	103	8.0e-7	nc	-	15	3.5e-8	nc	-
4	nc	-	nc	-	102	8.5e-7	nc	-	nc	-	nc	-
4.5	nc	-	nc	-	101	9.2e-7	nc	-	nc	-	nc	-
5	nc	-	nc	-	101	8.1e-7	nc	-	nc	-	nc	-
5.5	nc	-	nc	-	100	9.0e-7	nc	-	nc	-	nc	-
6	nc	-	nc	-	100	8.1e-7	nc	-	nc	-	nc	-

Table 7
Numerical results for non-square matrix without full rank of size 303×204.

β	SS			$PM3_{0.7}$			TS			HM			MP		
	it	res	e-t	it	res	e-t	it	res	e-t	it	res	e-t	it	res	e-t
0.1	8	3.6e-14	0.308	19	4.9e-7	0.266	11	8.4e-9	0.211	13	3.7e-8	0.240	14	3.9e-11	0.244
0.5	7	3.3e-14	0.266	17	9.7e-7	0.273	10	1.2e-9	0.245	12	6.0e-11	0.264	13	1.8e-15	0.236
0.9	7	3.3e-14	0.289	17	1.2e-10	0.266	10	4.1e-15	0.231	12	1.9e-15	0.238	12	1.1e-9	0.235
1	7	3.6e-14	0.276	17	1.5e-11	0.262	10	4.1e-15	0.242	11	6.4e-7	0.244	12	1.3e-10	0.248
1.5	7	3.2e-14	0.273	16	2.9e-8	0.266	9	7.9e-7	0.245	11	1.2e-9	0.240	12	3.7e-15	0.237
2	6	3.7e-7	0.265	16	2.4e-10	0.245	9	1.0e-8	0.248	11	3.1e-12	0.250	11	5.4e-7	0.240
2.5	6	1.1e-8	0.258	16	1.4e-12	0.258	9	1.5e-10	0.233	11	9.8e-5	0.242	11	1.9e-8	0.250
3	nc	-		15	2.0e-7	0.230	nc	-	-	nc	-	-	11	6.7e-10	0.264
3.5	nc	-		15	2.3e-8	0.231	nc		-	nc	-	-	24	6.7e-12	0.314
4	nc	-		15	2.8e-9	0.265	nc	-	-	nc	-	-	11	9.6e-13	0.228
4.5	nc	-		nc	-	-	nc	-	-	nc	-	-	nc	-	-

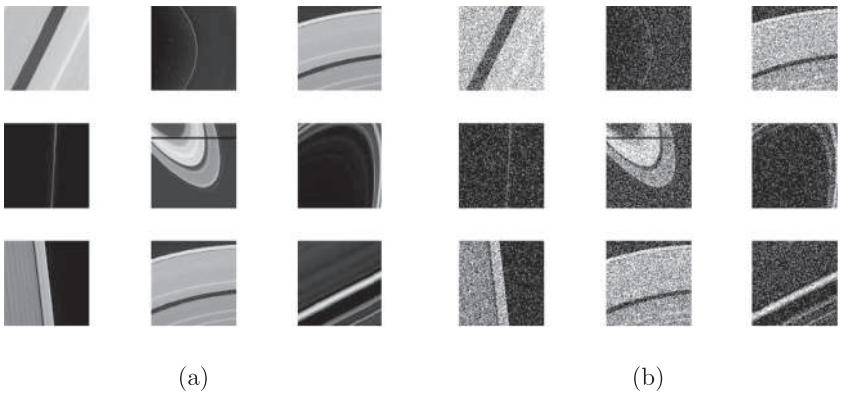


Fig. 8. (a) Some randomly selected images of Saturn. (b) Noisy versions of images in (a).

Now, we are going to apply the proposed schemes on a non-academic problem dealing with image processing.

6.1. Illustrative example: image denoising

This example illustrates an application of proposed method to an image denoising problem [31–33]. Specifically, we consider the problem of compression, filtering and decompression of a noisy image $\overline{C} \in \mathbb{R}^{90\times 90}$ estimation on the basis of the set of training images $\mathcal{A} = \left\{A^{(1)}, \ldots, A^{(s)}\right\}$, where $A^{(j)} \in \mathbb{R}^{90\times 90}$, for $j = 1, \ldots, s$. Our training set $\mathcal{A}$ consists of $s = 2000$ different satellite images from Saturn (see Fig. 8(a) for 9 sample images). Images in $\mathcal{A}$ were taken from the NASA Solar System Exploration Database [34].

It is assumed that instead of images in $\mathcal{A}$, we observed their noisy version $\mathcal{C} = \left\{C^{(1)}, \ldots, C^{(s)}\right\}$, where $C^{(j)} \in \mathbb{R}^{90\times 90}$, for $j = 1, \ldots, s$. Each $C^{(j)}$ is simulated as $C^{(j)} = A^{(j)} + 0.2N^{(j)}$, where $N^{(j)} \in \mathbb{R}^{90\times 90}$ is a random matrix generated from a normal distribution with zero-mean and standard deviation 1. For each image $A^{(j)}$, matrix $N^{(j)}$ simulates noise. (see Fig. 8(b) for 9

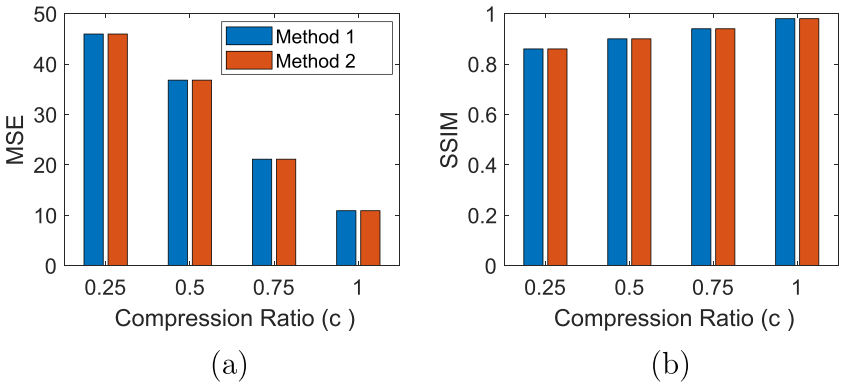


Fig. 9. Bar graphs of Method 1 and Method 2. Compression ratio c_r versus: (a) MSE and (b) SSIM between Methods 1 and 2.

sample images). It is assumed that noisy image $\overline{C}$ does not necessarily belong to $\mathcal{C}$, but it is "similar" to one of them, i.e., there is $C^{(j)} \in \mathcal{C}$ such that

$$C^{(j)} \in \arg\min_{C^{(i)} \in \mathcal{C}} \|C^{(i)} - \overline{C}\|_{fr} \leq \delta,$$

for a given $\delta \geq 0$.

Here, we vectorize matrices $A^{(j)}$ and $C^{(j)}$, i.e., convert each matrix into a column vector by stacking the columns of the matrix. Let $vect : \mathbb{R}^{m\times n} \to \mathbb{R}^{mn}$ be the vectorization transform. We write $a_j = vec(A^{(j)}) \in \mathbb{R}^{8100}$ and $c_j = vec(C^{(j)}) \in \mathbb{R}^{8100}$. If $A = [a_1\ a_2\ \ldots\ a_s] \in \mathbb{R}^{8100\times 2000}$ and $C = [c_1\ c_2\ \ldots\ c_s] \in \mathbb{R}^{8100\times 2000}$, the goal of the image denoising problem is find $\widehat{X} \in \mathbb{R}_r^{8100\times 8100}$ that gives a small reconstruction error for the training set, i.e., find $\widehat{X}$ such that

$$\|A - \widehat{X}C\|_{fr}^2 = \min_{X\in\mathbb{R}_r^{8100\times 8100}} \|A - XC\|_{fr}^2. \tag{8}$$

Here, $\mathbb{R}_r^{m\times n}$ denotes the set of all real $m \times n$ matrices of rank at most $r \leq \min\{m, n\}$. A solution of problem (8), given by Friedland and Torokhti [35], is

$$\widehat{X} = \lfloor AC^{\dagger}C \rfloor_r C^{\dagger}, \tag{9}$$

where $\lfloor M \rfloor_r$ is the r-truncated SVD of M. In this example, we use the proposed method to compute the pseudoinverse, with $C_0 = \frac{C^T}{\|C\|^2}$, $p = 2$, $\alpha_1 = 0.6$ and $\alpha_2 = 0.4$.

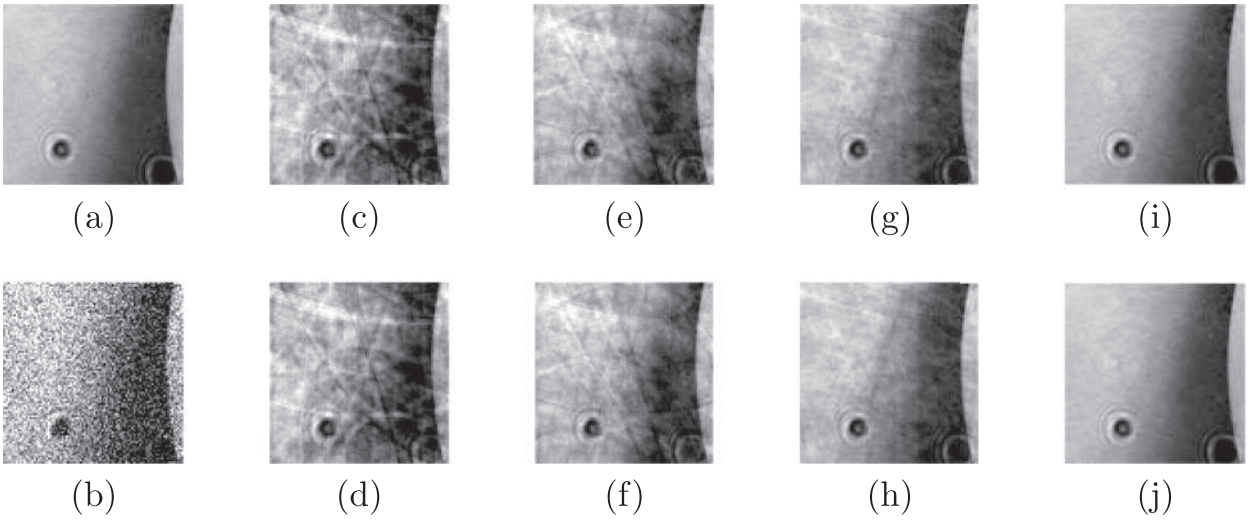


Fig. 10. Illustration of the estimation of noisy image $\overline{C}$ by Methods 1 and 2. (a) Source image. (b) Noisy observed image $\overline{C}$. (c)-(d) Estimation using Methods 1 and 2, respectively, for $c_r = 0.25$. (e)-(f) Estimation using Methods 1 and 2, respectively, for $c_r = 0.5$. (g)-(h) Estimation using Methods 1 and 2, respectively, for $c_r = 0.75$. (i)-(j) Estimation using Methods 1 and 2, respectively, for $c_r = 1$.

The compression ratio in problem (8) is given by $c_r = \frac{r}{\min\{p,q\}}$. Fig. 9 presents the execution MSE and structural similarity index[1] (SSIM) between Method 1, which compute (9) using Matlab command `pinv`, and Method 2, which compute (9) using proposed method, for compression ratio $c_r \in \{0.25, 0.5, 0.75, 1\}$, i.e., $r \in \{2025, 4050, 6075, 8100\}$. In Fig. 10, we show the estimates of $\overline{C}$ using both methods. The numerical results obtained in Figs. 9 and 10 clearly demonstrate the advantages of the proposed method. It is clear that both methods achieve the same MSE.

7. Conclusions

In this paper, we have developed a parametric family of iterative methods for computing inverse and pseudoinverse of a complex matrix, having arbitrary order of convergence. Moreover, we have shown in Theorems 1 and 2 that the order of the suggested method in (2) depends on the first non-zero parameter. Moreover, among all the procedures with the same order of convergence, those with the first non-zero parameter between 0.6 and 0.8 show the best performance, in terms of stability.

The proposed parametric family in (2) is a generalization of other methods which are obtained for particular values of the parameters. The numerical experiments show the feasibility and effectiveness of the new methods, for both nonsingular and rectangular matrices with or without full rank and arbitrary size. Indeed, the proposed methods have proven their applicability by solving image denoising problems with success.

References

[1] A. Torokhti, P. Soto-Quiros, Generalized brillinger-like transforms, IEEE Signal Process. Lett. 23 (6) (2016) 843–847.
[2] J. Chung, M. Chung, Computing optimal low-rank matrix approximations for image processing, in: 2013 Asilomar Conference on Signals, Systems and Computers, IEEE, 2013, pp. 670–674.
[3] S. Chountasis, V.N. Katsikis, D. Pappas, Applications of the Moore-Penrose inverse in digital image restoration, Math. Problems Eng. (2009). 2009 ID 170724
[4] S. Miljković, M. Miladinović, P. Stanimirović, I. Stojanović, Application of the pseudoinverse computation in reconstruction of blurred images, Filomat 26 (3) (2012) 453–465.
[5] J. Liu, H. Zhang, J. Jia, Cryptanalysis of schemes based on pseudoinverse matrix, Wuhan Univ. J. Natural Sci. 21 (3) (2016) 209–213.
[6] V.H. Dang, T.D. Nguyen, Construction of pseudoinverse matrix over finite field and its applications, Wirel. Pers. Commun. 94 (3) (2017) 455–466.
[7] C.-T. Nguyen, Y.-W. Tsai, Finite-time output feedback controller based on observer for the time-varying delayed systems: a Moore-Penrose inverse approach, Math. Problems Eng. (2017). 2017 ID 2808094
[8] U. Ansari, A.H. Bajodah, Robust launch vehicles generalized dynamic inversion attitude control, Aircraft Eng. Aerospace Tech. 89 (6) (2017) 902–910.
[9] P. Stanimirović, F. Soleymani, A class of numerical algorithms for computing outer matrices, Comput. Appl. Math. 263 (2014) 236–245.
[10] M. Petković, Generalized Schulz iterative methods for the computation of outer inverses, Comput. Math. Appl. 67 (10) (2014) 1837–1847.
[11] G. Schulz, Iterative berechnung der reziproken matrix, Z. Angew. Math. Mech. 13 (1933) 57–59.
[12] W. Li, Z. Li, A family of iterative methods for computing the approximate inverse of a square matrix and inner inverse of a non-square matrix, Appl. Math. Comput. 215 (9) (2010) 3433–3442.
[13] H. Chen, Y. Wang, A family of higher-order convergent iterative methods for computing the Moore–Penrose inverse, Appl. Math. Comput. 218 (8) (2011) 4012–4016.
[14] L. Weiguo, L. Juan, Q. Tiantian, A family of iterative methods for computing Moore-Penrose inverse of a matrix, Linear Algebra Appl. 438 (2013) 47–56.
[15] F. Toutounian, F. Soleymani, An iterative method for computing the approximate inverse of a square matrix and the Moore–Penrose inverse of a non-square matrix, Appl. Math. Comput. 224 (2013) 671–680.
[16] P. Stanimirović, A. Kumar, V. Katsikis, Further efficient hyperpower iterative methods for the computation of generalized inverses $A^{(2)}_{T,S}$, RACSAM 113 (2019) 3323–3339.
[17] M. Kaur, M. Kansal, S. Kumar, An efficient hyperpower iterative method for computating weighted Moore-Penrose inverse, AIMS Math. 5 (3) (2020) 1680–1692.
[18] R.A. Horn, C.R. Johnson, Matrix Analysis, Cambridge University Press, New York, 2012.
[19] F. Soleymani, P.S. Stanimirović, A higher order iterative method for computing the Drazin inverse, Sci. World (2013). 2013 ID 708647
[20] F. Soleymani, On a fast iterative method for approximate inverse of matrices, Commun. Korean Math. Soc. 28 (2) (2013) 407–418.
[21] F. Soleymani, P.S. Stanimirović, M.Z. Ullah, An accelerated iterative method for computing weighted Moore–Penrose inverse, Appl. Math. Comput. 222 (2013) 365–371.
[22] F. Soleymani, A fast convergent iterative solver for approximate inverse of matrices, Numer. Linear Algebra Appl. 21 (3) (2014) 439–452.
[23] F. Soleymani, H. Salmani, M. Rasouli, Finding the Moore–Penrose inverse by a new matrix iteration, J. Appl. Math. Comput. 47 (1–2) (2015) 33–48.
[24] M.K. Razavi, A. Kerayechian, M. Gachpazan, S. Shateyi, A new iterative method for finding approximate inverses of complex matrices, Abstract Appl. Anal. (2014). 2014 ID 563787
[25] A. Al-Fhaid, S. Shateyi, M. Ullah, F. Soleymani, A matrix iteration for finding Drazin inverse with ninth-order convergence, Abstract Appl. Anal. (2014). 2014 ID 137486
[26] F. Soleymani, A rapid numerical algorithm to compute matrix inversion, Int. J. Math. Math. Sci. (2012). 2012 ID 134653
[27] X. Liu, N. Cai, High-order iterative methods for the DMP inverse, J. Math. (2018). 2018 ID 8175935
[28] S. Srivastava, D. Gupta, A third order iterative method for $a^{\dagger}$, Int. J. Comput. Sci. Math. 4 (2) (2013) 140–151.
[29] S. Amat, S. Busquier, J. Gutiérrez, Geometric constructions of iterative functions to solve nonlinear equations, Comput. Appl. Math. 157 (1) (2003) 197–205.
[30] H. Li, T. Huang, Y. Zhang, X. Liu, T. Gu, Chebyshev–type methods and preconditioning techniques, Appl. Math. Comput. 218 (2011) 260–270.
[31] J. Chung, M. Chung, An efficient approach for computing optimal low-rank regularized inverse matrices, Inverse Probl. 30 (11) (2014) 114009.
[32] J. Chung, M. Chung, Computing optimal low-rank matrix approximations for image processing, in: 2013 Asilomar Conference on Signals, Systems and Computers, 2013, pp. 670–674.

[1] The structural similarity (SSIM) index is a method for predicting the perceived quality of digital images (and videos). The resultant SSIM index is a decimal value between -1 and 1, and value 1 is only reachable in the case of two identical sets of data. More details are available in [36].

[33] P. Soto-Quiros, A. Torokhti, Improvement in accuracy for dimensionality reduction and reconstruction of noisy signals. Part II: the case of signal samples, Signal Process. 154 (2019) 272–279.
[34] NASA, NASA solar system exploration database, (https://solarsystem.nasa.gov/raw-images/raw-image-viewer). Online; accessed 10 September 2020.
[35] S. Friedland, A. Torokhti, Generalized rank-constrained matrix approximations, SIAM J. Matrix Anal. Appl. 29 (2) (2007) 656–659.
[36] S. Channappayya, A. Bovik, R. Heath, Rate bounds on SSIM index of quantized images, IEEE Trans. Image Process. 17 (9) (2008) 1624–1639.

Anexo 3

Proceedings

of the

XXVI Congreso de Ecuaciones Diferenciales y Aplicaciones
XVI Congreso de Matemática Aplicada

Gijón (Asturias), Spain

June 14-18, 2021

SēMA
Sociedad Española
de Matemática Aplicada

Universidad de Oviedo

Editors:
Rafael Gallego, Mariano Mateos

Universidad de Oviedo
Servicio de Publicaciones de la Universidad de Oviedo
Campus de Humanidades. Edificio de Servicios. 33011 Oviedo (Asturias)
Tel. 985 10 95 03 Fax 985 10 95 07
http: www.uniovi.es/publicaciones
servipub@uniovi.es

ISBN: 978-84-18482-21-2

Foreword

It is with great pleasure that we present the Proceedings of the 26th Congress of Differential Equations and Applications / 16th Congress of Applied Mathematics (XXVI CEDYA / XVI CMA), the biennial congress of the Spanish Society of Applied Mathematics S$\vec{e}$MA, which is held in Gijón, Spain from June 14 to June 18, 2021.

In this volume we gather the short papers sent by some of the almost three hundred and twenty communications presented in the conference. Abstracts of all those communications can be found in the abstract book of the congress. Moreover, full papers by invited lecturers will shortly appear in a special issue of the S$\vec{e}$MA Journal.

The first CEDYA was celebrated in 1978 in Madrid, and the first joint CEDYA / CMA took place in Málaga in 1989. Our congress focuses on different fields of applied mathematics: Dynamical Systems and Ordinary Differential Equations, Partial Differential Equations, Numerical Analysis and Simulation, Numerical Linear Algebra, Optimal Control and Inverse Problems and Applications of Mathematics to Industry, Social Sciences, and Biology. Communications in other related topics such as Scientific Computation, Approximation Theory, Discrete Mathematics and Mathematical Education are also common.

For the last few editions, the congress has been structured in mini-symposia. In Gijón, we will have eighteen minis-symposia, proposed by different researchers and groups, and also five thematic sessions organized by the local organizing committee to distribute the individual contributions. We will also have a poster session and ten invited lectures. Among all the mini-symposia, we want to highlight the one dedicated to the memory of our colleague Francisco Javier "Pancho" Sayas, which gathers two plenary lectures, thirty-six talks, and more than forty invited people that have expressed their wish to pay tribute to his figure and work.

This edition has been deeply marked by the COVID-19 pandemic. First scheduled for June 2020, we had to postpone it one year, and move to a hybrid format. Roughly half of the participants attended the conference online, while the other half came to Gijón. Taking a normal conference and moving to a hybrid format in one year has meant a lot of efforts from all the parties involved. Not only did we, as organizing committee, see how much of the work already done had to be undone and redone in a different way, but also the administration staff, the scientific committee, the mini-symposia organizers, and many of the contributors had to work overtime for the change.

Just to name a few of the problems that all of us faced: some of the already accepted mini-symposia and contributed talks had to be withdrawn for different reasons (mainly because of the lack of flexibility of the funding agencies); it became quite clear since the very first moment that, no matter how well things evolved, it would be nearly impossible for most international participants to come to Gijón; reservations with the hotels and contracts with the suppliers had to be cancelled; and there was a lot of uncertainty, and even anxiety could be said, until we were able to confirm that the face-to-face part of the congress could take place as planned.

On the other hand, in the new open call for scientific proposals, we had a nice surprise: many people that would have not been able to participate in the original congress were sending new ideas for mini-symposia, individual contributions and posters. This meant that the total number of communications was about twenty percent greater than the original one, with most of the new contributions sent by students.

There were almost one hundred and twenty students registered for this CEDYA / CMA. The hybrid format allows students to participate at very low expense for their funding agencies, and this gives them the opportunity to attend different conferences and get more merits. But this, which can be seen as an advantage, makes it harder for them to obtain a full conference experience. Alfréd Rényi said: "a mathematician is a device for turning coffee into theorems". Experience has taught us that a congress is the best place for a mathematician to have a lot of coffee. And coffee cannot be served online.

In Gijón, June 4, 2021

The Local Organizing Committee from the Universidad de Oviedo

Scientific Committee

- Juan Luis Vázquez, Universidad Autónoma de Madrid
- María Paz Calvo, Universidad de Valladolid
- Laura Grigori, INRIA Paris
- José Antonio Langa, Universidad de Sevilla
- Mikel Lezaun, Euskal Herriko Unibersitatea
- Peter Monk, University of Delaware
- Ira Neitzel, Universität Bonn
- JoséÁngel Rodríguez, Universidad de Oviedo
- Fernando de Terán, Universidad Carlos III de Madrid

Sponsors

- Sociedad Española de Matemática Aplicada
- Departamento de Matemáticas de la Universidad de Oviedo
- Escuela Politécnica de Ingeniería de Gijón
- Gijón Convention Bureau
- Ayuntamiento de Gijón

Local Organizing Committee from the Universidad de Oviedo

- Pedro Alonso Velázquez
- Rafael Gallego
- Mariano Mateos
- Omar Menéndez
- Virginia Selgas
- Marisa Serrano
- Jesús Suárez Pérez del Río

Contents

CONTENTS

XXVI Congreso de Ecuaciones Diferenciales y Aplicaciones
XVI Congreso de Matemática Aplicada
Gijón, 14-18 junio 2021
(pp. 141–147)

Iterative processes with arbitrary order of convergence for approximating generalized inverses

Alicia Cordero[1], **Pablo Soto-Quiros**[2], **Juan R. Torregrosa**[1]
1. *Instituto de Matemática Multidisciplinar, Universitat Politècnica de València, València, Spain*
2. *Escuela de Matemática, Instituto Tecnológico de Costa Rica, Cartago, Costa Rica*

Abstract

A family of iterative schemes for finding approximate inverses of nonsingular matrices is suggested and established analytically. This class of methods can be used for finding the Moore-Penrose inverse of a rectangular complex matrix. The order of convergence is stated in each case, depending on the first non-zero parameter. For different examples, the accessibility of some schemes, that is, the set of initial estimations leading to convergence, is analyzed in order to select those with wider sets. This wideness is related with the value of the first non-zero value of the parameters defining the method. Finally, some numerical examples are provided to confirm the theoretical results and to show the feasibility and effectiveness of the new methods.

1. Introduction

Computing the matrix inverse of nonsingular matrices of higher size is difficult and is a time consuming task. Generally speaking, in wide variety of topics, one must compute the inverse or particularly the generalized inverses to comprehend and realize significant features of the involved problems.

In the last decade, many iterative schemes of different orders have been designed for approximating the inverse or some generalized inverse (Moore-Penrose inverse, Drazin inverse, etc.) of a complex matrix A. In this paper, we focus our attention on constructing a new class of iterative methods, free of inverse operators and with arbitrary order of convergence, for finding the inverse of a nonsingular complex matrix. We also study the proposed class for computing the Moore-Penrose inverse of complex rectangular matrices. The designed family depends on several real parameters, which by taking particular values provide us numerous known methods constructed by other authors with different procedures.

The most known iterative scheme for computing the inverse A^{-1} of a nonsingular complex matrix A is the Schulz's method whose iterative expression is

$$X_{k+1} = X_k(2I - AX_k), \quad k = 0, 1, \ldots \tag{1.1}$$

where I is the identity matrix with the same size of A. Schulz in [8] demonstrated the convergence of sequence $\{X_k\}_{k\geq 0}$, obtained from (1.1), to the inverse A^{-1} is guaranteed if the eigenvalues of matrix $I - AX_0$ are lower than 1. Taking into account that the residuals $E_k = I - AX_k$, $k = 0, 1, \ldots$ satisfy $\|E_{k+1}\| \leq \|E_k\|^2$, expression (1.1) has quadratic convergence. In general, in the Schulz-type methods it is common to use as initial approach $X_0 = \alpha A^*$ or $X_0 = \alpha A$, where $0 < \alpha < 2/\rho(A^*A)$, where A^* is the conjugate transpose of A and $\rho(\cdot)$ the spectral radius. In this paper, we use in the case of inverses and also in generalized inverses, the initial estimation $X_0 = \beta A^*/\|A\|^2$. We also study the values of the parameter β that guarantee convergence.

Li et al. in [5] proposed the family of iterative methods

$$X_{k+1} = X_k\left(\nu I - \frac{\nu(\nu-1)}{2}AX_k + \frac{\nu(\nu-1)(\nu-2)}{3!}(AX_k)^2 - \ldots + (-1)^{\nu-1}(AX_k)^{\nu-1}\right), \quad \nu = 2, 3, \ldots$$

with $X_0 = \alpha A^*$. They proved the convergence of ν-order of $\{X_k\}_{k\geq 0}$ to the inverse of matrix A. This class was used by Chen et al. in [2] and by Li et al. in [19] for approximating the Moore-Penrose inverse.

Soleymani et al. in [18] also constructed a fourth-order iterative scheme for calculating the inverse and the Moore-Penrose inverse, with iterative expression

$$X_{k+1} = \frac{1}{2}X_k(9I - AX_k(16I - AX_k(14I - AX_k(6I - AX_k)))), \quad k = 0, 1, \ldots$$

On the other hand, Stanimirović et al. in [16] designed the following scheme of order eleven for computing the generalized outer inverse $A^{(2)}_{T,S}$

$$X_{k+1} = X_k(I + (R_k + R_k^2)(I + (R_k^2 + R_k^4)(I + R_k^4))), \quad k = 0, 1, \ldots$$

being $R_k = I - AX_k$, $k = 0, 1, \ldots$.

Kaur et al. in [4], by using also the hyperpower iterative method, designed the following scheme of order five for obtaining the weighted Moore-Penrose inverse

$$X_{k+1} = X_k(5I - 10AX_k + 10(AX_k)^2 - 5(AX_k)^3 + (AX_k)^4), \quad k = 0, 1, \ldots$$

These papers are some of the manuscripts that have been published to approximate the inverse of a nonsingular matrix or some of the generalized inverses of arbitrary matrices. In this paper, we design a parametric family of iterative schemes with arbitrary order of convergence that contains many of the methods constructed up to date. For each fixed value of the order of convergence, we still have a class of iterative methods depending on several parameters.

The rest of this manuscript is organized as follows. Section 2 is devoted to the construction of the proposed class of iterative schemes, proving its convergence to the inverse of a nonsingular complex matrix, with arbitrary order of convergence. In Section 3, it is proven that the same family of iterative methods is able to converge to the Moore-Penrose inverse of a complex matrix of size $m \times n$. Some particular cases of this class are found in Section 4, corresponding to existing methods proposed by different authors. A wide range of numerical test are also found in Section 5, checking the robustness and applicability of the proposed methods on different kinds of matrices. With some conclusions and the references used finishes this manuscript.

2. A class of iterative schemes for matrix inversion

In this section, we present a parametric family of iterative schemes for approximating the inverse of nonsingular matrices and we prove the order of convergence of the different members of the family. First, we define the following polynomial matrix.

Definition 2.1 Let $U \in \mathbb{C}^{m\times m}$ be a complex square matrix and $p > 0$ a positive integer number. We define the polynomial matrix $H_p(U)$ as

$$H_p(U) = \sum_{j=1}^{p}(-1)^{j-1}C_p^j U^{j-1} = C_p^1 I - C_p^2 U + C_p^3 U^2 + \ldots + (-1)^{p-1}C_p^p U^{p-1},$$

where C_p^j is the combinatorial number $C_p^j = \binom{p}{j} = \dfrac{p!}{j!(p-j)!}$.

The following technical result can be proven by using mathematical induction with respect to parameter p.

Lemma 2.2 *Let $p > 0$ be a positive integer and $U \in \mathbb{C}^{m\times m}$. Then $(I - U)^p = I - UH_p(U)$.*

Let $A \in \mathbb{C}^{m\times m}$ be a nonsingular matrix and $p > 1$ a positive integer. Let $\{\alpha_1, \alpha_2, \ldots, \alpha_p\}$ be a set of real parameters such that $\alpha_i \in [0, 1]$, for $i = 1, 2, \ldots, p-1$, $\alpha_p \in]0, 1]$ and $\sum_{i=1}^{p}\alpha_i = 1$.

We assume a sequence of complex matrices $\{X_0, X_1, \ldots, X_n, \ldots\}$, of size $m \times m$, satisfying following conditions:

(a) $\|I - AX_0\| = \gamma_0 < 1$,

(b) $I - AX_{k+1} = \sum_{i=1}^{p}\alpha_i(I - AX_k)^i$.

We consider the family of methods with iterative expression

$$X_{k+1} = X_k \sum_{i=1}^{p}\alpha_i H_i(AX_k), \quad k = 0, 1, \ldots \tag{2.1}$$

For each positive integer p, $p > 1$, we have a different class of iterative methods, whose order of convergence depends on the value of parameters α_i, $i = 1, 2, \ldots, p$.

In the following results, the convergence of these schemes to the inverse of matrix A is proven.

Proposition 2.3 *Let $A \in \mathbb{C}^{m\times m}$ be a nonsingular matrix and $p > 1$ a positive integer. Let us consider the sequence of complex matrices constructed as*

$$X_{k+1} = X_k \sum_{i=1}^{p}\alpha_i H_i(AX_k), \quad k = 0, 1, \ldots,$$

where $\alpha_i \in [0,1]$, *for* $i = 1, 2, ..., p-1$, $\alpha_p \in]0,1]$ *and* $\sum_{i=1}^{p} \alpha_i = 1$. *Then, condition*

$$I - AX_{k+1} = \sum_{i=1}^{p} \alpha_i (I - AX_k)^i,$$

is equivalent to

$$X_{k+1} = X_k \sum_{i=1}^{p} \alpha_i \left(\sum_{j=1}^{i} (-1)^{j-1} C_i^j (AX_k)^{j-1} \right). \tag{2.2}$$

By mathematical induction it is also easy to prove the following result.

Proposition 2.4 *Let us consider sequence* $\{X_k\}_{k\geq 0}$ *obtained from expression* (2.1). *If* $\|I - AX_0\| < 1$, *then*

$$\|I - AX_k\| < 1, \quad k = 1, 2, \ldots$$

From these previous results, we can establish the following convergence theorem.

Theorem 2.5 *Let* $A \in \mathbb{C}^{m\times m}$ *be a nonsingular matrix and an initial guess* $X_0 \in \mathbb{C}^{m\times m}$. *Let* $\alpha_1, \ldots, \alpha_p$ *be nonnegative real numbers such that* $\alpha_i \in [0,1]$, $\alpha_p \neq 0$ *and* $\sum_{i=1}^{p} \alpha_i = 1$. *If* $\|I - AX_0\| < 1$, *then sequence* $\{X_k\}_{k\geq 0}$, *obtained by* (2.1), *converges to* A^{-1} *with convergence order* q *for any* $p > 1$, *where* $q = \min_{i=1,2,\ldots,p} \{i \mid \alpha_i \neq 0\}$.

In the next section, we extend the iterative schemes (2.1) for finding the Moore-Penrose inverse of any complex rectangular matrix.

3. A class of iterative schemes for computing Moore-Penrose inverse

Let A be a $m \times n$ complex matrix. The Moore-Penrose inverse of A (pseudoinverse), denoted by $A^\dagger$, is the unique $n \times m$ matrix X satisfying

$$AXA = A, \quad XAX = X, \quad (AX)^* = AX, \quad (XA)^* = XA.$$

This generalized inverse plays an important role in several fields, such as eigenvalue problems and the linear least square problems [3]. It can be obtained, explicitly, from the singular value decomposition of A but, with a high computational cost. Therefore, it is interesting to have efficient iterative methods to approximate this matrix. In this section, we prove how family (2.1) allows us to compute the pseudoinverse with the same order of convergence that in the previous section, where the inverse of a square matrix was calculated. First, we establish the following technical result, that is proven by mathematical induction, although other authors state similar results in the context of outer inverses (see, for example, [17]).

Lemma 3.1 *Let us consider* $X_0 = \alpha A^*$, *where* $\alpha \in \mathbb{R}$, *and sequence* $\{X_k\}_{k\geq 0}$ *generated by family* (2.1). *For any* $k \geq 0$, *it is satisfied*

$$(X_k A)^* = X_k A, \quad (AX_k)^* = AX_k, \quad X_k A A^\dagger = X_k, \quad A^\dagger A X_k = X_k. \tag{3.1}$$

Now, some technical results are presented.

Lemma 3.2 ([2]) *Let* $A \in \mathbb{C}^{m\times n}$ *and* $X_0 = \alpha A^*$ *be, where* $\alpha < \dfrac{1}{\sigma_1^2}$ *and* σ_1 *is the largest singular value of* A. *Then* $\|(X_0 - A^\dagger)A\| < 1$.

Lemma 3.3 *Let* $A \in \mathbb{C}^{m\times n}$ *and* $\{X_k\}_{k\geq 0}$ *be the sequence generated by* (2.1). *Let us consider* $E_k = X_k - A^\dagger$, $k = 0, 1, \ldots$. *Then,*

1. $\|X_k - A^\dagger\| \leq \|E_k A\| \|A^\dagger\|$,
2. $(I - A^\dagger A) E_k A = \mathbf{0}$.

Lemma 3.4 *Let $A \in \mathbb{C}^{m\times n}$ and $\{X_k\}_{k\geq 0}$ be the sequence generated by* (2.1). *By using $E_k = X_k - A^{\dagger}$, defined in the previous lemma, we have*

$$E_{k+1}A = \sum_{i=1}^{p} \alpha_i (-1)^{i-1} (E_k A)^i, \quad k = 0, 1, \ldots \tag{3.2}$$

Finally, we can state the following convergence result.

Theorem 3.5 *Let $A \in \mathbb{C}^{m\times n}$ and $q = \min_{i=1,2,\ldots,p} \{i \mid \alpha_i \neq 0\}$. Then, sequence $\{X_k\}_{k\geq 0}$ generated by* (2.1) *converges to the Moore-Penrose inverse $A^{\dagger}$ with qth-order provided that $X_0 = \alpha A^*$, where $\alpha < \frac{1}{\sigma_1^2}$ is a constant and σ_1 is the largest singular value of A.*

4. Some known members of the proposed class

The family of iterative schemes (2.1) is a generalization of other known methods constructed with different techniques. Now, we describe some of them.

1. For any $p > 1$, if $\alpha_1 = \cdots = \alpha_{p-1} = 0$ and $\alpha_p = 1$, then we obtain the method proposed by Li and Li. (see Eq. (2.3) in [5] for inverse case and Eq. (2.1) in [2] for pseudoinverse one). Recall that method proposed by Li and Li generalizes the Newton-Schultz ($p = 2$) and Chebyshev method ($p = 3$).

2. On the other hand, if $\alpha_i = 0$ for $i = 1, 2, \ldots, 8$, $\alpha_9 = \alpha_{12} = 1/8$ and $\alpha_{10} = \alpha_{11} = 3/8$, then we get the method proposed by Soleymani and Stanimirovic (see Eq. (12) in [9]).

3. Also, expression (2.1) gives us the method proposed by Toutounian and Soleymani (see Eq. (18) in [18]), if $\alpha_4 = 1/2$ and $\alpha_5 = 1/2$ and $\alpha_1 = \alpha_2 = \alpha_3 = 0$.

4. When the only not null parameter is $\alpha_7 = 1$, then we obtain method proposed by Soleymani (see Eq. (18) in [11]).

5. In a similar way, if the only parameter different from zero is $\alpha_6 = 1$, then the method proposed by Soleymani, Stanimirovic and Zaka (see Eq. (2.4) in [14]) is obtained.

6. When $\alpha_i = 0$ for $i = 1, 2, ...7$, $\alpha_8 = \alpha_{10} = 1/4$ and $\alpha_9 = 2/4$, the resulting scheme is that proposed by Soleymani in Eq. (9) in [12].

7. The method proposed by Soleymani et al in [13], Eq. (10), appears if $\alpha_1 = \cdots = \alpha_8 = 0$, $\alpha_9 = 7/9$ and $\alpha_{10} = 2/9$.

8. The scheme proposed by Razavi, Kerayechian, Gachpazan and Shateyi, (see Eq. (16) in [7]) is obtained if we choose $\alpha_1 = ... = \alpha_9 = 0$, $\alpha_{10} = \alpha_{12} = 1/4$ and $\alpha_{11} = 1/2$ in Equation (2.1).

9. When the first eight paramenters are null, $\alpha_9 = 343/729$, $\alpha_{10} = 294/729$, $\alpha_{11} = 84/729$ and $\alpha_{12} = 8/729$, we get the scheme proposed by Al-Fhaid et al in [1], Eq. (5).

10. If $\alpha_7 = 9/16$, $\alpha_8 = 6/16$, $\alpha_9 = 1/16$ and the rest of parameters are zero, then the scheme proposed by Soleymani is found (see Eq. (3.1) in [10]).

11. When $p = 12$ and the only parameters different from zero are $\alpha_9 = \alpha_{12} = 1/8$ and $\alpha_9 = \alpha_{10} = 3/8$, therefore the method proposed by Liu and Cai. see Eq. (4) in [6]) is obtained.

12. If $\alpha_2 = 0$, $\alpha_1 = 1 - \alpha$ and $\alpha_3 = \alpha$, where $\alpha \in (0, 1]$, then we find the method proposed by Srivastava and Gupta in [15]), Eq. (6).

5. Numerical examples

In this section, we check the performance of the proposed schemes, on small and large–scale matrices. Among them, we work with the Hilbert matrix as an example of ill-conditioned matrix. These numerical tests have been made with Matlab R2018b, by using double precision arithmetics. The convergence is checked by means of the stopping criterium of the residual, $\|AX_k - I\| < 10^{-6}$ and a maximum of 200 iterations. In all cases, the initial estimation taken is $X_0 = \beta \frac{A^T}{\|A\|^2}$, being A the matrix whose inverse we are estimating and choosing values of parameter β close to 0.7.

In Tables 1 - 2, elements $\alpha_1 = 0$, $\alpha_1 = 0.6$ and $\alpha_1 = 0.8$ for the class $p = 2$ (all of them with $\alpha_2 = 1 - \alpha_1$) are compared with the members of class $p = 3$ corresponding to $\alpha_1 = 0$ and $\alpha_2 = 0$, $\alpha_2 = 0.6$ and $\alpha_2 = 0.8$, where $\alpha_3 = 1 - \alpha_2$. The comparison is made through the number of iterations needed to converge (it) and the residual $\|AX_k - I\|$, denoted by (res). If the method does not converge (typically giving "NaN"), it is denoted by "nc" in the column of iterations; if the scheme simplify needs more than 200 iterations to converge, it is denoted by > 200.

In Table 1 the numerical results correspond to a Leslie matrix of size 100×100, and in Table 2 the results generated by a Hilbert matrix of size 5×5 are shown.

β	$p = 2$						$p = 3, \alpha_1 = 0$					
	$\alpha_1 = 0$		$\alpha_1 = 0.6$		$\alpha_1 = 0.8$		$\alpha_2 = 0$		$\alpha_2 = 0.6$		$\alpha_2 = 0.8$	
	it	res	it	res	it	res	it	res	it	res	it	res
1	18	6.9e-12	55	8.5e-7	113	9.2e-7	11	2.9e-8	14	6.4e-7	16	2.4e-10
1.5	17	4.2e-9	54	7.7e-7	111	8.8e-7	11	4.8e-12	14	2.4e-9	15	1.4e-7
2	53	3.7e-11	53	8.3e-7	107	9.6e-7	33	8.0e-11	14	1.3e-11	15	1.4e-9
2.5	nc	-	52	9.8e-7	108	9.2e-7	nc	-	13	3.9e-7	nc	-
3	nc	-	52	7.5e-7	107	9.2e-7	nc	-	13	3.7e-8	nc	-
3.5	nc	-	nc	-	106	9.5e-7	nc	-	26	1.1e-12	nc	-
4	nc	-	nc	-	106	8.1e-7	nc	-	nc	-	nc	-
4.5	nc	-	nc	-	105	8.7e-7	nc	-	nc	-	nc	-
5	nc	-	nc	-	104	9.6e-7	nc	-	nc	-	14	1.2e-12
5.5	nc	-	nc	-	104	8.5e-7	nc	-	nc	-	nc	-
6	nc	-	nc	-	> 200	-	nc	-	nc	-	nc	-

Tab. 1 Numerical results for a Leslie matrix of size 100×100

In Table 1, we notice that for large–scale (100×100) Leslie matrix, the numerical results obtained by $p = 2$, $\alpha_1 = 0$ and $\alpha_2 = 0.6$ show convergence to the inverse matrix even when parameter β of the initial estimation is not close to zero. However, in these cases the number of iterations is very high. Regarding the lowest number of iterations needed to converge, the best method corresponds to $p = 3$, $\alpha_1 = 0$ and $\alpha_2 = 0.6$ as it holds low number of iterations and high value of β.

Table 2 corresponds to a test on a 5×5 Hilbert matrix. It is clear that the number of iterations is high due to the bad conditioning of the matrix. Nevertheless, the performance is, in general similar to previous cases.

β	$p = 2$						$p = 3, \alpha_1 = 0$					
	$\alpha_1 = 0$		$\alpha_1 = 0.2$		$\alpha_1 = 0.4$		$\alpha_2 = 0$		$\alpha_2 = 0.6$		$\alpha_2 = 0.8$	
	it	res	it	res	it	res	it	res	it	res	it	res
1	42	3.9e-9	54	5.7e-7	72	4.3e-7	27	2.3e-11	34	9.5e-11	37	1.07e-7
1.5	41	4.9e-7	53	9.3e-7	71	4.8e-7	26	5.1e-8	33	1.5e-7	37	9.8e-11
2	56	5.7e-8	53	4.2e-7	70	6.9e-7	nc	-	33	2.3e-9	36	3.9e-7
2.5	nc	-	nc	-	70	4.5e-7	nc	-	33	4.8e-11	nc	-
3	nc	-	nc	-	nc	-	nc	-	33	1.3e-11	nc	-
3.5	nc	-	nc	-	nc	-	nc	-	32	2.1e-8	nc	-
4	nc	-	nc	-	nc	-	nc	-	nc	-	nc	-
4.5	nc	-	nc	-	nc	-	nc	-	nc	-	nc	-
5	nc	-	nc	-	nc	-	nc	-	nc	-	nc	-
5.5	nc	-	nc	-	nc	-	nc	-	nc	-	nc	-
6	nc	-	nc	-	nc	-	nc	-	nc	-	nc	-

Tab. 2 Numerical results for a Hilbert matrix of size 5×5

Finally, Table includes the results of pseudoinverse calculation for a random matrix of size 300×200. In this case, Chebyshev's method performs better than the most of schemes under study, as it need a very low number of iterations to converge to the pseudoinverse, although $p = 2$ can converge even with values of $\beta = 6$ or higher.

6. Conclusions

In this paper, we have developed a parametric family of iterative methods for computing inverse and pseudoinverse of a complex matrix, having arbitrary order of convergence. Moreover, we have shown in Theorems 2.5 and 3.5 that the order of the suggested method in (2.1) depends on the first non-zero parameter. The proposed parametric

β	$p=2$						$p=3, \alpha_1=0$					
	$\alpha_1=0$		$\alpha_1=0.6$		$\alpha_1=0.8$		$\alpha_2=0$		$\alpha_2=0.6$		$\alpha_2=0.8$	
	it	res	it	res	it	res	it	res	it	res	it	res
1	19	5.4e-8	56	6.7e-7	109	9.7e-7	13	4.9e-15	16	1.2e-9	18	5.7e-14
1.5	19	1.3e-11	55	6.0e-7	107	9.3e-7	12	4.3e-8	16	4.2e-13	17	3.8e-10
2	18	5.4e-8	54	6.5e-7	106	8.1e-7	12	1.5e-10	15	2.1e-8	17	6.8e-13
2.5	nc	-	53	7.7e-7	104	9.7e-7	nc	-	15	6.1e-10	nc	-
3	nc	-	52	9.7e-7	103	9.7e-7	nc	-	15	2.1e-11	nc	-
3.5	nc	-	52	7.7e-7	103	8.0e-7	nc	-	15	3.5e-8	nc	-
4	nc	-	nc	-	102	8.5e-7	nc	-	nc	-	nc	-
4.5	nc	-	nc	-	101	9.2e-7	nc	-	nc	-	nc	-
5	nc	-	nc	-	101	8.1e-7	nc	-	nc	-	nc	-
5.5	nc	-	nc	-	100	9.0e-7	nc	-	nc	-	nc	-
6	nc	-	nc	-	100	8.1e-7	nc	-	nc	-	nc	-

Tab. 3 Numerical results for the estimation of the pseudoinverse of a random matrix of size 300×200 with $X_0 = \beta \dfrac{A^T}{\|A\|^2}$

family in (2.1) is a generalization of other methods which are obtained for particular values of the parameters. The numerical experiments show the feasibility and effectiveness of the new methods, for both nonsingular and rectangular matrices with or without full rank and arbitrary size.

Acknowledgements

This research was supported in part by PGC2018-095896-B-C22 (MCIU/AEI/FEDER, UE) and in part by VIE from *Instituto Tecnológico de Costa Rica* (Research #1440037).

References

[1] AS Al-Fhaid, S Shateyi, M Ullah, and F Soleymani. A matrix iteration for finding Drazin inverse with ninth-order convergence. *Abstract Appl. Anal.*, 2014 ID 137486, 2014.

[2] Haibin Chen and Yiju Wang. A family of higher-order convergent iterative methods for computing the Moore–Penrose inverse. *Appl. Math. Comput.*, 218(8):4012–4016, 2011.

[3] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge University Press, New York, 2012.

[4] M. Kaur, M. Kansal, and S. Kumar. An efficient hyperpower iterative method for computating weighted Moore-Penrose inverse. *AIMS Mathematics*, 5(3):1680–1692, 2020.

[5] Weiguo Li and Zhi Li. A family of iterative methods for computing the approximate inverse of a square matrix and inner inverse of a non-square matrix. *Appl. Math. Comput.*, 215(9):3433–3442, 2010.

[6] Xiaoji Liu and Naping Cai. High-order iterative methods for the DMP inverse. *J. Math.*, 2018 ID 8175935, 2018.

[7] M Kafaei Razavi, Asghar Kerayechian, Mortaza Gachpazan, and Stanford Shateyi. A new iterative method for finding approximate inverses of complex matrices. *Abstract Appl. Anal.*, 2014 ID 563787, 2014.

[8] G. Schulz. Iterative berechnung der reziproken matrix. *Z. Angew. Math. Mech.*, 13:57–59, 1933.

[9] F Soleymani and P S Stanimirović. A higher order iterative method for computing the Drazin inverse. *Sci. World*, 2013 ID 708647, 2013.

[10] Fazlollah Soleymani. A rapid numerical algorithm to compute matrix inversion. *Int. J. Math. Math. Sci.*, 2012 ID 134653, 2012.

[11] Fazlollah Soleymani. On a fast iterative method for approximate inverse of matrices. *Communications of the Korean Mathematical Society*, 28(2):407–418, 2013.

[12] Fazlollah Soleymani. A fast convergent iterative solver for approximate inverse of matrices. *Numer. Linear Algebra Appl.*, 21(3):439–452, 2014.

[13] Fazlollah Soleymani, Hossein Salmani, and Masoud Rasouli. Finding the Moore–Penrose inverse by a new matrix iteration. *J. Appl. Math. Comput.*, 47(1-2):33–48, 2015.

[14] Fazlollah Soleymani, Predrag S Stanimirović, and Malik Zaka Ullah. An accelerated iterative method for computing weighted Moore–Penrose inverse. *Appl. Math. Comput.*, 222:365–371, 2013.

[15] Shwetabh Srivastava and DK Gupta. A third order iterative method for $a^{\dagger}$. *Int. J. Comput. Sci. Math.*, 4(2):140–151, 2013.

[16] P.S. Stanimirović, A. Kumar, and V.W. Katsikis. Further efficient hyperpower iterative methods for the computation of generalized inverses $A^{(2)}_{T,S}$. *RACSAM*, 113:3323–3339, 2019.

[17] P.S. Stanimirović and F. Soleymani. A class of numerical algorithms for computing outer matrices. *Comput. Appl. Math.*, 263:236–245, 2014.

[18] F Toutounian and F Soleymani. An iterative method for computing the approximate inverse of a square matrix and the Moore–Penrose inverse of a non-square matrix. *Appl. Math. Comput.*, 224:671–680, 2013.

[19] L Weiguo, L Juan, and Q Tiantian. A family of iterative methods for computating Moore-Penrose inverse of a matrix. *Linear Algebra Appl.*, 438:47–56, 2013.

Anexo 4

A Fast Algorithm for Image Deconvolution Based on a Rank Constrained Inverse Matrix Approximation Problem

Pablo Soto-Quiros, Juan Jose Fallas-Monge, and Jeffry Chavarría-Molina

Abstract In this paper, we present a fast method for image deconvolution, which is based on the rank constrained inverse matrix approximation (RCIMA) problem. The RCIMA problem is a general case of the low-rank approximation problem proposed by Eckart-Young. This new algorithm, so-called the fast-RCIMA method, is based on tensor product and Tikhonov's regularization to approximate the pseudoinverse and bilateral random projections to estimate the rank constrained approximation. The fast-RCIMA method reduces the execution time to estimate optimal solution and preserves the same accuracy of classical methods. We use training data as a substitute for knowledge of a forward model. Numerical simulations on measuring execution time and speedup confirmed the efficiency of the proposed method.

Keywords Rank constrained · Pseudoinverse · Fast algorithm · Speedup · Image deconvolution

1 Introduction

Image deconvolution is an image processing technique designed to remove blur or enhance contrast and resolution [1, 2]. If $\overline{X} \in \mathbb{R}^{m \times n}$ represents an digital image, then the mathematical model in image deconvolution can be written as

$$Ax + \eta = c, \tag{1}$$

P. Soto-Quiros (✉) · J. Jose Fallas-Monge · J. Chavarría-Molina
Escuela de Matemática, Instituto Tecnológico de Costa Rica, Cartago 30101, Costa Rica
e-mail: jusoto@tec.ac.cr

J. Jose Fallas-Monge
e-mail: jfallas@tec.ac.cr

J. Chavarría-Molina
e-mail: jchavarria@tec.ac.cr

X.-S. Yang et al. (eds.), *Proceedings of Sixth International Congress on Information and Communication Technology*, Lecture Notes in Networks and Systems 236,
https://doi.org/10.1007/978-981-16-2380-6_15

where $x \in \mathbb{R}^{mn}$ is a vectorized version of $\overline{X}$ (i.e., convert $\overline{X}$ into a column by stacking its columns), $A \in \mathbb{R}^{mn \times mn}$ is a matrix that models the forward process, $\eta \in \mathbb{R}^{mn}$ is an additive noise, and $c \in \mathbb{R}^{mn}$ is a vectorized version of the noisy image $\overline{C} \in \mathbb{R}^{m \times n}$. Given c and A, the goal of the inverse problem is to reconstruct x.

Entries in A depend on the point spread functions (PSFs) of the imaging system, and under some assumptions, matrix A may be highly structured so that an efficient algorithms can be used in the inverse problem to reconstruct x [1]. However, matrix A is unknown in most case, and only observed noisy image c is available. To solve this problem, training data is used as substitute for knowledge of the forward model by introducing a rank constrained inverse matrix problem that avoids including A in the problem formulation [2].

Let $\mathbb{R}_r^{p \times q}$ be the set of all real $p \times q$ matrices of rank at most $r \leq \min\{p, q\}$. If $\{x^{(k)}\}_{k=1}^S$ and $\{c^{(k)}\}_{k=1}^S$ are training data of vectorized images x and c, respectively, then the goal of the rank constrained inverse matrix problem is to find $\widehat{\boldsymbol{F}} \in \mathbb{R}_r^{mn \times mn}$ that gives a small reconstruction error for the given training set, i.e., find $\widehat{\boldsymbol{F}}$ such that

$$\widehat{\boldsymbol{F}} = \underset{\boldsymbol{F} \in \mathbb{R}_r^{mn \times mn}}{\arg\min} \frac{1}{S} \sum_{k=1}^{S} \| \boldsymbol{F} c^{(k)} - x^k \|_2^2, \tag{2}$$

where $\| \cdot \|_2$ is the Euclidean norm. Using relationships between Euclidean norm and Frobenius norm (i.e., $\| \cdot \|_{fr}$), problem (2) can be reformulated as follows:

$$\widehat{\boldsymbol{F}} = \underset{\boldsymbol{F} \in \mathbb{R}_r^{mn \times mn}}{\arg\min} \frac{1}{S} \| \boldsymbol{F} C - X \|_{fr}^2, \tag{3}$$

where X and C are created using training data, i.e., $X = [x^{(1)}\ x^{(2)}\ \ldots\ x^{(S)}] \in \mathbb{R}^{mn \times S}$ and $C = [c^{(1)}\ c^{(2)}\ \ldots\ c^{(S)}] \in \mathbb{R}^{mn \times S}$. Once matrix $\widehat{\boldsymbol{F}}$ is computed, a matrix-vector multiplication is required to solve the problem, i.e., $x = \widehat{\boldsymbol{F}} c$.

Problem (3) is so-called the rank constrained inverse matrix approximation (RCIMA) problem, which is a generalization of the well know low-rank approximation problem proposed by Eckart-Young [3]. In [2], Chung and Chung present a solution of the RCIMA problem, which uses the SVD to estimate the pseudoinverse and the low-rank approximation. The SVD method is very accurate but is critically blocked by computational complexity that makes it impractical in the case of large matrices [4, 5].

In this paper, we propose a new and efficient method to compute a solution of the RCIMA problem (3), so-called the fast-RCIMA method. The approach of this new method uses fast algorithms to approximate the pseudoinverse and low-rank matrix. Most specifically, the fast pseudoinverse technique used in the fast-RCIMA method utilizes a special type of tensor product of two vectors [7] and Tikhonov's regularization [8, 9] (see Sect. 3.1 below for further details). Besides, a bilateral

random projection and its power-scheme modification [10, 11] are used in this paper to estimate a low-rank approximation. Moreover, in this paper we proposed a generalization of the method developed in [11] to estimate a low-rank approximation, which includes analysis of particular case $\text{rank}(Y_2) < r$, where $Y_2 \in \mathbb{R}^{m\times r}$ is a right random projection (see Sect. 3.2 below for further details). The proposed implementation to estimate a solution of the RCIMA problem reduces the execution time to compute $\widehat{\boldsymbol{F}}$ in (3) and, moreover, preserves the same accuracy of classical method developed in [2].

In this work, we choose methods already mentioned briefly in [7, 8, 10, 11] for their easy computational implementation and their high-speed and efficiency to calculate the pseudoinverse and low-rank approximation. However, there are other fast algorithms to estimate the pseudoinverse and low-rank approximation (see, e.g., [6, 12–15]).

Throughout this paper, we use the following notation: $M^{\dagger}$ denotes the pseudoinvserse of M, $I_m \in \mathbb{R}^{m\times m}$ is the identity matrix, and $M(1:i,:)$ and $M(:,1:j)$ are formed with the first i rows and j columns, respectively, of M.

The paper is organized as follows. Section 2 presents a solution of the RCIMA problem based on the SVD method given in [2]. Fast algorithms to estimate the pseudoinverse and low-rank approximation, respectively, are explained in Sect. 3. In Sect. 4, the fast-GRLMA method is proposed to approximate a solution to problem (3). A numerical example based on image deconvolution is presented in Sect. 5. Finally, Sect. 6 contains some concluding remarks.

2 RCIMA Method

A solution of problem in (3) is proposed by Chung and Chung [2]. Let $C = U\Sigma V^T$ be the singular value decomposition (SVD) of C. If $k = \text{rank}(C)$ and $P = XV_kV_k^T$, where $V_k = V(:,1:k)$, then a solution of the RCIMA problem is given by

$$\widehat{\boldsymbol{F}} = P_r C^{\dagger}, \tag{4}$$

where P_r is the optimal low-rank approximation of P, such that $\text{rank}(P_r) \leq r$ (see Sect. 3.2 below for more details). Matrix P is known as the kernel of the RCIMA problem. The associated implementation of (4) is presented below in Algorithm 1. This implementation is so-called the RCIMA method.

Algorithm 1: RCIMA method

Input : Training Data $\{x^{(k)}\}_{k=1}^{S} \subseteq \mathbb{R}^{mn}$, $\{c^{(k)}\}_{k=1}^{S} \subseteq \mathbb{R}^{mn}$ and $r \leq mn$
Output: $\widehat{\boldsymbol{F}} \in \mathbb{R}_{r}^{mn \times mn}$

$X = [x^{(1)}\ x^{(2)}\ ...\ x^{(S)}]$
$C = [c^{(1)}\ c^{(2)}\ ...\ c^{(S)}]$
$[\sim, \sim, V] = \text{svd}(C)$
$k = \text{rank}(C)$
$V_k = V(:, 1:k)$
$P = X V_k V_k^T$
$[\widetilde{U}, \widetilde{S}, \widetilde{V}] = \text{svd}(P)$
$\widehat{\boldsymbol{F}} = \widetilde{U}(:, 1:r)\widetilde{S}(1:r, 1:r)[\widetilde{V}(:, 1:r)]^T C^{\dagger}$

3 Fast Algorithm to Compute Pseudoinverse and Low-Rank Approximation

As mentioned in the above Sect. 2, the RCIMA method uses the SVD to compute pseudoinverses and a low-rank approximation, respectively. However, the SVD method is usually very expensive for high-dimensional matrices. Therefore, in this section, we show two fast algorithms to compute pseudoinverse and low-rank approximation.

3.1 Pseudoinverse and Tensor Product Matrix

The pseudoinverse of $Z \in \mathbb{R}^{m \times n}$, denoted bv y $Z^{\dagger} \in \mathbb{R}^{n \times m}$, is the unique matrix satisfying the conditions $ZZ^{\dagger}Z = Z$, $Z^{\dagger}ZZ^{\dagger} = Z^{\dagger}$, $\left(Z^{\dagger}Z\right)^T = Z^{\dagger}Z$ and $\left(ZZ^{\dagger}\right)^T = ZZ^{\dagger}$. If $Z = U\Sigma V^T$ is the SVD of Z and $k = \text{rank}(Z)$, then the standard procedure to calculate $Z^{\dagger}$ is as follows:

$$Z^{\dagger} = U(:, 1:k)[\Sigma(1:k, 1:k)]^{-1}[V(:, 1:k)]^T. \tag{5}$$

Katsikis and Pappas [7] provides a fast and reliable algorithm in order to compute the pseudoinverse of full-rank matrices, which does not use the SVD method. This algorithm is based on a special type of tensor product of two vectors, that is usually used in infinite dimensional Hilbert spaces. The method to compute the pseudoinverse of a full-rank matrix Z in [7] is defined by

$$Z^{\dagger} = \begin{cases} (Z^T Z)^{-1} Z^T & \text{if } m \geq n \\ Z^T (ZZ^T)^{-1} & \text{if } m < n \end{cases}. \tag{6}$$

In 2008, Lu et al. [8] extended the method in [7] for computing the pseudoinverse of rank deficient matrices, using an approximation of Tikhonov's regularization in order to estimate $Z^{\dagger}$. If Z is a rank deficient matrix, then the method proposed in [8] to approximate $Z^{\dagger}$ is defined by

$$Z^{\dagger} \approx Z_P = \begin{cases} (Z^T Z + \alpha I_n)^{-1} Z^T & \text{if } m \geq n \\ Z^T (Z Z^T + \alpha I_m)^{-1} & \text{if } m < n \end{cases}, \tag{7}$$

where α is a positive number close to zero. Formula (7) is usefull to approximate $Z^{\dagger}$ because, if $\alpha \to 0$ in (7), then $Z^{\dagger} = Z_P$ (see Theorem 4.3 in [9]).

The method to estimate the pseudoinverse given by (6)–(7) is so-called the tensor product matrix (TPM) method.

3.2 Low-Rank Approximation and Bilateral Random Projection

Given $L \in \mathbb{R}^{m \times n}$ and $r \leq \min\{m, n\}$, the low-rank approximation problem is a minimization problem where the goal is to find $\widehat{L}_r \in \mathbb{R}_r^{m \times n}$ such that

$$\|L - \widehat{L}_r\|_{fr}^2 = \min_{L_r \in \mathbb{R}_r^{m \times n}} \|L - L_r\|_{fr}^2. \tag{8}$$

If $L = U \Sigma V^T$ is the SVD of L, then $\widehat{L}_r$ is given by

$$\widehat{L}_r = U(:, 1:r)\Sigma(1:r, 1:r)[V(:, 1:r)]^T. \tag{9}$$

An alternative method to estimate $\widehat{L}_r$ was developed by Fazel et al. [10]. These authors show that a fast method to estimate $\widehat{L}_r$ is

$$\widetilde{L}_r = Y_1 (A_2^T Y_1)^{\dagger} Y_2^T, \tag{10}$$

where

$$Y_1 = L A_1, \qquad Y_2 = L^T A_2 \tag{11}$$

are r-bilateral random projections (BRP) of L. Here, $A_1 \in \mathbb{R}^{n \times r}$ and $A_2 \in \mathbb{R}^{m \times r}$ are arbitrary full-rank matrices. Matrices Y_1 and Y_2 are called the left and right random projections of L, respectively. Comparing with randomized SVD in [14] that extracts the column space from unilateral random projections, the BRP method estimates both column and row spaces from bilateral random projections [11].

If L is a dense matrix, then number of flops to compute $\widetilde{L}_r$ is less than that of the SVD based approximation [11]. However, if the singular values of L decay gradually, then the accuracy of (10) may be lost. To solve this problem, Zhou and Tao [11] propose a power-scheme to improve the approximation precision, when $A_2^T Y_1$ is nonsingular. In this paper, we extend the method consider in [11] for computing the low-rank matrix, when $A_2^T Y_1$ is singular. The revised method considers a power-scheme given by left and right random projection of $X = (LL^T)^c L$ and L, respectively, i.e.,

$$Y_1 = XA_1 = (LL^T)^c LA_1, \qquad Y_2 = L^T A_2, \tag{12}$$

where c is a nonnegative integer and $A_1 \in \mathbb{R}^{n\times r}$, $A_2 \in \mathbb{R}^{m\times r}$ are arbitrary full-rank matrices. This power-scheme is useful because $(LL^T)^c LA_1$ has the same singular vectors as LA_1, while its singular values decay more rapidly [14]. Besides, note that left random projection Y_1 in (12) can be represented as $Y_1 = L\widetilde{A}_1$, where $\widetilde{A}_1 = L^T(LL^T)^{c-1}A_1$, i.e., formula (12) is a particular case of (11).

Theorem 1 below shows a new alternative formula to estimate a low-rank approximation of L, using a particular choice of A_2.

Theorem 1 *Consider random projections given by* (12)*, where* $A_1 \in \mathbb{R}^{n\times r}$ *is a arbitrary full-rank matrix and* $A_2 = L(L^T L)^{c-1}A_1$*. Then, estimation of low-rank approximation of* L *in* (10) *can be simplified to*

$$\widetilde{L}_r = LY_2Y_2^{\dagger}. \tag{13}$$

Proof Note that $(LL^T)^c L = L(L^T L)^c$, and therefore, left projection in (12) can be expressed as $Y_1 = LY_2$. Further, from Proposition 3.2 in [9], $\left(Y_2^T Y_2\right)^{\dagger} Y_2^T = Y_2^{\dagger}$. Then

$$\begin{aligned}
\widetilde{L}_r &= Y_1 \left(A_2^T Y_1\right)^{\dagger} Y_2^T \\
&= LY_2 \left[\left(L(L^T L)^{c-1}A_1\right)^T LY_2\right]^{\dagger} Y_2^T \\
&= LY_2 \left[\left(\left(L^T L\right)^c A_1\right)^T Y_2\right]^{\dagger} Y_2^T \\
&= LY_2 \left[Y_2^T Y_2\right]^{\dagger} Y_2^T \\
&= LY_2Y_2^{\dagger}.
\end{aligned}$$

Remark 1 $Y_2^{\dagger}$ in (13) can be computed by the TPM method explained in Sect. 3.2. Note that we only consider the case when number of rows is greater than or equal to number of columns, because $n \geq r$.

The method to estimate the low-rank approximation of L, given by (13) and the TPM method is so-called the modified bilateral random projections (MBRP) method. The pseudocode of the MBRP method is presented in Algorithm 2. Note that bilateral random projections $Y_1 = LA_1$ and $Y_2 = L^T A_2 = (L^T L)^c A_1$ can be computed efficiently using a loop in steps 1–4 of Algorithm 2.

Algorithm 2: MBRP Method for Computing Low-Rank Approximation

Input : $L \in \mathbb{R}^{m\times n}$, $r \leq \min\{m, n\}$, $A_1 \in \mathbb{R}^{m\times r}$ and $c \in \{1, 2, 3, ...\}$
Output: $\widetilde{L}_r \in \mathbb{R}_r^{m\times n}$

1 $Y_2 = A_1$
2 **for** $i = 1 : c$ **do**
3 $\quad Y_1 = LY_2$
4 $\quad Y_2 = L^T Y_1$
5 **if** Y_2 is full rank **then**
6 $\quad \widetilde{L}_r = LY_2(Y_2^T Y_2)^{-1} Y_2^T$
7 **else if** Y_2 is rank deficient **then**
8 $\quad \alpha \in]0, 1[$
9 $\quad \widetilde{L}_r = LY_2(Y_2^T Y_2 + \alpha I_n)^{-1} Y_2^T$
10

Remark 2 The Power-scheme is useful to improve accuracy of estimation of low-rank approximation. However, if MBRP method is executed in floating-point arithmetic and c is a large number, then all information associated with singular values smaller than $\mu^{1/(2c+1)}\|L\|_{fr}$ is lost, where μ is the machine precision [14]. Therefore, it is recommended to take a small value for c. Thus, numerical simulations in this paper are computed using $c = 3$.

4 Fast-RCIMA Method

In this section, we propose a new implementation to compute a solution of the RCIMA problem on the basis of the TPM and MBRP methods explained in Sects. 3.1 and 3.2, respectively. A block diagram of this implementation, called the fast-RCIMA method, is presented in Fig. 1, which is explained below:

Step 0: Obtain training data $\{x\}_{k=1}^S$ and $\{c\}_{k=1}^S$.

Step 1: Create matrices $X = [x^{(1)}\ x^{(2)}\ ...\ x^{(S)}]$ and $C = [c^{(1)}\ c^{(2)}\ ...\ c^{(S)}]$.
Step 2: Compute pseudoinverse C_P of C, using the TPM method.
Step 3: Compute the kernel matrix $P = XV_kV_k^T$, where $C = U\Sigma V^T$ is the SVD of C, $V_k = V(:, 1:k)$ and $k = \text{rank}(C)$. To avoid to compute the SVD of C, we use the fact that $V_kV_k^T = C^\dagger C$ (see, e.g., equation (2139) in [16]). Therefore, kernel matrix P can be compute as $P = XC_PC$.

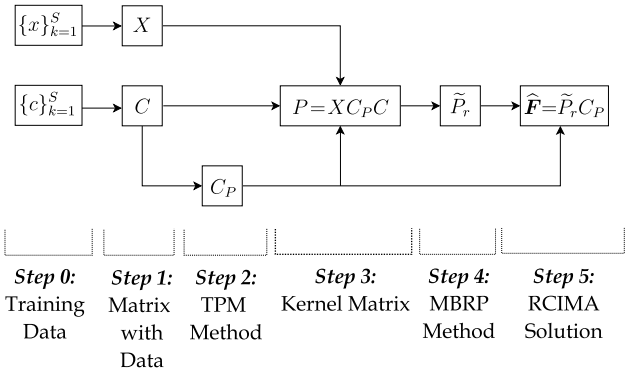


Fig. 1 Block diagram of the fast-RCIMA method

Step 4: Estimate low-rank approximation $\widetilde{P}_r$ of P, using the MBRP method.
Step 5: Approximate the RCIMA solution $\widehat{\boldsymbol{F}} = \widetilde{P}_r C_P$.

The associate pseudocode of the fast-RCIMA method is shown in Algorithm 3, where $c = 3$ is used at the power-scheme.

5 Numerical Simulation

In this section, we show a numerical simulation of the proposed fast-RCIMA method for image deconvolution. The numerical examples were run on a desktop computer with a 2.30 GHz processor (Intel(R) Core(TM) i7-4712MQ CPU) and a 8.00 RAM.

5.1 Speedup and Percent Difference

Consider two methods that solve the same problem, Method 1 and Method 2, with execution times T_1 and T_2, respectively. The computational performance analysis of the fast-RCIMA method is evaluated using the following two metrics:

- The *speedup* S (or acceleration) is the ratio between the execution times of both methods, i.e., $S = T_2/T_1$. If Method 1 is an improvement of Method 2, then speedup will be grater than 1. However, if Method 1 hurts performance, speedup will be less than 1 (see [17] for more details).
- The *percent difference* P between Method 1 and Method 2, where $T_1 < T_2$, is represented by $P = 100(T_2 - T_1)/T_2$. Then, we say that Method 1 is $P\%$ faster than Method 2 (see [17] for more details).

Algorithm 3: Fast-RCIMA Method

Input : Training Data $\{x^{(k)}\}_{k=1}^{S} \subseteq \mathbb{R}^{mn}$, $\{c^{(k)}\}_{k=1}^{S} \subseteq \mathbb{R}^{mn}$, $r \leq mn$ and $A_1 \in \mathbb{R}^{S\times r}$
Output: $\widehat{F} \in \mathbb{R}_r^{mn\times mn}$

/* Matrix Representation of Training Data */
$X = [x^{(1)}\ x^{(2)}\ ...\ x^{(S)}]$
$C = [c^{(1)}\ c^{(2)}\ ...\ c^{(S)}]$
/* Pseudoinverse of C with TPM Method */
if C is full rank **then**
  if $mn \geq S$, **then** $C_P = (C^T C)^{-1} C^T$
  if $mn < S$, **then** $C_P = C^T (CC^T)^{-1}$
else if C is rank deficient **then**
  $\alpha \in]0, 1[$
  if $mn \geq S$, **then** $C_P = (C^T C + \alpha I_S)^{-1} Z^T$
  if $mn < S$, **then** $C_P = C^T (CC^T + \alpha I_{mn})^{-1}$
/* Kernel Matrix P */
$P = XC_P C$
/* Low-Rank Matrix Approximation of P with MBRP Method */
$Y_2 = A_1$
for $i = 1 : 3$ **do**
  $Y_1 = PY_2$
  $Y_2 = P^T Y_1$
if Y_2 is full rank **then**
  $\widetilde{P}_r = PY_2 (Y_2^T Y_2)^{-1} Y_2^T$
else if Y_2 is rank deficient **then**
  $\alpha \in]0, 1[$
  $\widetilde{P}_r = PY_2 (Y_2^T Y_2 + \alpha I_r)^{-1} Y_2^T$
/* Rank Constrained Inverse Matrix Approximation */
$\widehat{F} = \widetilde{P}_r C_P$

5.2 *Numerical Example for Image Deconvolution*

This example illustrates the application of the RCIMA method to image deconvolution using training data [2, 18], which was briefly mentioned in Sect. 1. Specifically, we apply RCIMA and fast-RCIMA methods to the problem of filtering a noisy image $\overline{C} \in \mathbb{R}^{90\times 90}$ estimation on the basis of the set of a training images $\mathcal{X} = \left\{X^{(1)}, ..., X^{(S)}\right\}$, where $X^{(j)} \in \mathbb{R}^{90\times 90}$, for $j = 1, ..., S$. Our training set $\mathcal{X}$ consists of $s = 2000$ different satellite images taken from the NASA Solar System Exploration Database [19] (see Fig. 2a for 6 sample images).

Let $vect : \mathbb{R}^{m\times n} \to \mathbb{R}^{mn}$ be the vectorization transform. We write $x^{(j)} = vec(X^{(j)}) \in \mathbb{R}^{8100}$, for $j = 1, ..., 2000$. Instead of images in $\mathcal{X}$, we observed their noisy version $\mathcal{C} = \left\{C^{(1)}, ..., C^{(s)}\right\}$, where $C^{(j)} \in \mathbb{R}^{90\times 90}$. If $c^{(j)} = vec(C^{(j)}) \in \mathbb{R}^{8100}$, then $c^{(j)} = A^{(j)} x^{(j)} + n^{(j)}$, where $A^{(j)} \in \mathbb{R}^{81000\times 81000}$ models the forward process and $n^{(j)} \in \mathbb{R}^{81000}$ is generated using the standard normal distribution. Here, $A^{(j)} = \sigma_j I_{81000}$ and σ_j is a constant generated using the standard normal distribution (see Fig. 2b for 6 sample images). We assumed that noisy image $\overline{C}$ does not

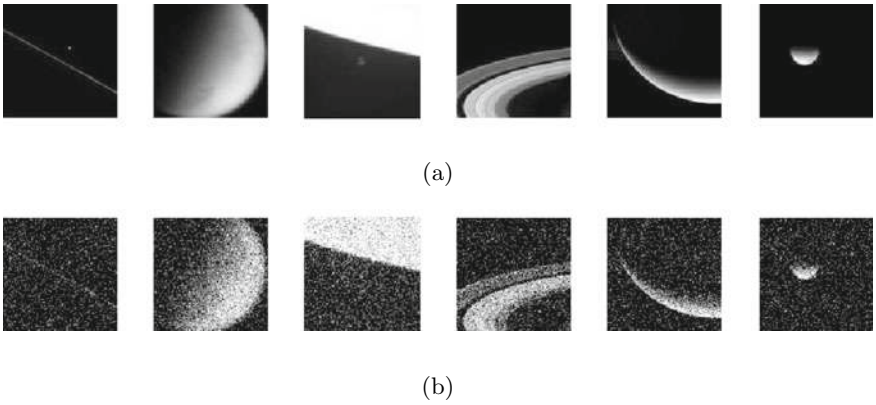


Fig. 2 **a** Some randomly selected satellite images. **b** Noisy versions of **a**

Table 1 Execution time, MSE, speedup, and percent difference between RCIMA and fast-RCIMA methods

c_r	RCIMA		fast-RCIMA		Speedup	Percent Difference (%)
	Time (s)	MSE	Time (s)	MSE		
0.25	484.73	102.36	30.47	102.36	15.91	93.71
0.5	454.70	38.72	32.75	38.72	13.88	92.80
0.75	426.31	20.23	34.91	20.23	12.21	91.81
1	420.45	2.01	38.01	2.01	11.06	90.95

necessarily belong to $\mathcal{C}$, but it is "similar" to one of them, i.e., there is $C^{(j)} \in \mathcal{C}$ such that

$$C^{(j)} \in \arg\min_{C^{(i)} \in \mathcal{C}} \|C^{(i)} - \overline{C}\|_{fr} \leq \delta,$$

for a given $\delta \geq 0$. Finally, to compute the optimal matrix $\widehat{\boldsymbol{F}} \in \mathbb{R}^{81000 \times 81000}$ in the RCIMA problem in (3), we defined matrices $X = [x^{(1)}\ x^{(2)}\ \ldots\ x^{(s)}] \in \mathbb{R}^{8100 \times 2000}$ and $C = [c^{(1)}\ c^{(2)}\ \ldots\ c^{(s)}] \in \mathbb{R}^{8100 \times 2000}$.

The compression ratio in RCIMA problem is given by $c_r = r/(mn)$. Table 1 presents the execution time, MSE, speedup and percent difference between RCIMA and fast-RCIMA methods, for compression ratio $c_r \in \{0.25, 0.5, 0.75, 1\}$, i.e., $r \in \{2025, 4050, 6075, 8100\}$. Figure 3 shows the estimates of $\overline{C}$ using both algorithms. Results in Table 1 suggest that fast-RCIMA method can obtain a nearly optimal approximation of the RCIMA problem in less time. Moreover, last column of Table 1 shows that the fast-RCIMA method is 90% faster, approximately, than the RCIMA method. Besides, Fig. 3 shows that the fast-RCIMA method compute optimal $\widehat{\boldsymbol{F}}$ in (3) with the same accuracy as RCIMA method.

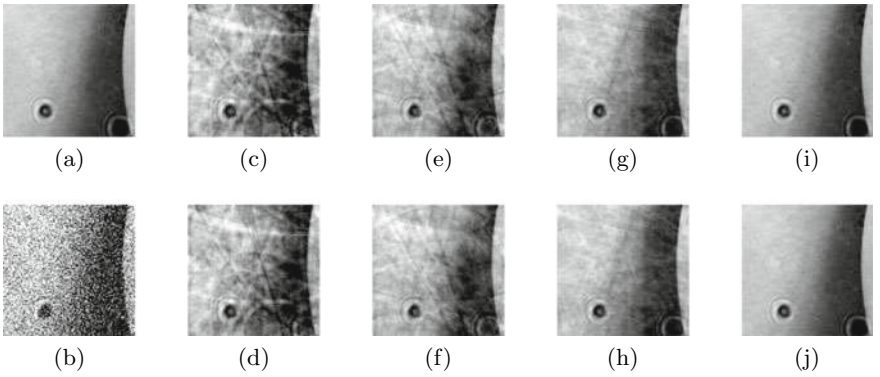


Fig. 3 Illustration to the estimation of noisy image $\overline{C}$ by RCIMA and fast-RCIMA methods. **a** Source image $\overline{X}$. **b** Noisy observed image $\overline{C}$. **c–d** Estimation using the RCIMA and fast-RCIMA methods, respectively, for $c_r = 0.25$. **e–f** Estimation using the RCIMA and fast-RCIMA methods, respectively, for $c_r = 0.5$. **g–h** Estimation using the RCIMA and fast-RCIMA methods, respectively, for $c_r = 0.75$. **i–j** Estimation using the RCIMA and fast-RCIMA methods, respectively, for $c_r = 1$

6 Conclusions

In this paper, we proposed in this paper a new and faster method for image deconvolution, based on the RCIMA problem in (3), using training data as a substitute for knowledge of a forward model. This new method, so-called the fast-RCIMA method, is based on tensor product and Tikhonov's regularization to approximate the pseudoinverse, and bilateral random projections to estimate the low-rank approximation. Moreover, in Theorem 1 we present an alternative approach to compute the low-rank matrix approximation. Based on the numerical simulations in Sect. 5, the fast-RCIMA method reduces significantly the execution time to compute optimal solution and increases the speedup, preserving the same accuracy of the classical method to solve the RCIMA problem. Numerical simulation to filter a noisy image in Sect. 5 demonstrates the advantages of the proposed method.

Acknowledgements This work was financially supported by the *Vicerrectoría de Investigación y Extensión* from *Instituto Tecnológico de Costa Rica*, Cartago, Costa Rica (Research #1440037).

References

1. Campisi P, Egiazarian K (2013) Blind image deconvolution: theory and applications. CRC Press
2. Chung J, Chung M (2013) Computing optimal low-rank matrix approximations for image processing. In: Proceddings IEEE 2013 Asilomar conference on signals, systems and computers, pp 670-674. https://doi.org/10.1109/ACSSC.2013.6810366

3. Eckart C, Young G (1936) The approximation of one matrix by another of lower rank. Psychometrika 1(3):211–218. https://doi.org/10.1007/BF02288367
4. Wu T, Sarmadi S, Venkatasubramanian V, Pothen A, Kalyanaraman A (2015) Fast SVD computations for synchrophasor algorithms. IEEE Trans Power Syst 31(2):1651–1652. https://doi.org/10.1109/TPWRS.2015.2412679
5. Allen-Zhu Z, Li Y (2016) LazySVD: even faster SVD decomposition yet without agonizing pain. In: Advances in neural information processing systems, 974–982
6. Courrieu P (2005) Fast computation of Moore-Penrose inverse matrices. Neural information processing. Lett Rev 8(2):25–29
7. Katsikis V, Pappas D (2008) Fast computing of the Moore-Penrose inverse matrix. Electron J Linear Algebra 17:637–650. https://doi.org/10.13001/1081-3810.1287
8. Lu S, Wang X, Zhang G, Zhou Z (2015) Effective algorithms of the Moore-Penrose inverse matrices for extreme learning machine. Intel Data Anal 19(4):743–760. https://doi.org/10.3233/IDA-150743
9. Barata J, Hussein M (2012) The Moore-Penrose pseudoinverse: a tutorial review of the theory. Brazilian J Phys 42:146–165. https://doi.org/10.1007/s13538-011-0052-z
10. Fazel M, Candes E, Recht B, Parrilo P (2008) Compressed sensing and robust recovery of low rank matrices. In: 42nd Asilomar conference on signals, systems and computers, pp 1043–1047. https://doi.org/10.1109/ACSSC.2008.5074571
11. Zhou T, Tao D (2012) Bilateral random projections. In: IEEE international symposium on information theory proceedings, pp 1286–1290. https://doi.org/10.1109/ISIT.2012.6283064
12. Telfer B, Casasent D (1994) Fast method for updating robust pseudoinverse and Ho-Kashyap associative processors. IEEE Trans Syst Man Cybern 24(9):1387–1390. https://doi.org/10.1109/21.310515
13. Benson M, Frederickson P (1986) Fast parallel algorithms for the Moore-Penrose pseudoinverse. In: Second conference on hypercube multiprocessors. https://www.osti.gov/biblio/7181991-fast-parallel-algorithms-moore-penrose-pseudo-inverse
14. Halko N, Martinsson P, Tropp J (2011) Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. SIAM Rev 53(2):217–288. https://doi.org/10.1137/090771806
15. Deshpande A, Vempala S (2006) Adaptive sampling and fast low-rank matrix approximation. In: Approximation, randomization, and combinatorial optimization. Algorithms and techniques. Springer, pp 292–303. https://doi.org/10.1007/11830924_28
16. Dattorro J (2005) Convex optimization † Euclidean distance geometry. Meboo Publishing. https://ccrma.stanford.edu/~dattorro/0976401304.pdf
17. Cragon H (2000) Computer architecture and implementation. Cambridge University Press
18. Soto-Quiros P, Torokhti A (2019) Improvement in accuracy for dimensionality reduction and reconstruction of noisy signals. Part II: the case of signal samples. Signal Process 154:272–279. https://doi.org/10.1016/j.sigpro.2018.09.020
19. NASA (2020) NASA solar system exploration database. Online, Accessed 10 Sept 2020. https://solarsystem.nasa.gov/raw-images/raw-image-viewer

Anexo 5

Journal of Computational and Applied Mathematics 401 (2022) 113763

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Effective implementation to reduce execution time of a low-rank matrix approximation problem

Jeffry Chavarría-Molina, Juan José Fallas-Monge, Pablo Soto-Quiros *

Escuela de Matemática, Instituto Tecnológico de Costa Rica, Cartago 30101, Costa Rica

ARTICLE INFO

Article history:
Received 22 February 2021
Received in revised form 29 June 2021

MSC:
15A09
15A23
15A29
15A60
65K10

Keywords:
Low-rank approximation
Bilateral random projection
Pseudoinverse
Execution time
Speedup

ABSTRACT

This paper proposes a new method to compute generalized low-rank matrix approximation (GLRMA). The GLRMA is a general case of the well-known low-rank approximation problem proposed by Eckart–Young in 1936. This new method, so-called the fast-GLRMA method, is based on tensor product and Tikhonov's regularization to approximate the pseudoinverse and bilateral random projections to estimate, in turn, the low-rank approximation. The fast-GLRMA method significantly reduces the execution time to compute the optimal solution, while preserving the accuracy of the classical method of solving the GLRMA. Computational experiments to measure execution time and speedup confirmed the efficiency of the proposed method.

1. Introduction

The low-rank approximation of $X \in \mathbb{R}^{m\times n}$ is a well-studied minimization problem, where the goal is to approximate X with a matrix $\widehat{X} \in \mathbb{R}^{m\times n}$, such that $\operatorname{rank}(\widehat{X}) \leq r \leq \min\{m, n\}$, in the sense that $\|X - \widehat{X}\|$ is small under some norm $\|\cdot\|$. This problem was proposed and solved in [1] by Eckart–Young in 1936, and is applied in a wide range of areas, such as linear system identification, machine learning, recommender systems and natural language processing [2,3].

Sonderman [4] and Friedland and Torokhti [5,6] solved a generalization of the low-rank problem, so-called the generalized low-rank matrix approximation (GLRMA). Given matrices $A \in \mathbb{R}^{p\times q}$, $B \in \mathbb{R}^{p\times m}$ and $C \in \mathbb{R}^{n\times q}$, the GLRMA problem finds a matrix $\widehat{X} \in \mathbb{R}_r^{m\times n}$ such that

$$\|A - B\widehat{X}C\|_{fr}^2 = \min_{X\in\mathbb{R}_r^{m\times n}} \|A - BXC\|_{fr}^2 , \tag{1}$$

where $\mathbb{R}_r^{m\times n}$ denotes the set of all real $m \times n$ matrices of rank at most $r \leq \min\{m, n\}$ and $\|\cdot\|_{fr}$ is the Frobenius norm.

In the literature, other low-rank matrix approximation problems related to (1) have been widely studied (see, e.g., Section 1 in [7] for further details). Some of them are as follows:

- $\min_X \|A - BXC\|_{fr}^2$ subject to $\operatorname{rank}(X) = r$ (see [8]);
- $\min_X \|A - BXB^*\|_{fr}^2$ subject to $\operatorname{rank}(X) = r$, where X is a Hermitian positive semidefinite matrix (see [9]);

* Corresponding author.
E-mail addresses: jchavarria@tec.ac.cr (J. Chavarría-Molina), jfallas@tec.ac.cr (J.J. Fallas-Monge), jusoto@tec.ac.cr (P. Soto-Quiros).

https://doi.org/10.1016/j.cam.2021.113763
0377-0427/

- $\min_X \|A - BX\|_{fr}$, subject to $\text{rank}(A_1 - B_1X) = r$ (see [10]).
- $\min_X \text{rank}(X)$ subject to $\|A - BXB\|_2$ being minimized, where X is a Hermitian positive semidefinite matrix (see [7]);

The GLRMA has been used successfully in many applications, such as, machine learning [11–13], matrix theory for computing inverse matrices [14–16], signal and image processing [15,17–21] and statistics [6,20]. Moreover, this problem is closely related to the mean-square error $\text{MSE} = \|A - \widehat{A}_{B,C}\|_{fr}^2$, where $\widehat{A}_{B,C} = B\widehat{X}C$.

In order to solve the GLRMA problem, it is necessary to use the singular value decomposition (SVD) to compute pseudoinverses and low-rank approximations of specific matrices [4–6]. The SVD method is highly accurate, but becomes impractical in the case of large matrices due to its computational complexity [22–24]. In particular, computing the SVD of an $m \times n$ matrix requires $O(\min\{m^2n, mn^2\})$ floating-points operations [25]. To reduce the elapsed-time computation, several techniques in real-life applications use fast algorithms to compute pseudoinverses and low-rank matrix approximations. Examples of these applications are neural learning algorithms [26–28], data mining methods [29], machine learning techniques [30–32] and image processing algorithms in background modeling [33] and biomedicine [23].

This paper proposes a new and efficient method to compute a solution for the GLRMA problem (1), so-called the fast-GLRMA method. The new method uses fast algorithms to approximate the pseudoinverse and low-rank matrix, respectively. The fast pseudoinverse algorithm used in the fast-GLRMA technique utilizes a special type of tensor product of two vectors [30] and Tikhonov's regularization [31,34] (see Section 3.1 for further details). Additionally, a bilateral random projection and its power-scheme modification [35,36] are used to estimate the low-rank approximation. Moreover, we propose a generalization of the method developed in [36] to estimate the low-rank approximation, which includes an analysis of the particular case $\text{rank}(Y_2) < r$, where $Y_2 \in \mathbb{R}^{m\times r}$ is a right random projection (see Section 3.2 for further details), and the theoretical error analysis when $\text{rank}(Y_2) = r$ (see Theorem 5 for further details).

The proposed method of estimating GLRMA significantly reduces execution time in computing matrix $\widehat{X}$ in (1) and, moreover, preserves the accuracy of the classical method developed in [5]. Finally, numerical simulations are presented to illustrate the advantages of the fast-GLRMA method.

In this paper, we have chosen methods mentioned in [30,31] and [35,36] for their easy computational implementation and high-speed and efficiency in calculating the pseudoinverse and low-rank approximation. However, other fast algorithms also exist to estimate the pseudoinverse and low-rank approximation. A brief review of other relevant methods is presented below:

- *Fast methods for computing the pseudoinverse*: Courrieu [26] presents an algorithm to approximate the pseudoinverse based on reverse order law and a full-rank Cholesky factorization of singular symmetric positive matrices. Brand [37] develops an identity for additive modifications of SVD, allowing for a fast solution for the pseudoinverse of a submatrix of an orthogonal matrix. Telfer and Casasent [38] propose a method based on the matrix inversion lemma to approximate a robust pseudoinverse. Benson and Frederickson [39] develop an efficient parallel implementation to compute the pseudoinverse. Schulz [40] presents an iterative method to approximate the pseudoinverse based on the classical Newton–Rapshon method. Other related works are presented in [32,41–44].
- *Fast methods for computing the low-rank approximation*: Deshpande and Vempala [45] combine adaptive sampling and volume sampling to obtain a low-rank approximation. Kannan, Mahoney and others [46,47] present a set of algorithms which compute the low-rank approximation based on the Monte Carlo method. Nguyen, Do and Tran [48] propose a fast algorithm that first spreads out the information in every column of a given matrix, then randomly selects some of the columns, and finally, generates the low-rank approximation based on the row space of the selected sets. Achlioptas and McSherry [49] introduce a simple technique for accelerating the computation of a low-rank approximation with a strong spectral structure, i.e., when the singular values of interest are significantly greater than those of a random matrix with similar size and entries. Other related works are developed in [37,50–53].

The rest of the paper is structured as follows. Section 2 presents a solution for the GLRMA problem based on the SVD method developed in [5]. Fast algorithms to estimate the pseudoinverse and low-rank approximation based on tensor product and bilateral projections, respectively, are explained in Section 3. In Section 4, we propose the fast-GRLMA method to approximate a solution to problem (1). Numerical examples based on random matrices and real-life applications are presented in Section 5. Finally, Section 6 contains some concluding remarks.

1.1. Notation

Throughout this paper, we use the following notation: $M^\dagger$ denotes the pseudoinverse of M. $I_m \in \mathbb{R}^{m\times m}$ is the identity matrix of order m. $M(:, i)$ denotes the ith column of M. Moreover, if $M \in \mathbb{R}^{m\times n}$ and $k \le n$, then $M(:, 1:k) = [M(:, 1)\ M(:, 2)\ \dots\ M(:, k)] \in \mathbb{R}^{m\times k}$.

Let $D = U_D\Sigma_D V_D^T$ be the singular value decomposition (SVD) of $D \in \mathbb{R}^{m\times n}$, where $U_D \in \mathbb{R}^{m\times m}$ and $V_D \in \mathbb{R}^{n\times n}$ are two orthogonal matrices (i.e., $U_D^TU_D = U_DU_D^T = I_m$ and $V_D^TV_D = V_DV_D^T = I_n$), and $\Sigma_D = \text{diag}(\sigma_1(D), \dots, \sigma_{\min(m,n)}(D)) \in \mathbb{R}^{m\times n}$ is a generalized diagonal matrix with singular values $\sigma_1(D) \ge \sigma_2(D) \ge \cdots \ge \sigma_{\min(m,n)}(D) \ge 0$ on the main diagonal. For $r \le \min\{m, n\}$, the r-truncated SVD is defined by

$$\lfloor D \rfloor_r = \sum_{i=1}^{r} \sigma_i(D) u_i v_i^T = U_{D,r}\Sigma_{D,r}V_{D,r}^T \in \mathbb{R}^{m\times n},$$

where $U_{D,r} \in \mathbb{R}^{m\times r}$ and $V_{D,r} \in \mathbb{R}^{n\times r}$ are formed with the first r columns of U_D and V_D, respectively, and $\Sigma_{D,r} = \text{diag}(\sigma_1(D), \ldots, \sigma_r(D)) \in \mathbb{R}^{r\times r}$. If $k = \text{rank}(D)$, then $P_{D,L} \in \mathbb{R}^{m\times m}$ and $P_{D,R} \in \mathbb{R}^{n\times n}$ are the orthogonal projection of D on the range of D and D^T, respectively, where $P_{D,L} = DD^\dagger = U_{D,k}U_{D,k}^T$ and $P_{D,R} = D^\dagger D = V_{D,k}V_{D,k}^T$.

2. GLRMA method

A solution for the GLRMA problem (1) was proposed and justified by Sonderman [4] and by Friedland and Torokhti [5]. Afterwards, Friedland and Torokhti showed in [6] that the solution for the GLRMA problem is not unique [5]. The following theorem presents the solution for the GLRMA problem given in [5].

Theorem 1. *Let $A \in \mathbb{R}^{p\times q}$, $B \in \mathbb{R}^{p\times m}$ and $C \in \mathbb{R}^{n\times q}$ be given matrices. Then, matrix*

$$\widehat{X} = B^\dagger \lfloor P_{B,L}AP_{C,R} \rfloor_r C^\dagger = B^\dagger \lfloor BB^\dagger AC^\dagger C \rfloor_r C^\dagger \tag{2}$$

minimizes the problem (1).

The matrix $M = P_{B,L}AP_{C,R} = BB^\dagger AC^\dagger C$ in (1) is known as the kernel of the GLRMA problem. The associated implementation of Theorem 1 is presented in Algorithm 1. This implementation is so-called the GLRMA method.

Algorithm 1: GLRMA method

Input : $A \in \mathbb{R}^{p\times q}$, $B \in \mathbb{R}^{p\times m}$, $C \in \mathbb{R}^{n\times q}$, $r \leq \min\{p, q\}$
Output: $\widehat{X} \in \mathbb{R}_r^{m\times n}$

1 $B_P = B^\dagger$
2 $C_P = C^\dagger$
3 $M = BB_PAC_PC$
4 $[U_M, S_M, V_M] = \text{svd}(M)$
5 $\widehat{X} = B_p \lfloor M \rfloor_r C_P = B_p U_{M,r} S_{M,r} V_{M,r}^T C_P$

Remark 1. In steps 3 and 5 of Algorithm 1, we use the so-called MMTimes algorithm developed by B. Luong [54]. The MMTimes algorithm considers the *matrix chain ordering problem* to find the most efficient way to multiply matrices (see, e.g., [55, p. 370]).

3. Fast algorithms to compute pseudoinverses and low-rank approximations

The GLRMA method uses the SVD to compute pseudoinverses and low-rank approximations. However, the SVD method is usually very expensive in terms of the execution time for high-dimensional matrices. Therefore, in this section, we show two fast algorithms to compute pseudoinverse and low-rank approximations, respectively. The fast algorithm to approximate the pseudoinverse uses a special type of tensor product of two particular vectors. The fast algorithm to approximate a low-rank approximation uses the so-called bilateral random projections. These two algorithms will reduce significantly the execution time to compute matrix $\widehat{X}$ in (1) and, moreover, will preserve the same accuracy of the GLRMA method.

3.1. Pseudoinverse and tensor product matrix

3.1.1. A brief review of pseudoinverse

The pseudoinverse of $Z \in \mathbb{R}^{m\times n}$, denoted by $Z^\dagger \in \mathbb{R}^{n\times m}$, is the unique matrix satisfying the following conditions:

$$ZZ^\dagger Z = Z, \qquad Z^\dagger ZZ^\dagger = Z^\dagger, \qquad \left(Z^\dagger Z\right)^T = Z^\dagger Z, \qquad \left(ZZ^\dagger\right)^T = ZZ^\dagger. \tag{3}$$

If $Z = U_Z \Sigma_Z V_Z^T$ is the SVD of Z and $k = \text{rank}(Z)$, then $Z^\dagger$ can be computed as follows:

$$Z^\dagger = V_{Z,k}\Sigma_{Z,k}^{-1}U_{Z,k}^T. \tag{4}$$

Eq. (4) is the standard procedure to calculate $Z^\dagger$. Table 1 shows the number of flops required to compute $Z^\dagger$ according to the Golub–Reinsch SVD and R-SVD methods, when $m \geq n$ [25].

3.1.2. Tensor product matrix method

Katsikis and Pappas [30] provide a very fast and reliable algorithm in order to compute the pseudoinverse of full-rank matrices, which does not use the SVD method. These authors use a special type of tensor product of two vectors, that is usually used in infinite dimensional Hilbert spaces. This type of tensor product gives conditions for the product of square

Table 1
Number of flops to compute $Z^{\dagger}$, when $m \geq n$.

Method	Flops
Golub–Reinsch SVD	$4m^2n + 8mn^2 + 9n^3 + (2k-1)(m+k)n$
R-SVD	$4m^2n + 22n^3 + (2k-1)(m+k)n$

matrices so that equations in (3) are satisfied. The method proposed in [30] to compute the pseudoinverse of a full-rank $Z \in \mathbb{R}^{m\times n}$ is defined by

$$Z^{\dagger} = \begin{cases} (Z^TZ)^{-1}Z^T & \text{if } m \geq n \\ Z^T(ZZ^T)^{-1} & \text{if } m < n \end{cases}. \tag{5}$$

Note that if Z is rank-deficient, then Z^TZ (if $m \geq n$) and ZZ^T (if $m < n$) are singular and, therefore, formula (5) is useless to compute $Z^{\dagger}$. Lu et al. [31] extended the method in [30] for computing the pseudoinverse of rank-deficient matrices, using an approximation of the Tikhonov's regularization in order to estimate $Z^{\dagger}$ (see, e.g., Theorem 4.3 in [34]). If Z is a rank-deficient matrix, the method proposed in [31] to approximate $Z^{\dagger}$ is defined by

$$Z^{\dagger} \approx Z_P = \begin{cases} (Z^TZ + \alpha I_n)^{-1}Z^T & \text{if } m \geq n \\ Z^T(ZZ^T + \alpha I_m)^{-1} & \text{if } m < n \end{cases}, \tag{6}$$

where α is a positive number close to zero. Note that if $x \in \mathbb{R}^m$ and $y \in \mathbb{R}^n$ are not null vectors, then $x^T(\alpha I_m + ZZ^T)x = \alpha\|x\|_2^2 + \|Z^Tx\|_2^2 > 0$ and $y^T(\alpha I_n + Z^TZ)y = \alpha\|y\|_2^2 + \|Zy\|_2^2 > 0$. Thus, $\alpha I_m + ZZ^T$ and $\alpha I_n + Z^TZ$ are positive definite and, accordingly, are invertible. Therefore, formula (6) is useful to approximate $Z^{\dagger}$. Moreover, from Theorem 4.3 in [34], if $\alpha \to 0$ in (6), then $Z^{\dagger} = Z_P$.

The method to estimate the pseudoinverse given by (5)–(6) is so-called the tensor product matrix (TPM) method, and its implementation is presented in Algorithm 2.

Algorithm 2: TPM method for computing pseudoinverse

Input : $Z \in \mathbb{R}^{m\times n}$
Output: $Z_P \in \mathbb{R}^{n\times m}$
if Z is full-rank **then**
  if $m \geq n$, **then** $Z_P = (Z^TZ)^{-1}Z^T$
  if $m < n$, **then** $Z_P = Z^T(ZZ^T)^{-1}$
else if Z is rank-deficient **then**
  $\alpha \in]0, 1[$
  if $m \geq n$, **then** $Z_P = (Z^TZ + \alpha I_n)^{-1}Z^T$
  if $m < n$, **then** $Z_P = Z^T(ZZ^T + \alpha I_m)^{-1}$

If $m \geq n$ and Z is a full-rank matrix, the number of flops to approximate the pseudoinverse with TPM method is shown in Table 2. Here, we assume that the inverse $(Z^TZ)^{-1}$ is computed using the LU decomposition with $\frac{4}{3}n^3 + \frac{3}{2}n^2 - \frac{5}{6}n$ flops [56]. The following two examples show the advantages of the TPM method to approximate the pseudoinverse.

Example 1. Let $D \in \mathbb{R}^{m\times n}$ be a full-rank matrix generated randomly from the standard uniform distribution. In Fig. 1, diagrams of the number of flops to compute $D^{\dagger}$ associated with the TPM, Golub–Reinsch SVD and R−SVD methods are presented. The number of flops of these methods are based on Tables 1 and 2. Figs. 1(a) and 1(b) show diagrams of the number of flops versus dimension m, when $n = m$ and $n = \frac{m}{2}$, respectively. It follows from Fig. 1 that the number of flops associated with the TPM method is less than those of the other methods.

Example 2. In this example, we compare the execution time of the TPM method and the MATLAB command `pinv` to compute the pseudoinverse. Command `pinv` uses the SVD given in (4) to compute the pseudoinverse [57]. Let $D \in \mathbb{R}^{m\times m}$ be a matrix generated randomly from the standard normal distribution, where $\text{rank}(D) = \frac{m}{2}$, i.e, D is rank-deficient. Diagrams of the time execution to compute $D^{\dagger}$ associated with TPM and command `pinv` versus dimension m are presented in Fig. 2(a). We use $\alpha = 0.1$. It follows from Fig. 2(a) that execution time associated with the TPM method is less than that of the command `pinv`. Moreover, Fig. 2(b) shows that accuracy of both methods are almost the same.

Based on Examples 1 and 2, the TPM method is a better option to approximate the pseudoinverse, compared to the classical method based on the SVD, given by (4).

Remark 2. The TPM method must calculate a matrix inverse in (5) and (6). To avoid this, there are iterative methods to approximate the pseudoinverse which involve only basic matrix operations such as multiplications and sums. For example,

Table 2
Estimates of the number of flops to compute $Z^{\dagger}$ with TPM method, when $m \geq n$ and Z is full-rank.

Steps	Flops
1. $Z^T Z$	$(2m-1)n^2$
2. $(Z^T Z)^{-1}$	$\frac{4}{3}n^3 + \frac{3}{2}n^2 - \frac{5}{6}n$
3. $(Z^T Z)^{-1} Z^T$	$(2n-1)mn$
Total	$\frac{1}{6}n(24mn - 6m + 8n^2 + 3n - 5)$

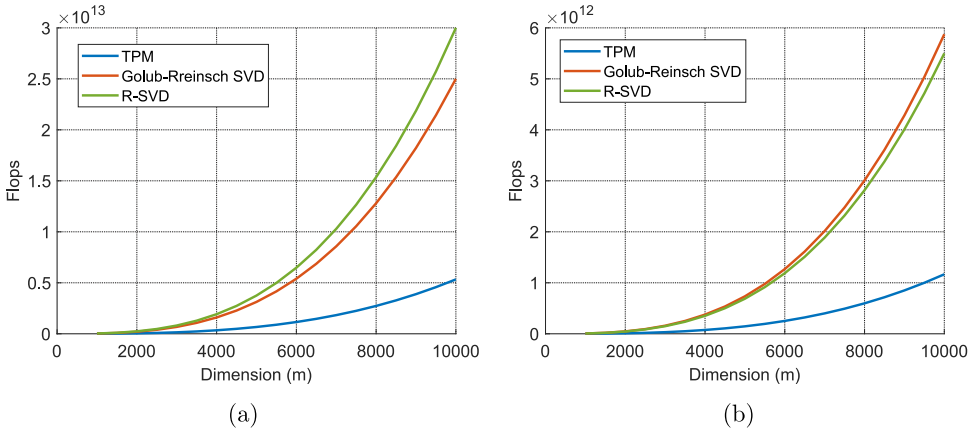


Fig. 1. Number of flops versus matrix dimension (m) used to execute the TPM, Golub–Reinsch SVD and $R-$SVD methods, for (a) $n = m$, (b) $n = m/2$.

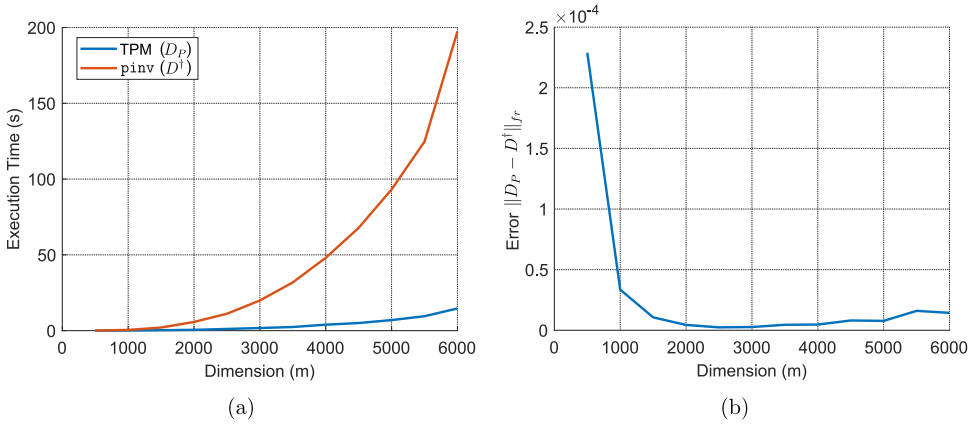


Fig. 2. (a) Execution time in seconds versus matrix dimension (m) used to run TPM method and command `pinv`. (b) Error versus matrix dimension (m) of the TPM method.

the classical Schulz's method [58] approximates $Z^{\dagger}$ with the iterative expression

$$X^{(k+1)} = X^{(k)}(2I_m - ZX^{(k)}), \quad k = 0, 1, 2, \ldots \tag{7}$$

where $X^{(0)} = \gamma Z^T$ and $\gamma \in]0, 2/\|Z\|_2^2[$. However, execution time of the Schulz's method is greater than that of the TPM method. For example, if $D \in \mathbb{R}^{2500\times 2000}$ is a full-rank random matrix generated from the standard uniform distribution, then the Schulz's method approximates $D^{\dagger}$ in 354.80s, using 22 iterations such that $\|X^{(22)} - D^{\dagger}\|_{fr} < 10^{-3}$. The TPM method computes $D^{\dagger}$ in 0.5485s. Other iterative methods to approximate the pseudoinverse are developed in [41,43,44,59,60].

Finally, in Theorem 2 we present an error analysis of the TPM method for computing the pseudoinverse.

Table 3
Number of flops to compute $\widehat{L}_r$, when $m \geq n$.

Method	Flops
Golub–Reinsch SVD	$4m^2n + 8mn^2 + 9n^3 + (2r-1)(m+r)n$
R-SVD	$4m^2n + 22n^3 + (2r-1)(m+r)n$

Theorem 2. *Consider a matrix $Z \in \mathbb{R}^{m\times n}$.*

1. *If Z is a full-rank matrix, then formula given by (5) computes exactly $Z^\dagger$.*
2. *If Z is a rank-deficient matrix, then*

$$\lim_{\alpha\to 0} \|Z^\dagger - Z_P\|_{fr} = 0,$$

where Z_p is given by (6).

Proof.

1. It is straightforward to check that formula given by (5) satisfies all four conditions in (3) and, therefore, represents an acceptable pseudoinverse of Z. By the unicity of the pseudoinverse, we conclude that formula given by (5) computes $Z^\dagger$.
2. See proof in the [34, Theorem 4.3]. □

3.2. Low-rank approximation and bilateral random projections

3.2.1. A brief review of low-rank approximation

Given $L \in \mathbb{R}^{m\times n}$ and $r \leq \min\{m, n\}$, the low-rank approximation problem is a minimization problem where the goal is to find $\widehat{L}_r \in \mathbb{R}_r^{m\times n}$ such that

$$\|L - \widehat{L}_r\|_{fr}^2 = \min_{L_r \in \mathbb{R}_r^{m\times n}} \|L - L_r\|_{fr}^2. \tag{8}$$

If $L = U_L \Sigma_L V_L^T$ is the SVD of L, then $\widehat{L}_r$ is given by the r-truncated SVD, i.e,:

$$\widehat{L}_r = \lfloor L \rfloor_r = U_{L,r} \Sigma_{L,r} V_{L,r}^T \tag{9}$$

Table 3 presents the number of flops required to compute $\widehat{L}_r$ based on the Golub–Reinsch SVD and R-SVD methods, when $m \geq n$ [25].

3.2.2. Bilateral random projection method

An alternative method to compute the low-rank approximation was developed by Fazel et al. in [35]. These authors showed that a fast method to estimate $\widehat{L}_r$ is given by

$$\widetilde{L}_r = Y_1 (A_2^T Y_1)^\dagger Y_2^T, \tag{10}$$

where

$$Y_1 = LA_1 \tag{11}$$

$$Y_2 = L^T A_2 \tag{12}$$

are r-bilateral random projections (BRP) of L. Here, $A_1 \in \mathbb{R}^{n\times r}$ and $A_2 \in \mathbb{R}^{m\times r}$ are arbitrary full-rank matrices. The matrices Y_1 and Y_2 are called the left and right random projections of L, respectively. Comparing with the randomized SVD in [53], that extracts the column space from unilateral random projections, the BRP method estimates both column and row spaces from bilateral random projections [36].

If L is a dense matrix, the number of flops to compute $\widetilde{L}_r$ is $r^2(2n + r) + mnr$, which is much less than that of the SVD based approximation [36]. However, if the singular values of L decay gradually, then the accuracy of (10) may be lost. To solve this problem, Zhou and Tao [36] propose a power-scheme to improve the approximation precision, when $A_2^T Y_1$ is nonsingular. In this paper, we extend the method presented in [36] for computing the low-rank matrix, when $A_2^T Y_1$ is singular. For this, we consider a power-scheme given by the left and right random projection of $X = (LL^T)^c L$ and L, respectively, i.e.,

$$Y_1 = XA_1 = (LL^T)^c LA_1 \tag{13}$$

$$Y_2 = L^T A_2, \tag{14}$$

where c is a nonnegative integer and $A_1 \in \mathbb{R}^{n\times r}$, $A_2 \in \mathbb{R}^{m\times r}$ are arbitrary full-rank matrices. This power-scheme is useful because $(LL^T)^c LA_1$ has the same singular vectors as LA_1, while its singular values decay more rapidly [53]. Besides, note

that the left random projection Y_1 in (13) can be represented as $Y_1 = L\widetilde{A}_1$, where $\widetilde{A}_1 = L^T(LL^T)^{c-1}A_1$, i.e., Eq. (13) is a particular case of (11).

Theorem 3 shows a new alternative to estimate a low-rank approximation of L, using a particular choice of A_2.

Theorem 3. *Consider the left and right random projections given by (13)–(14), respectively, where $A_1 \in \mathbb{R}^{n\times r}$ is a arbitrary full-rank matrix and $A_2 = L(L^TL)^{c-1}A_1$. Then, the estimation of the low-rank approximation of L in (10) can be simplified to*

$$\widetilde{L}_r = LY_2Y_2^{\dagger}. \tag{15}$$

Proof. Note that $(LL^T)^cL = L(L^TL)^c$ and, therefore, the left projection in (13) can be expressed as $Y_1 = LY_2$. Furthermore, from Proposition 3.2 in [34], we have $\left(Y_2^TY_2\right)^{\dagger}Y_2^T = Y_2^{\dagger}$. Then,

$$\begin{aligned}
\widetilde{L}_r &= Y_1\left(A_2^TY_1\right)^{\dagger}Y_2^T \\
&= LY_2\left[\left(L(L^TL)^{c-1}A_1\right)^TLY_2\right]^{\dagger}Y_2^T \\
&= LY_2\left[\left((L^TL)^cA_1\right)^TY_2\right]^{\dagger}Y_2^T \\
&= LY_2\left[Y_2^TY_2\right]^{\dagger}Y_2^T \\
&= LY_2Y_2^{\dagger}. \quad \square
\end{aligned}$$

If Y_2 is full-rank, then the formula (15) can be simplified. Theorem 4 presents an alternative method to estimate a low-rank approximation of L, when $\operatorname{rank}(Y_2) = r$.

Theorem 4. *Consider the left and right random projections given by (13)–(14), respectively, where $A_1 \in \mathbb{R}^{n\times r}$ is an arbitrary full-rank matrix and $A_2 = L(L^TL)^{c-1}A_1$. If Y_2 is a full-rank matrix, then, the estimation of the low-rank approximation of L in (10) can be simplified to*

$$\widetilde{L}_r = LQ_rQ_r^T, \tag{16}$$

where $Y_2 = QR$ is the QR decomposition of Y_2 and $Q_r = Q(:, 1:r)$.

Proof. If $Y_2 = QR$ is the QR decomposition of Y_2, it follows from Theorem 5.2.2 in [25] that $Y_2 = Q_rR_r$, where $R_r \in \mathbb{R}^{r\times r}$ is an upper triangular matrix formed with the first r rows of R and $Q_r = Q(:, 1:r) \in \mathbb{R}^{n\times r}$. Matrix R_r is nonsingular, because Y_2 is full-rank. Moreover, $\left[R_r^TR_r\right]^{\dagger} = \left[R_r^TR_r\right]^{-1}$ and $Q_r^TQ_r = I_r$. Then, it follows from Theorem 3 that

$$\begin{aligned}
\widetilde{L}_r &= LY_2Y_2^{\dagger} \\
&= LY_2\left[Y_2^TY_2\right]^{\dagger}Y_2^T \\
&= LQ_rR_r\left[(Q_rR_r)^TQ_rR_r\right]^{\dagger}(Q_rR_r)^T \\
&= LQ_rR_r\left[R_r^TQ_r^TQ_rR_r\right]^{\dagger}R_r^TQ_r^T \\
&= LQ_rR_r\left[R_r^TR_r\right]^{-1}R_r^TQ_r^T \\
&= LQ(R_rR_r^{-1})[(R_r^T)^{-1}R_r^T]Q_r^T \\
&= LQ_rQ_r^T. \quad \square
\end{aligned}$$

Remark 3. If Y_2 is rank-deficient, then $\operatorname{rank}(Y_2) < r$. Note that $Y_2 = L^TA_2 = (L^TL)^cA_1$. Moreover, $\operatorname{rank}((L^TL)^c) = \operatorname{rank}(L)$, because if $L = U_L\Sigma_LV_L^T$ is the SVD of L, then $(L^TL)^c = V_L\Sigma_L^{2c}V_L^T$ is the SVD of $(L^TL)^c$ and, therefore, L and $(L^TL)^c$ have the same number of positive singular values. From Corollary 2.5.10 in [61], we obtain that $\operatorname{rank}(Y_2) \leq \min\{\operatorname{rank}((L^TL)^c), \operatorname{rank}(A_1)\} = \min\{\operatorname{rank}(L), r\}$, and therefore

$$\operatorname{rank}(Y_2) \leq \min\{\operatorname{rank}(L), r\}.$$

If $\operatorname{rank}(L) \leq r$, then $L \in \mathbb{R}_r^{m\times n}$ and, therefore, $\widetilde{L}_r = L$. Otherwise, if $r < \operatorname{rank}(L)$, then $\widetilde{L}_r$ can be computed by (15) using the TPM method explained in Section 3.1.2, i.e.

$$\widetilde{L}_r = LY_2(Y_2^TY_2 + \alpha I_r)^{-1}Y_2^T, \tag{17}$$

where α is a positive number close to zero. However, if A_1 is generated randomly, then the case $\operatorname{rank}(Y_2) < r$ is unusual. For example, each numerical simulation in Section 5 generates a full-rank matrix Y_2.

The method to estimate the low-rank approximation of L, given by Theorems 3, 4 and Remark 3, is so-called the modified bilateral random projections (MBRP) method. The pseudocode of the MBRP method is presented in Algorithm 3.

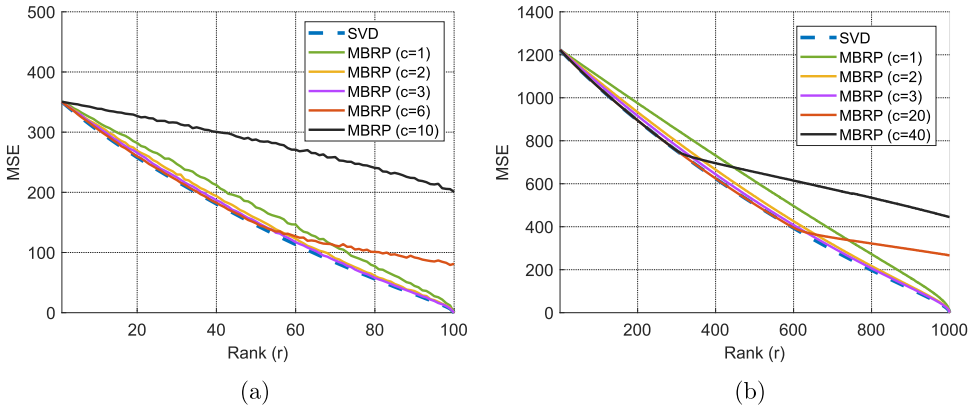


Fig. 3. MSE versus a rank constraint r via the MBRP method and the SVD approach. (a) $m = 100$ and $n = 150$, (b) $m = 2000$ and $n = 1500$.

Note that bilateral random projections $Y_1 = LA_1$ and $Y_2 = L^T A_2 = (L^T L)^c A_1$ can be computed efficiently using a loop in steps 1–4 of Algorithm 3.

Algorithm 3: MBRP method for computing the low-rank approximation

Input : $L \in \mathbb{R}^{m\times n}$, $r \leq \min\{m, n\}$, $A_1 \in \mathbb{R}^{n\times r}$ and $c \in \{1, 2, 3, \ldots\}$
Output: $\widetilde{L}_r \in \mathbb{R}_r^{m\times n}$

1 $Y_2 = A_1$
2 **for** $i = 1 : c$ **do**
3 $\quad Y_1 = LY_2$
4 $\quad Y_2 = L^T Y_1$
5 **if** Y_2 is full-rank **then**
6 $\quad [Q, \sim] = \text{qr}(Y_2)$
7 $\quad Q_r = Q(:, 1 : r)$
8 $\quad \widetilde{L}_r = LQ_r Q_r^T$
9 **else if** $\text{rank}(L) < r$ **then**
10 $\quad \widetilde{L}_r = L$
11 **else**
12 $\quad \alpha \in]0, 1[$
13 $\quad \widetilde{L}_r = LY_2(Y_2^T Y_2 + \alpha I_r)^{-1} Y_2^T$

The power-scheme is useful to improve accuracy of the estimation of the low-rank approximation. However, if MBRP method is executed in floating-point arithmetic and c is a large number, then all information associated with singular values smaller than $\mu^{1/(2c+1)}\|L\|_{fr}$ is lost, where μ is the machine precision [53]. Therefore, it is recommended to take a small value for c. Example 3 illustrates the above case.

Example 3. Let $L \in \mathbb{R}^{m\times n}$ be an arbitrary matrix generated from the standard normal distribution. We apply the MBRP method for different values of c to obtain estimations with rank from 1 to $\min\{m, n\}$. Fig. 3(a) and 3(b) show diagrams of the MSE versus a rank constraint r, for $(m, n) = (100, 150)$ and $(m, n) = (1000, 1500)$, respectively. The MSE of the corresponding SVD approximation is shown as a baseline. It is clear that the MBRP method can obtain a nearly optimal approximation when $c = 2$ or $c = 3$. However, in this example, if c is large, then accuracy of the MBRP method is not good enough.

Remark 4. Numerical simulations in this section and Section 5 are computed using $c = 3$ on the power-scheme of the MBRP method.

If $m \geq n$, $c = 3$ and Y_2 is a full-rank matrix, then the number of flops to estimate the low-rank approximation with MBRP method consists of computation of $Y_2 = (L^T L)^3 A_1$, QR decomposition of Y_2 and matrix multiplication $LQ_r Q_r^T$. Table 4 shows the number of flops required to compute $\widetilde{L}_r$, when $m \geq n$, $c = 3$ and Y_2 is a full-rank matrix. Here, we assume

Table 4
Estimates of the number of flops to compute $\widetilde{L}_r$ with MBRP method, when $m \geq n$, $c = 3$ and Y_2 is full-rank.

Steps	Flops
1. Y_2	$3(4mnr - mr - nr)$
2. $[Q, \sim] = \mathtt{qr}(Y_2)$	$4m^2r - 4mr^2 + \frac{4}{3}r^3$
3. $LQ_rQ_r^T$	$4mnr - mr - mn$
Total	$4m^2r + 16mnr - mn - 4mr^2 - 4mr - 3nr + \frac{4}{3}r^3$

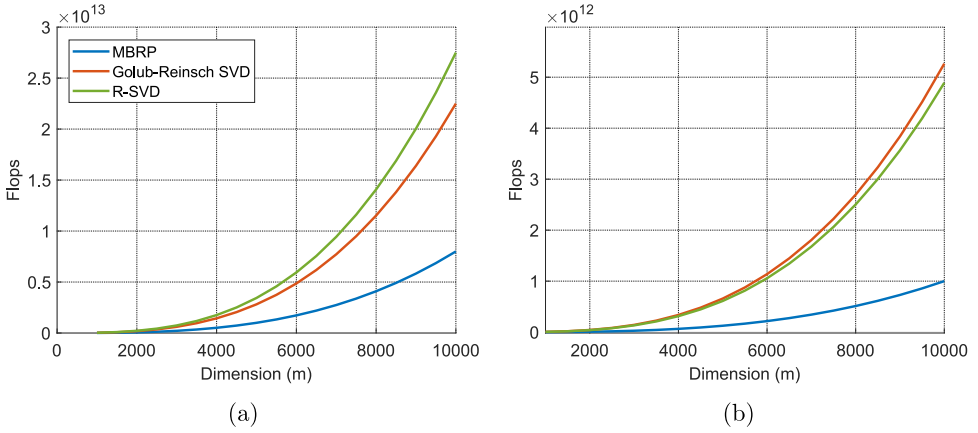


Fig. 4. Number of flops versus matrix dimension (m) used to execute the MBRP, Golub–Reinsch SVD and $R-$SVD methods for (a) $n = m$ and $r = \frac{m}{2}$, (b) $n = \frac{m}{2}$ and $r = \frac{m}{8}$.

that the QR decomposition is computed using the Householder algorithm,[1] which requires $4m^2r - 4mr^2 + \frac{4}{3}r^3$ flops [25]. Besides, Y_2 is optimally computed using a loop, as it is shown in steps 1–4 of Algorithm 3. Examples 4 and 5 illustrate the advantages of the proposed MBRP method to estimate the low-rank approximation.

Example 4. Let $L \in \mathbb{R}^{m\times n}$ be a full-rank matrix generated randomly from the standard normal distribution. In Fig. 4, diagrams of the number of flops to estimate L_r associated with the MBRP (when Y_2 is full-rank), Golub–Reinsch SVD and $R-$SVD methods, are presented. The number of flops of these methods are based on Tables 3 and 4. Figs. 4 shows the cases of the number of flops versus the dimension m. Figs. 4(a) shows the case $n = m$ and $r = \frac{m}{2}$. Fig. 4(b) show the cases $n = \frac{m}{2}$ and $r = \frac{m}{8}$. It follows from diagrams on Fig. 4 that the number of flops associated with the MBRP method is less than those of the other methods.

Example 5. In this example, we compare the execution time of the MBRP method versus the execution time of the r-truncated SVD, which is used by MATLAB to compute the low-rank approximation of a matrix. Let $L \in \mathbb{R}^{m\times n}$ be a matrix generated randomly from the standard uniform distribution, where $n = \frac{m}{2}$ and $r = \frac{m}{8}$. Fig. 5(a) presents diagrams of the time execution to compute the low-rank approximation associated with MBRP method and the r-truncated SVD, versus the dimension m. It follows from Fig. 5(a) that execution time associated with the TPM method is less than that of the r-truncated SVD. Furthermore, Fig. 5(b) shows that accuracy of both methods are almost the same.

Finally, in Theorem 5, we present an error analysis of the TPM method for computing the pseudoinverse, when $r = \mathrm{rank}(Y_2)$.

Theorem 5. *Let $L_r \in \mathbb{R}^{m\times n}$ be the low-rank matrix approximation of $L \in \mathbb{R}^{m\times n}$. If $r = \mathrm{rank}(Y_2)$, $\alpha = \sum_{i=r+1}^{\min\{m,n\}} \sigma_i^2(L)$ and $\beta = (n-r)\sigma_{max}^2(L)$, then*

$$\|L_r - \widetilde{L}_r\|_{fr}^2 \in [\max\{0, \alpha - \beta\}, \alpha + \beta].$$

Proof. See proof in Appendix. □

[1] The Householder algorithm is used by the function `qr` in MATLAB.

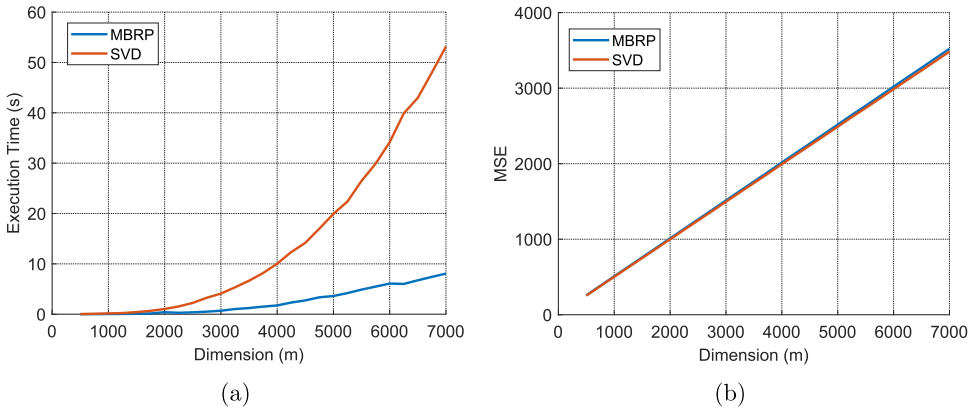


Fig. 5. (a) Execution time in seconds versus matrix dimension (m), used by the MBRP method and the r-truncated SVD. (b) MSE of MBRP method and r-truncated SVD.

4. Fast-GLRMA method

In this section, we propose a new implementation to compute a solution for the GLRMA problem on the basis of the TPM and the MBRP methods explained in Sections 3.1 and 3.2, respectively. A block diagram of this new implementation, called the fast-GLRMA method, is presented in Fig. 6, which is explained next: Given matrices A, B, C, first, we compute pseudoinverses B_P and C_P of B and C, respectively, using the TPM method. Afterwards, we compute the kernel matrix $M = BB_PAC_PC$ and estimate its low-rank approximation $\widetilde{M}_r$ using the MBRP method. Finally, we approximate the GLRMA solution $\widehat{X} = B_P\widetilde{M}_rC_P$. W.l.g, the associate pseudocode of the fast-GLRMA method is shown in Algorithm 4 when $p \geq m \geq q$ and the right random projection Y_2 is full-rank. In addition to this, the power-scheme uses $c = 3$.

Algorithm 4: Fast-GLRMA method

Input : $A \in \mathbb{R}^{p\times q}$, $B \in \mathbb{R}^{p\times m}$, $C \in \mathbb{R}^{n\times q}$, $r \leq \min\{p, q\}$, $A_1 \in \mathbb{R}^{n\times r}$, $\alpha \in]0, 1[$
Output: $\widehat{X} \in \mathbb{R}_r^{m\times n}$
/* Computing the pseudoinverse of B with the TPM Method */
1 **if** B is full-rank **then**
2 $\quad B_P = (B^TB)^{-1}B^T$
3 **else if** B is rank-deficient **then**
4 $\quad B_P = (B^TB + \alpha I_m)^{-1}B^T$
/* Computing the pseudoinverse of C with the TPM Method */
5 **if** C is full-rank **then**
6 $\quad C_P = (C^TC)^{-1}C^T$
7 **else if** C is rank-deficient **then**
8 $\quad C_P = (C^TC + \alpha I_q)^{-1}C^T$
/* Computing the kernel matrix M */
9 $M = BB_PAC_PC$
/* Computing the low-rank Matrix approximation of M with the MBRP method */
10 $Y_2 = A_1$
11 **for** $i = 1 : 3$ **do**
12 $\quad Y_1 = MY_2$
13 $\quad Y_2 = M^TY_1$
14 $[Q, \sim] = \text{qr}(Y_2)$
15 $Q_r = Q(:, 1 : r)$
16 $\widetilde{M}_r = MQ_rQ_r^T$
/* Computing the generalized low-rank matrix approximation */
17 $\widehat{X} = B_P\widetilde{M}_rC_P$

Remark 5. Similarly to the GLRMA algorithm, we used the MMTimes method in steps 9, 16 and 17 in Algorithm 4 (see Remark 1).

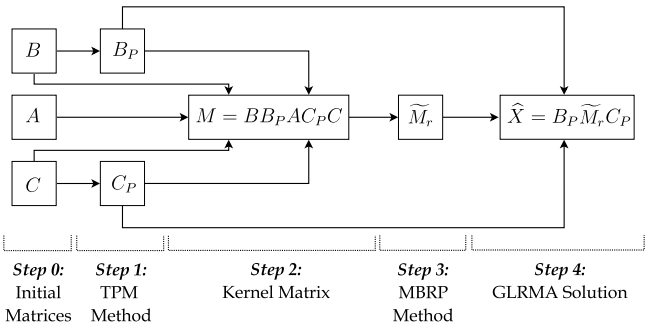


Fig. 6. Block diagram of the fast-GLRMA method.

Remark 6. As mentioned in Remark 2, matrices B_P and C_P computed in steps 1–4 and 5–8, respectively, can be approximated using a iterative method to avoid calculation of inverse matrices, for example, the Newton–Schulz method defined in (7).

5. Numerical examples

In this section, we show numerically the advantages of the proposed fast-GLRMA method. The numerical examples were run on a desktop computer with 2.30 GHz processor (Intel(R) Core(TM) i7-4712MQ CPU) and 8.00 RAM.

The computational performance analysis of the fast-GLRMA method is evaluated using the metrics *speedup* and *percent difference* [62, p. 27]. Consider two methods that solve the same numerical problem, Method 1 and Method 2, with execution times T_1 and T_2, respectively. The speedup S (or acceleration) is the ratio between the execution times of both methods, i.e.

$$S = \frac{T_2}{T_1}.$$

If Method 1 is an improvement of Method 2, then S will be greater than 1. However, if Method 1 hurts performance, speedup will be less than 1. The percent difference P between Method 1 and Method 2, where $T_1 < T_2$, is represented by

$$P = 100\left(\frac{T_2 - T_1}{T_2}\right)$$

Then, we say that Method 1 is P% faster than Method 2.

5.1. Illustrative example 1: random matrices

We first evaluate the efficiency of the fast-GLRMA method to approximate the solution for the GLRMA problem (1) using random matrices. We consider full-rank matrices $A \in \mathbb{R}^{p\times q}$, $B \in \mathbb{R}^{p\times m}$ and $C \in \mathbb{R}^{n\times q}$ generated from a uniform distribution with zero-mean and standard deviation 1. Here, $n = m$, $p = 2m$, $q = \frac{m}{2}$, where $m \in \{200, 400, 600, \ldots, 9600, 9800, 10000\}$.

Fig. 7 shows some diagrams of numerical simulations with $r = \frac{m}{4}$. Fig. 7(a) represents typical diagrams of the matrix dimension m versus execution time (in seconds) used to execute the GLRMA and the fast-GLRMA methods. It is observed that there is significant reduction in execution time of the fast-GLRMA method. Moreover, Fig. 7(b) shows that accuracy of both methods are the same. Figs. 7(c) and 7(d) represent speedup and percent difference between both methods. In particular, it is observed in Fig. 7(c) that acceleration is greater than 1, and thus, the fast-GLRMA method is an improvement of the GLRMA method. Moreover, Fig. 7(d) shows that the fast-GLRMA method is 65%–85% faster than the GLRMA method, approximately.

Therefore, this example illustrates that the fast-GLRMA method is faster than the GLRMA method and preserves the same accuracy.

5.2. Illustrative example 2: generalized low-rank approximation of the left inverse

In this example, we compute a generalized low-rank approximation inverse of $D \in \mathbb{R}^{n\times q}$ [14,15]. In particular, we consider the generalized low-rank approximation of the left inverse (GLRA-LI) of D, i.e., find matrix $\widehat{X} \in \mathbb{R}_r^{q\times n}$ such that

$$\|(I_q - \widehat{X}D)M\|_{fr}^2 + \beta\|\widehat{X}\|_{fr}^2 = \min_{X\in\mathbb{R}_r^{q\times n}} \|(I_q - XD)M\|_{fr}^2 + \beta\|X\|_{fr}^2, \tag{18}$$

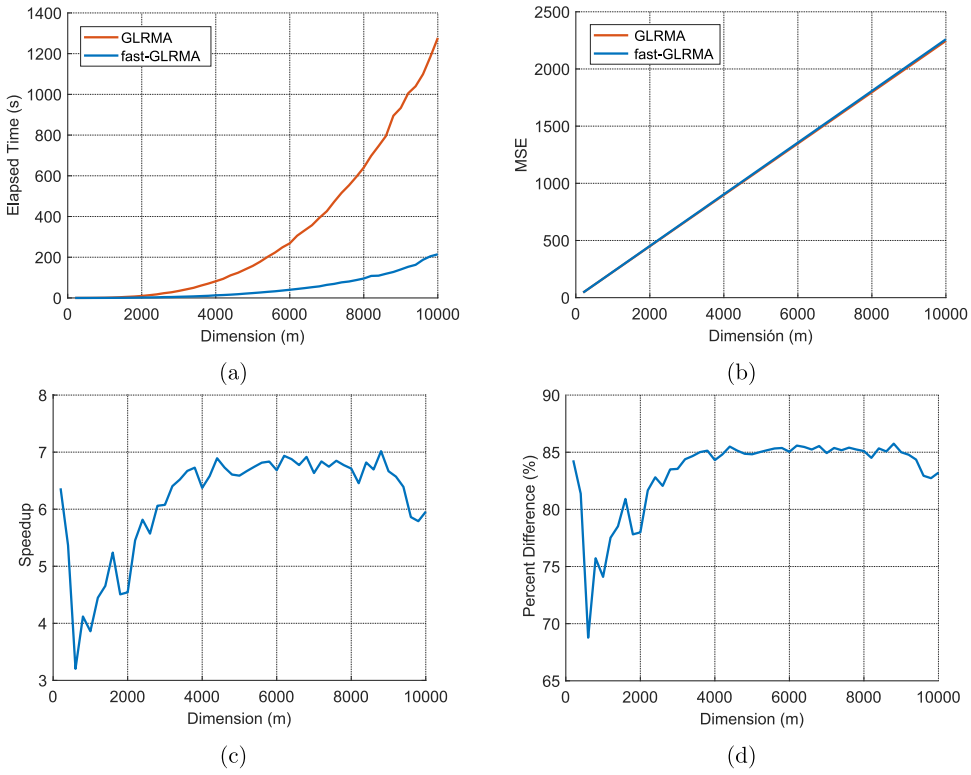


Fig. 7. Diagrams of the GLRMA and the fast-GLRMA methods. The matrix dimension m versus: (a) execution time (in seconds) (b) MSE (c) speedup S (d) percent difference between the fast-GLRMA and the GLRMA method.

where $M \in \mathbb{R}^{q\times q}$ is an invertible weighting matrix and $\beta > 0$. Problem (18) is used for computing solutions to ill-posed inverse problems in signal and image processing [15,17]. Here, we study the particular case $\beta = 0$, i.e.,

$$\|(I_q - \widehat{X}D)M\|_{fr}^2 = \min_{X\in\mathbb{R}_r^{q\times n}} \|(I_q - XD)M\|_{fr}^2. \tag{19}$$

If we take $A = M$, $B = I_q$ and $C = DM$, the problem (19) can be written as the GLRMA problem. In our numerical simulations, we consider full-rank matrices D and M of dimension $n = q$. Each matrix is generated from a Gaussian distribution with zero-mean and standard deviation 1. Here, M is a diagonal matrix.[2] Tables 5 and 6 present execution time, MSE, speedup and percent difference between GLRMA and fast-GLRMA methods to compute the GLRA-LI of D.

In Table 5, we consider $n \in \{100, 500, 1000, 2500, 5000, 7500, 10000\}$ and $r = m/2$. It is clear that the fast-GLRMA method achieves the same accuracy as the GLRMA method, in significantly less time. Moreover, last two columns of Table 5 show that speedup and percent difference of the fast-GLRMA method to approximate the GLRA-LI increase as dimension n increases. Thus, we conclude that the fast-GLRMA method is between 45% to 90% faster than the GRLMA method. In Table 6, we use different valued of rank r when $n \in \{2500, 5000\}$. Similar to Table 5, results in Table 6 suggest that the proposed method can obtain a nearly optimal approximation of the GLRA-LI in less time. Therefore, it follows from Tables 5 and 6 that the fast-GLRMA method is the fastest method and compute the GLRA-LI of D, with the same accuracy as the GRLMA method.

5.3. Illustrative example 3: image denoising

This example illustrates an application of the GLRMA problem to image denoising [15,17,19]. Specifically, we apply GLRMA and fast-GLRMA methods to the problem of compression, filtering and decompression of a noisy image $\overline{C} \in \mathbb{R}^{90\times 90}$ on the basis of the set of training images $\mathcal{A} = \{A^{(1)}, \ldots, A^{(s)}\}$, where $A^{(j)} \in \mathbb{R}^{90\times 90}$, for $j = 1, \ldots, s$. Our training set $\mathcal{A}$ consists of $s = 2000$ different satellite images from Saturn (see Fig. 8(a) for 9 sample images). Images in $\mathcal{A}$ were taken from the NASA Solar System Exploration Database [63].

[2] As mentioned in [14], the weight matrix M allows to emphasize some terms in the Frobenius norm over others. In particular, a diagonal M can be used to downweight columns of C that have more uncertainty than others.

Table 5
Execution time, MSE, speedup and percent difference between the GLRMA and the fast-GLRMA methods, using $r = m/2$.

n	GLRMA		fast-GLRMA		Speedup	Percent difference (%)
	Time (s)	MSE	Time (s)	MSE		
100	0.0100	7.0711	0.0054	7.0711	1.8329	45.4414
500	0.1831	15.8114	0.0511	15.8114	3.5838	72.0966
1000	1.1566	22.3607	0.3640	22.3607	3.1771	68.5246
2500	31.2303	35.3553	4.4705	35.3553	6.9859	85.6854
5000	222.1110	50.0000	28.0227	50.0000	7.9261	87.3834
7500	719.4189	61.2372	95.7267	61.2372	7.5153	86.6939
10000	2484.4010	70.7107	245.3393	70.7107	10.1264	90.1248

Table 6
Execution time, MSE, speedup and percent difference between the GLRMA and the fast-GLRMA methods.

n	r	GLRMA		fast-GLRMA		Speedup	Percent difference (%)
		Time (s)	MSE	Time (s)	MSE		
2500	50	31.2373	49.4975	2.9068	49.4975	10.7771	90.7211
	250	31.3206	47.4342	3.1143	47.4342	10.0570	90.0567
	500	31.3579	44.7214	3.2841	44.7214	9.5485	89.5272
	1000	31.3723	38.7298	4.1107	38.7298	7.6318	86.8969
	1500	32.0511	31.6228	4.5128	31.6228	7.1023	85.9200
	2000	31.9452	22.3607	4.9886	22.3607	6.4037	84.3839
	2500	32.2264	0.0000	5.7610	0.0000	5.5939	82.1234
5000	100	223.1532	70.0000	20.0127	70.0000	11.1506	91.0319
	500	220.8744	67.0820	22.5967	67.0820	9.7746	89.7694
	1000	220.7926	63.2456	23.7203	63.2456	9.3082	89.2568
	1500	223.4235	59.1608	25.8941	59.1608	8.6283	88.4103
	2000	222.0916	54.7723	28.3236	54.7723	7.8412	87.2469
	2500	223.2146	50.0000	30.5241	50.0000	7.3127	86.3252
	3000	223.3798	44.7214	32.9344	44.7214	6.7826	85.2563
	3500	225.4612	38.7298	35.2550	38.7298	6.3952	84.3632
	4000	224.3768	31.6228	39.6265	31.6228	5.6623	82.3393
	4500	225.8606	22.3607	39.9624	22.3607	5.6518	82.3066
	5000	227.0217	0.0000	42.3396	0.0000	5.3619	81.3500

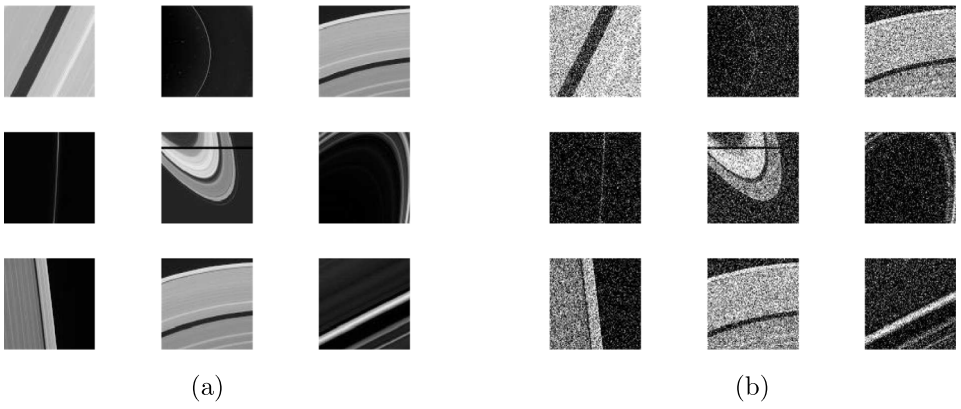


Fig. 8. (a) Some randomly selected images of Saturn. (b) Noisy versions of images in (a).

It is assumed that instead of images in $\mathcal{A}$, we observed their noisy version $\mathcal{C} = \left\{C^{(1)}, \ldots, C^{(s)}\right\}$, where $C^{(j)} \in \mathbb{R}^{90\times 90}$, for $j = 1, \ldots, s$. Each $C^{(j)}$ is simulated as $C^{(j)} = A^{(j)} + 0.2N^{(j)}$, where $N^{(j)} \in \mathbb{R}^{90\times 90}$ is a random matrix generated from a normal distribution with zero-mean and standard deviation 1. For each image $A^{(j)}$, matrix $N^{(j)}$ simulates noise. (see Fig. 8(b) for 9 sample images). It is assumed that noisy image $\overline{C}$ does not necessarily belong to $\mathcal{C}$, but it is "similar" to one of them, i.e., there is $C^{(j)} \in \mathcal{C}$ such that

$$C^{(j)} \in \arg\min_{i=1,\ldots,s} \|C^{(i)} - \overline{C}\|_{fr} \quad \text{and} \quad \|C^{(j)} - \overline{C}\|_{fr} \leq \delta,$$

for a given $\delta \geq 0$.

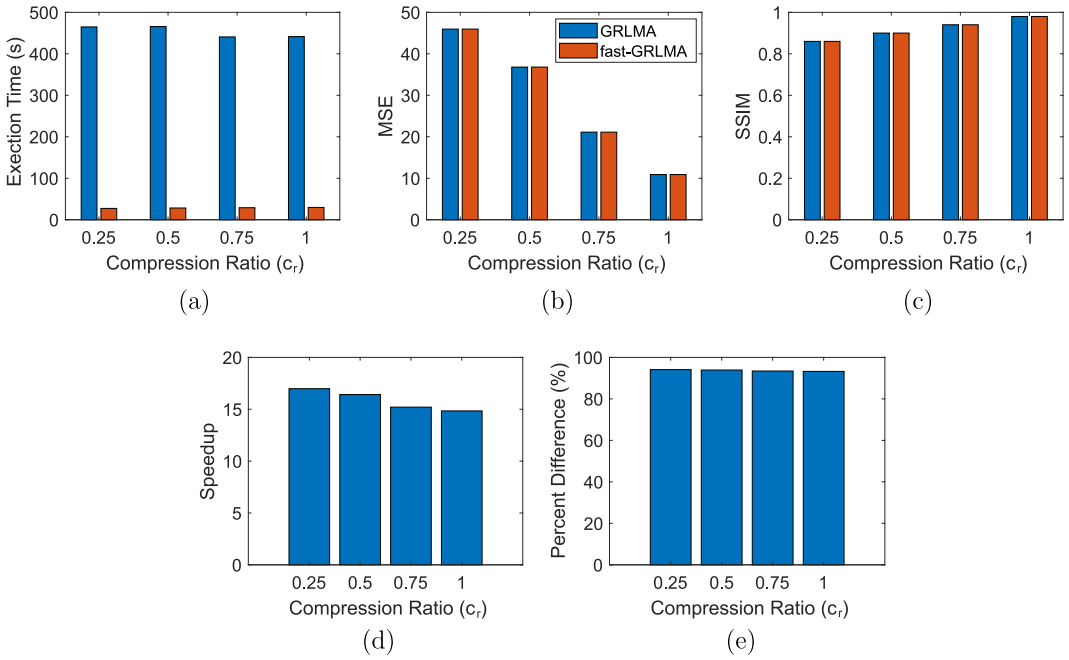


Fig. 9. Bar graphs of GLRMA and fast-GLRMA methods. Compression ratio c_r versus: (a) execution time (in seconds) (b) MSE (c) SSIM (d) speedup S (e) percent difference between fast-GLRMA and GLRMA methods.

We will formulate the problem under consideration in terms of the GRLM problem. To this end, we vectorize matrices $A^{(j)}$ and $C^{(j)}$, i.e., convert each matrix into a column vector by stacking the columns of the matrix. Let $vect : \mathbb{R}^{m\times n} \to \mathbb{R}^{mn}$ be the vectorization transform. We write $a_j = vec(A^{(j)}) \in \mathbb{R}^{8100}$ and $c_j = vec(C^{(j)}) \in \mathbb{R}^{8100}$. Then, matrices $A = [a_1\ a_2\ \ldots\ a_s] \in \mathbb{R}^{8100\times 2000}$, $C = [c_1\ c_2\ \ldots\ c_s] \in \mathbb{R}^{8100\times 2000}$ and $B = I_{8100}$ are the matrices to use in the GRLM problem to find a matrix $\widehat{X}$, i.e., find matrix $\widehat{X} \in \mathbb{R}_r^{8100\times 8100}$ such that

$$\|A - \widehat{X}C\|_{fr}^2 = \min_{X\in\mathbb{R}_r^{8100\times 8100}} \|A - XC\|_{fr}^2 . \tag{20}$$

From steps 16 and 17 in Algorithm 4, $\widehat{X} = MQ_rQ_r^TC_P = \mathcal{D}\mathcal{C}$, where $\mathcal{C} = Q_r^TC_P$ is the linear operator designed for compression and $\mathcal{D} = MQ_r$ is the linear operator designated for decompression [18]. Both linear operators filter a vectorized version of the noisy image $\overline{C}$.

The compression ratio in problem (20) is given by $c_r = r/\min\{p, q\}$. Fig. 9 presents the execution time, MSE, structural similarity index[3] (SSIM), speedup and percent difference between GLRMA and fast-GLRMA methods, for compression ratio $c_r \in \{0.25, 0.5, 0.75, 1\}$, i.e., $r \in \{2025, 4050, 6075, 8100\}$. In Fig. 10, we show the estimates of $\overline{C}$ using both algorithms. The numerical results obtained in Figs. 9 and 10 clearly demonstrate the advantages of the proposed algorithm. Similar to previous examples in Sections 5.1 and 5.2 , both methods achieve the same MSE, but the execution time associated with the fast-GLRMA method is faster than that of the GRLMA method.

6. Conclusions

In this paper, we proposed a new faster method to compute a solution for the GLRMA problem, so-called the fast-GLRMA method. This new technique is based on tensor product and Tikhonov's regularization to approximate the pseudoinverse, and bilateral random projections to estimate, in turn, the low-rank approximation.

Additionally, we presented an alternative approach in Theorems 3 and 4 to compute the low-rank matrix approximation, using partial information of an orthogonal basis of the full-rank random projection Y_2. This alternative method is so-called the modified bilateral random projections (MBRP) method. Also, in Theorem 5, we presented a theoretical error analysis of the MBRP method when $\text{rank}(Y_2) = r$.

The fast-GLRMA method significantly reduces the execution time in computing the optimal solution and increases speedup, preserving the accuracy of the classical method to solve the GLRMA. The theoretical development and numerical examples in this paper demonstrate the advantages of the proposed fast-GLRMA method.

[3] The structural similarity (SSIM) index is a method for predicting the perceived quality of digital images and videos. The resultant SSIM index is a decimal value between −1 and 1, and value 1 is only reachable in the case of two identical sets of data. More details are available in [64].

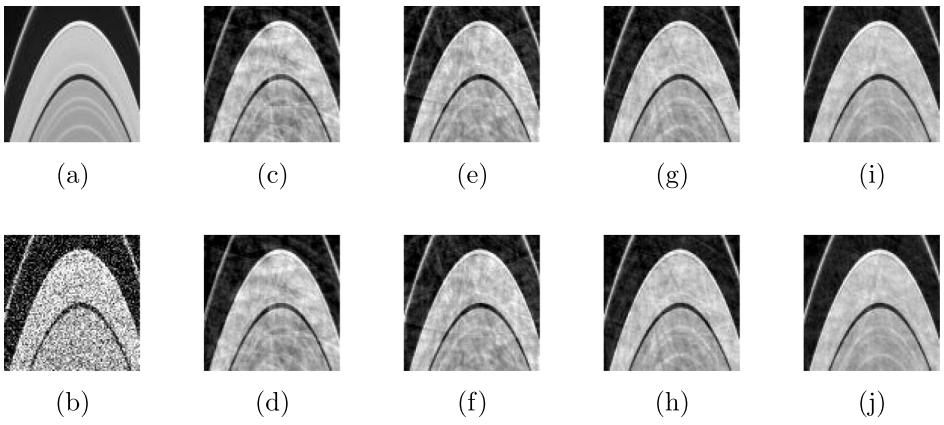


Fig. 10. Illustration of the estimation of noisy image $\overline{C}$ by GLRMA and fast-GLRMA methods. (a) Source image. (b) Noisy observed image $\overline{C}$. (c)–(d) Estimation using the GLRMA and fast-GLRMA methods, respectively, for $c_r = 0.25$. (e)–(f) Estimation using the GLRMA and fast-GLRMA methods, respectively, for $c_r = 0.5$. (g)–(h) Estimation using the GLRMA and fast-GLRMA methods, respectively, for $c_r = 0.75$. (i)–(j) Estimation using the GLRMA and fast-GLRMA methods, respectively, for $c_r = 1$.

CRediT authorship contribution statement

Jeffry Chavarría-Molina: Investigation, Computational implementation, Validation in section 3.2, Theoretical development of theorem 4. **Juan José Fallas-Monge:** Investigation, Computational implementation, Validation in section 3.1. **Pablo Soto-Quiros:** General investigation, Supervision, Writing – original draft, Writing – reviewing and editing, Computational implementation of numerical examples in section 5, Theoretical development of theorem 3 and 5, and proposition 5.

Acknowledgments

The author thanks Professor Anatoli Torokhti for his comments on this manuscript, and the reviewers of the paper for their insightful comments. This work was financially supported by *Vicerrectoría de Investigación y Extensión* from Instituto Tecnológico de Costa Rica (Research #1440037).

Appendix

The following propositions will be used in the proof of Theorem 5.

Proposition 1 (*Fact 9.9.10 in [61]*)**.** *If $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times q}$, then $\|AB\|_{fr}^2 \leq \sigma_{max}^2(A)\|B\|_{fr}^2$, where $\sigma_{max}^2(A)$ is the largest singular value of A.*

Proposition 2 (*Fact 6.1.6 in [61]*)**.** *If $A \in \mathbb{R}^{n\times p}$, then $AA^\dagger$ is a symmetric projector onto $\mathcal{R}(A)$. Moreover, if $\operatorname{rank}(A) = r$, then $\operatorname{rank}(I_n - AA^\dagger) = n - r$.*

Proposition 3 (*Theorem E.3.2.0.1 in [65]*)**.** *If $P \in \mathbb{R}^{n\times n}$ is a symmetric projector, then $\|I_n - P\|_{fr}^2 = \operatorname{rank}(I_n - P)$.*

Proposition 4. *If $A \in \mathbb{R}^{m\times n}$ and $\operatorname{rank}(A) = r$, then $\|I_n - AA^\dagger\|_{fr}^2 = n - r$.*

Proof. It follows from Propositions 2 and 3. □

Proposition 5. *Let $L_r \in \mathbb{R}^{m\times n}$ be the low-rank approximation matrix of $L \in \mathbb{R}^{m\times n}$. Let $\widetilde{L}_r = LY_2Y_2^\dagger$ be the estimation of L_r given by Theorem 3. If $r = \operatorname{rank}(Y_2)$, then $\|L - \widetilde{L}_r\|_{fr}^2 \leq \sigma_{max}^2(L)(n - r)$.*

Proof. It follows from Propositions 1 and 4 that

$$\begin{aligned}\|L - \widetilde{L}_r\|_{fr}^2 &= \|L - LY_2Y_2^\dagger\|_{fr}^2\\ &= \|L(I_n - Y_2Y_2^\dagger)\|_{fr}^2\\ &\leq \sigma_{max}^2(L)\|I_n - Y_2Y_2^\dagger\|_{fr}^2\\ &= \sigma_{max}^2(L)(n - r)\end{aligned}$$ □

The proof of Theorem 5 is given below.

Proof. It follows from the triangle inequality that

$$\|L_r - \widetilde{L}_r\|_{fr}^2 = \|(L_r - L) + (L - \widetilde{L}_r)\|_{fr}^2 \leq \|L - L_r\|_{fr}^2 + \|L - \widetilde{L}_r\|_{fr}^2 \tag{21}$$

and

$$\|L - L_r\|_{fr}^2 = \|(L - \widetilde{L}_r) + (\widetilde{L}_r - L_r)\|_{fr}^2 \leq \|L - \widetilde{L}_r\|_{fr}^2 + \|L_r - \widetilde{L}_r\|_{fr}^2. \tag{22}$$

From (22), we obtain that

$$\|L - L_r\|_{fr}^2 - \|L - \widetilde{L}_r\|_{fr}^2 \leq \|L_r - \widetilde{L}_r\|_{fr}^2 \tag{23}$$

It is a well known fact that $\|L - L_r\|_{fr}^2 = \sum_{i=r+1}^{\min\{m,n\}} \sigma_i^2(L)$ (see, e.g., [1]). Let us define $\alpha = \sum_{i=r+1}^{\min\{m,n\}} \sigma_i^2(L)$ and $\beta = \sigma_{max}^2(L)(n-r)$. It follows from Proposition 5 and Eqs. (21) and (23) that

$$\alpha - \beta \leq \|L_r - \widetilde{L}_r\|_{fr}^2 \leq \alpha + \beta. \tag{24}$$

Finally, it follows from Eq. (24) and the fact that $0 \leq \|L_r - \widetilde{L}_r\|_{fr}^2$ that

$$\|L_r - \widetilde{L}_r\|_{fr}^2 \in [\max\{0, \alpha - \beta\}, \alpha + \beta]. \quad \square$$

References

[1] C. Eckart, G. Young, The approximation of one matrix by another of lower rank, Psychometrika 1 (3) (1936) 211–218.
[2] I. Markovsky, Low Rank Approximation: Algorithms, Implementation, Applications, in: Communications and Control Engineering, Springer London, 2011.
[3] F. Fritzen, D. Ryckelynck, Machine Learning, Low-Rank Approximations and Reduced Order Modeling in Computational Mechanics, MDPI AG, 2019.
[4] D. Sondermann, Best approximate solutions to matrix equations under rank restrictions, Stat. Hefte 27 (1) (1986) 57–66.
[5] S. Friedland, A. Torokhti, Generalized rank-constrained matrix approximations, SIAM J. Matrix Anal. Appl. 29 (2) (2007) 656–659.
[6] A. Torokhti, S. Friedland, Towards theory of generic principal component analysis, J. Multivariate Anal. 100 (4) (2009) 661–669.
[7] X. Liu, L. Luo, Minimum rank positive semidefinite solution to the matrix approximation problem in the spectral norm, Appl. Math. Lett. 107 (2020).
[8] H. Wang, Rank constrained matrix best approximation problem, Appl. Math. Lett. 50 (2015) 98–104.
[9] M. Wei, Q. Wang, On rank-constrained Hermitian nonnegative-definite least squares solutions to the matrix equation $AXA^H = B$, Int. J. Comput. Math. 84 (6) (2007) 945–952.
[10] H. Wang, Least squares solutions to the rank-constrained matrix approximation problem in the Frobenius norm, Calcolo 56 (4) (2019) 1–18.
[11] P. Mineiro, N. Karampatziakis, Fast label embeddings via randomized linear algebra, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2015, pp. 37–51.
[12] F. Zhao, M. Xiao, Y. Guo, Predictive collaborative filtering with side information, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI'16, AAAI Press, 2016, pp. 2385–2390.
[13] F. Zhao, Learning Top-N Recommender Systems with Implicit Feedbacks, Temple University, 2017.
[14] J. Chung, M. Chung, D. O'Leary, Optimal regularized low rank inverse approximation, Linear Algebra Appl. 468 (2015) 260–269.
[15] J. Chung, M. Chung, An efficient approach for computing optimal low-rank regularized inverse matrices, Inverse Problems 30 (11) (2014) 114009.
[16] J. Chung, M. Chung, Optimal regularized inverse matrices for inverse problems, SIAM J. Matrix Anal. Appl. 38 (2) (2017) 458–477.
[17] J. Chung, M. Chung, Computing optimal low-rank matrix approximations for image processing, in: 2013 Asilomar Conference on Signals, Systems and Computers, 2013, pp. 670–674.
[18] A. Torokhti, P. Soto-Quiros, Generalized Brillinger-like transforms, IEEE Signal Process. Lett. 23 (6) (2016) 843–847.
[19] P. Soto-Quiros, A. Torokhti, Improvement in accuracy for dimensionality reduction and reconstruction of noisy signals. Part II: The case of signal samples, Signal Process. 154 (2019) 272–279.
[20] Y. Hua, W. Liu, Generalized Karhunen-Loève transform, IEEE Signal Process. Lett. 5 (6) (1998) 141–142.
[21] S. Chen, F. X., H. Zhang, Visual-SLIM: Integrated sparse linear model with visual features for personalized recommendation, in: Pacific Rim Conference on Multimedia, Springer, 2018, pp. 126–135.
[22] M. Holmes, J. Isbell, C. Lee, A. Gray, QUIC-SVD: Fast SVD using cosine trees, in: Advances in Neural Information Processing Systems, 2009, pp. 673–680.
[23] T. Wu, S. Sarmadi, V. Venkatasubramanian, A. Pothen, A. Kalyanaraman, Fast SVD computations for synchrophasor algorithms, IEEE Trans. Power Syst. 31 (2) (2015) 1651–1652.
[24] Z. Allen-Zhu, Y. Li, LazySVD: Even faster SVD decomposition yet without agonizing pain, in: Advances in Neural Information Processing Systems, 2016, pp. 974–982.
[25] G.H. Golub, C.F. Van Loan, Matrix Computations, Johns Hopkins University Press, 2012.
[26] P. Courrieu, Fast computation of Moore-Penrose inverse matrices, Neural Inf. Process. Lett. Rev 8 (2) (2005).
[27] F. Girosi, T. Poggio, Networks and the best approximation property, Biol. Cybernet. 63 (3) (1990) 169–176.
[28] T. Poggio, F. Girosi, Networks for approximation and learning, Proc. IEEE 78 (9) (1990) 1481–1497.
[29] F. Shang, L. Jiao, J. Shi, M. Gong, R. Shang, Fast density-weighted low-rank approximation spectral clustering, Data Min. Knowl. Discov. 23 (2) (2011) 345–378.
[30] V. Katsikis, D. Pappas, Fast computing of the Moore-Penrose inverse matrix, Electron. J. Linear Algebra 17 (2008) 637–650.
[31] S. Lu, X. Wang, G. Zhang, X. Zhou, Effective algorithms of the Moore-Penrose inverse matrices for extreme learning machine, Intell. Data Anal. 19 (4) (2015) 743–760.
[32] S. Miljković, M. Miladinović, P. Stanimirović, I. Stojanović, Application of the pseudoinverse computation in reconstruction of blurred images, Filomat 26 (3) (2012) 453–465.

[33] T. Zhou, D. Tao, GoDec: Randomized low-rank & sparse matrix decomposition in noisy case, in: Proceedings of the 28th International Conference on Machine Learning, ICML 2011, 2011, pp. 33–40.
[34] J.C.A. Barata, M.S. Hussein, The Moore-Penrose pseudoinverse: A tutorial review of the theory, Braz. J. Phys. 42 (1) (2012) 146–165.
[35] M. Fazel, E. Candes, B. Recht, P. Parrilo, Compressed sensing and robust recovery of low rank matrices, in: 2008 42nd Asilomar Conference on Signals, Systems and Computers, IEEE, 2008, pp. 1043–1047.
[36] T. Zhou, D. Tao, Bilateral random projections, in: 2012 IEEE International Symposium on Information Theory Proceedings, IEEE, 2012, pp. 1286–1290.
[37] M. Brand, Fast low-rank modifications of the thin singular value decomposition, Linear Algebra Appl. 415 (1) (2006) 20–30.
[38] B. Telfer, D. Casasent, Fast method for updating robust pseudoinverse and Ho-Kashyap associative processors, IEEE Trans. Syst. Man Cybern. 24 (9) (1994) 1387–1390.
[39] M. Benson, P. Frederickson, Fast Parallel Algorithms for the Moore-Penrose Pseudo-Inverse, Tech. Rep., Los Alamos National Lab., NM (USA); Lakehead Univ., Thunder Bay, Ontario, 1986.
[40] G. Schulz, Iterative Berechnung der Reziproken matrix, J. Appl. Math. Mech. 13 (1933) 57–59.
[41] H. Chen, Y. Wang, A family of higher-order convergent iterative methods for computing the Moore-Penrose inverse, Appl. Math. Comput. 218 (8) (2011) 4012–4016.
[42] A. Ataei, Improved Qrginv algorithm for computing Moore-Penrose inverse matrices, Int. Sch. Res. Not. 2014 (2014).
[43] S. Artidiello, A. Cordero, J.R. Torregrosa, M. Vassileva, Generalized inverses estimations by means of iterative methods with memory, Mathematics 8 (1) (2020) 2.
[44] A. Cordero, P. Soto-Quiros, J.R. Torregrosa, A general class of arbitrary order iterative methods for computing generalized inverses, Appl. Math. Comput. 409 (2021) 126381.
[45] A. Deshpande, S. Vempala, Adaptive sampling and fast low-rank matrix approximation, in: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, Springer, 2006, pp. 292–303.
[46] P. Drineas, R. Kannan, M. Mahoney, Fast Monte-Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix, SIAM J. Comput. 36 (1) (2006) 158–183.
[47] A. Frieze, R. Kannan, S. Vempala, Fast Monte-Carlo algorithms for finding low-rank approximations, J. ACM 51 (6) (2004) 1025–1041.
[48] N. Nguyen, T. Do, T. Tran, A fast and efficient algorithm for low-rank approximation of a matrix, in: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, 2009, pp. 215–224.
[49] D. Achlioptas, F. Mcsherry, Fast computation of low-rank matrix approximations, J. ACM 54 (2) (2007) 611–618.
[50] B. Li, Z. Yang, L. Zhi, Fast low rank approximation of a Sylvester matrix by structured total least norm, J. Jpn. Soc. Symb. Algebraic Comput. 11 (3) (2005) 165–174.
[51] M. Belabbas, P. Wolfe, Fast low-rank approximation for covariance matrices, in: 2007 2nd IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing, 2007, pp. 293–296.
[52] G. Xie, K. Xie, J. Huang, X. Wang, Y. Chen, J. Wen, Fast low-rank matrix approximation with locality sensitive hashing for quick anomaly detection, in: IEEE INFOCOM 2017-IEEE Conference on Computer Communications, 2017, pp. 1–9.
[53] N. Halko, P. Martinsson, J.A. Tropp, Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, SIAM Rev. 53 (2) (2011) 217–288.
[54] B. Luong, MMTimes: Matrix chain product, 2020, https://www.mathworks.com/matlabcentral/fileexchange/27950-mmtimes-matrix-chain-product. Online; [Accessed 3 August 2020].
[55] T. Cormen, C. Leiserson, R. Rivest, C. Stein, Introduction to Algorithms, MIT press, 2009.
[56] J. Trahan, A. Kaw, K. Martin, Computational time for finding the inverse of a matrix: LU decomposition vs. naive Gaussian elimination, in: Solution of Simultaneous Linear Equations, 2006, pp. 1–8.
[57] MathWorks, Command `pinv`, 2020, https://www.mathworks.com/help/matlab/ref/pinv.html. Online; [Accessed 3 August 2020].
[58] A. Ben-Israel, An iterative method for computing the generalized inverse of an arbitrary matrix, Math. Comp. (1965) 452–455.
[59] W. Li, Z. Li, A family of iterative methods for computing the approximate inverse of a square matrix and inner inverse of a non-square matrix, Appl. Math. Comput. 215 (9) (2010) 3433–3442.
[60] M. Kaur, M. Kansal, S. Kumar, An efficient hyperpower iterative method for computing weighted Moore-Penrose inverse, AIMS Math. 5 (3) (2020) 1680–1692.
[61] D.S. Bernstein, Matrix Mathematics: Theory, Facts, and Formulas, second ed., in: Princeton Reference, Princeton University Press, 2009.
[62] H. Cragon, Computer Architecture and Implementation, Cambridge University Press, 2000.
[63] NASA, NASA solar system exploration database, 2020, https://solarsystem.nasa.gov/raw-images/raw-image-viewer. Online; [Accessed 10 September 2020].
[64] S. Channappayya, A. Bovik, R. Heath, Rate bounds on SSIM index of quantized images, IEEE Trans. Image Process. 17 (9) (2008) 1624–1639.
[65] J. Dattorro, Convex Optimization † Euclidean Distance Geometry, Meboo Publishing USA, 2019.

Anexo 6

Electronic Journal of Linear Algebra, ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 37, pp. 544-548, July 2021.

ERROR ANALYSIS OF THE GENERALIZED LOW-RANK MATRIX APPROXIMATION*

PABLO SOTO-QUIROS†

Abstract. In this paper, we propose an error analysis of the generalized low-rank approximation, which is a generalization of the classical approximation of a matrix $A \in \mathbb{R}^{m\times n}$ by a matrix of a rank at most r, where $r \leq \min\{m, n\}$.

Key words. Generalized low-rank approximation, Frobenius norm, SVD, Pseudoinverse, Error analysis.

AMS subject classifications. 15A18, 15A29.

1. Introduction. Throughout this paper, we adopt the following notation: $\mathbb{R}_r^{m\times n}$ denotes the set of all real $m \times n$ matrices of rank at most r, where $r \leq \min\{m, n\}$, i.e., $A \in \mathbb{R}_r^{m\times n}$ if and only if $\text{rank}(A) \leq r$. $I_m \in \mathbb{R}^{m\times m}$ and $\boldsymbol{0}_{m\times n} \in \mathbb{R}^{m\times n}$ are the identity matrix of order m and the null $m \times n$ matrix, respectively. $M^\dagger$, $\text{tr}\{M\}$, $\|M\|$ and $\mathcal{N}(M)$ denote the pseudoinvserse, the trace, the Frobenius norm and the null space of M, respectively. Furthermore, $N^{1/2}$ is the square root of $N \in \mathbb{R}^{m\times m}$, i.e., $N = N^{1/2}N^{1/2}$.

Let $D = U_D\Sigma_D V_D^T$ be the singular value decomposition (SVD) of $D \in \mathbb{R}^{m\times n}$, where $U_D \in \mathbb{R}^{m\times m}$ and $V_D \in \mathbb{R}^{n\times n}$ are two orthogonal matrices, and $\Sigma_D = \text{diag}(\sigma_1(D), ..., \sigma_{\min(m,n)}(D)) \in \mathbb{R}^{m\times n}$ is a generalized diagonal matrix with singular values $\sigma_1(D) \geq \sigma_2(D) \geq ... \geq \sigma_{\min(m,n)}(D) \geq 0$ on the main diagonal. The r-truncated SVD is defined by

$$\lfloor D \rfloor_r = \sum_{i=1}^{r} \sigma_i(D) u_i v_i^T = U_{D,r}\Sigma_{D,r}V_{D,r}^T \in \mathbb{R}^{m\times n},$$

where $U_{D,r} \in \mathbb{R}^{m\times r}$ and $V_{D,r} \in \mathbb{R}^{n\times r}$ are formed with the first r columns of U_D and V_D, respectively, and $\Sigma_{D,r} = \text{diag}(\sigma_1(D), \ldots, \sigma_r(D)) \in \mathbb{R}^{r\times r}$. If $k = \text{rank}(D)$, then $P_{D,L} \in \mathbb{R}^{m\times m}$ and $P_{D,R} \in \mathbb{R}^{n\times n}$ are the orthogonal projections of D on the range of D and D^T, respectively, where $P_{D,L} = DD^\dagger = U_{D,k}U_{D,k}^T$ and $P_{D,R} = D^\dagger D = V_{D,k}V_{D,k}^T$.

A generalization of the low-rank approximation was proposed in [2, 3, 4]. Given matrices $A \in \mathbb{R}^{p\times q}$, $B \in \mathbb{R}^{p\times m}$ and $C \in \mathbb{R}^{n\times q}$, and rank $r \leq \min\{m, n\}$, the generalized low-rank approximation finds a matrix $\widehat{X}_r \in \mathbb{R}_r^{m\times n}$ such that

$$\|A - B\widehat{X}_r C\|^2 = \min_{X\in\mathbb{R}_r^{m\times n}} \|A - BXC\|^2 . \tag{1.1}$$

Note that if B and C are identity matrices, the problem (1.1) is the well-known low-rank approximation problem proposed by Eckart and Young [1]. The problem in (1.1) was studied in [2, 3, 4] by Sonderman, Friedland and Torokthi, respectively. The following theorem presents the solution of the generalized low-rank approximation given in [3].

*Received by the editors on March 4, 2021. Accepted for publication on June 27, 2021. Handling Editor: Shmuel Friedland.
†Escuela de Matemática, Instituto Tecnológico de Costa Rica, Cartago 30101, Costa Rica (jusoto@tec.ac.cr).

Electronic Journal of Linear Algebra, ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 37, pp. 544-548, July 2021.

THEOREM 1.1. *Let $A \in \mathbb{R}^{p\times q}$, $B \in \mathbb{R}^{p\times m}$ and $C \in \mathbb{R}^{n\times q}$, and let $M = U_M\Sigma_M V_M^T$ be the SVD of $M = P_{B,L}AP_{C,R} = BB^\dagger AC^\dagger C$. Then matrix*

$$\widehat{X}_r = B^\dagger \lfloor M \rfloor_r C^\dagger = B^\dagger U_{M,r}\Sigma_{M,r}V_{M,r}^T C^\dagger, \tag{1.2}$$

minimizes the problem (1.1). This solution is unique if and only if either

$$r \geq \operatorname{rank}(M),$$

or

$$1 \leq r < \operatorname{rank}(M) \quad \textit{and} \quad \sigma_r(M) \geq \sigma_{r+1}(M).$$

In this paper, we present an error analysis for the solution of the problem (1.1). The main result of this paper is presented below, in Theorem 3.1. In the related work, for the same problem, in [5, Theorem 3.2] Wang presented an error analysis for the specific case of $\operatorname{rank}(\widehat{X}_r) = r$. In this paper, we extend the analysis to the case $\operatorname{rank}(\widehat{X}_r) \leq r$. The formula for the error in [5, Theorem 3.2] is different from that in Theorem 3.1.

2. Preliminaries. In this section, we present some preliminary results that will be used in the next section to study the error associated with the solution of the problem (1.1).

LEMMA 2.1 (Theorem 2.8 in [6]). *If $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times m}$, then AB and BA have the same nonzero eigenvalues, counting multiplicity.*

LEMMA 2.2 (Propositions 3.1 and 3.2 in [7]). *$A^\dagger = A^T(A^\dagger)^T A^\dagger = A^\dagger(A^\dagger)^T A^T = (A^TA)^\dagger A^T$, for all $A \in \mathbb{R}^{m\times n}$.*

LEMMA 2.3 (Lemma 2.4.1 in [8]). *Let $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times m}$. Then $\mathcal{N}(A) \subseteq \mathcal{N}(BA)$.*

LEMMA 2.4 (Lemma 23 in [9] - Fact 2 in [10]). *For any $M \in \mathbb{R}^{m\times n}$, $N \in \mathbb{R}^{p\times n}$ and $S \in \mathbb{R}^{m\times s}$, the following statements hold.*

(a) *If $\mathcal{N}(M) \subseteq \mathcal{N}(N)$, then $NM^\dagger M = N$.*
(b) *If $\mathcal{N}(M^T) \subseteq \mathcal{N}(S^T)$, then $MM^\dagger S = S$.*

LEMMA 2.5. *Let $M \in \mathbb{R}^{m\times n}$ and $r \leq \min\{m,n\}$. Then $\mathcal{N}(M) \subseteq \mathcal{N}(\lfloor M \rfloor_r)$.*

Proof. Without loss of generality, we assume $m \leq n$. Let $M = U_M\Sigma_M V_M^T$ be the SVD of M. If $x \in \mathcal{N}(M)$, then $Mx = \boldsymbol{0}_{n\times 1}$, and therefore, $\Sigma_M V_M^T x = \boldsymbol{0}_{n\times 1}$, because U_M is an orthogonal matrix. If $\overline{\Sigma}_{M,r} = \operatorname{diag}(\sigma_1(M), ..., \sigma_r(M), 0, ..., 0) \in \mathbb{R}^{m\times n}$, then

$$\begin{aligned}\Sigma_M V_M^T x = \boldsymbol{0}_{n\times 1} &\Rightarrow \begin{bmatrix} I_r & \boldsymbol{0}_{k\times m-r} \\ \boldsymbol{0}_{m-r\times r} & \boldsymbol{0}_{m-r\times m-r}\end{bmatrix}\Sigma_M V_M^T x = \boldsymbol{0}_{n\times 1} \\ &\Rightarrow \overline{\Sigma}_{M,r}V_M^T x = \boldsymbol{0}_{n\times 1} \\ &\Rightarrow U_M\overline{\Sigma}_{M,r}V_M^T x = \boldsymbol{0}_{n\times 1}.\end{aligned}$$

Note that

$$U_M\overline{\Sigma}_{M,r}V_M^T = \sum_{i=1}^{\min\{m,n\}} \sigma_i(M)u_i v_i^T = \sum_{i=1}^{r}\sigma_i(M)u_i v_i^T = \lfloor M \rfloor_r.$$

Finally, $\lfloor M \rfloor_r x = \boldsymbol{0}_{n\times 1}$. Thus, $x \in \mathcal{N}(\lfloor M \rfloor_r)$. ∎

Electronic Journal of Linear Algebra, ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 37, pp. 544-548, July 2021.

LEMMA 2.6. *If $M \in \mathbb{R}^{m\times n}$, $S \in \mathbb{R}^{n\times s}$ and $R \in \mathbb{R}^{s\times m}$, then the following statements hold.*

(a) $\lfloor RM \rfloor_r M^\dagger M = \lfloor RM \rfloor_r$.
(b) $SS^\dagger \lfloor SR \rfloor_r = \lfloor SR \rfloor_r$.

Proof. We consider (a). It follows from Lemma 2.3 that $\mathcal{N}(M) \subseteq \mathcal{N}(RM)$. By Lemma 2.5, we obtain $\mathcal{N}(RM) \subseteq \mathcal{N}(\lfloor RM \rfloor_r)$, and therefore, $\mathcal{N}(M) \subseteq \mathcal{N}(\lfloor RM \rfloor_r)$. As a result, (a) follows from Lemma 2.4. The proof of (b) is similar to the proof of (a). ∎

3. Main Result.

THEOREM 3.1. *Let $A \in \mathbb{R}^{p\times q}$, $B \in \mathbb{R}^{p\times m}$, $C \in \mathbb{R}^{n\times q}$ and $r \leq \min\{m, n\}$. The error of the solution of the problem (1.1) is given by*

$$\min_{X\in\mathbb{R}_r^{m\times n}} \|A - BXC\|^2 = \|A\|^2 - \sum_{i=1}^{r} \lambda_i(T), \tag{3.3}$$

where $T = B^\dagger AC^\dagger CA^T B \in \mathbb{R}^{m\times m}$ and $\lambda_i(T)$ is the $i-$th eigenvalue of T with $\lambda_1(T) \geq \lambda_2(T) \geq ... \geq \lambda_m(T)$.

Proof. It follows from the identity $||D||^2 = \text{tr}\{DD^T\}$ and the linearity of the trace operator that

$$\|A - BXC\|^2 = \|A\|^2 - \|M\|^2 + \|M - (B^TB)^{1/2}X(CC^T)^{1/2}\|^2, \tag{3.4}$$

where $M = (B^TB)^{1/2\dagger}B^TAC^T(CC^T)^{1/2\dagger}$. Note that $\|M - (B^TB)^{1/2}X(CC^T)^{1/2}\|^2$ is the only term in (3.4) that depends on X. Therefore, problem (1.1) is equivalent to

$$\min_{X\in\mathbb{R}_r^{m\times n}} \|M - (B^TB)^{1/2}X(CC^T)^{1/2}\|^2. \tag{3.5}$$

Based on Theorem 1.1, the solution of the problem (3.5) is given by

$$\widehat{X}_r = (B^TB)^{1/2\dagger} \left\lfloor P_{(B^TB)^{1/2},L} M P_{(CC^T)^{1/2},R} \right\rfloor_r (CC^T)^{1/2\dagger}. \tag{3.6}$$

Note that $(B^TB)^{1/2\dagger}$ and $(CC^T)^{1/2\dagger}$ are symmetric matrices. Therefore, it follows from Lemma 2.2 that

$$P_{(B^TB)^{1/2},L}(B^TB)^{1/2\dagger} = (B^TB)^{1/2}(B^TB)^{1/2\dagger}(B^TB)^{1/2\dagger} = (B^TB)^{1/2\dagger}, \tag{3.7}$$

and

$$(CC^T)^{1/2\dagger}P_{(CC^T)^{1/2},R} = (CC^T)^{1/2\dagger}(CC^T)^{1/2\dagger}(CC^T)^{1/2} = (CC^T)^{1/2\dagger}. \tag{3.8}$$

Further, (3.7) and (3.8) imply

$$\begin{aligned} \lfloor P_{(B^TB)^{1/2},L} M P_{(CC^T)^{1/2},R} \rfloor_r &= \lfloor P_{(B^TB)^{1/2},L}(B^TB)^{1/2\dagger}B^TAC^T(CC^T)^{1/2\dagger}P_{(CC^T)^{1/2},R} \rfloor_r \\ &= \lfloor (B^TB)^{1/2\dagger}B^TAC^T(CC^T)^{1/2\dagger} \rfloor_r \\ &= \lfloor M \rfloor_r. \end{aligned} \tag{3.9}$$

It follows from (3.6) and (3.9) that $\widehat{X}_r = (B^TB)^{1/2\dagger} \lfloor M \rfloor_r (CC^T)^{1/2\dagger}$. On the basis of Lemma 2.6, we obtain that

$$(B^TB)^{1/2}\widehat{X}_r(CC^T)^{1/2} = (B^TB)^{1/2}(B^TB)^{1/2\dagger} \lfloor M \rfloor_r (CC^T)^{1/2\dagger}(CC^T)^{1/2} = \lfloor M \rfloor_r. \tag{3.10}$$

Electronic Journal of Linear Algebra, ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 37, pp. 544-548, July 2021.

By equations (3.4) and (3.10) and the facts that $\|M\|^2 = \mathrm{tr}\{MM^T\} = \sum_{i=1}^{m} \lambda_i(MM^T)$ and $\|\cdot\|$ is unitary invariant [6], we obtain the following identity

$$\begin{aligned}\min_{X\in\mathbb{R}_r^{p\times q}} \|A - BXC\|^2 &= \|A\|^2 - \|M\|^2 + \|M - (B^TB)^{1/2}\widehat{X}_r(CC^T)^{1/2}\|^2 \\ &= \|A\|^2 - \|M\|^2 + \|M - \lfloor M\rfloor_r\|^2 \\ &= \|A\|^2 - \sum_{i=1}^{m}\lambda_i(MM^T) + \sum_{i=r+1}^{m}\lambda_i(MM^T) \\ &= \|A\|^2 - \sum_{i=1}^{r}\lambda_i(MM^T). \qquad (3.11)\end{aligned}$$

Note that $MM^T = (B^TB)^{1/2\dagger}B^TAC^\dagger CA^TB(B^TB)^{1/2\dagger}$. From Lemmas 2.1 and 2.2, we obtain that

$$\lambda_i(MM^T) = \lambda_i(B^\dagger AC^\dagger CA^TB), \qquad (3.12)$$

for all $i = 1, ..., m$. Finally, (3.3) follows from (3.11) and (3.12). ∎

4. Advantage and Numerical Example. The error formula given by (3.3) is useful for choosing an optimal value for the rank r, before computing the matrix $\widehat{X}_r$ in (1.2). For example, we consider matrices $A \in \mathbb{R}^{20\times 35}$, $B \in \mathbb{R}^{20\times 30}$ and $C \in \mathbb{R}^{40\times 35}$ generated from a uniform distribution with zero-mean and standard deviation 1. Figure 1 shows the relationship between the error associated with the solution of problem (1.1) and the rank r. It follows from Figure 1 that the associated error is 0 when $r \geq 20$. Therefore, the smallest value of r that implies the minimal error of the solution of problem (1.1) is given by $r = 20$. Note that it was not necessary to compute each optimal matrix $\widehat{X}_r$, for $r = 1, ..., 30$, to obtain the associated error. In this example, we only compute the eigenvalues of $T = B^\dagger AC^\dagger CA^TB \in \mathbb{R}^{30\times 30}$ and use the formula (3.3). Furthermore, in this numerical simulation, we obtain that $\mathrm{rank}(T) = 20$ and $\sigma_i^2(A) = \lambda_i(T)$, for all $i = 1, ..., 20$. Thus,

$$\min_{X\in\mathbb{R}_r^{30\times 40}} \|A - BXC\|^2 = \|A\|^2 - \sum_{i=1}^{r}\lambda_i(T) = 0,$$

for all $r \geq 20$.

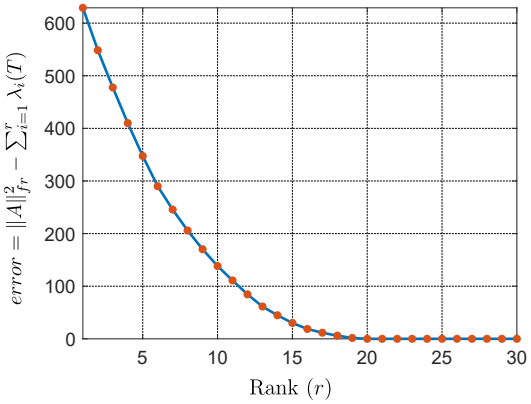


FIGURE 1. *Diagram of the error associated with the solution of problem* (1.1) *versus rank* r.

Electronic Journal of Linear Algebra, ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 37, pp. 544-548, July 2021.

Acknowledgments. The author thanks Juan José Fallas Monge for his comments on this manuscript and the reviewer of the paper for their insightful comments. This work was financially supported by *Vicerrectoría de Investigación y Extensión* from *Instituto Tecnológico de Costa Rica* (Research #1440037).

REFERENCES

[1] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.
[2] D. Sondermann. Best approximate solutions to matrix equations under rank restrictions. *Statistische Hefte*, 27(1):57–66, 1986.
[3] S. Friedland and A. Torokhti. Generalized rank-constrained matrix approximations. *SIAM J. Matrix Anal. Appl.*, 29(2):656–659, 2007.
[4] A. Torokhti and S. Friedland. Towards theory of generic principal component analysis. *J. Multivar. Anal.*, 100(4):661–669, 2009.
[5] H. Wang. Rank constrained matrix best approximation problem. *Appl. Math. Lett.*, 50:98–104, 2015.
[6] F. Zhang. *Matrix Theory: Basic Results and Techniques*. Springer Science & Business Media, 2011.
[7] J.C.A. Barata and M.S. Hussein. The MoorePenrose pseudoinverse: A tutorial review of the theory. *Braz. J. Phys.*, 42(1–2):146–465, 2012.
[8] D.S. Bernstein. *Matrix Mathematics: Theory, Facts, and Formulas*, 2nd edition. Princeton University Press, 2009.
[9] A. Torokhti and P. Howlett. *Computational Methods for Modeling of Nonlinear Systems* (Mathematics in Science and Engineering). Elsevier Science, 2007.
[10] A. Torokhti and P. Soto-Quiros. Generalized Brillinger-like transforms. *IEEE Sig. Process. Lett.*, 23(6):843–847, 2016.

Anexo 7

Data Compression: Multi-Term Approach

Pablo Soto-Quiros*,† and Anatoli Torokhti*
*Centre for Industrial and Applied Mathematics, University of South Australia, Adelaide, SA 5095, Australia
†Escuela de Matemáticas, Instituto Tecnológico de Costa Rica, Cartago 30101, Costa Rica
Emails: {juan.soto-quiros, anatoli.torokhti}@unisa.edu.au

***Abstract*—In terms of signal samples, we propose and justify a new rank reduced multi-term transform, abbreviated as MTT, which, under certain conditions, may provide the better associated accuracy than that of known optimal rank reduced transforms. The basic idea is to construct the transform with more parameters to optimize than those in the known optimal transforms. This is realized by the extension of the known transform structures to the form that includes additional terms - the MTT has four matrices to minimize the cost. The MTT structure has also a special transformation that decreases the numerical load. As a result, the MTT performance is improved by the variation of the MTT components.**

I. Introduction

Data compression is one of the central and widely studied problems in communication and signal processing. Assume, a reference signal and a noisy observed signal are represented by random vectors $\mathbf{x} \in L^2(\Omega, \mathbb{R}^m)$ and $\mathbf{y} \in L^2(\Omega, \mathbb{R}^n)$, respectively[1]. Then $\mathbf{y}$ compression (and denoising) to a 'shorter' (or compressed) vector $\mathbf{u} \in L^2(\Omega, \mathbb{R}^k)$ where $k \leq \min\{m, n\}$, and its consequent reconstruction (or decompression) can be represented as $F = DC$, where C and D are a 'compressor' and 'de-compressor', respectively.

An optimal determination of the compressor and de-compressor in the form of matrices $C \in \mathbb{R}^{k \times n}$ and $D \in \mathbb{R}^{m \times k}$, respectively, is given by the transform by Brillinger (BT) [1], the generalized Brillinger transform (GBT1) [2] and the generic Karhunen-Loève transform (GKLT) [3], [4]. Under some assumptions, the BT, GKLT and GBT1 provide the best associated accuracy among all linear transforms of the form $F = DC$, for the same compression ratio $c = k/\min\{m, n\}$, where $k = 1, 2, \ldots, \min\{m, n\}$. The second degree transforms developed in [2], [5] (and abbreviated here as the GBT2 and GKLT2, respectively), under the certain condition, improve the GKLT and GBT1 accuracy, for the same compression ratio. Nevertheless, it may happen that the accuracy is still not satisfactory.

In the practical setting or in the training stage, signals are replaced with their samples. Here, we consider the case. Sample matrices of $\mathbf{x}$ and $\mathbf{y}$ are denote by $X \in \mathbb{R}^{m \times s}$ and $Y \in \mathbb{R}^{n \times s}$, respectively, where s is a number of samples.

In terms of the signal samples, we propose and justify a new rank reduced multi-term transform, abbreviated as MTT, which, under certain conditions, may provide the better associated accuracy than that of the GBT1, GKLT, GKLT2 and GBT2. This is achieved by the extension of the known transform structures to the form that includes additional terms (see Section II). As a result, the MTT accuracy is improved by the variation of the MTT components. See Sections III-A and III-B, and Examples 1-3 below.

[1]Here, $\Omega = \{\omega\}$ is the set of outcomes, Σ a σ-field of measurable subsets of Ω, $\mu : \Sigma \to [0, 1]$ an associated probability measure on Σ with $\mu(\Omega) = 1$ and (Ω, Σ, μ) for a probability space.

A. Preliminaries

Each matrix $T \in \mathbb{R}^{m \times n}$ defines a bounded linear transformation $\mathcal{T} : L^2(\Omega, \mathbb{R}^n) \to L^2(\Omega, \mathbb{R}^m)$. We write T rather then $\mathcal{T}$ since $[\mathcal{T}(\mathbf{x})](\omega) = T[\mathbf{x}(\omega)]$, for each $\omega \in \Omega$.

The covariance matrix formed from $\mathbf{x}$ and $\mathbf{y}$ is denoted by E_{xy}. The pseudo-inverse of matrix M is denoted by $M^\dagger$.

Let the SVD of matrix $A \in \mathbb{R}^{m \times n}$ be given by $A = U_A \Sigma_A V_A^T$, where $U_A = [u_1\ u_2\ \ldots u_m] \in \mathbb{R}^{m \times m}$, $V_A = [v_1\ v_2\ \ldots v_n] \in \mathbb{R}^{n \times n}$ are unitary matrices, $\Sigma_A = \operatorname{diag}(\sigma_1(A), \ldots, \sigma_{\min(m,n)}(A)) \in \mathbb{R}^{m \times n}$ and $\sigma_1(A) \geq \sigma_2(A) \geq \ldots \geq 0$. For $k < m$, $j < n$ and $\ell < \min(m, n)$, we denote $U_{A,k} := [u_1\ u_2\ \ldots u_k]$, $V_{A,j} = [v_1\ v_2\ \ldots v_j]$ and $\Sigma_{A,\ell} = \operatorname{diag}(\sigma_1(A), \ldots, \sigma_\ell(A))$. Then $[A]_k = \sum_{i=1}^{k} \sigma_i(A) u_i v_i^T \in \mathbb{R}^{m \times n}$, for $k = 1, \ldots, \operatorname{rank}(A)$, is the truncated SVD of A.

1) Review of the Brillinger Transform: Let $M = E_{xy} E_{yy}^{-1} E_{yx}$. The Brillinger transform (BT) of $\mathbf{y}$ is represented by $B = B_1 B_2$ where $B_1 = U_{M,k}$, and $B_2 = U_{M,k}^T E_{xy} E_{yy}^{-1}$

2) Review of Generalized Brillinger-like Transforms: In [2], the generalized Brillinger transforms (GBT1 and GBT2) were developed. The GBT1 is given by $G_1(\mathbf{y}) = B_1 B_2 \mathbf{y}$, where $B_1 = U_{T_y,k}$, $B_2 = U_{T_y,k}^T E_{xy} E_{yy}^\dagger$ and $T_y = E_{xy} E_{yy}^\dagger E_{yx}$. The GBT1 extends the BT to the cases when E_{yy} is singular, i.e., the GBT1 always exists.

The GBT2 generalizes the GBT1 to the form $G_2(\mathbf{y}) = B_1[B_2\mathbf{y} + B_3\mathbf{v}]$ where $B_1 = U_{T_u,k}$, $[B_2, B_3] = U_{T_u,k}^T E_{xu} E_{uu}^\dagger$ and $\mathbf{u} = [\mathbf{y}^T \mathbf{v}^T]^T \in L^2(\Omega, \mathbb{R}^{2n})$. Here, $\mathbf{v} \in L^2(\Omega, \mathbb{R}^n)$ is an 'auxiliary' signal used to further optimize the transform. If $B_1 B_3 \mathbf{v} = \mathbb{O}$ then the GBT2 coincides with the GBT1.

II. The Proposed Approach

A. Generic Transform Structure

For the given compression ratio c, we consider transform $\mathcal{F} : \mathbb{R}^{n \times s} \times \mathbb{R}^{q \times s} \to \mathbb{R}^{m \times s}$ represented by

$$\mathcal{F}(Y, V) = D_1 C_1 Y + D_2 C_2 Q(Y, V), \tag{1}$$

Here, $D_1, \in \mathbb{R}^{m \times k_1}$, $D_2, \in \mathbb{R}^{m \times k_2}$, $C_1 \in \mathbb{R}^{k_1 \times n}$, $C_2 \in \mathbb{R}^{k_2 \times q}$, $k_1 + k_2 = k$, $V \in \mathbb{R}^{q \times s}$ is a sample matrix of an 'auxiliary'

random signal $\mathbf{v}$ called 'injection' and Q is a transformation of Y and V in matrix $Z \in \mathbb{R}^{q\times s}$, i.e., $Z = Q(Y,V)$.

B. Statement of Problems

Denote $W = [Y^T Z^T]^T$ and write $\|\cdot\|$ for the Frobenius norm. Find D_1, C_1, D_2, C_2, V that solve

$$\min_{V}\min_{D_1,C_1}\min_{D_2,C_2}\|X - [D_1C_1Y + D_2C_2Q(Y,V)]\|^2, \quad (2)$$

and Q which implies

$$WW^T = \begin{bmatrix} YY^T & \mathbb{O} \\ \mathbb{O} & ZZ^T \end{bmatrix}. \quad (3)$$

The condition (3) leads to the reduction of the numerical load associated with the solution of problem in (2). This is because the dimension of WW^T is $(n+q)\times(n+q)$ while the dimensions of YY^T and ZZ^T are less, they are $n\times n$ and $q\times q$, respectively. As a result, a solution of problem (2) in terms of matrices YY^T and ZZ^T requires less the numerical load than the solution in terms of matrix WW^T.

The abbreviation MTT will be referred to the multi-term transform based on the solution of the problem in (2)-(3).

Remark 1: We note that although the source signal $\mathbf{x}$ is not available, the sample matrix X can be represented in terms of matrices Y and sample noise matrix N. It is possible, for example, if the observed signal $\mathbf{y}$ is represented in terms of $\mathbf{x}$ and an additive noise (as in Examples 2 and 3 that follow).

C. Specific features of Problem in (2)-(3)

Interestingly, the problem in (2)-(3) can also be treated as a generalization and modification of the blind identification problem considered, in particular, in [6]. By the approach in [6], the problem in (2) is interpreted differently than that in Section II-B above, i.e., X is considered as an available output of the system, Y and V as two inputs, and D_1, C_1, D_2, C_2 make an unknown system function. Further differences from [6] are that in (2), Y is given, V is unknown and the system function contains more unknowns than those in [6]. Moreover, matrices D_1, C_1, D_2, C_2 have special sizes. If we denote $F_1 = D_1C_1$ and $F_2 = D_2C_2$ then F_1 and F_2 are of ranks k_1 and k_2, respectively. Therefore, the problem in (2) can be reformulated as

$$\min_{V}\min_{F_1}\min_{F_2}\|X - [F_1Y + F_2Z]\|^2, \quad (4)$$

subject to $\operatorname{rank} F_1 \le k_1$, $\operatorname{rank} F_2 \le k_2$ and condition (3). By the above reasons, the method in [6] cannot be applied here. Therefore, in Section III below, we propose a new method to solve the problem in (2)–(3).

Since Y is available and V is unknown, we call (2)–(3) the *semi-blind data compression* problem. In terms of [6], it can also be called the *semi-blind system identification* problem.

III. MTT: Solution of Problem in (2), (3)

Here, we represent an iterative method for finding optimal matrices $D_1 \in \mathbb{R}^{m\times k_1}$, $C_1 \in \mathbb{R}^{k_1\times n}$, $D_2 \in \mathbb{R}^{m\times k_2}$, $C_2 \in \mathbb{R}^{k_2\times q}$, V and transformation Q that solve problem (2)-(3).

A. MTT: Optimal Semi-Blind Data Compression

First, in (2), we determine $Q(Y,V)$ such that

$$Z = VG \quad \text{where} \quad G = I - Y^T(YY^T)^\dagger Y = I - Y^\dagger Y. \quad (5)$$

Then condition (3) is true.

As a result, the cost function in (2) can be written as

$$\|X - [D_1C_1Y + D_2C_2Z]\|^2 = \|X - D_1C_1Y\|^2 + \|X - D_2C_2Z\|^2 - \|X\|^2. \quad (6)$$

On the basis of representation (6), the proposed iterative method consists of the following steps. Let us denote $S_Y = XY^T(YY^T)^{1/2^\dagger}$ and $T_Y = S_YS_Y^T = XY^TY^\dagger X^T$.

Step 1: For arbitrary $V = V^{(0)}$ and $Z^{(0)} = V^{(0)}G$, solve

$$\min_{D_1,C_1}\min_{D_2,C_2}\|X - [D_1C_1Y + D_2C_2Z^{(0)}]\|^2. \quad (7)$$

Theorem 1: The solution of problem (7) is given by

$$D_1 = D_1^{(0)} = U_{T_Y,k_1}, \quad C_1 = C_1^{(0)} = U_{T_Y,k_1}^T X(Y^T)^\dagger, \quad (8)$$

$$D_2 = D_2^{(0)} = U_{T_{Z^{(0)}},k_2}, \; C_2 = C_2^{(0)} = U_{T_{Z^{(0)}},k_2}^T X(Z^{(0)^T})^\dagger. \quad (9)$$

Proof: Proof is given in Section IV. ■

Denote

$$\varepsilon^{(0)} = \|X - [D_1^{(0)}C_1^{(0)}Y + D_2^{(0)}C_2^{(0)}Z^{(0)}]\|^2.$$

Step 2: Solve

$$\min_{V}\|X - [D_1^{(0)}C_1^{(0)}Y + D_2^{(0)}C_2^{(0)}Z]\|^2 \quad (10)$$

or equivalently, solve

$$\min_{V}\|X - D_2^{(0)}C_2^{(0)}VG\|^2. \quad (11)$$

This is because other terms in the RHS of (6) do not depend on V. By [7], the minimal norm solution of (11) is

$$V = V^{(1)} = (D_2^{(0)}C_2^{(0)})^\dagger XG^\dagger. \quad (12)$$

Denote $Z^{(1)} = V^{(1)}G$ and

$$\varepsilon_V^{(1)} = \|X - [D_1^{(0)}C_1^{(0)}Y + D_2^{(0)}C_2^{(0)}Z^{(1)}]\|^2. \quad (13)$$

Step 3: Solve

$$\min_{D_2,C_2}\|X - D_2C_2Z^{(0)}\|^2. \quad (14)$$

Similar to (8), the solution is given by

$$D_2 = D_2^{(1)} = U_{T_Z^{(0)},k_2}, \; C_2 = C_2^{(1)} = U_{T_Z^{(0)},k_2}^T X([Z^{(0)}]^T)^\dagger. \quad (15)$$

For the first loop, (15) is the same as (9) but (15) is updated for the next loops (see Step 5 below). Denote

$$\varepsilon_{D,C}^{(1)} = \|X - [D_1^{(0)}C_1^{(0)}Y + D_2^{(1)}C_2^{(1)}Z^{(0)}]\|^2.$$

Step 4: If $\varepsilon_V^{(1)} = \min\{\varepsilon_{D,C}^{(1)}, \varepsilon_V^{(1)}\}$ then set $D_j^{(1)} := D_j^{(0)}$, $C_j^{(1)} := C_j^{(0)}$, for $j = 1,2$, choose $V^{(1)}$ as in (12) and denote $\varepsilon^{(1)} = \varepsilon_{D,C}^{(1)}$.

If $\varepsilon_{D,C}^{(1)} = \min\{\varepsilon_{D,C}^{(1)}, \varepsilon_V^{(1)}\}$ then choose $D_j^{(1)}$ and $C_j^{(1)}$ as

above, i.e., $D_j^{(1)} := D_j^{(0)}$, $C_j^{(1)} := C_j^{(0)}$, for $j = 1, 2$, and set $V^{(1)} := V^{(0)}$. In this case, denote $\varepsilon^{(1)} = \varepsilon_V^{(1)}$.

As a result, the reconstruction of the samples is given by

$$X^{(1)} = D_1^{(1)} C_1^{(1)} Y + D_2^{(1)} C_2^{(1)} Z^{(1)}. \quad (16)$$

Step 5: Repeat steps 2-4 where arbitrary matrix $V^{(0)}$ is replaced with $V^{(1)}$. Then in (14), $Z^{(0)}$ is replaced with $Z^{(1)}$. The new reconstruction of X is denoted by $X^{(2)}$ which is written similar to $X^{(1)}$ in (16).

In general, for $i = 1, 2, \ldots$, steps 2-4 are iterated with updated $V^{(i)}$, $D_2^{(i)}$, $C_2^{(i)}$ and the same $D_1^{(0)}$ and $C_1^{(0)}$. This is because $D_1^{(0)}$ and $C_1^{(0)}$ do not depend on V. The procedure is continued until a tolerance δ is achieved so that $|\varepsilon^{(i+1)} - \varepsilon^{(i)}| \le \delta$. As a result, compression of X is carried out by $C_1^{(1)}$ and $C_2^{(i+1)}$, and its reconstruction is carried out by $D_1^{(1)}$ and $D_2^{(i+1)}$. The associated algorithm is represented below where we denote

$$f(D_1, C_1, D_2, C_2, V) = \|X - [D_1 C_1 Y + D_2 C_2 Z]\|^2 \quad (17)$$

and $f(D_2, C_2, V) = \|X - D_2 C_2 V G\|^2$.

Algorithm MTT: Optimal semi-blind data compression by updated optimization of $V^{(i)}$, $D_2^{(i)}$ and $C_2^{(i)}$.

Input: $V^{(0)}$ and $\delta \in \mathbb{R}^+$.

Output: $D_1^{(0)}$, $C_1^{(0)}$, $D_2^{(i+1)}$, $C_2^{(i+1)}$ and $V^{(i+1)}$.

1. Solve $\min_{D_1, C_1} \min_{D_2, C_2} f(D_1, C_1, D_2, C_2, V)$;
   $D_1^{(0)} := D_1$, $C_1^{(0)} := C_1$; $D_2^{(0)} := D_2$, $C_2^{(0)} := C_2$,
   $V^{(0)} := V$;
2. $\varepsilon^{(0)} = f(D_1^{(0)}, C_1^{(0)}, D_2^{(0)}, C_2^{(0)}, V^{(0)})$;
3. for $i = 0, 1, 2, \ldots$
4. Solve $\min_V f(D_2^{(i)}, C_2^{(i)}, V)$;
5. $\varepsilon_V^{(i+1)} = f(D_1^{(0)}, C_1^{(0)}, D_2, C_2, V)$;
6. Solve $\min_{D_2, C_2} f(D_2, C_2, V^{(i)})$;
7. $\varepsilon_{D,C}^{(i+1)} = f(D_1^{(0)}, C_1^{(0)}, D_2, C_2, V^{(i)})$;
8. if $\varepsilon_{D,C}^{(i+1)} \le \varepsilon_V^{(i+1)}$
9. $D_2^{(i+1)} = D_2$, $C_2^{(i+1)} = C_2$, $V^{(i+1)} = V^{(i)}$;
10. $\varepsilon^{(i+1)} = \varepsilon_{D,C}^{(i+1)}$;
11. else
12. $D_2^{(i+1)} = D_2$, $C_2^{(i+1)} = C_2$, $V^{(i+1)} = V^{(i)}$;
13. $\varepsilon^{(i+1)} = \varepsilon_V^{(i+1)}$;
14. end
15. if $|\varepsilon^{(i+1)} - \varepsilon^{(i)}| \le \delta$
16. Stop
17. end
18. end

Algorithm MTT is based on the following result.

Theorem 2: For $i = 0, 1, \ldots$, let $D_1^{(0)}$, $C_1^{(0)}$, $D_2^{(i)}$, $C_2^{(i)}$, $V^{(i)}$ be determined by Algorithm MTT and let $\varepsilon^{(i)} = f(D_1^{(0)}, C_1^{(0)}, D_2^{(i)}, C_2^{(i)}, V^{(i)})$ be the associated error. Then the increase in the number of iterations i implies the decrease in the associated error, i.e., $\varepsilon^{(i+1)} \le \varepsilon^{(i)}$.

Proof: Proof is given in Section IV. ■

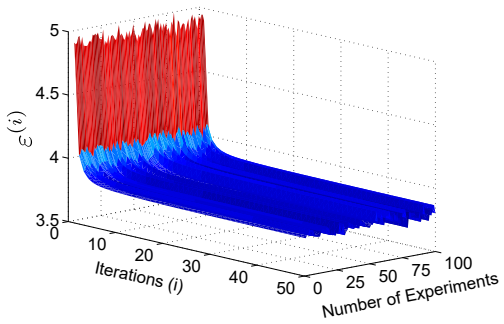


Fig. 1. Example 1. Diagrams of errors associated with Algorithm MTT.

Convergence of Algorithm MTT is considered in Section IV as well.

Example 1: Let $\mathbf{y} = \mathbf{s} \circ \mathbf{x} + 10\boldsymbol{\xi}$ where $\mathbf{x} \in L^2(\Omega, \mathbb{R}^{100})$ and $\mathbf{s} \in L^2(\Omega, \mathbb{R}^{100})$ are uniformly distributed random signals, and $\xi \in L^2(\Omega, \mathbb{R}^{100})$ is a Gaussian noise with mean 0 and variance 1. Symbol '$\circ$' means the Hadamard product. Let $X, S, V \in \mathbb{R}^{100 \times 300}$ be samples of signals $\mathbf{x}$, $\mathbf{s}$ and $\mathbf{v}$, respectively. Then signal $\mathbf{y}$ is represented by matrix $Y = S \circ X + 10N$ where $N \in \mathbb{R}^{100 \times 300}$ is a matrix whose entries are normally distributed with mean 0 and variance 1.

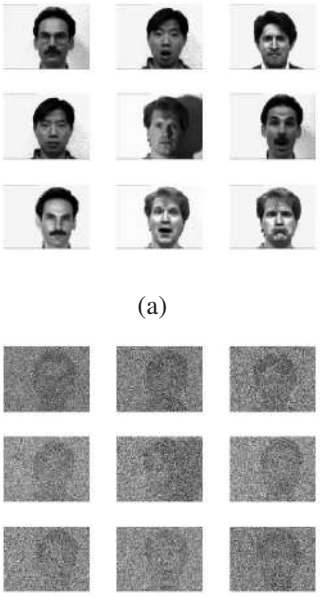
For $k_1 = k_2 = 25$ (and $k = k_1 + k_2 = 50$) and a randomly chosen $V = V^{(0)}$, the error associated with Step 1 of Algorithm MTT is $\varepsilon^{(0)} = 4.8714$ which is the sane as the error associated with the GBT2. Let a given tolerance be $\delta = 10^{-5}$. After 19 iterations of Algorithm MTT, the tolerance is achieved and the associated error is $\varepsilon^{(19)} = 3.7998$. For $k = 50$, the error associated with the GBT1 is $\varepsilon(D_1, C_1) = 7.5727$.

Thus, the error associated with the MTT is less than those for the GBT1 and GBT2 by 50% and 22%, respectively.

In Fig. 1, diagrams of the error associated with the Algorithm MTT are represented, for 100 experiments. In each experiment, matrix $V = V^{(0)}$ was chosen randomly. Fig. 1 illustrate Theorem 2. That is, the increase in the number of iterations i for updated optimization of $V^{(i)}$, $D_2^{(i)}$, $C_2^{(i)}$ implies the decrease in the associated error.

Example 2: Here, we illustrate an application of Algorithm MTT to the problem of compression, filtering and decompression of a set of noisy face images $\mathcal{X}_r = \{\widetilde{X}^{(1)}, \ldots, \widetilde{X}^{(r)}\}$ on the basis of a sample of $s \le r$ images $\mathcal{X}_{t_s} = \{\widetilde{X}^{(t_1)}, \ldots, \widetilde{X}^{(t_s)}\} \subset \mathcal{X}_r$. The sample $\widetilde{X}^{(t_1)}, \ldots, \widetilde{X}^{(t_s)}$ is a permutation of s images from the r images. We choose $r = 110$ and $s = 55$. Each image $\widetilde{X}^{(i)}$, for $i = 1, \ldots, r$, is simulated by MATLAB as $\widetilde{X}^{(i)} = X^{(i)} + \xi_X^{(i)}$ where $X^{(i)} \in \mathbb{R}^{81 \times 107}$ is a numerical representation of the reference face image taken from the Yale Face Database [8], and $\xi_X^{(i)} = \texttt{randn}(81, 107) \in \mathbb{R}^{81 \times 107}$ simulates noise. Some randomly selected face images from the Yale Face Database are given in Fig. 2 (a). Noisy versions of images in Fig. 2 (a) are given in Fig. 2 (b).

Further, let $Y^{(\ell)} \in \mathcal{X}_r$, for $\ell = 1, \ldots, r$. It is assumed that $Y^{(\ell)}$ does not necessary belong to the sample $\mathcal{X}_{t_s}$ but is 'close' to one of the images in the sample, say $\widetilde{X}^{(\alpha)}$, i.e., $\widetilde{X}^{(\alpha)}$, for



(a) (b)

Fig. 2. Example 2: (a): Some randomly selected images from the Yale Face Database. (b): Noisy versions of images in (a)

$\alpha = t_1, \ldots, t_s$, is such that, for all element in $\mathcal{X}_{t_s}$,

$$\widetilde{X}^{(\alpha)} \in \arg \min_{\nu=t_1,\ldots,t_s} \|\widetilde{X}^{(\nu)} - Y^{(\ell)}\|^2 \le \delta,$$

for a given $\delta \ge 0$. We wish to formulate the problem under consideration in terms of the problem in (2). To this end, we denote

$$X = [X^{(t_1)}, ..., X^{(t_s)}] \quad \text{and} \quad Y = [\widetilde{X}^{(t_1)}, ..., \widetilde{X}^{(t_s)}]$$

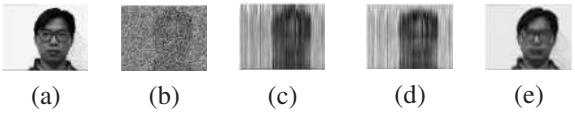
where $X, Y \in \mathbb{R}^{81\times(107\times s)}$. Matrix V (see (2)) is simulated as $V = [V^{(1)}, ..., V^{(g)}] \in \mathbb{R}^{81\times q}$ where $q = 107 \times g$ and entries of each $V^{(i)}$, for $i = 1, \ldots, g$, are uniformly distributed. In this notation, the problem under consideration is formulated as in (2), i.e., we wish to find D_1, C_1, D_2, C_2 and V that solve

$$\min_{V} \min_{D_1,C_1} \min_{D_2,C_2} \|X - [D_1C_1Y + D_2C_2Z]\|^2. \tag{18}$$

Matrix Z is represented it in the block form as $Z = [Z^{(t_1)}, ..., Z^{(t_s)}]$ where $Z^{(j)} \in \mathbb{R}^{k_2\times 107}$, for $j = 1, \ldots, s$.

Matrices D_1, C_1, D_2, C_2 and V are determined by Algorithm MTT. Then an estimate of $X^{(\alpha)}$, for $\alpha = 1, \ldots, r$, is given by $\widehat{X}^{(\alpha)} = D_1C_1Y^{(\ell)} + D_2C_2Z^{(\ell)}$. The estimate is represented, for $k_1 = 20$, $k_2 = 20$ and $k = k_1 + k_2 = 40$, in Fig. 3 (e), for V with arbitrarily uniformly distributed entries, and in Fig. 3 (f) with optimal V using 10 iterations.

The GBT1 and GBT2 [2] have also been applied to the problem in consideration, for the same $k = 40$. The mean square errors (MSEs) are represented in Fig. 3 and Table 1 where the MSE means the mean square error. The obtained numerical results clearly demonstrate the advantages of the Algorithm MTT. The MSE and SSIM associated with the MTT



(a) (b) (c) (d) (e)

Fig. 3. Example 2: Reference image $X^{(\alpha)}$ (a); Observed image $\widetilde{Y}^{(\ell)}$ (b); Estimates of $X^{(\alpha)}$ by the GBT1 (c); GBT2 (d); MTT (e).

are much better than those for the other methods.

Table 1

Transforms	MSEs
GBT1	3.50×10^2
GBT2	2.80×10^2
MTT	1.80×10^1

B. Particular Feature of MTT

On the basis of results presented in [9], it can be shown that the MTT converges faster if the dimension q of matrix V increases. Here, we illustrate this feature as follows.

Example 3: Let $\mathbf{y} = \mathbf{x} + \boldsymbol{\xi}$ where $\mathbf{x} \in L^2(\Omega, \mathbb{R}^{50})$ and $\boldsymbol{\xi} \in L^2(\Omega, \mathbb{R}^{50})$ are uniformly distributed source signal and white noise, respectively. Let $\mathbf{v} \in L^2(\Omega, \mathbb{R}^q)$ be a Gaussian random signal. Noise $\boldsymbol{\xi}$ is uncorrelated with $\mathbf{x}$. Vectors $\mathbf{y}$, $\boldsymbol{\xi}$ and $\mathbf{v}$ are represented by samples $Y \in \mathbb{R}^{50\times 100}$, $\Xi \in \mathbb{R}^{50\times 100}$ and $V \in \mathbb{R}^{q\times 100}$. Then $E_{yy} = \frac{1}{100}YY^T$, $E_{xx} = E_{yy} - E_{\xi\xi}$ and $E_{xy} = E_{xx}$ where $E_{\xi\xi} = \sigma^2 I$ and $\sigma = 2$.

In Fig. 4, for $k_1 = k_2 = 12$, diagrams of the error associated with Step 1 of the MTT versus dimension q of matrix V are given, for 100 experiments. The error decreases when q increases. In each experiment, matrix Y was chosen randomly.

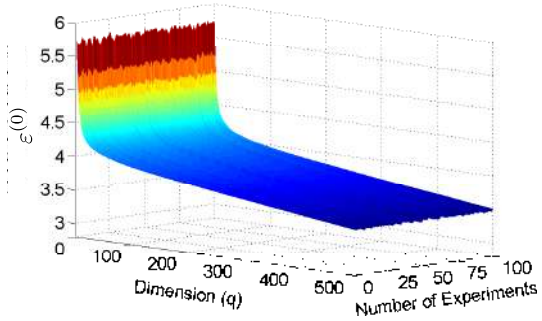


Fig. 4. Example 3: Diagrams of the error associated with Step 1 of the MTT versus dimension q of matrix V.

IV. Appendix

1) Proof of Theorem 1: Denote $W := W^{(0)} = [Y^T {Z^{(0)}}^T]^T$.

Then
$$\begin{aligned} &\|X - [D_1C_1Y + D_2C_2Z^{(0)}]\|^2 \\ &= \|(XX^T)^{1/2}\|^2 - \|XW^T(WW^T)^{1/2^\dagger}\|^2 \\ &\quad + \|XW^T(WW^T)^{1/2} - [F_1F_2](WW^T)^{1/2}\|^2. \end{aligned} \tag{19}$$

Further, $(WW^T)^{1/2^\dagger} = \begin{bmatrix} (YY^T)^{1/2^\dagger} & \mathbb{O} \\ \mathbb{O} & (ZZ^T)^{1/2^\dagger} \end{bmatrix}$. Then

$$\begin{aligned} &\|XW^T(WW^T)^{1/2} - [F_1F_2](WW^T)^{1/2}\|^2 \\ &\quad = \|S_Y - F_1(YY^T)^{1/2}\|^2 + \|S_Z - F_2(ZZ^T)^{1/2}\|^2. \end{aligned}$$

Further, let us denote by $\mathbb{R}(m,n,k)$ the set of all $m \times n$ matrices of rank at most k. The minimal norm solutions to the problems $\min_{F_1 \in \mathbb{R}(m,n,k_1)} \|S_Y - F_1(YY^T)^{1/2}\|^2$ and $\min_{F_2 \in \mathbb{R}(m,n,k_2)} \|S_Z - F_2(ZZ^T)^{1/2}\|^2$ are given [7] by

$$F_1 = [S_Y]_{k_1} (YY^T)^{1/2^\dagger} \quad \text{and} \quad F_2 = [S_Z]_{k_2} (ZZ^T)^{1/2^\dagger}, \tag{20}$$

respectively. Then (8), (9) follow from (20) on the basis of Fact 1 in [2].

2) Proof of Theorem 2: To prove Theorem 2 we need some preliminaries as follows.

Definition 1: Let $\mathcal{M}$ be a metric space and let $\mathbb{B}(\alpha,\epsilon) \subset \mathcal{M}$ be an open ball with center α and radius ϵ. Point α is called a *cluster point* [10] of a sequence $\{x_n\} \subset \mathcal{M}$ iff for each $\epsilon > 0$ and $m \in \mathbb{N}$, there is some $n \geq m$ such that $x_n \in \mathbb{B}(\alpha,\epsilon)$, for all $n \geq m$.

We denote $F_j = D_j C_j$, $F_j^{(i)} = D_j^{(i)} C_j^{(i)}$, $F = \{F_1, F_2\}$ and $F^{(i)} = \{F_1^{(i)}, F_2^{(i)}\}$ where $j = 1,2$ and $i = 0,1,\ldots$. Then

$$\begin{aligned} f(F,V) &= f(F_1,F_2,V) \\ &= f(D_1,C_1,D_2,C_2,V) = \|X - F_1 Y - F_2 V G\|^2 \end{aligned}$$

where $f(D_1,C_1,D_2,C_2,V)$ is defined by (17).

Let us now define compact sets K_1 and K_2 such that

$$K_1 = \{F : 0 \leq f(F,V^{(0)}) \leq f(F^{(0)},V^{(0)})\} \tag{21}$$

$$K_2 = \{V : 0 \leq f(F^{(0)},V) \leq f(F^{(0)},V^{(0)})\}. \tag{22}$$

Definition 2: Let $S^* = (F^*,V^*)$ be a cluster point of sequence $S^{(j)} = (F^{(j)},V^{(j)})$, for $j = 1,2,\ldots$. Point $\overline{F}_{V^*}$ is called a *best response* to F if $\overline{F}_{V^*} \in \arg\min_{F \in K_1} f(F,V^*)$.

The proof of Theorem 2 is as follows.

Proof: For given $F_1^{(0)} = D_1^{(0)} C_1^{(0)}$, $F_2^{(i)} = D_2^{(i)} C_2^{(i)}$ and $V^{(i)}$, we determine $\varepsilon^{(i)} = f(F_1^{(0)},F_2^{(i)},V^{(i)})$. Then $\varepsilon^{(i+1)}$ is determined with the following steps. By Algorithm MTT, the best responses are computed as follows:

Given $F_1^{(0)}$ and $F_2^{(i)}$, compute $\widetilde{V}$ which is the best response to V for $f(F_1^{(0)},F_2^{(i)},V)$. Given $F_1^{(0)}$ and $V^{(i)}$, compute $\widetilde{F}_2$ of rank $\leq k_2$ which is the best response to F_2 for $f(F_1^{(0)},F_2,V^{(i)})$.

Let us now define $\varepsilon_{F_2}^{(i)} = f(F_1^{(0)},\widetilde{F}_2,V^{(i)})$ and $\varepsilon_V^{(i)} = f(F_1^{(0)},F_2^{(i)},\widetilde{V})$. If $\varepsilon_{F_2}^{(i)} = \min\{\varepsilon_{F_2}^{(i)},\varepsilon_V^{(i)}\}$, then

$$\varepsilon^{(i+1)} = \varepsilon_{F_2}^{(i)} = f(F_1^{(0)},\widetilde{F}_2,V^{(i)}).$$

At the same time, for given $F_1^{(0)}$ and $V^{(i)}$, the best response to F_2, for $f(F_1^{(0)},F_2,V^{(i)})$, is $\widetilde{F}_2$. Then, for all F_2,

$$f(F_1^{(0)},\widetilde{F}_2,V^{(i)}) \leq f(F_1^{(0)},F_2,V^{(i)}).$$

In particular, $f(F_1^{(0)},\widetilde{F}_2,V^{(i)}) \leq f(F_1^{(0)},F_2^{(i)},V^{(i)})$. Then $\varepsilon^{(i+1)} \leq \varepsilon^{(i)}$. If $\varepsilon_V^{(i)} = \min\{\varepsilon_{F_2}^{(i)},\varepsilon_V^{(i)}\}$, then

$$\varepsilon^{(i+1)} = \varepsilon_V^{(i)} = f(F_1^{(0)},F_2^{(i)},\widetilde{V}).$$

Further, given $F_1^{(0)}$ and $F_2^{(i)}$, the best response to V, for $f(F_1^{(0)},F_2^{(i)},V)$, is $\widetilde{V}$. Then, for all V, we obtain

$$f(F_1^{(0)},F_2^{(i)},\widetilde{V}) \leq f(F_1^{(0)},F_2^{(i)},V).$$

In particular,

$$f(F_1^{(0)}F_2^{(i)},\widetilde{V}) \leq f(F_1^{(0)},F_2^{(i)},V^{(i)}).$$

i.e., $\varepsilon^{(i+1)} \leq \varepsilon^{(i)}$. Then the statement of Theorem 2 follows. ■

3) Convergence of Algorithm MTT:

Theorem 3: Let K_1 and K_2 be compact sets defined by (21) and (22), respectively, and $F \in K_1$ and $V \in K_2$. Let $F^{(j)}$ and $V^{(j)}$ be determined by Algorithm MTT, for $j = 0,1,\ldots$. Then any cluster point of sequence $S^{(j)} = (F^{(j)},V^{(j)})$, say $S^* = (F^*,V^*)$, is a coordinate-wise minimum point of $f(F,V)$, i.e.,

$$F^* \in \arg\min_{F \in K_1} f(F,V^*), \quad V^* \in \arg\min_{V \in K_2} f(F^*,V).$$

Proof: Since each K_j is compact, then there is a subsequence $S^{(j_t)} = (F^{(j_t)},V^{(j_t)})$ such that $S^{(j_t)} \to S^*$ when $t \to \infty$. Consider entry $F^{(j_t)}$ of $S^{(j_t)}$. Let $\overline{F}_{V^*}$ and $\overline{F}_{V^{(j_t)}}$ be best responses to F associated with $\overline{F}_{V^*}$ and $V^{(j_t)}$, respectively. Then we have

$$\begin{aligned} f(\overline{F}_{V^*},V^{(j_t)}) &\geq f(\overline{F}_{V^{(j_t)}},V^{(j_t)}) \geq f(F^{(j_t+1)},V^{(j_t+1)}) \\ &\geq f(F^{(j_{t+1})},V^{(j_{t+1})}) \end{aligned}$$

By continuity, $f(\overline{F}_{V^*},V^*) \geq f(F^*,V^*)$ as $t \to \infty$. It implies that latter should hold as an equality, since the inequality is true by the definition of the best response $\overline{F}_{V^*}$. Thus, F^* is the best response for V^*, or equivalently, $\overline{F}^*$ is the solution for the problem

$$\arg\min_{F \in K_1} f(F,V^*).$$

The proof is similar if we consider entry $V^{(j_t)}$ of $S^{(j_t)}$. ■

References

[1] D. R. Brillinger, *Time Series: Data Analysis and Theory*. San Francisco: Holden Day, 2001.

[2] A. Torokhti and P. Soto-Quiros, "Generalized Brillinger-Like Transforms," *IEEE Signal Processing Letters*, vol. 23, no. 6, pp. 843 – 847, 2016.

[3] Y. Hua and W. Liu, "Generalized Karhunen-Loeve transform," *IEEE Signal Processing Letters*, vol. 5, no. 6, pp. 141–142, June 1998.

[4] A. Torokhti and P. Howlett, *Computational Methods for Modelling of Nonlinear Systems*. Elsevier, 2007.

[5] ——, "Optimal fixed rank transform of the second degree," *IEEE Trans. CAS. Part II, Analog and Digital Signal Processing*, vol. 48, no. 3, pp. 309 – 315, 2001.

[6] K. Abed-Meriam, Q. Wanzhi, and H. Yingbo, "Blind system identification," *Proceedings of the IEEE*, vol. 85, no. 8, pp. 1310 – 1322, 1997.

[7] S. Friedland and A. Torokhti, "Generalized rank-constrained matrix approximations," *SIAM Journal on Matrix Analysis and Applications*, vol. 29, no. 2, pp. 656–659, 2007.

[8] Computer Vision Laboratory. University of California, San Diego. Yale face database. [Online]. Available: http://vision.ucsd.edu/datasets/yale_face_dataset_original

[9] A. Torokhti and P. Soto-Quiros, "Multi-term transform of random vectors," *Proceedings of IEEE International Symposium on Information Theory*, June 2014 (submited).

[10] H. Amann and J. Escher, *Analysis I*. Birkhäuser Basel, 2006.

Anexo 8

Second Degree Model for Multi-Compression and Recovery of Distributed Signals

Pablo Soto-Quiros, Anatoli Torokhti and Stanley J. Miklavcic

***Abstract*—We study the problem of multi-compression and reconstructing a stochastic signal observed by several independent sensors (or compressors) that transmit compressed information to a fusion center. The key aspect of this problem is to find models of the sensors and fusion center that are optimized in the sense of an error minimization under a certain criterion, such as the mean square error (MSE). A novel technique to solve this problem is developed. The novelty is as follows. First, the multi-compressors are non-linear and modeled using second degree polynomials. This may increase the accuracy of the signal estimation through the optimization in a higher dimensional parameter space compared to the linear case. Second, the required models are determined by a method based on a combination of the second degree transform (SDT) [1] with the maximum block improvement (MBI) method [2], [3] and the generalized rank-constrained matrix approximation [4], [5]. It allows us to use the advantages of the methods in [2], [3], [4], [5] to further increase the estimation accuracy of the source signal. Third, the proposed method is justified in terms of pseudo-inverse matrices. As a result, the models of compressors and fusion center always exist and are numerically stable. In other words, the proposed models may provide compression, de-noising and reconstruction of distributed signals in cases when known methods either are not applicable or may produce larger associated errors.**

***Keywords*—Data compression, stochastic signal recovery.**

I. Introduction

This paper deals with a new method for the multi-compression and recovery of a source stochastic signal for a non-linear wireless sensor network (WSN). Although WSNs exhibit a significant potential in many application fields (see, for example [6]), up till now the results obtained are based on linear models and fall short of a consideration of more effective non-linear models for WSNs. The objective of this paper is to provide a new effective technique for the modeling of a non-linear WSN. The technique is based on an extension of the approach in [1] in combination with ideas of the maximum block improvement (MBI) method [2], [3] and the generalized rank-constrained matrix approximation [4], [5].

Pablo Soto-Quiros is with the Centre for Industrial and Applied Mathematics, University of South Australia, SA 5095, Australia and Instituto Tecnologico de Costa Rica, Apdo. 159-7050, Cartago, Costa Rica (e-mail: juan.soto-quiros@mymail.unisa.edu.au).

Anatoli Torokhti is with the Centre for Industrial and Applied Mathematics, University of South Australia, SA 5095, Australia (e-mail: anatoli.torokhti@unisa.edu.au).

Stanley J. Miklavcic is with the Phenomics and Bioinformatics Research Centre, University of South Australia, SA 5095, Australia (e-mail: stan.miklavcic@unisa.edu.au).

Manuscript received XXXX XX, 2015; revised XXXX XX, 2015.

A. Motivation

We are motivated and inspired by the work in [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17] where the effective methods for modeling of WSNs were developed. The WSNs are widely used in many areas of signal processing such as, for example, battlefield surveillance, target localization and tracking, and acoustic beamforming using an array of microphones [18], [19], [20] . An associated scenario involves a set of spatially distributed sensors, $\mathcal{B}_1, \ldots, \mathcal{B}_p$, and a fusion center $\mathcal{T}$. The sensors are autonomous, have a limited energy supply and furthermore, cannot communicate with each other. The sensors make local noisy observations, $\mathbf{y}_1, \ldots, \mathbf{y}_p$, correlated with a source stochastic signal $\mathbf{x}$. Each sensor $\mathcal{B}_j$ transmits compressed information about its measurements, $\mathbf{u}_j$, to the fusion center which should recover the original signal within a prescribed accuracy. The compression level is predefined and is not data-dependent.

The problem is to find an effective way to compress and denoise each observation $\mathbf{y}_j$, where $j = 1, \ldots, p$, and then reconstruct all the compressed observations in the fusion center so that the reconstruction will be optimal in the sense of a minimization of the associated error under a certain criterion, such as the mean square error (MSE).

The known methods for the distributed signal compression and recovery [7]-[17] are based on a natural idea of extensions of the optimal *linear* transforms [21], [22], [23]. In many cases, this approach leads to a required associated accuracy. At the same time, the error associated with the optimal *linear* transform, for a given compression ratio, cannot be diminished by any other linear transform. Thus, if the performance of the methods based on the optimal *linear* transforms is not as good as required then a method based on a different idea should be applied. In this paper, such a method is provided.

The key questions motivating this work are as follows. How can one improve the accuracy of the distributed signal processing approach compared to that of other methods? Second, is it possible to achieve the improvement under the same assumptions that those in the known methods? The last question is motivated by an observation that the models with an "extended" structure often require additional initial information needed for their implementation. Third, the block coordinate descent (BCD) method [24] is known to work on the modeling of WSNs when applied only under certain conditions, which may restrict its applicability. Is it possible to avoid those restrictions?

B. Known techniques

Here, the term "compression" is treated in the same sense as in the previous works on data compression (developed, for instance, in [7]-[17], [25], [26]), i.e. we say that observed signal $\mathbf{y}_j$ with n_j components is compressed if it is represented by signal $\mathbf{u}_j$ with r_j components where $r_j < n_j$, for $j = 1, \ldots, p$. That is "compression" refers to dimensionality reduction and not quantization which outputs bits for digital transmission. This is similar to what is considered, in particular, in [7], [8], [9], [10], [11].

It is known that in the nondisrtibuted setting (in the other words, in the case of a single sensor only) the MSE optimal solution is provided by the generic Karhunen-Loève transform (GKLT) [1], [27] which provides the smallest associated error in the class of all *linear* transforms. Nevertheless, the GKLT cannot be applied to the above WSN since the entire data vector $\mathbf{y} = [\mathbf{y}_1^T, \ldots, \mathbf{y}_p^T]^T$ is not observed by each sensor (in this regard, see also [10], [11]). Therefore, several approaches to a determination of mathematical models for $\mathcal{B}_1, \ldots, \mathcal{B}_p$ and $\mathcal{T}$ have been pursued. In particular, in the information-theoretic context, distributed compression has been considered in the landmark works of Slepian and Wolf [28], and Wyner and Ziv [29]. The natural setting of the approach in [28], [29] in terms of the transform-based methodology has been considered in [7], [8], [9], [10], [11]. Intimate relations between these two perspectives have been shown in [30]. The methodology developed in [7], [8], [9], [10], [11] is based on the dimensionality reduction by linear projections. Such an approach has received considerable attention (see, for example, [12], [13], [14], [15], [16], [17], [31], [32]).

In particular, in [7], two approaches are considered. By the first approach, the fusion center model, $\mathcal{T}$, is given in the form $\mathcal{T} = [\mathcal{T}_1, \ldots, \mathcal{T}_p]$ where $\mathcal{T}_j$ is a 'block' of $\mathcal{T}$, for $j = 1, \ldots, p$, and then the original MSE cost function is represented as a sum of p decoupled MSE cost functions. Then approximations to $\mathcal{B}_j$ and $\mathcal{T}_j$ are found as solution to each of the p associated MSE minimization (MSEM) problems. Some more related results can be found in [33]. The second approach in [7] generalizes the results in [8], [9] in the following way. The original MSE cost function is represented, by re-grouping terms, in the form similar to that presented by the summand in the decoupled MSE cost function. Then the minimum is seeking for each $\mathcal{T}_j\mathcal{B}_j$, for $j = 1, \ldots, p$, while other terms $\mathcal{T}_k\mathcal{B}_k$, for $k = 1, \ldots, j-1, j+1, \ldots, p$, are assumed to be fixed. The minimizer follows from the result given in [21] (Theorem 10.2.4). To combine solutions of those p local MSEM problems, approximations to each sensor model $\mathcal{B}_j$ are determined from an iterative procedure. Values of $\mathcal{B}_1, \ldots, \mathcal{B}_p$ for the initial iteration are chosen randomly.

The method in [11] is based on ideas similar to those in the earlier references [7], [8], [9], [10][1], i.e., on the replacement of the original MSEM problem with the $p+1$ unconstrained MSEM problems for separate determination of approximations to $\mathcal{B}_j$, for each $j = 1, \ldots, p$, and then an approximation to $\mathcal{T}$. First, an approximation to each $\mathcal{B}_j$, for $j = 1, \ldots, p$, is determined under assumption that other $p-1$ sensors are fixed. Then, on the basis of known approximations to $\mathcal{B}_1, \ldots, \mathcal{B}_p$, an approximation of $\mathcal{T}$ is determined as the optimal Wiener filter. In [11], the involved signals are assumed to be zero-mean jointly Gaussian random vectors. Here, this restriction is not used.

The work in [8], [9], [10] can be considered as a particular case of [11].

The method in [34] is applied to the problem which is an approximation of the original problem. It implies an increase in the associated error compared to the method applied to the original problem. Further, the technique suggested in [34] is applicable under certain restrictions imposed on observations and associated covariance matrices. In particular, in [34], the observations should be presented in the special form $\mathbf{y}_j = H_j\mathbf{x} + \mathbf{v}_j$, for $j = 1, \ldots, p$ (where H_j is a measurement matrix and $\mathbf{v}_j$ is noise), and the covariance matrix formed by the noise vector should be *block-diagonal* and invertible. It is not the case here.

The approaches taken in the above references are based, in fact, on the BCD method [24]. The BCD method converges to a stationary point if the space of optimization is convex, and the objective function is continuous and regular [24] (pp. 480–481). It converges to a coordinatewise minimum if the space of optimization is convex and the objective function is continuous [24] (pp. 480–481). The BCD method converges to a global minimum if the objective function can be separated into sums of functions of single block variables [3] (p. 211). The above conditions are not satisfied in the present case.

Further, the WSN models in [7], [8], [9], [10], [11], [34] are justified in terms of inverse matrices. It is known that in cases when the matrices are close to singular this may lead to instability and a significant increase in the associated error. Moreover, when the matrices are singular, the algorithms [7], [8], [9], [10], [11] may not be applicable. This observation is illustrated by Examples 2, 3 in Section IV and Example 4 in Section VI below where the associated matrices are singular and the method [7] is not applicable. In [11], for the case when a matrix is singular, the inverse is replaced with the pseudo-inverse, but such a simple replacement does not follow from the justification of the model provided in [11]. As a result, a simple substitution by pseudo-inverse matrices may result in numerical instability as shown, in particular, in [35] and Example 2 in Section IV. In this regard, we also refer to references [4], [27], [36] where the case of rank-constrained data compression in terms of the pseudo-inverse matrices is studied.

C. Differences from known methods. Novelty and Contribution

The methods of [7]-[17], [34] are based on exploiting the *linear* minimizers of the mean square error [21], [22], [23] in the BCD method [24]. The key advantages and novelty of the proposed methodology, and differences from the techniques in [7]-[17], [34] are as follows.

Firstly, the minimizers developed in this paper are *non-linear*. They are based on the second degree transform (SDT) considered in [1]. It has been shown in [1] that under a certain

[1]In particular, it generalizes the work of [10] to the case when the vectors of interest are not directly observed by the sensors.

condition, the SDT leads to greater accuracy in the signal estimation compared to methods based on the optimal linear signal transform, the GKLT. In Section X-C3, a similar condition is determined for the case under consideration, i.e., for distributed signal compression and reconstruction. Secondly, unlike the previous works we apply a version of the maximum block improvement (MBI) method [2], [3], not the BCD method [24]. This is because a solution of the minimization problem we consider is not unique, the optimization space is not convex and the objective function is not regular in the sense of [24]. The objective function also does not separate into sums of functions of single block variables [3]. As a result, convergence of the BCD method cannot be guaranteed [3] for the problem we consider. The MBI method [2], [3] avoids the requirements of the BCD method. The main idea of the MBI method is to implement an update of the block of variables which is the best possible among all the updates. Further, unlike the previous methods, at each stage of the proposed greedy method the generalized rank-constrained matrix approximation [4], [5] has to be applied. This is required because of the special structure of the objective function under consideration. More details are given below in Section III-A. Thirdly, the minimizers in [21], [22], [23] used in [7]-[17], [34] have been obtained in terms of inverse matrices, i.e., they have only been *justified* for full rank covariance matrices. In our method, the minimizers are obtained and rigorously *justified* in terms of pseudo-inverse matrices and, therefore, the models of the sensors and the fusion center are numerically stable and always exist. In other words, the proposed WSN model may provide compression, de-noising and reconstruction of distributed signals for the cases when known methods either are not applicable (because of singularity of associated matrices) or produce larger associated errors. This observation is supported, in particular, by the error analysis provided in Section X-C and by the results of simulations shown in Section IV. Fourthly, despite the special structure of our method compared to those of known techniques, the same initial information is required for the method implementation. This observation is illustrated in Example 1 given in Section IV.

D. Notation

Here, we provide some notation which is required to formalize the problem in the form presented in Section below. Let us write (Ω, Σ, μ) for a probability space[2]. We denote by $\mathbf{x} \in L^2(\Omega, \mathbb{R}^m)$ the signal of interest[3] (a source signal to be estimated) represented as $\mathbf{x} = [\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}]^T$ where $\mathbf{x}^{(j)} \in L^2(\Omega, \mathbb{R})$, for $j = 1, \ldots, m$. Further, $\mathbf{y}_1 \in L^2(\Omega, \mathbb{R}^{n_1}), \ldots, \mathbf{y}_p \in L^2(\Omega, \mathbb{R}^{n_p})$ are observations made by the sensors. In this regard, we write

$$\mathbf{y} = [\mathbf{y}_1^T, \ldots, \mathbf{y}_p^T]^T \quad \text{and} \quad \mathbf{y} = [\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(n)}]^T \tag{1}$$

[2]Here $\Omega = \{\omega\}$ is the set of outcomes, Σ a $\sigma-$field of measurable subsets of Ω and $\mu : \Sigma \to [0, 1]$ an associated probability measure on Σ with $\mu(\Omega) = 1$.

[3]Space $L^2(\Omega, \mathbb{R}^m)$ has to be used because of the norm introduced in (3) below.

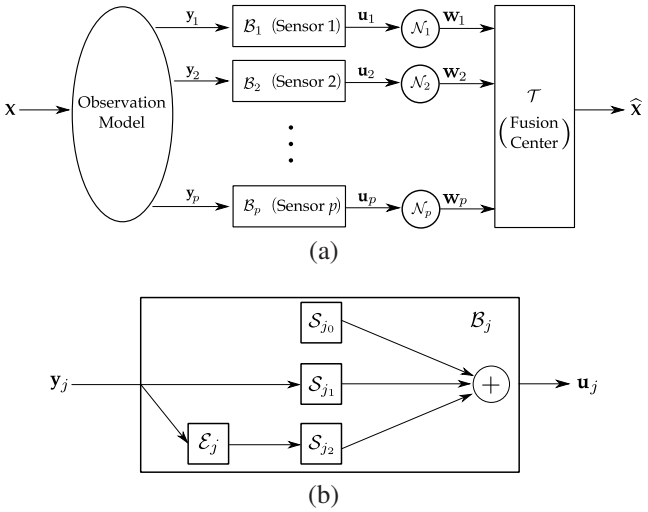


Fig. 1. (a): WSN model. (b): Sensor model.

where $\mathbf{y}^{(k)} \in L^2(\Omega, \mathbb{R})$, for $k = 1, \ldots, n$. We would like to emphasize a difference between $\mathbf{y}_j$ and $\mathbf{y}^{(k)}$: in (1), the observation $\mathbf{y}_j$, for $j = 1, \ldots, p$, is a random vector itself (i.e. $\mathbf{y}_j$ is a 'piece' of $\mathbf{y}$), and $\mathbf{y}^{(k)}$, for $k = 1, \ldots, n$, is a random variable (i.e. $\mathbf{y}^{(k)}$ is an entry of $\mathbf{y}$). Therefore, $\mathbf{y}_j = (\mathbf{y}^{(q_{j-1}+1)}, \ldots, \mathbf{y}^{(q_j)})^T$ where $q_j = q_{j-1} + n_j$, $j = 1, \ldots, p$ and $q_0 = 0$.

For $j = 1, \ldots, p$, we represent a sensor model by linear operator $\mathcal{B}_j : L^2(\Omega, \mathbb{R}^{n_j}) \to L^2(\Omega, \mathbb{R}^{r_j})$ defined by matrix $B_j \in \mathbb{R}^{r_j \times n_j}$ such that,[4] for $j = 1, \ldots, p$,

$$[\mathcal{B}_j(\mathbf{y}_j)](\omega) = B_j[\mathbf{y}_j(\omega)]. \tag{2}$$

Here, $r = r_1 + \ldots + r_p$, , $r_j \leq \min\{m, n_j\}$, and r_j is given and fixed for each jth sensor, for $j = 1, \ldots, p$. Let us denote $\mathbf{u}_j = \mathcal{B}_j(\mathbf{y}_j)$ and $\mathbf{u} = [\mathbf{u}_1^T, \ldots, \mathbf{u}_p^T]^T$, where for $j = 1, \ldots, p$, vector $\mathbf{u}_j \in L^2(\Omega, \mathbb{R}^{r_j})$ represents the compressed observation transmitted by a jth sensor $\mathcal{B}_j$ to the fusion center $\mathcal{T}$.

A fusion center model is represented by linear operator $\mathcal{T} : L^2(\Omega, \mathbb{R}^r) \to L^2(\Omega, \mathbb{R}^m)$ defined by matrix $T \in \mathbb{R}^{m \times r}$ so that $[\mathcal{T}(\mathbf{u})](\omega) = T[\mathbf{u}(\omega)]$, where $\mathbf{u} \in L^2(\Omega, \mathbb{R}^r)$. To state the problem in the next section, we also denote

$$\mathbb{E}(\|\mathbf{x}\|^2) := \|\mathbf{x}\|_\Omega^2 := \int_\Omega \|\mathbf{x}(\omega)\|_2^2 d\mu(\omega) < \infty, \tag{3}$$

where $\|\mathbf{x}(\omega)\|_2$ is the Euclidean norm of $\mathbf{x}(\omega) \in \mathbb{R}^m$.

II. Statements of the Problems

A. Formalization of the Generic Problem

For $\mathbf{x}$ represented by $\mathbf{x} = [\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}]^T$ where $\mathbf{x}^{(j)} \in L^2(\Omega, \mathbb{R})$ we write $E[\mathbf{x}\mathbf{y}^T] = E_{xy} = \left\{E_{x_j y_k}\right\}_{j,k=1}^{m,n} \in \mathbb{R}^{m \times n}$, where $E_{x_j y_k} = \int_\Omega \mathbf{x}^{(j)}(\omega)\mathbf{y}^{(k)}(\omega) d\mu(\omega) < \infty$ and $\mathbf{y} = [\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(n)}]^T$. An assumption used in previous methods [7], [8], [9], [10], [11] and associated works such as [21], [22],

[4]An explanation for the relation in (2) is given in Section X-B.

[23], [25], [37] is that the covariance matrices E_{xy} and E_{yy} are known. Here, we adopt this assumption. As mentioned, in particular, in [7] (and similarly, for example, in [37], [38], [39], [40], [41], [42], [43], [44], [45]), "*a priori* knowledge of the covariances can come either from specific data models, or, after sample estimation during a training phase." Importantly, although we wish to develop the method with an extended structure, no additional initial information is required. Example 1 in Section IV illustrates this observation.

First, we consider the case when the channels from the sensors to the fusion center are ideal, i.e., each channel operator $\mathcal{N}_j$ in Fig. 1 (a) is the identity. The case of not ideal channels will be considered in Section V. Then the problem is as follows: Find models of the sensors, $\mathcal{B}_1, \ldots, \mathcal{B}_p$, and a model of the fusion center, $\mathcal{T}$, that provide

$$\min_{\mathcal{T},\mathcal{B}_1,\ldots,\mathcal{B}_p} \left\| \mathbf{x} - \mathcal{T} \begin{bmatrix} \mathcal{B}_1(\mathbf{y}_1) \\ \vdots \\ \mathcal{B}_p(\mathbf{y}_p) \end{bmatrix} \right\|_{\Omega}^2. \tag{4}$$

The model of the fusion center, $\mathcal{T}$, can be represented as $\mathcal{T} = [\mathcal{T}_1, \ldots, \mathcal{T}_p]$ where for $j = 1, \ldots, p$, $\mathcal{T}_j : L^2(\Omega, \mathbb{R}^{r_j}) \to L^2(\Omega, \mathbb{R}^m)$. Let us write

$$\begin{aligned} &\min_{\substack{\mathcal{T}_1,\ldots,\mathcal{T}_p,\\ \mathcal{B}_1,\ldots,\mathcal{B}_p}} \left\| \mathbf{x} - [\mathcal{T}_1, \ldots, \mathcal{T}_p] \begin{bmatrix} \mathcal{B}_1(\mathbf{y}_1) \\ \vdots \\ \mathcal{B}_p(\mathbf{y}_p) \end{bmatrix} \right\|_{\Omega}^2 \\ &= \min_{\substack{\mathcal{T}_1,\ldots,\mathcal{T}_p,\\ \mathcal{B}_1,\ldots,\mathcal{B}_p}} \|\mathbf{x} - [\mathcal{T}_1\mathcal{B}_1(\mathbf{y}_1) + \ldots + \mathcal{T}_p\mathcal{B}_p(\mathbf{y}_p)]\|_{\Omega}^2 \end{aligned} \tag{5}$$

where $\mathbf{y} = [\mathbf{y}_1^T, \ldots, \mathbf{y}_p^T]^T$.

In this paper, we study the case where $\mathcal{B}_j$, for $j = 1, \ldots, p$, is represented by a second degree polynomial, i.e., is non-linear. Previous methods were developed for the case of *linear* $\mathcal{T}_j$ and $\mathcal{B}_j$, for $j = 1, \ldots, p$.

B. Inducement to Introduce Second Degree WSN

1) Linear WSN: Let us first consider the case when $\mathcal{T}_j$ and $\mathcal{B}_j$ are *linear* operators, and $\mathcal{N}_j = I$, for $j = 1, \ldots, p$. Denote by $\mathcal{R}(m, n_j, r_j)$ the variety of all linear operators $L^2(\Omega, \mathbb{R}^{n_j}) \to L^2(\Omega, \mathbb{R}^m)$ of rank at most r_j where $k \leq \min\{m, n_j\}$. For convenience we will sometimes write $\mathcal{R}_{r_j}$ instead of $\mathcal{R}(m, n_j, r_j)$. Then, for $\mathcal{F}_j = \mathcal{T}_j\mathcal{B}_j$, the generic problem in (5) can equivalently be reformulated as follows: Find $\mathcal{F}_1, \ldots \mathcal{F}_p$ that solve

$$\min_{\mathcal{F}_1 \in \mathcal{R}_{r_1}, \ldots, \mathcal{F}_p \in \mathcal{R}_{r_p}} \left\| \mathbf{x} - \sum_{j=1}^{p} \mathcal{F}_j(\mathbf{y}_j) \right\|_{\Omega}^2. \tag{6}$$

Recall that $r_j < \min\{m, n_j\}$, for $j = 1, \ldots, p$.

2) Second Degree WSN: Mathematically speaking, the problem in (6) is the problem of the best constrained *linear* approximation. At the same time, it is known that a 'proper' *non-linear* approximation would provide better associated accuracy, i.e., a better estimation of $\mathbf{x}$. This follows, in particular, from the famous Weierstrass approximation theorem [46] and its generalizations in [47], [48], [49], [50], [51], [52], [53].[5] In [1], this observation has been realized as the second degree transform. For $p = 1$ and $\mathcal{N}_1 = I$ in (5), it has been shown in [1] that if

$$\mathcal{T}_1\mathcal{B}_1(\mathbf{y}_1) = \mathcal{A}_1(\mathbf{y}) + \mathcal{A}_2(\mathbf{y}^2), \tag{7}$$

where $\mathbf{y}^2$ is given by $\mathbf{y}^2(\omega) := ([\mathbf{y}_1(\omega)]^2, \ldots, [\mathbf{y}_n(\omega)]^2)^T$, and operators $\mathcal{A}_1$ and $\mathcal{A}_2$ are determined from the minimization of the associated MSE, then under a quite unrestrictive condition obtained in [1], the error associated with the SDT [1] is less than the error associated with the GKLT, for the same compression ratio. The improvement in [1] is due to doubling of the number of parameters compared to the KLT. Indeed, the KLT is a particular case of transform [1] if $\mathcal{A}_2 = \mathbb{O}$ in (7), where $\mathbb{O}$ is zero operator. In the general case, optimization of $\mathcal{T}_1\mathcal{B}_1$ in (7) is achieved by a variation of two operators, $\mathcal{A}_1$ and $\mathcal{A}_2$, while the KLT is obtained on the basis of optimization of $\mathcal{A}_1$ only. More details are provided in Section X-A.

Based on these observations, we now extend the approach in [1] to the case of arbitrary p in (5) and combine it with the MBI method [2], [3] and results in [4], [5]. To this end, we write

$$\mathcal{B}_j(\mathbf{y}_j) = \mathcal{S}_{j,0} + \mathcal{S}_{j,1}(\mathbf{y}_j) + \mathcal{S}_{j,2}(\mathbf{y}_j^2), \tag{8}$$

where a random vector $\mathcal{S}_{j,0} \in L^2(\Omega, \mathbb{R}^{r_j})$, and linear operators $\mathcal{S}_{j,1} : L^2(\Omega, \mathbb{R}^{n_j}) \to L^2(\Omega, \mathbb{R}^{r_j})$ and $\mathcal{S}_{j,2} : L^2(\Omega, \mathbb{R}^{n_j}) \to L^2(\Omega, \mathbb{R}^{r_j})$ are to be determined, and $\mathbf{y}_j^2$ is defined by $\mathbf{y}_j^2(\omega) = ([\mathbf{y}^{(q_{j-1}+1)}(\omega)]^2, \ldots, [\mathbf{y}^{(q_j)}(\omega)]^2)^T$, for all $\omega \in \Omega$, where, as before, $q_j = q_{j-1} + n_j$, $j = 1, \ldots, p$ and $q_0 = 0$. In (8), the term $\mathcal{S}_{j,2}(\mathbf{y}_j^2)$ can be interpreted as a *'second degree term'*. In this regard, $\mathcal{B}_j(\mathbf{y}_j)$ is called the second degree polynomial. Furthermore, $\mathcal{B}_j(\mathbf{y}_j)$ can also be written as

$$\mathcal{B}_j(\mathbf{y}_j) = \mathcal{S}_j(\mathbf{z}_j) \tag{9}$$

where

$$\mathcal{S}_j = [\mathcal{S}_{j,0}, \mathcal{S}_{j,1}, \mathcal{S}_{j,2}] \quad \text{and} \quad \mathbf{z}_j = [1, \mathbf{y}_j^T, (\mathbf{y}_j^2)^T]^T. \tag{10}$$

The above implies the following definition.

Definition 1: Let $\mathcal{S} = \text{diag}[\mathcal{S}_1, \ldots \mathcal{S}_p]$, $\mathbf{z} = [\mathbf{z}_1^T, \ldots, \mathbf{z}_p^T]^T$ and $\mathcal{P}_j = \mathcal{T}_j\mathcal{S}_j$, for $j = 1, \ldots, p$. The WSN represented by the operator $\mathcal{P} = \mathcal{T}\mathcal{S}$ such that

$$\begin{aligned} \mathcal{P}(\mathbf{z}) &= \sum_{j=1}^{p} \mathcal{P}_j(\mathbf{z}_j) \\ &= \sum_{j=1}^{p} \mathcal{T}_j \left[\mathcal{S}_{j,0} + \mathcal{S}_{j,1}(\mathbf{y}_j) + \mathcal{S}_{j,2}(\mathbf{y}_j^2) \right], \end{aligned} \tag{11}$$

[5] A constructive choice of the optimal non-linear approximation, which provides the minimal associated error, is a special and hard research problem [54]. Its solution is subject to special conditions (convexity, for example) that are not satisfied for the problem under consideration. This is why in [7], [8], [9], [10], [11], [12], [13], [14], the special approaches for solving problem (6) have been developed.

where

$$\begin{aligned}\mathcal{P}_j(\mathbf{z}_j) &= \mathcal{T}_j\mathcal{S}_j(\mathbf{z}_j)\\ &= \mathcal{T}_j\left[\mathcal{S}_{j,0}+\mathcal{S}_{j,1}(\mathbf{y}_j)+\mathcal{S}_{j,2}(\mathbf{y}_j^2)\right] \qquad (12)\end{aligned}$$

will be called the *second degree* WSN. If $\mathcal{S}_{j,0}$ and $\mathcal{S}_{j,2}$ are the zero random vector and zero operator, respectively, for all $j=1,\dots,p$, then the WSN will be called the *linear* WSN.

C. Statement of the Problem for Second Degree WSN

On the basis of (8)-(11), for the second degree WSN, the generic problem in (5) is reformulated as follows. For $j=1,\dots,p$, let $\mathcal{B}_j$ and $\mathbf{z}_j$ be represented by (8)-(10), and $\mathcal{P}_j$ be as in (12). Find $\mathcal{P}_1,\dots\mathcal{P}_p$ that solve

$$\min_{\mathcal{P}_1\in\mathcal{R}_{r_1},\dots,\mathcal{P}_p\in\mathcal{R}_{r_p}}\left\|\mathbf{x}-\sum_{j=1}^{p}\mathcal{P}_j(\mathbf{z}_j)\right\|_\Omega^2. \qquad (13)$$

Note that $\mathcal{P}_j$ is linear with respect to $\mathbf{z}_j$ and non-linear with respect to $\mathbf{y}_j$. At the same time, we are interested in the optimal models for $\mathcal{B}_j$, for $j=1,\dots,p$, and $\mathcal{T}$. It will be show in Section III, that the models follow from the solution of problem (13).

III. Solution of Problem (13)

A. Greedy Approach to Determining $\mathcal{P}_1,\dots,\mathcal{P}_p$ that Solve (13)

We wish to find $\mathcal{P}_1,\dots,\mathcal{P}_p$ that provide a solution to the problem in (13) for an arbitrary finite number of sensors in the WSN model, i.e. for $p=1,2,\dots$ in (13). To this end, we need some more preliminaries which are given in Sections III-A1 and III-A2 that follow. The method itself and an associated algorithm are then represented in Section III-A3.

1) Reduction of Problem (13) to Equivalent Form: Let us set $M^{1/2\dagger}=(M^{1/2})^\dagger$, where $M^{1/2}$ is a square root of a matrix M, i.e. $M=M^{1/2}M^{1/2}$. The pseudo-inverse for a matrix M is denoted by $M^\dagger$.

We denote $P=[P_1,\dots,P_p]$, where $P\in\mathbb{R}^{m\times(2n+p)}$ and $P_j\in\mathbb{R}^{m\times(2n_j+1)}$, for all $j=1,\dots,p$, $\mathcal{P}_j(\mathbf{z}_j)=P_j\mathbf{z}_j$ and write $\|\cdot\|$ for the Frobenius norm. Then

$$\begin{aligned}\left\|\mathbf{x}-\sum_{j=1}^{p}\mathcal{P}_j(\mathbf{z}_j)\right\|_\Omega^2 &= \left\|\mathbf{x}-\sum_{j=1}^{p}P_j\mathbf{z}_j\right\|_\Omega^2\\ &= \|\mathbf{x}-P(\mathbf{z})\|_\Omega^2\\ &= \|E_{xx}^{1/2}\|^2-\|E_{xz}(E_{zz}^{1/2})^\dagger\|^2\\ &\quad+\|E_{xz}(E_{zz}^{1/2})^\dagger-PE_{zz}^{1/2}\|^2. \qquad (14)\end{aligned}$$

Let us write $H=E_{xz}(E_{zz}^{1/2})^\dagger$ and represent matrix $E_{zz}^{1/2}$ in blocks

$$E_{zz}^{1/2}=[G_1^T,\dots,G_p^T]^T \qquad (15)$$

where $E_{zz}^{1/2}\in\mathbb{R}^{(2n+p)\times(2n+p)}$ and $G_j\in\mathbb{R}^{(2n_j+1)\times(2n+p)}$, for $j=1,\dots,p$. Then

$$E_{xz}(E_{zz}^{1/2})^\dagger-PE_{zz}^{1/2}=H-\sum_{j=1}^{p}P_jG_j. \qquad (16)$$

Therefore, (14) and (16) imply

$$\begin{aligned}&\min_{P_1\in\mathcal{R}_{r_1},\dots,P_p\in\mathcal{R}_{r_p}}\left\|\mathbf{x}-\sum_{j=1}^{p}P_j(\mathbf{z}_j)\right\|_\Omega^2\\ &=\min_{P_1\in\mathbb{R}_{r_1},\dots,P_p\in\mathbb{R}_{r_p}}\left\|H-\sum_{j=1}^{p}P_jG_j\right\|^2 \qquad (17)\end{aligned}$$

where $\mathbb{R}_{r_j}$ denotes the variety of all $m\times(2n_j+1)$ matrices of rank at most r_j, for $j=1,\dots,p$, where $r_j\le\min\{m,n_j\}$. On the basis of (17), problem (13) and the problem

$$\min_{P_1\in\mathbb{R}_{r_1},\dots,P_p\in\mathbb{R}_{r_1}}\left\|H-\sum_{j=1}^{p}P_jG_j\right\|^2 \qquad (18)$$

are equivalent. Therefore, below we consider problem (18). Note that

$$\left\|H-\sum_{j=1}^{p}P_jG_j\right\|^2=\|Q_j-P_jG_j\|^2, \qquad (19)$$

where $Q_j=H-\sum_{\substack{i=1\\ i\neq j}}^{p}P_iG_i$. This representation will be used below.

2) SVD and Orthogonal Projections : Let the SVD of a matrix $C\in\mathbb{R}^{m\times s}$ be given by

$$C=U_C\Sigma_CV_C^T, \qquad (20)$$

where $U_C\in\mathbb{R}^{m\times m}$ and $V_C\in\mathbb{R}^{s\times s}$ are unitary matrices, $\Sigma_C=\operatorname{diag}(\sigma_1(C),\dots,\sigma_{\min(m,s)}(C))\in\mathbb{R}^{m\times s}$ is a generalized diagonal matrix, with the singular values $\sigma_1(C)\ge\sigma_2(C)\ge\dots 0$ on the main diagonal. Let $U_C=[u_1\ u_2\ \dots\ u_m]$ and $V_C=[v_1\ v_2\ \dots\ v_s]$ be the representations of U and V in terms of their m and s columns, respectively. Let $L_C\in\mathbb{R}^{m\times m}$ and $R_C\in\mathbb{R}^{s\times s}$, such that

$$L_C=\sum_{i=1}^{\operatorname{rank}C}u_iu_i^T \quad\text{and}\quad R_C=\sum_{i=1}^{\operatorname{rank}C}v_iv_i^T, \qquad (21)$$

be the orthogonal projections on the range of C and C^T, correspondingly. Define $C_r\in\mathbb{R}^{m\times s}$ such that

$$C_r=[C]_r=\sum_{i=1}^{r}\sigma_i(C)u_iv_i^T=U_{C_r}\Sigma_{C_r}V_{C_r}^T \qquad (22)$$

for $r=1,\dots,\operatorname{rank}C$, where

$$\begin{aligned}U_{C_r}&=[u_1\ u_2\ \dots\ u_r],\\ \Sigma_{C_r}&=\operatorname{diag}(\sigma_1(C),\dots,\sigma_r(C)),\end{aligned} \qquad (23)$$

$$V_{C_r} = [v_1\ v_2\ \ldots\ v_r].$$

For $r >$ rank C, we write $C^{(r)} = C(= C_{\text{rank } C})$. For $1 \le r <$ rank C, the matrix $C^{(r)}$ is uniquely defined if and only if $\sigma_r(C) > \sigma_{r+1}(C)$.

3) Greedy Method for Solution of Problem (18): An exact solution of problem (18) is unknown. That is why, similar to previous works [7]-[17], [34], we adopt a greedy approach for its solution.[6] In particular, in [7]-[17], [34], the BCD method [24] (mentioned in Section I-B) was used. At the same time, the conditions for convergence of the BCD method are not satisfied for problem (18). The recently developed MBI method [2], [3] avoids the restrictions of the BCD method. The advantage and main idea of the MBI method is that it accepts an "update of the block of variables that achieves the maximum improvement" [3] at each iteration. Further, the problem in (18) is different from that considered in [7]-[17], [34]. Therefore, unlike methods in [7]-[17], [34], a solution of problem (18) is represented by a version of the MBI method [2], [3] where at each iteration the result from [4], [5] is exploited; this is due to the specific form of the objective function in (18). More precisely, each iteration of the proposed method is based on the following result.

Theorem 1: Let $K_j = M_j\left(I - L_{G_j}\right)$ where M_j is an arbitrary matrix. For given $P_1, \ldots, P_{j-1}, P_{j+1}, \ldots, P_p$, the optimal matrix $\widehat{P}_j$ of rank at most r_j minimizing

$$\left\| \mathbf{x} - \sum_{j=1}^{p} \mathcal{P}_j(\mathbf{z}_j) \right\|_{\Omega}^2$$

is given by

$$\widehat{P}_j = \left[Q_j R_{G_j}\right]_{r_j} G_j^{\dagger}(I + K_j), \quad \text{for } j = 1, \ldots, p. \tag{24}$$

where Q_j and G_j are as in (15) and (19), and R_{G_j} and $[\cdot]_{r_j}$ are defined similarly to (21) and (22), respectively. Any minimizing $\widehat{P}_j$ has the above form if and only if either $r_j \ge$ rank $(Q_j R_{G_j})$ or $1 \le r_j <$ rank $(Q_j R_{G_j})$ and $\sigma_{r_j}(Q_j R_{G_j}) > \sigma_{r_j+1}(Q_j R_{G_j})$, where $\sigma_{r_j}(Q_j R_{G_j})$ is a singular value in the SVD for matrix $Q_j R_{G_j}$.

Proof: The proof is considered in Section X-C. We note that the arbitrary matrix M_j implies non-uniqueness of the optimal matrix $\widehat{P}_j$. ■

We represent the error associated with the optimal matrix $\widehat{P}_j$ in the way similar to that used in [7], [11]. Let us write $\overline{\mathbf{x}} = \mathbf{x} - \mathbf{w}_j$, where $\mathbf{w}_j = \sum_{\substack{i=1 \\ i \ne j}}^{p} \mathcal{P}_i \mathbf{z}_i$.Denote by $\delta_1, ..., \delta_{r_j}$ the first r_j eigenvalues in the SVD for matrix $E_{\overline{x}z_j} E_{z_j z_j}^{\dagger} E_{z_j \overline{x}}$, and by $\mu_{j,1}, ..., \mu_{j,m_j}$ the eigenvalues of $C_j H_j^{\dagger} C_j^T$, where $C_j = E_{\overline{x}y_j^2} - E_{\overline{x}y_j} E_{y_j y_j}^{\dagger} E_{y_j y_j^2}$ and $H_j = E_{y_j^2 y_j^2} - E_{y_j^2 y_j} E_{y_j y_j}^{\dagger} E_{y_j y_j^2}$. We also write $\beta_j = \text{tr}\{2\, E_{w_j x} - E_{w_j w_j}\}$ and

$$f(P_1, ..., P_p) = \left\| \mathbf{x} - \sum_{j=1}^{p} \mathcal{P}_j(\mathbf{z}_j) \right\|_{\Omega}^2.$$

[6]Other reason to apply the greedy approach is mentioned at the end of Section X-A.

Theorem 2: For given $P_1, \ldots, P_{j-1}, P_{j+1}, \ldots, P_p$, the error associated with the optimal matrix $\widehat{P}_j$ is given by

$$f(P_1, ..., P_{j-1}, \widehat{P}_j, P_{j+1}, ..., P_p) = \text{tr}\{E_{xx}\} - \sum_{i=1}^{r_j} \delta_i - \sum_{i=1}^{m} \mu_{j,i} - \beta_j. \tag{25}$$

Proof: The proof is considered in Section X-C. If the given matrices $P_1, \ldots, P_{j-1}, P_{j+1}, \ldots, P_p$ are optimal in the sense of minimizing $\left\| \mathbf{x} - \sum_{j=1}^{p} \mathcal{P}_j(\mathbf{z}_j) \right\|_{\Omega}^2$ then Theorem 1 returns the globally optimal solution of (13). ■

Further, let us denote $\mathbb{R}_{r_1,\ldots,r_p} = \mathbb{R}_{r_1} \times ... \times \mathbb{R}_{r_p}$, $\boldsymbol{P} = (P_1, ..., P_p) \in \mathbb{R}_{r_1,\ldots,r_p}$ and $\phi(\boldsymbol{P}) = \left\| H - \sum_{j=1}^{p} P_j G_j \right\|^2$. The computational procedure for the solution of problem (13) is based on the solution of problem (18) and consists of the following steps.

1*st step.* Given $\boldsymbol{P}^{(0)} = (P_1^{(0)}, ..., P_p^{(0)}) \in \mathbb{R}_{r_1,\ldots,r_p}$, compute $\widehat{P}_j^{(1)}$, for $j = 1, \ldots, p$, such that

$$\widehat{P}_j^{(1)} = \left[Q_j^{(0)} R_{G_j}\right]_{r_j} G_j^{\dagger}(I + K_j), \tag{26}$$

where

$$Q_j^{(0)} = H - \sum_{\substack{i=1 \\ i \ne j}}^{p} P_i^{(0)} G_i.$$

Here, G_j is given by (19), and $\widehat{P}_j^{(1)}$ is evaluated in the form (26) on the basis of Theorem 1 and using $P_1^{(0)}, ..., P_{j-1}^{(0)}, P_{j+1}^{(0)}, \ldots, P_p^{(0)}$. Matrices E_{xz} and E_{zz} are formed from matrices E_{xy} and E_{yy}. This is illustrated below by Example 1 in Section IV. A choice of initial iterations $P_1^{(0)}, \ldots, P_p^{(0)}$, is considered in Section III-B below.

2*nd step.* Denote

$$\overline{\boldsymbol{P}}_j^{(1)} = \left(P_1^{(0)}, \ldots, P_{j-1}^{(0)}, \widehat{P}_j^{(1)}, P_{j+1}^{(0)}, \ldots, P_p^{(0)}\right) \tag{27}$$

and select $\overline{\boldsymbol{P}}_k^{(1)}$, for $k = 1, \ldots, p$, such that $\phi(\overline{\boldsymbol{P}}_k^{(1)})$ is minimal among all $\phi(\overline{\boldsymbol{P}}_1^{(1)}), \ldots, \phi(\overline{\boldsymbol{P}}_k^{(1)}), \ldots, \phi(\overline{\boldsymbol{P}}_p^{(1)})$, i.e.

$$\overline{\boldsymbol{P}}_k^{(1)} = \arg \min_{\overline{\boldsymbol{P}}_1^{(1)}, \ldots, \overline{\boldsymbol{P}}_p^{(1)}} \left\{ \phi(\overline{\boldsymbol{P}}_1^{(1)}), \ldots, \phi(\overline{\boldsymbol{P}}_k^{(1)}), \ldots, \phi(\overline{\boldsymbol{P}}_p^{(1)}) \right\} \tag{28}$$

and write

$$\boldsymbol{P}^{(1)} = \overline{\boldsymbol{P}}_k^{(1)}, \tag{29}$$

where we denote $\boldsymbol{P}^{(1)} = (P_1^{(1)}, ..., P_p^{(1)}) \in \mathbb{R}_{r_1,\ldots,r_p}$.

Then we repeat procedure (26)-(29) with the replacement of $\boldsymbol{P}^{(0)}$ by $\boldsymbol{P}^{(1)}$ as follows: Given $\boldsymbol{P}^{(1)} = (P_1^{(1)}, ..., P_p^{(1)})$, compute, for $j = 1, \ldots, p$,

$$\widehat{P}_j^{(2)} = \left[Q_j^{(1)} R_{G_j}\right]_{r_j} G_j^{\dagger}(I + K_j),$$

where

$$Q_j^{(1)} = H - \sum_{\substack{i=1 \\ i\neq j}}^{p} P_i^{(1)} G_i.$$

Note that $\widehat{P}_j^{(2)}$ is computed similar to $\widehat{P}_j^{(1)}$, i.e., on the basis of Theorem 1 and using $P_1^{(1)}, ..., P_{j-1}^{(1)}, P_{j+1}^{(1)}, \ldots, P_p^{(1)}$. Then denote $\overline{\boldsymbol{P}}_j^{(2)} = \left(P_1^{(1)}, \ldots, P_{j-1}^{(1)}, \widehat{P}_j^{(2)}, P_{j+1}^{(1)}, \ldots, P_p^{(1)}\right)$, select $\overline{\boldsymbol{P}}_k^{(2)}$ that satisfies (28) where superscript (1) is replaced with superscript (2), and set $\boldsymbol{P}^{(2)} = \overline{\boldsymbol{P}}_k^{(2)}$ where $\boldsymbol{P}^{(2)} = (P_1^{(2)}, ..., P_p^{(2)}) \in \mathbb{R}_{r_1,...,r_p}$.

This process is continued up to the qth step when a given tolerance $\epsilon \geq 0$ is achieved in the sense

$$|\phi(\boldsymbol{P}^{(q+1)}) - \phi(\boldsymbol{P}^{(q)})| \leq \epsilon, \quad \text{for } q = 1, 2, \ldots. \tag{30}$$

It is summarized as follows.

Algorithm 1: Greedy solution of problem (18)

Initialization: $\boldsymbol{P}^{(0)}$, H, $G_1, ..., G_p$ and $\epsilon > 0$.

1. **for** $q = 0, 1, 2, ...$
2. **for** $j = 1, 2, ..., p$
3. $Q_j^{(q)} = H - \sum_{i=1,\, i\neq j}^{p} P_i^{(q)} G_i$
4. $\widehat{P}_j^{(q+1)} = \left[Q_j^{(q)} R_{G_j}\right]_{r_j} G_j^{\dagger}(I + K_j)$
5. $\overline{\boldsymbol{P}}_j^{(q+1)} = \left(P_1^{(q)}, ..., P_{j-1}^{(q)}, \widehat{P}_j^{(q+1)}, P_{j+1}^{(q)}, ..., P_p^{(q)}\right)$
6. **end**
7. Choose $\overline{\boldsymbol{P}}_k^{(q+1)}$ such that

$$\overline{\boldsymbol{P}}_k^{(q+1)} = \arg \min_{\overline{\boldsymbol{P}}_1^{(q+1)}, ..., \overline{\boldsymbol{P}}_p^{(q+1)}} \left\{\phi(\overline{\boldsymbol{P}}_1^{(q+1)}), \ldots, \phi(\overline{\boldsymbol{P}}_p^{(q+1)})\right\}$$

8. Set $\boldsymbol{P}^{(q+1)} := \overline{\boldsymbol{P}}_k^{(q+1)}$, where $\boldsymbol{P}^{(q+1)} = (P_1^{(q+1)}, ..., P_p^{(q+1)})$
9. **If** $|\phi(\boldsymbol{P}^{(q+1)}) - \phi(\boldsymbol{P}^{(q)})| \leq \epsilon$
10. **Stop**
11. **end**
12. **end**

Sequence $\{\boldsymbol{P}^{(q+1)}\}$ converges to a coordinate-wise minimum point of objective function $\phi(\boldsymbol{P})$ which is a local minimum of (18). Section X-C provides more associated details.

Theorem 3: For $\boldsymbol{P}^{(q+1)}$ determined by Algorithm 1, the error associated with the proposed model of the second order WSN is represented as

$$\|\mathbf{x} - \boldsymbol{P}^{(q+1)}(\mathbf{z})\|_{\Omega}^2 = \|E_{xx}^{1/2}\|^2 - \|E_{xz}(E_{zz}^{1/2})^{\dagger}\|^2 + \|E_{xz}(E_{zz}^{1/2})^{\dagger} - \boldsymbol{P}^{(q+1)} E_{zz}^{1/2}\|^2. \tag{31}$$

Proof: The proof follows from (14). ∎

Remark 1: Although (31) represents a posteriori error, it is convenient to use during a testing phase.

Remark 2: For $p = 1$, the formula in (26) coincides with the second degree transform (SDT) proposed in [1]. Therefore, the transform represented by $\mathcal{P}(\mathbf{z}) = \sum_{j=1}^{p} \mathcal{P}_j(\mathbf{z}_j)$ (see Definition 1) where P_j solves $\min_{P_j \in \mathbb{R}_{r_j}} \|H - \sum_{i=1}^{p} P_i G_i\|^2$, for $j = 1, \ldots, p$, can be regarded the multi-compressor SDT. Further, Algorithm 1 represents the version of the MBI method that uses the multi-compressor SDT. Therefore, the WSN model in the form $\mathcal{P}^{(q+1)}(\mathbf{z}) = \sum_{j=1}^{p} \mathcal{P}_j^{(q+1)}(\mathbf{z}_j)$ where $P_1^{(q+1)}, \ldots, P_p^{(q+1)}$ are determined by Algorithm 1 can be interpreted as a transform as well. We call this transform the multi-compressor SDT-MBI.

B. Determination of Initial Iterations

We determine initial iterations $P_j^{(0)}$, for $j = 1, \ldots, p$, for Algorithm 1 as follows. Let us denote $\mathbf{x} = [\mathbf{x}_1^T, \ldots, \mathbf{x}_p^T]^T$ where $\mathbf{x}_i \in L^2(\Omega, \mathbb{R}^{m_i})$, $i = 1, \ldots, p$ and $m_1 + \ldots + m_p = m$. Suppose that matrix $T \in \mathbb{R}^{m\times r}$ is given by $T = \text{diag}(T_{11}, \ldots, T_{pp})$ where $T_{jj} \in \mathbb{R}^{m_j \times r_j}$, for $j = 1, \ldots, p$. Then

$$\begin{aligned} & \left\| \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_p \end{bmatrix} - \text{diag}(\mathcal{T}_{11}, \ldots, \mathcal{T}_{pp}) \begin{bmatrix} \mathcal{S}_1(\mathbf{z}_1) \\ \vdots \\ \mathcal{S}_p(\mathbf{z}_p) \end{bmatrix} \right\|_{\Omega}^2 \\ = & \left\| \begin{bmatrix} \mathbf{x}_1 - \mathcal{P}_1(\mathbf{z}_1) \\ \vdots \\ \mathbf{x}_p - \mathcal{P}_p(\mathbf{z}_p) \end{bmatrix} \right\|_{\Omega}^2 \\ = & \sum_{j=1}^{p} \|\mathbf{x}_j - \mathcal{P}_j(\mathbf{z}_j)\|_{\Omega}^2 \end{aligned} \tag{32}$$

and

$$\min_{\mathcal{P}_1 \in \mathcal{R}_{r_1}, \ldots, \mathcal{P}_p \in \mathcal{R}_{r_p}} \sum_{j=1}^{p} \|\mathbf{x}_j - \mathcal{P}_j(\mathbf{z}_j)\|_{\Omega}^2 = \sum_{j=1}^{p} \min_{P_j \in \mathcal{R}_{r_j}} \|\mathbf{x}_j - \mathcal{P}_j(\mathbf{z}_j)\|_{\Omega}^2.$$

As a result, in this case, problem (13) is reduced to the problem of finding $\mathcal{P}_j$ that solves

$$\min_{\mathcal{P}_j \in \mathcal{R}_{r_j}} \|\mathbf{x}_j - \mathcal{P}_j(\mathbf{z}_j)\|_{\Omega}^2, \tag{33}$$

for $j = 1, \ldots, p$. Its solution is given by [1]

$$\widehat{P}_j = \left[E_{x_j z_j} (E_{z_j z_j}^{\dagger})^{1/2}\right]_{r_j} (E_{z_j z_j}^{\dagger})^{1/2} (I + \widetilde{K} j) \tag{34}$$

where $\widetilde{K} j = \widetilde{M}_j \left(I - L_{E_{z_j z_j}^{1/2}}\right)$ and $\widetilde{M}_j$ is an arbitrary matrix. Then the initial iterations for Algorithm 1 are defined by

$$P_j^{(0)} = \widehat{P}_j, \quad \text{for } j = 1, \ldots, p. \tag{35}$$

C. Models of Sensors and Fusion Center

To determine the sensor model B_j in the form (8)–(10), for $j = 1, \ldots, p$, and the fusion center model $T = [T_1, \ldots, T_p]$ we need to determine $S_{j,0}$ $S_{j,1}$, $S_{j,2}$ such that

$S_j = [S_{j,0}, S_{j,1}, S_{j,2}]$ and T_j, for $j = 1, \ldots, p$. Matrices $S_{j,0}$, $S_{j,1}$, $S_{j,2}$ and T_j are evaluated by $S_{j,0}^{(q+1)} \in \mathbb{R}^{r_j}$, $S_{j,1}^{(q+1)} \in \mathbb{R}^{r_j \times (2n_j+1)}$, $S_{j,2}^{(q+1)} \in \mathbb{R}^{r_j \times (2n_j+1)}$ and $T_j^{(q+1)} \in \mathbb{R}^{m \times r_j}$, respectively, as follows. By Algorithm 1, the mathematical model of the second degree WSN is given by

$$\begin{aligned} \boldsymbol{P}^{(q+1)}(\mathbf{z}) &= \sum_{j=1}^{p} P_j^{(q+1)}(\mathbf{z}_j) \\ &= \sum_{j=1}^{p} T_j^{(q+1)} S_j^{(q+1)}(\mathbf{z}_j) \\ &= T^{(q+1)} \begin{pmatrix} S_1^{(q+1)}(\mathbf{z}_1) \\ \vdots \\ S_p^{(q+1)}(\mathbf{z}_p) \end{pmatrix}, \end{aligned} \tag{36}$$

where $\boldsymbol{P}^{(q+1)} = (P_1^{(q+1)}, ..., P_p^{(q+1)})$ and $P_j^{(q+1)} = T_j^{(q+1)} S_j^{(q+1)}$. By step 7 of Algorithm 1,

$$P_j^{(q+1)} = \widehat{P}_j^{(q+1)} = \left[Q_j^{(q)} R_{G_j} \right]_{r_j} G_j^{\dagger}(I + K_j)$$

or $P_j^{(q+1)} = P_j^{(0)}$, where $P_j^{(0)}$ is represented by (34)-(35). For the case when $P_j^{(q+1)} = \widehat{P}_j^{(q+1)}$, we write

$$U_{r_j} \Sigma_{r_j} V_{r_j}^T = \left[Q_j^{(q)} R_{G_j} \right]_{r_j}, \tag{37}$$

where $U_{r_j} \Sigma_{r_j} V_{r_j}^T$ is the truncated SVD taken with first r_j singular values, $U_{r_j} \in \mathbb{R}^{m \times r_j}$, $\Sigma_{r_j} \in \mathbb{R}^{r_j \times r_j}$ and $V_{r_j}^T \in \mathbb{R}^{r_j \times (2n+p)}$. Then

$$T_j^{(q+1)} = U_{r_j} \quad \text{or} \quad T_j^{(q+1)} = U_{r_j} \Sigma_{r_j} \tag{38}$$

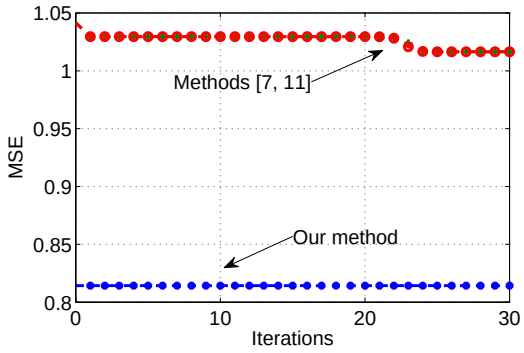
where $T_j^{(q+1)} \in \mathbb{R}^{m \times r_j}$ and $T_j^{(q+1)} \in \mathbb{R}^{m \times r_j}$, and

$$S_j^{(q+1)} = \Sigma_{r_j} V_{r_j}^T G_j^{\dagger}(I + K_j) \quad \text{or} \quad S_j^{(q+1)} = V_{r_j}^T G_j^{\dagger}(I + K_j), \tag{39}$$

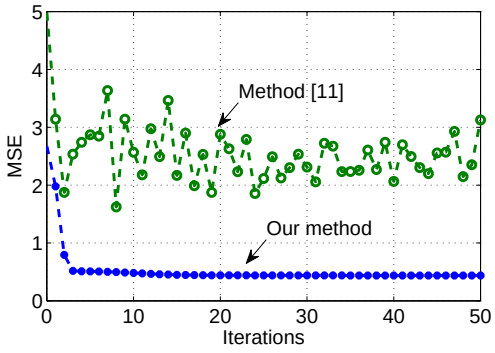
respectively, where $S_j^{(q+1)} \in \mathbb{R}^{r_j \times (2n_j+1)}$. Here, $S_j^{(q+1)} = [S_{j,0}^{(q+1)}, S_{j,1}^{(q+1)}, S_{j,2}^{(q+1)}]$.

As a result, for the jth sensor model represented by (8)–(10), $S_{j,0}$ $S_{j,1}$ and $S_{j,2}$ are given by $S_{j,0}^{(q+1)}$, $S_{j,1}^{(q+1)}$ and $S_{j,2}^{(q+1)}$, respectively, which are determined from (39) as blocks of matrix $S_j^{(q+1)}$. The fusion center model is given by $T^{(q+1)} = [T_1^{(q+1)}, \ldots, T_p^{(q+1)}]$ where $T_j^{(q+1)}$, for $j = 1, \ldots, p$, is determined by (38).

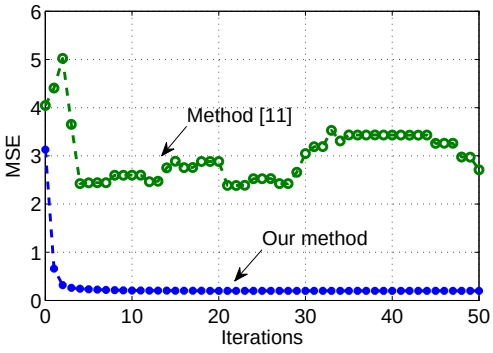
For the case when $P_j^{(q+1)} = P_j^{(0)}$, where $P_j^{(0)}$ is represented by (34)-(35), matrices T_j, $S_{j,0}$ $S_{j,1}$ and $S_{j,2}$ are evaluated similar to matrices $T_j^{(q+1)}$, $S_{j,0}^{(q+1)}$, $S_{j,1}^{(q+1)}$ and $S_{j,2}^{(q+1)}$ in (38) and (39), respectively. In this case, in (37), $\left[Q_j^{(q)} R_{G_j} \right]_{r_j}$ should be replaced with $\left[E_{x_j z_j} (E_{z_j z_j}^{\dagger})^{1/2} \right]_{r_j}$ and in (39), $G_j^{\dagger}(I + K_j)$ should be replaced with $(E_{z_j z_j}^{\dagger})^{1/2}(I + \widetilde{K} j)$.



(a) Example 1



(b) Example 2



(c) Example 2

Fig. 2. Diagrams of MSEs associated with our method and methods [7], [11]. In Fig. (a), MSEs associated with methods [7], [11] coincide.

IV. Numerical Results and Comparisons

We wish to illustrate the advantages of the proposed methodology by numerical examples where a comparison with known methods [7], [8], [9], [10], [11], [34] is provided. The method [11] represents a generalization of methods [8], [9], [10] and therefore, we provide a numerical comparison with method [11] which includes, in fact, a comparison with methods [8], [9], [10] as well. Further, covariance matrices used in the simulations associated with Figs. 2 (b) and (c) are singular, and therefore, method [7] is not applicable (some more associated

observations have been given at the end of Section I-B). Therefore, in Figs. 2 (b) and (c), results related to algorithm in [7] are not given.

For the same reason, the method presented in [34] is not applicable as well. Moreover, the method in [34] is restricted to the case when the covariance matrix formed by the noise vector is block diagonal. This is not the case here.

In the examples below, different types of noisy observed signals and different compression ratios are considered. In all examples, our method provides the better associated accuracy than that for the methods in [7], [11] (and methods in [8], [9], [10] as well, because they follow from [11]).

Example 1: Suppose that the source signal $\mathbf{x} \in L^2(\Omega, \mathbb{R}^3)$ is a Gaussian random vector with mean zero and covariance matrix $E_{xx} = \begin{bmatrix} 1 & 0.64 & 0.08 \\ 0.64 & 1 & 0.08 \\ 0.08 & 0.08 & 1 \end{bmatrix}$. The observed vectors $\mathbf{y}_1$ and $\mathbf{y}_2$ are noisy version of $\mathbf{x}$, i.e., $\mathbf{y}_1 = \mathbf{x} + \boldsymbol{\xi}_1$ and $\mathbf{y}_2 = \mathbf{x} + \boldsymbol{\xi}_2$, where $\boldsymbol{\xi}_1$ and $\boldsymbol{\xi}_2$ are Gaussian random vectors with mean zero and covariance matrices $E_{\xi_1,\xi_1} = \sigma_1^2 I$ and $E_{\xi_2,\xi_2} = \sigma_2^2 I$, for $j = 1, 2$. The vectors $\mathbf{x}$, $\boldsymbol{\xi}_1$ and $\boldsymbol{\xi}_2$ are assumed to be independent. Then $E_{xy} = [E_{xx}\ E_{xx}]$ and $E_{yy} = \begin{bmatrix} E_{y_1y_1} & E_{y_1y_2} \\ E_{y_2y_1} & E_{y_2y_2} \end{bmatrix} = \begin{bmatrix} E_{xx} + E_{\xi_1\xi_1} & E_{xx} \\ E_{xx} & E_{xx} + E_{\xi_2\xi_2} \end{bmatrix}$. For the sake of simplicity, in (11)–(12), we choose $\mathcal{S}_{1,0} = \mathcal{S}_{2,0} = \mathbb{O}$. Therefore, $\mathbf{z}_j = \begin{bmatrix} \mathbf{y}_j \\ {}_2 \\ \mathbf{y}_j \end{bmatrix}$, for$j = 1, 2$, and as before (see Definition 1), $\mathbf{z} = [\mathbf{z}_1^T, \mathbf{z}_2^T]^T$, i.e. $\mathbf{z} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_1^2 \\ \mathbf{y}_2 \\ \mathbf{y}_2^2 \end{bmatrix} = \begin{bmatrix} \mathbf{x} + \xi_1 \\ (\mathbf{x} + \xi_1)^2 \\ \mathbf{x} + \xi_2 \\ (\mathbf{x} + \xi_2)^2 \end{bmatrix} = \begin{bmatrix} \mathbf{x} + \xi_1 \\ \mathbf{x}^2 + \xi_1^2 \\ \mathbf{x} + \xi_2 \\ \mathbf{x}^2 + \xi_2^2 \end{bmatrix}$, because $\mathbf{x}$, $\boldsymbol{\xi}_1$ and $\boldsymbol{\xi}_2$ are independent. Matrices E_{xz} and E_{zz} are evaluated as follows. For a random variable a with the probability density function $f_a(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2\sigma_a^2}\right)$, associated covariances are given by $E_{aa} = \sigma^2$ and $E_{a^2a^2} = 2\sigma^4$. Here, σ^2 is variance. For random variables a and b with the joint probability density function $f_{a,b}(u, v) = \frac{1}{2\pi\sqrt{1-\rho_{a,b}^2}} \exp\left(-\frac{u^2 - 2\rho_{a,b}uv + v^2}{2(1-\rho_{a,b}^2)}\right)$, associated covariances are given by $E_{ab} = \rho_{a,b}$, $E_{a^2b} = E_{ab^2} = 0$, $E_{a^2b^2} = 2\rho_{a,b}^2$ and $E_{a^2a} = E_{aa^2} = 0$, where $\rho_{a,b}$ is the correlation parameter. Then $E_{x^2x^2} = \begin{bmatrix} 2 & 0.8192 & 0.0128 \\ 0.8192 & 2 & 0.0128 \\ 0.0128 & 0.0128 & 2 \end{bmatrix}$, $E_{\xi_1^2\xi_1^2} = 2\sigma_1^4 I$, $E_{\xi_2^2\xi_2^2} = 2\sigma_2^4 I$, $E_{xz} = [E_{xy}\ E_{xy^2}] = [E_{xy_1}\ E_{xy_2}\ E_{xy_1^2}\ E_{xy_1^2}] = [E_{xx}\ E_{xx}\ \mathbb{O}\ \mathbb{O}]$ and E_{zz} is given in (40). On this basis, Algorithm 1 produces the associated errors that were evaluated by (31), for different σ_j and r_j with $j = 1, 2$. The errors associated with methods [7], [11] were estimated in a similar way. A typical example of the errors, for $\sigma_1 = 0.9$, $\sigma_2 = 0.65$ and $r_1 = r_2 = 1$, is represented in Fig. 2 (a).

Example 2: Here, it is supposed that covariance matrices E_{xx}, E_{xy} and E_{yy} are known from a training phase. We assume that, for $j = 1, \ldots, p$, noisy observations are such that $\mathbf{y}_j = \mathcal{A}_j\mathbf{x} + \beta_j\boldsymbol{\xi}_j$, where $\mathcal{A}_j : L^2(\Omega, \mathbb{R}^m) \to L^2(\Omega, \mathbb{R}^m)$ is a linear operator defined by matrix $A_j \in \mathbb{R}^{m\times m}$ with uniformly distributed random entries, $\mathbf{x}$ is a random vector with uniform distribution, $\beta_j \in \mathbb{R}$ and $\boldsymbol{\xi}_j$ is noise represented by a Gaussian random vector with mean zero.

The errors associated with the proposed method and the method in [11], for the case of two and three sensors (i.e., for $p = 2$ and $p = 3$, respectively), and different choices of m, n_j and r_j, for $j = 1, 2$ and $j = 1, 2, 3$, are represented in Figs. 2 (b) and (c). Method [7] is not applicable since covariance matrices used in these simulations are singular. Note that method [11] is not numerically stable in these simulations. We believe this is because of the reason mentioned in Section I-B.

Example 3: Here, we wish to illustrate the obtained theoretical results in a more conspicuous and different way than that in the above examples. To this end, we use training signals themselves, not only covariance matrices as before. In this example, training reference signal $\mathbf{x}$ is simulated by its realizations, i.e. by matrix $\mathbf{X} \in \mathbb{R}^{m\times k}$ where each column represents a realization of the signal. To represent the obtained results in a visible way, signal $\mathbf{X} \in \mathbb{R}^{m\times k}$ is chosen as the known image 'Cameraman' given by the 256×256 matrix – see Fig. 3 (a), i.e. with $m, k = 256$. Then $X \in \mathbb{R}^{256\times 128}$. A sample $X \in \mathbb{R}^{m\times s}$ with $s < k$ is formed from $\boldsymbol{X}$ by choosing the even columns, i.e. $s = 128$.

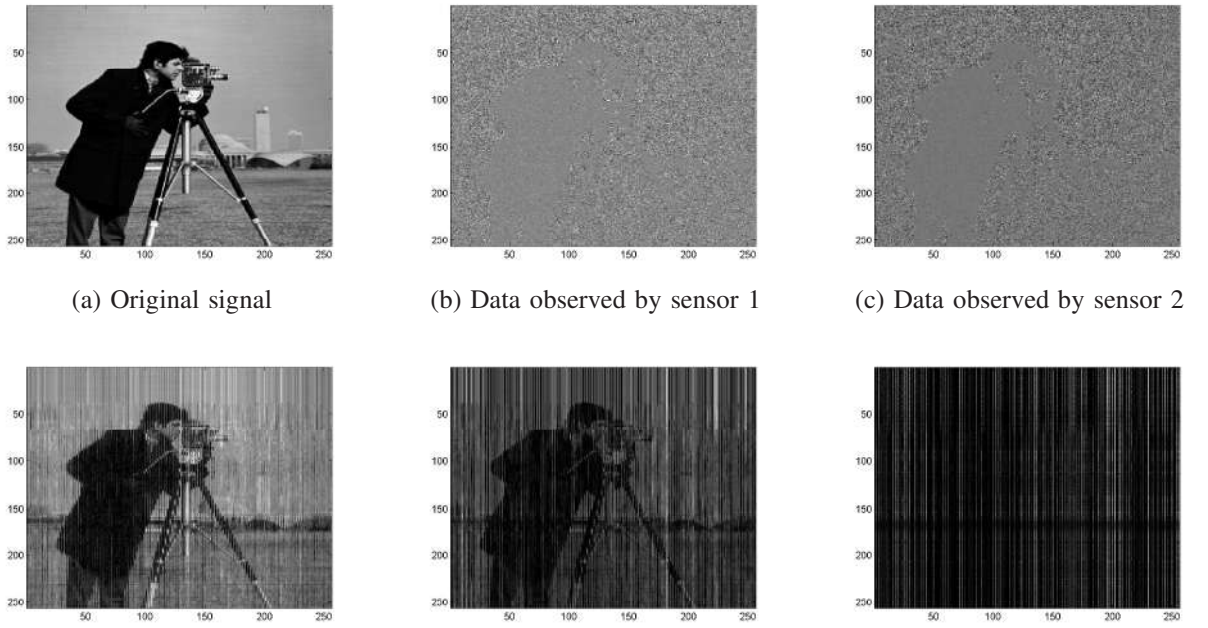
Further, we consider the WSN with two sensors, i.e. with $p = 2$, where the observed signal $\mathbf{Y_j}$, for $j = 1, 2$, is simulated as $\mathbf{Y}_j = A_j * \boldsymbol{X} + \beta_j \Upsilon_j$, where $A_j \in \mathbb{R}^{256\times 256}$ and $\Upsilon_j \in \mathbb{R}^{256\times 256}$ have random entries, chosen from a normal distribution with mean zero and variance one, $A_j * X$ represents the Hadamard matrix product and $\beta_1 = 0.2$ and $\beta_2 = 0.1$.

For $r_1 = r_2 = 128$, the simulation results are represented in Figs. 3 and 4. Our method and methods from [7] and [11] have been applied to the above signals with 50 iterations each. The associated errors are evaluated in the form $\|\mathbf{X}(i) - \widehat{\mathbf{X}}(i)\|_2^2$, for $i = 1, \ldots, 256$, where $\widehat{\mathbf{X}}$ is the reconstruction of $\mathbf{X}$ by the method we use (i.e. by our method or methods in [7] and [11]), and $\mathbf{X}(i)$ and $\widehat{\mathbf{X}}(i)$ are ith columns of matrices $\mathbf{X}$ and $\widehat{\mathbf{X}}$, respectively.

Method [7] has only been developed in terms of the inverses of certain covariance matrices. In this example, the covariance matrices are singular. Consequently, the estimate by method [7] has the form represented in Fig. 3 (f). Due to singularity of covariance matrices, the error associated with method [7] is very large. Therefore, those results are not included in Fig. 4.

Method [7] was applied in terms of pseudo-inverse matrices as it suggested in the section "Appendix I" of [7]. Nevertheless, the associated estimate (see Fig. 3 (e)) is still not so accurate as that by the proposed method. As mentioned in Example 2, it might be, in particular, because of the reason considered in Section I-B.

Similar to the other examples, Figs. 3 and 4 demonstrate a more accurate signal reconstruction associated with the proposed method than those associated with previous methods.



(a) Original signal (b) Data observed by sensor 1 (c) Data observed by sensor 2

(d) Signal estimate by our method (e) Signal estimate by method [11] (f) Signal estimate by method [7]

Fig. 3. Illustration to Example 3

$$
\begin{aligned}
E_{zz} &= \begin{bmatrix} E_{yy} & E_{yy^2} \\ E_{y^2y} & E_{y^2y^2} \end{bmatrix} \\
&= \begin{bmatrix} E_{y_1y_1} & E_{y_1y_2} & E_{y_1y_1^2} & E_{y_1y_2^2} \\ E_{y_2y_1} & E_{y_2y_2} & E_{y_2y_1^2} & E_{y_2y_2^2} \\ E_{y_1^2y_1} & E_{y_1^2y_2} & E_{y_1^2y_1^2} & E_{y_1^2y_2^2} \\ E_{y_2^2y_1} & E_{y_2^2y_2} & E_{y_2^2y_1^2} & E_{y_2^2y_2^2} \end{bmatrix} \\
&= \begin{bmatrix} E_{xx}+E_{\xi_1\xi_1} & E_{xx} & \mathbb{O} & \mathbb{O} \\ E_{xx} & E_{xx}+E_{\xi_2\xi_2} & \mathbb{O} & \mathbb{O} \\ \mathbb{O} & \mathbb{O} & E_{x^2x^2}+E_{\xi_1^2\xi_1^2} & E_{x^2x^2} \\ \mathbb{O} & \mathbb{O} & E_{x^2x^2} & E_{x^2x^2}+E_{\xi_2^2\xi_2^2} \end{bmatrix}.
\end{aligned} \tag{40}
$$

V. Second Degree WSN with Nonideal Channels

In the above sections, we dealt with the WSN model with ideal links between the sensors and the fusion center. Here, we consider nonideal channels which comprise multiplicative signal fadding and additive noise. These effects are modeled by operators $\mathcal{N}_j : L^2(\Omega, \mathbb{R}^{r_j}) \to L^2(\Omega, \mathbb{R}^{r_j})$, for $j = 1, \ldots, p$, which are depicted in Fig. 1 (a). Each $\mathcal{N}_j$ is such that $\mathcal{N}_j(\mathbf{u}_j) = \mathcal{D}_j(\mathbf{u}_j) + \boldsymbol{\eta}_j$ where the channel linear operator $\mathcal{D}_j$ and $\boldsymbol{\eta} \in L^2(\Omega, \mathbb{R}^{r_j})$ is random noise with zero mean which is uncorrelated with $\mathbf{x}$, $\mathbf{y}_j$, $\mathcal{D}_j$ and across the channels, i.e., $E_{\eta_i\eta_j} = \mathbb{O}$ for $i \neq j$. Operator $\mathcal{D}_j$ is represented by matrix $D_j \in \mathbb{R}^{r_j \times r_j}$ similar, in particular, to $\mathcal{B}_j$ in (2). Matrices D_j and $E_{\eta_j\eta_j}$, for $j = 1, \ldots, p$, are available at the fusion center. As in [7] it is assumed that channel matrix D_j and matrices $E_{\eta_j\eta_j}$ can be acquired similar to covariances E_{xy} and E_{yy}. At the same time, unlike [7] here, it is not required that D_j and $E_{\eta_j\eta_j}$ are necessarily invertible.

In the case of nonideal channels, we deal with the problem as follows. Find models of the sensors $\mathcal{B}_1, \ldots, \mathcal{B}_p$ and the fusion center $\mathcal{T} = [\mathcal{T}_1, \ldots, \mathcal{T}_p]$ that solve

$$
\min_{\substack{\mathcal{T}_1,\ldots,\mathcal{T}_p,\\ \mathcal{B}_1,\ldots,\mathcal{B}_p}} \left\| \mathbf{x} - \sum_{j=1}^{p} \mathcal{T}_j \left[\mathcal{D}_j \mathcal{B}_j(\mathbf{y}_j) + \boldsymbol{\eta}_j \right] \right\|_{\Omega}^2 . \tag{41}
$$

On the basis of (9), the above can equivalently be represented as the problem of finding $\mathcal{S}_1, \ldots, \mathcal{S}_p$ and $\mathcal{T}_1, \ldots, \mathcal{T}_p$ that solve

$$
\min_{\substack{\mathcal{T}_1,\ldots,\mathcal{T}_p,\\ \mathcal{S}_1,\ldots,\mathcal{S}_p}} \left\| \mathbf{x} - \sum_{j=1}^{p} \mathcal{T}_j \left[\mathcal{D}_j \mathcal{S}_j(\mathbf{z}_j) + \boldsymbol{\eta}_j \right] \right\|_{\Omega}^2 . \tag{42}
$$

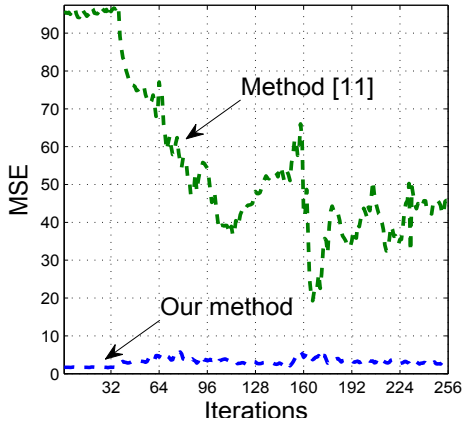


Fig. 4. Example 3: Diagrams of MSEs associated with our method and method [11]. MSE associated with method [7] is very large and, therefore, it is not given here.

A. Greedy Method for Solution of Problem (42)

An exact solution to problem (42) is unknown. By this reason, we apply a greedy approach to (42) in the form of an iterative procedure. At each iteration, the idea of the MBI method is combined with the optimal solutions to two pertinent minimization problems, so that one solution determines the optimal model of the sensor and another solution determines the optimal model of the fusion center. By this reason, it is called the alternating iterative (AI) procedure.

First, in Section V-A1, we provide the solutions to the minimization problems used in the AI procedure. In Section V-A2, a detailed description of the steps used at each iteration of the AI procedure is provided. Then the AI procedure itself is considered in Section V-A3.

1) Preliminaries for AI Procedure: Let us denote

$$\psi(\boldsymbol{P}) = \left\| \mathbf{x} - \sum_{j=1}^{p} \mathcal{T}_j \left[\mathcal{D}_j \mathcal{S}_j(\mathbf{z}_j) + \boldsymbol{\eta}_j \right] \right\|_{\Omega}^2$$

where $\boldsymbol{P} = (T_1, \ldots, T_p, S_1, \ldots, S_p)$.

Theorem 4: Let $\mathbf{w}_j = \mathcal{D}_j \mathcal{S}_j(\mathbf{z}_j) + \boldsymbol{\eta}_j$, for $j = 1, \ldots, p$, and $\mathbf{w} = [\mathbf{w}_1^T, \ldots, \mathbf{w}_p^T]^T$. Let $K = M(I - L_{E_{ww}})$ where M be an arbitrary matrix. For given $S_1, \ldots, S_p$, optimal matrix T minimizing $\psi(\boldsymbol{P})$ is given by

$$T = E_{xw} E_{ww}^{\dagger} (I + K). \tag{43}$$

The associated error is represented by

$$\min_{T} \psi(\boldsymbol{P}) = \|E_{xx}^{1/2}\|^2 - \|E_{xw}(E_{ww}^{1/2})^{\dagger}\|^2. \tag{44}$$

Proof: It follows that $\psi(\boldsymbol{P})$ can be represented as

$$\psi(\boldsymbol{P}) = \|\mathbf{x} - T\mathbf{w}\|_{\Omega}^2. \tag{45}$$

Then (43) and (44) follow from [5]. If given matrices $S_1, \ldots, S_p$ are optimal in the sense of minimizing $\psi(\boldsymbol{P})$ then Theorem 4 returns the globally optimal solution of (42). ■

Theorem 5: Let $\mathbf{x}_{(j)} = \mathbf{x} - \sum_{\substack{i=1 \\ i \neq j}}^{p} \mathcal{T}_i[\mathcal{D}_i \mathcal{S}_i(\mathbf{z}_i) + \boldsymbol{\eta}_i]$. For given T and $\{S_i\}_{i=1, i\neq j}^{p}$, optimal matrix S_j minimizing $\left\|\mathbf{x}_{(j)} - \mathcal{T}_j \mathcal{D}_j \mathcal{S}_j(\mathbf{z}_j)\right\|_{\Omega}^2$ is given by

$$S_j = (T_j D_j)^{\dagger} E_{x_{(j)} z_j} E_{z_j z_j}^{\dagger} (I + K_j), \tag{46}$$

where $K_j = M_j(I - L_{E_{z_j z_j}})$ and M_j is arbitrary. The associated error is represented by

$$\begin{aligned} &\min_{S_j} \left\|\mathbf{x}_{(j)} - \mathcal{T}_j \mathcal{D}_j \mathcal{S}_j(\mathbf{z}_j)\right\|_{\Omega}^2 \\ &= \left\|E_{x_{(j)} x_{(j)}}^{1/2}\right\|^2 - \left\|E_{x_{(j)} z_j} (E_{z_j z_j}^{\dagger})^{1/2}\right\|^2 \\ &\quad + \left\|E_{x_{(j)} z_j} E_{z_j z_j}^{1/2\dagger} \left[I - T_j D_j (T_j D_j)^{\dagger}\right]\right\|^2. \end{aligned} \tag{47}$$

Proof: We write

$$\begin{aligned} &\left\| \mathbf{x} - \sum_{j=1}^{p} \mathcal{T}_j \left[\mathcal{D}_j \mathcal{S}_j(\mathbf{z}_j) + \boldsymbol{\eta}_j \right] \right\|_{\Omega}^2 \\ &= \left\|\mathbf{x}_{(j)} - \mathcal{T}_j \mathcal{D}_j \mathcal{S}_j(\mathbf{z}_j)\right\|_{\Omega}^2 \\ &= \left\|E_{x_{(j)} x_{(j)}}^{1/2}\right\|^2 - \left\|E_{x_{(j)} z_j} (E_{z_j z_j}^{\dagger})^{1/2}\right\|^2 \\ &\quad + \left\|E_{x_{(j)} z_j} E_{z_j z_j}^{1/2\dagger} - T_j D_j S_j E_{z_j z_j}^{1/2}\right\|^2. \end{aligned} \tag{48}$$

By [5], S_j in (46) minimizes $\left\|E_{x_{(j)} z_j} E_{z_j z_j}^{1/2\dagger} - T_j D_j S_j E_{z_j z_j}^{1/2}\right\|^2$ and hence, minimizes

$$\left\|\mathbf{x}_{(j)} - \mathcal{T}_j \mathcal{D}_j \mathcal{S}_j(\mathbf{z}_j)\right\|_{\Omega}^2.$$

The error representation in (47) follows from (46) and (48).

Note that if given matrices T and $\{S_i\}_{i=1, i\neq j}^{p}$ are optimal in the sense of minimizing $\psi(\boldsymbol{P})$ then Theorem 5 returns the globally optimal solution of (42). ■

2) Description of Steps in AI Procedure: Before describing the AI procedure in Section V-A3 that follows, here we describe the steps made at its each iteration. The steps are based on Theorems 4 and 5 provided above.

1*st step.* Given $T^{(0)}, S_1^{(0)}, \ldots, S_p^{(0)}$, compute

$$T^{(1)} = E_{xw^{(0)}} E_{w^{(0)} w^{(0)}}^{\dagger} (I + K^{(1)}). \tag{49}$$

Here, $E_{xw^{(0)}} = [E_{xw_1^{(0)}}, \ldots, E_{xw_p^{(0)}}]$ and $E_{w^{(0)} w^{(0)}} = \{E_{w_i^{(0)} w_j^{(0)}}\}_{i,j=1}^{p}$, where, for $j = 1, \ldots, p$,

$$\begin{aligned} E_{xw_j^{(0)}} &= E_{xz_j} (S_j^{(0)})^T D_j^T, \\ E_{w_i^{(0)} w_j^{(0)}} &= D_i S_i^{(0)} E_{z_i z_j} (S_j^{(0)})^T D_j^T + E_{\eta_i \eta_j}, \\ K^{(1)} &= M^{(1)} (I - L_{E_{w^{(0)} w^{(0)}}}) \end{aligned} \tag{50}$$

where $M^{(1)}$ is an arbitrary matrix. Then $T_1^{(1)}, \ldots, T_p^{(1)}$ are determined as blocks of matrix $T^{(1)} = [T_1^{(1)}, \ldots, T_p^{(1)}]$ given by (49).

2nd step. Given $\{T_1^{(1)}\}_{j=1}^p$ and $\{S_i^{(0)}\}_{i=1,i\neq j}^p$, compute, for $j=1,\ldots,p$,

$$\widehat{S}_j^{(1)} = (T_j^{(1)} D_j)^\dagger E_{x_{(j,0)} z_j} E_{z_j z_j}^\dagger (I + K_j) \tag{51}$$

where

$$E_{x_{(j,0)} z_j} = E_{x z_j} - \sum_{\substack{i=1 \\ i\neq j}}^p T_i^{(1)} D_i S_i^{(0)} E_{z_i z_j}, \tag{52}$$

$$K_j = M_j (I - L_{E_{z_j z_j}}), \tag{53}$$

where M_j is arbitrary. Note that (49) and (51) are evaluated on the basis of (43) and (46), respectively.

3rd step. Denote

$$\overline{\boldsymbol{P}}_j^{(1)} = \left(T_1^{(1)}, ..., T_p^{(1)}, S_1^{(0)}, ..., S_{j-1}^{(0)}, \widehat{S}_j^{(1)}, S_{j+1}^{(0)}, ..., S_p^{(0)}\right), \tag{54}$$

set $S_j^{(1)} := \widehat{S}_j^{(1)}$ and select $\overline{\boldsymbol{P}}_k^{(1)}$, for $k=1,\ldots,p$, such that $\psi(\overline{\boldsymbol{P}}_k^{(1)})$ is minimal among all $\psi(\overline{\boldsymbol{P}}_1^{(1)})$, $\ldots$, $\psi(\overline{\boldsymbol{P}}_k^{(1)})$, $\ldots, \psi(\overline{\boldsymbol{P}}_p^{(1)})$, i.e.,

$$\overline{\boldsymbol{P}}_k^{(1)} = \arg \min_{\overline{\boldsymbol{P}}_1^{(1)},\ldots,\overline{\boldsymbol{P}}_p^{(1)}} \left\{\psi(\overline{\boldsymbol{P}}_1^{(1)}), \ldots, \psi(\overline{\boldsymbol{P}}_k^{(1)}), \ldots, \psi(\overline{\boldsymbol{P}}_p^{(1)})\right\}. \tag{55}$$

Then write

$$\boldsymbol{P}^{(1)} := \overline{\boldsymbol{P}}_k^{(1)} \tag{56}$$

and denote $\boldsymbol{P}^{(1)} = (P_1^{(1)}, ..., P_p^{(1)}) \in \mathbb{R}_{r_1,\ldots,r_p}$. Then we repeat steps (49)–(56) with the replacement of $T^{(0)}, S_1^{(0)}, \ldots, S_p^{(0)}$ by $T^{(1)}, S_1^{(1)}, \ldots, S_p^{(1)}$ as follows. Compute

$$T^{(2)} = E_{x w^{(1)}} E_{w^{(1)} w^{(1)}}^\dagger (I + K^{(2)}),$$

where $E_{x w^{(1)}}$ and $E_{w^{(1)} w^{(1)}}$ are evaluated similar to (50) with subscript (0) replaced by subscript (1). Then similar to (51) compute

$$\widehat{S}_j^{(2)} = (T_j^{(2)} D_j)^\dagger E_{x_{(j,1)} z_j} E_{z_j z_j}^\dagger (I + K_j) \tag{57}$$

where $E_{x_{(j,1)} z_j} = E_{x z_j} - \sum_{\substack{i=1 \\ i\neq j}}^p T_i^{(2)} D_i S_i^{(1)} E_{z_i z_j}$ and continue the computation with other pertinent replacement in the superscripts as it is represented in Algorithm 2 that follows.

3) Algorithm for AI Procedure: It follows from (45) that the cost function can be written as

$$\psi(\boldsymbol{P}) = \left\|E_{xx}^{1/2}\right\|^2 - \left\|E_{xw}(E_{ww}^{1/2})^\dagger\right\|^2 + \left\|E_{xw}(E_{ww}^{1/2})^\dagger - T E_{ww}^{1/2}\right\|^2. \tag{58}$$

The process described above is summarized in Algorithm 2.

Algorithm 2: Greedy approach to problem (42): AI procedure

Initialization: $T^{(0)}, S_1^{(0)}, ..., S_p^{(0)}$, $D_1, ..., D_p$ and $\epsilon > 0$.

1. **for** $q = 0, 1, 2, ...$
2. $\quad E_{x w_j^{(q)}} = E_{x z_j} (S_j^{(q)})^T D_j^T$
3. $\quad E_{w_i^{(q)} w_j^{(q)}} = D_i S_i^{(q)} E_{z_i z_j} (S_j^{(q)})^T D_j^T + E_{\eta_i \eta_j}$
4. $\quad T^{(q+1)} = E_{x w^{(q)}} E_{w^{(q)} w^{(q)}}^\dagger (I + K^{(q+1)})$
5. $\quad T_1^{(q+1)} = T^{(q+1)}(:, 1 : r_1)$
6. $\quad$ **for** $j = 2, 3, ..., p$
7. $\quad\quad k_j = \sum_{i=1}^j r_i$
8. $\quad\quad T_j^{(q+1)} = T^{(q+1)}(:, r_{j-1} + 1 : k_j)$
9. $\quad$ **end**
10. $\quad \overline{\mathbf{P}}_0^{(q+1)} = (T_1^{(q+1)}, , ..., T_p^{(q+1)}, S_1^{(q)}, ...S_p^{(q)})$
11. $\quad$ **for** $j = 1, 2, ..., p$
12. $\quad\quad E_{x_{(j,q)} z_j} = E_{x z_j} - \sum_{\substack{i=1 \\ i\neq j}}^p T_i^{(q+1)} D_i S_i^{(q)} E_{z_i z_j}$
13. $\quad\quad \widehat{S}_j^{(q+1)} = (T_j^{(q+1)} D_j)^\dagger E_{x_{(j,q)} z_j} E_{z_j z_j}^\dagger (I + K_j),$
14. $\quad\quad \overline{\mathbf{P}}_j^{(q+1)} = \left(T^{(q)}, S_1^{(q)}, ..., S_{j-1}^{(q)}, \widehat{S}_j^{(q+1)}, S_{j+1}^{(q)}, ..., , S_p^{(q)}\right)$
15. $\quad\quad S_j^{(q+1)} = \widehat{S}_j^{(q+1)}$
16. $\quad$ **end**
17. $\quad$ Choose $\overline{\boldsymbol{P}}_k^{(q+1)}$ such that $\overline{\boldsymbol{P}}_k^{(q+1)} = \arg \min_{\overline{\boldsymbol{P}}_1^{(q+1)},\ldots,\overline{\boldsymbol{P}}_p^{(q+1)}} \left\{\psi(\bar{\mathbf{P}}_0^{(q+1)}), ..., \psi(\bar{\mathbf{P}}_p^{(q+1)})\right\}$
18. $\quad$ Set $\boldsymbol{P}^{(q+1)} := \overline{\boldsymbol{P}}_k^{(q+1)}$ where $\boldsymbol{P}^{(q+1)} = (T^{(q)}, S_1^{(q)}, ..., S_p^{(q)})$
19. $\quad$ **If** $|\psi(\boldsymbol{P}^{(q+1)}) - \psi(\boldsymbol{P}^{(q)})| \leq \epsilon$
20. $\quad\quad$ **Stop**
21. $\quad$ **end**
22. **end**

On line 17, $\psi(\boldsymbol{P}^{(q+1)})$ is evaluated by (58) with the replacement of $E_{x w_j}$, $E_{w_i w_j}$, T with $E_{x w_j^{(q)}}$, $E_{w_i^{(q)} w_j^{(q)}}$, $T^{(q+1)}$, respectively, which have been computed in lines 2 and 3.

Sequence $\{\boldsymbol{P}^{(q+1)}\}$ in Algorithm 2 converges to a coordinate-wise minimum point of $\psi(\boldsymbol{P})$ which is a local minimum. Section X-D provides more associated details.

Similar to Algorithm 1, in a training stage, it is convenient to use the error representation given in Theorem 6 where $P^{(q+1)}$ should be understood as that determined by Algorithm 2.

B. Algorithm 2: Determination of $T^{(0)}, S_1^{(0)}, ..., S_p^{(0)}$

To start Algorithm 2, matrices $T^{(0)}, S_1^{(0)}, ..., S_p^{(0)}$ should be known. They can be determined by the procedure detailed in Section III-C for the case of the ideal channels. That is, for Algorithm 2, $T^{(0)}$ is determined as $T^{(q+1)} = [T_1^{(q+1)}, \ldots, T_p^{(q+1)}]$ where $T_j^{(q+1)}$, for $j=1,\ldots,p$, is given by (38). For $j=1,\ldots,p$, matrix $S_j^{(0)}$, for Algorithm 2, is determined as $S_j^{(q+1)}$ by (39).

C. Models of Sensors and Fusion Center by Algorithm 2

By (8)–(10), the sensor model B_j is defined by matrix $S_j = [S_{j,0}, S_{j,1}, S_{j,2}]$. Matrix S_j is evaluated as $S_j^{(q+1)}$

at line 14. Since $S_j^{(q+1)}$ can be written as $S_j^{(q+1)} = [S_{j,0}^{(q+1)}, S_{j,1}^{(q+1)}, S_{j,2}^{(q+1)}]$ where $S_{j,k}^{(q+1)}$ is a block of $S_j^{(q+1)}$, for $k = 0, 1, 2$, then $S_{j,0}, S_{j,1}, S_{j,2}$ are evaluated as $S_{j,0}^{(q+1)}, S_{j,1}^{(q+1)}, S_{j,2}^{(q+1)}$.

The fusion center $T = [T_1, \ldots, T_p]$ is evaluated as $T^{(q+1)} = [T_1^{(q+1)}, \ldots, T_p^{(q+1)}]$ where $T_j^{(q+1)}$, for $j = 1, \ldots, p$, is computed on the lines from 3 to 7.

VI. Numerical Results and Comparison

Here, we illustrate the performance of the proposed approach with numerical modeling of the second degree WSN with two sensors and nonideal channels.

Example 4: Suppose that source signal $\mathbf{x} \in L^2(\Omega, \mathbb{R}^4)$ is a Gaussian random vector with mean zero and covariance matrix

$$E_{xx} = \begin{bmatrix} 1.000 & 0.580 & 0.275 & 0.450 \\ 0.580 & 1.000 & 0.295 & 0.540 \\ 0.275 & 0.295 & 1.000 & 0.215 \\ 0.450 & 0.540 & 0.215 & 1.000 \end{bmatrix}.$$

Observed signals $\mathbf{y}_1$ and $\mathbf{y}_2$ are noisy versions of $\mathbf{x}$ such that $\mathbf{y}_j = \mathbf{x} + \delta_j^2 \xi_j$, $\delta_j \in \mathbb{R}$, for $j = 1, 2$, where ξ_1 and ξ_2 are Gaussian random vector with mean zero and covariance matrices $E_{\xi_i \xi_j} = \delta_j^2 I$, for $j = 1, 2$. The vectors $\mathbf{x}$, ξ_1 and ξ_2 are independent. Channel matrices are $D_1 = \begin{pmatrix} 6 & 6 \\ 2 & 8 \end{pmatrix}$ and $D_2 = \begin{pmatrix} 0 & 5 \\ 0 & 5 \end{pmatrix}$. Note D_2 is singular, i.e., method in [7] is not applicable here. Suppose the channel noise $\boldsymbol{\eta}_j$ is white, for $j = 1, 2$, i.e., $E_{\eta_j, \eta_j} = \gamma_j^2 I$.

The diagrams of the MSE, given in Fig. 5, for $r_1 = r_2 = 2$, $\delta_1 = 0.7$, $\delta_1 = 0.8$, $\gamma_1 = 0.6$, $\gamma_2 = 0.5$, demonstrate the advantage of the second degree model over its particular case, the linear model.

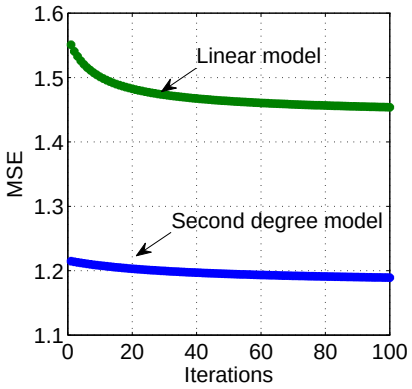


Fig. 5. Diagrams of the MSE for the WSNs with nonideal channels.

VII. Conclusion

We have addressed the problem of multi-compression and recovery of an unknown source stochastic signal when the signal cannot be observed centrally. In the mean square estimation framework, this means finding the optimal signal representation which minimizes the associated error. This scenario is based on multiple noisy signal observations made by distributed sensors. Each sensor compresses the observation and then transmits the compressed signal to the fusion center that decompresses the signals in such a way that the original signal is estimated within a prescribed accuracy. By known techniques, models of the sensors and the fusion center have been developed from optimal *linear* transforms and therefore, the associated errors cannot be improved by any other linear transform with the same rank. At the same time, in some problems, the performance of the known techniques may not be as good as required.

In this paper, we developed a new method for the determination of models of the sensors and the fusion center. The contribution and advantages are as follows. First, the proposed approach is based on the *non-linear* second degree transform (SDT) [1]. The special condition has been established under which the proposed methodology may lead to the higher accuracy of signal estimation compared to known methods based on linear signal transforms. Second, the SDT is combined with the maximum block improvement (MBI) method [2], [3]. Unlike the commonly used block coordinate descent (BCD) method, the MBI method converges to a local minimum under weaker conditions than those required by the BCD method (more details have been provided in Sections I-B and I-C). Third, the models of the sensors and the fusion center have been determined in terms of the pseudo-inverse matrices. Therefore, they are always well determined and numerically stable. As a result, this approach mitigates to some extent the difficulties associated with the existing techniques. Since a 'good' choice of the initial iteration gives reduced errors, special methods for determining initial iterations have been considered.

The error analysis of the proposed method has been provided.

VIII. Future Development

The problem of WSN modeling with nonideal channels subject to a power constraint per sensor was tackled with different approaches considered, in particular, in [7], [55], [56], [57], [58]. It appears that this is a specialized topic and we intend to extend the approach proposed in Section V to this important problem. Another important topic relates to the analysis of the influence of the covariance estimation errors on models of the sensors and the fusion center. As mentioned before, special methods were developed, in particular, in [38], [39], [40], [41], [42], [43], [44], [45]. In our future research, we intend to extend the known related results to the WSN modeling with the proposed methodology.

IX. Acknowledgment

The authors are grateful to anonymous reviewers for their useful suggestions that led to the significant improvement of the presented results.

X. Appendix

A. Advantages of Non-linear Approximation

Advantages of non-linear approximation have been considered, in particular, in [47], [48], [49], [50], [51], [52], [53] where the *non-linear* approximator is represented by the q-degree polynomial operator $\mathbb{P}_q : L^2 \to L^2$ defined as $\mathbb{P}_q(\mathbf{y}) = \mathcal{A}_0 + \mathcal{A}_1(\mathbf{y}) + \mathcal{A}_2(\mathbf{y},\mathbf{y}) + \ldots + \mathcal{A}_q(\underbrace{\mathbf{y},\ldots,\mathbf{y}}_{q})$. Here, $\mathcal{A}_0$ stands for a constant operator and $\mathcal{A}_k : (L^2)^k \to L^2$, for $k = 1,\ldots,q$, is a k-linear operator. In broad terms, it is proven that under certain conditions specified in [47], [48], [49], [50], [51], [52], [53], a non-linear transform $\Phi : L^2 \to L^2$, where $L^2 := L^2(\Omega,\mathbb{R}^n)$, defined as $\mathbf{x} = \Phi(\mathbf{y})$ can uniformly be approximated by $\mathbb{P}_q(\mathbf{y})$ to any degree of accuracy. In other words, for any$\epsilon > 0$, there is a q-degree polynomial $\mathbb{P}_q(\mathbf{y})$ such that, for an appropriate norm $\|\cdot\|$,

$$\|\mathbf{x} - \mathbb{P}_q(\mathbf{y})\| < \epsilon. \tag{59}$$

For $q > 1$, the q-degree polynomial $\mathbb{P}_q(\mathbf{y})$ has more parameters to optimize compared to the linear case when $\mathbb{P}_1(\mathbf{y}) = \mathcal{A}_1(\mathbf{y})$. That is why $\mathbb{P}_q(\mathbf{y})$ can provide any degree of accuracy. Indeed, optimization of $\mathbb{P}_q(\mathbf{y})$ is achieved by a variation of $q+1$ parameters, $\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_q$, while optimization of $\mathbb{P}_1(\mathbf{y}) = \mathcal{A}_1(\mathbf{y})$ follows from choosing one parameter only, $\mathcal{A}_1$.

An optimal determination of the q-degree polynomial $\mathbb{P}_q(\mathbf{y})$ is a hard research problem [54]. Its solution is subject to special conditions (convexity, for example) that are not satisfied for the problem in (6) if $\mathcal{F}_j(\mathbf{y}_j)$ is replaced by $\mathbb{P}_q(\mathbf{y})$. For a particular case in (6), when $p = 1$, $q = 1$ and $\mathcal{A}_0$ is the zero operator, this difficulty was surmounted in [7], [8], [9], [10], [11], [12], [13], [14] where a special approach for finding $\mathbb{P}_1(\mathbf{y})$ that solves

$$\min_{\mathbb{P}_1 \in \mathcal{R}_{r_1}} \|\mathbf{x} - \mathbb{P}_1(\mathbf{y})\|_\Omega^2 . \tag{60}$$

has been developed. In [1], for still $p = 1$ in (6), a more general case has been considered, when $q = 2$ and $\mathbb{P}_2(\mathbf{y})$ is represented in a special form given by $\mathbb{P}_2(\mathbf{y}) = \mathcal{A}_1(\mathbf{y}) + \mathcal{A}_2(\mathbf{y},\mathbf{y})$, where $\mathcal{A}_2(\mathbf{y},\mathbf{y}) = \widetilde{\mathcal{A}}_2(\mathbf{y}^2)$, $\widetilde{\mathcal{A}}_2$ is a linear operator and $\mathbf{y}^2$ is given by $\mathbf{y}^2(\omega) := ([\mathbf{y}_1(\omega)]^2,\ldots,[\mathbf{y}_n(\omega)]^2)^T$. The solution of the problem

$$\min_{\mathcal{A}_1,\widetilde{\mathcal{A}}_2} \|\mathbf{x} - \mathbb{P}_2(\mathbf{y})\|_\Omega^2$$
$$\text{s. t.} \quad \text{rank}\,[\mathcal{A}_1,\ \widetilde{\mathcal{A}}_2] \le r \le \min\{m, 2n\}$$

obtained in [1] shows that the increase in q to $q = 2$ allows us to achieve a better associated accuracy than that for the linear case, i.e. with $q = 1$, for the same compression ratio.

The result in [1] has been extended in [59], for an arbitrary q in $\mathbb{P}_q(\mathbf{y})$ and still for $p = 1$ in (6). Due to a special structure of the polynomial $\mathbb{P}_q(\mathbf{y})$ based on the orthogonalisation procedure proposed in [59], it has been shown in Theorem 2 in [59] that, for $p = 1$ and $\mathcal{F}_1$ represented by $\mathbb{P}_q(\mathbf{y})$ in (6), the accuracy of $\mathbf{x}$ estimation in (6) is further improved when q increases. The orthogonalisation procedure [59] cannot be used for the problem under consideration since the sensors should not communicate with each other. It is one of the reasons to develop the greedy approach represented in the above Sections I-C, III-A3.

B. Explanation for the Relation in (2)

In Fig. 1 (a), the input of the jth sensor is random vector $\mathbf{y}_j$ which is, by the definition, a function from Ω to $\mathbb{R}^{n_j}$. More specifically, to satisfy the boundedness condition, $\mathbf{y}_j \in L^2(\Omega,\mathbb{R}^{n_j})$. Since a multiplication of *function* $\mathbf{y}_j$ by a *matrix* is not defined, the sensor model is represented by linear operator $\mathcal{B}_j$. As a result, in the LHS of (2), $\mathcal{B}_j(\mathbf{y}_j)$ is a random vector as well (i.e., a function from Ω to $\mathbb{R}^{n_j}$) and then $[\mathcal{B}_j(\mathbf{y}_j)](\omega)$, for any $\omega \in \Omega$, is a vector in $\mathbb{R}^{n_j}$. To define operator $\mathcal{B}_j$, matrix B_j is used. To be consistent, operator $\mathcal{B}_j$ and matrix B_j relate to each other as in (2). In the RHS of (2), $\mathbf{y}_j(\omega)$ is a vector in $\mathbb{R}^{n_j}$, for every $\omega \in \Omega$, and therefore, multiplication of B_j by the vector $\mathbf{y}_j(\omega)$ is well defined.

C. Justification of Algorithm 1

1) Proof of Theorem 1: On the basis of (19),

$$\begin{aligned}\min_{\mathcal{P}_j \in \mathcal{R}_{r_j}} \left\| \mathbf{x} - \sum_{j=1}^p \mathcal{P}_j(\mathbf{z}_j) \right\|_\Omega^2 &= \min_{P_j \in \mathbb{R}_{r_j}} \left\| H - \sum_{j=1}^p P_j G_j \right\|^2 \\ &= \min_{P_j \in \mathbb{R}_{r_j}} \|Q_j - P_j G_j\|^2 ,\end{aligned}$$

where Q_j and G_j are as in (19), and $j = 1,\ldots,p$. The problem $\min_{P_j \in \mathbb{R}_{r_j}} \|Q_j - P_j G_j\|^2$ has been studied in [4], [5] where its solution has been obtained in form (24). In Theorem 1, the conditions associated with r_j, rank of matrix $Q_j R_{G_j}$ and the singular values of matrix $Q_j R_{G_j}$ follow from the conditions of uniqueness of the SVD (see, e.g., [4]).

2) Proof of Theorem 2 : Denote $\overline{\mathbf{x}} = \mathbf{x} - \sum_{\substack{i=1,\\ i\neq j}}^p \mathcal{P}_j \mathbf{z}_j$. Then we have

$$\begin{aligned} & f(P_1,\ldots P_{j-1}, \widehat{P}_j, P_{j+1},\ldots,P_p) \\ &= \|\mathbf{x} - \sum_{\substack{i=1,\\ i\neq j}}^p \mathcal{P}_j \mathbf{z}_j - \widehat{\mathcal{P}}_j \mathbf{z}_j\|_\Omega^2 \\ &= \|\overline{\mathbf{x}} - \widehat{\mathcal{P}}_j \mathbf{z}_j\|_\Omega^2 \end{aligned}$$

On the basis of Theorem 2 in[1], we obtain

$$\begin{aligned}\|\overline{\mathbf{x}} - \widehat{\mathcal{P}}_j \mathbf{z}_j\|_\Omega^2 &= \text{tr}\{E_{\overline{xx}}\} - \sum_{i=1}^{r_j} \delta_i - \text{tr}\{C_j H_j^\dagger C_j^T\} \\ &= \text{tr}\{E_{\overline{xx}}\} - \sum_{i=1}^{r_j} \delta_i - \sum_{i=1}^{m_j} \mu_{j,i}.\end{aligned}$$

Further,

$$\begin{aligned}\text{tr}\{E_{\overline{xx}}\} &= \text{tr}\{E_{x-w_j,x-w_j}\} \\ &= \text{tr}\{E_{xx}\} - 2\,\text{tr}\{E_{w_j x}\} + \text{tr}\{E_{w_j w_j}\}. \end{aligned} \tag{61}$$

Then (25) follows.

3) Comparison with the Linear WSN in (6): Recall that (25) represents the error associated with the second degree WSN represented by (11). For the linear WSN considered in Section II-B1, at each iteration of Algorithm 1, matrix $\widehat{P}_j$ should be replaced with matrix $\widehat{F}_j$ where $\widehat{F}_j$ is represented by the GKLT (considered, in particular, in [27]) or by its particular case given in [21]. We wish to compare the error in (25) with the error associated with the linear WSN. To this end, first, we establish the error representation for the linear WSN. Let us denote

$$f_L(F_1,...,F_p)=\left\|\mathbf{x}-\sum_{j=1}^{p}\mathcal{F}_j(\mathbf{y}_j)\right\|_{\Omega}^2,$$

$\mathcal{F}_j$ is as in (6), $\widetilde{\mathbf{w}}_j=\sum_{\substack{i=1\\ i\neq j}}^{p}\mathcal{F}_i\mathbf{y}_i$, $\alpha_j=\mathrm{tr}\{2\,E_{\widetilde{w}_j x}-E_{\widetilde{w}_j\widetilde{w}_j}\}$ and $\widetilde{\mathbf{x}}=\mathbf{x}-\widetilde{\mathbf{w}}_j$. Further, we write $\sigma_1,...,\sigma_{r_j}$ for the first r_j eigenvalues in the SVD for matrix $E_{\widetilde{x}y_j}E^{\dagger}_{y_jy_j}E_{y_j\widetilde{x}}$. In the theorem that follows we consider the case when $\widehat{F}_j$ is determined by the GKLT.

Theorem 6: For given $F_1,\ldots,F_{j-1},\ F_{j+1},\ \ldots,F_p$, the error associated with the optimal matrix $\widehat{F}_j$ for the linear WSN is given by

$$f_L(F_1,...,F_{j-1},\widehat{F}_j,F_{j+1},...,F_p)=\mathrm{tr}\{E_{xx}\}-\sum_{i=1}^{r_j}\sigma_i-\alpha_j. \tag{62}$$

Proof: On the basis of [27] (pp. 313),

$$\begin{aligned} f_L(F_1,...,F_{j-1},\widehat{F}_j,F_{j+1},...,F_p) &= \|\mathbf{x}-\sum_{\substack{i=1,\\ i\neq j}}^{p}\mathcal{F}_j\mathbf{y}_j-\widehat{\mathcal{F}}_j\mathbf{y}_j\|_{\Omega}^2\\ &= \|\widetilde{\mathbf{x}}-\widehat{\mathcal{F}}_j\mathbf{y}_j\|_{\Omega}^2\\ &= \mathrm{tr}\{E_{\widetilde{x}\widetilde{x}}\}-\sum_{i=1}^{r_j}\sigma_i \end{aligned}$$

where

$$\begin{aligned} \mathrm{tr}\{E_{\widetilde{x}\widetilde{x}}\} &= \mathrm{tr}\{E_{x-\widetilde{w}_j,x-\widetilde{w}_j}\}\\ &= \mathrm{tr}\{E_{xx}\}-2\,\mathrm{tr}\{E_{\widetilde{w}_j x}\}+\mathrm{tr}\{E_{\widetilde{w}_j\widetilde{w}_j}\}. \end{aligned} \tag{63}$$

The above implies (62). ■

The following theorem establishes a condition under which the error in (25) associated with the second degree WSN is less than that in (62) for the linear WSN.

Theorem 7: If

$$\alpha_j-\beta_j<\sum_{i=1}^{r_j}(\delta_i-\sigma_i)+\sum_{i=1}^{m}\mu_{j,i}$$

then

$$f(P_1,...,P_{j-1},\widehat{P}_j,P_{j+1},...,P_p)<f_L(F_1,...,F_{j-1},\widehat{F}_j,F_{j+1},...,F_p)$$

Proof: The proof follows directly from (25) and (62). ■

4) Comparison with Linear WSNs in [7], [11]: A theoretical comparison of the error associated with the second degree WSN in (25) and the errors associated with the linear WSNs studied in [7], [11] is difficult because of the following. The error analysis in [7], [11] is provided in terms which are quite different from those used in Theorem 2. Second, in [7], [11], the error analysis is given under the assumption that the covariance matrices are invertible, which is not the case in Theorem 2. At the same time, the errors associated with the methods in [7] and [11] might be represented in the form similar to that in (62) where σ_i and α_j should specifically be obtained for the methods in [7], [11]. In this case, the comparison would be provided in the form similar to that in Theorem 7. Other way for the numerical comparison is to use the error representation in (31). The error associated with the linear WSN in [7] or [11], follows from (31) where z should be replaced with y and $P^{(q+1)}$ should be replaced with the corresponding iteration of the method [7] or [11]. Of course, this is a posteriori comparison, nevertheless, it is an opportunity to obtain a numerical representation of the errors under consideration. In fact, this comparison is provided in the simulations in Section IV.

5) Convergence of Algorithm 1: Convergence of the method presented in Section III-A3 can be shown on the basis of the results given in [2], [3], [60] as follows. We call $\boldsymbol{P}=(P_1,...,P_p)\in\mathbb{R}_{r_1,...,r_p}$ a point in the space $\mathbb{R}_{r_1,...,r_p}$. For every point $\boldsymbol{P}\in\mathbb{R}_{r_1,...,r_p}$, define a set

$$\mathbb{R}_{r_j}^{\boldsymbol{P}}=\{(P_1,\ldots,P_{j-1})\}\times\mathbb{R}_{r_j}\times\{(P_{j+1},\ldots,P_p)\},$$

for $j=1,\ldots,p$. A coordinate-wise minimum point of the procedure represented by Algorithm 1 is denoted by $\boldsymbol{P}^*=\left(P_1^*,...,P_p^*\right)$ where[7]

$$P_j^*\in\left\{\arg\min_{P_j\in\mathbb{R}_{r_j}}\ \phi(P_1^*,...,P_{j-1}^*,P_j,P_{j+1}^*,...,P_p^*)\right\}. \tag{64}$$

This point is a local minimum of objective function in (18), $\phi(\boldsymbol{P})=\left\|H-\sum_{j=1}^{p}P_jG_j\right\|^2$.[8] Note that $\boldsymbol{P}^{(q+1)}$ in Algorithm 1 and $\boldsymbol{P}^*$ defined by (64) are, of course, different.

We also need the following auxiliary result.

Lemma 1: The sequence $\{\boldsymbol{P}^{(q)}\}$ generated by Algorithm 1 is bounded.

Proof: Consider $\widetilde{\phi}(\boldsymbol{P})=\widetilde{\phi}(P_1,\ldots,P_p)=\|PG-H\|$ where $G=[G_1,\ldots,G_p]$, and G_j, for $j=1,\ldots,p$, and H are given in (14) and (15). Then $\|PG-H\|\geq\|PG\|-\|H\|$, i.e.,

$$\|PG\|-\|H\|\leq\widetilde{\phi}(P_1,\ldots,P_p). \tag{65}$$

Suppose sequence $\{\boldsymbol{P}^{(q)}\}$ is unbounded, i.e., $\|P^{(q)}\|\to\infty$ as $q\to\infty$. Then (65) implies $\left\|P^{(q)}G\right\|-\|H\|\to\infty$ as

[7]The RHS in (64) is a set since the solution of problem $\min_{P_j\in\mathbb{R}_{r_j}}\ \phi(P_1^*,...,P_{j-1}^*,P_j,P_{j+1}^*,...,P_p^*)$ is not unique.

[8]There could be other local minimums defined differently from that in (64).

$q \to \infty$. Therefore, $\widetilde{\phi}(P_1^{(q)}, \ldots, P_p^{(q)}) \to \infty$ as $q \to \infty$ and then $\phi(P_1^{(q)}, \ldots, P_p^{(q)}) \to \infty$ as $q \to \infty$. But it conflicts with Algorithm 1 where

$$0 \leq \phi(P_1^{(q)}, \ldots, P_p^{(q)}) \leq \phi(P_1^{(0)}, \ldots, P_p^{(0)}).$$

Thus, $\{\boldsymbol{P}^{(q)}\}$ is bounded. ■

Now we are in the position to show convergence of Algorithm 1. For $\boldsymbol{P}^{(q)}$ defined by Algorithm 1, denote

$$\phi(\check{\boldsymbol{P}}) = \lim_{q\to\infty} \phi(\boldsymbol{P}^{(q)}). \tag{66}$$

Theorem 8: Point $\check{\boldsymbol{P}}$ defined by (66) is the coordinate-wise minimum of Algorithm 1.

Proof: For each fixed $\boldsymbol{P} = (P_1, ..., P_p)$, a so-called best response matrix to matrix P_j is denoted by $\Phi_j^{\boldsymbol{P}}$, where

$$\Phi_j^{\boldsymbol{P}} \in \left\{ \arg \min_{P_j \in \mathbb{R}_{r_j}} \phi(P_1, ...P_{j-1}, P_j, P_{j+1}, ..., P_p) \right\}.$$

Let $\{\boldsymbol{P}^{(q)}\}$ be a sequence generated by Algorithm 1, where $\boldsymbol{P}^{(q)} = (P_1^{(q)}, ..., P_p^{(q)})$. Since each $\mathbb{R}_{r_j}^{\boldsymbol{P}}$ is closed [61, p. 304] and sequence $\{\boldsymbol{P}^{(q)}\}$ is bounded, there is a subsequence $\{\boldsymbol{P}^{(q_s)}\}$ such that $(P_1^{(q_s)}, ..., P_p^{(q_s)}) \to (P_1^*, ..., P_p^*) = \boldsymbol{P}^*$ as $s \to \infty$. Then, for any $j = 1, ..., p$, we have

$$\begin{aligned}
&\phi(P_1^{(q_s)}, ..., P_{j-1}^{(q_s)}, \Phi_j^{\boldsymbol{P}^*}, P_{j+1}^{(q_s)}, ..., P_p^{(q_s)}) \\
&\geq \phi(P_1^{(q_s)}, ..., P_{j-1}^{(q_s)}, \Phi_j^{\boldsymbol{P}^{(q_s)}}, P_{j+1}^{(q_s)}, ..., P_p^{(q_s)}) \\
&\geq \phi(P_1^{(q_s+1)}, ..., P_{j-1}^{(q_s+1)}, P_j^{(q_s+1)}, P_{j+1}^{(q_s+1)}, ..., P_p^{(q_s+1)}) \\
&\geq \phi(P_1^{(q_{s+1})}, ..., P_{j-1}^{(q_{s+1})}, P_j^{(q_{s+1})}, P_{j+1}^{(q_{s+1})}, ..., P_p^{(q_{s+1})})
\end{aligned}$$

By continuity, when $s \to \infty$,

$$\begin{aligned}
&\phi(P_1^*, ..., P_{j-1}^*, \Phi_j^{\boldsymbol{P}^*}, P_{j+1}^*, ..., P_p^*) \geq \\
&\qquad \phi(P_1^*, ..., P_{j-1}^*, P_j^*, P_{j+1}^*, ..., P_p^*),
\end{aligned}$$

which implies that above should hold as an equality, since the inequality is true by the definition of the best response matrix $\Phi_j^{\boldsymbol{P}^*}$. Thus, P_j^* is such as in (64), i.e. P_j^* is a solution of the problem

$$\min_{P_j \in \mathcal{R}_{r_j}} \phi(P_1^*, ..., P_{j-1}^*, P_j, P_{j+1}^*, ..., P_p^*), \ \forall j = 1, ..., p.$$

■

D. Justification of Algorithm 2

For $\boldsymbol{P}^{(q)}$ defined by Algorithm 2, denote

$$\psi(\boldsymbol{P}^*) = \lim_{q\to\infty} \psi(\boldsymbol{P}^{(q)}). \tag{67}$$

Theorem 9: Point $\boldsymbol{P}^* = (T^*, S_1^*, ..., S_p^*)$ defined by (67) is the coordinate-wise minimum of Algorithm 2.

Proof: The proof is similar to that for Theorem 8. Therefore, we only provide a related sketch. For each fixed $\mathbf{P} = (T, S_1, ..., S_p)$, denote by $\Psi_j^{\mathbf{P}}$ a best response matrix to S_j defined by

$$R_j^{\mathbf{P}} \in \left\{ \arg \min_{S_j \in \mathbb{R}^{r_j \times n}} \psi(T, S_1, ...S_{j-1}, S_j, S_{j+1}, ..., S_p) \right\}.$$

By Algorithm 2, a member of sequence $\{\boldsymbol{P}^{(q)}\}$ is given by $\mathbf{P}^{(q)} = (T^{(q)}, S_1^{(q)}, ..., S_p^{(q)})$. Similar to Lemma 1, sequence $\{\boldsymbol{P}^{(q)}\}$ provided by Algorithm 2 is bounded. Then there is a subsequence $\{\mathbf{P}^{(q_v)}\}$ with $\mathbf{P}^{(q_v)} = (T^{(q_v)}, S_1^{(q_v)}, ..., S_p^{(q_v)})$ such that $(T^{(q_v)}, S_1^{(q_v)}, ..., S_p^{(q_v)}) \to (T^*, S_1^*, ..., S_p^*) = \mathbf{P}^*$ as $v \to \infty$. Then, for any $j = 1, ..., p$, we have

$$\begin{aligned}
&\psi(T^{(q_v)}, S_1^{(q_v)}, ..., S_{j-1}^{(q_v)}, R_j^{\mathbf{P}^*}, S_{j+1}^{(q_v)}, ..., S_p^{(q_v)}) \geq \\
&\psi(T^{(q_{t+1})}, S_1^{(q_{t+1})}, ..., S_{j-1}^{(q_{t+1})}, S_j^{(q_{t+1})}, S_{j+1}^{(q_{t+1})}, ..., S_p^{(q_{t+1})})
\end{aligned}$$

and, for $v \to \infty$,

$$\begin{aligned}
&\psi(T^*, S_1^*, ..., S_{j-1}^*, R_j^{\mathbf{P}^*}, S_{j+1}^*, ..., S_p^*) \geq \\
&\qquad \psi(T^*, S_1^*, ..., S_{j-1}^*, S_j^*, S_{j+1}^*, ..., S_p^*),
\end{aligned}$$

which implies that above should hold as an equality. Thus, S_j^* is the best response to $(S_1^*, ...S_{j-1}^*, S_{j+1}^*, ..., S_p^*)$, or equivalently, S_j^* is the optimal solution for the problem

$$\max_{S_j \in \mathbb{R}^{r_j \times n}} \psi(T^*, S_1^*, ..., S_{j-1}^*, S_j, S_{j+1}^*, ..., S_p^*), \ \forall j = 1, ..., p.$$

The proof is similar when

$$R_j^{\mathbf{P}} \in \left\{ \arg \min_{T \in \mathbb{R}^{m \times r_j}} \psi(T, S_1, ..., S_p) \right\}.$$

■

REFERENCES

[1] A. Torokhti and P. Howlett, "Optimal fixed rank transform of the second degree," *IEEE Trans. CAS. Part II, Analog and Digital Signal Processing*, vol. 48, no. 3, pp. 309 – 315, 2001.

[2] B. Chen, S. He, Z. Li, and S. Zhang, "Maximum block improvement and polynomial optimization," *SIAM Journal on Optimization*, vol. 22, no. 1, pp. 87–107, 2012.

[3] Z. Li, A. Uschmajew, and S. Zhang, "On convergence of the maximum block improvement method," *SIAM Journal on Optimization*, vol. 25, no. 1, pp. 210–233, 2015.

[4] S. Friedland and A. Torokhti, "Generalized rank-constrained matrix approximations," *SIAM Journal on Matrix Analysis and Applications*, vol. 29, no. 2, pp. 656–659, 2007.

[5] A. Torokhti and S. Friedland, "Towards theory of generic Principal Component Analysis," *Journal of Multivariate Analysis*, vol. 100, no. 4, pp. 661 – 669, 2009.

[6] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Commun. Mag.*, vol. 40, no. 8, pp. 102 – 114, 2002.

[7] I. D. Schizas, G. B. Giannakis, and Z.-Q. Luo, "Distributed estimation using reduced-dimensionality sensor observations," *IEEE Transactions on Signal Processing*, vol. 55, no. 8, pp. 4284–4299, Aug 2007.

[8] E. Song, Y. Zhu, and J. Zhou, "Sensors optimal dimensionality compression matrix in estimation fusion," *Automatica*, vol. 41, no. 12, pp. 2131 – 2139, 2005.

[9] Y. Zhu, E. Song, J. Zhou, and Z. You, "Optimal dimensionality reduction of sensor data in multisensor estimation fusion," *IEEE Transactions on Signal Processing*, vol. 53, no. 5, pp. 1631–1639, May 2005.

[10] G. M., D. P.L., and V. M., "The distributed Karhunen-Loève transform," *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5177–5196, Dec 2006.

[11] O. Roy and M. Vetterli, "Dimensionality reduction for distributed estimation in the infinite dimensional regime," *IEEE Transactions on Information Theory*, vol. 54, no. 4, pp. 1655–1669, April 2008.

[12] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Dimensionality reduction, compression and quantization for distributed estimation with wireless sensor networks," in *Wireless Communications*, ser. The IMA Volumes in Mathematics and its Applications. Springer New York, 2007, vol. 143, pp. 259–296.

[13] P. L. Dragotti and M. Gastpar, *Distributed Source Coding: Theory, Algorithms and Applications*. Academic Press, 2009.

[14] J. Fang and H. Li, "Optimal/near-optimal dimensionality reduction for distributed estimation in homogeneous and certain inhomogeneous scenarios," *IEEE Transactions on Signal Processing*, vol. 58, no. 8, pp. 4339–4353, Aug 2010.

[15] A. Amar, A. Leshem, and M. Gastpar, "Recursive implementation of the distributed Karhunen-Loeve transform," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5320–5330, Oct 2010.

[16] J. A. Saghri, S. Schroeder, and A. G. Tescher, "Adaptive two-stage Karhunen-Loeve-transform scheme for spectral decorrelation in hyperspectral bandwidth compression," *Optical Engineering*, vol. 49, no. 5, pp. 057 001–057 001–7, 2010.

[17] M. Lara and B. Mulgrew, "Performance of the distributed KLT and its approximate implementation," in *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, Aug 2012, pp. 724–728.

[18] D. Li, K. D. Wong, Y. H. Hu, and A. M. Sayeed, "Detection, classification, and tracking of targets," *IEEE Signal Processing Mag.*, vol. 19, no. 2, pp. 17 – 29, 2002.

[19] T.-H. L. W. J. Kaiser and G. J. Pottie, "Integrated low-power communication system design for wireless sensor networks," *IEEE Commun. Mag.*, vol. 42, no. 12, pp. 142 – 150, 2004.

[20] M. G. M. Vetterli and P. Dragotti, "Sensing reality and communicating bits: A dangerous liaison," *IEEE Signal Process. Mag.*, vol. 23, no. 4, pp. 70 – 83, 2006.

[21] D. R. Brillinger, *Time Series: Data Analysis and Theory*. San Francisco: Holden Day, 2001.

[22] T. Kailath, A. Saed, and B. Hassibi, *Linear estimation*. Upper Saddle River: Prentice-Hall, 2000.

[23] H. V. Poor, *An Introduction to Signal Processing and Estimation*. 2nd ed. New York: Springer-Verlag, 2001.

[24] P. Tseng, "Convergence of a block coordinate descent method for nondifferentiable minimization," *Journal of Optimization Theory and Applications*, vol. 109, no. 3, pp. 475–494, 2001.

[25] Y. Hua, M. Nikpour, and P. Stoica, "Optimal reduced-rank estimation and filtering," *IEEE Transactions on Signal Processing*, vol. 49, no. 3, pp. 457–469, Mar 2001.

[26] A. Torokhti and S. Miklavcic, "Data compression under constraints of causality and variable finite memory," *Signal Processing*, vol. 90, no. 10, pp. 2822 – 2834, 2010.

[27] A. Torokhti and P. Howlett, *Computational Methods for Modelling of Nonlinear Systems*. Elsevier, 2007.

[28] D. Slepian and J. Wolf, "Noiseless coding of correlated information sources," *IEEE Transactions on Information Theory*, vol. 19, no. 4, pp. 471–480, Jul 1973.

[29] A. Wyner and J. Ziv, "The rate-distortion function for source coding with side information at the decoder," *IEEE Transactions on Information Theory*, vol. 22, no. 1, pp. 1–10, Jan 1976.

[30] V. Goyal, "Theoretical foundations of transform coding," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 9–21, Sep 2001.

[31] A. Bertrand and M. Moonen, "Distributed adaptive node-specific signal estimation in fully connected sensor networks - Part I: Sequential node updating," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5277 – 5291, 2010.

[32] ——, "Distributed adaptive node-specific signal estimation in fully connected sensor networks—part ii: Simultaneous and asynchronous node updating," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5292 – 5306, 2010.

[33] V. Ejov and A. Torokhti, "How to transform matrices U_1,...,U_p to matrices V_1,...,V_p so that $V_iV_j = \mathbb{O}$ if $i \neq j$?" *Numerical Algebra, Control and Optimization*, vol. 2, no. 2, pp. 293–299, 2012.

[34] H. Ma, Y.-H. Yang, Y. Chen, L. K.J.R., and Q. Wang, "Distributed state estimation with dimension reduction preprocessing," *Signal Processing, IEEE Transactions on*, vol. 62, no. 12, pp. 3098–3110, June 2014.

[35] A. Grant, A. Torokhti, and P. Soto-Quiros, "Compression and recovery of distributed random signals," *IEEE Transactions on Information Theory*, (submitted).

[36] A. Torokhti and S. Friedland, "Towards theory of generic principal component analysis," *Journal of Multivariate Analysis*, vol. 100, no. 4, pp. 661 – 669, 2009.

[37] A. Torokhti, P. Howlett, and H. Laga, "Estimation of stochastic signals under partially missing information," *Signal Processing*, vol. 11, pp. 199–209, 2015.

[38] L. Perlovsky and T. Marzetta, "Estimating a covariance matrix from incomplete realizations of a random vector," *IEEE Transactions on Signal Processing*, vol. 40, no. 8, pp. 2097–2100, Aug 1992.

[39] O. Ledoit and M. Wolf, "A well-conditioned estimator for large-dimensional covariance matrices," *Journal of Multivariate Analysis*, vol. 88, no. 2, pp. 365 – 411, 2004.

[40] ——, "Nonlinear shrinkage estimation of large-dimensional covariance matrices," *The Annals of Statistics*, vol. 40, no. 2, pp. 1024–1060, 2012.

[41] R. Adamczak, A. E. Litvak, A. Pajor, and N. Tomczak-Jaegermann, "Quantitative estimates of the convergence of the empirical covariance matrix in log-concave ensembles," *Journal of the American Mathematical Society*, no. 23, pp. 535–561, 2009.

[42] R. Vershynin, "How close is the sample covariance matrix to the actual covariance matrix?" *Journal of Theoretical Probability*, vol. 25, no. 3, pp. 655–686, 2012.

[43] S.-J. K. Joong-Ho Won, Johan Lim and B. Rajaratnam, "Condition-number-regularized covariance estimation," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 75, no. 3, pp. 427–450, 2013.

[44] B. W. Schmeiser and M. H. Chen, "On hit-and-run Monte Carlo sampling for evaluating multidimensional integrals," *Technical Report 91-39, Dept. Statistics, Purdue Univ.*, 1991.

[45] R. Yang and J. O. Berger, "Estimation of a covariance matrix using the reference prior," *The Anna1s of Statistics*, vol. 22, no. 3, pp. 1195–1211, 1994.

[46] E. Kreyszig, *Introductory Functional Analysis with Applications*. New York: John Willey & Sons, Inc, 1978.

[47] P. M. Prenter, "A Weierstrass theorem for real, separable Hilbert spaces," *J. Approximation Theory*, vol. 4, pp. 341 – 357, 1970.

[48] V. I. Istrăţescu, "A Weierstrass theorem for real Banach spaces," *J. Approximation Theory*, vol. 19, pp. 118–122, 1977.

[49] V. I. Bruno, "An approximate Weierstrass theorem in topological vector space," *J. Approximation Theory*, vol. 42, pp. 1–3, 1984.

[50] N. I. Akhiezer, *Theory of Approximation*. New York: Dover, 1992.

[51] I. Sandberg, "Time-delay polynomial networks and quality of approximation," *IEEE Trans. on Circuits and Systems. Part 1, Fundamental theory and applications*, vol. 47, pp. 40 – 49, 2000.

[52] V. J. Mathews and G. L. Sicuranza, *Polynomial Signal Processing*. New York: John Willey & Sons, Inc, 2001.

[53] P. G. Howlett, A. P. Torokhti, and C. Pearce, "A philosophy for the modelling of realistic non-linear systems," *Proc. of Amer. Math. Soc.*, vol. 131, no. 2, pp. 353–363, 2003.

[54] F. Deutsch, *Best Approximation in Inner Product Spaces*. New York: Springer, 2001.

[55] A. Ribeiro and G. B. Giannakis, "Bandwidth-constrained distributed estimation for wireless sensor networks - Part I: Gaussian case," *IEEE Transactions on Signal Processing*, vol. 54, no. 3, pp. 1131–1143, 2006.

[56] J.-J. Xiao, A. Ribeiro, Z.-Q. Luo, and G. B. Giannakis, "Distributed compression-estimation using wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 27, 2006.

[57] J. Li and G. AlRegib, "Distributed estimation in energy-constrained wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 57, no. 10, pp. 3746–3758, 2009.

[58] S. Shah and B. Beferull-Lozano, "In-network iterative distributed estimation for power-constrained wireless sensor networks," *8th IEEE International Conference on Distributed Computing in Sensor Systems*, pp. 239–246, 2012.

[59] A. Torokhti and P. Howlett, "Optimal transform formed by a combination of nonlinear operators: The case of data dimensionality reduction," *IEEE Trans. on Signal Processing*, vol. 54, no. 4, pp. 1431 – 1444, 2006.

[60] X. Li, H. Liu, and X. Zheng, "Non-monotone projection gradient method for non-negative matrix factorization," *Computational Optimization and Applications*, vol. 51, no. 3, pp. 1163–1171, 2012.

[61] L. Tu, *An Introduction to Manifolds*, ser. Universitext. Springer, 2007.

Anexo 9

Extended Principal Component Analysis

Pablo Soto-Quiros[a,b], Anatoli Torokhti[a,*]

[a] *School of Information Technology and Mathematical Sciences, University of South Australia, SA 5095, Australia*
[b] *Instituto Tecnologico de Costa Rica, Apdo. 159-7050, Cartago, Costa Rica*

Abstract

Principal Component Analysis (PCA) is a transform for finding the principal components (PCs) that represent features of random data. PCA also provides a reconstruction of the PCs to the original data. We consider an extension of PCA which allows us to improve the associated accuracy and diminish the numerical load, in comparison with known techniques. This is achieved due to the special structure of the proposed transform which contains two matrices T_0 and T_1, and a special transformation $\mathcal{F}$ of the so called auxiliary random vector $\mathbf{w}$. For this reason, we call it the three-term PCA. In particular, we show that the three-term PCA always exists, i.e. is applicable to the case of singular data. Both rigorous theoretical justification of the three-term PCA and simulations with real-world data are provided.

Keywords: Principal component analysis, Least squares linear estimate, Matrix computation, Singular value decomposition.

1. Introduction

In this paper, an extension of Principal Component Analysis (PCA) and its rigorous justification are considered. In comparison with known techniques, the proposed extension of PCA allows us to improve the associated accuracy and diminish the numerical load. The innovation of the proposed methodology,

*Corresponding author
Email addresses: juan.soto-quiros@mymail.unisa.edu.au (Pablo Soto-Quiros), anatoli.torokhti@unisa.edu.au (Anatoli Torokhti)

differences from the known results and advantages are specified in Sections 3, 6, 9 and 11.

PCA is a technique for finding so called principal components (PCs) of the data of interest represented by a large random vector, i.e. components of a smaller vector which preserve the principal features of the data. PCA also provides a reconstruction of PCs to the original data vector with the least possible error among all linear transforms. According to Jolliffe [15], "Principal component analysis is probably the oldest and best known of the techniques of multivariate analysis". This is a topic of intensive research which has an enormous number of related references. For instance, a Google search for 'principal component analysis' returns about $8,160,000$ results. In particular, the references which are most related to this paper (from our point view) are represented in [3, 13, 14, 18, 17, 23, 20]. PCA is used in a number of application areas (in an addition to the previous references see, for example, [4, 5, 19, 7]). Therefore, related techniques with a better performance are of vital importance.

By PCA in [15, 3], under the strong restriction of invertibility of the covariance matrix, the procedures for finding the PCs and their reconstruction are determined by the matrix of the rank less or equal to k where k is the number of required PCs. For a fixed number of PCs, the PCA accuracy of their reconstruction to the original data cannot be improved. In other words, PCA has the *only degree of freedom* to control the accuracy, it is the number of PCs. Moreover, PCA in the form obtained, in particular, in [15, 3], is not applicable if the associated covariance matrix is singular. Therefore, in [24], the generalizations of PCA, called generalized Brillinger transforms (GBT1 and GBT2), have been developed. First, the GBT1 and GBT2 are applicable to the case of singular covariance matrices. Second, the GBT2 allows us to improve the errors associated with PCA and the GBT1. We call it the generic Karhunen-Loève transform (GKLT). The GKLT requires an additional knowledge of covariance matrices E_{xy^2} and $E_{y^2y^2}$ where $\mathbf{x}$ and $\mathbf{y}$ are stochastic vectors, and $\mathbf{y}^2$ is a Hadamard square of $\mathbf{y}$ (more details are provided in Section 2 below). Such knowledge may be difficult. Another difficulty associated with the GBT2 and GKLT is their

numerical load which is larger than that of PCA. Further, it follows from [25] that the GKLT accuracy is better than that of PCA in [15, 3, 13] subject to the condition which is quite difficult to verify (see Corollary 3 in [25]). Moreover, for the GBT2 in [24], such an analysis has not been provided.

We are motivated by the development of an extension of PCA which covers the above drawbacks. We provide both a detailed theoretical analysis of the proposed extension of PCA and numerical examples that illustrate the theoretical results.

2. Review of PCA and its known generalizations GB1, GB2, GKLT

First, we introduce some notation which is used below.

Let $\mathbf{x} = [\mathbf{x}_1, \ldots, \mathbf{x}_m]^T \in L^2(\Omega, \mathbb{R}^m)$ and $\mathbf{y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n]^T \in L^2(\Omega, \mathbb{R}^n)$ be random vectors[1]. Vectors $\mathbf{x}$ and $\mathbf{y}$ are interpreted as reference data and observable data, respectively, i.e. $\mathbf{y}$ is a noisy version of $\mathbf{x}$. Dimensions m and n are assumed to be large. Suppose we wish to denoise $\mathbf{y}$ and reduce it to a 'shorter' vector $\mathbf{u} \in L^2(\Omega, \mathbb{R}^k)$ where $k \leq \min\{m, n\}$, and then reconstruct $\mathbf{u}$ to vector $\widetilde{\mathbf{x}}$ such that $\widetilde{\mathbf{x}}$ is as close to $\mathbf{x}$ as possible. Entries of vector $\mathbf{u}$ are called the principal components (abbreviated above as PCs).

Let us write $\|\mathbf{x}\|_\Omega^2 = \int_\Omega \|\mathbf{x}(\omega)\|_2^2 d\mu(\omega) < \infty$, where $\|\mathbf{x}(\omega)\|_2$ is the Euclidean norm of $\mathbf{x}(\omega) \in \mathbb{R}^m$. Throughout the paper, we assume that means $E[\mathbf{x}]$ and $E[\mathbf{y}]$ are known. Therefore, without loss of generality, we will assume henceforth that $\mathbf{x}$ and $\mathbf{y}$ have zero means. Then the covariance matrix formed from $\mathbf{x}$ and $\mathbf{y}$ is given by $E_{xy} = E[\mathbf{x}\mathbf{y}^T] = \{e_{ij}\}_{i,j=1}^{m,n} \in \mathbb{R}^{m\times n}$ where $e_{ij} = \int_\Omega \mathbf{x}_i(\omega)\mathbf{y}_j(\omega)d\mu(\omega)$.

Further, the singular value decomposition (SVD) of matrix $M \in \mathbb{R}^{m\times n}$ is given by $M = U_M \Sigma_M V_M^T$ where $U_M = [u_1 \ u_2 \ \ldots u_m] \in \mathbb{R}^{m\times m}, V_M = [v_1 \ v_2 \ \ldots v_n] \in \mathbb{R}^{n\times n}$ are unitary matrices, and $\Sigma_M = \mathrm{diag}(\sigma_1(M), \ldots, \sigma_{\min(m,n)}(M)) \in \mathbb{R}^{m\times n}$ is a generalized diagonal matrix, with the singular val-

[1] Here, $\Omega = \{\omega\}$ is the set of outcomes, Σ a σ-field of measurable subsets of Ω, $\mu : \Sigma \to [0, 1]$ an associated probability measure on Σ with $\mu(\Omega) = 1$ and (Ω, Σ, μ) for a probability space.

ues $\sigma_1(M) \geq \sigma_2(M) \geq \ldots \geq 0$ on the main diagonal. Further, $M^\dagger$ denotes the Moore-Penrose pseudo-inverse matrix for matrix M.

The generalizations of PCA mentioned in Section 1, GBT1 and GBT2 [24], are represented as follows. Let us first consider the GBT2 since the GBT1 is a particular case of the GBT2. The GBT2 is given by

$$B_2(\mathbf{y}) = R_1[P_1\mathbf{y} + P_2\mathbf{v}], \tag{1}$$

where $\mathbf{v} \in L^2(\Omega, \mathbb{R}^n)$ is an 'auxiliary' random vector used to further optimize the transform, **(Assoc. Edit.: More explanations!!!)** and matrices $R_1 \in \mathbb{R}^{m\times k}$, $P_1 \in \mathbb{R}^{k\times n}$ and $P_2 \in \mathbb{R}^{k\times n}$ solve the problem[2]

$$\min_{R,\ [P_1 P_2]} \|\mathbf{x} - R_1[P_1\mathbf{y} + P_2\mathbf{v}]\|_\Omega^2 \tag{2}$$

so that, for $\mathbf{q} = [\mathbf{y}^T\mathbf{v}^T]^T \in L^2(\Omega, \mathbb{R}^{2n})$ and $G_q = E_{xq}E_{qq}^\dagger E_{qx}$,

$$R_1 = U_{G_q,k}, \quad [P_1, P_2] = U_{G_q,k}^T E_{xq} E_{qq}^\dagger, \tag{3}$$

where $U_{G_q,k}$ is formed by the first k columns of U_{G_k}, and P_1 and P_2 are represented by the corresponding blocks of matrix $U_{G_q,k}^T E_{xq} E_{qq}^\dagger$. The principal components are then given by $\mathbf{u} = [P_1, P_2][\mathbf{y}^T\mathbf{v}^T]^T$.

The GBT1 follows from the GBT2 if $R_1P_2\mathbf{v} = \mathbf{0}$, where $\mathbf{0}$ is the zero vector. That is, the GBT2 has one matrix more (i.e., one degree of freedom more) than the GBT1. This allows us to improve the GBT2 performance compared to that by the GBT1 (see [24] for more detail).

[2]Note that in (1) and (2), strictly speaking, R_1 P_1 and P_2 should be replaced with operators $\mathcal{R}_1 : L^2(\Omega, \mathbb{R}^k) \to L^2(\Omega, \mathbb{R}^m)$, $\mathcal{P}_1 : L^2(\Omega, \mathbb{R}^n) \to L^2(\Omega, \mathbb{R}^k)$ and $\mathcal{P}_2 : L^2(\Omega, \mathbb{R}^n) \to L^2(\Omega, \mathbb{R}^k)$, respectively. This is because each matrix, say, $R_1 \in \mathbb{R}^{m\times k}$ defines a bounded linear transformation $\mathcal{R}_1 : L^2(\Omega, \mathbb{R}^k) \to L^2(\Omega, \mathbb{R}^m)$. Nevertheless, it is customary to write R_1 rather then $\mathcal{R}_1$, since $[\mathcal{R}_1(\mathbf{u})](\omega) = R_1[\mathbf{u}(\omega)]$, for each $\omega \in \Omega$. We keep this type of notation throughout the paper.

The GKLT [25] is given by

$$\mathcal{K}(\mathbf{y}) = K_1\mathbf{y} + K_2\mathbf{y}^2, \tag{4}$$

where matrices $K_1 \in \mathbb{R}^{m\times n}$ and $K_2 \in \mathbb{R}^{m\times n}$ solve the problem

$$\min_{\substack{[K_1 K_2] \\ \text{rank } [K_1 K_2]\leq k}} \|\mathbf{x} - [K_1\mathbf{y} + K_2\mathbf{y}^2]\|_\Omega^2, \tag{5}$$

and $\mathbf{y}^2$ is given by the Hadamard square so that $\mathbf{y}^2(\omega) = [\mathbf{y}_1^2(\omega), \ldots, \mathbf{y}_n^2(\omega)]^T$, for all $\omega \in \Omega$.

PCA is a particular case of the GKLT if $K_2\mathbf{y}^2 = \mathbf{0}$ and matrix E_{yy} is non-singular. The PCA, BT, GBT1, GKLT and GBT2 follow from the solution of essentially the same optimization problem. The differences are that first, the associated solutions are obtained under different assumptions and second, the solutions result in transforms that have different computational schemes. More details can be found in [24]. Further, each of PCA and the GBT1 has $m \times n$ parameters to optimize, which are entries of matrices K_1 and R_1P_1, respectively. Similar to PCA, the GBT1 has one degree of freedom to improve the associated accuracy, it is the number k of PCs, i.e., the dimension of vector $\mathbf{u}$. Thus, for fixed k, the GBT1 accuracy cannot be improved. The GKLT [25] and GBT2 [24] each have two degrees of freedom, k and one matrix more than in PCA and GBT1. That is, the GKLT and GBT2 have twice as many parameters to optimize compared to PCA and GBT1. It is shown in [24, 25] that this feature allows us to improve the accuracy associated with the GKLT and GBT2.

3. Contribution and novelty

We propose and justify the PCA extension which

• always exists, i.e. is applicable to the case of singular data (this is because it is constructed in terms of pseudo-inverse matrices; see Section 7),

• has better associated accuracy than that of the GBT1, GBT2 and GKLT

(Sections 9.1, 9.3, 12),

• has more degrees of freedom to improve the associated accuracy than the PCA, GBT1, GBT2 and GKLT (Sections 9.2 and 10),

• has a lower computational load than that of the GBT2 and GKLT; in fact, for large m, n, it is about 37% of that of the GBT2 and 22% of that of the GKLT (Section 11.2),

• does not require the usage of matrices E_{xy^2} and $E_{y^2y^2}$ (as required in [25]) which are difficult to determine.

Further, we show, in particular, that

• the condition for the GKLT [25] mentioned in Section 2 (under which the accuracy improvement is achieved) can be omitted (Section 9.1).

In more detail, in the proposed PCA extension, the additional degrees of freedom are provided by the auxiliary random vectors $\mathbf{w}$ and $\mathbf{h}$ which are introduced below in Sections 4 and 9.2, respectively. Vectors $\mathbf{w}$ and $\mathbf{h}$ are called $\mathbf{w}$-injection and $\mathbf{h}$-injection. An improvement in the accuracy of the proposed transform follows from the increase in the number of parameters to optimize, which are represented by matrices T_0, T_1, specific vector transformation $\mathcal{F}$ (Sections 4 and 7.2), and $\mathbf{w}$-injection and $\mathbf{h}$-injection (Sections 5, 9 and 10).

4. Structure of the proposed PCA extension

The above advantages are achieved due to the special structure of the proposed transform as follows. Let $\mathbf{w} \in L^2(\Omega, \mathbb{R}^\ell)$ be a random vector and $\mathcal{F} : L^2(\Omega, \mathbb{R}^n) \times L^2(\Omega, \mathbb{R}^\ell) \to L^2(\Omega, \mathbb{R}^\ell)$ be a transformation of $\mathbf{y}$ and $\mathbf{w}$ in a random vector $\mathbf{s} \in L^2(\Omega, \mathbb{R}^\ell)$, i.e.

$$\mathbf{s} = \mathcal{F}(\mathbf{y}, \mathbf{w}).$$

Reasons for using vector $\mathbf{w}$ and transformation $\mathcal{F}$ are detailed in Sections 5, 9 and 10 below.

We propose to determine the PCs and their reconstruction $\widetilde{\mathbf{x}}$ by the trans-

form $\mathcal{T}$ given by

$$\widetilde{\mathbf{x}} = \mathcal{T}(\mathbf{y}, \mathbf{w}) = T_0\mathbf{y} + T_1\mathcal{F}(\mathbf{y}, \mathbf{w}), \tag{6}$$

where T_0 and T_1 are represented by $m \times n$ and $m \times \ell$ matrices, respectively, and

$$\text{rank } [T_0\ T_1] \leq k, \tag{7}$$

where $k = \min\{m, n\}$. Here, three terms, T_0, T_1 and $\mathcal{F}$, are to be determined. Therefore, transform $\mathcal{T}$ will be called the three-term k-rank transform.

A special version of the three-term k-rank transform is considered in Section 9.2 below.

5. Statement of the problems

Below, we assume that $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{w}$ are nonzero vectors.

Problem 1: Find matrices T_0 and T_1 that solve

$$\min_{[T_0\ T_1]} \|\mathbf{x} - [T_0\mathbf{y} + T_1\mathcal{F}(\mathbf{y}, \mathbf{w})]\|_\Omega^2 \tag{8}$$

subject to constraint (7), and determine $\mathcal{F}$ that provides, for $\mathbf{z} = [\mathbf{y}^T\mathbf{s}^T]^T$,

$$E_{zz} = \begin{bmatrix} E_{yy} & \mathbb{O} \\ \mathbb{O} & E_{ss} \end{bmatrix}, \tag{9}$$

where $\mathbb{O}$ denotes the zero matrix. The importance of the condition in (9) is twofold. First, this allows us to facilitate computation associated with a determination of T_0 and T_1. Second, the condition in (9) is used in the solution of the Problem 2 stated below.

The transform obtained from the solution of Problem 1 (see Section 7.2 that follows) is called the *optimal* three-term k-rank transform or the *three-term PCA*.

Problem 2: Show that the error associated with the three-term PCA is less than that of the PCA, GBT1 (see Section 9.1), GBT2 and GKLT (see Section

9.2). Further, show that the computational load associated with the tree-term PCA is less than that of the GBT2 (see Section 11.1).

6. Differences from known techniques

The proposed three-term PCA differs from PCA in [15, 3] in several instances. Unlike PCA in [15, 3], the three-term PCA has the additional terms T_1, $\mathcal{F}$ and $\mathbf{w}$-injection, which lead to the improvement in the associated accuracy of determining PCs and the consecutive reconstruction of PCs to the original vector. As distinct from the PCA in [15, 3], the three-term PCA is always applicable to singular data since it is determined in terms of pseudo-inverse matrices.

Differences of the three-term PCA from the GBT2 are threefold. First, the three-term PCA contains transformation $\mathcal{F}$ aimed to facilitate computation. Second, in the three-term PCA, the procedure for determining principal components and their reconstruction to an estimate of $\mathbf{x}$ is different from that in the GBT2. Indeed, the three-term PCA can be written as

$$\mathcal{T}(\mathbf{y}, \mathbf{w}) = R_1 P_1 \mathbf{y} + R_2 P_2 \mathbf{s}, \tag{10}$$

where R_1, R_2, P_1 and P_2 are obtained in the form different from those in the GBT2 (see Theorem 1 below). Third, in Section 9.2 that follows, we show that a special transformation of vector $\mathbf{s}$ to a vector $\widetilde{\mathbf{s}}$ of a greater dimensionality allows us to achieve the better associated accuracy of $\mathbf{x}$ estimation. Differences from the GKLT in (4) are similar and even stronger since the GKLT contains vector $\mathbf{y}^2$ (not $\mathbf{v}$ as the GBT2) which cannot be changed.

The above differences imply the improvement in the performance of the three-term PCA. This issue is detailed in Sections 9 and 11 that follow.

7. Solution of Problem 1

7.1. Preliminaries

First, we recall some known results that will be used in the solution of Problems 1 and 2.

Proposition 1. [12, Theorem 1.21, p. 44] *Let M be a positive semi-definite matrix given in the block form $M = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix}$, where blocks A and C are square. Let $S = C - B^T A^\dagger B$ and*

$$N = \begin{bmatrix} A^\dagger + A^\dagger B S^\dagger B^T A^\dagger & -A^\dagger B S^\dagger \\ -S^\dagger B^T A^\dagger & S^\dagger \end{bmatrix}.$$

Then $N = M^\dagger$ if and only if $rank(M) = rank(A) + rank(C)$.

Proposition 2. [27, p. 217] *If M is positive definite, then the condition $rank(M) = rank(A) + rank(C)$ of Proposition 1 is always true and $N = M^{-1}$.*

Proposition 3. [10, Lemma 4.5.11] *For any matrices A and B,*

$$\operatorname{rank} \begin{bmatrix} A & \mathbb{O} \\ \mathbb{O} & B \end{bmatrix} = \operatorname{rank}(A) + \operatorname{rank}(B). \tag{11}$$

Proposition 4. (Weyl's inequality) [11, Corollary 4.3.15] *Let A and B be $m \times m$ symmetric matrices and let singular values $\sigma_i(A)$, $\sigma_i(B)$ and $\sigma_i(A+B)$, for $i = 1, \ldots, m$, be arranged in decreasing order. Then, for $i = 1, \ldots, m$,*

$$\sigma_i(A) + \sigma_m(B) \leq \sigma_i(A+B) \leq \sigma_i(A) + \sigma_1(B). \tag{12}$$

7.2. Determination of three-term PCA

Let

$$P_{M,L} = \sum_{k=1}^{\operatorname{rank}(M)} u_k u_k^T \in \mathbb{R}^{m \times m}, \quad P_{M,R} = \sum_{j=1}^{\operatorname{rank}(M)} v_j v_j^T \in \mathbb{R}^{n \times n}$$

be the orthogonal projections on the range of matrices M and M^T respectively, and let

$$[M]_k = \sum_{i=1}^{k} \sigma_i(M) u_i v_i^T \in \mathbb{R}^{m \times n} \tag{13}$$

for $k = 1, \ldots, \text{rank}\,(M)$, be the truncated SVD of M. For $k > \text{rank}\,(M)$, we define $[M]_k = M$ $(= M_{\text{rank}\,(M)})$. For $1 \leq k < \text{rank}\,(M)$, the matrix M_k is uniquely defined if and only if $\sigma_k(M) > \sigma_{k+1}(M)$.

Further, $M^{1/2}$ denotes a matrix square root for matrix M. For the covariance matrix E_{xx}, we denote $E_{xx}^{1/2\dagger} := (E_{xx}^{1/2})^\dagger$. Matrix $E_{xx}^{1/2\dagger}$ is unique since E_{xx} is positive semidefinite. The Frobenius matrix norm is denoted by $\|\cdot\|$.

Let us denote $G_{xy} = E_{xy}E_{yy}^\dagger$ and $G_z = E_{xz}E_{zz}^\dagger E_{zx}$. Similar to $U_{G_q,k}$ in (3), $U_{G_z,k}$ denotes the matrix formed by the first k columns of U_{G_z}. Recall that, as before in (9), $\mathbf{z} = [\mathbf{y}^T \mathbf{s}^T]^T$.

Theorem 1. *Let transformation $\mathcal{F}$ in (6) be determined by*

$$\mathcal{F}(\mathbf{y}, \mathbf{w}) = \mathbf{s} = \mathbf{w} - G_{wy}\mathbf{y}. \tag{14}$$

Then (9) is true, and T_0 and T_1 that solve the problem in (8), (7) are such that

$$[T_0 \;\; T_1] = U_{G_z,k} U_{G_z,k}^T [G_{xy} \;\; G_{xs}](I + N) \tag{15}$$

where $N = M(I - P_{E_{zz}^{1/2},L})$ and matrix M is arbitrary[3]*. The unique minimum norm solution of problem (8), (7) is given by*

$$T_0 = U_{G_z,k} U_{G_z,k}^T G_{xy} \quad \textit{and} \quad T_1 = U_{G_z,k} U_{G_z,k}^T G_{xs}, \tag{16}$$

where

$$G_z = G_y + G_s. \tag{17}$$

[3] In other words, the solution is not unique.

Proof 1. *For vector* $\mathbf{s}$ *defined by (14),*

$$E_{ys} = E[\mathbf{y}(\mathbf{w} - E_{wy}E_{yy}^{\dagger}\mathbf{y})^T] = E_{yw} - E_{yy}E_{yy}^{\dagger}E_{yw} = \mathbb{O}$$

because by Corollary 1 in [25]*,* $E_{yw} = E_{yy}E_{yy}^{\dagger}E_{yw}$*. Then (9) follows. Further, since* $\|\mathbf{x}\|_{\Omega}^2 = \operatorname{tr} E[\mathbf{x}\mathbf{x}^T]$ *(see* [22, pp. 166-167]*) then, for* $T = [T_0\, T_1]$*,*

$$\begin{aligned} \|\mathbf{x} - [T_0\mathbf{y} + T_1\mathcal{F}(\mathbf{y},\mathbf{w})]\|_{\Omega}^2 &= \|\mathbf{x} - T\mathbf{z}\|_{\Omega}^2 \\ &= \operatorname{tr} E\{(\mathbf{x} - T\mathbf{z})(\mathbf{x} - T\mathbf{z})^T\} \\ &= \|E_{xx}^{1/2}\|^2 - \|E_{xz}E_{zz}^{1/2^{\dagger}}\|^2 \\ &\quad + \|E_{xz}E_{zz}^{1/2^{\dagger}} - TE_{zz}^{1/2}\|^2. \end{aligned} \tag{18}$$

Therefore, the problem in (8)–(7) is reduced to

$$\min_{T:\operatorname{rank} T \le k} \|E_{xz}E_{zz}^{1/2^{\dagger}} - TE_{zz}^{1/2}\|^2. \tag{19}$$

Its solution is given in [23] by

$$T = [T_0\ T_1] = [E_{xz}E_{zz}^{\dagger\ 1/2}]_k E_{zz}^{\dagger\ 1/2}(I + N). \tag{20}$$

Let us write $U_Q\Sigma_Q V_Q^T = Q$ *for the SVD of* $Q = E_{xz}E_{zz}^{\dagger\ 1/2}$*. Then by [24],*

$$[E_{xz}E_{zz}^{\dagger\ 1/2}]_k = U_{Q,k}U_{Q,k}^T E_{xz}E_{zz}^{\dagger\ 1/2}. \tag{21}$$

Since $G_z = QQ^T$ *then* $U_{G_z} = U_Q$ *and* $U_{G_z,k} = U_{Q,k}$*. Therefore, (20) and (21) imply*

$$T = [T_0\ T_1] = U_{G_z,k}U_{G_z,k}^T G_{xz}(I + N). \tag{22}$$

Here, on the basis of (9),

$$G_{xz} = [E_{xy} \;\; E_{xs}] \begin{bmatrix} E_{yy}^{\dagger} & \mathbb{O} \\ \mathbb{O} & E_{ss}^{\dagger} \end{bmatrix} = [G_{xy} \;\; G_{xs}] \tag{23}$$

and

$$\begin{aligned} G_z &= [E_{xy} \;\; E_{xs}] \begin{bmatrix} E_{yy}^{\dagger} & \mathbb{O} \\ \mathbb{O} & E_{ss}^{\dagger} \end{bmatrix} \begin{bmatrix} E_{yx} \\ E_{sx} \end{bmatrix} \\ &= E_{xy}E_{yy}^{\dagger}E_{yx} + E_{xs}E_{ss}^{\dagger}E_{sx} = G_y + G_s. \end{aligned} \tag{24}$$

Then (15), (16) and (17) follow. ■

Thus, the three-term PCA is represented by (6), (14), (15) and (16).

8. Analysis of the error associated with three-term PCA

Let us denote the error associated with the three-term PCA by

$$\varepsilon_{m,n,\ell}(T_0, T_1) = \min_{\substack{[T_0 \;\; T_1]: \\ \text{rank } [T_0 \;\; T_1] \le k}} \|\mathbf{x} - [T_0\mathbf{y} + T_1\mathcal{F}(\mathbf{y}, \mathbf{w})]\|_{\Omega}^2. \tag{25}$$

The following theorem establishes a priori determination of $\varepsilon_{m,n,\ell}(T_0, T_1)$.

Theorem 2. *Let $\mathcal{F}$, T_0 and T_1 be determined by Theorem 1. Then*

$$\varepsilon_{m,n,\ell}(T_0, T_1) = \|E_{xx}^{1/2}\|^2 - \sum_{i=1}^{k} \sigma_i(G_z). \tag{26}$$

Proof 2. *It follows from (18) and (20) that*

$$\begin{aligned} &\varepsilon_{m,n,\ell}(T_0, T_1) \\ &= \|E_{xx}^{1/2}\|^2 - \|E_{xz}{E_{zz}^{1/2}}^{\dagger}\|^2 + \|E_{xz}{E_{zz}^{1/2}}^{\dagger} - [E_{xz}{E_{zz}^{\dagger}}^{1/2}]_k {E_{zz}^{\dagger}}^{1/2}(I+N)E_{zz}^{1/2}\|^2 \\ &= \|E_{xx}^{1/2}\|^2 - \|E_{xz}{E_{zz}^{1/2}}^{\dagger}\|^2 + \|E_{xz}{E_{zz}^{1/2}}^{\dagger} - [E_{xz}{E_{zz}^{\dagger}}^{1/2}]_k\|^2. \end{aligned} \tag{27}$$

The latter is true because by Lemma 42 in [22, p. 311],

$$[E_{xz}E_{zz}^{\dagger\ 1/2}]_k = [E_{xz}E_{zz}^{\dagger\ 1/2}]_k E_{zz}^{\dagger\ 1/2}E_{zz}^{1/2}.$$

Then

$$\begin{aligned}\varepsilon_{m,n,\ell}(T_0,T_1) &= \|E_{xx}^{1/2}\|^2 - \sum_{i=1}^{m}\sigma_i(G_z) + \sum_{i=k+1}^{m}\sigma_i(G_z)\\ &= \|E_{xx}^{1/2}\|^2 - \sum_{i=1}^{k}\sigma_i(G_z). \qquad (28)\end{aligned}$$

Thus, (26) is true. ■

9. Advantages of three-term PCA

Here and in Section 11 below, we justify in detail the advantages of the three-term PCA that have been highlighted in Section 3.

9.1. Solution of Problem 2. Improvement in the associated error compared to PCA and GBT1

We wish to show that the error associated with the three-term PCA, $\varepsilon_{m,n,\ell}(T_0,T_1)$, is less than that of PCA and the GBT1 [24]. A similar statement has been provided in Corollary 3 in [25] under the condition which is difficult to verify. In Theorems 3 and 4 below, we show that the condition can be omitted. Let us denote the error associated with the GBT1 by

$$\varepsilon_{m,n}(B_0) = \min_{\substack{B_0\in\mathbb{R}^{m\times n}:\\ \text{rank }(B_0)\le k}} \|\mathbf{x}-B_0\mathbf{y}\|_\Omega^2. \qquad (29)$$

Matrix $B_0 = R_1P_1$ that solves the RHS in (29) follows from (3) if $R_1P_2\mathbf{v} = \mathbf{0}$ (see Section 2).

Theorem 3. *For any non-zero random vectors* $\mathbf{x},\mathbf{y}$ *and* $\mathbf{w}$,

$$\varepsilon_{m,n,\ell}(T_0,T_1) \le \varepsilon_{m,n}(B_0). \qquad (30)$$

If $G_s = E_{xs}E_{ss}^{\dagger}E_{sx}$ is positive definite then

$$\varepsilon_{m,n,\ell}(T_0, T_1) < \varepsilon_{m,n}(B_0). \tag{31}$$

Proof 3. *It is known [24] that*

$$\varepsilon_{m,n}(B_0) = \|E_{xx}^{1/2}\|^2 - \sum_{i=1}^{k} \sigma_i(G_y). \tag{32}$$

Consider $G_z = G_y + (G_z - G_y)$. Clearly, $G_z - G_y$ is a symmetric matrix. Then on the basis of (12) in the above Proposition 4,

$$\sigma_i(G_y) + \sigma_m(G_z - G_y) \leq \sigma_i(G_z), \tag{33}$$

where

$$G_z - G_y = G_s = MM^T$$

and $M = E_{xs}E_{ss}^{\dagger\ 1/2}$. Thus, by Theorem 7.3 in [27], $G_z - G_y$ is a positive semi-definite matrix and then all its eigenvalues are nonnegative [9, p. 167], i.e., $\sigma_i(G_z - G_y) \geq 0$. Therefore, (33) implies $\sigma_i(G_y) \leq \sigma_i(G_z)$, for all $i = 1, \ldots, n$, and then

$$\sum_{i=1}^{k} \sigma_i(G_y) \leq \sum_{i=1}^{k} \sigma_i(G_z). \tag{34}$$

As a result, (30) follows from (26), (32) and (34). In particular, if G_s is positive definite then $\sigma_i(G_z - G_y) > 0$, for $i = 1, \ldots, n$ and therefore, (31) is true. ■

In the following Theorem 4, we refine the result obtained in the above Theorem 3.

Theorem 4. *Let, as before, $k = \min\{m, n\}$. There exists $\gamma \in [\sigma_m(G_s), \sigma_1(G_s)]$ such that*

$$\varepsilon_{m,n,\ell}(T_0, T_1) = \varepsilon_{m,n}(B_0) - k\gamma, \tag{35}$$

i.e., the error associated with the three-term PCA is less than that of the GBT1 by $k\gamma$.

Proof 4. *The Weyl's inequality in (12) and the equality in (24) imply, for* $i = 1, \ldots, m$,

$$\sigma_i(G_y) + \sigma_m(G_s) \leq \sigma_i(G_z) \leq \sigma_i(G_y) + \sigma_1(G_s)$$

which, in turn, implies

$$\sigma_m(G_s) \leq \sigma_i(G_z) - \sigma_i(G_y) \leq \sigma_1(G_s)$$

and

$$\sum_{i=1}^{k} \sigma_m(G_s) \leq \sum_{i=1}^{k} [\sigma_i(G_z) - \sigma_i(G_y)] \leq \sum_{i=1}^{k} \sigma_1(G_s).$$

Therefore,

$$k\sigma_m(G_s) \leq \sum_{i=1}^{k} \sigma_i(G_z) - \sum_{i=1}^{k} \sigma_i(G_y) \leq k\sigma_1(G_s)$$

and

$$k\sigma_m(G_s) \leq \left(\|E_{xx}^{1/2}\|^2 - \sum_{i=1}^{k} \sigma_i(G_y) \right) - \left(\|E_{xx}^{1/2}\|^2 - \sum_{i=1}^{k} \sigma_i(G_z) \right) \leq k\sigma_1(G_s).$$

Thus

$$k\sigma_m(G_s) \leq \varepsilon_{m,n}(B_0) - \varepsilon_{m,n,\ell}(T_0, T_1) \leq k\sigma_1(G_s)$$

and

$$\frac{\varepsilon_{m,n}(B_0) - \varepsilon_{m,n,\ell}(T_0, T_1)}{k} \in [\sigma_m(G_s), \sigma_1(G_s)],$$

and then (35) follows. ■

Remark 1. *If G_s is a full rank matrix then $\sigma_m(G_s) \neq 0$ and therefore, $\gamma \neq 0$, i.e. in this case, (35) implies that $\varepsilon_{m,n,\ell}(T_0, T_1)$ is always less than $\varepsilon_{m,n}(B_0)$. If* rank $(G_s) = r_s$ *where $r_s < m$ then $\sigma_m(G_s) = 0$ and $\gamma \in [0, \sigma_1(G_s)]$, i.e. in this case, γ might be equal to* 0.

Remark 2. *Recall that PCA is a particular case of the GBT1 (see Section 2). Therefore, in Theorems 3 and 4, $\varepsilon_{m,n}(B_0)$ can be treated as the error associated with PCA, under the restriction that matrix E_{yy} is non-singular.*

9.2. Decrease in the error associated with the three-term PCA with the increase in the injection dimension

In Theorem 5 that follows we show that the error $\varepsilon_{m,n,\ell}(T_0, T_1)$ associated with the three-term PCA (represented by (6), (14)–(16)) can be decreased if vector $\mathbf{s}$ is extended to a new vector $\widetilde{\mathbf{s}}$ of a dimension which is larger than that of vector $\mathbf{s}$. The vector $\widetilde{\mathbf{s}}$ is constructed as $\widetilde{\mathbf{s}} = [\mathbf{s}^T \mathbf{g}^T]^T$ where $\mathbf{g} = \mathbf{h} - G_{hz}\mathbf{z} \in L^2(\Omega, \mathbb{R}^\eta)$, $G_{hz} = E_{hz}E_{zz}^\dagger$ and $\mathbf{h} \in L^2(\Omega, \mathbb{R}^\eta)$ is arbitrary. As we mentioned before, similar to $\mathbf{w}$-injection, vector $\mathbf{h}$ is called the $\mathbf{h}$-injection. As before, $\mathbf{s}$ is defined by (14) and $\mathbf{z} = [\mathbf{y}^T \mathbf{s}^T]^T$. Thus, $\widetilde{\mathbf{s}} \in L^2(\Omega, \mathbb{R}^{(\ell+\eta)})$ while $\mathbf{s} \in L^2(\Omega, \mathbb{R}^\ell)$, i.e. the dimension of $\widetilde{\mathbf{s}}$ is larger than that of $\mathbf{s}$ by η entries. In terms of $\widetilde{\mathbf{s}}$, the three-term PCA is represented as

$$\mathcal{S}(\mathbf{y}, \mathbf{w}, \mathbf{h}) = S_0\mathbf{y} + S_1\widetilde{\mathbf{s}}, \tag{36}$$

where similar to T_0 and T_1 in (16), and for $\widetilde{\mathbf{z}} = [\mathbf{y}^T \ \widetilde{\mathbf{s}}^T]^T$, matrices S_0 and S_1 are given by

$$S_0 = U_{G_{\widetilde{z}},k}U_{G_{\widetilde{z}},k}^T G_{xy} \quad \text{and} \quad S_1 = U_{G_{\widetilde{z}},k}U_{G_{\widetilde{z}},k}^T G_{x\widetilde{s}}. \tag{37}$$

Here, $G_{\widetilde{z}} = G_y + G_{\widetilde{s}}$ and $G_{\widetilde{s}} = E_{x\widetilde{s}}E_{\widetilde{s}\widetilde{s}}^\dagger E_{\widetilde{s}x}$. The associated error is denoted by

$$\varepsilon_{m,n,\ell+\eta}(S_0, S_1) = \min_{\substack{[S_0 \ S_1] \in \mathbb{R}^{m\times(n+\ell+\eta)}: \\ \text{rank } [S_0 \ S_1] \leq k}} \|\mathbf{x} - [S_0\mathbf{y} + S_1\widetilde{\mathbf{s}}]\|_\Omega^2. \tag{38}$$

Theorem 5. *For any non-zero random vectors* $\mathbf{x}, \mathbf{y}, \mathbf{w}$ *and* $\mathbf{h}$*,*

$$\varepsilon_{m,n,\ell+\eta}(S_0, S_1) \leq \varepsilon_{m,n,\ell}(T_0, T_1). \tag{39}$$

If $G_s = E_{xs}E_{ss}^{\dagger}E_{sx}$ *is positive definite then*

$$\varepsilon_{m,n,\ell+\eta}(S_0, S_1) < \varepsilon_{m,n,\ell}(T_0, T_1). \tag{40}$$

Proof 5. *Let us represent* S_1 *in terms of two blocks,* S_{11} *and* S_{12}*, i.e.,* $S_1 = [S_{11}\ S_{12}]$*, and also write* $\widehat{S} = [S_0\ S_{11}]$*. Then*

$$\begin{aligned}
&\min_{\substack{[S_0\ S_1]\in\mathbb{R}^{m\times(n+\ell+\eta)}:\\ \operatorname{rank}[S_0\ S_1]\leq k}} \|\mathbf{x} - [S_0\mathbf{y} + S_1\widetilde{\mathbf{s}}]\|_{\Omega}^2 \\
&\qquad = \min_{\substack{[S_0\ S_1]\in\mathbb{R}^{m\times(n+\ell+\eta)}:\\ \operatorname{rank}[S_0\ S_1]\leq k}} \left\|\mathbf{x} - \left[S_0\mathbf{y} + S_1\begin{bmatrix}\mathbf{s}\\ \mathbf{g}\end{bmatrix}\right]\right\|_{\Omega}^2 \\
&\qquad = \min_{\substack{[S_0\ S_1]\in\mathbb{R}^{m\times(n+\ell+\eta)}:\\ \operatorname{rank}[S_0\ S_1]\leq k}} \|\mathbf{x} - [S_0\mathbf{y} + S_{11}\mathbf{s} + S_{12}\mathbf{g}]\|_{\Omega}^2 \\
&\qquad = \min_{\substack{[S_0\ S_1]\in\mathbb{R}^{m\times(n+\ell+\eta)}:\\ \operatorname{rank}[S_0\ S_1]\leq k}} \left\|\mathbf{x} - [\widehat{S}\mathbf{z} + S_{12}\mathbf{g}]\right\|_{\Omega}^2.
\end{aligned} \tag{41}$$

Here, $[S_0\ S_1] = [S_0\ S_{11}\ S_{12}] = [\widehat{S}\ S_{12}]$*. Therefore,*

$$\begin{aligned}
&\min_{\substack{[S_0\ S_1]\in\mathbb{R}^{m\times(n+\ell+\eta)}:\\ \operatorname{rank}[S_0\ S_1]\leq k}} \left\|\mathbf{x} - [\widehat{S}\mathbf{z} + S_{12}\mathbf{g}]\right\|_{\Omega}^2 \\
&\qquad = \min_{\substack{[\widehat{S}\ S_{12}]\in\mathbb{R}^{m\times(n+\ell+\eta)}:\\ \operatorname{rank}[\widehat{S}\ S_{12}]\leq k}} \|\mathbf{x} - [\widehat{S}\mathbf{z} + S_{12}\mathbf{g}]\|_{\Omega}^2.
\end{aligned} \tag{42}$$

In (25), let us write $\varepsilon_{m,n,\ell}(T_0, T_1)$ *in terms of* $\mathbf{s}$*,*

$$\varepsilon_{m,n,\ell}(T_0, T_1) = \min_{\substack{[T_0\ T_1]\in\mathbb{R}^{m\times(n+\ell)}:\\ \operatorname{rank}[T_0\ T_1]\leq k}} \|\mathbf{x} - [T_0\mathbf{y} + T_1\mathbf{s}]\|_{\Omega}^2. \tag{43}$$

Then by (30) in Theorem 3,

$$\begin{aligned}
&\min_{\substack{[\widehat{S}\ S_{12}]\in\mathbb{R}^{m\times(n+\ell+\eta)}:\\ \text{rank }[\widehat{S}\ S_{12}]\le k}} \|\mathbf{x}-[\widehat{S}\mathbf{z}+S_{12}\mathbf{g}]\|_\Omega^2 \\
&\qquad\le \min_{\substack{T\in\mathbb{R}^{m\times(n+\ell)}:\\ \text{rank }(T)\le k}} \|\mathbf{x}-T\mathbf{z}\|_\Omega^2 \\
&\qquad= \min_{\substack{[T_1\ T_2]\in\mathbb{R}^{m\times(n+\ell)}:\\ \text{rank }[T_1\ T_2]\le k}} \left\|\mathbf{x}-[T_0\ T_1]\begin{bmatrix}\mathbf{y}\\ \mathbf{s}\end{bmatrix}\right\|_\Omega^2 \\
&\qquad= \min_{\substack{[T_0\ T_1]\in\mathbb{R}^{m\times(n+\ell)}:\\ \text{rank }[T_0\ T_1]\le k}} \|\mathbf{x}-[T_0\mathbf{y}+T_1\mathbf{s}]\|_\Omega^2, \qquad (44)
\end{aligned}$$

where $T=[T_0\ T_1]$, $T_0\in\mathbb{R}^{m\times n}$ *and* $T_1\in\mathbb{R}^{m\times\ell}$. *Then (39) follows from (42) and (44), and (31) implies (40).* ■

Remark 3. *An intuitive explanation of the statement of Theorem 5 is that the increase in the dimension of vector* $\widetilde{\mathbf{s}}$ *implies the increase in the dimension of matrix* S_1 *in (36) so that* $S_1\in\mathbb{R}^{m\times(n+\ell+\eta)}$ *while in (6), (14)–(16),* $T_1\in\mathbb{R}^{m\times(n+\ell)}$. *Therefore, the optimal matrix* S_1 *has* $m\times\eta$ *entries more than* T_1 *to further minimize the associated error. As a result, the three-term PCA in form (36), for the same number of principal components* k, *provides the more accurate reconstruction of* $\mathbf{x}$ *than the three-term PCA in form (6).*

9.3. Decrease in the error associated with the three-term PCA compared with that of the GBT2 and GKLT

In Theorem 6 below, we show that the three-term PCA in (36)-(37) provides the associated accuracy which is better than that of the GBT2 in (1)-(3). Let us denote the error associated with the GBT2 by

$$\varepsilon_{m,n}(B_0,B_1)=\min_{\substack{[B_0\ B_1]\in\mathbb{R}^{m\times(2n)}:\\ \text{rank }[B_0\ B_1]\le k}} \|\mathbf{x}-[B_0\mathbf{y}+B_1\mathbf{v}]\|_\Omega^2, \qquad (45)$$

where $B_0=R_1P_1$ and $B_1=R_1P_2$ are determined by (3). In fact, the result below is a version of Theorem 5 as follows.

Theorem 6. *Let* $\mathbf{s} = \mathbf{v}$ *where* $\ell = n$*, and random vector* $\mathbf{v}$ *is the same as in (1)-(3). Then for any non-zero random vectors* $\mathbf{x}, \mathbf{y}$ *and* $\mathbf{h}$*,*

$$\varepsilon_{m,n,n+\eta}(S_0, S_1) \leq \varepsilon_{m,n}(B_0, B_1). \tag{46}$$

If $G_s = E_{xs}E_{ss}^{\dagger}E_{sx}$ *is positive definite then*

$$\varepsilon_{m,n,n+\eta}(S_0, S_1) < \varepsilon_{m,n}(B_0, B_1). \tag{47}$$

Proof 6. *We observe that Theorem 5 is true for any form of* $\mathbf{s}$*. In particular, it is true for* $\mathbf{s} = \mathbf{v}$ *where* $\ell = n$ *and random vector* $\mathbf{v}$ *is the same as in (1)-(3). Then (46) and (47) follow from (39) and (40), respectively.* ■

Remark 4. *Theorem 6 is also valid for* $\mathbf{s} = \mathbf{y}^2$ *with* $\ell = n$*. Thus, in this case, the three-term PCA in (36)-(37) provides the associated accuracy which is better than that of the GKLT in (4).*

10. Special cases of three-term PCA

In both forms of the three-term PCA represented by (6) and (36), vectors $\mathbf{s}$ and $\widetilde{\mathbf{s}}$ are constructed from auxiliary random vectors called $\mathbf{w}$-injection and $\mathbf{h}$-injection, respectively, which are assumed to be arbitrary. At the same time, a natural desire is to understand if there are approaches for choosing $\mathbf{w}$ and $\mathbf{h}$ which may (or may not) improve the three-term PCA performance. Here, such approaches are considered.

10.1. Case 1. Choice of $\mathbf{w}$ *on the basis of Weierstrass theorem*

For the three-term PCA represented by (6), a seemingly reasonable choice of vector $\mathbf{w}$ is $\mathbf{w} = \mathbf{y}^2$ where $\mathbf{y}^2$ is defined by the Hadamard product, $\mathbf{y}^2 = \mathbf{y} \circ \mathbf{y}$, i.e. by $\mathbf{y}^2(\omega) = [\mathbf{y}_1^2(\omega), \ldots, \mathbf{y}_n^2(\omega)]^T$, for all $\omega \in \Omega$. This is because if $\mathcal{T}(\mathbf{y}, \mathbf{w})$ in (6) is written as $\mathcal{T}(\mathbf{y}, \mathbf{y}^2) = T_0\mathbf{y} + T_1\mathcal{F}(\mathbf{y}, \mathbf{y}^2)$, then $\mathcal{T}(\mathbf{y}, \mathbf{y}^2)$ can be interpreted as a polynomial of the second degree. If T_0 and T_1 are defined by Theorem 1

then $\mathcal{T}(\mathbf{y},\mathbf{y}^2)$ can be considered as an approximation to an 'idealistic' transform $\mathcal{P}$ such that $\mathbf{x}=\mathcal{P}(\mathbf{y})$. On the basis of the Stone-Weierstrass theorem [16, 21], $\mathcal{T}(\mathbf{y},\mathbf{y}^2)$ should seemingly provide a better associated approximation accuracy of $\mathcal{P}(\mathbf{y})$ than that of the first degree polynomial $\mathcal{T}(\mathbf{y})=T_0\mathbf{y}$. Nevertheless, the constraint of the reduced ranks implies a deterioration of $\mathcal{P}(\mathbf{y})$ approximation by $\mathcal{T}(\mathbf{y},\mathbf{y}^2)=T_0\mathbf{y}+T_1\mathbf{y}^2$. That constraint is not a condition of the Stone-Weierstrass theorem. To the best of our knowledge, an extension of the Stone-Weierstrass theorem to a best *rank constrained* estimation of $\mathbf{x}$ is still not justified. Further, if for example, $\mathbf{y}=\mathbf{x}+\boldsymbol{\xi}$ where $\boldsymbol{\xi}$ is a random noise, then $\mathbf{y}^2=\mathbf{x}^2+2\mathbf{x}\circ\boldsymbol{\xi}+\boldsymbol{\xi}^2$, i.e., $\mathbf{y}^2$ becomes even more corrupted than $\mathbf{y}$. Another inconvenience is that a knowledge or evaluation of matrices E_{xy^2} and $E_{y^2y^2}$ is difficult. For the above reasons, the choice of $\mathbf{w}$ in the form $\mathbf{w}=\mathbf{y}^2$ is not preferable.

10.2. Case 2. Choice of $\mathbf{w}$ *as an optimal estimate of* $\mathbf{x}$

Another seemingly reasonable choice of $\mathbf{w}$-injection in (6) is $\mathbf{w}=A\mathbf{y}$ where $A=E_{xy}E_{yy}^{\dagger}$, i.e. $\mathbf{w}=E_{xy}E_{yy}^{\dagger}\mathbf{y}$ is the optimal minimal-norm linear estimation of $\mathbf{x}$ [22]. Nevertheless, in this case, $\mathbf{s}=A(\mathbf{y}-E_{yy}E_{yy}^{\dagger}\mathbf{y})$ and then

$$E_{xs}=(E_{xy}-E_{xy}E_{yy}^{\dagger}E_{yy})A^T=\mathbb{O}$$

since $E_{xy}=E_{xy}E_{yy}^{\dagger}E_{yy}$ [22, p. 168]. The latter implies $E_{xs}E_{ss}^{\dagger}=\mathbb{O}$. As a result, by (16), $T_1=\mathbb{O}$ and then in (6), $\mathcal{T}(\mathbf{y},\mathbf{w})=T_0\mathbf{y}$. In other words, this choice of $\mathbf{w}$ is unreasonable since then the three-term PCA in (6), (16) is reduced to the GBT1 in [24].

10.3. Case 3. Choice of $\mathbf{h}$*: 'worse is better'*

It has been shown in Theorem 5 that the error associated with the three-term PCA represented by (36) decreases if vector $\mathbf{s}\in L^2(\Omega,\mathbb{R}^{\ell})$ is replaced with a new vector $\widetilde{\mathbf{s}}\in L^2(\Omega,\mathbb{R}^{(\ell+\eta)})$ of a larger dimension. Recall, in (36), $\widetilde{\mathbf{s}}$ is formed from an arbitrary $\mathbf{h}\in L^2(\Omega,\mathbb{R}^{\eta})$. In particular, for $\eta=0$, (39) implies

$\varepsilon_{m,n,\ell+\eta}(S_0, S_1) = \varepsilon_{m,n,\ell}(T_0, T_1)$. Thus, the increase in η implies the decrease in the error associated with the three-term PCA represented by (36). Thus, a reasonable (and quite surprising) choice of $\mathbf{h}$ is as follows: $\mathbf{h}$ is random and η is large. This is similar to the concept 'worse is better' [6], i.e., a random $\mathbf{h}$ with large dimension η ('worse') is a preferable option ('better') in terms of practicality and usability over, for example, the choice of $\mathbf{w}$ considered, for instance, in Case 1. In Case 1, the dimension of $\mathbf{w} = \mathbf{y}^2$ is n and cannot be changed while dimension ℓ can vary and, in particular, can be increased. Another advantage over Case 1 is that there is no need to evaluate matrices E_{xy^2} and $E_{y^2y^2}$ as required in Case 1.

In Example 1 that follows, we numerically illustrate Theorem 5 and Case 3.

Example 1. *Let* $\mathbf{x} = [\mathbf{x}_1, \ldots, \mathbf{x}_m]^T \in L^2(\Omega, \mathbb{R}^m)$ *represent a temperature distribution in* m *locations in Australia. Entries of* $\mathbf{x}$ *and corresponding locations are represented in Table 1.*

Entry	Location	Entry	Location
$\mathbf{x}_1$	Canberra	$\mathbf{x}_{18}$	Perth
$\mathbf{x}_2$	Tuggeranong	$\mathbf{x}_{19}$	Kalgoorlie-Boulder
$\mathbf{x}_3$	Sydney	$\mathbf{x}_{20}$	Broome
$\mathbf{x}_4$	Penrith	$\mathbf{x}_{21}$	Hobart
$\mathbf{x}_5$	Wollongong	$\mathbf{x}_{22}$	Launceston
$\mathbf{x}_6$	Melbourne	$\mathbf{x}_{23}$	Devonport
$\mathbf{x}_7$	Ballarat	$\mathbf{x}_{24}$	Darwin
$\mathbf{x}_8$	Albury-Wodonga	$\mathbf{x}_{25}$	Alice Springs
$\mathbf{x}_9$	Bendigo	$\mathbf{x}_{26}$	Tennant Creek
$\mathbf{x}_{10}$	Brisbane	$\mathbf{x}_{27}$	Casey
$\mathbf{x}_{11}$	Cairns	$\mathbf{x}_{28}$	Davis
$\mathbf{x}_{12}$	Townsville	$\mathbf{x}_{29}$	Mawson
$\mathbf{x}_{13}$	Gold Coast	$\mathbf{x}_{30}$	Macquarie Island
$\mathbf{x}_{14}$	Adelaide	$\mathbf{x}_{31}$	Christmas Island
$\mathbf{x}_{15}$	Mount Gambier	$\mathbf{x}_{32}$	Cocos Island
$\mathbf{x}_{16}$	Renmark	$\mathbf{x}_{33}$	Norfolk Island
$\mathbf{x}_{17}$	Port Lincoln	$\mathbf{x}_{34}$	Howe Island

Table 1: Distribution of entries of signal $\mathbf{x}$ and locations in Australia.

Suppose $\omega \in \Omega$ *is associated with time* $t_\omega \in [0, 24]$ *of the temperature measurement. Then* $\mathbf{x}_j(\omega)$*, for* $j = 1, \ldots, m$*, is a temperature in the* j*th location at*

time t_ω. The values of the minimum and maximum daily temperature, and the temperature at 9am and 3pm, in $m = 34$ specific locations of Australia, for each day in 2016, are provided[4] *by the Bureau of Meteorology of the Australian Government [2]. In particular, the distribution of the maximum daily temperature in 2016 in all 34 locations in Australia is diagrammatically represented in Fig. 1. Let t_{min} and t_{max} denote times when minimal and maximum temperature occur. We denote by $\omega_{t_{min}}$, $\omega_{t_{max}}$, ω_{9am} and ω_{3pm} outcomes associated with times t_{min}, t_{max}, 9am and 3pm, respectively. Further, for $j = 1, \dots, m$, we denote by $\mathbf{x}_{j,(date)}(\omega)$ a temperature in the j-th location at time t_ω on the date labeled as 'date'. For example, on 07/07/2016, the corresponding temperature values in Ballarat are $\mathbf{x}_{7,(07/07/2016)}(\omega_{t_{min}}) = 7.4$, $\mathbf{x}_{7,(07/07/2016)}(\omega_{9am}) = 8.1$, $\mathbf{x}_{7,(07/07/2016)}(\omega_{3pm}) = 9.2$, $\mathbf{x}_{7,(07/07/2016)}(\omega_{t_{max}}) = 9.5$.*

Let $\mathbf{y} = A\mathbf{x} + \boldsymbol{\xi}$ where $A \in \mathbb{R}^{m\times m}$ is an arbitrary matrix with uniformly distributed random entries and $\boldsymbol{\xi} \in L^2(\Omega, \mathbb{R}^m)$ is white noise, i.e. $E_{\xi\xi} = \sigma^2 I$. Further, let $\mathbf{w} \in L^2(\Omega, \mathbb{R}^\ell)$ and $\mathbf{h} \in L^2(\Omega, \mathbb{R}^\eta)$ be Gaussian random vectors used in the three-term PCA given by (36), (37). It is assumed that noise $\boldsymbol{\xi}$ is uncorrelated with $\mathbf{x}$, $\mathbf{w}$ and $\mathbf{h}$. Covariance matrices are represented in terms of samples. For example, $E_{xw} = \frac{1}{p}XW^T$ and $E_{hh} = \frac{1}{p}HH^T$ where $X \in \mathbb{R}^{m\times p}$, $W \in \mathbb{R}^{\ell\times p}$ and $H \in \mathbb{R}^{\eta\times p}$ are samples of $\mathbf{x}$, $\mathbf{w}$ and $\mathbf{h}$, and p is the number of samples. Other covariance matrices are represented similarly. In this example, we consider four specific samples of $\mathbf{x}$ as follows. Matrices $X = X_{9am} \in \mathbb{R}^{m\times 366}$ and $X = X_{3pm} \in \mathbb{R}^{m\times 366}$ represent the temperature taken each day in 2016 at 9am in all m locations. Entries of matrices $X_{max} \in \mathbb{R}^{m\times 366}$ and $X_{min} \in \mathbb{R}^{m\times 366}$ are values of the maximum and minimum temperature, respectively, for each day in 2016 in all m locations. The database was taken from the Bureau of Meteorology website of Australian Government [2]. Matrices W and H were created by MATLAB command `rand(m,p)`.

[4] There were 366 days in 2016. There are four values of temperature per day. Thus, we have 1464 temperature values for 2016. From a statistical point of view, the temperature measurements should be provided in more times of a day but such data are not available for us.

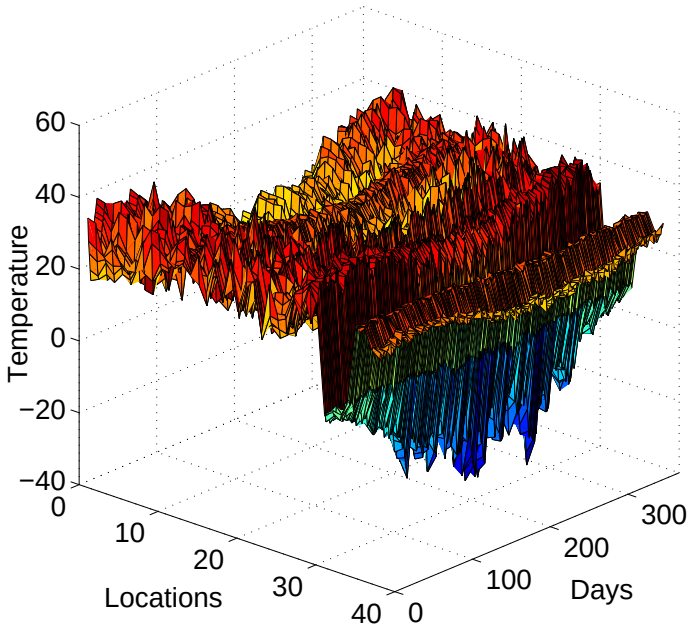


Figure 1: Values of maximal temperature in 34 locations of Australia in 2016

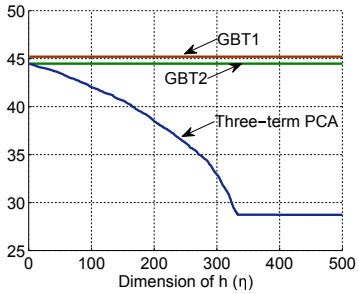
We consider eight different cases of simulations. In each case, matrix X is chosen either as one of matrices X_{9am}, X_{3pm}, X_{min}, X_{max}, or their combinations as follows: For each case, the diagrams of the error associated with the

	Case 1	Case 2	Case 3	Case 4	Case 5
X	X_{min}	X_{9am}	X_{3pm}	X_{max}	$[X_{9am}, X_{3pm}]$
p	366	366	366	366	732

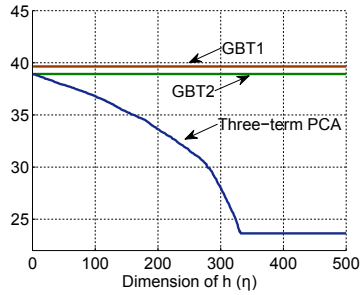
	Case 6	Case 7	Case 8
X	$[X_{min}, X_{max}]$	$[X_{min}, X_{9am}, X_{3pm}]$	$[X_{min}, X_{9am}, X_{3pm}, X_{max}]$
p	732	1098	1464

GBT1, GBT2 and three-terms PCA in form (36)-(37), for $m = \ell = 34$, $k = 17$ and $\sigma = 1$, versus the dimension $\eta = 0, 1, \ldots, 500$ of vector $\mathbf{h}$ are shown in Fig. 2.

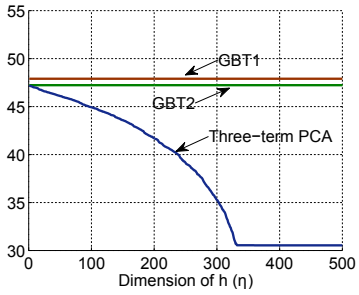
It follows from the diagrams for all cases represented in Fig. 2, that the



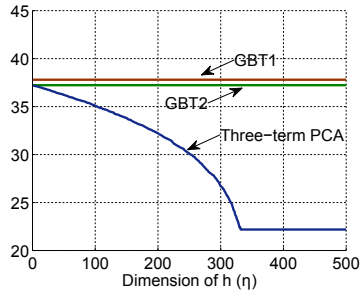
(a) Case 1: $X = X_{min}$



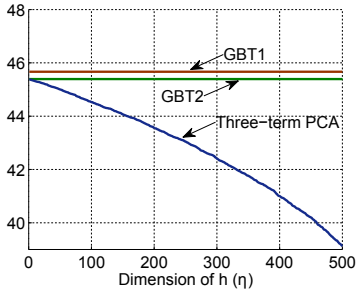
(b) Case 2: $X = X_{9am}$



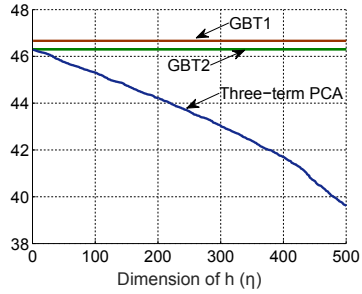
(c) Case 3: $X = X_{3pm}$



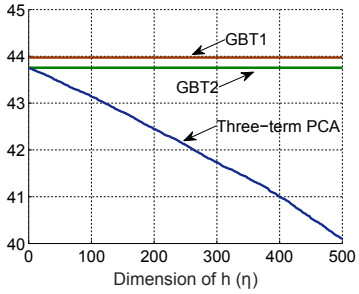
(d) Case 4: $X = X_{max}$



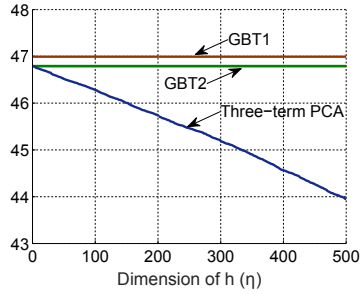
(e) Case 5: $X = [X_{9am}\ X_{3pm}]$



(f) Case 6: $X = [X_{min}\ X_{max}]$



(g) Case 7: $X = [X_{min}\ X_{9am}\ X_{3pm}]$



(h) Case 8: $X = [X_{min}\ X_{9AM}\ X_{3PM}\ X_{max}]$

Figure 2: Errors associated with the GBT1, GBT2 and three-terms PCA versus dimension η of H.

error associated with the three-term PCA decreases as η increases. This is the numerical illustration of Theorem 5 and the Case 3 considered in Section 10.3. In particular, in Figs. 2 (a)-(d), for $\eta = 335, \ldots, 500$, the error associated with the three-term PCA coincides with that of the GBT1 applied to the case when $\mathbf{y} = \mathbf{x}$, i.e.. with that of PCA applied to the data without any noise.

Further, Fig. 3 illustrates the following observation. For $\sigma \in [0, 2]$ and $\eta \in [0, 300]$, the behavior of the error associated with the three-term PCA is similar: the increase in η implies the decrease in the error. Interestingly, for $\eta \in [300, 500]$, the error remains constantly small regardless of the value of σ. Similar to the above, for $\eta \in [300, 500]$, the error associated with the three-term PCA coincides with that that of PCA applied to the data without any noise.

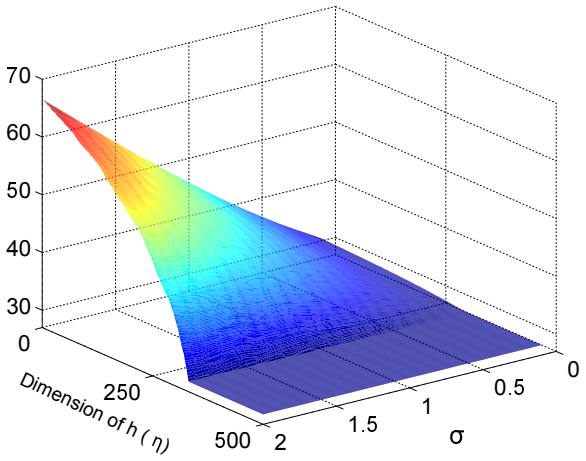


Figure 3: Diagrams of errors associated with the three-terms PCA versus dimension η of $\mathbf{h}$ and values of σ of $E_{\xi\xi}$.

10.4. Case 4. Pure filtering

One more special case of the three-term PCA is as follows. Consider transform $\mathcal{T}_1$ defined by

$$\mathcal{T}_1(\mathbf{y}, \mathbf{w}) = A_0\mathbf{y} + A_1\mathcal{F}(\mathbf{y}, \mathbf{w}), \tag{48}$$

where $A_0 \in \mathbb{R}^{m\times n}$ and $A_1 \in \mathbb{R}^{m\times \ell}$ are full rank matrices. Optimal A_0 and A_1 are determined from the solution of the following problem: Given E_{xy}, E_{yy}, E_{yw} are E_{ww}, find full rank A_0 and A_1 that solve

$$\min_{A_0,A_1} \|\mathbf{x} - [A_0\mathbf{y} + A_1\mathcal{F}(\mathbf{y},\mathbf{w})]\|_\Omega^2. \tag{49}$$

In other words, $\mathcal{T}_1$ is a pure filter, with no principal component determination. As before, we write $\mathbf{s} = \mathcal{F}(\mathbf{y},\mathbf{w}) = \mathbf{w} - E_{wy}E_{yy}^\dagger\mathbf{y}$. We call $\mathcal{T}_1$ the three-term filter (TTF).

Theorem 7. *Minimal norm solution to problem (49) is given by*

$$A_0 = E_{xy}E_{yy}^\dagger \quad \textit{and} \quad A_1 = E_{xs}E_{ss}^\dagger. \tag{50}$$

The associated error is represented by

$$\begin{aligned}&\min_{A_0,A_1} \|\mathbf{x} - [A_0\mathbf{y} + A_1\mathbf{s}]\|_\Omega^2 \\ &\quad = \|E_{xx}^{1/2}\|^2 - \|[E_{xy}{E_{yy}^{1/2}}^\dagger\|^2 - \|E_{xs}{E_{ss}^{1/2}}^\dagger]\|^2.\end{aligned} \tag{51}$$

Proof 7. *Optimal full rank A_0 and A_1 given by (50) follow from (20) where ${E_{zz}^\dagger}^{1/2}$ is represented by ${E_{zz}^\dagger}^{1/2} = \begin{bmatrix} {E_{yy}^\dagger}^{1/2} & \mathbb{O} \\ \mathbb{O} & {E_{ss}^\dagger}^{1/2} \end{bmatrix}$ and $[E_{xz}{E_{zz}^\dagger}^{1/2}]_k$ should be replaced with $E_{xz}{E_{zz}^\dagger}^{1/2}$. Further, for $A = [A_0\ A_1]$,*

$$\|\mathbf{x} - [A_1\mathbf{y} + A_1\mathbf{s}]\|_\Omega^2 = \|E_{xx}^{1/2}\|^2 - \|E_{xz}{E_{zz}^{1/2}}^\dagger\|^2 + \|E_{xz}{E_{zz}^{1/2}}^\dagger - AE_{zz}^{1/2}\|^2. \tag{52}$$

For A_0 and A_1 given by (50), $A = [E_{xy}E_{yy}^\dagger\ E_{xs}E_{ss}^\dagger] = E_{xz}E_{zz}^\dagger$. Therefore,

$$\begin{aligned}\min_{A_1,A_1} \|\mathbf{x} - [A_1\mathbf{y} + A_1\mathbf{s}]\|_\Omega^2 \ &= \|E_{xx}^{1/2}\|^2 - \|E_{xz}{E_{zz}^{1/2}}^\dagger\|^2 \\ &\quad + \|E_{xz}{E_{zz}^{1/2}}^\dagger - E_{xz}E_{zz}^\dagger E_{zz}^{1/2}\|^2 \\ &= \|E_{xx}^{1/2}\|^2 - \|E_{xz}{E_{zz}^{1/2}}^\dagger\|^2\end{aligned}$$

because $E_{zz}^{\dagger}E_{zz}^{1/2} = E_{zz}^{1/2^{\dagger}}$ [22, p. 313]. *Then (51) follows.* ■

The accuracy of the optimal TTF $\mathcal{T}_1$ is better than that of the optimal linear filter $\widetilde{F} = A_0 = E_{xy}E_{yy}^{\dagger}$ [22]. More specifically, the following is true.

Theorem 8. *The error associated with the optimal TTF* $\mathcal{T}_1$ *is less than that of the optimal linear filter* $\widetilde{F} = A_1$ *by* $\|E_{xs}E_{ss}^{1/2^{\dagger}}]\|^2$, *i.e.,*

$$\min_{A_0,A_1} \|\mathbf{x} - [A_0\mathbf{y} + A_1\mathcal{F}(\mathbf{y},\mathbf{w})]\|_{\Omega}^2 = \min_{A_0} \|\mathbf{x} - A_0\mathbf{y}\|^2 - \|E_{xs}E_{ss}^{1/2^{\dagger}}]\|_{\Omega}^2. \tag{53}$$

Proof 8. *The proof follows directly from (51).* ■

11. Advantages of three-term PCA (continued)

11.1. Increase in accuracy compared to that of GBT2 [24] and GKLT

The three-term PCA represented by (6) and (36) has more parameters to optimize than those in the GBT2 (1) and GKLT (4), i.e., it has more degrees of freedom to control the performance. Indeed, in the three-term PCA, the dimension of **w**-injection and **h**-injection are ℓ and η, and they can be varied while in the GBT2, the dimension of auxiliary vector **v** is n and it is fixed. In the GKLT dimension of vector $\mathbf{y}^2$ is also fixed. Thus, in the three-term PCA, ℓ and η represent the additional degrees of freedom.

By Theorem 5, the increase in η implies the improvement in the accuracy of the three-term PCA. Thus, unlike the GBT2, the performance of the three-term PCA is improved because of the increase in the dimension of **h**-injection.

11.2. Improvement in the associated numerical load

In a number of applied problems, dimensions m, n of associated covariance matrices are large. For instance, in the DNA array analysis [1, 26], $m = O(10^4)$. In this case, the associated numerical load needed to compute the covariance matrices increases significantly. Therefore, a method which requires a lower associated numerical load is, of course, preferable.

Here, we wish to illustrate the computational advantage of the three-term PCA represented by (6), (14) and (16) compared to that of the GBT2 in (1) and the GKLT [25]. In both methods, a pseudo-inverse matrix is evaluated by the SVD. The computational load of the three-term PCA (abbreviated as C_{PCA3}) consists of the matrix products in (16), and computation of SVDs for $E_{yy}^{\dagger} \in \mathbb{R}^{n\times n}$, $E_{ss}^{\dagger} \in \mathbb{R}^{\ell\times \ell}$ and $G_z \in \mathbb{R}^{m\times m}$. Recall that $E_{zz}^{\dagger} = \begin{bmatrix} E_{yy}^{\dagger} & \mathbb{O} \\ \mathbb{O} & E_{ss}^{\dagger} \end{bmatrix}$. The GBT2 computational load (C_{GBT2}) consists of computation of the matrix products in (3) and the SVD for $E_{qq}^{\dagger} \in \mathbb{R}^{2n\times 2n}$. The computational load of the GKLT (C_{GKLT}) contains computation of the matrix products given in [25], and computation of the SVDs for $2n \times 2n$ and $m \times 2n$ matrices. Importantly, the dimensions of matrices in the three-term PCA are less than those in the GBT2 and GKLT. For example, the dimension of E_{yy} is twice less than that of E_{qq}. This circumstance implies the decrease in the computational load of the three-term PCA compared to that in the GBT2 and GKLT. Indeed, the product of $m \times n$ and $n \times p$ matrices requires approximately $2mnp$ flops, for large n. The Golub-Reinsch SVD method (as given in [8], p. 254) requires $4mn^2 + 8n^3$ flops to compute the pseudo-inverse for a $m \times n$ matrix[5]. As a result, for $m = n = \ell$,

$$C_{PCA3} = 52m^3 + 2m^2(k+1) \tag{54}$$

while

$$C_{GBT2} = 140m^3 + 2m^2(k+2) \text{ and } C_{GKLT} = 240m^3 + 4m^2(k+1) + mk. \tag{55}$$

That is, for large m and n, C_{PCA3} is about 37% of C_{GBT2} and 22% of C_{GKLT} .

Thus, the three-term PCA may provide a better associated accuracy than that of the GBT2 and GKLT (see Section 9.3, Example 1 and Section 11.1) under the computational load which, for large m, n, is about one third of C_{GBT2} and

[5]The Golub-Reinsch SVD method appears to be more effective than other related methods considered in [8].

a quarter of C_{GKLT}. This observation is illustrated by the following numerical example.

Example 2. *Let* $\mathbf{y} = A\mathbf{x} + \xi$ *where* $\mathbf{x} \in L^2(\Omega, \mathbb{R}^m)$ *is a uniformly distributed random vector,* $\xi \in L^2(\Omega, \mathbb{R}^m)$ *is a Gaussian random vector with variance one and* $A \in \mathbb{R}^{m\times m}$ *is a matrix with normally distributed random entries. We choose* $\mathbf{w} \in L^2(\Omega, \mathbb{R}^\ell)$ *as an uniformly distributed random vector. Covariance matrices* E_{xx}, E_{ww} *and* $E_{\xi\xi}$ *are represented by* $E_{yy} = \frac{1}{s}YY^T$, $E_{ww} = \frac{1}{s}WW^T$, $E_{\xi\xi} = \sigma^2 I$, *where* $Y \in \mathbb{R}^{m\times p}$ *and* $W \in \mathbb{R}^{m\times p}$ *are corresponding sample matrices,* p *is a number of samples, and* $\sigma = 1$. *Suppose only samples of* $\mathbf{y}$ *and* $\mathbf{w}$ *are available, and for simplicity let us assume that matrix* A *is invertible. Then, in particular,* $E_{xx} = A^{-1}(E_{yy} - E_{\xi\xi})A^{-T}$ *and* $E_{xy} = E_{xx}A^T$.

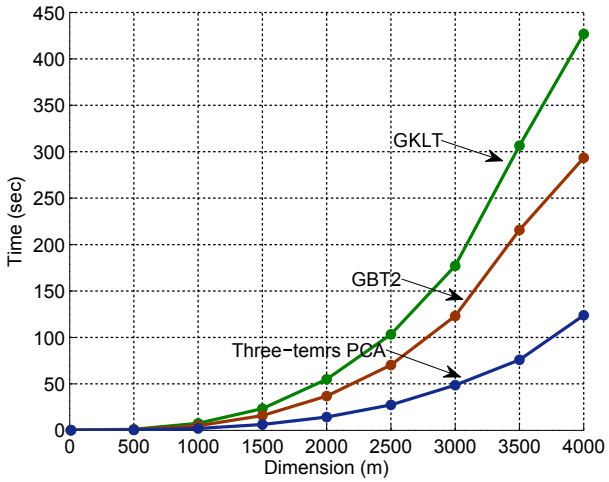


Figure 4: Example 2: Time versus matrix dimension m used to execute the three-term PCA (blue line), GBT2 (red line) and GKLT (green line).

In Fig. 4, for a randomly chosen A, *and* $m = \ell$ *and* $p = 3m$, *typical diagrams of time (in sec.) used to execute the three-term PCA in form (6) and (16), GBT2 (1)-(2) and GKLT [25] versus dimension* m *are represented. The diagrams in Fig. 4 confirm the observation made before this example, i.e., for large* m, C_{PCA3} *is significantly less than* C_{GBT2} *and* C_{GKLT} *(see (54) and (55), respectively).*

12. Final discussion

We have developed the extension of PCA called the three-term PCA. The three-term PCA is presented in two forms considered in Sections 4, 7.2, and 9.2. The associated advantages have been detailed in Sections 9 and 11. We would like to highlight the following observation. The proposed three-term PCA is applied to observed data represented by random vector $\mathbf{y}$. Here, $\mathbf{y}$ is a noisy version of original data $\mathbf{x}$. It is shown that in this case, the error associated with the three-term PCA, $\varepsilon_{m,n,\ell+\eta}(S_0, S_1)$, is less than that of the known generalizations of PCA, i.e. the GBT1, GKLT and GBT2 (see Section 9.3). At the same time, in the ideal case of the observed data *without any noise*, i.e. when $\mathbf{y} = \mathbf{x}$, the three-term PCA coincides with the GBT1 (with $\mathbf{y} = \mathbf{x}$ in the GBT1 as well) which is a generalization of PCA for the case of singular data. Its associated error is minimal among all transforms of the same rank and cannot be improved. Let us denote this error by $\varepsilon_{(y=x)}$. Example 1 above has discovered an important and quite unanticipated feature of the proposed technique as follows. As dimension η of $\mathbf{h}$-injection increases, the error $\varepsilon_{m,n,\ell+\eta}(S_0, S_1)$ decreases up to $\varepsilon_{(y=x)}$. This implies a conjecture that this is true in general, i.e. $\lim_{\eta\to\infty} \varepsilon_{m,n,\ell+\eta}(S_0, S_1) = \varepsilon_{(y=x)}$. We intend to develop a justification of this observation.

References

[1] Orly Alter and Gene H. Golub. Singular value decomposition of genome-scale mRNA lengths distribution reveals asymmetry in RNA gel electrophoresis band broadening. *Processing of the National Academy of Sciences of USA*, 103(32):11828 –11833, 2006.

[2] Australian Goverment. Bureau of Meteorology. http://www.bom.gov.au/climate/dwo/index.shtml, 2016.

[3] D. R. Brillinger. *Time Series: Data Analysis and Theory*. SIAM, San Francisco, 2001.

[4] K. I. Diamantaras and S. Y. Kung. *Principal Component Neural Networks. Theory and Applications*. Jojn Wiley & Soans, Inc., New York, 1996.

[5] K.-L. Du and M. N. S. Swamy. *Neural Networks in Softcomputing Framework*. Springer, London, 2006.

[6] Richard P. Gabriel. The rise of worse is better. http://dreamsongs.com/RiseOfWorseIsBetter.html, 1991.

[7] Yi Gao, Jiazhong Chen, Shengsheng Yu, Jingli Zhou, and Lai-Man Po. The training of Karhunen - Loève transform matrix and its application for H.264 intra coding. *Multimedia Tools and Applications*, 41(1):111–123, 2009.

[8] G.H. Golub and C. F. Van Loan. *Matrix Computations*. Jons Hopkins University Press, 1996.

[9] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 4 edition, 2013.

[10] D.A. Harville. *Matrix Algebra from a Statistician's Perspective*. Springer New York, 2008.

[11] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[12] Roger A. Horn and Fuzhen Zhang. Basic Properties of the Schur Complement. In Fuzhen Zhang, editor, *The Schur Complement and Its Applications*, chapter 1, pages 17–46. Springer US, Boston, MA, 2005.

[13] Y. Hua and Wanquan Liu. Generalized Karhunen-Loeve transform. *IEEE Signal Processing Letters*, 5(6):141–142, June 1998.

[14] Y. Hua, M. Nikpour, and Petre Stoica. Optimal reduced-rank estimation and filtering. *IEEE Transactions on Signal Processing*, 49(3):457–469, Mar 2001.

[15] I.T. Jolliffe. *Principal Component Analysis, Second Edition*. Springer, New York, 2002.

[16] E. Kreyszig. *Introductory Functional Analysis with Applications*. John Willey & Sons, Inc, New York, 1978.

[17] Tuan Nguyen and Isao Yamada. Necessary and sufficient conditions for convergence of the DDT systems of the normalized PAST algorithms. *Signal Processing*, 94:288 – 299, 2014.

[18] T. Piotrowski, R. Cavalcante, and I. Yamada. Stochastic MV-PURE estimator - robust reduced-rank estimator for stochastic linear model. *IEEE Transactions on Signal Processing*, 57(4):1293 – 1303, 2009.

[19] J. A. Saghri, S. Schroeder, and A. G. Tescher. Adaptive two-stage Karhunen-Loeve-transform scheme for spectral decorrelation in hyperspectral bandwidth compression. *Optical Engineering*, 49(5):057001–1 – 057001–7, 2010.

[20] L. Scharf. The SVD and reduced rank signal processing. *Signal Processing*, 25(2):113 – 133, 1991.

[21] V. Timofte. Stone-Weierstrass theorem revisited. *J. Approx. Theory*, 536:45–59, 2005.

[22] A. Torokhti and P. Howlett. *Computational Methods for Modelling of Nonlinear Systems*. Elsevier, Amsterdam, 2007.

[23] Anatoli Torokhti and Shmuel Friedland. Towards theory of generic Principal Component Analysis. *Journal of Multivariate Analysis*, 100(4):661 – 669, 2009.

[24] Anatoli Torokhti and Pablo Soto-Quiros. Generalized Brillinger-Like Transforms. *IEEE Signal Processing Letters*, 23(6):843 – 847, 2016.

[25] A.P. Torokhti and P.G. Howlett. Optimal fixed rank transform of the second degree. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 48(3):309–316, 2001.

[26] Z. Yang and G. Michailidis. A non-negative matrix factorization method for detecting modules in heterogeneous omics multi-modal data. *Biginformatics*, pages 1–8, 2015.

[27] Fuzhen Zhang. *Matrix Theory: Basic Results and Techniques*. Springer-New York, 2011.

Anexo 10

International Journal of Applied and Computational Mathematics

Matrix approximation by a sum of matrix products

--Manuscript Draft--

Manuscript Number:	IACM-D-21-00650
Full Title:	Matrix approximation by a sum of matrix products
Article Type:	Full length article
Funding Information:	
Abstract:	A number of applied tasks is reduced to a matrix approximation. In this paper, we give solutions to the problems of multiple rank constrained matrix approximation in Frobenius norm, which are extensions of the classical approximation of an $m \times n$ matrix A by a matrix of rank at most r.
Corresponding Author:	Pablo Soto-Quiros Instituto Tecnológico de Costa Rica Cartago, CARTAGO COSTA RICA
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	Instituto Tecnológico de Costa Rica
Corresponding Author's Secondary Institution:	
First Author:	Anatoli Torokhti
First Author Secondary Information:	
Order of Authors:	Anatoli Torokhti
	Pablo Soto-Quiros
	Vladimir Ejov
Order of Authors Secondary Information:	
Author Comments:	
Suggested Reviewers:	Esteban Segura Ugalde, Dr. Professor, Universidad de Costa Rica estebansu@gmail.com
	Arnab Patra Indian Institute of Technology Kharagpur arnptr91@gmail.com
	P. D. SRIVASTAVA Indian Institute of Technology pds@iitbhilai.ac.in
	Baiyu Wang Changsha University of Science and Technology - Yuntang Campus: Changsha University of Science and Technology wangbaiyumath@163.com
	Ali Özyapıcı Cyprus International University: Uluslararasi Kibris Universitesi aozyapici@ciu.edu.tr

Highlights of Paper "Matrix approximation by a sum of matrix products"

- In this paper, we solve the problem of multiple ranks constrained matrix approximation in the Frobenius norm, which are extensions of the classical approximation of an $m \times n$ matrix A by a matrix of rank at most r.
- This new problem has several applications in real-world situations, i.e., feature selection, data compression, image processing, blind system identification, signal processing, and system theory.
- The solution to this problem is not developed in the literature review. The proposed solution presented in this paper is new, and it is based on the Generalized rank-constrained matrix approximations.
- Numerical simulations on measuring and theoretical results confirmed the efficiency of the proposed method.

Professor Santanu Saha Ray
Editor-in-Chief
International Journal of Applied and Computational Mathematics

Dear Professor,

I would like to submit the paper "Matrix approximation by a sum of matrix products" by Anatoli Torokthi, Vladimir Ejov, and myself to be considered for publication in the International Journal of Applied and Computational Mathematics

The significance and novelty of this paper are presented as follows:

- We propose a new optimization problem in (3), i.e.,

$$\min_{\substack{X_0\in\mathbb{C}(s_0,g_0,r_0),\ldots,X_p\in\mathbb{C}(s_p,g_p,r_p)\\ C_0\in\mathbb{C}^{g_0\times n},\ldots,C_p\in\mathbb{C}^{g_p\times n}}} \left\| A-\sum_{j=0}^{p} B_j X_j C_j \right\|^2 .$$

- The solution of this problem in equation (3) is not developed in the literature review. The proposed solution presented in this paper is new, and it is based on the Generalized rank-constrained matrix approximations [13].
- The problem (3) and its solution is motivated by requirements associated with a feature selection [3, 17], data compression [15, 16], and image processing [10]. In particular:
    - System output given by matrix *A* is available in tasks associated with the blind system identification, but inputs matrices *C_0,…, C_p* and "system matrices" *X_0,…, X_p* are unknown. They need to be determined so that an associated error is minimized. The input and output signals are random vectors [7, 12, 16, 23], and in reality, the random signal is represented by a sample matrix [2] where each column is a realization of the random vector. This motivates the consideration of the problem and, in particular, the minimization with respect to *C_0,…, C_p*. The work in [1,9] on blind system identification is most relevant to an illustration of the problem under consideration.
    - When matrices *C_0, . . . , C_p* are assumed to be known, is motivated by the tasks in signal processing and system theory where, for the identification of a digital filter or digital system, one needs to determine *X_0, . . . , X_p* only.

I look forward to hearing from you.

Best regards

Juan Pablo Soto Quirós
Escuela de Matemática
Instituto Tecnológico de Costa Rica
https://www.tec.ac.cr/juan-pablo-soto-quiros

Noname manuscript No.
(will be inserted by the editor)

Matrix approximation by a sum of matrix products

Anatoli Torokhti · Pablo Soto-Quiros · Vladimir Ejov

Received: date / Accepted: date

Abstract A number of applied tasks is reduced to a matrix approximation. In this paper, we give solutions to the problems of multiple rank constrained matrix approximation in Frobenius norm, which are extensions of the classical approximation of an $m \times n$ matrix A by a matrix of rank at most r.

Keywords Frobenius Norm · Low-Rank · Optimization

1 Introduction

The problem we consider arises in a number of engineering tasks associated, e.g., with modeling of non-liner systems, data compression and feature selection. We cite [6,13,14,18,22,24] as some related references. In more details, this issue is discussed in Section 2 below.

Let $\mathbb{C}^{m\times n}$ be the set of $m \times n$ complex matrices. Denote by $\mathbb{C}(m,n,r) \subseteq \mathbb{C}^{m\times n}$ the variety of all $m \times n$ matrices of rank at most r. We write $A^* \in \mathbb{C}^{m\times n}$ for the conjugate transpose of $A \in \mathbb{C}^{m\times n}$, $\|\cdot\|$ for the Frobenius norm and $\mathbb{O}$ for the zero matrix.

Anatoli Torokhti
CIAM, University of South Australia, SA 5095, Australia
Tel.: +61-8-83023812
E-mail: anatoli.torokhti@unisa.edu.au

Pablo Soto-Quiros
Instituto Tecnologico de Costa Rica, Apdo. 159-7050, Cartago, Costa Rica
Tel.: +506-25502025
E-mail: jusoto@tec.ac.cr

Vladimir Ejov
Flinders University of South Australia, SA 5001, Australia
Tel.: +61-8-82012110
E-mail: vladimir.ejov@flinders.edu.au

The well-known Eckart-Young theorem [11] provides the solution to the following problem: Given $A \in \mathbb{C}^{m\times n}$, find matrix X that solves

$$\min_{X_0\in\mathbb{C}(m,n,r)} \|A - X\|^2 . \tag{1}$$

The solution is given in terms of the truncated singular value decomposition (SVD) of A. It has been used as an effective tool in a number of important applied problems, and implied a number of extensions. In particular, in [13,22], the following problem has been solved: Given matrices $A \in \mathbb{C}^{m\times n}$, $B \in \mathbb{C}^{m\times s}$ and $C \in \mathbb{C}^{g\times n}$, determine matrix X that solves

$$\min_{X\in\mathbb{C}(m,n,r)} \|A - BXC\|^2 . \tag{2}$$

Important special cases of (2) are considered in [6, 18, 24].

Here, we extend some ideas from [11,13,22] to a solution of the problem as follows. Given matrices $A \in \mathbb{C}^{m\times n}$ and $B_j \in \mathbb{C}^{m\times s_j}$, for $j = 0, \ldots, p$, find minimal Frobenius norm matrices X_j and C_j, for $j = 0, \ldots, p$, that solve

$$\min_{\substack{X_0\in\mathbb{C}(s_0,g_0,r_0),\ldots,X_p\in\mathbb{C}(s_p,g_p,r_p)\\ C_0\in\mathbb{C}^{g_0\times n},\ldots,C_p\in\mathbb{C}^{g_p\times n}}} \left\| A - \sum_{j=0}^{p} B_j X_j C_j \right\|^2 . \tag{3}$$

We call p the degree of $\sum_{j=0}^{p} B_j X_j C_j$. Note that (1) follows from (3) if $p = 0$, $s_0 = m$, $g_0 = n$, $r = r_0$ and $B_0 = C_0 = I$. The problem studied in [13, 22] is a particular case of (3), for $p = 0$. Another particular case of the problem in (3), for $p = 1$ and $B_0 = B_1 = I$, is considered in [20].

2 Motivation

The motivation of the problem in (3) is threefold. First, the rank restrictions for matrices $X_0, \ldots, X_p$ in (3) are motivated, in particular, by requirements associated with a feature selection [3, 17], data compression [15, 16] and image processing [10]. Due to the rank restriction, each X_j is represented as $X_j = G_j H_j$ where $H_j \in \mathbb{C}^{r_j\times g_j}$ transforms matrix C_j to a matrix of the smaller $r_j \times n$ size and $G_j \in \mathbb{C}^{s_j\times r_j}$ reconstructs that smaller matrix to a $s_j \times n$ matrix (i.e., G_j contributes to a reconstruction of matrix A). In this regard, see Remark 1 in Section 3.2 for further details.

Second, in tasks associated with the blind system identification, system output A is available but inputs $C_0, \ldots, C_p$ and 'system matrices' $X_0, \ldots, X_p$ are unknown. They need to be determined so that an associated error is minimized. The input and output signals are random vectors [7, 12, 16, 23] and in reality, the random signal is represented by a sample matrix [2] where each column is a realization of the random vector. This motivates the consideration of the problem in (3) and, in particular, the minimization with respect to $C_0, \ldots, C_p$. The work in [1,9] on the blind system identification is most relevant to an illustration of the problem under consideration. There are many other references on the blind system identification. See, for example, the bibliography in [1,9].

Third, the particular case of the problem in (3) where matrices $C_0, \ldots, C_p$ are assumed to be known, is motivated by the tasks in signal processing and system theory where, for the identification of a digital filter or digital system, one needs to determine $X_0, \ldots, X_p$ only.

Further, in (3), $B_0, \ldots, B_p$ are weighted matrices. An impetus for the development of the model with weighted matrices follows from a number of important problems such as the parameter estimation in linear regression [8, 25] and the identification of multi-input multi-output systems [21]. Based on a priori information, the weighting matrices are used to place a greater importance on some particular entries of the observed data. In particular, it is customary to choose small weights where the errors associated with particular entries of observed data are expected to be large, and vice versa.

A greedy solution to problem (3) is provided in Section 3. Its particular case where matrices $C_0, \ldots, C_p$ are assumed to be known is considered in Section 4. Numerical experiments that illustrate specific features of the proposed methods are represented in Section 5. A discussion of the obtained results and associated open problems are given in Section 6.

3 Solution of problem in (3)

3.1 Solution device of problem in (3)

For $i = 0, 1, \ldots$ and $\ell = 0, \ldots, p$, we denote $\mathcal{X}_\ell^{(i)} = (X_0^{(i+1)}, \ldots, X_{\ell-1}^{(i+1)}, X_{\ell+1}^{(i)}, \ldots, X_p^{(i)})$ where $\mathcal{X}_0^{(i)} = (X_1^{(i)}, \ldots, X_p^{(i)})$ and $\mathcal{X}_p^{(i)} = (X_0^{(i+1)}, \ldots, X_{p-1}^{(i+1)})$.

The device of the solution of problem (3) is represented by a greedy procedure where the i-th loop, for $i = 0, 1, \ldots$, is as follows.

The i-th iterative loop, for $i = 0, 1, \ldots$.

Step 1. For $j=0,\dots,p$, given $\mathcal{X}_j^{(i)}$ and $C_j^{(i)}$, find X_j that solves

$$\min_{X_j\in\mathbb{C}(s_j,g_j,r_j)}\left\|A_j^{(i)}-B_jX_jC_j^{(i)}\right\|^2, \tag{4}$$

where

$$A_0^{(i)}=A-\sum_{k=1}^{p}B_kX_k^{(i)}C_k^{(i)},\quad A_p^{(i)}=A-\sum_{k=0}^{p-1}B_kX_k^{(i+1)}C_k^{(i)}$$

and, for $j=1,\dots,p-1$,

$$A_j^{(i)}=A-\sum_{s=0}^{j-1}B_sX_s^{(i+1)}C_s^{(i)}-\sum_{k=j+1}^{p}B_kX_k^{(i)}C_k^{(i)}.$$

We denote the solution by $X_j^{(i+1)}$.

Step 2. Denote

$$\mathcal{C}_\ell^{(i)}=(C_0^{(i+1)},\dots,C_{\ell-1}^{(i+1)},C_{\ell+1}^{(i)},\dots,C_p^{(i)})$$

where $\mathcal{C}_0^{(i)}=(C_1^{(i)},\dots,C_p^{(i)})$ and $\mathcal{C}_p^{(i)}=(C_0^{(i+1)},\dots,C_{p-1}^{(i+1)})$.

For $j=0,\dots,p$, given $\mathcal{C}_j^{(i)}$ and $X_j^{(i+1)}$, find C_j that solves

$$\min_{C_j\in\mathbb{C}^{g_j\times n}}\left\|\Lambda_j^{(i)}-B_jX_j^{(i+1)}C_j\right\|^2, \tag{5}$$

where

$$\Lambda_0^{(i)}=A-\sum_{k=1}^{p}B_kX_k^{(i+1)}C_k^{(i)},\quad \Lambda_p^{(i)}=A-\sum_{k=0}^{p-1}B_kX_k^{(i+1)}C_k^{(i+1)}$$

and, for $j=1,\dots,p-1$,

$$\Lambda_j^{(i)}=A-\sum_{s=0}^{j-1}B_sX_s^{(i+1)}C_s^{(i+1)}-\sum_{k=j+1}^{p}B_kX_k^{(i+1)}C_k^{(i)}.$$

We denote the solution by $C_j^{(i+1)}$ and write

$$\varepsilon_{C,j}^{(i+1)}=\|\Lambda_j^{(i)}-B_jX_j^{(i+1)}C_j^{(i+1)}\|^2, \tag{6}$$

where $j=0,\dots,p$. Further, denote

$$\varepsilon^{(i+1)}=\min_{\substack{j=0,\dots,p\\ i=0,1,\dots}}\{\varepsilon_{C,j}^{(i+1)}\}. \tag{7}$$

If, for a given tolerance δ and $i = 1, 2, \ldots$,

$$|\varepsilon^{(i+1)} - \varepsilon^{(i)}| \leq \delta, \tag{8}$$

then the iterations stop. If not, then (4)– (8) are repeated for $i := i + 1$.

3.2 Particularities of solution device

The problem in (4) has been solved in [13] and here, we provide the solution for a completeness of the representation of the proposed method.

Let $M = U_M \Sigma_M V_M^*$ be the SVD of $M \in \mathbb{C}^{m\times n}$ where $U_M \in \mathbb{C}^{m\times m}, V_M \in \mathbb{C}^{n\times n}$ are unitary matrices, $\Sigma_M = \text{diag}(\sigma_1(M), \ldots, \sigma_{\min(m,n)}(M)) \in \mathbb{C}^{m\times n}$ is a generalized diagonal matrix, with the singular values $\sigma_1(M) \geq \sigma_2(M) \geq \ldots \geq 0$ on the main diagonal. The number of positive singular values of M is r, which is equal to the rank of M, denoted by rank M. Let $U_{M,m} = U_M = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m]$, $V_{M,n} = V_M = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n]$ be the column representations of U_M and V_M, respectively. Let

$$L_M = U_{M,r} U_{M,r}^* \in \mathbb{C}^{m\times m} \quad \text{and} \quad R_M = V_{M,r} V_{M,r}^* \in \mathbb{C}^{n\times n}. \tag{9}$$

For $\Sigma_{M,k} = \text{diag}(\sigma_1(M), \ldots, \sigma_k(M))$ and $k = 1, \ldots, \text{rank } M$, denote

$$M_{(k)} := U_{M,k} \Sigma_{M,k} V_{M,k}^* \in \mathbb{C}^{m\times n}, \tag{10}$$

i.e., $M_{(k)}$ is the k-truncated SVD of M. For $k > \text{rank } M$ we define $M_{(k)} := M$. For $1 \leq k < \text{rank } M$, the matrix $M_{(k)}$ is uniquely defined if and only if $\sigma_k(M) > \sigma_{k+1}(M)$.

Theorem 1 [13] *The minimal Frobenius norm solution to problem in (4) is given, for $i = 0, 1, \ldots$ and $j = 0, \ldots, p$, by*

$$X_j^{(i+1)} = B_j^\dagger (L_{B_j} A_j^{(i)} R_{C_j^{(i)}})_{(r_j)} (C_j^{(i)})^\dagger. \tag{11}$$

This solution is unique if and only if either

$$r_j \geq \text{rank } L_{B_j} A_j^{(i)} R_{C_j^{(i)}} \tag{12}$$

or

$$1 \le r_j < \operatorname{rank} L_{B_j} A_j^{(i)} R_{C_j^{(i)}} \quad \textit{and} \quad \sigma_{r_j}(L_{B_j} A_j^{(i)} R_{C_j^{(i)}}) > \sigma_{r_j+1}(L_{B_j} A_j^{(i)} R_{C_j^{(i)}}). \tag{13}$$

The minimal Frobenius norm solution of problem (5) is given, for $j = 0, \ldots, p$ and $i = 0, 1, \ldots$, by

$$C_j^{(i+1)} = (B_0 X_j^{(i+1)})^{\dagger} \Lambda_j^{(i)}. \tag{14}$$

Proof. The proof is given in [13]. ■

Remark 1 Because $(L_{B_j} A_j^{(i)} R_{C_j^{(i)}})_{(r_j)}$ is the r_j-truncated SVD (see (10)) then $X_j^{(i+1)}$ can be represented in the form $X_j^{(i+1)} = G_j^{(i+1)} H_j^{(i+1)}$ where $G_j^{(i+1)} \in \mathbb{C}^{s_j \times r_j}$ and $H_j^{(i+1)} \in \mathbb{C}^{r_j \times g_j}$. Recall that $r_j \le \min\{g_j, n\}$ and in practice, $r_j < \min\{g_j, n\}$. Matrix $G_j^{(i+1)}$ performs the transformation of matrix $C_j^{(i)}$ to a matrix of the smaller $r_j \times n$ size (i.e., $G_j^{(i+1)}$ is for a data compression [16] or feature selection [17]). Matrix $H_j^{(i+1)}$ reconstructs that smaller matrix to a $s_j \times n$ matrix which contributes to a reconstruction of the whole matrix A.

3.3 Associated error analysis

Let us write $X = (X_0, \ldots, X_p)$, $C = (C_0, \ldots, C_p)$,

$$F(X, C) = \left\| A - \sum_{j=0}^{p} B_j X_j C_j \right\|^2,$$

$X^{(i)} = (X_0^{(i)}, \ldots, X_p^{(i)})$, $C^{(i)} = (C_0^{(i)}, \ldots, C_p^{(i)})$ and $W^{(i)} = (X^{(i)}, C^{(i)})$ where $i = 0, 1, \ldots$. We also denote $W = (W_0, \ldots, W_p)$ where $W_j = (X_j, C_j)$ and $j = 0, \ldots, p$.

Theorem 2 *For each $j = 0, \ldots, p$, let $K_j^{(1)} \subset \mathbb{C}(s_j, g_j, r_j)$ and $K_j^{(2)} \subset \mathbb{C}^{g_j \times n}$ be compact sets. Let $X_j \in K_j^{(1)}$, $K^{(1)} = \{X_j \in K_j^{(1)}, \forall j = 0, \ldots, p \mid F(X, C) \le F(X^{(0)}, C)\}$, $C_j \in K_j^{(2)}$ and $K^{(2)} = \{C_j \in K_j^{(2)}, \forall j = 0, \ldots, p \mid F(X, C) \le F(X, C^{(0)})\}$. Let*

$$X_j^{(i+1)} = \arg \min_{X_j \in \mathbb{C}(s_j, g_j, r_j)} F(X_0^{(i+1)}, \ldots, X_{j-1}^{(i+1)}, X_j, X_{j+1}^{(i)} \ldots, X_p^{(i)}, C^{(i)}) \tag{15}$$

$$C_j^{(i+1)} = \arg \min_{C_j \in \mathbb{C}^{g_j \times n}} F(X^{(i+1)}, C_0^{(i+1)}, \ldots, C_{j-1}^{(i+1)}, C_j, C_{j+1}^{(i)} \ldots, C_p^{(i)}). \tag{16}$$

be the unique solutions to problems (4) and (5), respectively. Then any limit point of sequence $\{(X^{(i)}, C^{(i)})\}$ *is a coordinate-wise minimum point of* $F(X, C)$.

Proof. For $j = 0, \ldots, p$, let us denote $K_j = K_j^{(1)} \times K_j^{(2)}$ and $Z_j^{(i+1)} = (W_0^{(i+1)}, \ldots, W_j^{(i+1)}, W_{j+1}^{(i)}, \ldots, W_p^{(i)})$. Then (15), (16) imply

$$\begin{aligned}
F(W^{(i)}) &\geq \min_{W_0 \in K_0} F(W_0, W_1^{(i)}, \ldots, W_p^{(i)}) \\
&= F(W_0^{(i+1)}, W_1^{(i)}, \ldots, W_p^{(i)}) \\
&= F(Z_0^{(i+1)}) \\
&\geq \min_{W_1 \in K_1} F(W_0^{(i+1)}, W_1, W_2^{(i)}, \ldots, W_p^{(i)}) \\
&= F(W_0^{(i+1)}, W_1^{(i+1)}, W_2^{(i)}, \ldots, W_p^{(i)}) \\
&= F(Z_1^{(i+1)}) \\
&\;\;\vdots \\
&\geq \min_{W_p \in K_p} F(W_0^{(i+1)}, \ldots, W_p^{(i+1)}, W_p) \\
&= F(W^{(i+1)}) \qquad (17) \\
&= F(Z_p^{(i+1)}). \qquad (18)
\end{aligned}$$

Therefore, for any $W^{(0)} \in K = K^{(1)} \times K^{(2)}$,

$$F(W^{(0)}) \geq F(W^{(1)}) \geq \ldots \geq F(W^{(i)}) \geq \ldots \geq 0$$

(because $F(W) \geq 0$ by the definition). Thus, sequence $\{F(W^{(i)})\}$ is nondecreasing and bounded from below, i.e., $\{F(W^{(i)})\}$ is convergent. By the definition of set K, $\{W^{(i)}\} \subseteq K$. Since K is compact, there is a subsequence $\{W^{(i_k)}\}$ convergent to a point $\bar{W} = (\bar{W}_0, \ldots, \bar{W}_p) \in \mathbb{C}_{s,g,r}$ where $\mathbb{C}_{s,g,r} = \mathbb{C}(s, g, r) \times \mathbb{C}^{g \times n}$, $\mathbb{C}(s, g, r) = \mathbb{C}(s_0, g_0, r_0) \times \ldots \times \mathbb{C}(s_p, g_p, r_p)$ and $\mathbb{C}^{g \times n} = \mathbb{C}^{g_0 \times n} \times \ldots \times \mathbb{C}^{g_j \times n}$.

Consider now sequence $\{Z_0^{(i+1)}\}$ where $Z_0^{(i+1)} = (W_0^{(i+1)}, W_1^{(i)}, \ldots, W_p^{(i)})$. Since $\{F(Z_0^{(i+1)})\}$ is nondecreasing then $\{Z_0^{(i+1)}\} \subseteq K$ and there is a subsequence $\{Z_0^{(i_k)}\}$ convergent to a point $\bar{W}_0 = (V, \bar{W}_1, \ldots, \bar{W}_p) \in \mathbb{C}_{s,g,r}$. Further, because

$$F(W^{(i)}) \geq F(Z_0^{(i+1)})) \geq F(W^{(i+1)})$$

and, for any $W_0 \in \mathbb{C}_{s,g,r}$,

$$F(W_0^{(i_k+1)}, W_1^{(i_k)}, \ldots, W_p^{(i_k)}) \leq F(W_0, W_1^{(i_k)}, \ldots, W_p^{(i_k)}),$$

then by continuity of F as $k \to \infty$,

$$F(V, \bar{W}_1, \ldots, \bar{W}_p) \leq F(W_0, \bar{W}_1, \ldots, \bar{W}_p).$$

Since $F(W^{(i)})$ and $F(Z_0^{(i+1)})$ converge to the same value then

$$F(V, \bar{W}_1, \ldots, \bar{W}_p) = F(\bar{W}_0, \bar{W}_1, \ldots, \bar{W}_p).$$

Therefore,

$$F(\bar{W}_0, \bar{W}_1, \ldots, \bar{W}_p) \leq F(W_0, \bar{W}_1, \ldots, \bar{W}_p).$$

By uniqueness of the minimizer, the latter implies that $V = \bar{W}_0$. Based on the similar arguments,

$$F(\bar{W}_0, \bar{W}_1, \ldots, \bar{W}_p) \leq F(\bar{W}_0, W_1, \ldots, \bar{W}_p)$$

and in general, for $j = 0, \ldots, p$,

$$F(\bar{W}_0, \ldots, \bar{W}_j, \ldots, \bar{W}_p) \leq F(\bar{W}_0, \ldots, W_j, \ldots, \bar{W}_p).$$

The above proof is based on some known facts in [5]. ■

Corollary 1 *Let $\varepsilon^{(i)}$ be the error defined by (6). Then the increase in the number of iterations i implies the decrease in the associated error, i.e.,*

$$\varepsilon^{(i+1)} \leq \varepsilon^{(i)}. \tag{19}$$

Proof. The proof follows directly from (17). ■

Remark 2 A proof of convergence of the proposed method to a global minimum is hampered by special conditions that should be imposed on the problem and method under consideration. The

conditions are similar to those that are considered in Theorem 14.9 in [4] (p. 413) and they are not held here. In this regard, see also Section 6 below.

4 Particular case of problem in (3)

Here, we consider a particular case of the problem in (3) where matrices $C_0, \ldots, C_p$ are assumed to be known. An associated motivation has been considered in Section 1. The problem is as follows. Given matrices $A \in \mathbb{C}^{m\times n}$, $B_j \in \mathbb{C}^{m\times s_j}$ and $C_j \in \mathbb{C}^{g_j\times n}$, for $j = 0, \ldots, p$, find minimal Frobenius norm matrices $X_0, \ldots, X_p$, that solve

$$\min_{X_0\in\mathbb{C}(s_0,g_0,r_0),\ldots,X_p\in\mathbb{C}(s_p,g_p,r_p)} \left\| A - \sum_{j=0}^{p} B_j X_j C_j \right\|^2 . \tag{20}$$

4.1 Solution of problem in (20)

The device of the solution of the problem in (20) is represented by a greedy procedure which is a particular case of the the solution device in Section 3.1. Its i-th iterative loop, for $i = 0, 1, \ldots$, is as follows.

The i-th greedy loop, for $i = 0, 1, \ldots$.

For $j = 0, \ldots, p$, given $\mathcal{X}_j^{(i)}$, find X_j that solves

$$\min_{X_j\in\mathbb{C}(s_j,g_j,r_j)} \left\| A_j^{(i)} - B_j X_j C_j \right\|^2 , \tag{21}$$

where

$$A_0^{(i)} = A - \sum_{k=1}^{p} B_k X_k^{(i)} C_k, \quad A_p^{(i)} = A - \sum_{k=0}^{p-1} B_k X_k^{(i+1)} C_k$$

and, for $j = 1, \ldots, p-1$,

$$A_j^{(i)} = A - \sum_{s=0}^{j-1} B_s X_s^{(i+1)} C_s - \sum_{k=j+1}^{p} B_k X_k^{(i)} C_k.$$

We denote the solution by $X_j^{(i+1)}$ and write

$$\varepsilon_j^{(i+1)} = \|A_j^{(i)} - B_j X_j^{(i+1)} C_j\|^2, \tag{22}$$

where $j = 0, \ldots, p$.

Further, denote

$$\varepsilon^{(i+1)} = \min\{\varepsilon_0^{(i+1)}, \ldots, \varepsilon_p^{(i+1)}\}. \tag{23}$$

If, for a given tolerance δ and $i = 1, 2, \ldots$,

$$|\varepsilon^{(i+1)} - \varepsilon^{(i)}| \leq \delta, \tag{24}$$

then the iterations stop. If not, then (21)– (24) are repeated for $i := i + 1$.

For each $j = 0, \ldots, p$, the solution $X_j^{(i+1)}$ of the problem in (21) is given by (11)-(13) where $C_j^{(i)}$ should be replaced by C_j, for $i = 0, 1, \ldots$.

Then the solution of the problem in (20) is represented by $X_0^{(i+1)}, \ldots, X_p^{(i+1)}$.

Let us denote $F(X_0, \ldots, X_p) = \left\| A - \sum_{j=0}^{p} B_j X_j C_j \right\|^2$.

Fig. 1 Example 1: Selected images from set $X_1, ..., X_{50}$.

Theorem 3 *For each $j = 0, \ldots, p$, let $K_j \subset \mathbb{C}(s_j, g_j, r_j)$ be a compact set, $X_j \in K_j$ and $K = \{X_j \in K_j, \forall j = 0, \ldots, p \mid F(X) \leq F(X^{(0)})\}$. Let*

$$X_j^{(i+1)} = \arg \min_{X_j \in \mathbb{C}(s_j, g_j, r_j)} F(X_0^{(i+1)}, \ldots, X_{j-1}^{(i+1)}, X_j, X_{j+1}^{(i)} \ldots, X_p^{(i)}) \tag{25}$$

be the unique solution of problem (21). Then any limit point of sequence $\{(X_0^{(i)}, \ldots, X_p^{(i)})\} \subseteq K$ is a coordinate-wise minimum point of $F(X_0, \ldots, X_p)$.

Proof. The proof is similar to the proof of Theorem 2. ■

5 Numerical experiments

Fig. 2 Example 1: Selected noisy images from set $Z_1, ..., Z_{50}$.

Example 1 Here, we wish to numerically illustrate the methods considered in Sections 3 and 4 in a conspicuous way when $X_1, ..., X_{50}$ are represented by digital images[1]. For $k = 1, \ldots, 50$, each image X_k is given by matrix $X_k \in \mathbb{R}^{112 \times 88}$. Further, let $Z_1, ..., Z_{50}$ be observed noisy images such that

[1] The images are selected from the database of faces (AT&T) provided in `https://git-disl.github.io/GTDLBench/datasets/att_face_dataset/`

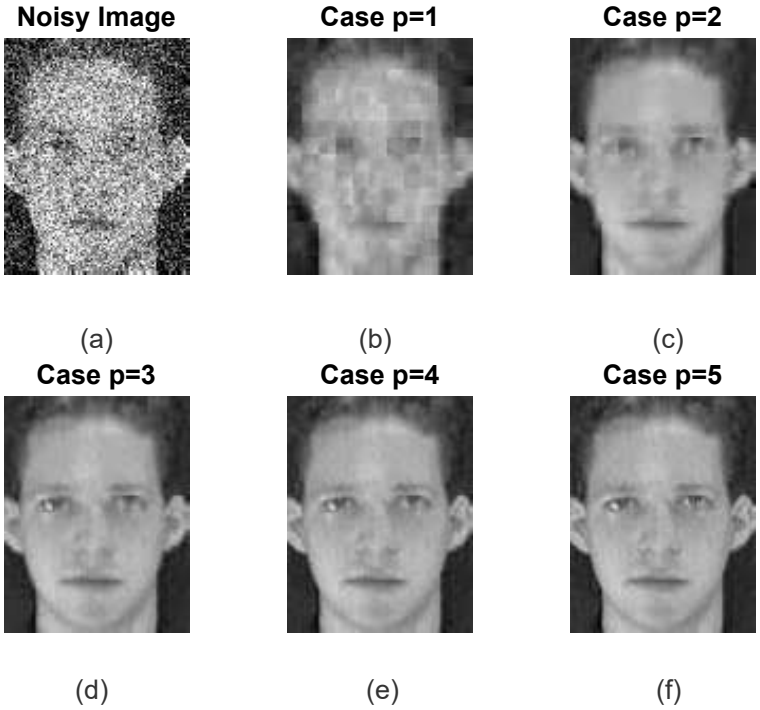


Fig. 3 Example 1: Estimates of a typical image by the method considered in Section 3, for $p = 1, \ldots, 5$.

	Degree	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$
Section 3	Error	359.03	8.01	6.89	4.97	3.97
Section 4	Error	433.02	15.55	10.71	7.03	6.10

Table 1 Errors versus degrees of methods in Sections 3 and 4, for 15 iterations

$Z_k = X_k + N_k$ where $N_k \in \mathbb{R}^{112\times 88}$ is a matrix with normally distributed random numbers. Matrix N_k simulates noise. Selected noisy images from set $Z_1, ..., Z_{50}$ are represented in Fig. 2.

Matrix A is constructed in the form $A = [a_1\ a_2\ ...\ a_{7700}] \in \mathbb{R}^{64\times 7700}$ where, for $i = 1, \ldots, 7700$, column a_i is a vectorization of an 8×8 block of matrix X_k. For $j = 0, \ldots, p$, B_j is the 64×64 identity matrix.

Recall, that in the method considered in Section 3, matrices $C_0^{(i)}, \ldots, C_p^{(i)}$ are determined optimally, for each $i = 1, 2, \ldots$, and in the method represented in Section 4, matrices $C_0, \ldots, C_p$ are given.

Further, in Step 1 of the method given in Section 3, $C_j^{(0)}$, for $j = 1, \ldots, p$, is a matrix with normally distributed random entries. For a consistency, in the method considered in Section 4, C_j, for $j = 1, \ldots, p$, is a matrix with normally distributed random entries as well.

We set $r_j = 16$, for $j = 0, \ldots, p$, and apply the methods developed in Sections 3 and 4 to the data described above. The obtained results are given in Table 1 and Fig. 3, for $p = 1, 2, 3, 4, 5$ and 15 iterations for each p.

In particular, Figs. 3 (b)–(f) represent the images reconstructed after the dimensionality reduction[2] by the method considered in Section 3, for $p = 1, \ldots, 5$.

We observe from Table 1 that the error associated with the method of Section 3 is less than the error of the method given in Section 4. We explain it by the optimal choice of matrices $C_0^{(i)}, \ldots, C_p^{(i)}$ in the method of Section 3.

Example 2 Here, we wish to illustrate the errors associated with the methods of Sections 3 and 4 in a way which different from that in Example 1.

Let $A \in \mathbb{R}^{50\times 60}$, $B_j \in \mathbb{R}^{50\times 70}$, $C_j \in \mathbb{R}^{80\times 60}$ and $C_j^{(0)} \in \mathbb{R}^{80\times 60}$ be matrices with normally distributed entries, and $r_j = 35$, for $j = 0, 1, 2$ (i.e., $p = 2$).

In Fig. 4, the errors associated with the methods proposed in Sections 3 and 4 versus the number of iterations are presented. The diagrams in Fig. 4 confirms the observation made in Example 1. Similar to Example 1, the error associated with the method of Section 3 is less than the error of the method given in Section 4.

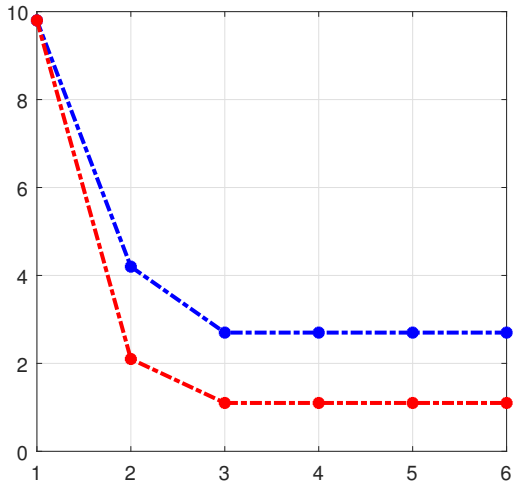


Fig. 4 Example 2: Diagrams of the errors of the methods in Section 3 (red) and 4 (blue) versus the number of iterations.

Example 3 In [19] (pp. 3101-3102), a solution of the particular case of problem (20) is given, for $B_j = I$ and $j = 0, \ldots, p$. The solution in [19] is based on an approach which is different from that in Section 4. Here, we wish to numerically compare the method proposed in Section 4 with the method considered in [19].

[2] In this regard, see Remark 1 in Section 3.2.

Let $A \in \mathbb{R}^{150\times 180}$ and $C_j \in \mathbb{R}^{60\times 180}$ be matrices with normally distributed entries, $p = 3$ and $r_j = 30$, for $j = 0, 1, 2, 3$. Since the method in [19] is applicable when B_j is the identity matrix, we set $B_j = I \in \mathbb{R}^{150\times 150}$. In Fig. 5, the errors associated with the proposed method and the method in [19], for $\lambda = 0.5$ (see [19]), versus numbers of iterations are represented.

It follows from Fig. 5 that the method proposed in Section 4 is more accurate than the method in [19]. We believe, this is, in particular, because in [19], the solution of the problem in (20), for $B_j = I$ and $j = 0, \ldots, p$, is represented as a solution of the problem which is an approximation of the problem of interest. This involves an additional error.

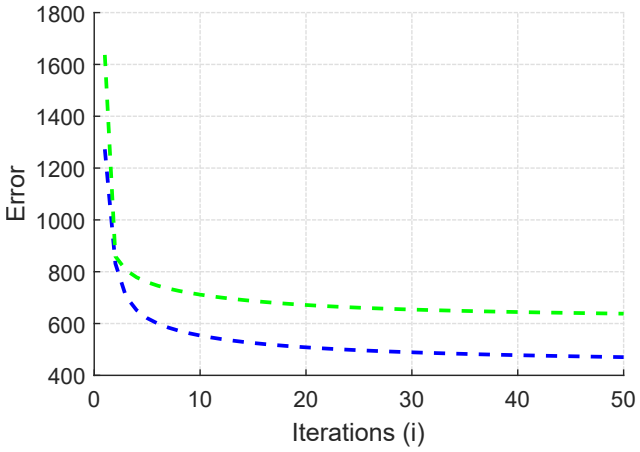


Fig. 5 Example 3: Diagrams of the errors associated with the method proposed in Section 4 (blue) and the method in [19] (green).

Example 4 Here, we consider r_j, p and matrices that are different from those in the above Examples 1-3. Let $A \in \mathbb{R}^{150\times 180}$, $B_j \in \mathbb{R}^{150\times 150}$, $C_j \in \mathbb{R}^{60\times 180}$ and $C_j^{(0)} \in \mathbb{R}^{60\times 180}$ be matrices with entries drawn from the standard uniform distribution on the open interval (0,1), and $r_j = 30$, for $j = 0, 1, 2, 3$ (i.e., $p = 3$).

In Fig. 6, the errors associated with the methods proposed in Sections 3, 4 and the method in [19], for $B_j = I$ and $\lambda = 0.5$ (see [19]), versus the number of iterations are presented. It follows from Fig. 6 that the error associated with the method considered in Section 3 is less than those for methods in Sections 4 and [19].

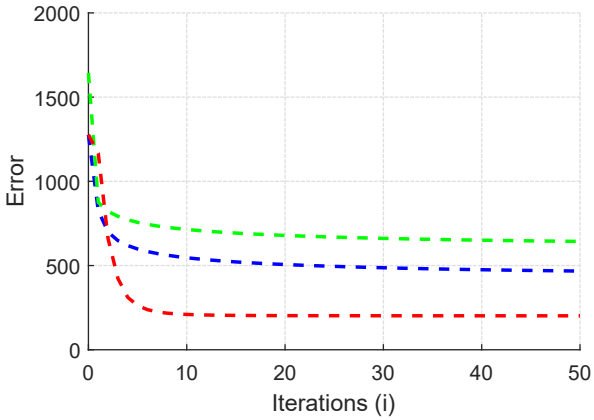


Fig. 6 Example 4: Diagrams of the errors of the methods considered in Sections 3 (red), 4 (blue), and [19] (green).

6 Discussion of the obtained results

We believe the problems given by (3) and (20), and their solutions represented in Sections 3 and 4 are important in their own right given the reasons mentioned in Section 1. Recall that the solution given in Section 3 is related, in particular, to the blind identification problem mentioned in Section 1. The solution provided in Section 4 represents, in particular, the model of a digital filter. A number of numerical simulations similar to those in Examples 1-4 shows that the error of the method in Section 3 is less than that in Section 4. We believe this is because of the larger number of matrices to optimize in the method of Section 3. We plan to corroborate this observation rigorously.

Open problems associated with the techniques provided in Sections 3 and 4 are as follows. First, this is a convergence proof to the global minimum for the methods given in Sections 3 and 4. Associated difficulties are mainly related to a large number of unknowns (which is a well recognized complication in numerical methods) and the special structures of the iteration procedures. In particular, as mentioned in Remark 2, the known related results (e.g., Theorem 14.9 in [4]) require special conditions that are not held for the proposed methods.

Second, it is an extension of the above error analysis to a theoretical comparison of the proposed methods in terms of the associated computational load and a faster convergence.

We will tackle those problems in succeeding work. In particular, we believe that the convergence to the global minimum and the theoretical justification of the advantages of the proposed methods can be developed on the basis of a combination of some ideas in [4] with the special structures of the proposed methods.

Declarations

Acknowledgements This work was financially supported in part by *Vicerrectoría de Investigación y Extensión* from *Instituto Tecnológico de Costa Rica* (Research #1440037).

Author Contributions All authors read and approved the final manuscript.

Data Availability The images are selected from the database of faces (AT&T) provided in `https://git-disl.github.io/GTDLBench/datasets/att_face_dataset/`

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Abramovich, Y.I., Johnson, B.A.: Expected likelihood support for blind SIMO channel identification. In: 2013 5th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), pp. 480–483 (2013)
2. Avdonin, S., Ivanov, S.: Sampling and interpolation problems for vector valued signals in the Paley - Wiener spaces. IEEE Transactions on Signal Processing **56**(11), 5435– 5441 (2008)
3. Bair, E., Hastie, T., Paul, D., Tibshirani, R.: Prediction by supervised principal components. Journal of the American Statistical Association **101**(473), 119 – 137 (2006)
4. Beck, A.: First-Order Methods in Optimization. SIAM (2017)
5. Bertsekas, D.: Nonlinear Programming, 3 edn. Athena Scientific (2016)
6. Boutsidis, C., Woodruff, D.P.: Optimal CUR matrix decompositions. SIAM Journal on Computing **46**(2), 543–589 (2017)
7. Brillinger, D.R.: Time Series: Data Analysis and Theory. SIAM, San Francisco (2001)
8. Bühlmann, P., van de Geer, S.: Statistics for High-Dimensional Data. Springer, New York (2011)
9. Chi, C.Y., Feng, C.C., Chen, C.H., Chen, C.Y.: Blind Equalization and System Identification: Batch Processing Algorithms, Performance and Applications. Springer (2005)
10. Chung, J., Chung, M.: Computing optimal low-rank matrix approximations for image processing. In: 2013 Asilomar Conference on Signals, Systems and Computers, pp. 670–674 (2013)
11. Eckart, C., Young, G.: The approximation of one matrix by another of lower rank. Psychometrika **1**(3), 211–218 (1936)
12. Fomin, V.N., Ruzhansky, M.V.: Abstract optimal linear filtering. SIAM Journal on Control and Optimization **38**(5), 1334 – 1352 (2000)
13. Friedland, S., Torokhti, A.: Generalized rank-constrained matrix approximations. SIAM Journal on Matrix Analysis and Applications **29**(2), 656–659 (2007)
14. Frieze, A., Kannan, R., Vempala, S.: Fast Monte-Carlo algorithms for finding low-rank approximations. Journal of the ACM **51**(6), 1025–1041 (2004)

15. Hua, Y., Liu, W.: Generalized Karhunen-Loeve transform. IEEE Signal Processing Letters **5**(6), 141–142 (1998)
16. Hua, Y., Nikpour, M., Stoica, P.: Optimal reduced-rank estimation and filtering. IEEE Transactions on Signal Processing **49**(3), 457–469 (2001)
17. Jolliffe, I.T., Cadima, J.: Principal component analysis: a review and recent developments. Phil. Trans. R. Soc. A **374**, 1 – 16 (2016)
18. Liu, X., Li, W., Wang, H.: Rank constrained matrix best approximation problem with respect to (skew) Hermitian matrices. Journal of Computational and Applied Mathematics **319**, 77 – 86 (2017)
19. Ma, H., Yang, Y.H., Chen, Y., Liu, K.J.R., Wang, Q.: Distributed state estimation with dimension reduction preprocessing. IEEE Transactions on Signal Processing **62**(12), 3098–3110 (2014)
20. Soto-Quiros, P., Torokhti, A.: Improvement in accuracy for dimensionality reduction and reconstruction of noisy signals. Part II: The case of signal samples. Signal Processing **154**, 272–279 (2019)
21. Stoica, P., Jansson, M.: MIMO system identification: State-space and subspace approximation versus transfer function and instrumental variables. IEEE Transactions on Signal Processing **48**(1), 3087– 3099 (2000)
22. Torokhti, A., Friedland, S.: Towards theory of generic Principal Component Analysis. Journal of Multivariate Analysis **100**(4), 661 – 669 (2009)
23. Torokhti, A., Soto-Quiros, P.: Generalized Brillinger-Like Transforms. IEEE Signal Processing Letters **23**(6), 843 – 847 (2016)
24. Wang, H.: Rank constrained matrix best approximation problem. Applied Mathematics Letters **50**, 98 – 104 (2015)
25. Werner, K., Jansson, M.: Reduced rank linear regression and weighted low rank approximation. IEEE Transactions on Signal Processing **54**(6), 2063 – 2075 (2006)

Received xxxx 20xx; revised xxxx 20xx.

Anexo 11

SeMA Journal

Fast Multiple Rank-Constrained Matrix Approximation

--Manuscript Draft--

Manuscript Number:	SEMJ-D-21-00138
Full Title:	Fast Multiple Rank-Constrained Matrix Approximation
Article Type:	Original Research
Funding Information:	
Abstract:	Our work addresses methods for a fast computation of the solution to the problem of a matrix approximation subject to multiple rank constraints. The problem arises in a number of applications and, for large matrices, its solution may require a quite long time. We provide techniques that allow us to accelerate the associated computation. The associated error analysis is given. Numerical examples illustrate advantages of the proposed methods.
Corresponding Author:	Pablo Soto-Quiros Instituto Tecnológico de Costa Rica Cartago, CARTAGO COSTA RICA
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	Instituto Tecnológico de Costa Rica
Corresponding Author's Secondary Institution:	
First Author:	Pablo Soto-Quiros
First Author Secondary Information:	
Order of Authors:	Pablo Soto-Quiros
	Jeffry Chavarría-Molina
	Juan José Fallas-Monge
	Anatoli Torokhti
Order of Authors Secondary Information:	
Author Comments:	Professor Carlos Vazquez-Cendon Editor-in-Chief SeMA Journal: Bulletin of the Spanish Society of Applied Mathematics Dear Professor Carlos Vazquez-Cendon, We would appreciate it very much if you could consider the manuscript entitled "Fast Multiple Rank-Constrained Matrix Approximation" by Jeffry Chavarria-Molina, Jose Fallas-Monge, Pablo Soto-Quiros and Anatoli Torokhti for a possible publication in the ‘SeMA Journal: Bulletin of the Spanish Society of Applied Mathematics’. In the manuscript, we propose and justify methods for a fast computation of the solution to the problem of a matrix approximation subject to multiple rank constraints. The problem arises in a number of applications and, for large matrices, its solution may require a quite long time. We provide techniques that allow us to accelerate the associated computation. The associated error analysis is given. Numerical examples illustrate advantages of the proposed methods. Thank you very much for your consideration. On behalf of the co-authors With best regards

	Dr Pablo Soto-Quiros Escuela de Matematica Instituto Tecnologico de Costa Rica Costa Rica jusoto@tec.ac.cr

Fast Multiple Rank-Constrained Matrix Approximation

Pablo Soto-Quiros[a], Jeffry Chavarría-Molina[a], Juan José Fallas-Monge[a], Anatoli Torokhti[b]

[a] *Escuela de Matemática, Instituto Tecnológico de Costa Rica, Cartago 30101, Costa Rica*

[b] *STEM, University of South Australia, SA 5095, Australia*

Abstract

Our work addresses methods for a fast computation of the solution to the problem of a matrix approximation subject to multiple rank constraints. The problem arises in a number of applications and, for large matrices, its solution may require a quite long time. We provide techniques that allow us to accelerate the associated computation. The associated error analysis is given. Numerical examples illustrate advantages of the proposed methods.

Keywords: Pseudo-inverse matrix, Rank-reduced matrix approximation.

1. Introduction

In practical computations we often encounter with a compromise between the cost of a numerical solution of the problem and an associated accuracy. In many practical cases, faster algorithms are preferable if they provide a still acceptable associated error which might be 'a little bit worse' than that by more costly and complicated methods.

In this paper, we consider a fast method for the solution of the following problem. Let $\mathbb{R}(m,n,r)$ be the set of all real $m \times n$ matrices of rank at most $r \leq \min\{m,n\}$ and $\|\cdot\|$ denote the Frobenius norm. Given $r_1,\ldots,r_p$, and matrices $A \in \mathbb{R}^{m\times n}$, $B_j \in \mathbb{R}^{m\times s_j}$ and $C_j \in \mathbb{R}^{g_j\times n}$, for $j = 1,\ldots,p$, find minimal Frobenius norm matrices $X_1,\ldots,X_p$ that solve

$$\min_{X_1\in\mathbb{R}(s_1,g_1,r_1),\ldots,X_p\in\mathbb{R}(s_p,g_p,r_p)} \left\| A - \sum_{j=1}^{p} B_j X_j C_j \right\|^2 . \tag{1}$$

Email addresses: jusoto@tec.ac.cr (Pablo Soto-Quiros), jchavarria@tec.ac.cr (Jeffry Chavarría-Molina), jfallas@tec.ac.cr (Juan José Fallas-Monge), anatoli.torokhti@unisa.edu.au (Anatoli Torokhti)

1.1. Motivation

It will be shown in Sections 2.1 and 2.2 that the solution of problem (1) is provided in terms of pseudo-inverse matrices and low-rank matrix approximations. The low-rank matrix approximation is represented by a truncated singular value decomposition (SVD). Computations of the pseudo-inverse matrices and SVD are highly time consuming, for large matrices. In particular, computing the SVD of an $m \times n$ matrix requires $O(\min\{m^2n, mn^2\})$ floating-points operations [1]. Taking into account a wide range of applications of the solution of the problem in (1) (see below), we believe, a development of the fast associated method is highly. This is a major motivation for the techniques considered in our paper.

The problem in (1) itself is motivated, in particular, by the tasks in system theory where rank restricted matrices $X_1, \ldots, X_p$ represent a model of a digital system, and input and output signals are available and considered as random vectors [2, 3, 4, 5, 6]. For the identification of the system, matrices $X_1, \ldots, X_p$ should determined in a best possible way. In reality, the random signal is represented by a sample matrix [7] where each column is a realization of the random vector. In (1), A and C_j can be treated, in particular, as sample matrices of a desired random output signal and random input signal, respectively, for the system with p inputs. Matrices $B_0, \ldots, B_p$ can be interpreted as weighted matrices. Based on a priori information, the weighting matrices are used to place a greater importance on some particular entries of the observed data as, for example, in [8, 9, 10].

1.2. Related work

Fast methods for computing the pseudo-inverse matrix: Courrieu [11] presents an algorithm to approximate the pseudo-inverse based on reverse order law and a full-rank Cholesky factorization of singular symmetric positive matrices. Brand [12] develops an identity for additive modifications of SVD, allowing for a fast solution for the pseudo-inverse of a submatrix of an orthogonal matrix. Telfer and Casasent [13] propose a method based on the matrix inversion lemma to approximate a robust pseudo-inverse. Benson and Frederickson [14] develop an efficient parallel implementation to compute the pseudo-inverse. Schulz [15] presents an iterative method to approximate the pseudo-inverse based on the classical Newton-Rapshon method. Other related works are presented in [16, 17, 18, 19]. Katsikis and Pappas [20] provide a very fast and reliable algorithm in order to compute the pseudoinverse of full-rank matrices, which does not use the SVD method. These authors use a special type of tensor product of two vectors, that is usually used in infinite dimensional Hilbert spaces. This type of tensor product gives conditions for the product of square matrices so that the conditions of the definition of the pseudo-inverse matrix are satisfied. Lu et al. [21] extended the method in [20] for computing the pseudo-inverse of rank-deficient matrices using Tikhonov's regularization (in this regard, see also, e.g., Theorem 4.3 in [22]).

Fast methods for computing the low-rank approximation: Deshpande and Vempala [23] combine adaptive sampling and volume sampling to obtain a low-rank approximation. Kannan, Mahoney and others [24, 25] present a set of algorithms which compute the low-rank approximation based on the Monte Carlo method. Nguyen, Do and Tran [26] propose a fast

algorithm that first spreads out the information in every column of a given matrix, then randomly selects some of the columns, and finally, generates the low-rank approximation based on the row space of the selected sets. Achlioptas and McSherry [27] introduce a simple technique for accelerating the computation of a low-rank approximation with a strong spectral structure, i.e., when the singular values of interest are significantly greater than those of a random matrix with similar size and entries. In [28], Halko, Martinsson and Tropp consider a randomized SVD method that extracts the column space from unilateral random projections. Fazel et al. [29], and Zhou and Tao [30], propose methods for computing the low-rank approximation based on a so called bilateral random projections. Comparing with the randomized SVD in [28], the methods in [29, 30] estimates both column and row spaces from bilateral random projections. Other related works are represented in [31, 32, 33].

Particular cases of problem (1): For the case $p = 1$, the problem in (1) has been studied in [34, 35, 36, 37] and, for $p = 1, s_1 = m, g_1 = n$, in [38]. The various applications of the particular case of the solution of problem (1), for $p = 1$, are considered, e.g., in [39, 40, 41, 42].

1.3. Contribution and novelty

The major contribution and novelty of this work are represented by the methods of the fast computation developed in Section 3.2, and by the associated error analysis given in Section 4. Some more related issues are also provided in Section 3.2.4.

2. Preliminaries

2.1. Solution device of problem in (1)

For $i = 1, \ldots$ and $j = 1, \ldots, p$, we denote

$$\mathbb{X}_j^{(i)} = (X_1^{(i+1)}, \ldots, X_{j-1}^{(i+1)}, X_{j+1}^{(i)}, \ldots, X_p^{(i)})$$

where

$$\mathbb{X}_1^{(i)} = (X_2^{(i)}, \ldots, X_p^{(i)}) \quad \text{and} \quad \mathbb{X}_p^{(i)} = (X_1^{(i+1)}, \ldots, X_{p-1}^{(i+1)}).$$

For $i = 1, \ldots,$ let us also write

$$A_1^{(i)} = A - \sum_{k=2}^{p} B_k X_k^{(i)} C_k, \quad A_p^{(i)} = A - \sum_{k=1}^{p-1} B_k X_k^{(i+1)} C_k$$

and, for $j = 1, \ldots, p-1$,

$$A_j^{(i)} = A - \sum_{s=1}^{j-1} B_s X_s^{(i+1)} C_s - \sum_{k=j+1}^{p} B_k X_k^{(i)} C_k.$$

The device of the solution of problem (1) is represented by a greedy procedure where the i-th loop, for $i = 1, \ldots,$ is as follows.

For $j = 1, \ldots, p$, given $\mathbb{X}_j^{(i)}$, find X_j that solves

$$\min_{X_j \in \mathbb{R}(s_j, g_j, r_j)} \left\| A_j^{(i)} - B_j X_j C_j \right\|^2. \tag{2}$$

We denote the solution by $X_j^{(i+1)}$ and write

$$\varepsilon^{(i+1)} = \min_{j=1,\ldots,p} \|A_j^{(i)} - B_j X_j^{(i+1)} C_j\|^2. \tag{3}$$

If, for a given tolerance δ and $i = 1, 2, \ldots,$

$$|\varepsilon^{(i+1)} - \varepsilon^{(i)}| \leq \delta, \tag{4}$$

then the iterations stop. If not, then (2)– (3) are repeated for $i := i + 1$.

The above procedure in (2)–(4) is a special case of the block coordinate descent type methods [43, 44, 45] which are extensions of the Gauss–Seidel method [1].

Remark 1. *The rank restrictions for matrices $X_0, \ldots, X_p$ in (1) are motivated, in particular, by requirements associated with a feature selection [46, 47], data compression [4, 48] and image processing [49]. Due to the rank restriction, each X_j is represented as $X_j = G_j H_j$ where $H_j \in \mathbb{R}^{r_j \times g_j}$ transforms matrix C_j to a matrix of the smaller $r_j \times n$ size and $G_j \in \mathbb{R}^{s_j \times r_j}$ reconstructs that smaller matrix to a $s_j \times n$ matrix (i.e., G_j contributes to a reconstruction of matrix A).*

2.2. Particularities for constructing the fast solution

Recall that the singular value decomposition (SVD) of $M \in \mathbb{R}^{m \times n}$ is given by $M = U_M \Sigma_M V_M^T$ where $U_M \in \mathbb{R}^{m \times m}$ and $V_M \in \mathbb{R}^{n \times n}$ are two orthogonal matrices, and $\Sigma_M = \text{diag}(\sigma_1(M), ..., \sigma_{\min(m,n)}(M)) \in \mathbb{R}^{m \times n}$ is a generalized diagonal matrix with singular values $\sigma_1(M) \geq \sigma_2(M) \geq ... \geq \sigma_{\min(m,n)}(M) \geq 0$ on the main diagonal.

For $r \leq \min\{m, n\}$, the r-truncated SVD is defined by

$$\lfloor M \rfloor_r = \sum_{i=1}^{r} \sigma_i(M) u_i v_i^T = U_{M,r} \Sigma_{M,r} V_{M,r}^T \in \mathbb{R}^{m \times n}, \tag{5}$$

where $U_{M,r} \in \mathbb{R}^{m \times r}$ and $V_{M,r} \in \mathbb{R}^{n \times r}$ are formed with the first r columns of U_M and V_M, respectively, and $\Sigma_{M,r} = \text{diag}(\sigma_1(M), \ldots, \sigma_r(M)) \in \mathbb{R}^{r \times r}$.

If $k = \text{rank}(M)$, then the pseudo invserse $M^\dagger$ of M is represented by [1]

$$M^\dagger = V_{M,k} \Sigma_{M,k}^{-1} U_{M,k}^T, \tag{6}$$

where $\Sigma_{M,k}^{-1} = \mathrm{diag}(\frac{1}{\sigma_1(M)}, \ldots, \frac{1}{\sigma_k(M)}) \in \mathbb{R}^{k\times k}$.

We denote by $P_{M,L} \in \mathbb{R}^{m\times m}$ and $P_{M,R} \in \mathbb{R}^{n\times n}$ the orthogonal projection of M on the range of M and M^T, respectively, i.e.

$$P_{M,L} = MM^\dagger = U_{M,k}U_{M,k}^T \quad \text{and} \quad P_{M,R} = M^\dagger M = V_{M,k}V_{M,k}^T. \tag{7}$$

Note that $P_{B,L}AP_{C,R} = BB^\dagger AC^\dagger C$.

For $i = 1, \ldots$ and $j = 1, \ldots, p$, denote

$$K_j^{(i)} = B_j B_j^\dagger A_j^{(i)} C_j^\dagger C_j \in \mathbb{R}^{m\times n}. \tag{8}$$

Then the result obtained in [35] can be represented as follows.

Theorem 1. [35] *The minimal Frobenius norm matrix $X_j^{(i+1)}$ that solves (2) is given by*

$$X_j^{(i+1)} = B_j^\dagger \lfloor K_j^{(i)} \rfloor_{r_j} C_j^\dagger. \tag{9}$$

This solution is unique if and only if either

$$r_j \geq \mathrm{rank}\ K_j^{(i)} \quad or \quad 1 \leq r_j \leq \mathrm{rank}\ K_j^{(i)} \quad and \quad \sigma_{r_j}(K_j^{(i)}) > \sigma_{r_j+1}(K_j^{(i)}).$$

3. Main result: Proposed approach to the fast solution of problem (1)

3.1. Fast computation of $B_j^\dagger$ and $C_j^\dagger$, for $j = 1, \ldots, p$

An analysis and comparison of a number of methods for the fast computation of the pseudo-inverse matrix, say, $B_j^\dagger$ provided, in particular, in [21] shows a preference of the method originated in [20] over other ones considered in [21]. This is because of its simplicity and high speed resulted, in particular, from of a 'replacement' of $B_j^\dagger$ computation by computation of $(B_j^T B_j)^{-1}$ or $(B_j B_j^T)^{-1}$ followed by the peculiar technique for the matric product in [20].

Recall that, for a full rank matrix $B_j \in \mathbb{R}^{m\times s_j}$, the pseudo-inverse is given by $B_j^\dagger = (B_j^T B_j)^{-1} B_j^T = B_j^T (B_j B_j^T)^{-1}$. As shown in [20], computation of $B^\dagger$ is much faster than that by SVD if the special type of the tensor product of vectors is applied to matrix product for computation of $B_j^\dagger$ represented by

$$B_j^\dagger = \begin{cases} (B_j^T B_j)^{-1} B_j^T, & \text{if } m \geq s_j \\ B_j^T (B_j B_j^T)^{-1}, & \text{if } m < s_j \end{cases}. \tag{10}$$

If $B_j \in \mathbb{R}^{m\times n}$ is rank deficient then

$$B_j^{\dagger} = \lim_{\alpha_j \to 0_+} (B_j^T B_j + \alpha_j I)^{-1} B_j^T = \lim_{\alpha_j \to 0_+} B_j^T (B_j B_j^T + \alpha_j I)^{-1}. \tag{11}$$

In this regard, see, for example, [50], p. 167, and for more details, [22]. Note if $x \in \mathbb{R}^m$ and $y \in \mathbb{R}^n$ are non zero vectors, then

$$x^T(\alpha_j I + B_j B_j^T)x = \alpha_j \|x\|_2^2 + \|B_j^T x\|_2^2 > 0$$

and

$$y^T(\alpha_j I + B_j^T B_j)y = \alpha_j \|y\|_2^2 + \|B_j y\|_2^2 > 0.$$

It is common to choose $\alpha \in (0, 1)$. Thus, $\alpha_j I + B_j B_j^T$ and $\alpha_j I + B_j^T B_j$ are positive definite, and as a result, are invertible. Therefore, based on (11), $B_j^{\dagger}$ is approximately represented by

$$\widetilde{B}_j^{\dagger} = \begin{cases} (B_j^T B_j + \alpha_j I)^{-1} B_j^T & \text{if } m \geq s_j \\ B_j^T (B_j B_j^T + \alpha_j I)^{-1} & \text{if } m < s_j \end{cases}, \tag{12}$$

where, for the fast computation, matrix products in (12) are fulfilled with the method [20] based the tensor product of vectors (in this regard, see [21]).

Equation (6) represents the standard procedure for $B_j^{\dagger}$ computation. By [1], the Golub-Reinsch SVD method and R-SVD method are most preferable to compute the SVD. In Table 1, for $m \geq s_j$, numbers of flops needed for computing $B^{\dagger}$ by the Golub-Reinsch SVD method, R-SVD method and the method based on (10) with the vector tensor product, are presented.

Table 1. Number of flops needed for computing $B_j^{\dagger}$

Method of computation of $B_j^{\dagger}$	Number of flops
Golub-Reinsch SVD	$4m^2 s_j + 8ms_j^2 + 9s_j^3 + (2k-1)(m+k)s_j$
R-SVD	$4m^2 s_j + 22s_j^3 + (2k-1)(m+k)s_j$
(10) with the vector tensor product	$4ms_j^2 + \frac{4}{3}s_j^3 + \frac{1}{2}s_j^2 - (ms_j + \frac{5}{6}s_j)$

It follows from the Table 1 that the number of flops needed for computation of $\widetilde{B}_j^{\dagger}$ by (12) is much less than those needed by the Golub-Reinsch SVD and R-SVD. Recall, on the basis of the results represented in [21], the method in (12) appears preferable than others considered in [21].

Fast estimate of $C_j^{\dagger} \in \mathbb{R}^{n\times g_j}$ is similar:

$$\widetilde{C}_j^{\dagger} = \begin{cases} (C_j^T C_j + \beta_j I)^{-1} C_j^T & \text{if } g_j \geq n \\ C_j^T (C_j C_j^T + \beta_j I)^{-1} & \text{if } g_j < n \end{cases} \tag{13}$$

where $\beta_j \in (0, 1)$.

By the above reasons, pseudo inverses $B_j^\dagger$ and $C_j^\dagger$ are evaluated by the procedures represented by (12) and (13), respectively, with the use of the tensor product of vectors [20, 21]. The implementations is presented in Algorithm 1.

Algorithm 1: Fast method for estimating pseudoinverse of Z

Input : $Z \in \mathbb{R}^{m\times n}$, $\alpha \in]0,1[$
Output: $Z_P \in \mathbb{R}^{n\times m}$

procedure: `fast_pinv(`Z`,`α`)`
  if Z is full-rank **then**
    if $m \geq n$, **then** $Z_P = (Z^T Z)^{-1} Z^T$
    if $m < n$, **then** $Z_P = Z^T (Z Z^T)^{-1}$
  else if Z is rank-deficient **then**
    if $m \geq n$, **then** $Z_P = (Z^T Z + \alpha I_n)^{-1} Z^T$
    if $m < n$, **then** $Z_P = Z^T (Z Z^T + \alpha I_m)^{-1}$
  return Z_p

3.2. Fast computation of $\lfloor K_j^{(i)} \rfloor_{r_j}$, for $j = 1, \ldots, p$

3.2.1. Device of the proposed fast computation of $\lfloor K_j^{(i)} \rfloor_{r_j}$

In (8), let us write matrix $K_j^{(i)}$ in terms of the SVD:

$$K_j^{(i)} = U_{K_j^{(i)}} \Sigma_{K_j^{(i)}} V_{K_j^{(i)}}^T. \tag{14}$$

By (5), in (9), matrix $\lfloor K_j^{(i)} \rfloor_{r_j}$, for $i = 1, 2, \ldots$ and $j = 1, \ldots, p$, is given by

$$\lfloor K_j^{(i)} \rfloor_{r_j} = U_{K_j^{(i)}, r_j} \Sigma_{K_j^{(i)}, r_j} V_{K_j^{(i)}, r_j}^T. \tag{15}$$

By (15), computation of $\lfloor K_j^{(i)} \rfloor_{r_j}$, for fixed i and j, consists of computation of the SVD of $K_j^{(i)}$ and matrix product in the LHS of (15). For the case when the SVD is computed by the Golub-Reinsch SVD and R-SVD methods [1], the associated numbers of flops to compute $\lfloor K_j^{(i)} \rfloor_{r_j}$, for $m \geq n$, are given in Table 2.

Table 2. Number of flops needed for computing $\lfloor K_j^{(i)} \rfloor_{r_j}$ by (15)

Method of SVD computation	Number of flops
Golub-Reinsch SVD	$4m^2n + 8mn^2 + 9n^3 + (2r_j - 1)(m + r_j)n$
R-SVD	$4m^2n + 22n^3 + (2r_j - 1)(m + r_j)n$

We propose a faster evaluation of $\lfloor K_j^{(i)} \rfloor_{r_j}$ as follows.

Let $A_j^{(i)} \in \mathbb{R}^{n\times r}$ be an arbitrary full-rank matrix. For clarity, let us now omit subscript j and superscript (i), i.e. let us write, for example, $A, K, Q, Y, \lfloor K \rfloor_r$ and r instead of $A_j^{(i)}, K_j^{(i)}, Q_j^{(i)}, Y_j^{(i)}, \lfloor K_j^{(i)} \rfloor_{r_j}$ and r_j, respectively, etc.

For $q = 1, 2, \ldots$, denote

$$Y = (K^T K)^q A. \tag{16}$$

and set

$$\bar{K} = KYY^{\dagger}. \tag{17}$$

For the *rank deficient* Y with rank $Y <$ rank K, set

$$\widetilde{K} = KY[Y^T Y + \gamma I]^{-1} Y^T, \tag{18}$$

where similar to α_j and β_j in (12) and (13), $\gamma := \gamma_j \in (0, 1)$ is a small parameter.

Further, let $Y = QR$ be the QR decomposition of Y. For the *full rank* Y, put

$$\widehat{K} = KQ_r Q_r^T, \tag{19}$$

where $Q_r = Q(:, 1:r) = [Q(:,1)\ Q(:,2)\ ...\ Q(:,r)]$ and $Q(:,i)$ denotes the ith column of Q.

In (18) and (19), matrix multiplications are performed with the tensor vector product [20]. In the following sections, we show that $\widetilde{K} := \widetilde{K}_j^{(i)}$ in (18) and $\widehat{K} := \widehat{K}_j^{(i)}$ in (19) represent fast estimates of $\lfloor K \rfloor_r := \lfloor K_j^{(i)} \rfloor_{r_j}$.

3.2.2. Motivation for the procedures in (18) and (19)

We still keep the subscript j and superscript (i) omitted. Let $A_1 \in \mathbb{R}^{n\times r}$ and $A_2 \in \mathbb{R}^{m\times r}$ be randomly chosen full-rank matrices, and

$$Y_1 = KA_1 \quad \text{and} \quad Y_2 = K^T A_2. \tag{20}$$

It has been shown in [51, 29] that $\breve{K}$ given by

$$\breve{K} = Y_1 (A_2^T Y_1)^{\dagger} Y_2^T \tag{21}$$

provides the fast estimate of $\lfloor K \rfloor_r := \lfloor K_j^{(i)} \rfloor_{r_j}$. The number of flops needed for computing $\breve{K}$ by (20) and (21) is given by $6mnr + 4mr^2 + 21r^3$ where $21r^3$ is the number of floops needed for computation of $(A_2^T Y_1)^{\dagger}$ by the Golub-Reinsch SVD. This is much less than that for computing $\lfloor K_j^{(i)} \rfloor_{r_j}$ by (15) (see Table 2 above).

At the same time, if the singular values of K decay gradually, then the accuracy of (21) may be lost [30, 28]. In [30], to avoid this bottleneck, for the case when $A_2^T Y_1$ is invertible, i.e., when

$$\breve{K} = Y_1 (A_2^T Y_1)^{-1} Y_2^T, \tag{22}$$

an additional factor $(KK^T)^q$ for Y_1 in (20) is included so that Y_1, for $q = 0, 1, ...$, becomes

$$Y_1 = \widetilde{Y}_1 = (KK^T)^q KA_1. \tag{23}$$

This is because $(KK^T)^q KA_1$ has the same singular vectors as KA_1, while the singular values decay more rapidly [28]. Besides, $\widetilde{Y}_1$ can be represented as $\widetilde{Y}_1 = K\widetilde{A}_1$ where $\widetilde{A}_1 = K^T(KK^T)^{q-1}KA_1$, i.e., $\widetilde{Y}_1$ is a particular case of Y_1 in (20).

3.2.3. Justification of the procedures in (18) and (19)

First in Theorem 2 that follows, we show that $\bar{K}$ in (17) is equivalent to $\breve{K}$ in (21). The latter means, in particular, that errors associated with (17) and (21) are the same. At the same time, as we show in Section 3.2.4 below, (18) which is an extension of (17) for the rank deficient Y, is a faster estimate of $\lfloor K \rfloor_r := \lfloor K_j^{(i)} \rfloor_{r_j}$ than that given by (21) (considered in [51, 29]). Note, that (21) is, of course, more general than (22) proposed in [30].

Theorem 2. *Let $A_1 \in \mathbb{R}^{n\times r}$ be an arbitrary full-rank matrix,*

$$A = A_1, \quad A_2 = K(K^TK)^{q-1}A_1, \quad Y_1 = KA_1 \quad \textit{and} \quad Y_2 = K^TA_2. \tag{24}$$

Then $\bar{K}$ in (17) and $\breve{K}$ in (21) are the same.

Proof. Note that $(KK^T)^qK = K(K^TK)^q$, and therefore, in (24), Y_1 can be written as $Y_1 = KY_2$. Also in (24), Y_2 is represented as $Y_2 = K^TK(K^TK)^{q-1}A_1 = (K^TK)^qA_1 = Y$ where Y is defined by (16). Furthermore, $\left(Y_2^TY_2\right)^\dagger Y_2^T = Y_2^\dagger$. Then

$$\begin{aligned}
\bar{K} &= KY_2Y_2^\dagger \qquad (25)\\
&= KY_2\left[Y_2^TY_2\right]^\dagger Y_2^T\\
&= KY_2\left[\left(\left(K^TK\right)^q A_1\right)^T Y_2\right]^\dagger Y_2^T\\
&= KY_2\left[\left(K(K^TK)^{q-1}A_1\right)^T KY_2\right]^\dagger Y_2^T\\
&= Y_1\left(A_2^TY_1\right)^\dagger Y_2^T = \breve{K}. \qquad (26)
\end{aligned}$$

□

Corollary 1. *If matrix $A_2^TY_1$ is invertible then, for A_2 given by (24), $\bar{K}$ in (17) and $\breve{K}$ in (22) are the same.*

Proof. The proof follows directly from (25)–(26). □

A justification of (18) is as follows. By (11), for the rank-deficient Y,

$$Y^\dagger = \lim_{\alpha_j\to 0_+}(Y^TY + \alpha_jI)^{-1}Y^T = \lim_{\alpha_j\to 0_+} Y^T(YY^T + \alpha_jI)^{-1}. \tag{27}$$

Therefore, similar to (12), the fast estimate of $\lfloor K \rfloor_r := \lfloor K_j^{(i)} \rfloor_{r_j}$ is chosen as that in (18). Further, note that $Y_2 = K^T A_2 = (K^T K)^q A_1$. Moreover, $\text{rank}((K^T K)^q) = \text{rank}(K)$. This is because if $K = U_K \Sigma_K V_K^T$ is the SVD of K, then $(K^T K)^q = V_K (\Sigma_K)^{2q} V_K^T$ is the SVD of $(K^T K)^q$, and therefore, K and $(K^T K)^q$ have the same number of positive singular values.

Further, if Y_2 is rank-deficient, then $\text{rank}(Y_2) < r$. By Corollary 2.5.10 in [52], $\text{rank}(Y_2) \leq \min\{\text{rank}((K^T K)^q), \text{rank}(A_1)\} = \min\{\text{rank}(K), r\}$, and therefore,

$$\text{rank}(Y_2) \leq \min\{\text{rank}(K), r\}.$$

If $\text{rank}(K) \leq r$, then $K \in \mathbb{R}_r^{m \times n}$, and therefore, $\bar{K} = K$. Otherwise, if $r < \text{rank}(K)$, then $\bar{K}$ can be computed by (17) using the method represented in Section 3.1. As a result, (18) follows.

Remark 2. *Our empirical study in Section 6 shows that the case rank$(Y_2) < r$ is unusual. For example, each numerical simulation in Section 6 generates a full-rank matrix Y_2.*

The method represented by (19) is justified on the basis of the following Theorem 3 which establishes a simplification of the estimate of K in (17) to that in (19), for the full-rank matrix Y.

Theorem 3. *Let Y, Y_2 and $\widetilde{Y}_1$ be given by (16), (20) and (23), respectively. Let $A_1 \in \mathbb{R}^{n \times r}$ be an arbitrary full-rank matrix, A_2 be given by (24), Y_2 be a full-rank matrix and $Y_2 = QR$ be the QR decomposition of Y_2. Then $\bar{K}$ in (17), $\widehat{K}$ in (19) and $\breve{K}$ in (21) are equivalent.*

Proof. By Theorem 5.2.2 in [1], $Y_2 = QR = Q_r R_r$ where $R_r \in \mathbb{R}^{r \times r}$ is an upper triangular matrix formed with the first r rows of R. Matrix R_r is nonsingular, because Y_2 is the full-rank matrix. Moreover, $\left[R_r^T R_r\right]^\dagger = \left[R_r^T R_r\right]^{-1}$ and $Q_r^T Q_r = I$. Then

$$\begin{aligned}
\widehat{K} &= K Q_r Q_r^T && (28)\\
&= K Q (R_r R_r^{-1})[(R_r^T)^{-1} R_r^T] Q_r^T \\
&= K Q_r R_r \left[R_r^T R_r\right]^{-1} R_r^T Q_r^T \\
&= K Q_r R_r \left[R_r^T Q_r^T Q_r R_r\right]^\dagger R_r^T Q_r^T \\
&= K Q_r R_r \left[(Q_r R_r)^T Q_r R_r\right]^\dagger (Q_r R_r)^T \\
&= K Y_2 \left[Y_2^T Y_2\right]^\dagger Y_2^T && (29)\\
&= K Y_2 Y_2^\dagger = \breve{K}, && (30)
\end{aligned}$$

where (30) follows from (29) by Theorem 2. □

Corollary 2. *Let the conditions of Theorem 3 be true and let matrix $A_2^T Y_1$ be invertible. Then $\widehat{K}$ in (19) and $\breve{K}$ in (22) are the same.*

Proof. The proof follows directly from (28)–(30). □

Remark 3. *If the methods in (18) and (19) are executed in floating-point arithmetic and q is a quite large number (i.e., $q = 4, 5, ...$), then an information associated with singular*

values smaller than $\mu^{1/(2q+1)}\|K\|$ is lost, where μ is the machine precision [28]. Therefore, it is recommended to take a small value for q, e.g., $q = 2, 3$.

3.2.4. Advantages: the procedures in (18) and (19) are faster than that in [30]

By Corollaries 1 and 3, the proposed estimates (18) and (19), for A_2 given by (24), are equivalent to the estimate considered in [30]. Therefore, the accuracy associated with the procedures in (18), (19) and [30] are the same. At the same time, unlike [30] the procedures provided by (18) and (19) are applicable for the case when matrix $A_2^T Y_1$ is non-invertible.

In Table 3, in terms of arbitrary q and large m, n, the numbers of flops needed for computing $\lfloor K_j^{(i)} \rfloor_{r_j}$ by the methods in (18), (19) and [30] are provided, for fixed i and j. Although the method in [30] is established for the case of an invertible matrix $A_2^T Y_1$ (i.e., when it is given by (22), (23)), in the bottom row of Table 3, we also consider the number of flops needed for an extension of the method in [30] to the case when $A_2^T Y_1$ is non-invertible (i.e., when it is given by (21), (23)).

In particular, the $r \times r$ matrix inverses in (18) and (22) take $2r^3/3$ flops. In (19), the QR decomposition of Y is computed by the Householder method [1]. It requires $4n^2r - 4nr^2 + 4r^3/3$ flops. In (21), the pseudo-inverse $(A_2^T Y_1)^\dagger$ is computed by the Golub-Reinsch SVD method [1] which takes $21r^3$ flops.

Table 3. Number of flops needed for computing $\lfloor K_j^{(i)} \rfloor_{r_j}$, for fixed i and j, and arbitrary m, n, r, q

Method	Number of flops
(18)	$2qmn^2 + 4mnr + 2n^2r + 2(m+n)r^2 + \frac{2}{3}r^3$
(19)	$2qmn^2 + 4mnr + 6n^2r - 4nr^2 + \frac{4}{3}r^3$
[30] by (22), (23)	$2qm^2n + 4mnr + 2m^2r + 4mr^2 + \frac{2}{3}r^3$
[30] by (21), (23)	$2qm^2n + 4mnr + 2m^2r + 4mr^2 + 21r^3$

In Section 6, diagrams of time needed by procedures (18), (19) and [30] versus different values of rank r_j of X_j are represented. The numerical simulations confirm the above analysis: the methods represented by (18), (19) are faster.

The associated pseudocode is represented by Algorithm 2.

3.3. Summary of fast computation of $X_j^{(i)}$ in (9)

The proposed fast computation of $X_j^{(i)}$ in (9) consists of a combination of the methods considered in Sections 3.1 and 3.2.

That is, in (9), matrices $B_j^\dagger$ and $C_j^\dagger$, for each$j = 1, \ldots, p$, are computed by the procedures

Algorithm 2: Fast method for estimating low-rank approximation of K

Input : $K \in \mathbb{R}^{m\times n}$, $r \leq \min\{m,n\}$, $q \in \{1,2,3,...\}$ and $\alpha \in]0,1[$
Output: $\widetilde{K}_r \in \mathbb{R}(m,n,r)$

procedure: `fast_low_rank(`K,r,q,α`)`
  Choose $A_1 \in \mathbb{R}^{m\times r}$ randomly, such that $\text{rank}(A_1) = r$
  $Y_2 = A_1$
  for $i = 1 : q$ **do**
    $Y_1 = KY_2$
    $Y_2 = K^T Y_1$
  if Y_2 is full-rank **then**
    $[Q, \sim] = \text{qr}(Y_2)$
    $Q_r = Q(:, 1:r)$
    $\widetilde{K}_r = KQ_rQ_r^T$
  else if $\text{rank}(K) < r$ **then**
    $\widetilde{K}_r = K$
  else
    $\widetilde{K}_r = KY_2Y_2^T(Y_2Y_2^T + \alpha I)^{-1}$
  return $\widetilde{K}_r$

represented by (11) and (12), with the use of the tensor product of vectors [20, 21].

Matrix $\lfloor K_j^{(i)} \rfloor_{r_j}$ is computed by (18) or (19) dependently on the rank of $X_j^{(i)}$.

The associated pseudocode is represented by Algorithm 3 where superscript (i) and subscript j are omitted, and $\widehat{X} := \widehat{X}_j^{(i)}$ is the proposed estimate of $X := X_j^{(i)} \in \mathbb{R}(s,g,r)$.

4. Error analysis

Here, we provide the error analysis associated with the fast computation of matrices $B_j^{(i)}$, $C_j^{(i)}$ and $X_j^{(i)}$. For convenience, we still omit superscript i and subscript j.

Definition 1. *Let M_1, M_2 be some matrices and $\alpha \in \mathbb{R}$, where M_2 depends on α, i.e., $M_2 = M_2(\alpha)$. We say $M_2(\alpha)$ is asymptotically close to M_1 if*

$$\lim_{\alpha\to 0}[M_1 - M_2(\alpha)] = 0. \tag{31}$$

Theorem 4. *Matrices $\widetilde{B_j}^\dagger$ and $\widetilde{C_j}^\dagger$ are asymptotically close to ${B_j}^\dagger$ and ${C_j}^\dagger$, respectively.*

Proof. The proof follows from (10)–(13). □

Algorithm 3: Fast computation of rank reduced $X := X_j^{(i)}$ in (9), for fixed i and j. Below, superscript (i) and subscript j are omitted

Input : $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{m\times s}$, $C \in \mathbb{R}^{g\times n}$ and $r \leq \min\{p, q\}$

Output: $\widehat{X} \in \mathbb{R}(s, g, r)$

/* Estimation of $B^\dagger$ with the method of Section 3.1 */

Choose $\alpha_1 \in]0, 1[$

$\widehat{B}^\dagger =$ `fast_pinv(`B`,`α_1`)`

/* Estimation of $C^\dagger$ with the method of Section 3.1 */

Choose $\alpha_2 \in]0, 1[$

$\widehat{C}^\dagger =$ `fast_pinv(`C`,`α_2`)`

/* Computing matrix $\widehat{K}$ */

$\widehat{K} = B\widehat{B}^\dagger A\widehat{C}^\dagger C$

/* Estimation of low-rank of $\widehat{K}$ with the method of Section 3.2 */

Choose $q \in \{1, 2, 3, ...\}$

Choose $\alpha_3 \in]0, 1[$

$\widetilde{K}_r =$ `fast_low_rank(`$\widehat{K}$`,`r`,`q`,`α_3`)`

/* Computing the estimate of X */

$\widehat{X} = \widehat{B}^\dagger \widetilde{K}_r \widehat{C}^\dagger$

Let us now consider the error estimate for $\bar{K}$. To this end, we need the following auxiliary results.

Proposition 1. ([52], Fact 9.9.10.) *Let M_1 and M_2 be two matrices. Then*

$$\|M_1 M_2\|^2 \leq \sigma_1^2(M_1)\|M_2\|^2, \tag{32}$$

where $\sigma_1(M_1)$ is the largest singular value of M_1.

Proposition 2. ([52], Fact 9.6.7.) *Let $M_1 \in \mathbb{R}^{m\times n}$ and $M_2 \in \mathbb{R}^{n\times g}$. If $m \geq n$, then*

$$\sigma_n^2(M_1)\|M_2\|^2 \leq \|M_1 M_2\|^2, \tag{33}$$

where $\sigma_n(M_1)$ is the smallest singular value of M_1.

Now we are in the position to provide Theorems 5–4 where the error estimates for $\bar{K}$ are given. Recall $Y \in \mathbb{R}^{n\times r}$ and $K \in \mathbb{R}^{m\times n}$ where $K := K_j^{(i)}$, and $K_j^{(i)}$ is given by (8).

Theorem 5. *If $k = rank(Y)$, then*

$$\|K - \bar{K}\|^2 \leq \sigma_1^2(K)(n - k). \tag{34}$$

Proof. We have

$$\begin{aligned} \|K - \bar{K}\|^2 &= \|K - KYY^\dagger\|^2 \\ &= \|K(I - YY^\dagger)\|^2 \\ &\leq \sigma_1^2(K)\|I - YY^\dagger\|^2. \end{aligned}$$

The latter is true by (32). Since $YY^\dagger$ is a symmetric projector then by Theorem E.3.2.0.1 in [53],

$$\|I - YY^\dagger\|^2 = \text{rank } (I - YY^\dagger). \tag{35}$$

Further, by Fact 6.1.6 in [52],

$$\text{rank } (I - YY^\dagger) = (n - k). \tag{36}$$

Thus, (34) follows. □

Theorem 6. *If $m \geq n$ and $k = rank(Y)$, then*

$$\sigma_n^2(K)(n - k) \leq \|K - \bar{K}\|^2. \tag{37}$$

Proof. We have

$$\|K - \tilde{K}\|^2 = \|K(I - YY^\dagger)\|^2,$$

where by (33),

$$\|K(I - YY^\dagger)\|^2 \geq \sigma_n^2(K)\|I - YY^\dagger\|^2.$$

Then on the basis of (35)–(36), inequality (37) follows. □

Corollary 3. *If $m \geq n$ and $k = rank(Y)$, then*

$$\sigma_n^2(K)(n - k) \leq \|K - \bar{K}\|^2 \leq \sigma_1^2(K)(n - k). \tag{38}$$

Proof. The proof immediately follows from (34) and (37). □

Denote $K_r := \lfloor K_j^{(i)} \rfloor_{r_j}$ (see (9)). We wish to estimate .. To this end, let us first consider the following Lemma 1.

Lemma 1. *Let $k = rank(Y)$ and $\mu_r = \sum_{i=r+1}^{\min\{m,n\}} \sigma_i^2(K)$. Then*

$$\left| \|K_r - \bar{K}\|^2 - \mu_r \right| \leq \sigma_1^2(K)(n - k) \tag{39}$$

Proof. Consider

$$\|K_r - \bar{K}\|^2 \leq \|K - K_r\|^2 + \|K - \bar{K}\|^2$$

and

$$\|K - K_r\|^2 \leq \|K - \bar{K}\|^2 + \|K_r - \bar{K}\|^2.$$

Then

$$\|K - K_r\|^2 - \|K - \bar{K}\|^2 \leq \|K_r - \bar{K}\|^2 \leq \|K - K_r\|^2 + \|K - \bar{K}\|^2,$$

and

$$-\|K - \bar{K}\|^2 \leq \|K_r - \bar{K}\|^2 - \|K - K_r\|^2 \leq \|K - \bar{K}\|^2$$

and

$$\left|\|K_r - \bar{K}\|^2 - \|K - K_r\|^2\right| \leq \|K - \bar{K}\|^2.$$

By [54], $\|K - K_r\|^2 = \mu_r$. Therefore, taking into account (34),

$$\left|\|K_r - \bar{K}\|^2 - \mu_r\right| \leq \sigma_1^2(K)(n-k).$$

□

Denote $\delta(\gamma) = \|\bar{K} - \widetilde{K}\|^2$ (see (18) in this regard).

Theorem 7. *Let $k = rank(Y)$, $\mu_r = \sum_{i=r+1}^{\min\{m,n\}} \sigma_i^2(K)$ and $\rho = \sigma_n^2(K)(n-k)$. Then*

$$\|K_r - \widetilde{K}\|^2 \leq \mu_r + \rho + \delta(\gamma), \tag{40}$$

where $\delta(\gamma) \to 0$ as $\gamma \to 0$. In other words,

$$\lim_{\gamma \to 0} \|K_r - \widetilde{K}\|^2 = \mu_r + \rho. \tag{41}$$

Proof. It follows from (39) that

$$-\rho \leq \|K_r - \bar{K}\|^2 - \mu_r \leq \rho,$$

i.e.,

$$\mu_r - \rho \leq \|K_r - \bar{K}\|^2 \leq \mu_r + \rho.$$

Then

$$\begin{aligned} \|K_r - \widetilde{K}\|^2 &\leq \|K_r - \bar{K}\|^2 + \|\bar{K} - \widetilde{K}\|^2 \\ &\leq \mu_r + \rho + \delta(\gamma). \end{aligned}$$

□

Theorem 8. *For fixed i and j, let*

$$X := X_j^{(i+1)} = B_j^{\dagger} \lfloor K_j^{(i)} \rfloor_{r_j} C_j^{\dagger}, \quad \widetilde{X} := \widetilde{X}_j^{(i+1)} = \widetilde{B}_j^{\dagger} \lfloor \widetilde{K}_j^{(i)} \rfloor_{r_j} \widetilde{C}_j^{\dagger}, \tag{42}$$

$$\widetilde{B}^\dagger = \widetilde{B}_j^\dagger, \quad \widetilde{C}^\dagger = \widetilde{C}_j^\dagger, \quad K_r := \lfloor K_j^{(i)} \rfloor_{r_j} \quad \textit{and} \quad \widetilde{K}_r := \lfloor \widetilde{K}_j^{(i)} \rfloor_{r_j}. \tag{43}$$

Then

$$\lim_{\alpha,\beta,\gamma\to 0} \|X - \widetilde{X}\| = (\mu_r + \rho)\|\widetilde{B}^\dagger\|\|\widetilde{C}^\dagger\|. \tag{44}$$

Proof. For the notation in (43), matrices X and $\widetilde{X}$ in (42) become $X = B^\dagger K_r C^\dagger$ and $\widetilde{X} = \widetilde{B}^\dagger \widetilde{K}_r \widetilde{C}^\dagger$. Let us also denote $\Delta B^\dagger = B^\dagger - \widetilde{B}^\dagger$, $\Delta C^\dagger = C^\dagger - \widetilde{C}^\dagger$ and $\Delta K_r = K_r - \widetilde{K}_r$. Then

$$\begin{aligned} X - \widetilde{X} &= B^\dagger K_r C^\dagger - \widetilde{B}^\dagger \widetilde{K}_r \widetilde{C}^\dagger \\ &= \Delta B^\dagger K_r C^\dagger + \widetilde{B}^\dagger K_r C^\dagger - \widetilde{B}^\dagger \widetilde{K}_r \widetilde{C}^\dagger \\ &= \Delta B^\dagger K_r C^\dagger + \widetilde{B}^\dagger \Delta K_r C^\dagger + \widetilde{B}^\dagger \widetilde{K}_r C^\dagger + \widetilde{B}^\dagger \widetilde{K}_r \widetilde{C}^\dagger \\ &= (\Delta B^\dagger K_r + \widetilde{B}^\dagger \Delta K_r)\Delta C^\dagger + (\Delta B^\dagger K_r + \widetilde{B}^\dagger \Delta K_r)\widetilde{C}^\dagger + \widetilde{B}^\dagger \widetilde{K}_r \Delta C^\dagger \\ &= (\Delta B^\dagger K_r + \widetilde{B}^\dagger \Delta K_r)\widetilde{C}^\dagger + \widetilde{B}^\dagger \widetilde{K}_r \Delta C^\dagger \end{aligned} \tag{45}$$

and

$$\|X - \widetilde{X}\| \le \Big(\|\Delta B^\dagger\|\|K_r\| + \|\widetilde{B}^\dagger\|\|\Delta K_r\|\Big)\|\widetilde{C}^\dagger\| + \|\widetilde{B}^\dagger\|\|\widetilde{K}_r\|\|\Delta C^\dagger\|. \tag{46}$$

Here, $\lim\limits_{\alpha\to 0} \|\Delta B^\dagger\| = 0$ and $\lim\limits_{\beta\to 0} \|\Delta C^\dagger\| = 0$. Therefore, by (41) and the squeeze theorem,

$$\lim_{\alpha,\beta,\gamma\to 0} \|X - \widetilde{X}\| = \|\widetilde{B}^\dagger\| \lim_{\gamma\to 0} \|\Delta K_r\|\|\widetilde{C}^\dagger\| = \|\widetilde{B}^\dagger\|(\mu_r + \rho)\|\widetilde{C}^\dagger\|.$$

□

Further, we also wish to characterize an error associated with the general procedure represented in Section 2.1. To this end, let us denote $F(X_0, \ldots, X_p) = \left\| A - \sum\limits_{j=0}^{p} B_j X_j C_j \right\|^2$.

Theorem 9. *For each $j = 1, \ldots, p$, let $K_j \subset \mathbb{R}(s_j, g_j, r_j)$ be a compact set, $X_j \in K_j$ and $K = \{X_j \in K_j, \forall j = 1, \ldots, p \mid F(X) \le F(X^{(1)})\}$. Let*

$$X_j^{(i+1)} = \arg \min_{X_j \in \mathbb{R}(s_j,g_j,r_j)} F(X_1^{(i+1)}, \ldots, X_{j-1}^{(i+1)}, X_j, X_{j+1}^{(i)} \ldots, X_p^{(i)}) \tag{47}$$

be the unique solution of problem (1). Then any limit point of sequence $\{(X_1^{(i)}, \ldots, X_p^{(i)})\} \subseteq K$ is a coordinate-wise minimum point of $F(X_1, \ldots, X_p)$.

Proof. The proof follows from [43]. □

Remark 4. *A proof of convergence of the procedure considered in Section 2.1 to a global minimum is hampered by special conditions that should be imposed on the problem and method under consideration. The conditions are similar to those that are provided in Theorem 14.9 in [55] (p. 413) and they are not held here.*

5. Application to the identification of a distributed signal processing system

Here, we consider an application of the proposed fast technique to the identification of a signal processing system that consists of a set of p spatially distributed sensors $S_1, ..., S_p$ and a fusion center F. Let Ω be a set of outcomes in probability space (Ω, Σ, μ) for which Σ is a σ–algebra of measurable subsets of Ω and $\mu : \Sigma \to [0, 1]$ is an associated probability measure. We assume without loss of generality that all random vectors are of the zero mean. Let us denote

$$\|\mathbf{x}\|_{\Omega}^2 = \int_{\Omega} \sum_{i=1}^{m} [\mathbf{x}_{(i)}(\omega)]^2 d\mu(\omega) \quad \text{and} \quad E_{xy} = \int_{\Omega} \mathbf{x}(\omega)[\mathbf{y}(\omega)]^T d\mu(\omega), \tag{48}$$

where E_{xy} is the covariance matrix formed from $\mathbf{x}$ and $\mathbf{y}$. The sensors make local noisy observations $\mathbf{y}_i \in L^2(\Omega, \mathbb{R}^{n_i})$, for $i = 1, .., p$, correlated with a signal of interest $\mathbf{x} \in L^2(\Omega, \mathbb{R}^m)$ and cannot communicate with each other. Each sensor $S_i : L^2(\Omega, \mathbb{R}^{n_i}) \to L^2(\Omega, \mathbb{R}^{r_i})$ transmits compressed information about its measurements $\mathbf{u}_i \in L^2(\Omega, \mathbb{R}^{r_i})$, $r_i \leq \min\{m, n_i\}$, to the fusion center $F : L^2(\Omega, \mathbb{R}^r) \to L^2(\Omega, \mathbb{R}^m)$, $r = r_1 + ... + r_p$, which should recover the original signal with a prescribed accuracy.

The model of the fusion center can be represented as $F = [F_1 \ ... \ F_p]$, where $F_i : L^2(\Omega, \mathbb{R}^{r_i}) \to L^2(\Omega, \mathbb{R}^m)$. Thus, the problem can be stated as follows: Find models of sensors $S_1, ..., S_p$ and fusion center $F = [F_1 \ ... \ F_p]$ that provide

$$\min_{\substack{F_1,...,F_p \\ S_1,...,S_p}} \left\| \mathbf{x} - \sum_{i=1}^{p} F_i S_i \mathbf{y}_i \right\|_{\Omega}^2. \tag{49}$$

Let $X_i = F_i S_i$, then problem (49) can be represented by

$$\min_{X_1,...,X_p} \left\| \mathbf{x} - \sum_{i=1}^{p} X_i \mathbf{y}_i \right\|_{\Omega}^2, \tag{50}$$

subject to

$$\operatorname{rank}(X_i) \leq r_i, \quad \text{for all} \quad i = 1, ..., p.$$

If $\mathbf{y} = (\mathbf{y}_1^T, \ldots, \mathbf{y}_p^T)^T$ and $X = [F_1 \ ... \ X_p]$, then (50) can be represented by

$$\min_{X_1,...,X_p} \|\mathbf{x} - X\mathbf{y}\|_{\Omega}^2, \tag{51}$$

where $\text{rank}(X_i) \leq r_i$, for all $i = 1, ..., p$. Note that

$$\begin{aligned} \|\mathbf{x} - X\mathbf{y}\|_{\Omega}^2 &= \text{tr}\left\{E_{xx} - E_{xy}X^T - XE_{yx} + XE_{yy}X^T\right\} \\ &= \text{tr}\left\{E_{xx} - E_{xy}E_{yy}^{\dagger}E_{yx}\right\} + \left\|E_{xy}(E_{yy}^{1/2})^{\dagger} - XE_{yy}^{1/2}\right\|^2 \end{aligned} \tag{52}$$

We define $A = E_{xy}(E_{yy}^{1/2})^{\dagger}$ and $C = E_{yy}^{1/2} = [C_1^T, \ldots, C_p^T]^T$. It follows from (52) that problem (51) is equivalent to

$$\min_{X_1,\ldots,X_p} \|A - XC\|^2 = \min_{X_1,\ldots,X_p} \left\|A - \sum_{i=1}^{p} X_i C_i\right\|^2, \tag{53}$$

subject to $\text{rank}(X_i) \leq r_i$, for all $i = 1, ..., p$. Thus, the problem in (49) is reduced to the problem in (53) which is a particular case of the problem in (1).

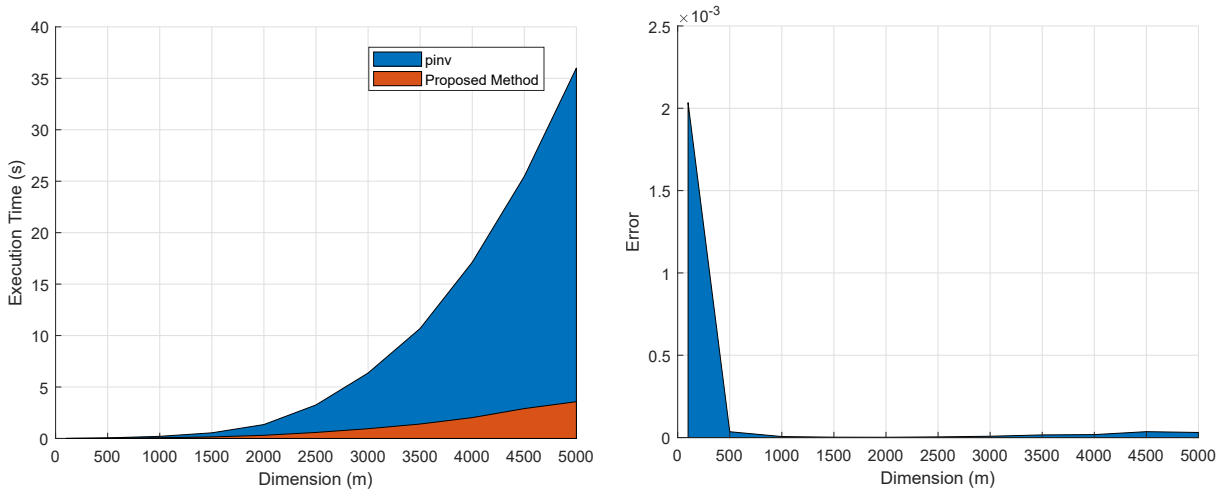
Therefore, the method considered above and summarized in Sections 3.3 can be applied to the problem in (49). Results of the related numerical simulation is provided in Example 5 of the following Section 6.

6. Empirical study

Example 1. *In this example, we compare an execution time needed for the pseudoinverse computation by the method considered in Section 3.1 and by the SVD. Let $D \in \mathbb{R}^{m\times m}$ be a matrix generated randomly from the standard uniform distribution such that rank$(D) = m/4$, i.e, D is rank-deficient. Diagrams of the execution time needed to compute $D^{\dagger}$ by the proposed method and by the SVD (using the* MATLAB *command* `pinv`*) versus dimension m, are presented in Fig. 1(a). In (12), we use $\alpha_1 = \alpha = 0.05$. It follows from Fig. 1(a) that, for the computation of $D^{\dagger}$, the execution time used by the procedure (12) is less than that needed by the SVD. At the same time, Fig. 1(b) shows that accuracy of both methods are almost the same. The difference is so small that it is invisible in Fig. 1(b). The Matlab code is provided in the Appendix 8.1.*

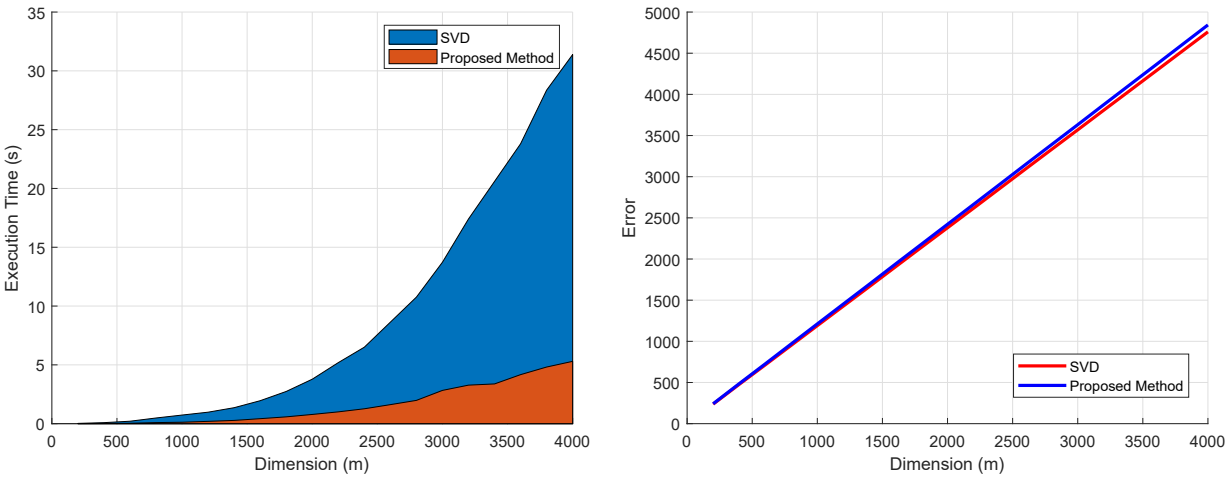
Example 2. *Here, for a computation of $\lfloor K_j^{(i)} \rfloor_{r_j}$ (see Section 3.2), we compare an execution time used by the method proposed in Section 3.2.1 and by the r_j-truncated SVD, for $j = 1$, used, in particular, in [34, 35, 36, 37]. To this end, we consider matrix $K = k_1 \in \mathbb{R}^{m\times n}$ which is generated randomly from the standard normal distribution, for $n = 2m$ and $r = r_1 = m/8$. Fig. 2(a) represents associated diagrams versus dimension m of K. It follows from Fig. 2(a) that execution time associated with the proposed method is less than that of the r-truncated SVD. At the same time, as it follows from Fig. 2(b), accuracy of both methods are almost the same. The Matlab code is provided in the Appendix 8.2.*

Example 3. *In this example, we compare the proposed fast method (see Sections 2.1 and 3.3) with the method described in Sections 2.1 and 2.2 when in (9), matrices $B_j^{\dagger}$, $\lfloor K_j^{(i)} \rfloor_{r_j}$ and $C_j^{\dagger}$, for $j = 1, \ldots, p$, are computed by the SVD which is similar to that in [34, 35, 36, 37]*



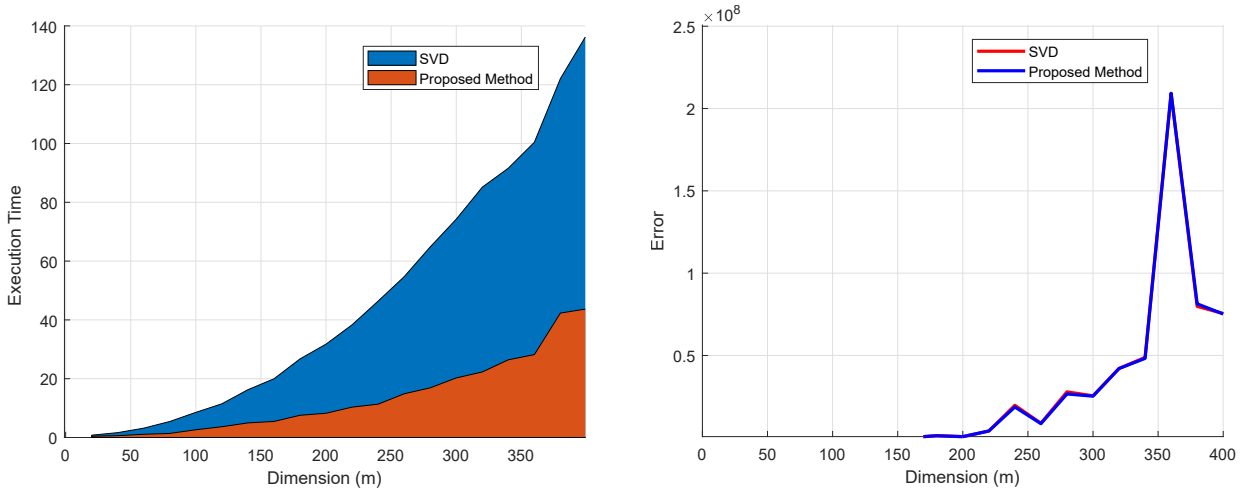
(a) (b)

Figure 1: (a) Execution time (in seconds) versus matrix dimension m used by the proposed method and by the MATLBB command `pinv` based on the SVD. (b) Errors associated with the proposed method and the SVD versus matrix dimension m.



(a) (b)

Figure 2: (a) Execution time (in seconds) versus matrix dimension m used by the proposed method and the r-truncated SVD. (b) MSE associated with proposed method and r-truncated SVD.



(a) (b)

Figure 3: (a) Execution time (in seconds) versus matrix dimension m used by the proposed method and the direct method (labeled by SVD). (b) MSE associated with proposed method and the direct method (labeled by SVD).

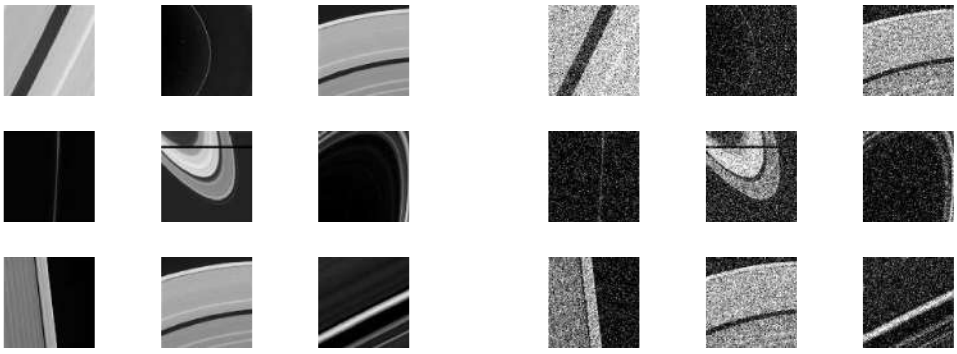
(we call it the direct method*). To this end, we consider full-rank matrices* $A \in \mathbb{R}^{m\times m}$, $B_j \in \mathbb{R}^{m\times m/2}$ *and* $C_j \in \mathbb{R}^{m/2\times m}$ *generated from a uniform distribution with zero-mean and standard deviation 1, for* $j = 1, 2$, *i.e., for* $p = 2$, *and* $m = 20, 40, 60, ..., 320, 360, 400$.

For $r = r_1 = r_2 = m/4$, *Fig. 3(a) represents typical diagrams of the execution time (in seconds) versus matrix dimension m used by the proposed fast method and the direct method. The execution time of the proposed method is significantly less than that by the direct method. At the same time, by Fig. 3(b), accuracy of both methods is almost the same. The Matlab code is provided in the Appendix 8.3.*

Example 4. *Here, we compare the performance of the proposed fast method, for* $p = 1$, *(see Sections 2.1 and 3.3) and the direct method (see Example 3) applied to compression, filtering and decompression of the noisy digital image represented by a matrix. Similar applications are considered in [41, 49, 56]. The noisy image* $\overline{P} \in \mathbb{R}^{90\times 90}$ *is estimated on the basis of a set of noisy training images (i.e., matrices)* $\mathcal{P} = \left\{P^{(1)}, ..., P^{(s)}\right\}$, *where* $P^{(j)} \in \mathbb{R}^{90\times 90}$, *for* $j = 1, ..., s$, *which are noisy versions of corresponding images in* $\mathcal{T} = \left\{T^{(1)}, ..., T^{(s)}\right\}$, *where* $T^{(j)} \in \mathbb{R}^{90\times 90}$, *for* $j = 1, ..., s$. *The set* $\mathcal{T}$ *consists of* $s = 2000$ *different satellite images of Saturn. See Fig. 4(a) where a sample of nine such images is given. Images in* $\mathcal{T}$ *are taken from the NASA Solar System Exploration Database [57].*

Each image $P^{(j)}$ *is represented as* $P^{(j)} = T^{(j)} + 0.2N^{(j)}$ *where* $N^{(j)} \in \mathbb{R}^{90\times 90}$ *is a matrix generated from a normal distribution with zero-mean and standard deviation 1. Matrices* $N^{(1)}, \ldots, N^{(s)}$ *simulate noise. See Fig. 4(b) where a sample of nine images from* $\mathcal{P}$ *is given. It is assumed that noisy image* $\overline{P}$ *does not necessarily belong to* $\mathcal{P}$, *but is "similar" to one of them, i.e., there is* $P^{(j)} \in \mathcal{P}$ *such that*

$$P^{(j)} \in \arg\min_{P^{(i)}\in\mathcal{P}} \|P^{(i)} - \overline{P}\|_{fr} \leq \delta,$$



(a) (b)

Figure 4: (a) Some randomly selected images of Saturn. (b) Noisy versions of images in (a).

for a given $\delta \geq 0$.

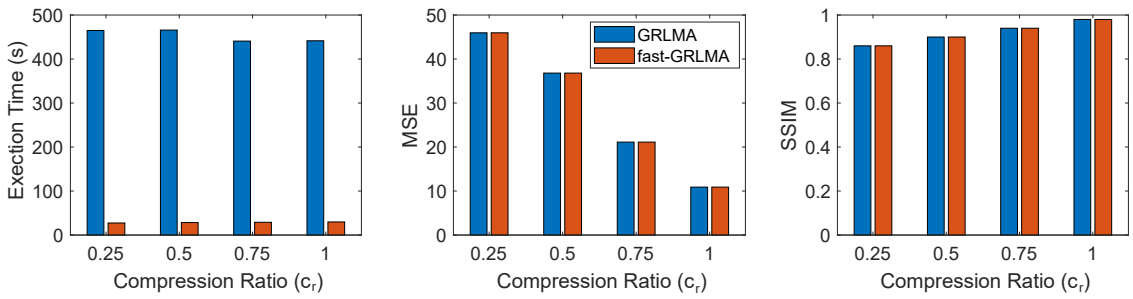
To formulate the problem under consideration in terms of the problem in (1), we vectorize each matrix $T^{(j)}$ *and* $P^{(j)}$*, i.e., convert the corresponding matrix into a column vector by stacking the columns of the matrix. Let* $vect : \mathbb{R}^{m\times n} \to \mathbb{R}^{mn}$ *be the vectorization transform. We write* $a_j = vec(T^{(j)}) \in \mathbb{R}^{8100}$ *and* $c_j = vec(P^{(j)}) \in \mathbb{R}^{8100}$*. Then in (1), for* $p = 1$*, matrices* A *and* C_1 *are given by* $A = [a_1\ a_2\ ...\ a_s] \in \mathbb{R}^{8100\times 2000}$ *and* $C_1 = [c_1\ c_2\ ...\ c_s] \in \mathbb{R}^{8100\times 2000}$*, respectively, and we set* $B_1 = I \in \mathbb{R}^{8100\times 8100}$ *. Therefore, now the problem is to find matrix* $\widehat{X} \in \mathbb{R}_r^{8100\times 8100}$ *such that*

$$\|A - \widehat{X}C_1\|^2 = \min_{X_1 \in \mathbb{R}_r^{8100\times 8100}} \|A - X_1C_1\|^2 . \tag{54}$$

Matrix $\widehat{X}$ *is determined by Algorithm 3. Further, in (54), the compression ratio is given by* $c_r = r_1/\min\{m, g_1\}$*. Fig. 5 represents the execution time, MSE and structural similarity index*[1] *(SSIM) between the proposed method (see Sections 2.1 and 3.3) and direct method (see Example 3), for compression ratio* $c_r \in \{0.25, 0.5, 0.75, 1\}$*, i.e., for* $r \in \{2025, 4050, 6075, 8100\}$*. In Fig. 6, we show estimates of* $\overline{P}$ *using both algorithms. The numerical results represented in Figures 5 and 6 clearly demonstrate the advantages of the proposed method. The proposed method is clearly faster than the direct method while the source image estimates by of both methods are the same.*

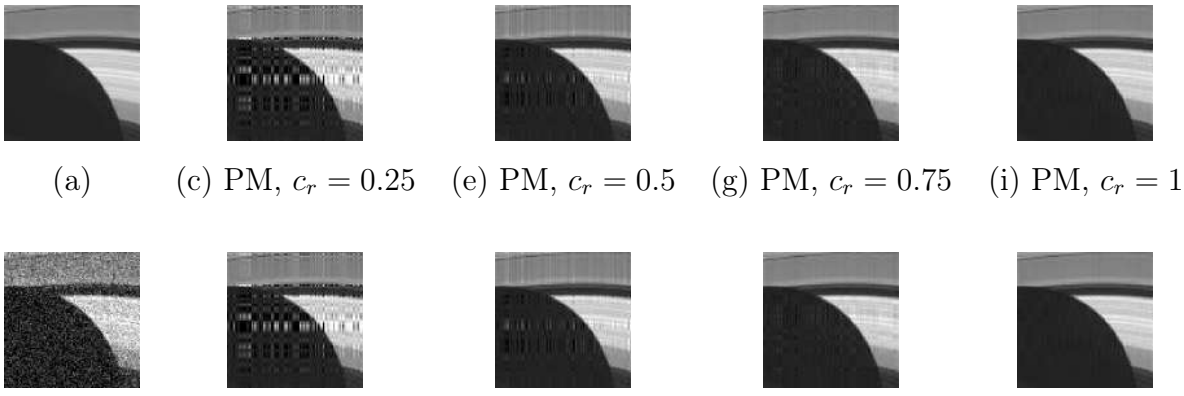
Example 5. *In this example, we represent an application of the proposed method to the identification a distributed signal processing system as described in Section 5. We consider the case as follows. The source signal* $\mathbf{x} \in L^2(\Omega, \mathbb{R}^m)$ *is a Gaussian random vector with zero-mean,* $S \in \mathbb{R}^{m\times s}$ *is a sample of signal* $\mathbf{x}$ *and* $E_{xx} = \frac{1}{s}SS^T$*. The observed vector* $\mathbf{y}_i \in L^2(\Omega, \mathbb{R}^m)$*, for* $p = 2$ *and* $i = 1, 2$*, is a noisy version of* $\mathbf{x}$ *given by* $\mathbf{y}_i = A_i\mathbf{x} + \mathbf{n}_i$*,*

[1]The structural similarity (SSIM) index is a method for predicting the perceived quality of digital images (and videos). The resultant SSIM index is a decimal value between -1 and 1, and value 1 is only reachable in the case of two identical sets of data. More details are available in [58].



(a) (b) (c)

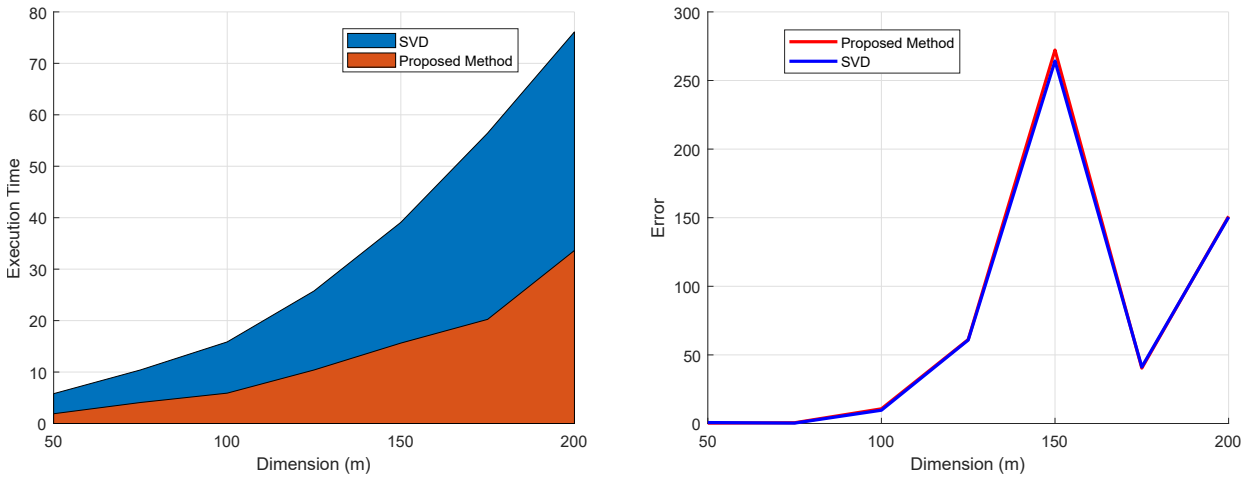
Figure 5: Illustration of advantages of the proposed method over direct method in terms of execution time (a), MSE (b) and structural similarity index (c), vs different compression ratios c_r.



(a) (c) PM, $c_r = 0.25$ (e) PM, $c_r = 0.5$ (g) PM, $c_r = 0.75$ (i) PM, $c_r = 1$

(b) (d) DM, $c_r = 0.25$ (f) DM, $c_r = 0.5$ (h) DM, $c_r = 0.75$ (j) DM, $c_r = 1$

Figure 6: (a) Source image; (b) Noisy observed image $\overline{P}$; (c)-(j) Estimates of the source image by the proposed method (PM) and the direct method (DM) for different compression ratios.



(a) (b)

Figure 7: (a) Execution time (in seconds) versus dimension m of X_1 and X_2 needed by the proposed method and the direct method (abbreviated by SVD) (b) MSE associated with the proposed method and the direct method versus dimension m of X_1 and X_2.

where $\mathbf{n}_i \in L^2(\Omega, \mathbb{R}^m)$ *is white noise with covariance matrix* $E_{n_i n_i} = \sigma_i^2 I_m$ *and* $A_i \in \mathbb{R}^{m \times m}$. *Signals* $\mathbf{x}$ *and* $\mathbf{n}_i$ *are assumed to be uncorrelated. Therefore,* $E_{xy_i} = E_{xx} A_i^T$, $E_{y_1 y_2} = A_1 E_{xx} A_2^T$ *and* $E_{\mathbf{y}_i \mathbf{y}_i} = A_i E_{xx} A_i^T + E_{n_i n_i}$, *for* $i = 1, 2$. *Here,* $m \in \{50, 75, ..., 175, 200\}$.

For $r_1 = r_2 = 3m/4$, *Fig. 7(a) represents diagrams of the execution time (in seconds) used to run the proposed method and direct method (abbreviated by SVD) versus signal* $\mathbf{x}$ *dimension* m. *In Fig. 7(b), diagrams of accuracy of both methods are represented. There is the significant reduction in execution time of the proposed method while the associated accuracy of both methods is very similar. The Matlab code is provided in the Appendix 8.4.*

7. Conclusion

In this paper, we proposed a fast method for the solution of the matrix approximation problem subject to multiple rank constraints. Our approach consists of the combination of the techniques of the fast pseudoinverse matrix computation and the fast low-rank matrix approximation. The first technique is based on the use of the vector tensor product. The second technique is based on the new extension of the method of the bilateral random projections. The theoretical study and numerical examples demonstrate the significant reduction of the execution time used by the proposed method in comparison with the methods based on the SVD computation. It is achieved, in particular, in the cost of 'a little bit' worse approximation accuracy which, in many practical cases, might be acceptable.

Acknowledgements

This work was financially supported by *Vicerrectoría de Investigación y Extensión* from *Instituto Tecnológico de Costa Rica* (Research Project #1440037).

8. Appendix

8.1. Example 1 - Execution time (in seconds) versus matrix dimension

```
% Example 1: Execution time in secons versus matrix dimension

clc; clear; close all

dimension=[100 500:500:5000];

time1=zeros(1,length(dimension)); time2=zeros(1,length(dimension));
   error=zeros(1,length(dimension));

k=0;
for m=dimension
   k=k+1;
   r=m/4;
   D=rand(m,r)*rand(r,m);
   %SVD Method
   tic
   P1=pinv(D);
   t1=toc;
   time1(k)=t1;

   %Proposed Method
   tic
   alpha=0.05;
   P2=(D'*D+alpha*eye(m))\D';
   t2=toc;
   time2(k)=t2;

   error(k)=norm(P1-P2,'fro');
end

%Diagrams
hold on
area(dimension,time1)
area(dimension,time2)
grid on
xlabel('Dimension (m)')
ylabel('Execution Time (s)')
```

```
figure
area(dimension,error)
grid on
xlabel('Dimension (m)')
ylabel('Error')
```

8.2. Example 2 - Execution time (in seconds) versus matrix dimension

```
% Example 2: Execution time in secons versus matrix dimension

clc; clear; close all

inter=200:200:4000;

time1=zeros(1,length(inter)); time2=zeros(1,length(inter));
error1=zeros(1,length(inter)); error2=zeros(1,length(inter));

k=0;
for m=inter
   k=k+1;
   n=2*m;
   K=randn(m,n);
   r=m/8;
   %SVD Method
   tic
   [U1,S1,V1]=svd(K);
   K1=U1(:,1:r)*S1(1:r,1:r)*(V1(:,1:r))';
   t1=toc;
   time1(k)=t1;

   %Proposed Method
   tic
   Y2=randn(n,r);
   for j=1:3
       Y1=K*Y2;
       Y2=K'*Y1;
   end
   [Qr,~]=qr(Y2,0);
   K2=K*(Qr*Qr');
   t2=toc;
   time2(k)=t2;

   error1(k)=norm(K-K1,'fro');
   error2(k)=norm(K-K2,'fro');
end

%Diagrams
```

```
figure
hold on
area(inter,time1)
area(inter,time2)
grid on
xlabel('Dimension (m)')
ylabel('Execution Time (s)')

figure
hold on
plot(inter,error1,'r')
plot(inter,error2,'b')
grid on
xlabel('Dimension (m)')
ylabel('Error')
```

8.3. Example 3 - Matlab Code

```
function example_general_1()
    clc; clear; close all

    intervalo=20:20:400;
    time1=zeros(length(10:10:50),1); time2=zeros(length(10:10:50),1);
    e1=zeros(length(10:10:50),1); e2=zeros(length(10:10:50),1);
    k=0;
    for m=intervalo
        k=k+1;
        r=floor(m/4); p=2;
        A=rand(m); B=rand(m,m/2,p); C=rand(m/2,m,p);

        tic
        [~,error1]=multiple_low_rank_opt1(A,B,C,r);
        t1=toc;
        time1(k)=t1;
        e1(k)=error1;
        tic
        [~,error2]=multiple_low_rank_opt2(A,B,C,r);
        t2=toc;
        time2(k)=t2;
        e2(k)=error2;
    end

    figure
    hold on
    area(intervalo, time1)
    area(intervalo, time2)

    figure
```

```
    hold on
    plot(intervalo, e1,'r')
    plot(intervalo, e2,'b')

end

function Y=pinv_new(X)
    [m,n]=size(X);
    if m>n
        Y=(X'*X)\X';
    else
        Y=X'/(X*X');
    end
end

function Y=GBRP(X,r)
   n=size(X,2);
   Y2=randn(n,r);
   for j=1:3
       Y1=X*Y2;
       Y2=X'*Y1;
   end
   [Qr,~]=qr(Y2,0);
   Y=X*(Qr*Qr');
end

function Y=low_rank_opt1(A,B,C,r)
    Bp=pinv(B); Cp=pinv(C);
    T=B*Bp*A*Cp*C;
    [U,S,V]=svd(T);
    Y=Bp*U(:,1:r)*S(1:r,1:r)*(V(:,1:r))'*Cp;
end

function Y=low_rank_opt2(A,B,C,r)
    Bp=pinv_new(B); Cp=pinv_new(C);
    T=B*Bp*A*Cp*C;
    Tr=GBRP(T,r);
    Y=Bp*Tr*Cp;
end

function y=obj_func(A,B,C,X)
    p=size(B,3);
    Aux=A;
    for i=1:p
        Aux=Aux-B(:,:,i)*X(:,:,i)*C(:,:,i);
    end
    y=norm(Aux,'fro')^2;
end
```

```
function [X,error1]=multiple_low_rank_opt1(A,B,C,r)
    iterMax=1000;
    m=size(A,1); p=size(B,3);

    X=zeros(m/2,m/2,p);
    for i=1:p
        X(:,:,i)=rand(m/2,r)*rand(r,m/2);
    end

    e_old=obj_func(A,B,C,X);

    for k=1:iterMax
        for i=1:p
            At=A;
            for j=1:p
                if j~=i
                    At=At-B(:,:,j)*X(:,:,j)*C(:,:,j);
                end
            end
            X(:,:,i)=low_rank_opt1(At,B(:,:,i),C(:,:,i),r);
        end
        e_new=obj_func(A,B,C,X);
        error1=abs(e_new-e_old);
        if error1<10^-5
            break
        end
        e_old=e_new;
    end
end

function [X,error1]=multiple_low_rank_opt2(A,B,C,r)
    iterMax=1000;
    m=size(A,1); p=size(B,3);

    X=zeros(m/2,m/2,p);
    for i=1:p
        X(:,:,i)=rand(m/2,r)*rand(r,m/2);
    end

    e_old=obj_func(A,B,C,X);

    for k=1:iterMax
        for i=1:p
            At=A;
            for j=1:p
                if j~=i
                    At=At-B(:,:,j)*X(:,:,j)*C(:,:,j);
```

```
                end
            end
            X(:,:,i)=low_rank_opt2(At,B(:,:,i),C(:,:,i),r);
        end
        e_new=obj_func(A,B,C,X);
        error1=abs(e_new-e_old);
        if error1<10^-5
            break
        end
        e_old=e_new;
    end
end
```

8.4. Example 5 - Matlab Code

```
function example_distributed_1()
    clc; clear; close all

    tam=50:25:200; p=2; s=1000;

    h=length(tam);

    time1=zeros(h,1); time2=zeros(h,1);
    e1=zeros(h,1); e2=zeros(h,1);

    k=0;

    for m=tam

        X=randn(m,s);
        Exx=(1/s)*(X*X');

        Enn=zeros(m,m,p); A=zeros(m,m,p);
        A_d=[]; Enn_d=[]; Exz=[];

        for i=1:p
            sigma=rand(1)/4;
            Enn(:,:,i)=sigma^2*eye(m);
            A(:,:,i)=rand(m);
            A_d=blkdiag(A_d,A(:,:,i));
            Enn_d=blkdiag(Enn_d,Enn(:,:,i));
            Exz=[Exz Exx*(A(:,:,i))'];
        end

        Ezz=A_d*kron(ones(p),Exx)*A_d'+Enn_d;

        M=Exz*pinv(sqrtm(Ezz));
```

```
        Ezz_sqrtm=sqrtm(Ezz);

        C=zeros(m,m*p,p);

        B=zeros(m,m,p);

        for i=1:p
           C(:,:,i)=Ezz_sqrtm((i-1)*m+1:m*i,:);
           B(:,:,i)=eye(m);
        end

        k=k+1; r=round(3*m/4);

        tic
        [~,error1]=multiple_low_rank_opt1(M,B,C,r);
        t1=toc;
        time1(k)=t1;
        e1(k)=error1;
        tic
        [~,error2]=multiple_low_rank_opt2(M,B,C,r);
        t2=toc;
        time2(k)=t2;
        e2(k)=error2;
    end

    figure
    hold on
    area(tam, time1)
    area(tam, time2)
    title('Time')

    figure
    hold on
    plot(tam, e1,'r')
    plot(tam, e2,'b')
    title('Error')

end

function Y=pinv_new(X)
    [m,n]=size(X);
    if m>n
        Y=(X'*X)\X';
    else
        Y=X'/(X*X');
    end
end
```

```
function Y=GBRP(X,r)
   n=size(X,2);
   Y2=randn(n,r);
   for j=1:3
       Y1=X*Y2;
       Y2=X'*Y1;
   end
   [Qr,~]=qr(Y2,0);
   Y=X*(Qr*Qr');
end

function Y=low_rank_opt1(A,B,C,r)
    Bp=pinv(B); Cp=pinv(C);
    T=B*Bp*A*Cp*C;
    [U,S,V]=svd(T);
    Y=Bp*U(:,1:r)*S(1:r,1:r)*(V(:,1:r))'*Cp;
end

function Y=low_rank_opt2(A,B,C,r)
    Bp=pinv_new(B); Cp=pinv_new(C);
    T=B*Bp*A*Cp*C;
    Tr=GBRP(T,r);
    Y=Bp*Tr*Cp;
end

function y=obj_func(A,B,C,X)
    p=size(B,3);
    Aux=A;
    for i=1:p
        Aux=Aux-B(:,:,i)*X(:,:,i)*C(:,:,i);
    end
    y=norm(Aux,'fro')^2;
end

function [X,error1]=multiple_low_rank_opt1(A,B,C,r)
    iterMax=1000;
    m=size(A,1); p=size(B,3);

    X=zeros(m,m,p);
    for i=1:p
        X(:,:,i)=rand(m,r)*rand(r,m);
    end

    e_old=obj_func(A,B,C,X);

    for k=1:iterMax
        for i=1:p
            At=A;
```

```
            for j=1:p
                if j~=i
                    At=At-B(:,:,j)*X(:,:,j)*C(:,:,j);
                end
            end
            X(:,:,i)=low_rank_opt1(At,B(:,:,i),C(:,:,i),r);
        end
        e_new=obj_func(A,B,C,X);
        error1=abs(e_new-e_old);
        if error1<10^-5
            break
        end
        e_old=e_new;
    end
end

function [X,error1]=multiple_low_rank_opt2(A,B,C,r)
    iterMax=1000;
    m=size(A,1); p=size(B,3);

    X=zeros(m,m,p);
    for i=1:p
        X(:,:,i)=rand(m,r)*rand(r,m);
    end

    e_old=obj_func(A,B,C,X);

    for k=1:iterMax
        for i=1:p
            At=A;
            for j=1:p
                if j~=i
                    At=At-B(:,:,j)*X(:,:,j)*C(:,:,j);
                end
            end
            X(:,:,i)=low_rank_opt2(At,B(:,:,i),C(:,:,i),r);
        end
        e_new=obj_func(A,B,C,X);
        error1=abs(e_new-e_old);
        if error1<10^-5
            break
        end
        e_old=e_new;
    end
end
```

References

[1] G. Golub, C. Van Loan, Matrix Computations, Matrix Computations, Johns Hopkins University Press, 2012.

[2] D. R. Brillinger, Time Series: Data Analysis and Theory, San Francisco, 2001.

[3] V. N. Fomin, M. V. Ruzhansky, Abstract optimal linear filtering, SIAM Journal on Control and Optimization 38 (5) (2000) 1334 – 1352.

[4] Y. Hua, M. Nikpour, P. Stoica, Optimal reduced-rank estimation and filtering, IEEE Transactions on Signal Processing 49 (3) (2001) 457–469.

[5] S. A. Billings, Nonlinear System Identification - Narmax Methods in the Time, Frequency, and Spatio-temporal Domains, John Wiley and Sons, Ltd., 2013.

[6] A. Torokhti, P. Soto-Quiros, Generalized Brillinger-Like Transforms, IEEE Signal Processing Letters 23 (6) (2016) 843 – 847.

[7] S. Avdonin, S. Ivanov, Sampling and interpolation problems for vector valued signals in the Paley - Wiener spaces, IEEE Transactions on Signal Processing 56 (11) (2008) 5435– 5441.

[8] K. Werner, M. Jansson, Reduced rank linear regression and weighted low rank approximation, IEEE Transactions on Signal Processing 54 (6) (2006) 2063 – 2075.

[9] P. Bühlmann, S. van de Geer, Statistics for High-Dimensional Data, Springer, New York, 2011.

[10] P. Stoica, M. Jansson, MIMO system identification: State-space and subspace approximation versus transfer function and instrumental variables, IEEE Transactions on Signal Processing 48 (1) (2000) 3087– 3099.

[11] P. Courrieu, Fast computation of Moore-Penrose inverse matrices, arXiv preprint arXiv:0804.4809 (2008).

[12] M. Brand, Fast low-rank modifications of the thin singular value decomposition, Linear algebra and its applications 415 (1) (2006) 20–30.

[13] B. Telfer, D. Casasent, Fast method for updating robust pseudoinverse and ho-kashyap associative processors, IEEE transactions on systems, man, and cybernetics 24 (9) (1994) 1387–1390.

[14] M. Benson, P. Frederickson, Fast parallel algorithms for the Moore-Penrose pseudoinverse, Tech. rep., Los Alamos National Lab., NM (USA); Lakehead Univ., Thunder Bay, Ontario (1986).

[15] G. Schulz, Iterative berechnung der reziproken matrix, Z. Angew. Math. Mech. 13 (1933) 57–59.

[16] S. Miljković, M. Miladinović, P. Stanimirović, I. Stojanović, Application of the pseudoinverse computation in reconstruction of blurred images, Filomat 26 (3) (2012) 453–465.

[17] H. Chen, Y. Wang, A family of higher-order convergent iterative methods for computing the Moore-Penrose inverse, Applied Mathematics and Computation 218 (8) (2011) 4012–4016.

[18] A. Ataei, Improved qrginv algorithm for computing Moore-Penrose inverse matrices, International Scholarly Research Notices 2014 (2014).

[19] S. Artidiello, A. Cordero, J. Torregrosa, M. Vassileva, Generalized inverses estimations by means of iterative methods with memory, Mathematics 8 (1) (2020) 2.

[20] V. Katsikis, D. Pappas, Fast computing of the Moore-Penrose inverse matrix, The Electronic Journal of Linear Algebra 17 (2008) 637–650.

[21] S. Lu, X. Wang, G. Zhang, X. Zhou, Effective algorithms of the Moore-Penrose inverse matrices for extreme learning machine, Intelligent Data Analysis 19 (4) (2015) 743–760.

[22] J. Barata, M. Hussein, The Moore–Penrose pseudoinverse: A tutorial review of the theory, Brazilian Journal of Physics 42 (1-2) (2012) 146–165.

[23] A. Deshpande, S. Vempala, Adaptive sampling and fast low-rank matrix approximation, in: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, Springer, 2006, pp. 292–303.

[24] P. Drineas, R. Kannan, M. Mahoney, Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix, SIAM Journal on computing 36 (1) (2006) 158–183.

[25] A. Frieze, R. Kannan, S. Vempala, Fast monte-carlo algorithms for finding low-rank approximations, Journal of the ACM (JACM) 51 (6) (2004) 1025–1041.

[26] N. Nguyen, T. Do, T. Tran, A fast and efficient algorithm for low-rank approximation of a matrix, in: Proceedings of the forty-first annual ACM symposium on Theory of computing, 2009, pp. 215–224.

[27] D. Achlioptas, F. McSherry, Fast computation of low-rank matrix approximations, Journal of the ACM (JACM) 54 (2) (2007) 9–es.

[28] N. Halko, P. Martinsson, J. Tropp, Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, SIAM review 53 (2) (2011) 217–288.

[29] M. Fazel, E. Candes, B. Recht, P. Parrilo, Compressed sensing and robust recovery of low rank matrices, in: 2008 42nd Asilomar Conference on Signals, Systems and Computers, IEEE, 2008, pp. 1043–1047.

[30] T. Zhou, D. Tao, Bilateral random projections, in: 2012 IEEE International Symposium on Information Theory Proceedings, IEEE, 2012, pp. 1286–1290.

[31] B. Li, Z. Yang, L. Zhi, Fast low rank approximation of a sylvester matrix by structured total least norm, J. JSSAC (Japan Society for Symbolic and Algebraic Computation) 11 (3) (2005) 4.

[32] M. Belabbas, P. Wolfe, Fast low-rank approximation for covariance matrices, in: 2007 2nd IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing, IEEE, 2007, pp. 293–296.

[33] G. Xie, K. Xie, J. Huang, X. Wang, Y. Chen, J. Wen, Fast low-rank matrix approximation with locality sensitive hashing for quick anomaly detection, in: IEEE INFOCOM 2017-IEEE Conference on Computer Communications, IEEE, 2017, pp. 1–9.

[34] D. Sondermann, Best approximate solutions to matrix equations under rank restrictions, Statistische Hefte 27 (1) (1986) 57–66.

[35] S. Friedland, A. Torokhti, Generalized rank-constrained matrix approximations, SIAM Journal on Matrix Analysis and Applications 29 (2) (2007) 656–659.

[36] A. Torokhti, S. Friedland, Towards theory of generic principal component analysis, Journal of Multivariate Analysis 100 (4) (2009) 661 – 669.

[37] H. Wang, Rank constrained matrix best approximation problem, Applied Mathematics Letters 50 (2015) 98–104.

[38] C. Eckart, G. Young, The approximation of one matrix by another of lower rank, Psychometrika 1 (3) (1936) 211–218.

[39] F. Fritzen, D. Ryckelynck, Machine Learning, Low-Rank Approximations and Reduced Order Modeling in Computational Mechanics, Mdpi AG, 2019.

[40] J. Chung, M. Chung, D. O'Leary, Optimal regularized low rank inverse approximation, Linear Algebra and its Applications 468 (2015) 260–269.

[41] J. Chung, M. Chung, An efficient approach for computing optimal low-rank regularized inverse matrices, Inverse Problems 30 (11) (2014) 114009.

[42] J. Chung, M. Chung, Optimal regularized inverse matrices for inverse problems, SIAM Journal on Matrix Analysis and Applications 38 (2) (2017) 458–477.

[43] D. Bertsekas, Nonlinear Programming, 3rd Edition, Athena Scientific, 2016.

[44] B. Chen, S. He, Z. Li, S. Zhang, Maximum block improvement and polynomial optimization, SIAM Journal on Optimization 22 (1) (2012) 87–107.

[45] G. Calafiore, Parallel block coordinate minimization with application to group regularized regression, Optimization and Engineering 17 (4) (2016) 941–964.

[46] E. Bair, T. Hastie, D. Paul, R. Tibshirani, Prediction by supervised principal components, Journal of the American Statistical Association 101 (473) (2006) 119 – 137.

[47] I. T. Jolliffe, J. Cadima, Principal component analysis: a review and recent developments, Phil. Trans. R. Soc. A 374 (2016) 1 – 16.

[48] Y. Hua, W. Liu, Generalized Karhunen-Loeve transform, IEEE Signal Processing Letters 5 (6) (1998) 141–142.

[49] J. Chung, M. Chung, Computing optimal low-rank matrix approximations for image processing, in: 2013 Asilomar Conference on Signals, Systems and Computers, 2013, pp. 670–674.

[50] D. G. Luenberger, Optimization by Vector Space Methods, Wliey, 1997.

[51] F. Woolfe, E. Liberty, V. Rokhlin, M. Tygert, A fast randomized algorithm for the approximation of matrices, Applied and Computational Harmonic Analysis 25 (3) (2008) 335 – 366.

[52] D. Bernstein, Matrix Mathematics: Theory, Facts, and Formulas (Second Edition), Princeton reference, Princeton University Press, 2009.

[53] J. Dattorro, Convex optimization † Euclidean distance geometry, Meboo Publishing USA, 2019.

[54] A. Torokhti, P. Howlett, Computational Methods for Modelling of Nonlinear Systems, Elsevier, Amsterdam, 2007.

[55] A. Beck, First-Order Methods in Optimization, SIAM, 2017.

[56] P. Soto-Quiros, A. Torokhti, Improvement in accuracy for dimensionality reduction and reconstruction of noisy signals. part ii: The case of signal samples, Signal Processing 154 (2019) 272–279.

[57] NASA, NASA solar system exploration database, `https://solarsystem.nasa.gov/raw-images/raw-image-viewer`, online; accessed 10 September 2020.

[58] S. S. Channappayya, A. C. Bovik, R. W. Heath, Rate bounds on SSIM index of quantized images, IEEE Transactions on Image Processing 17 (9) (2008) 1624–1639.

Anexo 12

Multinomial Karhunen-Loève Transform

Anatoli Torokhti, Phil Howlett, and Pablo Soto-Quiros

***Abstract*—The Karhunen-Loève transform (KLT) is represented by a matrix of fixed rank. Here, we consider an extension of the KLT to the case when it is represented by a collection of p special matrices where each matrix has a fixed rank. We call this a multinomial KLT. This extension is motivated by a number of particular applied problems closely related to the multinomial KLT. The multinomial KLT follows from a solution of the multi rank constrained error minimization problem. The problem is represented, in particular, by a matrix equation. The proposed approach is based on the reduction of that equation to an equation with a lower block triangular matrix. This allows us to represent the original problem as a set of p problems such that each of them is subject to only one rank constraint. The multinomial KLT follows from the solution of those equations.**

The Matlab code for the realization of the multinomial KLT is provided.

***Index Terms*—Karhunen-Loève transform, Principal Component Analysis.**

I. Introduction

THE Karhunen-Loève transform (KLT) is a powerful technique that has been successfully applied to the solution of a number of important signal processing problems. See, for example, [1, 2, 3, 4]. It is also known as Principal Component Analysis (PCA) [4, 5]. The KLT is represented by a matrix of fixed rank. Here, we consider an extension of the KLT to the case when it is represented by p matrices $F_1, \ldots, F_p$ where each matrix has a fixed rank. We call this the multinomial KLT. The multinomial KLT follows from the solution of the problem (2) given below. This problem has been a formidable challenge for a half of century (see [6] and also [7]) till our days. The problem attracts researchers because of its close relationship to topics not only in signal processing but also in bio-informatics, medicine, computer science and statistics.

To the best of our knowledge the exact solution of the problem in (2) was still unknown. The known approximate iterative solutions (see, for example, [8]–[22]) are mainly based on implementations of the block coordinate descent method and its modifications [23]–[26] or on a replacement of problem (2) by another problem which is an approximation of (2).

In this paper, the direct solution of the problem in (2) is given.

II. Statement of the Problem

Let Ω be a set of outcomes in probability space (Ω, Σ, μ) for which Σ is a σ–algebra of measurable subsets of Ω and $\mu : \Sigma \to [0, 1]$ is an associated probability measure. Without loss of generality, we assume that all random vectors are of the zero mean. Let us denote $\mathbf{x} \in L^2(\Omega, \mathbb{R}^m)$, $\mathbf{x} = [\mathbf{x}_{(1)}, \ldots, \mathbf{x}_{(m)}]^T$, $\mathbf{x}_{(i)} \in L^2(\Omega, \mathbb{R})$, for $i = 1, \ldots, m$, $\mathbf{y}_j \in L^2(\Omega, \mathbb{R}^{n_j})$, for $j = 1, \ldots, p$, and $n_1 + \ldots + n_p = n$. We also write

$$\|\mathbf{x}\|_\Omega^2 = \int_\Omega \sum_{i=1}^{m} [\mathbf{x}_{(i)}(\omega)]^2 d\mu(\omega) \tag{1}$$

and $E_{xy} = \int_\Omega \mathbf{x}(\omega)[\mathbf{y}(\omega)]^T d\mu(\omega)$ where E_{xy} is the covariance matrix formed from $\mathbf{x}$ and $\mathbf{y}$.

Further, let $\mathbb{R}_r^{m\times n}$ be the set of all real $m \times n$ matrices of rank at most $r \le \min\{m, n\}$. Given $r_1, \ldots, r_p$, find minimal Frobenius norm matrices $F_1, \ldots, F_p$ that solve

$$\min_{F_1 \in \mathbb{R}_{r_1}^{s_1\times g_1}, \ldots, F_p \in \mathbb{R}_{r_p}^{s_p\times g_p}} \left\| \mathbf{x} - \sum_{j=1}^{p} F_j \mathbf{y}_j \right\|_\Omega^2. \tag{2}$$

Note that for $p = 1$, (2) represents the classical problem studied, e.g., in [1, 2, 3, 5].

III. Preliminaries

Let us write $F = [F_1, \ldots, F_p]$, $\mathbf{y} = [\mathbf{y}_1^T, \ldots, \mathbf{y}_p^T]^T \in L^2(\Omega, \mathbb{R}^n)$ and $\|\cdot\|$ for the Frobenius norm. Then (see, for example, [2])

$$\begin{aligned} \left\| \mathbf{x} - \sum_{j=1}^{p} F_j \mathbf{y}_j \right\|_\Omega^2 &= \|\mathbf{x} - F\mathbf{y}\|_\Omega^2 \\ &= \|E_{xx}^{1/2}\|^2 - \|E_{xy}(E_{yy}^{1/2})^\dagger\|^2 \\ &\quad + \|E_{xy}(E_{yy}^{1/2})^\dagger - FE_{yy}^{1/2}\|^2. \end{aligned} \tag{3}$$

As a result, (2) and (3) imply that $F_1, \ldots, F_p$ solve (2) if F solves the equation

$$E_{xy}(E_{yy}^{1/2})^\dagger - FE_{yy}^{1/2} = \mathbb{O} \tag{4}$$

where $\mathbb{O}$ is the zero matrix, subject to

$$\text{rank } F_1 = r_1, \quad \ldots, \quad \text{rank } F_p = r_p. \tag{5}$$

Our device for the solution of the problem in (4)–(5) is as follows. First, we represent equation (4) in blocked matrices. It is given in the form (7) below. This allows us to apply the basic idea of our approach, a reduction of equation (7) to the equation with a lower block-triangular matrix. It is done in Section III-B. Then the solution is formed from p solutions of p equations with only one constraint. For clarity, in the following Section III-A, we first represent the reduction for the case $p = 2$ which is, in fact, a generic basis for the proposed procedure.

P. Howlett and A. Torokhti are with the STEM discipline, University of South Australia, Mawson Lakes, SA 5076, Australia, e-mail: P.Howlet@unisa.edu.au, e-mail: anatoli.torokhti@unisa.edu.au.

P. Soto-Quiros is with Department of Mathematics, Institute of Technology of Costa Rica, Cartago 30101, Costa Rica, e-mail: jusoto@tec.ac.cr

Manuscript received ... .

Let us now denote

$$G = E_{xy}(E_{yy}^{1/2})^{\dagger} \quad \text{and} \quad H = E_{yy}^{1/2},$$

and represent matrices A and C in block forms as follows:

$$G = [G_1 \dots G_p] \quad \text{and} \quad H = \begin{bmatrix} H_{11} & \dots & H_{1p} \\ \vdots & \vdots & \vdots \\ H_{p1} & \dots & H_{pp} \end{bmatrix}, \quad (6)$$

respectively, where $G_j =\in \mathbb{R}^{m\times n_j}$, for $j = 1,\dots,p$, and $H_{ij} \in \mathbb{R}^{q_j\times n_j}$, for $j,i = 1,\dots,p$. Then (4) implies

$$[F_1,\dots,F_p]\begin{bmatrix} H_{11} & \dots & H_{1p} \\ \vdots & \vdots & \vdots \\ H_{p1} & \dots & H_{pp} \end{bmatrix} = [G_1 \dots G_p]. \quad (7)$$

A. Generic Basis for the Reduction of Equation (4)

For a matrix M, the Moore-Penrose pseudo-inverse of M is denoted by $M^{\dagger}$.

For $p = 2$, equation (7) takes the form

$$[F_1 \;\; F_2]\begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} = [G_1 \;\; G_2]. \quad (8)$$

Let $\mathcal{N}(M)$ be the null space of M. We denote

$$H_{22}^{(1)} = H_{22} - H_{21}H_{11}^{\dagger}H_{12}, \quad G_2^{(1)} = G_2 - G_1H_{11}^{\dagger}H_{12}. \quad (9)$$

Lemma 1: Let F_1 and F_2 satisfy equation (8). Let

$$\mathcal{N}(H_{11}^T) \subseteq \mathcal{N}(H_{12}^T). \quad (10)$$

Then (8) decomposes into the equations

$$F_1H_{11} = G_1 - F_2H_{21} \quad \text{and} \quad F_2H_{22}^{(1)} = G_2^{(1)}. \quad (11)$$

Proof: Let us multiply (8) from the right by $N_{21} = \begin{bmatrix} I & -{H_{11}}^{\dagger}H_{12} \\ \mathbb{O} & I \end{bmatrix}$, i.e. write

$$[F_1 \;\; F_2]\begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} N_{21} = [G_1 \;\; G_2]N_{21}. \quad (12)$$

Then

$$[F_1 \;\; F_2]\begin{bmatrix} H_{11} & \mathbb{O} \\ H_{21} & H_{22}^{(1)} \end{bmatrix} = [G_1 \;\; G_2^{(1)}] \quad (13)$$

because $H_{11}H_{11}^{\dagger}H_{12} = H_{12}$ by Lemma 2 in Appendix A. Therefore, (11) follows from (13).

More associated details for the justification of the procedure in (8)–(13) are given in Appendix A. ∎

B. Reduction of equation (7) to p equations

Here, we extend the procedure represented by (11), (12) and (13) to the case of an arbitrary p. We do it by the following steps where each step is justified by an obvious extension of Lemma 1. This procedure is inspired by the idea which is used in the Gauss-Jordan elimination.

Step 1. First, in (7), for $k = 1,\dots,p$ and $j = 2,\dots,p$, we wish to update blocks H_{kj} and G_k to $H_{kj}^{(1)}$ and $G_k^{(1)}$, respectively. To this end, define

$$N_{p,1} = \begin{bmatrix} I & -{H_{11}}^{\dagger}H_{12} & \dots & -{H_{11}}^{\dagger}H_{1p} \\ \mathbb{O} & I & \dots & \mathbb{O} \\ \dots & \dots & \dots & \dots \\ \mathbb{O} & \dots & I & \mathbb{O} \\ \mathbb{O} & \dots & \mathbb{O} & I \end{bmatrix}.$$

Let $\mathcal{N}(H_{11}^T) \subseteq \mathcal{N}(H_{1j}^T)$ for $j = 2,\dots,p$. Then by (37),

$$H_{11}H_{11}^{\dagger}H_{1j} = H_{1j}, \quad \text{for } j = 2,\dots,p. \quad (14)$$

Therefore, we obtain

$$\begin{bmatrix} H_{11} & \mathbb{O} & \dots & \mathbb{O} \\ H_{21} & H_{22}^{(1)} & \dots & H_{2p}^{(1)} \\ \dots & \dots & \dots & \dots \\ H_{p-1,1} & H_{p-1,2}^{(1)} & \dots & H_{p-1,p}^{(1)} \\ H_{p1} & H_{p2}^{(1)} & \dots & H_{pp}^{(1)} \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & \dots & H_{1p} \\ H_{21} & H_{22} & \dots & H_{2p} \\ \dots & \dots & \dots & \dots \\ H_{p-1,1} & H_{p-1,2} & \dots & H_{p-1,p} \\ H_{p1} & H_{p2} & \dots & H_{pp} \end{bmatrix} N_{p,1} \quad (15)$$

and

$$[G_1 \;\; G_2^{(1)} \;\; \dots \;\; G_p^{(1)}] = [G_1 \;\; G_2 \;\; \dots \;\; G_p]N_{p,1}, \quad (16)$$

where, for $k = 1,\dots,p$ and $j = 2,\dots,p$,

$$H_{kj}^{(1)} = H_{kj} - H_{k1}H_{11}^{\dagger}H_{1j}, \quad (17)$$

$$G_j^{(1)} = G_j - G_1H_{11}^{\dagger}H_{1j}. \quad (18)$$

We continue this updating procedure up to the final Step $(p-1)$. In Step $(p-2)$, the following matrices are obtained:

$$\begin{bmatrix} H_{11} & \mathbb{O} & \dots & \mathbb{O} & \mathbb{O} \\ H_{21} & H_{22}^{(1)} & \dots & \mathbb{O} & \mathbb{O} \\ \dots & \dots & \dots & \dots & \dots \\ H_{p-1,1} & H_{p-1,2}^{(1)} & \dots & H_{p-1,p-1}^{(p-2)} & H_{p-1,p}^{(p-2)} \\ H_{p1} & H_{p2}^{(1)} & \dots & H_{p,p-1}^{(p-2)} & H_{pp}^{(p-2)} \end{bmatrix} \quad (19)$$

and

$$[G_1 \;\; G_2^{(1)} \;\; \dots \;\; G_{p-2}^{(p-3)} \;\; G_{p-1}^{(p-2)} \;\; G_p^{(p-2)}]. \quad (20)$$

Step $(p-1)$. Let $\mathcal{N}([H_{p-1,p-1}^{(p-2)}]^T) \subseteq \mathcal{N}([H_{p-1,p}^{(p-2)}]^T)$. Then (36) and (37) imply

$$H_{p-1,p-1}^{(p-2)}[H_{p-1,p-1}^{(p-2)}]^{\dagger}H_{p-1,p}^{(p-2)} = H_{p-1,p}^{(p-2)}. \quad (21)$$

In (19), we now update only two blocks $H_{p-1,p}^{(p-2)}$ and $H_{pp}^{(p-2)}$, and the single block $G_p^{(p-2)}$ in (20). To this end, define

$$N_{p,p-1} = \begin{bmatrix} I & \mathbb{O} & \dots & \mathbb{O} & \mathbb{O} \\ \mathbb{O} & I & \dots & \mathbb{O} & \mathbb{O} \\ \dots & \dots & \dots & \dots & \dots \\ \mathbb{O} & \mathbb{O} & \dots & I & -{H_{p-1,p-1}^{(p-2)}}^{\dagger}H_{p-1,p}^{(p-2)} \\ \mathbb{O} & \mathbb{O} & \dots & \mathbb{O} & I \end{bmatrix},$$

and obtain

$$\begin{bmatrix} H_{11} & \mathbb{O} & \dots & \mathbb{O} & \mathbb{O} \\ H_{21} & H_{22}^{(1)} & \dots & \mathbb{O} & \mathbb{O} \\ \dots & \dots & \dots & \dots & \dots \\ H_{p-1,1} & H_{p-1,2}^{(1)} & \dots & H_{p-1,p-1}^{(p-2)} & \mathbb{O} \\ H_{p1} & H_{p2}^{(1)} & \dots & H_{p,p-1}^{(p-2)} & H_{pp}^{(p-1)} \end{bmatrix}$$

$$= \begin{bmatrix} H_{11} & \mathbb{O} & \dots & \mathbb{O} & \mathbb{O} \\ H_{21} & H_{22}^{(1)} & \dots & \mathbb{O} & \mathbb{O} \\ \dots & \dots & \dots & \dots & \dots \\ H_{p-1,1} & H_{p-1,2}^{(1)} & \dots & H_{p-1,p-1}^{(p-2)} & H_{p-1,p}^{(p-2)} \\ H_{p1} & H_{p2}^{(1)} & \dots & H_{p,p-1}^{(p-2)} & H_{pp}^{(p-2)} \end{bmatrix} N_{p,p-1}$$

and

$$[G_1 \;\; G_2^{(1)} \;\; \dots \;\; G_{p-2}^{(p-3)} G_{p-1}^{(p-2)} \;\; G_p^{(p-1)}] = [G_1 \;\; G_2^{(1)} \;\; \dots \;\; G_{p-2}^{(p-3)} \;\; G_{p-1}^{(p-2)} \;\; G_p^{(p-2)}] N_{p,p-1}, \quad (22)$$

where

$$H_{pp}^{(p-1)} = H_{pp}^{(p-2)} - H_{p,p-1}^{(p-2)} (H_{p-1,p-1}^{(p-2)})^{\dagger} H_{p-1,p}^{(p-2)} \quad (23)$$

and

$$G_p^{(p-1)} = G_p^{(p-2)} - G_{p-1}^{(p-2)} (H_{p-1,p-1}^{(p-2)})^{\dagger} H_{p-1,p}^{(p-2)}. \quad (24)$$

As a result, on the basis of (21)–(24), equation (7) is reduced to the equation with the block-lower triangular matrix

$$[F_1 \;\; F_2 \;\; \dots \;\; F_{p-1} \;\; F_p] \times \begin{bmatrix} H_{11} & \mathbb{O} & \dots & \mathbb{O} & \mathbb{O} \\ H_{21} & H_{22}^{(1)} & \dots & \mathbb{O} & \mathbb{O} \\ \dots & \dots & \dots & \dots & \dots \\ H_{p-1,1} & H_{p-1,2}^{(1)} & \dots & H_{p-1,p-1}^{(p-2)} & \mathbb{O} \\ H_{p1} & H_{p2}^{(1)} & \dots & H_{p,p-1}^{(p-2)} & H_{pp}^{(p-1)} \end{bmatrix} = [G_1 \;\; G_2^{(1)} \;\; \dots \;\; G_{p-1}^{(p-2)} \;\; G_p^{(p-1)}]. \quad (25)$$

Using a back substitution, equation (25) is represented by p equations as follows:

$$F_p H_{pp}^{(p-1)} = G_p^{(p-1)}, \quad (26)$$

$$F_{p-1} H_{p-1,p-1}^{(p-2)} + F_p H_{p,p-1}^{(p-2)} = G_{p-1}^{(p-2)},$$

$$\vdots$$

$$F_2 H_{22}^{(1)} + \ldots + F_{p-1} H_{p-1,2}^{(1)} + F_p H_{p2}^{(1)} = G_2^{(1)},$$

$$F_1 H_{11} + \ldots + F_{p-1} H_{p-1,1} + F_p H_{p1} = G_1. \quad (27)$$

IV. Solution of the Problem in (2)

For $i = 1, \dots, p$, we write

$$D_i^{(i-1)} = G_i^{(i-1)} - \sum_{j=i+1}^{p} \bar{F}_j H_{ji}^{(i-1)}, \quad (28)$$

where

$$D_1^{(0)} = G_1^{(0)} - \sum_{j=2}^{p} \bar{F}_j H_{j1},$$

$$G_1^{(0)} = G_1, \quad H_{j1}^{(0)} = H_{j1}, \quad D_p^{(p-1)} = G_p^{(p-1)}$$

and $\bar{F}_p, \dots, \bar{F}_2$ are solutions to the following problems in (29)–(31), respectively.

Based on (7), (25)–(28), the problem in (2), (4), (5) is reduced to p problems

$$\min_{F_p \in \mathbb{R}_{r_p}^{s_p \times q_p}} \|D_p^{(p-1)} - F_p H_{pp}^{(p-1)}\|^2, \quad (29)$$

$$\min_{F_{p-1} \in \mathbb{R}_{r_{p-1}}^{s_{p-1} \times q_{p-1}}} \left\| D_{p-1}^{(p-2)} - F_{p-1} H_{p-1,p-1}^{(p-2)} \right\|^2, \quad (30)$$

$$\vdots$$

$$\min_{F_2 \in \mathbb{R}_{r_2}^{s_2 \times q_2}} \left\| D_2^{(1)} - F_2 H_{22}^{(1)} \right\|^2, \quad (31)$$

$$\min_{F \in \mathbb{R}_{r_1}^{s_1 \times q_1}} \left\| D_1^{(0)} - F_1 H_{11} \right\|^2. \quad (32)$$

The solution to each of the problems in (29)–(32) is given in [27]. To represent it here we use the following notation.

Let $M = U_M \Sigma_M V_M^*$ be the singular value decomposition (SVD) of $M \in \mathbb{R}^{m \times n}$ where $U_M \in \mathbb{R}^{m \times m}, V_M \in \mathbb{R}^{n \times n}$ are unitary matrices,

$$\Sigma_M = \mathrm{diag}(\sigma_1(M), \dots, \sigma_{\min(m,n)}(M)) \in \mathbb{R}^{m \times n}$$

is a generalized diagonal matrix, with the singular values $\sigma_1(M) \geq \sigma_2(M) \geq \ldots \geq 0$ on the main diagonal. Let $U_{M,m} = U_M = [u_1, \dots, u_m]$, $V_{M,n} = V_M = [v_1, \dots, v_n]$ be the column representations of U_M and V_M, respectively. For $\Sigma_{M,k} = \mathrm{diag}(\sigma_1(M), \dots, \sigma_k(M))$ and $k = 1, \dots, \mathrm{rank}\, M$, denote

$$\{M\}_{(k)} := U_{M,k} \Sigma_{M,k} V_{M,k}^T \in \mathbb{R}^{m \times n}, \quad (33)$$

i.e., $\{M\}_{(k)}$ is the k-truncated SVD of M. We also write

$$K_1^0 = D_1^{(0)} H_{11}^{\dagger} H_{11}$$

and, for $i = 2, \dots, p$,

$$K_i^{i-1} = D_i^{(i-1)} [H_{ii}^{(i-1)}]^{\dagger} H_{ii}^{(i-1)}.$$

Then the following is true.

Theorem 1: Let $\mathcal{N}([H_{kk}^{(k-1)}]^T) \subseteq \mathcal{N}([H_{kj}^{(k-1)}]^T)$ for $k = 1, \dots, p-1$ and $j = k+1, \dots, p$ where $H_{11} = H_{11}^{(0)}$. Then the multinomial KLT, i.e., the minimal Frobenius norm solution $\bar{F}_p, \dots, \bar{F}_1$ to the problem in (2) is given by

$$\bar{F}_p = \left\{ K_p^{p-1} \right\}_{(r_p)} [H_{pp}^{(p-1)}]^{\dagger}, \quad (34)$$

$$\bar{F}_{p-1} = \left\{ K_{p-1}^{p-2} \right\}_{(r_{p-1})} [H_{p-1,p-1}^{(p-2)}]^{\dagger},$$

$$\vdots$$

$$\bar{F}_2 = \left\{ K_2^1 \right\}_{(r_2)} [H_{22}^{(1)}]^{\dagger},$$

$$\bar{F}_1 = \left\{ K_1^0 \right\}_{(r_1)} H_{11}^{\dagger}, \quad (35)$$

respectively.

Proof: The proof follows from (29)–(32) and [27]. ■

For $p = 1$, the above solution coincides with the result represented in [1, 2, 3].

V. Numerical experiments

Here, we consider one in possible applications of the proposed multinomial KLT.

Let the source signal $\mathbf{x}$ be sent to p different unconnected local receivers $R_1, \ldots, R_p$. Each receiver R_j observes a noisy version of $\mathbf{x}$ denoted by $\mathbf{y}_j$, for $j = 1, \ldots, p$.. Receivers $R_1, \ldots, R_p$ should filter observed signals $\mathbf{y}_1, \ldots, \mathbf{y}_p$, reduce their dimensions and then sent them to a so called collecting receiver R. The collecting receiver should recover signal $\mathbf{x}$ from all outputs of the receivers $R_1, \ldots, R_p$. On the basis of (33), the model of a jth receiver R_j is represented by $r_j \times n_j$ matrix obtained from $\{K_j^{j-1}\}_{(r_j)}$ in the form $U_{M,k}\Sigma_{M,k}$ or $U_{M,k}$, for $M = K_j^j$ and $k = r_j$.

Signals $\mathbf{x}$ and $\mathbf{y}_j$ are simulated as matrices $X \in \mathbb{R}^{m\times n}$ and $Y_j = X + 100N_j$, respectively, where $m = n = 256$ and $N_j \in \mathbb{R}^{m\times n}$ is a matrix generated randomly from the standard normal distribution, for $j = 1, \ldots, p$. Matrix X represents the data obtained from an aerial digital photograph of a plant[1]. It is represented by Fig. 1 (a). As an example, in Fig. 1 (b), matrix Y_{10} is represented, for the case of $p = 20$ local receivers.

In this notation, the multinomial KLT given by (34)–(35) has been applied to dimensionality reduction, filtering and subsequent recovering the source signal. Fig. 1 (c) represents image $\widetilde{X}$ which is the recovered image X by the multinomial KLT after dimensionality reduction and filtering, for the case $p = 20$ and $r_j = m/2$, for $j = 1, \ldots, 20$.

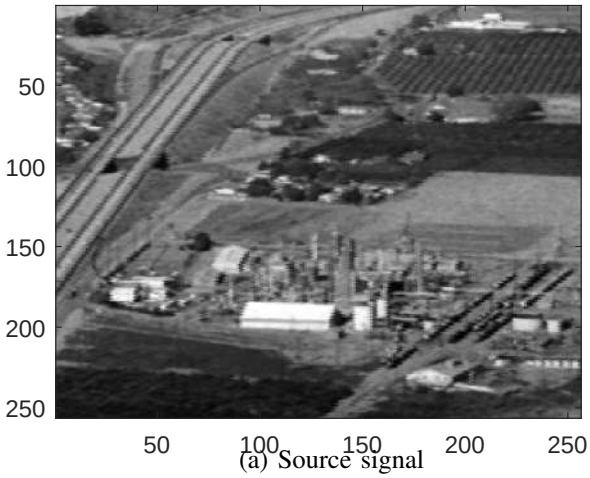
In Table 1, values of the error $\|X - \widetilde{X}\|^2$ are given, for some p and r_j, $j = 1, \ldots, p$. For a fixed r_j, the increase in $\|X - \widetilde{X}\|^2$ with the increase in p is associated with accumulated numerical errors involved by computation of different SVDs in (34)–(35). For example, for $r_j = \lfloor m/3 \rfloor$ and $j = 1, \ldots, p$, the multinomial KLT error associated with 50 local receivers is 1.61×10^6 while that associated with 3 local receivers is 1.47×10^6.

To illustrate the error associated with a particular row in matrices X and $\widetilde{X}$ we provide Fig. 2. It represents the diagram of the error $\|X(128,:) - \widetilde{X}(128,:)\|^2$, for $p = 20$, where $X(128,:)$ and $\widetilde{X}(128,:)$ are 128th row of matrices X and $\widetilde{X}$. For all other rows, the values and behavior of the error are similar.

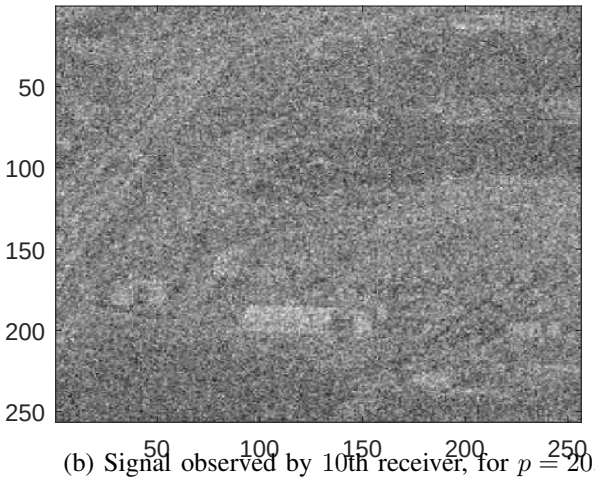
The Matlab code used in the above simulations is given in Appendix B.

VI. Conclusion

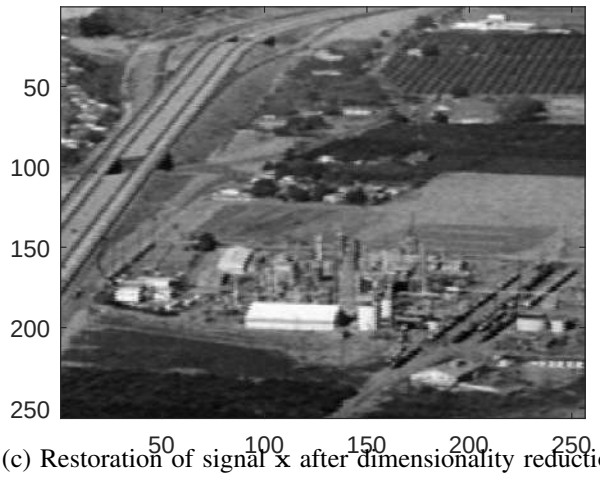
We have proposed a new transform for random vectors called the multinomial Karhunen-Loève transform (KLT). The multinomial KLT is motivated by a number of important applied problems and is based on the solution of the error minimization problem subject to multiple rank constraints. The solution of that problem has been provided. The main idea of the solution is a reduction of the associated matrix equation to the matrix with a lower block-triangular form. The multinomial KLT is illustrated by the numerical simulations. The Matlab code for the implementation of the multinomial KLT is provided.



(a) Source signal



(b) Signal observed by 10th receiver, for $p = 20$.



(c) Restoration of signal $\mathbf{x}$ after dimensionality reduction and filtering, for $p = 20$ and $r_j = m/2$ $\forall j = 1, \ldots, 20$.

Fig. 1. Illustrations for numerical simulations with the multinomial KLT.

[1]The database is available in http://sipi.usc.edu/services/database/Database.html.

Table 1. Values of the error associated with X restoration by the multinomial KLT, for some p and r_j

p	2	3	5	6	8	10	20	50
$r_j = m/2$, for $j = 1, \ldots, p$.								
$\|X - \widetilde{X}\|^2$	1.0	3.60×10^5	3.72×10^5	3.74×10^5	3.77×10^5	3.82×10^5	3.87×10^5	3.91×10^5
$r_j = \lfloor m/3 \rfloor$, for $j = 1, \ldots, p$.								
$\|X - \widetilde{X}\|^2$	0.38×10^5	1.47×10^6	1.44×10^6	1.52×10^6	1.55×10^6	1.56×10^6	1.57×10^6	1.61×10^6
$r_j = m/4$, for $j = 1, \ldots, p$.								
$\|X - \widetilde{X}\|^2$	2.3×10^5	2.90×10^6	2.95×10^6	2.94×10^6	2.98×10^6	2.97×10^6	3.09×10^6	3.17×10^6

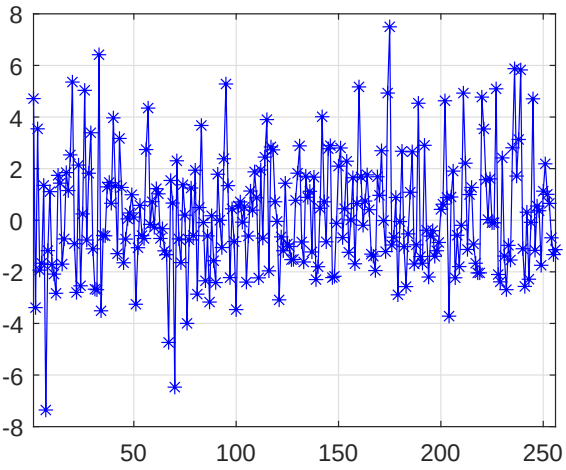


Fig. 2. Diagram of the error $\|X(128,:) - \widetilde{X}(128,:)\|^2$, for $p = 20$, where $X(128,:)$ and $\widetilde{X}(128,:)$ are 128th row of matrices X and $\widetilde{X}$.

APPENDIX A
JUSTIFICATION OF THE PROCEDURE IN (12)–(13)

Here, we wish to show an equivalence of equations (8) and (11). To this end below, we first provide Lemma 2 where a basis for the procedure under consideration is given. Then in Lemma 3, we provide a different proof of the representation of equation (8) by two equations in (11). A particular purpose of the proof of Lemma 3 is to establish the required equivalence.

In this regard, see also Remark 1 below.

Lemma 2: Let $P \in \mathbb{R}^{m\times n}$ and $Q \in \mathbb{R}^{m\times n}$. Then

$$\mathcal{N}(P^T) \subseteq \mathcal{N}(Q^T) \tag{36}$$

implies

$$PP^\dagger Q = Q. \tag{37}$$

Proof: Let $\mathcal{R}(M)$ be the range space of a matrix M. Then $\mathcal{N}(P^T) \subseteq \mathcal{N}(Q^T)$ is equivalent to $\mathcal{R}(P) \supseteq \mathcal{R}(Q)$. Now suppose $y \in \mathbb{R}^n$. Then we can find some $x \in \mathbb{R}^n$ such that $Px = Qy$. Therefore $Qy = Px = PP^\dagger Px = PP^\dagger Qy$. Since y is arbitrary it follows that $Qy = PP^\dagger Qy$ for all $y \in \mathbb{R}^n$. Therefore $Q = PP^\dagger Q$. ∎

Lemma 3: Let F_1 and F_2 satisfy equation (8). Let

$$\mathcal{N}(H_{11}) \subseteq \mathcal{N}(H_{21}) \quad \text{and} \quad \mathcal{N}(H_{11}^T) \subseteq \mathcal{N}(H_{12}^T). \tag{38}$$

Then (8) decomposes into equations in (11).

Proof: Let us denote $\bar{F}_1 = F_1 + F_2 H_{21} H_{11}^\dagger$. Then

$$[\bar{F}_1 \;\; F_2] = [F_1 \;\; F_2] \begin{bmatrix} I & \mathbb{O} \\ H_{21}H_{11}^\dagger & I \end{bmatrix}$$

and

$$[F_1 \;\; F_2] = [\bar{F}_1 \;\; F_2] \begin{bmatrix} I & \mathbb{O} \\ -H_{21}H_{11}^\dagger & I \end{bmatrix}. \tag{39}$$

Thus, (8) is represented as

$$[\bar{F}_1 \;\; F_2] \begin{bmatrix} I & \mathbb{O} \\ -H_{21}H_{11}^\dagger & I \end{bmatrix} \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} = [G_1 \;\; G_2]$$

which implies

$$\begin{aligned} [\bar{F}_1 \;\; F_2] &\begin{bmatrix} I & \mathbb{O} \\ -H_{21}H_{11}^\dagger & I \end{bmatrix} \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \\ &\times \begin{bmatrix} I & -H_{11}^\dagger H_{12} \\ \mathbb{O} & I \end{bmatrix} \\ &= [G_1 \;\; G_2] \begin{bmatrix} I & -H_{11}^\dagger H_{12} \\ \mathbb{O} & I \end{bmatrix}, \end{aligned} \tag{40}$$

where by (36)–(38),

$$H_{21}H_{11}^\dagger H_{11} = H_{21} \quad \text{and} \quad H_{11}^\dagger H_{11} H_{12} = H_{12}. \tag{41}$$

On the basis of (40), we see that (8) reduces to the equation

$$[\bar{F}_1 \;\; F_2] \begin{bmatrix} H_{11} & \mathbb{O} \\ \mathbb{O} & H_{22}^{(1)} \end{bmatrix} = [G_1 \;\; G_2^{(1)}]. \tag{42}$$

The latter implies (11). ∎

Remark 1: The equivalence of equations (8) and (11) follows from (39). Having that, the procedure for reducing (8) to (11) can be simplified so that in (38), the assumption $\mathcal{N}(H_{11}) \subseteq \mathcal{N}(H_{21})$ is omitted. The procedure is given in the proof of Lemma 1 in Section III-A.

APPENDIX B
MATLAB CODE FOR THE IMPLEMENTATION OF MULTINOMIAL KLT IN (34)–(35)

The Matlab code used in Section V for the numerical experiments is as follows.

Matlab code for the multinomial KLT:

```
load plant.mat
m=256; X=plant(1:m,1:m); p=20; r=floor(m/2);
Y=[]; nr=m;     (% nr is a number of samples)
for j=1:p do
    Y_j=X+100*randn(m,nr); Y=[Y; Y_j];
end for
```

```
Eyy=(1/nr)*(Y*Y'); Exy=(1/nr)*(X*Y');
[ug,sg,vg]=svd(Eyy); dg=sqrt(diag(sg));
Eyy12=ug*diag(dg)*vg'; Hm=Eyy12;
pEyy12=pinv(Eyy12); Gm=Exy*pEyy12;
q=m;
G=zeros(q,q,p,p); Gorgnl=zeros(q,q,p,p);
for k=1:p do
    for j=1:p do
        G(:,:,k,j)=Hm((k-1)*m+1:k*m,(j-1)*m+1:j*m);
        Gorgnl(:,:,k,j)=G(:,:,k,j);
    end for
end for
H=zeros(m,q,p);
for k=1:p do
    H(:,:,k)=Gm(:,(k-1)*m+1:k*m);
    Horgnl(:,:,k)=H(:,:,k);
end for
for s=1:p-1 do
    for k=s:p do
        for j=s+1:p do
            pGs=pinv(G(:,:,s,s));
            G(:,:,k,j)=G(:,:,k,j)-G(:,:,k,s)*pGs*G(:,:,s,j);
            H(:,:,k)=H(:,:,k)-H(:,:,s)*pGs*G(:,:,s,k);
        end for
    end for
end for
F=zeros(m,q,p); pGp=pinv(G(:,:,p,p));
[U1,S1,V1]=svd(H(:,:,p)*pGp*G(:,:,p,p));
F(:,:,p)=U1(:,1:r)*S1(1:r,1:r)*(V1(:,1:r))'*pGp;
for s=p-1:-1:2 do
    Sum=zeros(m,q);
    for j=s+1:p do
        Sum=Sum+F(:,:,j)*G(:,:,j,s);
    end for
    pGs=pinv(G(:,:,s,s));
    M_s=(H(:,:,s)-Sum)*pGs*G(:,:,s,s);
    [U2,S2,V2]=svd(M_s);
    F(:,:,s)=U2(:,1:r)*S2(1:r,1:r)*(V2(:,1:r))'*pGs;
end for
Sum=zeros(m,q);
for j=2:p do
    Sum=Sum+F(:,:,j)*Gorgnl(:,:,j,1);
end for
pG1=pinv(Gorgnl(:,:,1,1));
M_1=(Horgnl(:,:,1)-Sum)*pG1*G(:,:,1,1);
[U3,S3,V3]=svd(M_1);
F(:,:,1)=U3(:,1:r)*S3(1:r,1:r)*(V3(:,1:r))'*pG1;
FA=[];
for i=1:p do
    FA=[FA F(:,:,i)];
end for
Xest=FA*Y; error=norm(X-Xest,'fro')^2;
```

REFERENCES

[1] L. Scharf, "The SVD and reduced rank signal processing," *Signal Processing*, vol. 25, no. 2, pp. 113 – 133, 1991.

[2] Y. Hua and W. Liu, "Generalized Karhunen-Loeve transform," *IEEE Signal Processing Letters*, vol. 5, no. 6, pp. 141–142, June 1998.

[3] Y. Hua, M. Nikpour, and P. Stoica, "Optimal reduced-rank estimation and filtering," *IEEE Transactions on Signal Processing*, vol. 49, no. 3, pp. 457–469, Mar 2001.

[4] H. Kamrani, A. Z. Asli, P. P. Markopoulos, M. Langberg, D. A. Pados, and G. N. Karystinos, "Reduced-rank l1-norm principal-component analysis with performance guarantees," *IEEE Transactions on Signal Processing*, vol. 69, pp. 240–255, 2021.

[5] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," *Phil. Trans. R. Soc. A*, vol. 374, pp. 1 – 16, 2016.

[6] Y. Bar-Shalom, "Redundancy and data compression in recursive estimation," *IEEE Transactions on Automatic Control*, vol. 17, no. 5, pp. 684–689, 1972.

[7] D. Willner, C. B. Chang, and K. P. Dunn, "Kalman filter configurations for multiple radar systems," *Technical Note 1976-21, Lincoln Laboratory, MIT*, 1976.

[8] M. Gastpar, P. Dragotti, and M. Vetterli, "The distributed Karhunen-Loève transform," *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5177–5196, Dec 2006.

[9] P. L. Dragotti and M. Gastpar, *Distributed Source Coding: Theory, Algorithms and Applications*. Academic Press, 2009.

[10] E. Song, Y. Zhu, and J. Zhou, "Sensors' optimal dimensionality compression matrix in estimation fusion," *Automatica*, vol. 41, no. 12, pp. 2131 – 2139, 2005.

[11] Y. Zhu, E. Song, J. Zhou, and Z. You, "Optimal dimensionality reduction of sensor data in multisensor estimation fusion," *IEEE Transactions on Signal Processing*, vol. 53, no. 5, pp. 1631 – 1639, 2005.

[12] I. D. Schizas, G. B. Giannakis, and Z.-Q. Luo, "Distributed estimation using reduced-dimensionality sensor observations," *IEEE Transactions on Signal Processing*, vol. 55, no. 8, pp. 4284–4299, Aug 2007.

[13] O. Roy and M. Vetterli, "Dimensionality reduction for distributed estimation in the infinite dimensional regime," *IEEE Transactions on Information Theory*, vol. 54, no. 4, pp. 1655–1669, April 2008.

[14] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Dimensionality reduction, compression and quantization for distributed estimation with wireless sensor networks," in *Wireless Comm.*, ser. The IMA Volumes in Mathematics and its Applications. Springer New York, 2007, vol. 143, pp. 259–296.

[15] J. Fang and H. Li, "Optimal/near-optimal dimensionality reduction for distributed estimation in homogeneous and certain inhomogeneous scenarios," *IEEE Transactions on Signal Processing*, vol. 58, no. 8, pp. 4339–4353, Aug 2010.

[16] A. Amar, A. Leshem, and M. Gastpar, "Recursive implementation of the distributed Karhunen-Loeve transform," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5320–5330, Oct 2010.

[17] J. Li and G. AlRegib, "Distributed estimation in energy-

constrained wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 57, no. 10, pp. 3746–3758, 2009.

[18] M. Lara and B. Mulgrew, "Performance of the distributed KLT and its approximate implementation," *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pp. 724–728, Aug 2012.

[19] H. Ma, Y.-H. Yang, Y. Chen, K. J. R. Liu, and Q. Wang, "Distributed state estimation with dimension reduction preprocessing," *IEEE Transactions on Signal Processing*, vol. 62, no. 12, pp. 3098–3110, June 2014.

[20] D. E. Marelli and M. Fu, "Distributed weighted least-squares estimation with fast convergence for large-scale systems," *Automatica*, vol. 51, pp. 27 – 39, 2015.

[21] S. Xua, R. C. de Lamare, and H. V. Poor, "Distributed low-rank adaptive estimation algorithms based on alternating optimization," *Signal Processing*, vol. 144, pp. 41 – 51, 2018.

[22] L. Zhang, D. Niu, E. Song, J. Zhou, Q. Shi, and Y. Zhu, "Joint optimization of dimension assignment and compression in distributed estimation fusion," *IEEE Transactions on Signal Processing*, vol. 67, no. 9, pp. 2453–2468, 2019.

[23] P. Tseng, "Convergence of a block coordinate descent method for nondifferentiable minimization," *Journal of Optimization Theory and Applications*, vol. 109, no. 3, pp. 475–494, 2001.

[24] M. Razaviyayn, M. Hong, and Z.-Q. Luo, "A unified convergence analysis of block successive minimization methods for nonsmooth optimization," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 1126–1153, 2013.

[25] Z. Li, A. Uschmajew, and S. Zhang, "On convergence of the maximum block improvement method," *SIAM Journal on Optimization*, vol. 25, no. 1, pp. 210–233, 2015.

[26] A. Beck and L. Tetruashvili, "On the convergence of block coordinate descent type methods," *SIAM J. Optim.*, vol. 23, no. 4, pp. 2037 – 2060, 2013.

[27] S. Friedland and A. Torokhti, "Generalized rank-constrained matrix approximations," *SIAM Journal on Matrix Analysis and Applications*, vol. 29, no. 2, pp. 656–659, 2007.

Anexo 13

http://dx.doi.org/10.15446/ing.investig.xxxx

FroImPro: A Mathematical-Computational Framework Based on Frobenius Norm for a Set of Image Processing Problems

FroImPro: Un Marco Matemático-Computacional Basado en la Norma de Frobenius para un Conjunto de Problemas de Procesamiento de Imágenes

[Juan José Fallas-Monge][1], [Jeffry Chavarría-Molina][2], and [Pablo Soto-Quiros][3]

ABSTRACT

This paper proposes a new mathematical-computational framework that generalizes a set of minimization problems in image processing. A mathematical-computational framework will be understood as the set formed by a mathematical formulation of an algorithm and its implementation in some programming language. Our interest is to provide a new mathematical framework that generalizes a set of minimization problems in image processing, where the objective function includes the Frobenius norm. Additionally, this mathematical model is validated numerically using a computational implementation in MATLAB. We create a MATLAB toolbox called `FroImPro` with the collections of functions built. This mathematical-computational framework will allow having a better understanding of each technique to solve each image processing problem mentioned above.

Keywords: [Frobenius Norm], [Mathematical-Computational Framework], [MATLAB], [Image Processing].

RESUMEN

Este artıculo propone un nuevo marco matematico-computacional que generaliza un conjunto de problemas de minimizacion en el procesamiento de imagenes. Se entendera por marco matematico-computacional el conjunto formado por la formulacion matematica de un algoritmo y su implementacion en algun lenguaje de programacion. Nuestro interes es proporcionar un nuevo marco matematico que generalice un conjunto de problemas de minimizacion en el procesamiento de imagenes, donde la funcion objetivo incluye la norma de Frobenius. Adicionalmente, este modelo matematico se valida numericamente mediante una implementacion computacional en MATLAB. Creamos un *toolbox* de MATLAB llamada `FroImPro` con las colecciones de funciones construidas. Este marco matematico-computacional permitira tener una mejor comprension de cada tecnica para resolver cada problema de procesamiento de imagenes mencionado anteriormente.

Palabras clave: [Norma de Frobenius], [Marco Matematico Computacional], [MATLAB], [Procesamiento de Imagenes].

Received: January _th 20xx
Accepted: January _th 20xx

Introduction

Let $\mathbb{R}^{m\times n}$ be the set of all real $m \times n$ matrices. The Frobenius norm is a matrix norm in $\mathbb{R}^{m\times n}$ defined as the square root of the sum of the squares of its elements, i.e.,

$$\|A\|_{fr} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} A(i,j)^2},$$

where $A \in \mathbb{R}^{m\times n}$. This matrix norm is used in other set, e.g., complex matrices (Friedland & Torokhti, 2007), tensors (Wang, Che & Wei, 2020) and quaternions matrices (Ahmad, Ali & Slapnicar, 2021).

The Frobenius norm measures the size of a matrix in terms of Euclidean distance. The minimization of the square of a Frobenius norm is similar to the classical least-square minimization. Additionally, you can constrain the optimization to satisfy what you expect from images if you add a penalty. The square of a Frobenius norm puts you in the framework of proximity operators (a generalization of projections), for which efficient algorithms exist. For these reasons, the Frobenius norm is frequently used to define optimization problems in engineering, for example, speech enhancement (Sun, Xie & Leng, 2016), computer algebra, system and control theory (Markovsky, 2008), seismic data acquisition (Siahsar, Gholtashi, Abolghasemi & Chen, 2017), multi-view subspace clustering (Brbić & Kopriva, 2018), WiFi-based indoor localization (Gu, Chen, Zhang, Zhu, Lu & Chen, 2016) and signal processing (Torokhti & Soto-Quiros, 2016).

One of the most crucial areas where the Frobenius norm is used is developing optimization problems related to image processing. For example, the Frobenius norm is

[1]Graduate Title Abbreviation, Institution, Country. Postgraduate Titles, Institution, Country. Affiliation: Current position, Institution, Country. E-mail: xxxx@xx.xx.xx
[2]Electrical Engineer, Universidad Nacional de Colombia, Colombia. M.Sc. Electrical Engineering, Universidad Nacional de Colombia, Colombia. Affiliation: Assistant Professsor, Universidad Nacional de Colombia, Colombia. E-mail: xxxx@xx.xx.xx
[3]Electronics Engineer, Universidad Nacional de Colombia, Colombia. Ph.D. Compu-ter Science, Universidad Nacional de Colombia, Colombia. Affiliation: Emeritus Professor, Universidad Nacional de Colombia, Colombia. E-mail: xxxx@xx.xx.xx

How to cite: Uparela, M., Gonzalez, R., Jimenez, J., Quintero, C. (2018). Intelligent System for non-technical losses management in residential users of the electricity sector. Ingeniería e Investigación, xx(xx), xx-xx. DOI: 10.15446/ing.investig.xxxx

used in researches related to image compression, image denoising (Chung & Chung, 2013; Soto-Quiros, Jose Fallas-Monge & Chavarría-Molina, 2022), background modeling and shadow/light removal (Zhou & Tao, 2011), feature extraction and face recognition (Lee & Seung, 1999; Gillis, 2020), and image reconstruction (Aharon, Elad & Bruckstein, 2006; Aharon et al., 2006; Bryt & Elad, 2008).

The image processing problems mentioned have been studied in various scientific disciplines. However, there is a need to develop a more robust mathematical-computational framework than the existing ones to analyze and extend known methods to improve the accuracy and performance of those methods. A mathematical-computational framework will be understood as the set formed by a mathematical formulation of an algorithm and its implementation in some programming language.

This paper proposes a new mathematical-computational framework that generalizes a set of minimization problems in image processing. This model can be represented mathematically as

$$\min_{\mathbf{X}\in\mathbb{F}} \|A - \mathcal{F}(\mathbf{X})\|_{fr}^2, \tag{1}$$

where $A \in \mathbb{R}^{m\times n}$, $\mathbb{F}$ is the constrained set, $\mathbf{X}$ is a element in $\mathbb{F}$ and $\mathcal{F}$ is a matrix operator. In this paper, we consider the following five Frobenius norm minimization problems in image processing:

- the low-rank matrix approximation (LRMA) problem (Eckart & Young, 1936; Datta, 2010),
- the generalized low-rank approximation(GLRMA) problem (Friedland & Torokhti, 2007; Chavarría-Molina, Fallas-Monge & Soto-Quiros, 2022; Soto-Quiros et al., 2022)
- the go decomposition (GoDec) (Zhou & Tao, 2011)
- the nonnegative matrix factorization (NNMF) (Lee & Seung, 1999; Yuan & Oja, 2005; Gillis, 2020)
- the K-SVD method (Aharon et al., 2006; Bryt & Elad, 2008)

The minimization problem in each image processing problem mentioned before can be represented by the mathematical model represented by (1). Furthermore, this framework includes a new image processing MATLAB toolbox based on the aforementioned Frobenius norm minimization problems. This new toolbox is called `FroImPro` (*Frobenius - Image Processing*). The algorithms implemented in the toolbox allow solving a set of problems related to image processing, for example, image compression, image denoising, background modeling, feature extraction, and image reconstruction.

Motivation

Although there is a connection between the mathematical formulation of the problems in image processing mentioned above, there is no research that studies this relationship. This gap in the literature is the primary motivation of this research. Our interest is to provide a new mathematical framework that generalizes a set of minimization problems in image processing, where the objective function includes the Frobenius norm. Additionally, this mathematical model is validated numerically using a computational implementation in MATLAB. This mathematical-computational framework will allow having a better understanding of each technique to solve each image processing problem mentioned above.

Table 1. Minimization Problems in Image Processing Associated with the Mathematical Framework in (2)

Problem	$\mathbb{F}$	$\mathbf{X}$	$\mathcal{F}(\mathbf{X})$
LRMA	$\mathbb{R}_r^{m\times n}$	X	X
GLRMA	$\mathbb{R}_r^{m\times s}$	X	XB
GoDec	$\mathbb{R}_r^{m\times n} \times \mathbb{S}_s^{m\times n}$	(X_1, X_2)	$X_1 + X_2$
NNMF	$\mathbb{R}_{\geq 0}^{m\times r} \times \mathbb{R}_{\geq 0}^{r\times n}$	(X_1, X_2)	X_1X_2
K-SVD	$\mathbb{R}^{m\times r} \times \mathbb{T}_c^{r\times n}$	(X_1, X_2)	X_1X_2

Contribution

The main contribution of the research can be summarized as follows:

- A new mathematical model that generalizes a set of minimization problems in image processing. This mathematical model allows analysis and extends known methods to improve the accuracy and performance of those methods.
- A brief review of five relevant problems in image processing based on the Frobenius norm.
- The computational implementation of the proposed mathematical framework in MATLAB. Additionally, we create a MATLAB toolbox called `FroImPro` with the collections of functions built. This toolbox includes a manual that provides the documentation of the`FroImPro` toolbox for solving a set of image processing problems: image compression, image denoising, background modeling, feature extraction, and image reconstruction. Additionally, the proposed manual briefly explains each image processing problem mentioned above.

A Mathematical Framework of Image Processing Problems Based on Frobenius Norm

The image processing problems mentioned previously have a similar mathematical structure. This paper considers a general Frobenius norm minimization problem as a mathematical framework to represent each image processing problem mentioned above. The following minimization problem represents this mathematical framework:

$$\min_{\mathbf{X}\in\mathbb{F}} \|A - \mathcal{F}(\mathbf{X})\|_{fr}^2, \tag{2}$$

where $A \in \mathbb{R}^{m\times n}$ represents the observed data, $\mathbb{F}$ is the constrained set, $\mathbf{X}$ is a element in $\mathbb{F}$ and $\mathcal{F}$ is a matrix operator. Characterization of $\mathbb{F}$, $\mathbf{X}$ and $\mathcal{F}$, for each minimization problem considered in this paper, are presented in Table 1. Additionally, Table 2 presents each matrix subset used in the mathematical framework in (2).

Table 2. Description of Each Set Associated with the Mathematical Framework in (2)

Set	Description
$\mathbb{R}^{p\times q}$	Real $p \times q$ matrices
$\mathbb{R}_r^{p\times q}$	Real $p \times q$ matrices of rank at most $r \leq \min\{p, q\}$
$\mathbb{S}_s^{p\times q}$	Real $p \times q$ matrices with sparsity cardinality $s \leq pq$
$\mathbb{R}_{\geq 0}^{p\times q}$	Real $p \times q$ matrices with nonnegative entries
$\mathbb{T}_c^{p\times q}$	Real $p \times q$ matrices such that each column has sparsity cardinality c

If $\widehat{\mathbf{X}}$ is a solution of problem (2), then the mean squared error between A and $\widehat{\mathbf{X}}$ is given by

$$MSE(A, \widehat{\mathbf{X}}) = \frac{1}{K}\|A - \mathcal{F}(\widehat{\mathbf{X}})\|_{fr}^2, \tag{3}$$

where K is the number of entries of A.

In the following sections, we present a brief description of each minimization problem and its relation with image processing.

Problem 1: LRMA and Image Compression

The goal of the LRMA is to approximate $A \in \mathbb{R}^{p\times q}$ with a low-rank matrix $X \in \mathbb{R}_r^{m\times n}$, such that X solve the following problem

$$\min_X \|A - X\|_{fr}^2 \tag{4}$$

This problem was proposed and solved in (Eckart & Young, 1936) and is applied in a wide range of areas, such as machine learning and recommender systems (Fritzen & Ryckelynck, 2019). If $A = USV^T$ is the SVD of A, then a solution of (4) is given by $X = U_r S_r V_r^T$, where $U_r \in \mathbb{R}^{p\times r}$ and $V_r \in \mathbb{R}^{q\times r}$ are formed with the first r columns of U and V and $S_r \in \mathbb{R}^{r\times r}$ is formed tithe first r rows and columns of S. However, the SVD method is usually costly in terms of the execution time for high-dimensional matrices. For this reason, a fast algorithm to estimate the low-rank approximation was proposed in (Chavarría-Molina et al., 2022), which is based on the so-called bilateral random projections. This algorithm is so-called the modified bilateral random projections (MBRP). Briefly, the MBRP method estimates the low-rank of A as the product of two matrices $D \in \mathbb{R}^{p\times r}$ and $C \in \mathbb{R}^{r\times q}$, i.e., $X \approx DC$. Theoretical and numerical results in (Chavarría-Molina et al., 2022) show the advantages of the MBRP method.

An application of the LRA problem is image compression. Here, the idea of image compression is to compress a grayscale image A to the one which corresponds to a lower-order approximation of A but whose quality is still acceptable. If we store image A, we keep in memory pq different values (entries). Otherwise, if we store a low-rank approximation of A, i.e., X, then, using the MBRP method, $X \approx DC$. Thus, we store in memory the entries of D and C. i.e., pr and qr values, respectively. Therefore, the amount of data needed to store is $pr + qr = r(p + q)$. A numerical example of image compression using the LRA is given by Figure 1.

Problem 2: GLRMA and Image Denoising

In this problem, we consider matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{s\times n}$. The GLRMA problem estimates $X \in \mathbb{R}_r^{m\times s}$ such that minimizes the following minimization problem

$$\min_X \|A - XB\|_{fr}^2 \tag{5}$$

A effective implementation to obtain a solution of this problem was proposed in (Chavarría-Molina et al., 2022). This method is so-called the fast-GRLMA, which is is based on tensor product and Tikhonov's regularization to approximate the Moore-Penrose inverse and bilateral random projections to estimate, in turn, the LRMA. As mentioned in (Chavarría-Molina et al., 2022), the fast-GLRMA method significantly reduces the execution time to compute the optimal solution, while preserving the accuracy of the classical method of solving the GLRMA.

The GRLMA problem is used to solve the problem of compression, filtering and decompression of a noisy image, using training data (Soto-Quiros et al., 2022; Chung & Chung, 2013). We consider two training set of images $\mathcal{A} = \{A^{(1)}, ..., A^{(m)}\}$ and $\mathcal{B} = \{B^{(1)}, ..., B^{(m)}\}$, where $B^{(j)}$ is a noisy version image of $A^{(j)}$, for all $j = 1, ..., m$. Let a_j and b_j be the vectorized representation of $A^{(j)}$ and $B^{(j)}$, respectively, i.e., convert each matrix into a column vector by stacking the columns of the matrix. Then, using matrices $A = [a_1\ a_2\ ...\ a_m]$ and $B = [b_1\ b_2\ ...\ b_m]$, we can obtain a low-rank matrix filter X by solving problem (5). This matrix X filters a vectorized version of a new noisy image $\overline{B}$, where $\overline{B}$ does not necessarily belong to $\mathcal{B}$, but it is "similar" to one of them. This process is represented graphically in Figure 2.

Problem 3: GoDec and Background Modeling

The "Go Decomposition" (GoDec) is an algorithm developed in (Zhou & Tao, 2011) to estimate the low-rank part $X_1 \in \mathbb{R}_r^{m\times n}$ and the sparse part $X_2 \in \mathbb{S}_s^{m\times n}$ of a matrix $A \in \mathbb{R}^{m\times n}$, where $A = X_1 + X_2 + N$ with noise $N \in \mathbb{R}^{m\times n}$. The optimization problem of GoDec is represented as follows

$$\min_{X_1, X_2} \|A - X_1 - X_2\|_{fr}^2. \tag{6}$$

Problem (6) can be solved by alternatively solving the following two subproblems until convergence

$$X_1^{(k+1)} = \arg\min_{X_1 \in \mathbb{R}_r^{m\times n}} \|A - X_1 - X_2^{(k)}\|_{fr}^2 \tag{7}$$

$$X_2^{(k+1)} = \arg\min_{X_2 \in \mathbb{S}_s^{m\times n}} \|A - X_1^{(k+1)} - X_2\|_{fr}^2. \tag{8}$$

Solution of subproblem (7) is given by the LRMA of $A - X_2^{(k)}$, and solution of subproblem (8) can be solved by updating $X_2^{(k+1)}$ via entry-wise hard thresholding of $A - X_1^{(k+1)}$. It follows from Theorem 1 in (Zhou & Tao, 2011) that GoDec algorithm produces a sequence $\{X_1^{(k)}, X_2^{(k)}\}_{k=0}^{\infty}$ that converges linearly to a local minimum of (6).

An application of the GoDec method is background modeling. As mentioned in (Bouwmans, Aybat & Zahzah, 2016; Zhou & Tao, 2011), the background modeling shows the correlation between video frames, model background variations, and foreground moving objects. A video

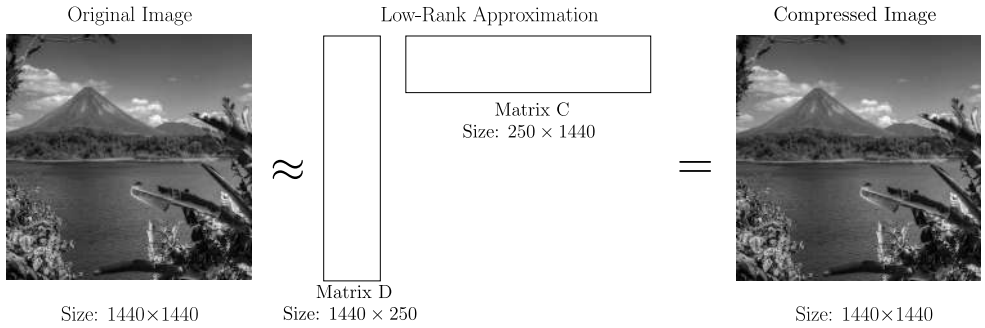


Figure 1. Example of image compression using LRMA, where original image stores 2 073 600 different entries (pixels). However, using a low-rank approximation of the original image, we can store 720 000 entries, using matrices D and C. Finally, we multiply matrices D and C to obtain a low-rank approximation of the original image. In this example, the MSE is 4.4199×10^{-4}.

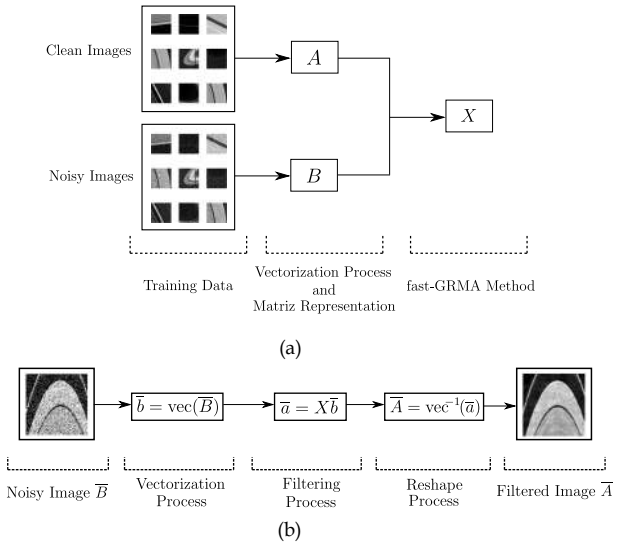


Figure 2. (a) A block diagram about obtaining the filter matrix X with the fast-GLRMA method using images training data. (b) A block diagram about how to filter a noisy image $\overline{B}$, using the filter matrix X.

sequence satisfies the low-rank+sparse structure because the backgrounds of all the frames are related, while the variation and the moving objects are sparse and independent. Let $\mathcal{A} = \{A^{(1)}, ..., A^{(m)}\}$ be a set of m frames of a grayscale video, and let a_j be the vectorized representation of $A^{(j)}$. By applying the GoDec Algorithm to $A = [a_1\ a_2\ ...\ a_m]$, the matrices X_1 and X_2 are determined. For each frame, X_1 stores the information associated with the background, and X_2 contains the information related to moving objects. Therefore, the i-th column of X_1 (represented as a matrix) corresponds to the background of the i-th frame (pixels that remain invariant), and the i-th column of X_2 (expressed as a matrix) corresponds to the pixels of the moving objects of the i-th frame. A numerical example of this application is presented in Figure 3.

Problem 4: NNMF and Feature Extraction

Given a nonnegative matrix $A \in \mathbb{R}^{m\times n}_{\geq 0}$ and a factorization rank $r \leq \min\{m, n\}$, the NNMF problem estimates $X_1 \in \mathbb{R}^{m\times r}_{\geq 0}$ and $X_q \in \mathbb{R}^{r\times n}_{\geq 0}$ such that minimize the optimization problem

$$\min_{X_1,X_2} \|A - X_1X_2\|^2_{fr}. \tag{9}$$

There are several algorithms to estimate X_1 and X_2 from (9) (see, e.g., Chapter 8 in (Gillis, 2020) for more details). One of the most relevant algorithms is the multiplicative update rule developed by Lee and Seung in (Lee & Seung, 1999). This algorithm has been a popular method due to the simplicity of implementation.

As mentioned in (Gillis, 2020), the main reason why NNMF is so popular is because of its ability to extract sparse and easily interpretable factors automatically. Moreover, NNMF has several applications in real life, such as text mining, hyperspectral imaging, computational biology, clustering, music analysis, and community detection.

An application of NNMF is facial feature extraction, which is related to face recognition (see Chapter 12 in (Suykens, Signoretto & Argyriou, 2014) for more details). Here, we consider a gray-level image of a face containing t pixels and squash the data into a single vector such that the j-th entry represents the value of the j-th pixel. Let the rows of A represent the t pixels, and the n columns represent one image. NNMF will produce two matrices X_1 and X_2. The columns of X_2 are interpreted as images, which we refer to as the basis images. X_2 tells us to sum up the basis images to reconstruct an approximation to a given face. As the weights in the linear combinations are nonnegative, these basis images can only be summed up to reconstruct each original image. This process is represented graphically in Figure 4, using the Yale face database set (Georghiades, Belhumeur & Kriegman, 2001).

Problem 5: K-SVD and Filling in Missing Pixels

The K-SVD is a dictionary learning algorithm for creating a dictionary for sparse representations via a singular value decomposition approach. This algorithm is a generalization of the classical k-means clustering method (Hartigan & Wong, 1979). The K-SVD algorithm follows the construction flow of the k-means method, where the sparsity term of the constraint is relaxed so that the number of nonzero entries of each column can be more than 1, but less than a number c. So, the objective function becomes

$$\min_{X_1,X_2} \|A - X_1X_2\|^2_{fr}. \tag{10}$$

where $A \in \mathbb{R}^{m\times n}$, $X_1 \in \mathbb{R}^{m\times r}$ and $X_2 \in \mathbb{T}^{r\times n}_c$. Solution of minimization problem in (10) can be found widely in image processing, audio processing, biology, and document analysis applications.

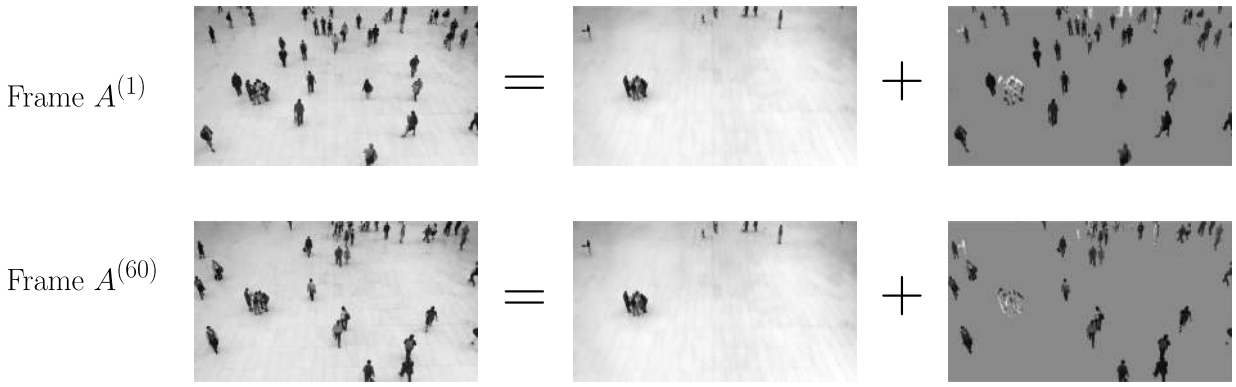


Figure 3. Background modelling results of two 500-frame surveillance video sequences in $A = X_1 + X_2$ mode. This video is taken from a shopping center, with a resolution of 365×648 and learning time of 15 seconds.

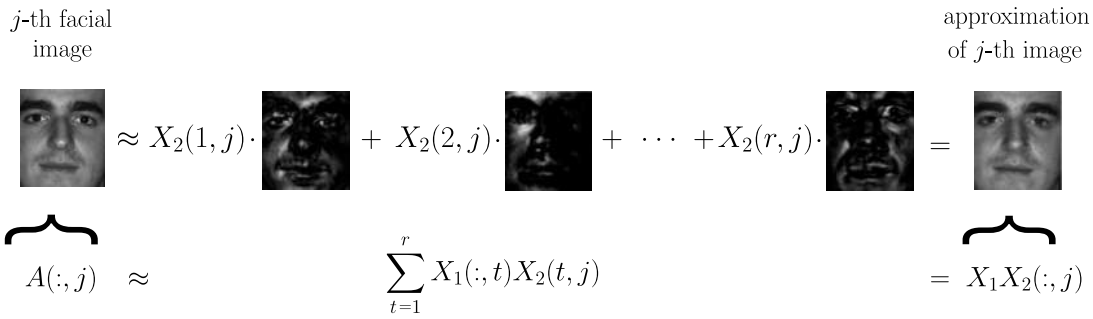


Figure 4. NNMF applied on the Yale face database set with $r = 240$ (4050 images with 640×480 pixels). Image on the left is a column of A reshape as an image

Figure 5. K-SVD applied on the Georgia tech face database set (60 images with 112×88 pixels) with $r = 15$ and $c = 2$. Image on the left is a corrupted image with the missing pixels. Image on the right is the reconstructed image using the K-SVD algorithm.

The K-SVD algorithm is a method developed by (Aharon et al., 2006) that gives a solution to problem (10). The K-SVD method is an iterative method that is divided into two steps. First, X_1 is fixed and each column of X_2 is found, using a compress sensing algorithm (see Chapter 8 in (Eldar & Kutyniok, 2012) for more details). Secondly, X_2 is fixed, and X_1 is found, where the process is to update only one column of the dictionary X_2 each time.

The K-SVD method is used for filling in missing pixels in a grayscale image using a training set of images $\mathcal{A}$. As explained in (Aharon et al., 2006), each column of A is a vectorized 8×8 block from each image in $\mathcal{A}$. Afterward, A is used in the K-SVD algorithm to obtain the X_1 and X_2 that solve the problem (10). Finally, using dictionary X_1, we can fill missing pixels in each 8×8 block from a denoised image. One corrupted image an its reconstruction is shown in Figure 5, using the Georgia face database set (Tech, n.d.).

Overview of the `FroImPro` Toolbox

A complete explanation of the use of the `FroImPro` Toolbox is in the document *Toolbox de aplicaciones de la norma de Frobenius en procesamiento de imágenes*.

Acknowledgements

This work was financially supported by *Vicerrectoría de Investigación y Extensión* from *Instituto Tecnológico de Costa Rica* (Research #1440037).

References

Aharon, M., Elad, M. and Bruckstein, A. (2006). K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation, *IEEE Transactions on signal processing* **54**(11): 4311–4322.

Ahmad, S. S., Ali, I. and Slapnicar, I. (2021). Perturbation analysis of matrices over a quaternion division algebra, *Electronic Transactions on Numerical Analysis* **54**: 128–149.

Bouwmans, T., Aybat, N. and Zahzah, E. (2016). *Handbook of Robust Low-Rank and Sparse Matrix Decomposition: Applications in Image and Video Processing*, CRC Press.

Brbić, M. and Kopriva, I. (2018). Multi-view low-rank sparse subspace clustering, *Pattern Recognition* **73**: 247–258.

Bryt, O. and Elad, M. (2008). Compression of facial images using the k-SVD algorithm, *Journal of Visual Communication and Image Representation* **19**(4): 270–282.

Chavarría-Molina, J., Fallas-Monge, J. J. and Soto-Quiros, P. (2022). Effective implementation to reduce execution time of a low-rank matrix approximation problem, *Journal of Computational and Applied Mathematics* **401**: 113763.

Chung, J. and Chung, M. (2013). Computing optimal low-rank matrix approximations for image processing, *2013 Asilomar Conference on Signals, Systems and Computers*, IEEE, pp. 670–674.

Datta, B. (2010). *Numerical Linear Algebra and Applications, Second Edition*, Other Titles in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104).

Eckart, C. and Young, G. (1936). The approximation of one matrix by another of lower rank, *Psychometrika* **1**(3): 211–218.

Eldar, Y. and Kutyniok, G. (2012). *Compressed Sensing: Theory and Applications*, Compressed Sensing: Theory and Applications, Cambridge University Press.

Friedland, S. and Torokhti, A. (2007). Generalized rank-constrained matrix approximations, *SIAM Journal on Matrix Analysis and Applications* **29**(2): 656–659.

Fritzen, F. and Ryckelynck, D. (2019). *Machine Learning, Low-Rank Approximations and Reduced Order Modeling in Computational Mechanics*, MDPI AG.

Georghiades, A. S., Belhumeur, P. N. and Kriegman, D. J. (2001). From few to many: Illumination cone models for face recognition under variable lighting and pose, *IEEE transactions on pattern analysis and machine intelligence* **23**(6): 643–660.

Gillis, N. (2020). *Nonnegative Matrix Factorization*, Data Science, Society for Industrial and Applied Mathematics.

Gu, Z., Chen, Z., Zhang, Y., Zhu, Y., Lu, M. and Chen, A. (2016). Reducing fingerprint collection for indoor localization, *Computer Communications* **83**: 56–63.

Hartigan, J. A. and Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm, *Journal of the royal statistical society. series c (applied statistics)* **28**(1): 100–108.

Lee, D. D. and Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization, *Nature* **401**(6755): 788–791.

Markovsky, I. (2008). Structured low-rank approximation and its applications, *Automatica* **44**(4): 891–909.

Siahsar, M. A. N., Gholtashi, S., Abolghasemi, V. and Chen, Y. (2017). Simultaneous denoising and interpolation of 2d seismic data using data-driven non-negative dictionary learning, *Signal Processing* **141**: 309–321.

Soto-Quiros, P., Jose Fallas-Monge, J. and Chavarría-Molina, J. (2022). A fast algorithm for image deconvolution based on a rank constrained inverse matrix approximation problem, *Proceedings of Sixth International Congress on Information and Communication Technology*, Springer, pp. 165–176.

Sun, C., Xie, J. and Leng, Y. (2016). A signal subspace speech enhancement approach based on joint low-rank and sparse matrix decomposition, *Archives of Acoustics* **41**.

Suykens, J., Signoretto, M. and Argyriou, A. (2014). *Regularization, Optimization, Kernels, and Support Vector Machines*, Chapman & Hall, CRC Press.

Tech, G. (n.d.). Georgia tech face database [repositorio de datos].
URL: *http://sipi.usc.edu/database/*

Torokhti, A. and Soto-Quiros, P. (2016). Generalized brillinger-like transforms, *IEEE Signal Processing Letters* **23**(6): 843–847.

Wang, X., Che, M. and Wei, Y. (2020). Tensor neural network models for tensor singular value decompositions, *Computational Optimization and Applications* **75**(3): 753–777.

Yuan, Z. and Oja, E. (2005). Projective nonnegative matrix factorization for image compression and feature extraction, *Scandinavian Conference on Image Analysis*, Springer, pp. 333–342.

Zhou, T. and Tao, D. (2011). Godec: Randomized low-rank & sparse matrix decomposition in noisy case, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*.

Anexo 14

TOOLBOX DE APLICACIONES DE LA NORMA DE FROBENIUS

En procesamiento de imágenes

Pablo Soto-Quirós
Instituto Tecnológico de Costa Rica

Jeffry Chavarría-Molina
Instituto Tecnológico de Costa Rica

Juan José Fallas-Monge
Instituto Tecnológico de Costa Rica

CONTENIDOS

SIMBOLOS

$\mathbb{C}^m$	Conjunto de vectores de entradas complejas con m entradas
$\mathbb{C}^{m\times n}$	Conjunto de matrices de entradas complejas con m filas y n columnas
$\|X\|_{fr}$	Norma de Frobenious de la matriz X
$\mathbb{C}_k^{m\times n}$	Conjunto de las matrices de m filas, n columnas y de rango a lo sumo k
$A^\dagger$	Pseudoinversa de Moore-Penrose de la matriz A
A^*	Transpuesta conjugada de la matriz A
I_n	Matriz identidad de orden n
$A_{m\times n}$	Matriz de m filas y n columnas
SVD	Descomposición en valores singulares
$\mathrm{rank}\,(A)$	Rango de la matriz A
$\mathrm{card}\,(A)$	Cardinalidad de la matriz A. Cantidad de entradas distintas de cero.

INTRODUCCIÓN

El presente manual ofrece una descripción detallada del `Toolbox de aplicaciones de la norma de Frobenius en procesamiento de imágenes`. De manera sintetizada, el toolbox aborda las siguientes aplicaciones de la norma de Frobenius, las cuales están organizadas a partir del Capítulo 2 de la siguiente manera:

1. **Capítulo 2**: Construcción de filtros para remover ruido de una imagen.
2. **Capítulo 3**: Factorización no negativa de matrices aplicada a la reconstrucción de imágenes.
3. **Capítulo 4**: Aproximación de la matriz de rango bajo mediante el algoritmo BRP y su aplicación a la compresión de imágenes
4. **Capítulo 5**: Algoritmo `GoDec` aplicado a la detección de variaciones en un conjunto de imágenes.
5. **Capítulo 6**: Algoritmo K-SVD aplicado a la reconstrucción de imágenes con pixeles perdidos.

Como todas las aplicaciones anteriores se relacionan con procesamiento de imágenes, es fundamental tener clara la forma de cómo se representa una imagen computacionalmente. Además de cómo proceder con la representación en caso que se requiera trabajar con una base de imágenes. Sobre esto se detallará a continuación con el objetivo de que el lector se familiarice con la representación matricial que se utilizará en las funciones del toolbox. En concreto, en la Figura I.1 se muestra la representación matricial de un bloque de una imagen en escala de grises. La imagen I.2 muestra el proceso que se debe seguir para

vectorizar, como vector columna, la matriz anterior. Si se realiza el proceso mostrado en las figuras I.1 y I.2, pero a la imagen completa, se termina con una representación vectorizada de toda la imagen (ver [1]).

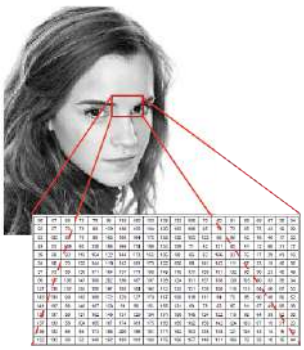


Figura I.1: Representación matricial de un bloque de una imagen.

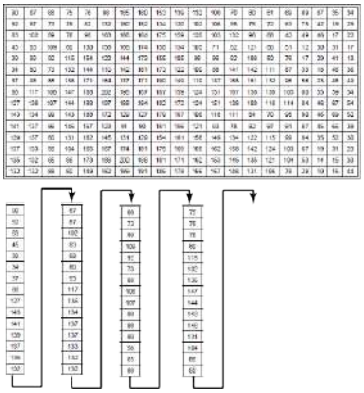
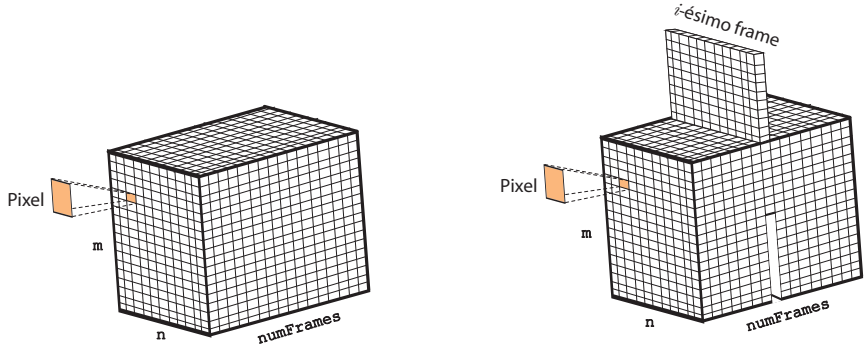


Figura I.2: Vectorización (como columna) de la representación matricial.

Si se cuenta con una base de datos de imágenes, en escala de grises, todas del mismo tamaño, podemos hacer este proceso de vectorización a cada una de las imágenes. A partir de la vectorización de todas la imágenes de la base se diseña una matriz A de tal manera que la información asociada a la $i-$ésima imagen quedará almacena en la $i-$ésima columna de A. Por lo tanto, si cada imagen de la base tiene una representación matricial de dimensión $m \times n$, su vectorización tendrá tamaño $(mn) \times 1$. Si en la base existen z imágenes, entonces la matriz A es de dimensión $(mn) \times z$.

En el caso que se esté procesando un video V, entonces se sigue una codificación matricial similar. En efecto, suponga que V es un video con `numFrames` frames, cada uno de tamaño `m` $\times$ `n` pixeles. Matricialmente V puede representarse mediante un arreglo tridimensional de tamaño `m` $\times$ `n` $\times$ `numFrames`, tal como se muestra en la Figura I.3. Por su parte, las Figuras I.4 y I.5 muestran la vectorización que se realiza para el $i-$ésimo frame como matriz columna de tamaño $mn \times 1$. Si este proceso se repite para todos los frames, el video puede codificarse mediante una matriz A de tamaño `mn` $\times$ `numFrames`, la cual se muestra en la Figura I.6.



(a) Representación matricial de un video. (b) Representación del $i-$ésimo frame.

Figura I.3: Representación tridimensional de un video.

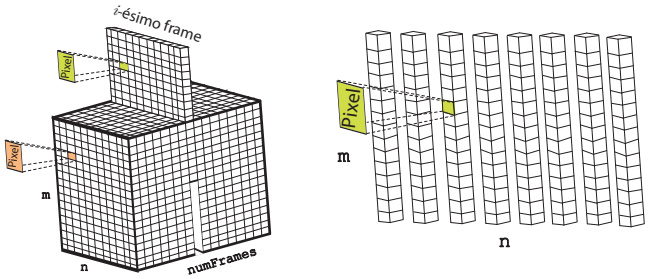


Figura I.4: Procesamiento del $i-$ésimo frame.

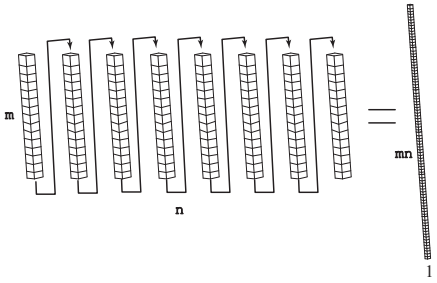


Figura I.5: Vectorización del $i-$ésimo frame.

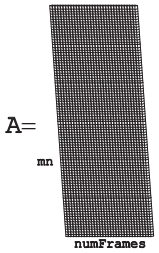


Figura I.6: Construcción de la matriz A.

Capítulo 1

FUNCIONES GLOBALES DEL TOOLBOX

En este capítulo se describen las funciones generales que se utilizan, de manera común, en las diferentes aplicaciones mostradas en este manual. Por su caracter global, estas funciones pueden ser de gran utilidad para cualquier usuario que esté realizando implementaciones en Matlab relacionadas con el procesamiento de imágenes. Procederemos con la descripción de las funciones y su respectiva sintaxis.

1.1 `ReadImageDataBase`

Esta función recibe la dirección de una carpeta con imágenes (la ruta se especifica en el parámetro `Textpath`), todas del mismo tamaño, en el formato especificado y en escala de grises (en caso que no lo esté, la función las convierte a escala de grises). Los formatos admitidos son `.jpg, .pgm, .png, .tif, .bpm.` La función asume como formato por defecto `.jpg`. Retorna los siguientes argumentos:

- `m,n`: Son valores enteros tales que $m \times n$ representa la dimensión de cada una de las imágenes en la carpeta.
- `A`: Es una matriz que contiene en sus columnas la información numérica de las imágenes en la carpeta. La columna i de dicha matriz, corresponde a la $i-$ésima imagen. La matriz `A` tiene dimensión `mn` $\times$ `numImg`.

Nota: No hay restricción en que en la carpeta existan archivos de otra naturaleza u otras carpetas. La función valida y lee únicamente los archivos en el formato especificado.

1.1.1 Sintaxis

```
[A,m,n] = ReadImageDataBase('Textpath','Extension','ext')
```

Donde el parámetro `Extension` es opcional.

1.2 Matrix2Image

Esta función toma una matriz X cuyas columnas corresponden a la vectorización de un grupo de imágenes, una columna por cada imagen. La función reconstruye dichas imágenes y las guarda en un directorio. La función recibe:

- `m` y `n`: Valores enteros tales que $m \times n$ corresponde a la dimensión de cada una de las imágenes.
- `X`: Es una matriz de tamaño `mn` $\times$ `numImage`, tal que cada columna representa una imagen de tamaño `m` $\times$ `n` y `numImage` es el número de imágenes.
- `Textpath`: Ruta de un directorio donde se guardarán las imágenes.
- `Name`: Raíz común de los nombres de las imágenes que se guardarán en `Textpath`. La función guarda en el directorio `Textpath` cada imagen con el nombre `NameYYY.jpg`, donde `YYY` representa una secuencia numérica generada automáticamente por la función.

1.2.1 Sintaxis

```
Matrix2Image(X,m,n,'Textpath','Name');
```

1.3 double2uint8

Esta función convierte los valores de una matriz `X` en `uint8`. Para lo cual cada valor es reescalado en el rango [0,255] de forma lineal. Es decir, el $\min(\texttt{X})$ se asigna a cero (0), mientras que el $\max(\texttt{X})$ se asigna a 255. Los valores intermedios son asignados de forma proporcional.

1.3.1 Sintaxis

```
X = double2uint8(X)
```

1.4 Video2Matrix

Esta función recibe un video y lo codifica como una matriz, lo cual será de gran utilidad en el Capítulo 5. La función recibe los siguientes parámetros:

- `PathTextVideo`: Es la ruta de un archivo de video con extensión `.mp4`.

- `Scale`: Este valor se utiliza para escalar cada uno de los frames en una fracción del tamaño original, debe cumplir que: $0 <$`Scale`≤ 1. Su valor por defecto es 1, en cuyo caso se utilizará el tamaño de los frames del video original.
- `numFrames`: Número de frames a utilizar en el video. Este valor es un entero positivo que debe ser menor o igual que el número total de frames en el video original. De omitirse, se utilizarán todos los frames del video original.

Esta función retorna:

- `X0`: Una matriz de tamaño `mn`$\times$`numFrames`, donde cada columna de `A` corresponde a un frame del video, redimensionado como vector columna y en escala de grises.
- `m1` y `n1`: Tamaño de cada uno de los frames del video, luego del proceso de escalado.

1.4.1 Sintaxis

```
[X0,m1,n1]=Video2Matrix('PathTextVideo','Scale',Scale,'numFrames',numFrames);
```

Los parámetros `Scale` y `numFrames` son opcionales.

1.5 Matrix2Video

Esta función se encarga de construir un video con extensión `.mp4` a partir de una matriz `X` de tamaño `mn`$\times$`numFrames`, donde cada una de las columnas de `X` representa un frame de tamaño `m`$\times$`n`. La función recibe los siguientes argumentos:

- `X`: Matriz de tamaño `mn`$\times$`numFrames`.
- `m` y `n`: Valores enteros tal que el número de filas de `X` corresponde a `mn`. Estos valores se utilizan para redimensionar las columnas de `X` a una matriz de tamaño `m`$\times$`n`. De manera que el $i-$ésimo frame del video corresponde a la $i-$ésima columna de `X`, redimensionada a tamaño `m`$\times$`n`.
- `Path\Name`: Representa la dirección del directorio donde será guardado el archivo de video con el nombre `Name.mp4`. En caso que `Path` se omita, entonces la función guardará el archivo de video en el directorio de trabajo actual de Matlab.

1.5.1 Sintaxis

```
Matrix2Video(X,m,n,'Path\Name')
```

1.6 img_miss_pix

Esta función recibe la dirección de una imagen y un escalar `porc`, tal que $0 <$ `porc` < 1, y genera una imagen, en el mismo directorio, con la cantidad de pixeles perdidos indicados por `porc`. Por ejemplo, si `porc` $= 0.25$ significa que se va a generar una imagen con 25% de pixeles perdidos.

- `Textpath:` Es la ruta de la imagen.
- `porc:` Escalar que denota el porcentaje de pixeles perdidos que tendrá la nueva imagen.

1.6.1 Sintaxis

```
img_miss_pix('Textpath',porc);
```

Capítulo 2

FILTRO PARA REMOVER RUIDO

Esta primera aplicación supone que se cuenta con una base de datos de imágenes, la cual se codifica matricialmente en una matriz A, mediante la aplicación de la función `ReadImageDataBase` (ver la función 1.1). A partir de A se genera otra matriz C con la aplicación de algún tipo de ruido (sobre esto se detallará en la sección 2.2). Así que C se puede conceptualizar como la matriz de imágenes asociadas a la base de datos, pero con ruido. El objetivo de la aplicación es determinar una matriz X de rango reducido, que se denomina un *filtro*, que permitirá limpiar el ruido a una nueva imagen que no es parte de la base de datos, pero que tiene cierta similitud con las imágenes en ella. La calidad del resultado dependerá de factores como qué tan similar o no es la imagen que se está tratando de limpiar, en comparación con las imágenes en la base, y la similitud del tipo de ruido que tiene la imagen, en comparación con el tipo de ruido que se utilizó para construir a la matriz C. Si dicha imagen con ruido se vectoriza y se representa con $\widetilde{w}$, entonces la imagen filtrada quedará determinada por la vectorización:

$$w = X \cdot \widetilde{w}$$

Como problema de optimización, la aplicación se formula de la siguiente manera. Dadas $A \in \mathbb{C}^{m\times n}$, $B \in \mathbb{C}^{m\times p}$ y $C \in \mathbb{C}^{q\times n}$, se busca encontrar la matriz $X \in \mathbb{C}^{p\times q}$ que de solución al problema de minimización:

$$\min_{X\in\mathbb{C}_k^{m\times n}} \|A - BXC\|_{fr},$$

tal que $\|X\|_{fr}$ sea mínima también. Si se toma $m = p$ y $B = I_m$, el problema se convierte en:

$$\min_{X \in \mathbb{C}_k^{m \times n}} \|A - XC\|_{fr} \tag{2.1}$$

La solución del problema (ver [2]) 2.1 está dada por:

$$X = (A \cdot C^{\dagger} \cdot C)_k \cdot C^{\dagger},$$

y

$$(A \cdot C^{\dagger} \cdot C)_k = \sum_{i=1}^{k} (\sigma_i u_i v_i^*) \quad \text{si } 1 \leq k \leq \operatorname{rank}(A \cdot C^{\dagger} \cdot C)$$

En este último caso, σ_i es el i-ésimo valor singular de la matriz $A \cdot C^{\dagger} \cdot C$. Además, u_i y v_i son los vectores singulares por izquierda y por derecha, respectivamente, asociados a la SVD de la matriz $A \cdot C^{\dagger} \cdot C$.

A continuación se procede con la descripción de las funciones asociadas al problema, junto con su respectiva sintaxis.

2.1 `RankConstrainedFilterX`

Esta función calcula la matriz X de rango reducido que es la solución del problema 2.1.

2.1.1 Sintaxis

```
X=RankConstrainedFilterX(A,C,k)
```

Donde $A \in \mathbb{C}^{m \times n}$, $C \in \mathbb{C}^{q \times n}$ y $X \in \mathbb{C}^{m \times q}$. Además, el parámetro k es la restricción para el rango.

Ejemplo

```
clc; clear; close all
A = [-1 1 2; 3 0 -1; 2 -2 3;4 0 3];
C=[-2 1 6; -2 4 5];
k=2;
X=RankConstrainedFilterX(A,C,k)
```

Lo anterior genera:

```
X =
 0.1671    0.2145
-0.3042    0.0125
 1.0623   -0.8304
 0.4589   -0.2319
```

2.2 `NoiseFunction`

Esta función recibe la dirección de una carpeta con imágenes (la ruta se especifica en el parámetro `Textpath`), todas del mismo tamaño y en escala de grises (en caso que no lo

esté, la función las convierte a escala de grises). Utiliza la función `ReadImageDataBase` (ver sección 1.1) y en ella puede consultar los formatos admitidos para las imágenes. Retorna los siguientes argumentos:

- `A`: Es una matriz que contiene en sus columnas la información numérica de las imágenes en la carpeta. La columna i de dicha matriz, corresponde a la $i-$ésima imagen. La matriz `A` tiene dimensión `mn` $\times$ `numImg`. La descripción de los parámetros `m, n` y `numImg` debe ser consultada en la sección 1.1.
- `C`: Es una matriz que se genera a partir de `A`, mediante la aplicación de un ruido. Las opciones y parámetros asociados al ruido, que se utilizan para construir la matriz `C`, son:
  - `gaussian`, `meangauss` y `sigmagauss` que representan, respectivamente, que se utilizará ruido blanco gaussiano con media `meangauss` y varianza `sigmagauss`. Los valores por defecto para la media y la varianza son `meangauss` $= 0$ y `sigmagauss` $= 0.01$, respectivamente.
  - `s&p` y `d`, que representan, respectivamente, que se utilizará el ruido denominado *Salt and pepper*, y `d` es la densidad de dicho ruido. El valor por defecto para `d` es `d` $= 0.05$.
  - `speckle` y `sigmaspeckle` que representan, respectivamente, que se utilizará el ruido denominado *speckle*, con una varianza `sigmaspeckle`. El valor por defecto para `sigmaspeckle` es `sigmaspeckle` $= 0.05$.
- `m, n`: Son valores enteros tales que $m \times n$ representa la dimensión de cada una de las imágenes en la carpeta.

2.2.1 Sintaxis

```
[C,A,m,n]=NoiseFunction('Textpath')
```

Al utilizar la sintaxis anterior se creará la matriz `C`, a partir de `A`, utilizando por defecto un ruido gaussiano, con `meangauss` $= 0$ y `sigmagauss` $= 0.01$. La función `NoiseFunction` tiene argumentos opcionales que pueden ser personalizados por el usuario, en cuanto a los tres tipos de ruido disponibles (`gaussian`, `s&p` y `speckle`), así como los parámetros inherentes a cada uno de ellos. Esto se detallará a continuación.

2.2.2 Sintaxis alternativa

A continuación se le muestran los tres tipos adicionales de sintaxis que puede utilizar para esta función. En todos los casos, si solo se especifica el tipo de ruido y no se incluyen los parámetros adicionales, entonces la función utiliza los valores definidos por defecto.

```
[C,A,m,n]=NoiseFunction('Textpath','NoiseOption','gaussian','sigmagauss',n_0,'meangauss',n_1);
```

En este caso se aplica, para la construcción de la matriz `C`, ruido blanco gaussiano, con la desviación y media que especifique en los valores `n_0` y `n_1`, respectivamente.

```
[C,A,m,n]=NoiseFunction('Textpath','NoiseOption','s&p','d',n_0);
```

Por su parte, con la sintaxis anterior, para construir a `C`, se aplica el tipo de ruido *Salt and pepper*, con la densidad que se especifique en `n_0`.

```
[C,A,m,n]=NoiseFunction('Textpath','NoiseOption','speckle','sigmaspeckle',n_0);
```

Finalmente, con la sintaxis anterior, para construir a `C`, se aplica el tipo de ruido *speckle*, con la desviación que se especifique en `n_0`.

2.3 FullRankConstrainedFilterX

Esta función es una versión más completa de `RankConstrainedFilterX` (ver sección 2.1), aplicada específicamente para procesamiento de imágenes. Esta función recibe una dirección de una carpeta con imágenes(la ruta se especifica en el parámetro `Textpath`), todas del mismo tamaño y en escala de grises (en caso que no lo esté, la función las convierte a escala de grises). Esta función utiliza las funciones `ReadImageDataBase` (ver sección 1.1) y `NoiseFunction` (ver sección 2.2). En la función `ReadImageDataBase` puede consultar los formatos admitidos para las imágenes. Retorna los siguientes argumentos:

- `m, n`: Son valores enteros tales que $m \times n$ representa la dimensión de cada una de las imágenes en la carpeta.
- `A`: Es una matriz que contiene en sus columnas la información numérica de las imágenes en la carpeta. La columna i de dicha matriz, corresponde a la $i-$ésima imagen. La matriz `A` tiene dimensión `(mn)` $\times$ `numImg`, donde `numImg` es la cantidad de imágenes en formato `.jpg` que hay en la carpeta.
- `X`: Es la matriz de rango reducido que es la solución del problema 2.1. Corresponde al filtro construido a partir de la función `NoiseFunction` (ver sección 2.2).

2.3.1 Sintaxis

```
[X,m,n,A]=FullRankConstrainedFilterX('Textpath')
```

2.3.2 Sintaxis alternativa

La función `FullRankConstrainedFilterX` cuenta con parámetros adicionales que le permiten al usuario personalizar los siguientes valores:

- `k`: Es el valor que restringe el rango de la matriz `X`. Se debe cumplir que

$$0 < \mathtt{k} \leq \min\{\mathtt{m} \cdot \mathtt{n}, \mathtt{numImg}\}$$

  Si este valor no se especifica, se toma el valor por defecto $\mathtt{k} = \min\{\mathtt{m} \cdot \mathtt{n}, \mathtt{numImg}\}$.

- `NoiseOption`: Permite personalizar parámetros asociados al ruido. Dado que `FullRankConstrainedFilterX` utiliza a la función `NoiseFunction`, entonces se pueden personalizar todos los parámetros incluidos en esa función. Para más detalles, ver la sección 2.2, específicamente en la descripción de la matriz `C`.

A continuación se le muestran algunas alternativas de sintaxis para esta función:

```
[X,m,n,A]=FullRankConstrainedFilterX('Textpath','k',k_0,'NoiseOption','gaussian','sigmagauss',n_0,'
    meangauss',n_1);
```

En este caso se aplica, para la construcción de la matriz `C`, ruido blanco gaussiano, con la desviación y media que especifique en los valores `n_0` y `n_1`, respectivamente.

```
[X,m,n,A]=FullRankConstrainedFilterX('Textpath','k',k_0,'NoiseOption','s&p','d',n_0);
```

Por su parte, con la sintaxis anterior, para construir a `C`, se aplica el tipo de ruido *Salt and pepper*, con la densidad que se especifique en `n_0`.

```
[X,m,n,A]=FullRankConstrainedFilterX('Textpath','k',k_0,'NoiseOption','speckle','sigmaspeckle',n_0);
```

Finalmente, con la sintaxis anterior se aplica el tipo de ruido *speckle*, con la desviación que se especifique en `n_0`.

2.4 SaveBlurredImages

Esta función toma una base de datos de imágenes en escala de grises (en caso que no lo esté, la función las convierte a escala de grises), todas del mismo tamaño y permite crear otra base de imágenes, a partir de la anteriores, pero con ruido. La función recibe la dirección de una carpeta con imágenes (parámetro `pathOriginal`) y otra dirección de la carpeta en la que escribirá las imágenes con ruido (parámetro `pathWriteNoise`). Esta función utiliza la función `NoiseFunction`, entonces se pueden personalizar todos los parámetros adicionales incluidos en esa función. Para más detalles, ver la sección 2.2, específicamente en la descripción de la matriz `C` y los formatos admitidos para las imágenes.

2.4.1 Sintaxis

```
SaveBlurredImages('pathOriginal','pathWriteNoise');
```

Al utilizar la sintaxis anterior se crearán las imágenes con ruido, utilizando los parámetros por defecto. En este caso, utilizando por defecto un ruido gaussiano, con `meangauss` $= 0$ y `sigmagauss` $= 0.01$. En el caso que se quiera personalizar el tipo de ruido y sus parámetros, se debe utilizar alguna de las sintaxis que se muestran a continuación.

2.4.2 Sintaxis alternativa

```
SaveBlurredImages('pathOriginal','pathWriteNoise','NoiseOption','gaussian','sigmagauss',n_0,'
    meangauss',n_1);
```

En este caso, para la construcción de las nuevas imágenes, se aplica ruido blanco gaussiano, con la desviación y media que especifique en los valores `n_0` y `n_1`, respectivamente.

```
SaveBlurredImages('pathOriginal','pathWriteNoise','NoiseOption','s&p','d',n_0);
```

Por su parte, en este caso se aplica el tipo de ruido *Salt and pepper*, con la densidad que se especifique en `n_0`.

```
SaveBlurredImages('pathOriginal','pathWriteNoise','NoiseOption','speckle','sigmaspeckle',n_0);
```

Finalmente, con la sintaxis anterior se aplica el tipo de ruido *speckle*, con la desviación que se especifique en `n_0`.

2.5 SaveFilteredImages

Esta función recibe una base de datos de imágenes en escala de grises (en caso que no lo esté, la función las convierte a escala de grises), todas del mismo tamaño y con ruido. Luego, la función permite crear otra base de imágenes, con las imágenes filtradas. La función recibe la dirección de una carpeta con imágenes con ruido (`pathNoise`) y otra dirección de la carpeta en la que escribirá las imágenes filtradas (`pathWriteFiltered`). Utiliza a `ReadImageDataBase` (ver sección 1.1), y en ella puede consultar los formatos admitidos para las imágenes.

Es recomendable utilizarla posterior a la función `FullRankConstrainedFilterX`, dado que utiliza la matriz `X` de rango reducido que es la solución del problema 2.1, y que corresponde al filtro construido para limpiar a las imágenes con ruido.

2.5.1 Sintaxis

```
SaveFilteredImages('pathNoise','pathWriteFiltered',X);
```

Capítulo 3

FACTORIZACIÓN NO NEGATIVA APLICADA A LA RECONSTRUCCIÓN DE IMÁGENES

Esta segunda aplicación también supone que se cuenta con una base de datos de imágenes, la cual se codifica matricialmente en una matriz A, mediante la aplicación de la función `ReadImageDataBase` (ver la función 1.1). La aplicación consiste en determinar matrices W y H, de entradas no negativas, tales que $A \approx WH$. En el contexto del procesamiento de imágenes, las columnas de la matriz W pueden verse como una base para el espacio de las caras. Y la matriz H son los coeficientes de las combinaciones lineales que permiten reconstruir cualquier cara del conjunto original de imágenes, utilizando a la matriz W. Esto es, al multiplicar W por la columna i de H se obtiene la columna i de A, que es la vectorización de la $i-$ésima imagen. Adicionalmente, si se tiene una imagen que no pertenece al conjunto de imágenes almacenadas en A, pero que guardan cierta similitud con ellas, entonces es posible utilizar la matriz W para realizar una reconstrucción de la imagen. En efecto, si ω es la vectorización de la imagen que se va a reconstruir, entonces el producto:

$$W^{\dagger} \cdot \omega \tag{3.1}$$

genera una vectorización de la imagen reconstruida.

Para contextualizar mejor esta aplicación y antes de proceder con detalles más técnicos, se mostrará un ejemplo del uso de las funciones utilizando para ello un grupo de imágenes extraídas de **the extended Yale face database B** ([3]), las cuales son de tamaño 192×168. Las 570 imágenes utilizadas para construir la matriz A, que es de tamaño 32256×570, pueden ser descargadas del siguiente **vínculo**. Al ejecutar el siguiente código (luego de la introducción se muestra el detalle de las funciones utilizadas):

```
[W,H,m,n]=ConstructionNNFM('BaseParaTrabajar570Caras','IteraMax',1000);
```

se obtiene la matriz W de tamaño 32256×285 que corresponde a la base de caras (285 caras en total, que corresponde al valor de r por defecto calculado para esta matriz). Una parte de dicha base de caras se muestra en la Figura 3.1.

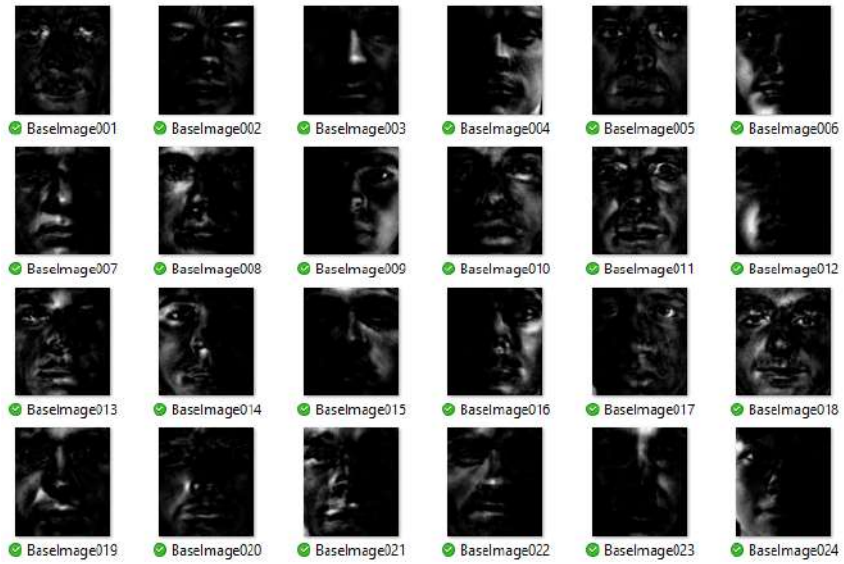


Figura 3.1: Parte de la base de caras.

Por otra parte, en la carpeta disponible **aquí** se incluyeron 39 archivos que no forman parte de las 570 imágenes usadas para construir la matriz A, pero tienen cierta similitud con dichas imágenes, y por ello pueden ser utilizadas para probar el proceso de reconstrucción. En nuestro caso usaremos la imagen *yaleB39(1).jpg*, que se muestra en la Figura 3.2.

Figura 3.2: Imagen externa *yaleB39(1).jpg* que será reconstruida.

Al ejecutar el código siguiente:

```
ReconstructionExternalFace('BaseCaras_Yale_JJ\1caraDeCadaPersona\yaleB39(1).jpg',W,'Imagen');
```

se utiliza la fórmula mostrada en 3.1 y se obtiene como resultado la reconstrucción que se muestra en la Figura 3.3:

Figura 3.3: Reconstrucción de la imagen *yaleB39(1).jpg* utilizando la matriz W.

Como problema de optimización, la factorización no negativa parte de una matriz $A_{m\times n}$ y busca determinar matrices $W_{m\times r}$ y $H_{r\times n}$ ($W, H \geq 0$) que sean solución de:

$$\min_{W,H\geq 0} \|A - WH\|_{fr}^2, \tag{3.2}$$

donde $0 < r \leq \min\{m, n\}$. La solución del problema anterior se puede realizar mediante un método iterativo aplicando las *multiplicative update rules* (ver [4]), las cuales aplican las siguientes reglas de actualización para las matrices H y W:

$$H_{au} = H_{au} \cdot \frac{(W^T A)_{au}}{(W^T WH)_{au}}, \qquad W_{ia} = W_{ia} \cdot \frac{(AH^T)_{ia}}{(WHH^T)_{ia}}$$

En [4] se demuestra que el método iterativo propuesto converge a un punto estacionario de la distancia: $\|A - WH\|_{fr}$. A continuación se muestra el pseudocódigo.

Algoritmo 1: Pseudocódigo general.

Result: matrices W y H
Inicialice W y H;
for *z=1: iteraciones* **do**
  for *a=1:r* **do**
    for *u=1:n* **do**
      $H_{au} \leftarrow H_{au} \cdot \frac{(W^T A)_{au}}{(W^T WH)_{au}}$
    end
    for *i=1:m* **do**
      $W_{ia} \leftarrow W_{ia} \cdot \frac{(AH^T)_{ia}}{(WHH^T)_{ia}}$
    end
  end
end

Ahora procederemos con la descripción de las funciones que son parte de esta aplicación.

3.1 `nnfLS`

Esta función usa las *multiplicative update rules* para realizar una factorización no negativa de la matriz A, de tal manera que $A \approx WH$.

Esta función recibe:

- `A`: Es la matriz que se desea factorizar, de tamaño $mA \times nA$.
- `r`: Es la restricción para el rango. El valor por defecto es $\lfloor \frac{\min\{mA, nA\}}{2} \rfloor$, donde $\lfloor \cdot \rfloor$ denota la función piso.
- `IteraMax`: Número máximo de iteraciones. Su valor por defecto es $2 \cdot nA$.
- `Tol`: Es la tolerancia para el error. Su valor por defecto es 10^{-6}.

3.1.1 Sintaxis

```
[W,H,VectorError] = nnfLS(A,'r',r,'IteraMax',IteraMax,'Tol',Tol)
```

Donde los parámetros `r`, `IteraMax` y `Tol` son opcionales.

Ejemplo

Considere la matriz:

```
A=[1 2 5; 2 3 5; 6 7 1; 6 1 0];
```

Al ejecutar la función con

```
rng(1); %Se establece la semilla en 1 para poder replicar el experimento.
[W,H,VectorError]=nnfLS(A,'r',3,'IteraMax',1000)
```

Se obtiene:

```
W =

    1.0041    0.0007    0.3556
    0.9744    0.0690    0.5137
    0.0000    0.3967    1.1398
    0.0003    0.4070    0.0000

H =

    0.9378    0.1176    4.6689
   14.7401    2.4567    0.0000
    0.1341    5.2866    0.8774
```

Y luego, al realizar el producto WH se obtiene:

```
ans =

    1.0000    2.0000    5.0000
    2.0000    3.0000    5.0000
    6.0000    7.0000    1.0000
    6.0000    1.0000    0.0012
```

3.2 ConstructionNNFM

Utiliza la función `ReadImageDataBase` (ver la sección 1.1) para leer una carpeta de imágenes y construir la matriz `A`, que contiene la vectorización de las imágenes en sus columnas. Posteriormente, usa la función `nnfLS` (ver la sección 3.1) para construir las matrices W y H, las cuales se almacenan en la carpeta "Results" (si la carpeta no existe, la función la creará automáticamente) como archivos independientes en formato `.mat`. Finalmente, la función genera una carpeta dentro de "Results" de nombre "ImagesBase", en la cual se guardarán las imagénes generadas a partir de la matriz W y que corresponden a las imágenes de la base del espacio de caras.

Esta función recibe:

- `Extension`: Es la extensión del conjunto de imágenes que se van leer. Los formatos admitidos son `.jpg, .pgm, .png, .tif, .bpm.` La función asume como formato por defecto `.jpg`.
- `r`: Es la restricción para el rango. El valor por defecto es $\lfloor \frac{\min\{mA, nA\}}{2} \rfloor$, donde $\lfloor \cdot \rfloor$ denota la función piso.
- `IteraMax`: Número máximo de iteraciones. Su valor por defecto es $2 \cdot nA$.
- `Tol`: Es la tolerancia para el error. Su valor por defecto es 10^{-6}.

Además de las matrices W y H la función devuelve los valores m y n, tales que $m \times n$ representa la dimensión de cada una de las imágenes.

3.2.1 Sintaxis

```
[W,H,m,n] = ConstructionNNFM('Textpath','Extension','ext','r',r,'IteraMax',itera,'Tol',tol)
```

Donde los parámetros `Extension`, `r`, `IteraMax` y `Tol` son opcionales.

3.3 ReconstructionExternalFace

Para utilizar esta función ya debe tenerse calculada la matriz W, ya sea directamente con la función `nnfLS` (ver la sección 3.1) o con la función `ConstructionNNFM` (ver la sección 3.2). La función `ReconstructionExternalFace` recibe un path de una imagen que no forma parte del grupo original de imágenes, pero guarda cierta similitud con ellas. Luego, usa la matriz W para reconstruir dicha imagen. Este proceso permite evaluar visualmente la calidad de la matriz W, que corresponde a la base del espacio de caras. La función `ReconstructionExternalFace` guardará la nueva imagen en formato `.jpg`, en el directorio "Results" dentro de la carpeta de trabajo. Si dicha carpeta no existe, la crea automáticamente. La función recibe:

- `Textpath`: Ruta de la imagen que se quiere reconstruir.
- `W`: Es la matriz que corresponde a la base del espacio de caras, obtenida a partir de la función `nnfLS` (ver la sección 3.1) o con la función `ConstructionNNFM` (ver la sección 3.2).
- `NameOutput`: Es el nombre que se utilizará para guardar la imagen reconstruida.

3.3.1 Sintaxis

```
ReconstructionExternalFace('Textpath',W,'NameOutput');
```

3.4 ReconstructionExternalFacesDirectory

Esta función extiende a `ReconstructionExternalFace` (ver la sección 3.3) a un directorio de imágenes por reconstruir, las cuales no forman parte del grupo inicial de imágenes, pero guardan cierta similitud con ellas. Los parámetros de esta función son:

- `Textpath`: Es la dirección de la carpeta de las imágenes que desea resconstruir.
- `W`: Es la matriz que corresponde a la base del espacio de caras, obtenida a partir de la función `nnfLS` (ver la sección 3.1) o con la función `ConstructionNNFM` (ver la sección 3.2).
- `NameOutput`: Es la raíz común que se utilizará para guardar las imágenes reconstruidas, al cual se le agrega un consecutivo numérico. El valor por defecto es `Reconstruted`.
- `Extension`: Es la extensión que se debe utilizar para guardar las imágenes reconstruidas. Los formatos admitidos son `.jpg, .pgm, .png, .tif, .bpm`. La función asume como formato por defecto `.jpg`.

3.4.1 Sintaxis

```
ReconstructionExternalFacesDirectory('Textpath',W,'Extension','ext','NameOutput','name')
```

Donde los parámetros `Extension` y `NameOutput` son opcionales.

Capítulo 4

APROXIMACIÓN DE LA MATRIZ DE RANGO BAJO MEDIANTE EL ALGORITMO BRP Y SU APLICACIÓN A LA COMPRESIÓN DE IMÁGENES

La aproximación de rango bajo de una matriz $L \in \mathbb{R}^{m\times n}$ es un problema de optimización que busca aproximar a L con una matriz $\widehat{L}_r \in \mathbb{R}^{m\times n}$ tal que

$$\|L - \widehat{L}_r\|_{fr}^2 = \min_{L_r \in \mathbb{R}_r^{m\times n}} \|L - L_r\|_{fr}^2$$

Si $L = U_L \Sigma_L V_L^T$ es la descomposición en valores singulares (SVD) de L, entonces $\widehat{L}_r$ queda determinada por la SVD r-truncada (ver [5]), i.e,:

$$\widehat{L}_r = \lfloor L \rfloor_r = U_{L,r} \Sigma_{L,r} V_{L,r}^T \tag{4.1}$$

La SVD es altamente precisa, pero tiene una alta complejidad computacional para matrices grandes [6]. Por ese motivo, se utilizará un método alternativo para aproximar la matriz $\widehat{L}_r$, denominado *proyecciones aleatorias bilaterales* (bilateral random projection method, cuyas siglas en inglés son BRP), el cual fue desarrollado por [7] y mejorado mediante un esquema de potencias, bajo ciertas condiciones de invertibilidad, por Zhou and Tao (ver [8]). En [9] se extiende el esquema de potencias introducido por Zhou and Tao, relajando las condiciones de invertibilidad. Con respecto a lo anterior, si $A_1 \in \mathbb{R}^{n\times r}$ es una matriz cualquiera de rango completo, entonces se define la matriz $Y_2 = (L^T L)^c A_1$ de tamaño $n \times r$, donde c es un parámetro entero y positivo que se utiliza para mejorar la generación de las matrices de proyección bilateral, y puede ser ajustado por el usuario. En nuestro caso fijamos $c = 3$, pero este será un parámetro opcional de las funciones del Toolbox, de manera que el usuario puede explorar otras opciones. Luego, se calcula la descomposición

QR de Y_2, de tal manera que $Y_2 = Q_{n\times r}R_{r\times r}$. Finalmente, el algoritmo BRP aproxima a $\widehat{L}_r$ por:

$$\widehat{L}_r \approx LQ_rQ_r^T,$$

donde $Q_r = Q(:, 1:r)$, esto es, la matriz construida con las primeras r columnas de Q. Si se define $A_{m\times r} = LQ_r$ y $B_{r\times n} = Q_r^T$, entonces podemos almacenar las matrices A y B, a partir de las cuales se puede reconstruir una aproximación para la matriz de rango bajo, quien a su vez corresponde a una aproximación de la matriz L.

Ejemplo aplicado a la compresión de imágenes

Para contextualizar la presente aplicación de la matriz de rango bajo en la compresión de imágenes y su aproximación mediante el algoritmo BRP, desarrollaremos un ejemplo tomando como base la siguiente imagen de un **soldado**, que es de tamaño 3376×6000, y que fue tomada con fines ilustrativos del sitio **Wa11papers.com**. Si se representa dicha imagen mediante la matriz L, entonces $L \in \mathbb{R}^{3376\times 6000}$ y al ejecutar la instrucción:

```
1 ImageCompression('images/soldado.jpg',0.01,'BRP');
```

se obtienen las matrices $L_{3376\times 6000}$ (que representa la imagen original), $A_{3376\times 33}$ y $B_{33\times 6000}$, las cuales son almacenadas en archivos de texto. El archivo asociado a L tiene un tamaño de $149MB$, mientras que los tamaños de los archivos de texto asociados a A y B son $868kb$ y $1.91MB$, respectivamente. Con ello se observa que la suma de los tamaños de estos últimos archivos es significativamente menor que el tamaño del archivo asociado a L. En la Figura 4.1 se muestra la imagen original del soldado construida con la matriz L, mientras que en la Figura 4.2 se muestra la imagen reconstruida utilizando las matrices A y B, usando un radio de compresión del 1%, esto es, apenas se está utilizando un 1% del número de valores singulares no nulos, que para este caso son 33. Obviamente, si el radio de compresión se aumenta, el tamaño de los archivos aumentará, pero la calidad de la reconstrucción mejorará. Por ejemplo, en la Figura 4.3 se muestra la reconstrucción de la imagen usando un radio de compresión del 5% (en este caso el archivo de texto de A tiene un tamaño de $4.60MB$, mientras que el de B es $9.66MB$).

Figura 4.1: Imagen original del soldado.

Con respecto a los tiempos de ejecución, la construcción de las matrices A y B para la imagen del **soldado** tomó un tiempo promedio de 1.22 segundos. Para observar la ganancia en términos de tiempos de ejecución del algoritmo BRP, con respecto a la fórmula 4.1, se repitió el experimento utilizando dicha fórmula y tomando $A = U_{L,r}\Sigma_{L,r}$ y $B = V_{L,r}^T$. En la Figura 4.4 se muestra el resultado de la compresión usando la SVD truncada y un radio de compresión del 5%. Como se puede ver, las diferencias entre las Figuras 4.3 y

Figura 4.2: Imagen del soldado luego de la compresión, usando un radio de compresión del 1%.

Figura 4.3: Imagen del soldado luego de la compresión, usando un radio de compresión del 5%.

4.4 son casi imperceptibles. Sin embargo, la diferencia fundamental está en el tiempo de ejecución. Para este caso, la construcción de las matrices A y B tomó un tiempo promedio de 35.93 segundos. Con ello se refuerza el hecho que para la aproximación de la matriz de rango bajo es mucho más rápido si se realiza con el algoritmo BRP, en lugar de la SVD truncada.

Figura 4.4: Imagen del soldado luego de la compresión con la SVD truncada, usando un radio de compresión del 5%.

Ahora se procederá con la descripción de las funciones del toolbox.

4.1 LowRankMatrixBRP

Esta función recibe una matriz L y calcula una aproximación de rango bajo usando el método BRP. Los parámetros de esta función son:

- `L`: Es la matriz que se quiere aproximar con la matriz de rango bajo usando el método BRP. Su dimensión es $m \times n$.
- `r`: Condición para el rango tal que $0 < r <= \text{rank}(L)$. El valor por defecto se toma como `floor(min(m,n)/2)`.
- `c`: Parámetro del esquema de potencias utilizado por el método, se utiliza para mejorar la generación de las matrices de proyección bilateral (ver [10]). Debe ser entero, positivo y podría ser ajustado por el usuario. Su valor por defecto es 3.

El algoritmo retorna las matrices A y B, de tal manera que luego el usuario puede reconstruir la aproximación de rango bajo de L mediante el producto $A \cdot B$.

4.1.1 Sintaxis

```
[A,B]=LowRankMatrixBRP(L,'r',r,'c',c);
```

Donde los parámetros `r` y `c` son opcionales. En caso de ser omitidos, se tomarán sus valores por defecto.

4.2 ImageCompression

Esta función recibe una imagen y la comprime utilizando un radio de compresión indicado por el usuario, el cual corresponde al porcentaje de valores singulares no nulos que se deben utilizar. El radio de compresión es un decimal d tal que $\frac{1}{\min\{m,n\}} \leq d \leq 1$. Los parámetros de esta función son:

- `Textpath`: Es la dirección de la imagen que desea comprimir.
- `radio`: Corresponde al porcentaje de valores singulares no nulos que se deben utilizar. Es un decimal d tal que $\frac{1}{\min\{m,n\}} \leq d \leq 1$.
- `Opcion`: Etiqueta con la que se indica si se desea aplicar el método BRP o la SVD truncada. Las etiquetas válidas con `'BRP'` y `'SVD'`.

Al utilizar esta función en la carpeta "Results", dentro del directorio de trabajo, se guardará la imagen original, la imagen reconstruida y la información de las matrices A y B en dos archivos de texto, que son las que representan la compresión. Si la carpeta "Results" no existe, la crea automáticamente.

4.2.1 Sintaxis

```
ImageCompression('Textpath',radio,'Opcion');
```

Capítulo 5

ALGORITMO GODEC APLICADO A LA DETECCIÓN DE VARIACIONES EN UN CONJUNTO DE IMÁGENES

En este capítulo se mostrarán dos aplicaciones del algoritmo `GoDec` relacionadas con la detección de variantes en un conjunto de imágenes. En la primera de ellas (que denominaremos `GoDec-Video`) se aplica `GoDec` para la detección de movimientos en un video grabado sobre un fondo estático, para obtener dos videos nuevos, uno con el fondo libre de los objetos en movimiento y otro en donde únicamente se muestren los objetos en movimiento, sin el fondo. En la segunda aplicación que se mostrará de `GoDec` (que denominaremos `GoDec-Imágenes`) se cuenta con una lista de imágenes de un rostro, de tal manera que todas ellas tienen algún tipo de sombra o reflejo que impide ver el rostro completo. La aplicación consiste en la reconstrucción del rostro pero libre de sombras y reflejos.

`GoDec-Video`

Luego de codificar matricialmente un video V, siguiendo los pasos expuestos en la introducción de este manual, se obtiene una matriz `A` que contiene la información de los frames del video. La aplicación del algoritmo `GoDec` a la matriz `A`, bajo ciertas condiciones, permite determinar una matriz L de rango reducido y una matriz S de baja cardinalidad o esparcida, de tal manera que $A \approx L + S$. Si se invierte el proceso a partir del cual se construyó la matriz A, pero aplicándolo a las matrices L y S, se pueden generar, respectivamente, los videos que muestran de manera separada el fondo fijo y los objetos en movimiento en V. A continuación se le muestra un ejemplo concreto generado con la instrucción:

```
GoDecVideoFull('GoDec\VideoWalking.mp4',2,2211840)
```

Donde los valores 2 y 2211840 utilizados en este ejemplo corresponden, respectivamente, a la restricción para el rango máximo de la matriz `L` y a la restricción máxima para la cardinalidad de `S`. La Figura 5.1 muestra el mismo frame para el video original y para los videos generados con las matrices L y S. La imagen generada con L corresponde a los objetos que no presentan movimiento en el frame correspondiente en el video original, mientras que la imagen generada con S presenta los objetos que sí estaban en movimiento. Es natural pensar que la visualización de los tres videos permitiría una comprensión más clara de la separación que se logra con el algoritmo GoDec. Por ese motivo en el siguiente **vínculo** se le incluye una animación que le permitirá ver los resultados del experimento. El video original que se utilizó para este ensayo fue tomado de **aquí**.

Frame del video original (matriz **A**) | Frame del video producido por la matriz **S** | Frame del video producido por la matriz **L**

Figura 5.1: Muestra del mismo frame, tanto del video original, como de los videos generados con las matrices `S` y `L`.

GoDec-Imágenes

En este caso se cuenta con un conjunto de `numImage` imágenes del mismo objeto (que puede ser un rostro) y todas de tamaño `m` $\times$ `n`. Las imágenes poseen sombras o reflejos que dificultan la visualización del objeto completo. El objetivo de la aplicación es utilizar el algoritmo `GoDec` para eliminar las sombras y los reflejos y así poder reconstruir una imagen limpia del objeto.

La codificación de las imágenes se realiza siguiendo el proceso descrito en la introducción de este manual y para así obtener la matriz `A`. Al aplicar el algoritmo `GoDec` nuevamente se obtienen matrices L y S, tales que $A \approx L + S$. Si la diferencia entre las imágenes originales corresponde a sombras o reflejos, entonces dichas distorsiones se pueden interpretar como pixeles de objetos en movimiento. De esta manera, de la matriz L se puede extraer la imagen reconstruida (cualquiera de las columnas de L representa una posible reconstrucción de la imagen). Por su parte, en la matriz `S` se almacenan las sombras y los reflejos. Procederemos con un ejemplo que permita mostrarle al lector el experimento realizado con la instrucción:

```
GoDecImageFull('sixFaces',1,50);
```

Donde los valores 1 y 50 utilizados en este ejemplo corresponden, respectivamente, a la restricción para el rango máximo de la matriz `L` y a la restricción máxima para la cardinalidad de `S`. Para este ejemplo, se supone que `sixFaces` es un directorio con 6 rostros con sombras. Los rostros utilizados se muestran en la Figura 5.2 y fueron tomados de **the extended Yale face database B** ([3]). Al ejecutar la instrucción anterior se crean dos nuevos directorios en la carpeta actual de trabajo, con los nombres `ImageForL` y `ImageForS`, donde se guardarán las imágenes filtradas. En la Figura 5.2 se puede observar el resultado obtenido con la aplicación de `GoDec` para el filtrado de sombras.

Figura 5.2: Filtrado de sombras mediante GoDec.

Teniendo claro las aplicaciones anteriores, se procederá con la descripción del problema de optimización intrínseco al algoritmo GoDec. Para ello se parte de una matriz densa $A \in \mathbb{R}^{m\times n}$ y se busca determinar matrices $L \in \mathbb{R}^{m\times n}$ y $S \in \mathbb{R}^{m\times n}$, con $\text{rank}\,(L) \leq r$ y $\text{card}\,(S) \leq k$, tales que:

$$\min_{\text{rank}(L)\leq r;\text{card}(S)\leq k} ||A - L - S||_{fr}^2 \tag{5.1}$$

La solución del problema 5.1 se hace de forma iterativa utilizando el algoritmo propuesto en [10] y denominado *Go Decomposition* (GoDec), el cual consiste en la actualización sucesiva de las matrices L_t y S_t, mediante las siguientes reglas:

$$L_t = \sum_{i=1}^{r} \lambda_i U_i V_i^T, \;\; \text{SVD}(A - S_{t-1}) = U\Lambda V^T$$

$$S_t = \mathcal{P}_\Omega(A - L_t), \;\; \Omega : |(A - L_t)_{i,j\in\Omega}| \neq 0, \quad |\Omega| \leq k$$

La función $\mathcal{P}_\Omega$ toma una matriz densa Y y retorna una matriz esparcida de cardinalidad k, la cual contiene las k entradas de mayor valor absoluto en Y, las cuales se almacenan en su misma posición y las demás entradas se toman como cero. El algoritmo clásico de `GoDec` se puede ver en el Algoritmo 2.

Algoritmo 2: GoDec clásico.

Input: k, r, q, $A_{m\times n}$
Result: L, S

$L_0 := A$, $S_0 := \mathbf{0}_{m\times n}$, $t := 0$;
while *Verdadero* **do**
  $t := t + 1$;
  $SVD = \text{SVD}(A - S_t)$;
  $L_t := S_r V_r D_r$;
  $S_t := \mathcal{P}_\Omega\,(A - L_t)$;
  $E_t := \dfrac{\|A - L_t - S_t\|_{fr}^2}{\|A\|_{fr}^2}$;
  if $|E_t - E_{t-1}| < tol$ **then**
    break
  end
end
Retornar L, S

En [10] se presenta una mejora (en términos de cantidad de cálculos y tiempo de ejecución) del método `GoDec` al sustituir la descomposición en valores singulares (SVD) por el método `BRP` (*bilateral random projection*). De esta forma, la actualización de L se puede realizar por medio de la fórmula 5.2, donde $Y_1 = L_t A_1$ y $Y_2 = L_t^T A_2$, con $A_1 \in \mathbb{R}^{n\times r}$ y $A_2 \in \mathbb{R}^{m\times r}$ matrices aleatorias. El Algoritmo 3 presenta el método `GoDec` modificado.

$$L_{t+1} = Y_1 \left(A_2^T Y_1\right)^{-1} Y_2^T \tag{5.2}$$

Algoritmo 3: GoDec modificado con el método `BRP`.

Input: k, r, q, $A_{m\times n}$
Result: L, S

$L_0 := A$, $S_0 := \mathbf{0}_{m\times n}$, $t := 0$;
while *Verdadero* **do**
  $t := t + 1$;
  $L = A - S$;
  $Y_2 = randn(n, r)$;
  for *i=1:q+1* **do**
    $Y_1 = L \cdot Y_2$;
    $Y_2 = L^T \cdot Y_1$;
  end
  $QR = qr(Y_2)$;
  $L_t := (L \cdot Q) \cdot Q^T$;
  $S_t := \mathcal{P}_\Omega (A - L_t)$;
  $E_t := \frac{\|A - L_t - S_t\|_{fr}^2}{\|A\|_{fr}^2}$;
  if $|E_t - E_{t-1}| < tol$ **then**
    break
  end
end
Retornar L, S

Ahora se procederá con la descripción de las funciones del toolbox.

5.1 GoDec

Esta función calcula las matrices L y S correspondientes a la solución del problema 5.1. Recibe los siguientes argumentos:

- `X`: Matriz de tamaño `m` × `n`, para la que se resolverá el problema 5.1.
- `r`: Es la cota máxima del rango de L. Este valor debe ser entero positivo y menor o igual que el rango de `X`.
- `k`: Es la cardinalidad máxima que tendrá S. Este valor debe ser entero, positivo y menor que `mn`.
- `c`: Parámetro del esquema de potencias utilizado por el método, se utiliza para mejorar la generación de las matrices de proyección bilateral (ver [10]). Debe ser entero, positivo y podría ser ajustado por el usuario. Su valor por defecto es 3.

- `Tol`: Tolerancia para la estabilización del error, tal que $10^{-15} <$`Tol`≤ 100. Su valor por defecto es 10^{-8}.
- `IteraMax`: Número máximo de iteraciones realizadas por el algoritmo, en caso que no se alcance la tolerancia `Tol`. Este valor debe ser entero tal que $1 <$`IteraMax`< 10000 y su valor por defecto corresponde a 100.
- `Opcion`: Etiqueta con la que se indica si se desea realizar `GoDec` con el método BRP o la SVD truncada. Las etiquetas válidas con `'BRP'` y `'SVD'`.

Esta función retorna:

- `L`: Matriz de tamaño $m \times n$ y de rango menor o igual que `r`.
- `S`: Matriz de tamaño $m \times n$ y de cardinalidad menor o igual a `k`.
- `Errors`: Vector en el que se almacena, para cada iteración, el valor del error calculado con la fórmula $\|X - L - S\|_{fr}^2$.

5.1.1 Sintaxis

```
[L,S,Errors] = GoDec(X,r,k,'Opcion','c',c,'IteraMax',IteraMax,'Tol',Tol)
```

Los parámetros `c`, `IteraMax` y `Tol` son opcionales.

Ejemplo

Considere la matriz:

```
A=[−3  4  5  −4; 2  3  −5  1; 6  −5  −2  5; 3  −6  1  2];
```

Al ejecutar la función `GoDec` con

```
rng(1); %Se establece la semilla en 1 para poder replicar el experimento.
X=[−3 4 5 −4;2 3 −5 1;6 −5 −2 5;3 −6 1 2];
[L,S,Errors]=GoDec(X,2,10,'BRP','IteraMax',50,'Tol',0.000001);
```

Genera:

```
L =

−4.4074    3.1177    4.1548   −4.0026
 1.2590    2.9964   −5.0038    1.3115
 5.1303   −5.7649   −2.7389    4.5667
 2.9935   −6.0047    0.9952    2.5504

S =

 1.4074    0.8823    0.8452         0
 0.7410         0         0   −0.3115
 0.8697    0.7649    0.7389    0.4333
      0         0         0   −0.5504
```

5.2 GoDecVideoFull

Esta función aplica el proceso de GoDec a un video y genera, en el directorio de trabajo, los videos L.mp4 y S.mp4, construidos con las matrices L y S, respectivamente. Esta función recibe los siguientes argumentos:

- Textpath: Es la dirección del archivo de video con extensión .mp4, el cual será codificado internamente, en escala de grises, como una matriz X.
- r: Este valor es la cota máxima para el rango de la matriz L que se determina al resolver el problema 5.1. Debe ser entero positivo y menor que el rango de X.
- k: Es la cardinalidad máxima que tendrá la matriz S. Debe ser entero, positivo y menor que mn · numFrames (número total de pixeles en el video completo).
- Scale: Este valor se utiliza para escalar cada uno de los frames en una fracción del tamaño original. Este escalamiento es útil para videos con alta resolución. Debe cumplir que $0 <$Scale≤ 1. Su valor por defecto es 1, en cuyo caso se estaría utilizando el tamaño de los frames del video original.
- numFrames: Es el número de frames que se utilizarán en el video. Este valor es un entero positivo que debe ser menor o igual que el número total de frames que tiene el video original. Si se omite, se utilizarán todos los frames.
- c: Parámetro del esquema de potencias utilizado por el método, se utiliza para mejorar la generación de las matrices de proyección bilateral (ver [10]). Debe ser entero, positivo y podría ser ajustado por el usuario. Su valor por defecto es 3.
- Tol: Tolerancia para la estabilización del error, tal que $10^{-15} <$Tol≤ 100. Su valor por defecto es 10^{-8}.
- IteraMax: Número máximo de iteraciones realizadas por el algoritmo, en caso que no se alcance la tolerancia Tol. Este valor debe ser entero tal que $1 <$IteraMax$<$ 10000 y su valor por defecto corresponde a 100.
- Opcion: Etiqueta con la que se indica si se desea realizar GoDec con el método BRP o la SVD truncada. Las etiquetas válidas con 'BRP' y 'SVD'.

5.2.1 Sintaxis

```
GoDecVideoFull('Textpath',r,k,'Opcion','Scale',Scale,'numFrames',numFrames,'c',c,'Tol',Tol,'IteraMax
    ',IteraMax)
```

Los parámetros Scale, numFrames, c, Tol e IteraMax son opcionales.

5.3 GoDecImageFull

Esta función aplica el proceso de GoDec a un conjunto de imágenes en una base de datos. La función recibe:

- Textpath: Es la dirección del directorio de las imágenes. Para la lectura del directorio se utiliza la función ReadImageDataBase (ver la sección 1.1). El número de

imágenes se denota con `numImage`. Las imágenes se codifican en una matriz X de tamaño `mn` $\times$ `numImage`, que es a la que se le aplica el algoritmo `GoDec`.

- `r`: Este valor corresponde a la cota máxima del rango de la matriz L. Debe ser entero positivo y menor que el mínimo entre el número de imágenes en la base de datos y el número de pixeles en cada imagen.
- `k`: Este valor corresponde a la cardinalidad máxima que tendrá la matriz S. Debe ser entero, positivo y menor que `mn` $\cdot$ `numImage`.
- `c`: Parámetro del esquema de potencias utilizado por el método, se utiliza para mejorar la generación de las matrices de proyección bilateral (ver [10]). Debe ser entero, positivo y podría ser ajustado por el usuario. Su valor por defecto es 3.
- `Tol`: Tolerancia para la estabilización del error, tal que $10^{-15} <$`Tol`≤ 100. Su valor por defecto es 10^{-8}.
- `IteraMax`: Número máximo de iteraciones realizadas por el algoritmo, en caso que no se alcance la tolerancia `Tol`. Este valor debe ser entero tal que $1 <$`IteraMax`$<$ 10000 y su valor por defecto corresponde a 100.
- `Opcion`: Etiqueta con la que se indica si se desea realizar `GoDec` con el método BRP o la SVD truncada. Las etiquetas válidas con `'BRP'` y `'SVD'`.

Al ejecutar esta función se crean dos carpetas en el directorio actual de trabajo, denominadas `ImagesForS` y `ImagesForL`. En la primera de ellas se guardarán las imágenes generadas con la matriz `S`, mientras que en la segunda, las imágenes generadas con la matriz `L`.

5.3.1 Sintaxis

```
GoDecImageFull('Textpath',r,k,'Opcion','c',c,'Tol',Tol,'IteraMax',IteraMax);
```

Los parámetros `c`, `Tol` e `IteraMax` son opcionales.

Capítulo 6

ALGORITMO K-SVD APLICADO A LA RECONSTRUCCIÓN DE IMÁGENES CON PIXELES PERDIDOS

En este capítulo se mostrará una aplicación de un método, denominado K-SVD, que permite la recuperación de pixeles perdidos en una imagen, la cual se basa en el artículo [11]. Matemáticamente, el método se explica de la siguiente manera: dada la matriz $Y \in \mathbb{R}^{m\times n}$, se busca aproximar matrices $D \in \mathbb{R}^{m\times t}$ y $X \in \mathbb{R}^{t\times n}$, $1 \leq t \leq \min\{m, n\}$, que sean solución de

$$\min_{D,X} \|Y - DX\|_{fr}^2 \tag{6.1}$$

sujeto a las condiciones $\text{Card}(x_i) \leq c_0$, para $i = 1, \ldots, n$, donde $x_i \in \mathbb{R}^n$ representa la i-ésima columna de X y $\text{Card}(x_i)$ representa la cantidad de entradas de x_i diferentes de cero. En este caso, la constante c_0 es un número entero positivo que se conoce como constante de esparcidad y corresponde a un parámetro de entrada para el algoritmo.

El problema (6.1) busca realizar una representación esparcida de los datos de entrada, determinando un cierto número de patrones elementales, denominados *átomos*, que combinados entre sí permitan realizar una reconstrucción. El algoritmo que da solución al problema (6.1) se llama K-SVD y corresponde a un método iterativo que alterna entre la escasa codificación de los datos almacenados en la matriz actual D, denominada *diccionario*, y un proceso de actualización de los átomos del diccionario para que haya un mejor ajuste. En el Algoritmo 4 se presenta el pseudocódigo.

Algoritmo 4: Algoritmo K-SVD.

Input: $Y \in \mathbb{R}^{m\times n}$, $D^{(0)} \in \mathbb{R}^{m\times t}$, $c_0 \in \{1, 2, ..., n\}$, $iter \in \mathbb{N}^*$
Result: $D^{(iter)} \in \mathbb{R}^{m\times t}$ y $X^{(iter)} \in \mathbb{R}^{t\times n}$

1 $D = D^{(0)}$;
2 **for** *k=1:iter* **do**
3 $\quad \forall i = 1, ..., n$ determine $x_i = \underset{x}{\text{argmín}} \|y_i - Dx\|_2^2$, tal que Card$(x) \leq c_0$;
4 $\quad$ Defina $X = (x_1 \ \dots \ x_n)$;
5 $\quad$ Calcule $R = Y - DX$;
6 $\quad$ **for** *j=1:t* **do**
7 $\quad\quad$ Calcule I, el vector de índices asociados a las entradas de x_j no nulas;
8 $\quad\quad$ Calcule $\widetilde{R} = R_I + d_j x_j(I)$;
9 $\quad\quad$ Determine $\widetilde{R} = U\Sigma V^T$ (SVD de $\widetilde{R}$);
10 $\quad\quad$ Sustituya $d_j = u_1$, donde u_1 es la primera columna de U;
11 $\quad\quad$ Sustituya $x_j(I) = \sigma_1^2 v_1^T$, donde σ_1 es el primer valor singular de $\widetilde{R}$ y v_1 es la primera columna de V;
12 $\quad$ **end**
13 $\quad$ Defina $D^{(k)} = (d_1 \ \dots \ d_t)$;
14 **end**

El Algoritmo 4, en el paso 3, plantea aproximar cada columna de X, sujeto a una condición de esparcidad. En este paso se utilizó el algoritmo *orthogonal matching pursuit* (OMP) (tomado de [12]). Por otra parte, en el paso 8 del Algoritmo 4 tome en cuenta que:

- R_I es la matriz definida a partir de R tomando las columnas asociadas a los índices en I. Por ende, el tamaño de R_I es $m \times$ Card(I).
- d_j es la $j-$ésima columna de D (de dimensión $m \times 1$).
- $x_j(I)$ es el vector construido con las entradas no nulas de x_j (usando para ello los índices almacenados en I). Por lo tanto, los vectores I y $x_j(I)$ tienen tamaño $1 \times$ Card(I).

Ejemplo de la aplicación en procesamiento de imágenes

Como se mencionó anteriormente, una aplicación del algoritmo K-SVD consiste en la recuperación de pixeles perdidos en una imagen. Primero, debemos considerar una base de datos de s imágenes $\mathcal{A} = \{A_1, A_2, ..., A_s\}$, donde A_j es una imagen de tamaño $m \times n$, para todo $j = 1, ..., s$. Luego, se debe obtener la matriz Y a partir de bloques generados con cada imagen A_j, siguiendo los siguientes pasos:

- Cada imagen A_j se separa en bloques de tamaño $p \times q$, donde p y q son divisores de m y n, respectivamente.
- Se obtienen un total de $\frac{mn}{pq}$ sub-bloques de tamaño $p\times q$ de cada imagen A_j y $r = \frac{smn}{pq}$ sub-bloques para toda la base de imágenes $\mathcal{A}$.
- Cada sub-bloque se transforma en un vector y_j de tamaño $pq \times 1$.
- Finalmente, se define una matriz $Y = [y_1 \ y_2 \ ... \ y_r]$ de tamaño $pq \times r$.

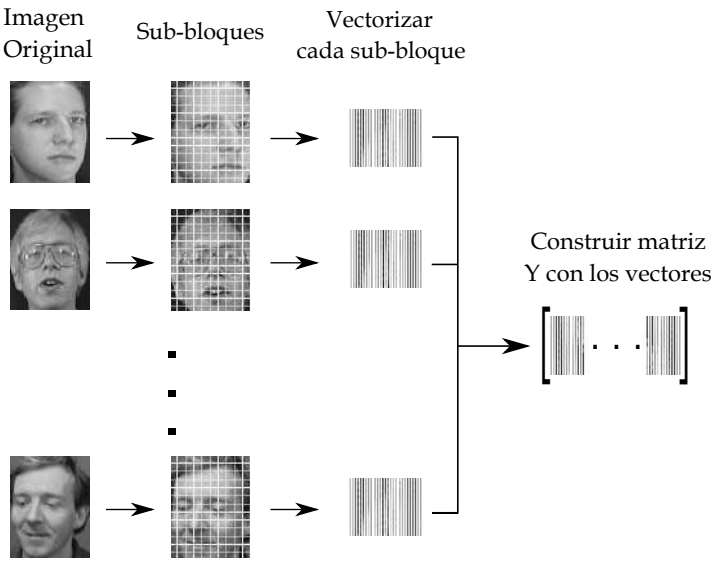


Figura 6.1: Representación matricial de las imágenes.

Para el ejemplo se utilizó la base de imágenes $\mathcal{A}$ que puede descargar del siguiente vínculo. En la Figura 6.1 se visualiza una representación de los pasos a seguir para la construcción de la matriz Y a partir de $\mathcal{A}$. Posteriormente, se utiliza la matriz Y y se aplica el algoritmo K-SVD para obtener las matrices $D^{(k)}$ y $X^{(k)}$ que dan solución al problema (6.1). La matriz $D^{(k)}$ corresponde al diccionario generado con el Algoritmo 4, que permite reconstruir una imagen con pixeles perdidos. La calidad de la reconstrucción depende de qué tan similar es la imagen por reconstruir, con respecto a las imágenes en $\mathcal{A}$. Para el ejemplo, se utilizó la siguiente imagen con pixeles perdidos.

Para reconstruir la imagen con pixeles perdidos, que también debe ser de tamaño $m \times n$, se divide la imagen en bloques de tamaño $p \times q$ y se realiza la reconstrucción por bloques. Cada bloque se transforma en un vector z_j de tamaño $pq \times 1$ y para cada bloque se realizan los siguientes pasos:

- El vector z_j puede tener algunas entradas iguales a 0, las cuales representan los pixeles perdidos. Utilizando esta información se define la *matriz diezmada* asociada a z_j, denotada D_{z_j}, de la siguiente manera:

$$D_{z_j}(i,:) = \begin{cases} D^{(k)}(i,:) & \text{si } z_j(i) \neq 0 \\ \mathbf{0} & \text{si } z_j(i) = 0 \end{cases},$$

  donde $D_{z_j}(i,:)$ y $D^{(k)}(i,:)$ representan la fila i de las matrices D_{z_j} y $D^{(k)}$, respectivamente, y $\mathbf{0}$ es el vector fila nulo. Esto es, la fila i de D_{z_j} se toma igual a la fila i de $D^{(k)}$, en caso que la $i-$ésima entrada del vector z_j no sea cero. Caso contrario, a la fila i de D_{z_j} se le asigna todas sus entradas iguales a cero.

- Utilizando un algoritmo de *compressed sensing* se determina x_{z_j}, tal que

$$x_{z_j} = \underset{x}{\text{argmín}} \|z_j - D_{z_j} x\|_2^2,$$

sujeto a Card$(x_{z_j}) \leq c_0$. En este caso nuevamente se utilizó el algoritmo OMP.

- Finalmente, para reconstruir el $j-$ésimo bloque se realiza la multiplicación $D^{(k)} \cdot x_{z_j}$. El resultado de este producto se convierte en un bloque de tamaño $p \times q$, y corresponde a la reconstrucción del $j-$ésimo bloque de la fotografía.

Para efectos del ejemplo, la Figura 6.2 muestra los resultados obtenidos en la reconstrucción de la fotografía seleccionada.

75% de pixeles perdidos

Reconstrucción con el algoritmo K-SVD

Figura 6.2: Reconstrucción de una imagen con 75% de pixeles perdidos mediante el algoritmo K-SVD.

Ahora se procederá con la descripción de las funciones del toolbox.

6.1 `ksvd`

Esta función calcula la matriz diccionario D y la matriz esparcida X del método K-SVD. Recibe los siguientes argumentos:

- `Textpath:` Ruta de la carpeta con las imágenes de entrenamiento.
- `r:` Número de columnas del diccionario (número de átomos).
- `c:` Constante de esparcidad.
- `Extension:` Las extensiones que se aceptan son jpg, pgm, png, tif, bpm. El valor por defecto es .jpg
- `IteraMax:` Número máximo de iteraciones a realizar. El valor por defecto es 200.
- `Tol:` Tolerancia, para el método de paro, diferencia entre dos errores consecutivos menor que Tol. EL valor por defecto es 10^{-3}.

Esta función retorna:

- `D:` Matriz diccionario de tamaño $m \times r$.
- `X:` Matriz esparcida de tamaño $r \times n$.
- `Err:` Error relativo normalizado del método.

6.1.1 Sintaxis

```
[D,X,Err]=ksvd('Textpath',r,c,'Extension','.ext','IteraMax',IteraMax,'Tol',Tol);
```

Los parámetros `Extension`, `IteraMax` y `Tol` son opcionales.

6.2 clean_ksvd

Esta función recibe la ruta de una imagen que presenta pixeles perdidos y la reconstruye utilizando la matriz de diccionario D generada por el método K-SVD. Esta función debe utilizarse posterior a la función `ksvd`, que es la que genera la matriz D.

Recibe los siguientes argumentos:

- `Textpath`: Ruta de la imagen que se desea reconstruir.
- `D`: Matriz diccionario generada por el método K-SVD.

Esta función retorna:

- `Y`: Matriz que representa la imagen reconstruida.

6.2.1 Sintaxis

```
Y=clean_ksvd('Textpath',D);
```

REFERENCIAS

[1] J. Chung and M. Chung, "Computing optimal low-rank matrix approximations for image processing," in *2013 Asilomar Conference on Signals, Systems and Computers*, Nov 2013, pp. 670–674.

[2] S. Friedland and A. Torokhti, "Generalized rank-constrained matrix approximations," *SIAM Journal on Matrix Analysis and Applications*, vol. 29, no. 2, pp. 656–659, 2007.

[3] A. Georghiades, P. Belhumeur, and D. Kriegman, "From few to many: Illumination cone models for face recognition under variable lighting and pose," *IEEE Trans. Pattern Anal. Mach. Intelligence*, vol. 23, no. 6, pp. 643–660, 2001.

[4] D. Lee and H. Seung, "Algorithms for non-negative matrix factorization," *Adv. Neural Inform. Process. Syst.*, vol. 13, 02 2001.

[5] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936.

[6] Z. Allen-Zhu and Y. Li, "LazySVD: Even faster SVD decomposition yet without agonizing pain," in *Advances in Neural Information Processing Systems*, 2016, pp. 974–982.

[7] M. Fazel, E. Candes, B. Recht, and P. Parrilo, "Compressed sensing and robust recovery of low rank matrices," in *2008 42nd Asilomar Conference on Signals, Systems and Computers*. IEEE, 2008, pp. 1043–1047.

[8] T. Zhou and D. Tao, "Bilateral random projections," in *2012 IEEE International Symposium on Information Theory Proceedings*. IEEE, 2012, pp. 1286–1290.

[9] J. Chavarría-Molina, J. J. Fallas-Monge, and P. Soto-Quiros, "Effective implementation to reduce execution time of a low-rank matrix approximation problem," *Journal of Computational and Applied Mathematics*, vol. 401, p. 113763, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S037704272100385X

[10] T. Zhou and D. Tao, "Godec: Randomized low-rank & sparse matrix decomposition in noisy case," in *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, 2011.

[11] M. Aharon, M. Elad, and A. Bruckstein, "K-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on signal processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[12] Y. Eldar and G. Kutyniok, *Compressed Sensing: Theory and Applications*, ser. Compressed Sensing: Theory and Applications. Cambridge University Press, 2012.