Facultat de Matemàtiques i Estadística de la Universitat Politècnica de Catalunya

Descomposició en valors singulars (SVD)

Propietats, algorisme i exemples d'aplicacions

Ramon Celma Cercadillo [1] i David Olivé Farga [2]

Estudiants del Grau en Matemàtiques - Assignatura: Càlcul Numèric

Barcelona, a 23 de desembre de 2012

1 Introducció: descomposició en valors singulars i propietats

Aquest treball pràctic tracta sobre la **descomposició en valors singulars (SVD)**, un tipus de factorització amb aplicacions molt diverses. Al nostre treball hem vist com resoldre els problemes de mínims quadrats i com comprimir imatges emprant aquesta descomposició. No obstant, hem trobat que hi ha molts altres exemples en els que la SVD també es útil: en el tractament de senyals [1], en el càlcul de pseudoinverses de matrius [2], en el problema estadístic de l'anàlisi de components principals [3] o en la predicció numèrica del fenòmens metereològics [4] [5].

En aquesta primera part del nostre informe volem mostrar als professors la petita investigació que hem fet per tal de conèixer més a fons la descomposició SVD. Al paràgraf anterior hem citat algunes aplicacions pràctiques que hem trobat, però ara intentarem interpretar el teorema geomètricament i després demostrarem unes propietats esmentades a l'enunciat de la pràctica de gran utilitat. Recordem ara el teorema esmentat a l'enunciat de la pràctica, que ens defineix la descomposició en valors singulars.

Teorema: *(Descomposició en valors singulars (SVD)) Sigui $A \in M_{m,n}(\mathbb{R})$, amb $m \geq n$. Aleshores, existeix una descomposició de la matriu A de la forma*

$$A = U \cdot \Sigma \cdot V^T \tag{1}$$

satisfent les següents propietats:

- *$U \in M_{m,n}(\mathbb{R})$ és una matriu ortogonal, $U^T \cdot U = I_n$. Els vectors-columna que formen U, és a dir $U = (u_1, u_2, ..., u_n)$, reben el nom de vectors singulars per la esquerra.*
- *$V \in M_{n,n}(\mathbb{R})$ és també ortogonal, $V^T \cdot V = I_n$. Anàlogament al punt anterior, els vectors-columna que formen $V = (v_1, v_2, ..., v_n)$, s'anomenen vector singulars per la dreta.*
- *$\Sigma \in M_{n,n}(\mathbb{R}) = diag(\sigma_1, \sigma_2, ..., \sigma_n)$ amb $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0$, anomenats valors singulars de la matriu A. El valor singular σ_1 es diu valor singular principal.*

Si la nostra matriu $A \in M_{m,n}(\mathbb{R})$ fos tal que $n > m$, considerant A^T ja podem aplicar el nostre teorema, obtenint la descomposició $A^T = V \cdot \Sigma \cdot U^T$.

Demostració: Inclosa a l'enunciat de la pràctica.

La idea geomètrica del SVD és que sota unes circunstàncies adequades tota matriu actua com una diagonal, agafant els sistemes de coordenades ortogonals adients a l'espai de sortida i arribada i tenint en compte dominis i rangs. Fixem-nos que, a partir d'una matriu $A \in M_{m,n}(\mathbb{R})$ que envia vectors $x \in \mathbb{R}^n$ a vectors $y = Ax \in \mathbb{R}^m$, podem trobar un sistema de coordenades ortogonal per a $\mathbb{R}^n$ (les columnes de V) i una altra referència del subespai vectorial $Ax \in \mathbb{R}^m$ (les columnes de la matriu U, amb $m \geq n$), tals

[1] Per correcció d'errors al document o aclaracions, contactar a: ramon.celma@estudiant.upc.edu
[2] Per aclaracions, contactar a: david.olive.farga@estudiant.upc.edu

que en aquestes referències A passa a ser Σ diagonal. Efectivament: per $x \in \mathbb{R}^n$, $x = \sum_{i=1}^n \beta_i v_i$, que per l'aplicació lineal A s'envia a $Ax = y = \sum_{i=1}^n \sigma_i \beta_i u_i$. Per tant, A actua com una matriu diagonal.
Com a conseqüència del teorema, tenim que la descomposició SVD satisfà moltes propietats de gran interès. Algunes d'elles van ser esmentades a l'enunciat de la pràctica però **sense la demostració**, que es va deixar com a exercici. Creiem interessant exposar la demostració d'aquestes propietats en el nostre informe d'entrega ja que aquestes ajuden a entendre més a fons el SVD. A més, **utilitzarem algunes d'elles al diseny del nostre algorisme i al problema de mínims quadrats**:

Teorema: *(Propietats de la descomposició SVD [6])* Sigui $A = U \cdot \Sigma \cdot V^T$ la descomposició SVD de la matriu $A \in M_{m,n}(\mathbb{R})$. Aleshores:

1. Suposeu que A és simètrica i que, per tant, admet una descomposició de la forma $A = U \cdot D \cdot U^T$, amb $D = diag(\lambda_1, \lambda_2, ..., \lambda_n)$, $U = (u_1, u_2, ..., u_n)$ ortogonal, és a dir, $U \cdot U^T = I_n$. Aleshores, una SVD d'A és $A = U \cdot \Sigma \cdot V^T$ amb $\sigma_i = |\lambda_i|$ i $v_i = sign(\lambda_i) \cdot u_i$, on $sign$ es la funció "signe" i $sign(0) = 1$.

2. Els vaps de la matriu simètrica $A^T A \in M_{n,n}(\mathbb{R})$ són σ_i^2. Els vectors singulars per la dreta $v_1, v_2, ..., v_n$ formen una base ortonormal de veps.

3. Els vaps de la matriu simètrica $AA^T \in M_{m,m}(\mathbb{R})$ són σ_i^2, $i = 1, 2, ..., n$, i $m - n$ zeros. Els vectors singulars per la esquerra u_i són els veps ortonormals corresponents als vaps σ_i^2. Un pot agafar uns altres $m - n$ vectors ortogonals com veps de vap 0.

4. Sigui $A \in M_{n,n}(\mathbb{R})$ una matriu quadrada i $A = U \cdot \Sigma \cdot V^T$ la seva SVD, amb $\Sigma = diag(\sigma_1, \sigma_2, ..., \sigma_n)$, $U = (u_1, u_2, ..., u_n)$ i $V = (v_1, v_2, ..., v_n)$. Considereu la matriu per blocs: $H = \begin{pmatrix} 0 & A^T \\ A & 0 \end{pmatrix}$ Aleshores, els vaps d'H són $\pm\sigma_1, \pm\sigma_2, ..., \pm\sigma_n$ i els seus vectors propis unitaris corresponents són $\frac{1}{\sqrt{2}} \cdot \begin{pmatrix} v_i \\ \pm u_i \end{pmatrix}$.

5. Si A té rang màxim, és a dir, si $rang(A) = n$, llavors

$$\min_{x \in \mathbb{R}^n} ||Ax - b||_2 = ||Ax^* - b||_2 \tag{2}$$

   amb $x^* = V\Sigma^{-1}U^T b$. Per tant, la SVD proporciona la solució del problema de mínims quadrats.

6. La norma matricial $|| \cdot ||_2$ d'A és el valor singular principal, $||A||_2 = \sigma_1$. Si $A \in M_{n,n}(\mathbb{R})$ és no singular aleshores $||A^{-1}||_2 = \frac{1}{\sigma_n}$ i el nombre de condició $\mu_2(A) = ||A||_2||A^{-1}||_2 = \frac{\sigma_1}{\sigma_n}$.

7. Suposem $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > \sigma_{r+1} = \sigma_{r+2} = ... = \sigma_n = 0$. Llavors $rang(A) = r$, i a més $KerA =< v_{r+1}, v_{r+2}, ..., v_n >$ i $ImA =< u_1, u_2, ..., u_r >$.

8. Sigui $S_n = x \in \mathbb{R}^n : ||x||_2 = 1$ l'esfera unitat n-dimensional i $A(S_n) = Ax : x \in S_n$ la seva imatge per la matriu $A \in M_{m,n}(\mathbb{R})$. Aleshores, $A(S_n)$ és un elipsoide de centre l'origen de $\mathbb{R}^m$ i eixos principals $\sigma_i u_i$.

Demostració: Algunes estàn incloses al [6], però hem intentat adaptar-les i detallar-les molt més

1. Tenim A simètrica amb la descomposició $A = U \cdot D \cdot U^T$ assegurada pel teorema espectral. Si volem transformar aquesta descomposició en la del SVD la matriu diagonal ha de ser sempre de termes positius. Per tant, redefinim D com $\tilde{D} = diag(|\lambda_1|, |\lambda_2|, ..., |\lambda_n|)$ i U^T com $\tilde{U^T} = (sign(\lambda_i) \cdot u_i)$. És trivial veure que $U \cdot D \cdot U^T = U \cdot \tilde{D} \cdot \tilde{V^T}$, i considerant $\tilde{D} = \Sigma$ i $\tilde{U^T} = V^T$ ja tenim la nostra descomposició SVD d'A.

2. Considerem la descomposició SVD d'A: $A = U\Sigma V^T$. Llavors $A^T A = V\Sigma U^T U\Sigma V^T = V\Sigma^2 V^T$, que és una descomposició d'$A^T A$ en valors propis σ_i^2 i vectors propis ortonormals v_i a les columnes de la matriu ortogonal V.

3. Agafem la matriu $U \in M_{m,n}(\mathbb{R})$ i constrium una matriu ortogonal $\tilde{U} \in M_{m,m}(\mathbb{R})$ afegint $m-n$ columnes de vectors després de u_n i conservant les columnes ja existents. Això sempre ho podem trobar: de fet, estem completant el conjunt de n vectors de $\mathbb{R}^m$ a una base de l'espai (recordem que $m \geq n$), cosa que sempre podem fer pel Teorema de Steinitz, i després ortonormalitzant aquesta base, cosa que també es pot fer mitjançant, potser, l'algorisme de Gram-Schmidt. Ara, com el SVD d'A és $A = U\Sigma V^T$, $AA^T = U\Sigma V^T V\Sigma U^T = U\Sigma^2 U^T$. Però $U\Sigma^2 U^T = \tilde{U} \begin{pmatrix} \Sigma^2 & 0 \\ 0 & 0 \end{pmatrix} \tilde{U^T}$, tenint així una descomposició en valors i vectors propis d'AA^T que satisfà les exigències de l'enunciat.

4. Considerem la matriu de l'enunciat $H = \begin{pmatrix} 0 & A^T \\ A & 0 \end{pmatrix}$. Com $A = U\Sigma V^T$, voldriem veure com podem expressar $H = \begin{pmatrix} 0 & V\Sigma U^T \\ U\Sigma V^T & 0 \end{pmatrix}$ per tal d'aprofitar que, sent H simètrica, tindrà una descomposició en vaps reals i veps formant base de $\mathbb{R}^{2n}$. Efectivament, si calculem el producte $\begin{pmatrix} V & V \\ U & -U \end{pmatrix} \begin{pmatrix} \Sigma & 0 \\ 0 & -\Sigma \end{pmatrix} \begin{pmatrix} V & V \\ U & -U \end{pmatrix}^T$, veiem que la matriu resultant és $2\begin{pmatrix} 0 & V\Sigma U^T \\ U\Sigma V^T & 0 \end{pmatrix}$, que és $2H$. Aquesta matriu, per la descomposició que acabem de trobar, té vaps $\pm\sigma_I$ i veps de la forma $\begin{pmatrix} v_i \\ \pm u_i \end{pmatrix}$. Per tant, d'aquí treiem els veps i vaps d'H: els vaps són $\pm\sigma_1, \pm\sigma_2, ..., \pm\sigma_n$ i els veps són, normalitzant, $\frac{1}{\sqrt{2}} \begin{pmatrix} v_i \\ \pm u_i \end{pmatrix}$, tal i com voliem.

5. Tenim que $\min_{x\in\mathbb{R}^n} ||Ax-b||_2^2 = ||U\Sigma V^T x - b||_2^2$. Com suposem que A té rang màxim, aleshores per definició del SVD la matriu Σ és invertible. Si considerem la matriu $\begin{pmatrix} U & \tilde{U} \end{pmatrix}$ tal que aquesta és quadrada i ortogonal (a la demostració de la proposició anterior es veu que aquesta matriu sempre existeix), llavors tenim que $||U\Sigma V^T x - b||_2^2 = ||\begin{pmatrix} U^T \\ \tilde{U^T} \end{pmatrix}(U\Sigma V^T x - b)||_2^2 = ||\begin{pmatrix} \Sigma V^T x - U^T b \\ -\tilde{U^T} b \end{pmatrix}||_2^2 = ||\Sigma V^T x - U^T b||_2^2 + ||\tilde{U^T} b||_2^2$. De l'últim terme d'aquesta cadena d'igualtats deduïm que si volem minimitzar hem d'agar una $x \in \mathbb{R}^n$ tal que faci zero el primer terme, ja que el segon terme $||\tilde{U^T} b||_2^2$ no depèn de cap variable. Però si volem $||\Sigma V^T x - U^T b||_2^2 = 0$ per definició de norma necessitem $\Sigma V^T x - U^T b = 0 \Leftrightarrow \Sigma V^T x = U^T b \Leftrightarrow V^T x = \Sigma^{-1} U^T b \Leftrightarrow x = V\Sigma^{-1} U^T b$, com volíem veure.

6. De la definició de la norma matricial $||\cdot||_2$ ($||A||_2 = \sqrt{\rho(A^T A)}$) es dedueix que la norma matricial $||\cdot||_2$ de matrius diagonals és l'element diagonal més gran en valor absolut. Per propietats bàsiques de la norma matricial $||\cdot||_2$, $||ABC||_2 = ||B||_2$ si A i B són matrius ortogonals (no deixa de ser altra cosa que el teorema de Pitàgores). Amb aquests dos comentaris previs, es fàcil deduïr que $||A||_2 = ||U^T AV||_2 = ||\Sigma||_2 = \sigma_1$, el valor singular principal. Anàlogament, $||A^{-1}||_2 = ||V^T A^{-1} U||_2 = ||\Sigma^{-1}||_2 = \frac{1}{\sigma_n}$. Com $\mu_2(A) = ||A||_2 ||A^{-1}||_2$, tenim que $\mu_2(A) = \frac{\sigma_1}{\sigma_n}$.

7. Escollim de nou una matriu $\tilde{U} \in M_{m,m-n}(\mathbb{R})$ tal que la matriu $\hat{U} = \begin{pmatrix} U & \tilde{U} \end{pmatrix} \in M_{m,m}(\mathbb{R})$ és no singular i ortogonal. Com, per hipòtesi, V tampoc ho es, les matrius A i $\hat{U^T} AV = \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} = \hat{\Sigma}$ tenen el mateix rang, sent A llavors de rang k, donat que hem suposat que el rang de Σ és r. Ara, el nucli de A està format per vectors $u \in \mathbb{R}^n$ tals que $Au = 0$. Però això passa si i només si $\hat{U^T} AV(V^T u) = 0$. Però el nucli de $\hat{\Sigma} = \hat{U^T} AV$ està generat clarament per les $r+1$ columnes fins n de la matriu I_n (recordem que la matriu Σ té $\sigma_{r+1} = \sigma r + 2 = ... = \sigma_n = 0$. Llavors, com busquem $u \in \mathbb{R}^n$ tals que $\hat{\Sigma}(V^T u) = 0$, el nucli de A està generat per les columnes $v_{r+1}, v_{r+2}, ..., v_n$. Anàlogament es pot veure que la imatge de A és generada per les r primeres columnes de $\hat{U}$, és a dir $Im(A) =< u_1, u_2, ..., u_r >$, donat que tenim la igualtat $\hat{U^T} AV = \hat{\Sigma}$.

8. Construirem, pas a pas, el conjunt $A(S_n)$ a partir de la descomposició SVD d'$A = U\Sigma V^T$. Com $x \in S_n \Leftrightarrow ||x||_2 = 1$ i $V^T \in M_{n,n}(\mathbb{R})$ és ortogonal, llavors $V^T(S_n) = S_n$. Seguint el mateix raonament, $w \in \Sigma S_n \Leftrightarrow ||\Sigma^{-1} w||_2 = 1$. Això defineix una elipsoide amb eixos principals $o_i e_i$, on e_i és l'i-èsim vector canònic de $\mathbb{R}^n$. Finalment, multiplicar ΣS_n per U només ens rota la elipse i

l'envia a un espai de dimensió superior ($m \geq n$), convertint cada e_i en u_i. Per tant, la nostra $A(S_n)$ resultant és un elipsoide amb centre l'origen de $\mathbb{R}^m$ i eixos principals $\sigma_i u_i$ $\square$

2 Disseny d'un algorisme per calcular la descomposició SVD

2.1 Una demostració constructiva no apta per a ordinadors

Una vegada estudiades les propietats del SVD a fons, la nostra primera idea per a dissenyar el nostre algorisme va ser aprofitar-nos de la demostració, la cual és plenament constructiva (no només prova l'existència de la factorització SVD, sino que et proporciona una manera de trobar-la). Davant d'aquesta idea, ens vam trobar amb un gran inconvenient que ens va fer desistir: el mètode constructiu de la demostració és bo per fer-lo a mà i amb matrius petites, però si volem que sigui un ordinador el que faci la feina per nosaltres amb matrius de gran tamany hem de pensar en els possibles errors computacionals:

- Tractar amb matrius plenes de gran dimensió pot ser perillós perque l'excés de càlculs ens pot fer arrossegar molts errors, que es van fent cada vegada més grans.
- La condició de la nova matriu $S = A^T A$ que utilitza la demostració del teorema pot arribar a ser el quadrat de la condició d'A! Jugar amb la possibilitat de aumentar el número de condició de la matriu de manera tan gran pot comportar-nos molts error numèrics que ens farien impossible trobar una descomposició en valors singulars acurada. A la referència [**?**] hi ha un estudi sobre la extremada sensibilitat que la matriu $A^T A$ pot arribar a tindre davant de perturbacions petites: els valor singulars més grans quasi no varien, però els petits poden fer-ho molt depenent del problema, i són aquests els que fan variar la condició, tal i com hem dit.
- Apart de que la condició de $S = A^T A$ pot ser dolenta, podem perdre algunes propietats com la no singularitat. Per exemple, si $A = \begin{pmatrix} 1 & 1 \\ 0 & \sqrt{\eta} \end{pmatrix}$ llavors $A^T A = \begin{pmatrix} 1 & 1 \\ 1 & 1+\eta \end{pmatrix}$. Recordem que si un número η és més petit que l'epsilon de la màquina, llavors $1 + \eta = 1$, encara que $\eta > 0$. Per tant, si això passés, tindriem que la nostra matriu $A^T A (= S)$ seria detectada com singular, quan en realitat no ho és, si $\eta \in (0, \epsilon_0)$, on ϵ_0 és el epsilon de la màquina. Deixem la referència d'aquest exemple [8]. Un altre cas de matrius L on calcular $L^T L$ seria letal són la familia de matrius de Läuchli $L = \begin{pmatrix} 1 & 1 & 1 \\ \epsilon & 0 & 0 \\ 0 & \epsilon & 0 \\ 0 & 0 & \epsilon \end{pmatrix}$, on ϵ pot ser un valor qualsevol. Aquí, $L^T L = \begin{pmatrix} 3 & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon^2 & 0 & 0 \\ \epsilon & 0 & \epsilon^2 & 0 \\ \epsilon & 0 & 0 & \epsilon^2 \end{pmatrix}$, un càlcul desastrós numèricament. Per més detalls [9].

2.2 Redüint la nostra matriu a una forma bidiagonal (superior)

Per tots els motius exposats a l'apartat anterior, és necesari agafar una via alternativa. Una bona idea seria trobar una matriu anàloga a la A que ens permeti treballar en millors condicions, és a dir: trobar una nova matriu més senzilla que la anterior (o més ben dit, més esparsa) a partir de la qual obtenir el SVD sigui més fàcil. Però el procés de transformació la nostra matriu A en una d'aquestes matrius més "bones" no ens ha de suposar problemes afegits (és a dir, no ha d'alterar el problema). Totes aquestes idees superficials que acabem d'exposar han sigut les que han quedat plasmades als dissenys de pràcticament tots els algortimes per a calcular la descomposició SVD, desde els més antics fins els més moderns: sempre es busca passar de la nostra matriu A a una triangular, tridiagonal o bidiagonal mitjançant transformacions ortogonals (és a dir, multiplicar a banda i banda de la nostra matriu A per matrius ortogonals). Amb això s'aconsegueix redüir significativament la xifra d'elements de la matriu que poden prendre valors diferents de zero i, com veurem més endavant, simplifica el problema de resoldre el SVD a un problema de valors i vectors propis que ja no implicarà necessàriament multiplicar una matriu per la seva transposada.

Pel nostre treball hem decidit **redüir la nostra matriu A a una bidiagonal (superior)**, que és el que fan la majoria d'algorismes que hem pogut trobar. De fet, el mètode que van publicar al *Journal*

of the Society for Industrial and Applied Mathematics a l'any 1965 Gene Golub i William Kahan [10] buscava que la matriu passés a ser bidiagonal. La importància d'aquesta publicació es majúscula: va ser el primer mètode pràctic per trobar la descomposició en valors singulars que buscava transformar abans de tot la matriu plena en una més senzilla (i esparsa), trencant amb els primers algorismes (lents i costosos) per obtenir la descomposició, sorgits a partir del 1954 i basats en el mètode de Jacobi combinat amb rotacions de Givens (podeu veure una reconstrucció al Demmel [11]). Aquesta publicació va suposar una revolució, donant lloc a la variació que al 1970 van introdüir Gene Golub i Christian Reinsch [12], la qual és la base de moltes variacions i optimitzacions de l'algorisme utilitzades avui en dia (més anotacions històriques a [13]).

Una pregunta natural és com trobar aquesta matriu bidiagonal (superior) B, i les matrius ortogonals M i N tals que $A = MBN^T$. El mètode de bidiagonalització proposat al Golub-Kahan [10] és la transformació de la nostra matriu mitjançant matrius de Householder. Caracteritzem primer aquestes matrius:

2.2.1 Les matrius de Householder:

(*Més informació a la referència [14]*)

Una matriu de Householder és una transformació lineal de l'espai que consisteix en fer una reflexió respecte d'un hiperplà determinat. La matriu es defineix com $H = I - \frac{2uu^T}{u^Tu}$, on u és el vector normal al plà (que passa pel zero). Si agafem u tal que $||u||_2 = 1$, llavors $H = I - 2uu^T$. A més, la matriu H té propietats molt bones que la fan idònea per trobar matrius semblants: és simètrica (donat que $H^T = I - 2uu^T = H$) i a més ortogonal (donat que $HH^T = (I - 2uu^T)(I - 2uu^T) = I - 2uu^T - 2uu^T + 4v(v^Tv)v^T = I$). Intuitivament: fer dos cops la reflexió d'un vector respecte a un plà ens dona el vector original, i.e. H és una simetria ortogonal. La idea gràfica és:

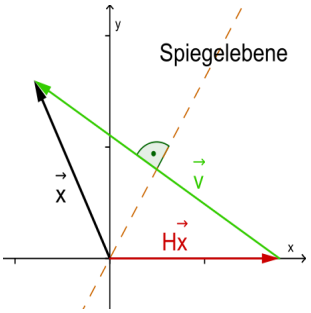

Figura 1: Una reflexió de Householder en un espai 2D [15]

Com nosaltres volem passar la nostra matriu A a bidiagonal superior, hem de "fer zeros" a moltes components dels vectors columna i fila. Donat un vector x (que veurem més endavant que serà un vector fila o columna de la nostra matriu), voldrem trobar una matriu de Householder H tal que $Hx = \begin{pmatrix} \lambda \\ 0 \\ \dots \\ 0 \end{pmatrix} = \lambda \cdot e_1$. Podem trobar la u unitària tal que $H = I - 2uu^T$ faci el que volem: Si $Hx = x - 2u(u^Tx) = \lambda \cdot e_1$, llavors aillem i ens queda $u = \frac{x - \lambda \cdot e_1}{2(u^Tx)}$, una combinació lineal de x i e_1. Per les propietats de la nostra matriu H, hem de tenir que $||x||_2 = ||Hx||_2 = |\lambda|$, pel que u ha de ser paral·lel al vector $\hat{u} = x \pm \lambda e_1$, i per tant podem agafar aquest vector $\hat{u}$ normalitzat ($u = \frac{\hat{u}}{||\hat{u}||_2}$) i ja tenim la reflexió H que voliem. Considerar aquesta H adequada per a les diferents columnes i files de la nostra matriu A ens dona l'algorisme de bidiagonatzació de Golub-Kahan (el nom prové de ser l'algortime que es va publicar al [10]).

2.2.2 Bidiagonalització de Golub-Kahan:

(*Hem utilitzat la referència [16], amb un llenguatge més actual que el de l'article de Golub-Kahan [10] del 1965*)

Sigui

$$A = \begin{pmatrix} X & X & X & X \\ X & X & X & X \\ X & X & X & X \\ X & X & X & X \\ X & X & X & X \end{pmatrix}$$

el diagrama de Wilkinson d'una matriu qualsevol A, és a dir, una forma "intuitiva" de representar **adimensionalment** una matriu A (insistim en que no vol dir per res que, per exemple, aquesta matriu sigui $5x4$!).

Volem ara agafar la primera columna i convertir-la en un vector del tipus $\begin{pmatrix} X \\ 0 \\ ... \\ 0 \end{pmatrix}$. Això s'obté utilitzant una matriu de Householder M_1 que ha quedat definida constructivament a l'apartat anterior. Si apliquem aquesta M_1 a tota la nostra matriu A, ens quedarà:

$$M_1 \cdot A = \begin{pmatrix} X & X & X & X \\ 0 & X & X & X \\ 0 & X & X & X \\ 0 & X & X & X \\ 0 & X & X & X \end{pmatrix}$$

Ara voldrem fer zeros a la primera fila de la nova matriu $M_1 \cdot A$. Com la matriu ha de ser bidiagonal superior necessitarem fer zeros a tots els números de la primera fila exceptuant els **dos** primers. Però tenim un problema, ja que el nostre mètode troba reflexions que anul·len totes les components d'un vector donat excepte la primera. Com no podem construir una reflexió de Householder a partir de tota la primera fila (perquè no volem fer zero el segon element del vector fila), una solució serà considerar el menor $(M_1A)_{(1:m,2:n)}$ i d'aquest ja podem agafar la primera fila i fer totes les components zero excepte la primera (que, a la matriu sencera, és el segon element de la primera fila). Si construim una matriu $\hat{N}_1$ que ens modifiqui la primera fila del menor $(M_1A)_{(1:m,2:n)}$, tindrem que no la podrem multiplicar per la dreta [3] per M_1A donat que les dimensions no quadraràn (només podria multiplicar per la dreta a $(M_1A)_{(1:m,2:n)}$). Una idea astuta és agafar aquesta matriu de Householder que transforma el menor contingut a $M_1A \in M_{m,n}(\mathbb{R})$ i mesclarla amb la matriu identitat $Id_{n,n}$ de forma que, multiplicant per blocs, el menor $(M_1A)_{(1:m,2:n)}$ quedi multiplicat per la nostra matriu de Householder i la part que no agafem en el nostre menor no quedi alterada (multiplicada per la identitat). D'aquesta contrucció de la que anomenarem matriu N_1, s'obtè:

$$M_1A \cdot N_1 = \begin{pmatrix} X & X & X & X \\ 0 & X & X & X \\ 0 & X & X & X \\ 0 & X & X & X \\ 0 & X & X & X \end{pmatrix} \begin{pmatrix} 1 & 0 \cdots 0 \\ \hline 0 & \\ \vdots & \hat{N}_1 \\ 0 & \end{pmatrix} = \begin{pmatrix} X & X & 0 & 0 \\ 0 & X & X & X \\ 0 & X & X & X \\ 0 & X & X & X \\ 0 & X & X & X \end{pmatrix}$$

Ara ja tenim la primera fila i la primera columna con desitjariem, i diem que hem completat un pas a la bidiagonalització de Golub-Kahan. El següent pas seria fer zeros a la segona columna, però ens trobariem amb el mateix problema que amb la primera fila: no podem construir la nostra matriu de Householder a partir de la segona columna sencera perquè no volem que se'ns anul·lin els dos primers números. El procediment seria anàleg al que vam explicar a la primera fila: agafariem la primera columna del menor $(M_1AN_1)_{(2:m),2:n}$, crearíem la transformació de Householder $\hat{M}_2$ que "fes zeros" a les seves components excepte la primera i construiriem M_2 a partir de completar la matriu $\hat{M}_2$ amb la identitat per quadrar les dimensions. Ens quedaria:

$$M_2M_1AN_1 = \begin{pmatrix} 1 & 0 \cdots 0 \\ \hline 0 & \\ \vdots & \hat{M}_2 \\ 0 & \end{pmatrix} \begin{pmatrix} X & X & 0 & 0 \\ 0 & X & X & X \\ 0 & X & X & X \\ 0 & X & X & X \\ 0 & X & X & X \end{pmatrix} = \begin{pmatrix} X & X & 0 & 0 \\ 0 & X & X & X \\ 0 & 0 & X & X \\ 0 & 0 & X & X \\ 0 & 0 & X & X \end{pmatrix}$$

[3] Quan es volen fer modificacions a les columnes d'una matriu es multiplica per la esquerra, i quan es volen fer a les files per la dreta, simplement per motius de com està construida la multiplicació de matrius

Si volguessim fer zeros a la segona fila de la matriu que estem bidiagonalitzant, tornaria a ser anàleg als passos anteriors, però aquí la matriu identitat ja hauria d'ocupar les dues primeres files i columnes: $N_2 = \left(\begin{array}{c|c} I_2 & 0_2 \\ \hline 0_2 & \hat{N}_2 \end{array}\right)$. Després de $n-2$ passos, ens trobarem que no podem eliminar cap element ni de la penúltima ni de la última fila, ja que formen part de la bidiagonal i diagonal, respectivament. Per tant, als passos $n-1$ i n només farem multiplicacions per la esquerra (fent zeros a les columnes), i quan arribem al pas n tindrem una matriu del tipus:

$$M_n \cdots M_2 M_1 A N_1 N_2 \cdots N_{n-2} = \begin{pmatrix} X & X & 0 & 0 \\ 0 & X & X & 0 \\ 0 & 0 & X & X \\ 0 & 0 & 0 & X \\ 0 & 0 & 0 & 0 \end{pmatrix} =^{4} \left(\begin{array}{cccc} a_1 & b_1 & 0 & \cdots \\ 0 & a_2 & b_2 & \cdots \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & b_n \\ \hline 0 & 0 & \cdots & a_n \\ 0 & 0 & 0 & 0 \end{array}\right)$$

que ja clarament ja es una bidiagonal de dimensió nxn amb $m-n$ files de zeros al final. Els zeros que es fan a un pas es conserven als altres per la construcció que hem fet del mètode, agafant els menors adients i completant astutament les matrius de Householder amb la aplicació identitat per tal que al fer la multiplicació de matrius per blocs no s'alteri cap transformació ja aplicada. Si diem que $M = M_1 M_2 \cdots M_N$ (d'on $M^T = M_n^T M_{n-1}^T \cdots M_2^T M_1^T = M_n M_{n-1} \cdots M_2 M_1$ per ser les matrius de Householder simètriques, com hem dit a l'apartat anterior) i $N = N_1 N_2 \cdots N_{n-2}$, tenim que $M \in M_{m,m}(\mathbb{R})$ i $N_{n,n}(\mathbb{R})$ són composició de matrius de Householder, i per tant són matrius ortogonals. **Definitivament, la nostra matriu bidiagonal** $B \in M_{m,n}(\mathbb{R})$ **serà igual a** $B = M_n \cdots M_2 M_1 A N_1 N_2 \cdots N_{n-2} = M^T A N$**, d'on** $MM^T ANN^T = MBN^T \Leftrightarrow A = MBN^T$**, que és la descomposició que voliem per a la nostra matriu** A. Un esquema de l'algorisme seria el següent, de collita pròpia (donat que els altres que hem trobat només es preocupen d'obtindre la matriu bidiagonal sense conservar la A ni dir quines són les matrius M i N de la descomposició):

Algoritmo 1 Bidiagonalització de Golub-Kahan

Entrada: La matriu $A \in M_{m,n}(\mathbb{R})$ del problema inicial
Salida: $M \in M_{m,m}(\mathbb{R})$ i $N \in M_{n,n}(\mathbb{R})$ ortogonals i $B \in M_{m,n}(\mathbb{R})$ bidiagonal tals que $A = MBN^T$

Creem les matrius $M \in M_{m,m}(\mathbb{R})$, $N \in M_{n,n}(\mathbb{R})$ i $B \in M_{m,n}(\mathbb{R})$, on anirem guardant la informació
Fem $M = I_m$, $N = I_n$ i $B = A$ per tal que funcioni aquest algorisme
para $i = 1$ fins a n **hacer**
  $x_k = B_{(k:m,k)}$
  $u_k = x_k + sign(x_k(1)) \cdot e_1$
  $u_k = \frac{u_k}{||u_k||_2}$
  $\hat{M}_k = I_{m-k+1} - 2u_k u_k^T$
  $M_k = \left(\begin{array}{c|c} I_{k-1} & 0_{k-1} \\ \hline 0_{k-1} & \hat{M}_k \end{array}\right)$
  $B = M_k \cdot B$
  $M = M \cdot M_k$
  si $k \geq n-2$ **entonces**
    $y_k = B_{(k,k+1:n)}$
    $v_k = y_k + sign(y_k(1)) \cdot e_1$
    $v_k = \frac{v_k}{||v_k||_2}$
    $\hat{N}_k = I_{n-k} - 2v_k v_k^T$
    $N_k = \left(\begin{array}{c|c} I_k & 0_k \\ \hline 0_k & \hat{N}_k \end{array}\right)$
    $B = B \cdot N_k$
    $N = N \cdot N_k$
  fin si
fin para
devolver Les matrius M, N y B

[4] Sense utilitzar diagrames de Wilkinson

Com a comentaris respecte a l'algorisme, remarcar primer de tot que no fa falta reservar un espai de memòria diferent per a cada M_k i N_k, dona que una vegada que en cada pas arribem a les línies 10 i 18, respectivament, ja no les necesitem per res, així que podem emprar el mateix espai de memoria per a totes les M_i i N_i, sobreescribint. Fer la bidiagonalització de la matriu $A \in M_{m,n}(\mathbb{R})$ és quasi igual que fer dos factoritzacions QR, pel que el número d'operacions és aproximadament $4mn^2 - \frac{4}{3}n^3$ segons [16] (a les quals hem d'afegir el cost de calcular M i N en cada pas, que en tot cas no altera el factor dominant d'ordre mn^2). Això pot donar problemes per a matrius on tenim una m molt més gran que la n, és per això interessant considerar una petita variació de l'algorisme de Golub-Kahan.

2.2.3 Algorisme de Lawson-Hanson-Chan

(*Hem utilitzat com a referència l'excepcional document de Peter Blomgren [17]*)

L'algorisme de Lawson-Hanson-Chan és una lleugera modificació de l'algorisme de Golub-Kahan basada en que, per a matrius on la m és molt més gran que la n, es fa molta més feina de la que en realitat es podria fer. La idea és passar de la molt rectangular matriu A a una completament quadrada de dimensions $n \cdot n$ per tal que l'algorisme de Golub-Kahan costi menys treball, i això s'aconsegueix fent una factorització QR previa de la nostra matriu A: $A = Q \cdot R$, on $Q \in M_{m,n}(\mathbb{R})$ i $R \in M_{n,n}(\mathbb{R})$ (nosaltres hem fet servir l'algorisme de Gram-Schmidt modificat). A partir d'aquí s'aplica l'algorisme de Golub-Kahan a la matriu R i queda $R = MBN^T$, amb $M, B, N \in M_{n,n}(\mathbb{R})$. El número d'operacions de l'algorisme de Lawson-Hanson-Chan és $2mn^2 + 2n^3$ (també, al igual que abans, sense comptar els costos associats a guardar les matrius N i M, inferior també ara que abans donat que són matrius més petites, ja que estem treballant amb R i no amb A). Amb els costos de l'algorisme de Golub-Kahan i de Lawson-Hanson-Chan, es pot saber a partir de quin moment surt més a compte utilitzar un algorisme o un altre: quan $m > \frac{5}{3}n$. Gràficament:

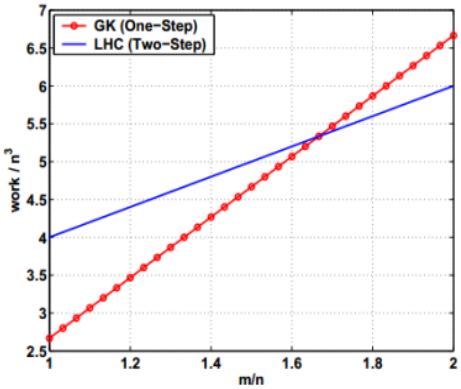

Figure: Comparing the work for Golub-Kahan and Lawson-Hanson-Chan bidiagonalization. The break-even point is $\frac{m}{n} = \frac{5}{3}$.

Figura 2: Gràfic extret de [17]

Al nostre programa, nosaltres hem comprovat al principi de tot si el quocient $\frac{m}{n}$ era més petit o igual que $\frac{5}{3}$: si era així utilitzavem l'algorisme de Golub-Kahan i en cas contrari el de Lawson-Hanson-Chan. Tot i això es pot dissenyar un algorisme híbrid per a fer una combinació més astuta dels dos mètodes de bidiagonalització esmentats: comença amb Golub-Kahan i quan es passa pel moment en que es té $\frac{m-k}{n-k} = 2$ canvia de Golub-Kahan a Lawson-Hanson-Chan mitjançant operacions entre menors de la matriu que estem bidiagonalitzant (per a més informació redirigim a [17]). El número d'operacions del mètode híbrid és $4mn^2 - \frac{4}{3}n^3 - \frac{2}{3}(m-n)^3$. Deixem un gràfic comparatiu de l'algorisme de Golub-Kahan, del de Lawson-Hanson-Chan i del híbrid:

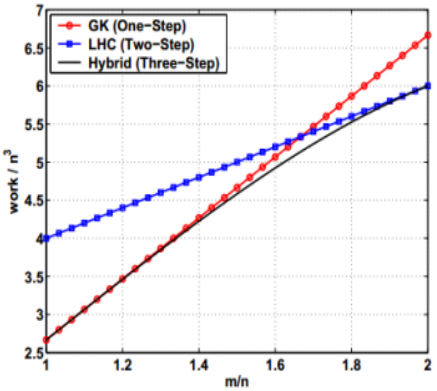

Figure: The work for the hybrid method is $\sim 4mn^2 - \frac{4}{3}n^3 - \frac{2}{3}(m-n)^3$, and provides a small improvement in the range $n < m < 2n$.

Figura 3: Gràfic extret de [17]

2.3 Transformant el problema de la descomposició SVD en un de valors i vectors propis

De la secció anterior tenim com trobar la semblança entre la matriu A i una bidiagonal. Si $\frac{m}{n} \leq \frac{5}{3}$, tindrem $A = MBN^T$, amb $M \in M_{m,n}(\mathbb{R})$, $B \in M_{m,n}(\mathbb{R})$ i $N \in M_{n,n}(\mathbb{R})$. Altrament, tindrem $A = Q \cdot (MBN^T)$, on $R = MBN^T$ (la R de la factorització QR) i $M, B, N \in M_{n,n}(\mathbb{R})$. En tot cas, sigui quin sigui l'opció escollida, el nostre problema no es ramifica, ja que dels dos casos podem extreure sempre una matriu bidiagonal quadrada $\hat{B}$: si $\frac{m}{n} \leq \frac{5}{3}$ agafem la $\hat{B}$ de la matriu bidiagonal $B = \begin{pmatrix} \hat{B}_{n,n} \\ 0_{m-n,n} \end{pmatrix}$, altrament agafem directament $\hat{B} = B$. El següent lema és de vital importància:

Lema: *Donada una matriu $A \in M_{m,n}(\mathbb{R})$ i la seva matriu bidiagonal quadrada $\hat{B} \in M_{n,n}(\mathbb{R})$ associada tenim que les dos tenen els mateixos valors singulars.*

Demostració: Sigui $A = MBN^T$ la factorització resultant d'un algorisme de bidiagonalització, amb M i N ortogonals. Llavors $A^TA = (MBN^T)^T(MBN^T) = NB^TM^TMBN^T = NB^TBN^T$. Per ser N ortogonal, de la igualtat anterior es dedueix que A^TA i B^TB són matrius semblants, i com a conseqüència d'això tenen els mateixos valors propis. Com els valors singulars d'A són l'arrel quadrada dels valors propis d'A^TA i els valors singulars de B són l'arrel quadrada dels valors propis de B^TB, per tenir A^TA i B^TB els mateixos valors propis A i B tenen els mateixos valors singulars □

Corol·lari: *Donada una matriu $A \in M_{m,n}(\mathbb{R})$ i la seva matriu bidiagonal quadrada $\hat{B} \in M_{n,n}(\mathbb{R})$ associada, tenim que és equivalent calcular la descomposició en valors singulars d'A o de B (o, millor dit, amb la descomposició d'una d'aquestes matrius es té l'altra).*

Demostració: Pels algorismes de bidiagonalització vists abans, tenim que $A = MBN^T$ i $B = M^TAN$. A més, pel teorema anterior tenim que si $A = U_1\Sigma_1V_1^T$ i $\hat{B} = U_2\Sigma_2V_2^T$ són les descomposicions en valors singulars d'A i de $\hat{B}$ (que sempre existeixen pel *Teorema de Descomposició en valors singulars (SVD)*, llavors com A i $\hat{B}$ tenen els mateixos valors singulars es té que $\Sigma_1 = \Sigma_2$. Amb tot això:

- Suposem que tenim $A = U_1\Sigma_1V_1^T$. Llavors, com $\Sigma_1 = \Sigma_2$ i $B = M^TAN$, tenim que $B = M^TU_1\Sigma_2V_1^TN$. Si diem que $U_2 = M^TU_1$ i $V_2^T = V_1^TN$ (aquestes dues matrius són ortogonals per ser producte d'ortogonals), tenim que $B = U_2\Sigma_2V_2^T$, que és la descomposició en valors singulars de B.
- Suposem que tenim $B = U_2\Sigma_2V_2^T$. Llavors, com $\Sigma_2 = \Sigma_1$ i $A = MBN^T$, tenim que $A = MU_2\Sigma_1V_2^TN^T$. Si diem que $U_1 = MU_2$ i $V_1^T = V_2^TN^T$ (aquestes dues matrius són ortogonals per ser producte d'ortogonals), tenim que $A = U_1\Sigma_1V_1^T$, que és la descomposició en valors singulars d'A □

Amb aquest corol·lari acabem de redüir, formalment, el problema de trobar la descomposició en valors singulars d'A al de trobar la descomposició en valors singulars de B, una matriu bidiagonal (i, per tant, molt més tractable que la matriu A). Anem més enllà: veurem que trobar el SVD de B és el mateix que trobar els valors i vectors propis d'una certa matriu tridiagonal que ens construirem:

Teorema: *Sigui $\hat{B} \in M_{n,n}(\mathbb{R})$ una matriu bidiagonal quadrada amb diagonal $a_1, a_2, ..., a_{n-1}, a_n$ i diagonal superior $b_1, b_2, ..., b_{n-2}, b_{n-1}$. Llavors, si construim la matriu tridiagonal simètrica $T_{ps} \in M_{2n,2n}(\mathbb{R})$ (coneguda com "perfect shuffle") tal que tota la seva diagonal és nul·la i la diagonal superior i inferior está formada per la seqüència $a_1, b_1, a_2, b_2, ..., a_{n-2}, b_{n-2}, a_{n-1}, b_{n-1} a_n$, tenim que els valors propis d'aquesta matriu tridiagonal són $\alpha_i = \pm\sigma_i$ (els valors singulars de $\hat{B}$ en positiu i negatiu) i, si x_i és el vector propi associat a $\alpha_i = \pm\sigma_i$, llavors $Px_i = \frac{1}{\sqrt{2}} \begin{pmatrix} v_i \\ \pm u_i \end{pmatrix}$, on $P = [e_1, e_{n+1}, e_2, e_{n+2}, ..., e_n, e_{2n}]$ és una matriu de permutació i u_i i v_i són els vectors singulars per la esquerra i la dreta de la matriu $\hat{B}$, respectivament. Així dons, si calculem la descomposició en valors i vectors propis de T_{ps}, tenim la descomposició en valors singulars de la matriu bidiagonal quadrada $\hat{B} \in M_{n,n}(\mathbb{R})$.*

Demostració: Veiem d'on surten les matrius T_{ps} i P de l'espai $M_{2n,2n}(\mathbb{R})$. Primer de tot definim una matriu $H = \begin{pmatrix} 0 & \hat{B}^T \\ \hat{B} & 0 \end{pmatrix}$. Considerem la matriu P de permutació definida a l'enunciat del teorema, $P = [e_1, e_{n+1}, e_2, e_{n+2}, ..., e_n, e_{2n}]$. Per ser matriu de permutació, P és ortogonal. Ara, veiem que:

$$P^T \cdot H \cdot P = \begin{pmatrix} 0 & a_1 & 0 & 0 & 0 & 0 \\ a_1 & 0 & b_1 & 0 & 0 & 0 \\ 0 & b_1 & 0 & a_2 & 0 & 0 \\ & & \ddots & & \ddots & \\ 0 & 0 & & 0 & & 0 \\ 0 & 0 & 0 & b_{n-1} & 0 & a_n \\ 0 & 0 & 0 & 0 & a_n & 0 \end{pmatrix} = T_{ps}$$

Podem veure com la matriu P està dissenyada per aprofitar la bidiagonalitat de $\hat{B}$ i fer de la matriu H (una matriu esparsa sense cap peculiaritat) una matriu tridiagonal simètrica amb la diagonal nul·la. De la expressió $P^T H P = T_{ps}$ tenim que les matrius H i T_{ps} són semblants: tenen, per tant, els mateixos valors propis. Ara recordem la quarta propietat del *Teorema de les propietats de la descomposició SVD* que hem enunciat a la primera part d'aquest informe. Aquesta propietat té una relació total amb el teorema que estem demostrant ara: directament ens diu que els valors propis de la nostra H són $\pm\sigma_i$ (la demostració d'aquesta propietat és a la demostració del *Teorema de les propietats de la descomposició SVD*), i per semblança també ho són de T_{ps}. A més a més, la propietat esmentada ens diu que els vectors propis de la matriu H són $\frac{1}{\sqrt{2}} \begin{pmatrix} v_i \\ \pm u_i \end{pmatrix}$. Per tant, com $T_{ps} = P^T H P \Leftrightarrow P T_{ps} = H P$, tenim que $Px_i = \frac{1}{\sqrt{2}} \begin{pmatrix} v_i \\ \pm u_i \end{pmatrix}$, acabant així de demostrar el teorema □

Un parell d'anotacions interessants:

- Es pot comprovar que una altra manera d'obtindre la descomposició en valors singulars de la matriu bidiagonal quadrada $\hat{B}$ es calcular $\hat{B}\hat{B}^T$ i $\hat{B}^T\hat{B}$, dues matrius simètriques tridiagonals pertanyents a $M_{n,n}(\mathbb{R})$. Les dues tenen els mateixos valors propis, sent les seves respectives arrels quadrades els valors singulars de $\hat{B}$. Els vectors propis de $\hat{B}\hat{B}^T$ són els vectors singulars per l'esquerra i els de $\hat{B}^T\hat{B}$ són els vectors singulars per la dreta. Pot semblar interessant considerar aquestes dues matrius per a calcular el SVD de $\hat{B}$ en comptes de la més gran matriu T_{ps}, però hi ha dos motius de pes per no fer-ho: com vam discutir al principi del disseny del nostre algorisme calcular productes de matrius concretes per les seves transpostes és una font d'erros numèrics (cap mètode estable fa aquesta operació, i tractar amb T_{ps} sí que ens proporciona un mètode estable), i a més no és necessari escriure la matriu T_{ps} explícitament, donat que només té $2n-1$ components significatives, que es poden enmagatzemar en una altra estructura de dades (com un vector, per exemple).
- Si teniem la propietat quatre del *Teorema de les propietats de la descomposició SVD* desde el principi... perquè no la hem intentat utilitzar abans? La resposta és simple: la matriu havia de ser quadrada. I, encara que ho fos (amb un simple QR modificat a la A ens bastaria, agafant la R), no seria una bona estrategia ja que la matriu H resultant seria difícil de tractar numèricament. El procés que hem seguit busca astutament que la matriu tingui la forma adequada per a poder buscar-hi valors i vectors propis sense correr riscos numèrics.

Per a buscar valors i vectors propis de matrius tridiagonals simètriques, existeixen diferents mètodes iteratius. Els més populars són la iteració QR i les seves variants, dividir i vèncer, bisecció i iteració inversa i el mètode de Jacobi, encara que hi han algorismes com la modificació FMM del mètode de dividir i vèncer [18] o l'algorisme RRR [19] estables extramadament competents (i complexos) que tenen cost $O(n^2)$. Nosaltres **hem escollit construir la matriu T_{ps} a partir de la nostra $\hat{B}$ tal i com el teorema anterior indica i a aquesta matriu tridiagonal simètrica li hem aplicat el mètode iteratiu QR.**

2.4 El mètode QR iteratiu

(*Per a aquesta secció hem utilitzat els treballs de Raymond J. Spiteri [20] i uns altres dos documents de Peter Blomgren [21] [22]*)

Començem aquest punt amb una matriu $T_{ps} \in M_{2n,2n}(\mathbb{R})$ de la que volem conèixer els seus valors i vectors propis (per simplicitat, als mètodes que describim més endavant diem per comoditat que $T_{ps} \in M_{n,n}(\mathbb{R})$ perque són generals i no específics d'aquesta matriu tridiagonal, encara que poguem seguir referint-nos a una matriu tridiagonal arbitrària com T_{ps} en comptes de T, ja que arrosegar aquest nom no ens fa nosa). Fem notar de l'existència d'un mètode iteratiu que ens transforma mitjançant matrius ortogonals una matriu arbitrària A fins que la fa convergir a la seva forma de Schur: triangular superior en general i diagonal si la matriu original és simètrica. Aquest és l'anomenat "Algorisme QR pur", que va fent sucesives factoritzacions QR de matrius per després obtindre una de nova a partir de considerar el producte $R \cdot Q$. El veiem en detall:

Algoritmo 2 Algorisme QR pur

Entrada: Una matriu $A \in M_{n,n}(\mathbb{R})$

Salida: La forma de Schur d'A: $\hat{A}$, triangular superior en general i diagonal si la A era simètrica

1: Creem la matriu d'iteració $A^{(0)} = A$
2: **para** $k = 1$ fins que $A^{(k)}$ sigui la forma de Schur d'A **hacer**
3: Obtenir $Q^{(k)} \cdot R^{(k)}$ factorització QR d'$A^{(k-1)}$
4: $A^{(k)} = R^{(k)} \cdot Q^{(k)}$
5: **fin para**
6: **devolver** La última matriu $A^{(k)}$

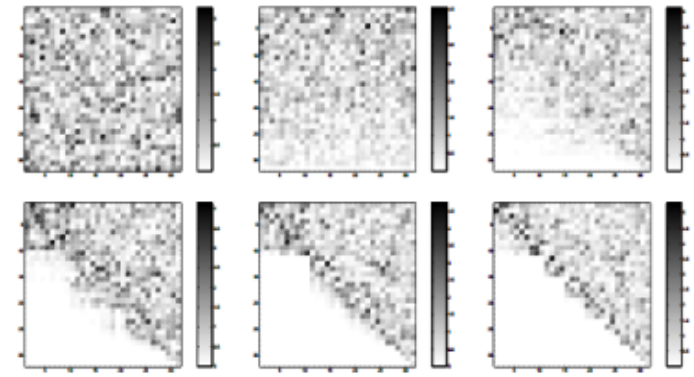

Figure: The QR-algorithm applied to a non-Hermitian matrix A. The panels show the initial matrix, and iterations 1, 4, 16, 32, and 64.

Figura 4: Gràfic extret de [21]

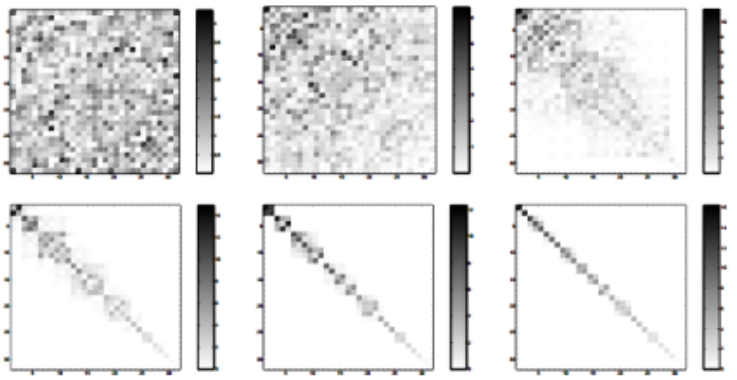

Figure: The QR-algorithm applied to a Hermitian matrix A. The panels show the initial matrix, and iterations 1, 4, 16, 32, and 64.

Figura 5: Gràfic extret de [21]

Per tant, si tenim la nostra matriu T_{ps} simètrica, aquest algorisme ens convergirà a una matriu diagonal. Veiem ara que aquesta matriu contindrà els valors propis de T_{ps}. Considerem el mètode de la potència en bloc, és a dir, en comptes d'anar aplicant a la nostra matriu un vector hi apliquem una altra matriu, que en realitat és un conjunt de n vectors linealment independents. D'aquesta manera, ens convergirà no només a la parella vector/valor propi corresponent al valor propi més gran en valor absolut, sinó que convergirà a les n parelles corresponents als n valors propis més grans en valor absolut (sempre que siguin diferents entre ells). A més, si volem que convergeixin a vectors propis ortonormal, l'únic que haurem de fer serà considerar la descomposició QR al final de cada iteració, i aixi les diferents Q tendiran als vectors propis ja ortonormalitzats. Tindrem doncs, que l'algorisme corresponent al mètode de la potència en bloc serà el següent:

Algoritmo 3 Mètode de la potència en bloc

Entrada: Una matriu $A \in M_{n,n}(\mathbb{R})$ i una matriu $V_0 \in M_{n,n}(\mathbb{R})$ qualsevol de rang màxim.
Salida: Una matriu Q de vectors propis de la que podem dedüir, mitjançant $A \cdot Q$, els seus valors propis associats.

1: **para** $k = 1$ fins que la matriu V_k s'estabilitzi **hacer**
2: $V_k = A^k \cdot V_{k-1}$
3: $V_k = Q_k \cdot R_k$ descomposició QR de V_k
4: $V_k = Q_k$, vectors de V_k normalitzats
5: **fin para**
6: **devolver** La última matriu Q_k, que conté els vectors propis però normalitzats (per aixo fem QR)

Tanmateix, aquest algorisme és inestable. Una millora consisteix en aconseguir una sèrie de matrius similars a les V_k de l'algorisme anterior i que ens portaran als mateixos resultats, però que estaran millor condicionades. L'algorisme corresponent a aquesta millora és el següent:

Algoritmo 4 Mètode de la potència en bloc millorat

Entrada: Una matriu $A \in M_{n,n}(\mathbb{R})$ i una matriu $Q_0 \in M_{n,n}(\mathbb{R})$ ortogonal com, per exemple $Q_0 = I_n$.
Salida: Una matriu Q de vectors propis de la que podem dedüir, mitjançant $A \cdot Q$, els seus valors propis associats.

1: **para** $k = 1$ fins que la matriu V_k s'estabilitzi **hacer**
2: $Z_k = A \cdot Q_{k-1}$
3: $Z_k = Q_k \cdot R_k$ descomposició QR de Z_k
4: **fin para**
5: **devolver** La última matriu Q_k, que conté els vectors propis normalitzats

Tal i com es pot veure a [20] o a partir de la pàgina 19 en [21], es pot demostrar que l'algorisme QR pur és equivalent a la versió millorada del mètode de la potència en bloc aquí descrit. Aleshores, si considerem $A_k = Q_k A_0 Q_k^T$ i definim el quocient de Rayleigh com $R(A, v) = \frac{v^T A v}{v^T v}$ (que quan és aplicat a un vep retorna el seu vap associat), donat que els elements de la diagonal de A_k són els quocients de Rayleigh d'A corresponents a les columnes de Q tindrem que ja que les columnes de Q tendeixen als vectors propis d'A, els elements de la diagonal de A_k tendiran als valors propis d'A associats a aquests veps. Per un raonament anàleg, es pot veure que els elements de fora la diagonal principal tendiran a zero.
Després d'haver vist l'equivalència entre el mètode de la potència en bloc millorat i l'algorisme de factorització QR, deixem de tractar amb una matriu A general i passem a la nostra matriu tridiagonal simètrica T_{ps} (que serà la nostra A_0).

Lema: *Si seguim aquest mètodes iteratius, acabarem tenint $T = QDQ^T$, que serà la descomposició en una matriu diagonal D de valors propis i una matriu Q ortogonal de vectors propis, tal i com asegura el teorema espectral (donat que T_{ps} és simètrica).*
Demostració: Comprovem a l'algorisme QR pur que es compleix $A_k = Q_k^T A_{k-1} Q_k$ i $T = \hat{Q} D \hat{Q}^T$: tenim que $A_k = R_k Q_k = Q_k^T Q_k R_k Q_k = Q_k^T A_{k-1} Q_k$ i que $A_{k-1} = Q_k A_k Q_k^T$. Aleshores, tindrem $T = A_0 = Q_1 Q_2 Q_3 \cdots Q_k A_k Q_k^T Q_{k-1}^T \cdots Q_2^T Q_1^T = \hat{Q_k} A_k \hat{Q_k}^T$, i com si $k \to \infty$ tenim que $\hat{Q_k} \to \hat{Q}$ i $A_k \to D$, llavors $T = \hat{Q} D \hat{Q}^T$, tal i com voliem veure □

Tal i com hem vist, a cada iteració de l'algorisme obtenim una matriu que no és més que la anterior multiplicada a l'esquerra per una matriu ortogonal Q_i a la dreta per la seva inversa (o transposada, ja que és ortogonal). Així doncs, tindrem que les matrius de cada iteració serán semblants, i això ens garanteix que tindran els mateixos valors propis i que es mantindrà l'estructura tridiagonal simètrica.

2.4.1 L'algorisme QR modificat

L'algorisme QR, encara que efectiu, pot arribar a requerir moltes iteracions per convergir, i per això convé afegir-li un paràmetre η o "shift" que acceleri la seva convergencia a cada iteració. D'aquesta manera obtindrem l'algorisme modificat, que té convergència cúbica i que és vàlid per a matrius tridiagonals:

Algoritmo 5 Algorisme QR modificat

Entrada: Una matriu $T_{ps} \in M_{n,n}(\mathbb{R})$ tridiagonal (si tinguessim una que no ho és, l'hauriem de fer tridiagonal mitjançant Hessemberg, per exemple).
Salida: Les matrius Q i D tals que $T = QDQ^T$.
1: $A_0 = T_{ps}$
2: **para** $k = 1$ fins que A_k sigui diagonal amb una tolerància fixada **hacer**
3: $A_{k-1} - \eta_k I_n = Q_k R_k$ factorització QR
4: $A_k = R_k Q_k + \eta_k I_n$
5: **fin para**
6: **devolver** La última matriu Q_k, que conté els vectors propis normalitzats, i la última matriu A_k, diagonal amb una tolerància fixada

Podem comprovar que, encara que afegim aquest paràmetre que triarem adientment més tard per tal d'accelerar la convergència, els resultats finals acaben sent els mateixos que els de l'algorisme QR pur, és a dir: tenim que $A_K = Q_k^T A_{k-1} Q_k$. Efectivament, $A_k = R_k Q_k + \eta_k I_n = Q_k^T Q_k R_k Q_k + \eta_k I_n = Q_k^T (A_{k-1} - \eta_k I_n) Q_k + eta_k I_n = Q_k^T A_{k-1} Q_k - Q_k^T \eta_k Q_k + \eta_k I_n = Q_k^T A_{k-1} Q_k$, com voliem veure.

Parlem ara del paràmetre η_k: la idea és triar un paràmetre el més proper possible a un dels valors propis de la matriu. Com que, a priori, no coneixem cap dels valos propis de la matriu, hem de triar paràmetres que funcionin de forma similar. Les dues opcions mes típiques (segons les referències que hem adjuntat) són:

- Prendre com a paràmetre l'últim element de la diagonal: això ens dóna convergència cúbica, però malauradament amb aquest paràmetre l'algorisme no sempre convergeix.
- Prendre el paràmetre de Wilkinson, que consisteix en prendre l'últim menor de mida 2 de la matriu que tenim, calcular-ne els valors propis, i triar el que sigui més proper a l'últim element de la diagonal. Aquest paràmetre si que ens garanteix que l'algorisme convergeixi sempre, tot i que en casos dolents la convergència serà quadràtica i no cúbica. Una expressió numèricament estable pel paràmetre de Wilkinson és:
  $\eta_k = a_n - \frac{sign(\delta) b_{n-1}^2}{|\delta| + \sqrt{\delta^2 + b_{n-1}^2}}$, on $\delta = \frac{a_{n-1} - a_n}{2}$ i on a_n, a_{n-1} i b_{n-1} corresponen al menor següent de la matriu

de la que estem calculant el paràmetre (les diferents A_k segons l'iteració, que tal i com hem vist abans $A_0 = T_{ps}$ i les altres també son tridiagonals): $\begin{pmatrix} a_{n-1} & b_{n-1} \\ b_{n-1} & a_n \end{pmatrix}$.

2.4.2 L'algorisme QR pràctic

Pel que hem fins ara, sembla que la millor opció seria utilitzar l'algorisme QR amb el paràmetre de Wilkinson, però per fer-lo més pràctic i eficient, es poden introduir dues millores addicionals.

Per una banda, com que la nostra matriu T_{ps} és tridiagonal i simètrica, si a mesura que anem fent iteracions veiem que un dels elements de la supradiagonal (i per simetria, de la infradiagonal), és prou proper a 0, entenent per això que és inferior a una certa tolerància fixada (com estem utilitzant un algorisme iteratiu, tenim una tolerància fixada), podem convertir aquest element en un 0 i obtenir així dues matrius tridiagonals i simètriques a partir de la nostra, just considerant dins d'ella blocs diferents de matrius a partir de l'element de la infradiagonal (i supradiagonal) que hem vist que era zero amb la nostra tolerància fixada. Aleshores, la idea és seguir aplicant l'algorisme QR modificat a aquestes dues noves matrius, i després juntar-ho tot de nou. D'aquesta manera, es poden estalviar moltes operacions a canvi d'un error despreciable, si triem una bona tolerància. Seguint aquesta idea, l'algorisme **recursiu** obtingut seria el següent:

Algoritmo 6 Algorisme QR pràctic

Entrada: Una matriu $T_{ps} \in M_{n,n}(\mathbb{R})$ tridiagonal (si no ho és fem Hessemberg).
Salida: Les matrius Q i D tals que $T = QDQ^T$.

1: $A_0 = T_{ps}$
2: **si** la nostra matriu es tan petita que la seva tridiagonalització és trivial **entonces**
3: **devolver** Matrius Q_0 i A_0 tals que $A = Q_0^T A_0 Q_0$
4: **fin si**
5: **para** $k = 1$ fins que algun $a_{j,j+1}^{(k)}$ sigui més petit que una certa tolerància **hacer**
6: $A_{k-1} - \eta_k I_n = Q_k R_k$ factorització QR
7: $A_k = R_k Q_k + \eta_k I_n$
8: **fin para**
9: Per aquest $a_{j,j+1}^{(k)}$ trobat, fem $a_{j,j+1}^{(k)} = 0$ i separem la nostra matriu per blocs $A_k = \begin{pmatrix} A_k^1 & 0 \\ 0 & A_k^2 \end{pmatrix}$
10: **Cridem una altra vegada a l'algorisme QR pràctic** amb les noves matrius tridiagonals i simètriques A_k^1 i A_k^2
11: Si arribem aquí, ja hem trobat les descomposicions de totes les matrius triangulars generades, i juntant menors obtenim la nostra descomposició $T = QDQ^T$.
12: **devolver** Matrius Q i D tals que $T = QDQ^T$.

Aquesta era la primera millora, però encara ens queda la segona: un altre tema a tenir en compte es que l'algorisme QR, com el seu nom indica, utilitza moltes vegades (de fet, a cada iteració) la descomposició QR convencional. Donat que treballem amb matrius tridiagonals i simètriques (hem definit $A_0 = T_{ps}$ tridiagonal simètrica i hem vist que les A_k també ho són), aquesta informació es pot aprofitar per triar una descomposició adaptada a aquest tipus de matrius i estalviar així moltes operacions, ja que una matriu tridiagonal simètrica és esparsa, però a més a més sabem com estan distribuits els zeros de la nostra matriu (més enllà de la diagonal, supradiagonal i infradiagonal). Si usem l'algorisme de Gram-Schmidt modificat per obtenir la descomposició QR, el podem modificar (encara més) de la forma següent:

Algoritmo 7 Factorització QR eficient per a matrius tridiagonals

Entrada: Una matriu $T \in M_{n,n}(\mathbb{R})$ tridiagonal
Salida: Dos matrius $Q, R \in M_{n,n}(\mathbb{R})$ tals que $T = QR$ amb Q ortogonal i R triangular superior.

1: **para** $k = 1$ fins a n **hacer**
2: $r_{k,k} = ||a_k||_2$ i $q_k = \frac{a_k}{r_{k,k}}$
3: $r_{k,s} = q_k^T a_s$ i $a_s = a_s - r_{k,s} q_k$ per a $s = k+1$ i $s = k+2$.
4: **fin para**
5: **devolver** Les matrius $Q, R \in M_{n,n}(\mathbb{R})$ de la factorització $T = QR$

Us recomanem aquesta genial referència [23], que és un article de recerca sobre la factorització QR per a matrius tridiagonals. Allà asseguren que l'algoritme per a fer la factorització QR adaptada a matrius tridiagonals té un cost de $O(n^2)$, contra el cost dels convencionals (com a mínim $O(n^3)$).

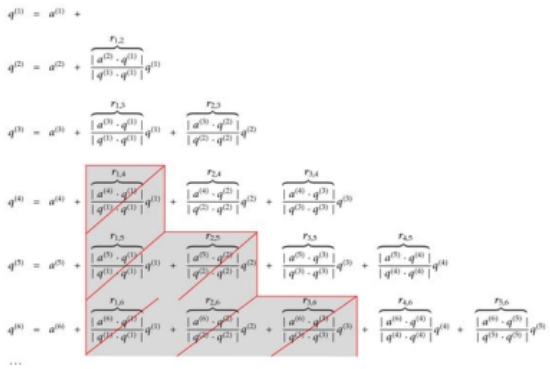

Figure 1: The zero and nonzero terms in the equation of each orthogonal column, where $q^{(m)}$ is the mth column of orthogonal matrix and $a^{(m)}$ is mth column of matrix A. Also, the zero terms are crossed out.

Figura 6: Gràfic extret de [23] amb el que ens fem la idea de quantes operacions innecesaries fem si apliquem el QR estàndard a una matriu tridiagonal

Definitivament, el nostre algorisme per trobar la descomposició en valors i vectors propis d'una matriu tridiagonal simètrica $T_{ps} \in M_{2n,2n}(\mathbb{R})$ és l'*Algorisme QR pràctic* amb la factorització QR adaptada per a matrius tridiagonals que té un cost cúbic en la majoria de casos.

2.5 Final del nostre algorisme per a calcular la descomposició en valors singulars (SVD)

Tornant a l'eix del nostre problema de trobar els valors i vectors propis de la nostra matriu tridiagonal T_{ps}, podem considerar l'*Algorisme QR pràctic* amb la factorització QR adaptada per a matrius tridiagonals, tal i com l'hem descrit a l'apartat anterior. Si el fem, el següent pas serà agafar els valors propis positius (que són exactament n, tal i com vam veure al Teorema anunciat a l'apartat *Transformant el problema de la descomposició SVD en un de valors i vectors propis*: $\alpha_i = \pm\sigma_i$) i posar-los ordenats a la matriu Σ, i omplir les matrius U_2 i V_2 amb els valors singulars per la dreta i per la esquerra, que s'obtenen mitjançant el mateix Teorema al que estem fent ara referència.

Ara mateix ja tenim, llavors, la nostra factorització $B = U_1\Sigma V_1^T$. Amb això ja podem donar la nostra descomposició en valors singulars segons d'on provingués la nostra matriu $\hat{B}$:

- Si haviem aplicat l'algorisme de Golub-Kahan tenim $A = MBN^T$, amb $B = \begin{pmatrix} \hat{B}_{n,n} \\ 0_{m-n,n} \end{pmatrix}$. Tenim que $\hat{B} = U_1\Sigma V_1^T$, d'on $B = \begin{pmatrix} U_1 \\ 0_{m-n,n} \end{pmatrix} \Sigma V_1^T = \begin{pmatrix} U_1\Sigma V_1^T \\ 0_{m-n,n} \end{pmatrix} = \begin{pmatrix} \hat{B}_{n,n} \\ 0_{m-n,n} \end{pmatrix}$. Així dons, tindrem que: $A = MBN^T = M \begin{pmatrix} U_1 \\ 0_{m-n,n} \end{pmatrix} \Sigma V_1^T N^T$, d'on si $U = M \begin{pmatrix} U_1 \\ 0_{m-n,n} \end{pmatrix}$ (ortogonal per ser producte d'ortogonals), amb $U \in M_{m,n}(\mathbb{R})$ i $V = NV_1$ (ortogonal per ser producte d'ortogonals), amb $V \in M_{n,n}(\mathbb{R})$, llavors tenim una factorització del tipus $A = U\Sigma V^T$ que compleix totes les condicions de la descomposició en valors singulars. Per tant, ja hem trobat la nostra descomposició □
- Si haviem aplicat l'algorisme de Lawson-Hanson-Chan, tenim $A = Q(MBN^T)$, amb $B = \hat{B} = U_1\Sigma V_1^T$. Per tant: $A = Q(MBN^T) = QMU_1\Sigma V_1^T N^T$, d'on si diem $U = QMU_1$ (ortogonal per ser producte d'ortogonals), amb $U \in M_{m,n}(\mathbb{R})$ i $V = NV_1$ (ortogonal per ser producte d'ortogonals), amb $V \in M_{n,n}(\mathbb{R})$, llavors tenim una factorització del tipus $A = U\Sigma V^T$ que compleix totes les condicions de la descomposició en valors singulars. Per tant, ja hem trobat la nostra descomposició □

L'esquema algorítmic de la nostra descomposició SVD seria el següent:

Algoritmo 8 Descomposició en valors singulars (SVD)

Entrada: Una matriu $A \in M_{m,n}(\mathbb{R})$ qualsevol
Salida: Tres matrius $U \in M_{m,n}(\mathbb{R})$, $\Sigma \in M_{n,n}(\mathbb{R})$ i $V \in M_{n,n}(\mathbb{R})$ tals que $A = U\Sigma V^T$ és el SVD d'A.

1: **si** $\frac{m}{n} \leq \frac{5}{3}$ **entonces**
2: Trobem M, B, N tals que $A = MBN^T$ amb la bidiagonalització de Golub-Kahan.
3: Agafem la matriu bidiagonal quadrada $\hat{B}$ (menor de la matriu B) i construim la matriu T_{ps} associada.
4: Trobem els vaps i veps de T_{ps} mitjançant l'Algoritme QR pràctic amb la factorització QR per a matrius tridiagonals.
5: A partir de la descomposició en valors i vectors propis de T_{ps}, trobem el SVD de $\hat{B} = U_1\Sigma V_1^T$.
6: **devolver** Les matrius $U = M \begin{pmatrix} U_1 \\ 0_{m-n,n} \end{pmatrix}$, $V = NV_1$ i Σ, que són les que busquem (tals que $A = U\Sigma V^T$).
7: **fin si**
8: **si** $\frac{m}{n} > \frac{5}{3}$ **entonces**
9: Trobem Q, M, B, N tals que $A = Q(MBN^T)$ amb la bidiagonalització de Lawson-Hanson-Chan.
10: Agafem la matriu bidiagonal quadrada B i construim la matriu T_{ps} associada.
11: Trobem els vaps i veps de T_{ps} mitjançant l'Algoritme QR pràctic amb la factorització QR per a matrius tridiagonals.
12: A partir de la descomposició en valors i vectors propis de T_{ps}, trobem el SVD de $B = U_1\Sigma V_1^T$.
13: **devolver** Les matrius $U = QMU_1$, $V = NV_1$ i Σ, que són les que busquem (tals que $A = U\Sigma V^T$).
14: **fin si**

2.6 Problemes a la implementació del nostre algorisme

Moltes vegades el bon disseny d'un algorisme es pot veure malmès per la necesitat d'una complexa implementació. Nosaltres vam intentar implementar el nostre algorisme tal i com l'hem descrit, tot i conèixer que no és gens trivial. Ens semblava un repte personal que ens ha fet adquirir coneixements i maduresa, renunciant a utilitzar vies més accesibles com les rotacions de Givens. Fer la part de bidiagonalització de la matriu, encara que hem agafat una modificació de l'algorisme de Golub-Kahan convencional, no era la pitjor fase, donat que no deixa de ser un procés directe (amb un número finit d'iteracions). Per desgràcia, les complicacions estaven a l'algorisme del QR iteratiu. No vam poder fer funcionar el nostre programa de la manera que voliem i ens hem quedat sense més temps per a continuar intentant-lo (no descartem, quan acabi el quatrimestre, millorar la nostra implementació de l'algorisme aquí descrit per tal de que el nostre codi funcioni amb normalitat). Pero això no ens va suposar el final: vam intentar fer una lleugera simplificació de l'algorisme que el fa funcional per a matrius de mida raonable, tot agafant un paràmetre que anem modificant a la pràctica fins que trobem els seu valor òptim. Entrem en detalls:

L'algorisme QR pràctic, encara que és molt eficient des d'un punt de vista computacional és difícil d'implementar, sobretot la part recursiva, on es passa a treballar amb dues matrius enlloc d'una, ja que a l'hora d'obtenir les matrius ortogonals de les successives iteracions s'han d'omplir de manera adient amb uns i zeros per tal de quadrar les dimensions. A més, cal saber triar la tolerància, ja que si es tria massa gran estarem fent zeros elements que no haurien de ser-ho, i si es tria massa petita correm el risc de necessitar moltes iteracions i això implica un gran error numèric (com hem pogut comprovar).

Com que la implementació era complicada, vam decidir fer proves amb l'algorisme sense utilitzar aquesta part recursiva, i ens vam donar compte que, la majoria de vegades, l'element que més ràpidament es feia 0 era l'últim de la infradiagonal (que és el mateix que el de la supradiagonal). Suposant que això fos cert, la implementació de l'algorisme seria més senzilla, ja que a l'hora de d'aplicar la recursió, una de les matrius constaria sols d'un element, i per tant la seva descomposició seria trivial i només hauriem d'anar utilitzant una matriu en tot moment (no dos a la vegada).

Ara bé, com hem dit, això no sempre és cert, i aquesta versió simplificada de l'algorisme funciona en general per matrius petites si triem una tolerància adequada per a cada matriu que ens donen, però a vegades cal posar una tolerància massa gran, induïnt així a un error de càlcul que es va fent gran a mesura que fem iteracions, i per això com més gran sigui la matriu pitjors seran els resultats obtinguts.

Tota aquesta situació ens ha portat a fer un ràpid i eficient algorisme per a matrius de mida raonable però

un mètode amb bastants erros acumulats si la matriu és gran.

3 Exercicis: aplicacions pràctiques de la descomposició en valors singulars (SVD)

En els pròxims dos exercicis hem aplicat l'algorisme per a trobar la descomposició en valors singulars que hem dissenyat a la secció anterior. Tal i com hem dit a l'apartat *"Problemes a la implementació del nostre algorisme"*, hem vist que la nostra implementació de l'algorisme descrit (que és totalment vàlid i dona un resultat força satisfactori) té alguns problemes que no hem pogut arreglar durant el curt període que se'ns ha estipulat per a fer la pràctica. Això ha tingut conseqüències: l'exercici de mínims quadrats l'hem pogut fer però el de la compresió de fotografies ens ha donat bastants problemes per la gran mida de la nostra matriu. Tot i que el nostre algorisme és força ràpid els resultats que ens proporcionava no eren els esperats i no ens permetien fer una compresió de la nostra fotografia (dels dos integrants d'aquest grup). **Hem optat per fer la compressió d'imatges íntegra amb MATLAB** (i així ho indiquem) ja que creiem que és interessant jugar amb les propietats de la descomposició SVD i veure, per nosaltres mateixos, la multitut d'aplicaccions pràctiques que aquesta té. La compresió d'imatges és una de les més divertides, i per tant no voliem renunciar a fer-la encara que amb el nostre programa propi no funcionés bé.

3.1 Problema de mínims quadrats

El següent problema serà fet amb la factorització QR i la SVD tot comparant els resultats que ens donin cada descomposició:

Les xifres oficials corresponents al padró municipal de Catalunya (vegeu la pàgina web de l'Institut d'Estadística de Catalunya IDESCAT) entre els anys 2000 i 2010 són aquestes:

2000	2001	2002	2003	2004	2005	2006	2007
6.261.999	6.361.365	6.506.440	6.704.146	6.813.319	6.995.206	7.134.697	7.210.508

2008	2009	2010
7.364.078	7.475.420	7.512.381

Feu un estudi de l'aproximació i extrapolació de dades de la taula segons indiquen les següents preguntes:

Abans, un plantejament inicial del problema: en un problema d'aproximació de funcions com és el d'aproximar una taula de punts per un polinomi de grau escaient tenim sempre que tractar un sistema sobredeterminat que en general no té solució. Efectivament, tenim que si a és el vector de coeficients del polinomi i A és la matriu de Vandermonde de les abcisses desde el grau zero fins el grau del polinomi que estem aproximant, volem $Aa = b$, on b és el vector d'imatges. Aquest sistema és sobredeterminat excepte si el número de punts és el mateix que el grau del polinomi (cosa que no ha de passar mai perquè el Teorema d'interpolació ens assegura que el polinomi interpolador té como a màxim grau el nombre de punts menys u, i a més a més és únic, sent per tant $Aa = b$ sempre sobredeterminat). Per tant, la millor aproximació a la solució generalment inexistent és a l'espai generat per les columnes d'A: A^TA, d'on si $Aa = b$ ara plantejem $A^TAa = A^Tb$. Aquí podem utilitzar:

- La factorització QR (amb Q ortogonal i R triangular superior) pot ser emprada per resoldre el nostre sistema: $A^TAa = A^Tb \Leftrightarrow R^TQ^TQR = R^TQ^Tb \Leftrightarrow Ra = Q^Tb$.
- La descomposicio en valor singulars, tal i com hem vist i demostrat al cinquè punt del *Teorema de les propietats de la descomposició SVD* de la primera secció, ens dona directament la x^* millor solució per al problema $\min_{x\in\mathbb{R}^n} ||Ax - b||_2 = ||Ax^* - b||_2$. Aquí tindrem que la nostra a òptima serà $a^* = V\Sigma^{-1}U^Tb$ (de fet, si substituim $A = U\Sigma V^T$ en el sistema anterior $A^TAa = A^Tb$ i aillem ens acaba quedant el mateix).

Ara ja tenim les eines per buscar el polinomi del grau que volguem que millor aproxima la taula de punts. Recordem que l'error de l'aproximació polinòmica ve definit per: $||f - p_n(x)||_2 = \sqrt{\sum_{k=0}^{m}(f(x_k) - p_n(x_k))^2}$, on f és la funció de la població de Catalunya (de la que coneixem només els valors en els punts donats) i m és tal que al principi ens donen $m + 1$ punts $x_0, x_1, ...x_m$. En aquest cas en particular, $m = 10$. Per tant, **considerarem els polinomis de grau zero fins a grau deu**.

(i) Busqueu el polinomi del grau escaient que millor aproximi aquesta taula en norma sub-2. Justifiqueu l'elecció mostrant una fita de l'error d'aquest i de la resta amb els que heu provat.

Primer de tot, hem construit tots els polinomis dels graus possibles (des de grau zero fins a grau deu, que aquest segur que serà l'interpolador, que a més a més és únic) amb la factorització QR i la descomposició en valors singulars seguint el mètode descrit just a dalt, i hem fet vàries taules amb l'error comès:

La taula associada a l'error (en persones) comès a l'aproximació de polinomis **mitjançant la factorització QR** és de:

Grau del polinomi	0	1	2	3	4	5
Error comès (persones)	1400863.7329	143451.5819	92720.3509	67167.8465	65506.0820	53056.3173

Grau del polinomi	6	7	8	9	10
Error comès (persones)	50966.2710	48552.9492	48379.8961	9043.5036	$8.61 \cdot 10^{-6}$

La taula associada a l'error (en persones) comès a l'aproximació de polinomis **mitjançant la descomposició en valors singulars** és de:

Grau del polinomi	0	1	2	3	4	5
Error comès (persones)	1400863.7329	143451.5820	92720.3511	67167.8467	65506.0825	53056.3179

Grau del polinomi	6	7	8	9	10
Error comès (persones)	50966.2721	48552.9507	48380.9347	9092.3541	3.5235

La taula associada a la comparativa entre els errors comesos al QR i al SVD per als diferents polinomis, on e^i_{QR} i e^i_{SVD} representen l'error en l'aproximació per al polinomi de grau i-èsim mitjançant la factorització QR i la descomposició en valors singulars, respectivament:

Grau del polinomi	0	1	2	3	4	5
Quocient $\frac{e^i_{SVD}}{e^i_{QR}}$	1	$1 + 6.97 \cdot 10^{-10}$	$1 + 2.16 \cdot 10^{-9}$	$1 + 2.98 \cdot 10^{-9}$	$1 + 7.63 \cdot 10^{-9}$	$1 + 1.13 \cdot 10^{-8}$

Grau del polinomi	6	7	8	9	10
Quocient $\frac{e^i_{SVD}}{e^i_{QR}}$	$1 + 2.16 \cdot 10^{-8}$	$1 + 3.09 \cdot 10^{-8}$	$1 + 2.15 \cdot 10^{-5}$	$1 + 5.40 \cdot 10^{-3}$	409233.4494

Com podem observar, la factorització QR ens dona una millor aproximació per al problema de mínims quadrats que la descomposició en valors singulars, **en el nostre cas en particular** i amb les implementacions de cada factorització que nosaltres hem fet. Encara que la descomposició en valors singulars és més completa que la descomposició QR nosaltres ja esperàvem que ens donés un pitjor resultat el SVD que el QR, degut als problemes descrits a l'apartat anterior i a que, al nostre algorisme, no deixem de fer molts QR i això va arrossegant cada vegada més errors. Ens hem quedat sobtats pel resultat que ens ha donat el polinomi de grau 10 (i interpolador) amb la SVD, ja que té un error més baix que els altres però molt més gran que zero (o una bona aproximació de zero, com l'error al polinomi de grau 10 procedent del QR). Això es degut a errors d'estabilitat al nostre algorisme, tal i com hem explicat abans: l'error va decreixent a mida que anem aumentant el grau del polinomi (això ho fa bé, la taula ha de funcionar així), però no arriba a decreixer tant com per tindre un número proper a zero al polinomi interpolador.

Per acabar, encara que polinomis de grau més gran ens donin menys error, escollim els polinomis de grau tres obtinguts amb les dos factoritzacions pels dos motius següents:

- Ens donen un error més gran que els polinomis de grau més gran, pero trobem que comparant el número d'operacions necesàries amb aquest polinomi i els altres, la relació error-operacions és més que raonable.
- Si després volem extrapolar, hem de tenir en compte que el polinomi no tingui un grau massa gran, ja que el fenòmen de Runge ens adverteix que quan més a prop estem del polinomi interpolador millors aproximacions tindrem dels valors dels punts x_k escollits a l'eix d'abcisses però pitjor comportament tindrà aquest polinomi als altres punts, podent arribar a donar el polinomi una gràfica caòtica.

(ii) Useu-lo per a extrapolar la població de Catalunya l'any 2011 i compareu-la amb la xifra oficial (7.535.251). Quina seria la predicció pel 2012?

Si utilitzem el polinomi obtingut amb la factorització QR: $p_3(x) = 6248858.4056 \cdot x^0 + 123716.7483 \cdot x^1 + 8461.6247 \cdot x^2 - 813.2331 \cdot x^3$, que ens dona una previsió per l'any 2011 d'una població de $p_3(11) = 7551185.9697$ (7551186 persones), sent la dada real de 7535251 persones. Per tant, veiem que hi ha una diferència de només 15935 persones. La predicció per a l'any 2012 seria de $p_3(12) = 7546666.5455$ (7546667 persones). Ara sabem que a l'any 2012 la dada va ser de 7565603 persones, per tant hi ha una diferència de 18936 persones, una xifra molt

bona tenint en compte que estem ja a distància dos de l'últim punt que hem utilitzat a la taula (el $x_10 = 10$). Per tant podem veure com no ens hem equivocat escollint aquest polinomi. De fet, experimentalment hem comprovat que els polinomis de grau més petit són "massa lineals", i els de grau més gran comencen a tenir comportaments irreals més enllà del 2011 (punt $x = 11$).

Amb la descomposició en valors singulars es pràcticament anàleg, amb xifres tan semblants que sense posar molts decimals és impossible de veure la diferència, així que optem per dir que l'estudi del QR per al polinomi de grau 3 es prácticament el mateix que l'estudi del SVD per al polinomi de grau 3, així que descartem posar dues vegades la mateixa cosa. De fet, l'error al polinomi de grau 3 del QR és de 67167.8465 i al del SVD és de 67167.8467, pràcticament impossible de veure. També podem comparar el polinomi de grau 3 obtingut abans $p_3(x) = 6248858.4056 \cdot x^0 + 123716.7483 \cdot x^1 + 8461.6247 \cdot x^2 - 813.2331 \cdot x^3$ amb l'obtingut amb la descomposició en valors singulars $p_3^{'}(x) = 6248858.4056 \cdot x^0 + 123716.7483 \cdot x^1 + 8461.6247 \cdot x^2 - 813.2331 \cdot x^3$, iguals si no incloem molts decimals als coeficients.

(iii) Feu una gràfica on apareguin els punts de la taula i les gràfiques de les tres millors aproximacions.

Com la nostra millor aproximació la dona el polinomi de grau 3, hem decidit agafar com els altres polinomis els de grau 2 i 4 (que hem vist que tenen un bon comportament a l'intèrval $[0, 10]$ i a punts més llunyans). Per generar la gràfica, hem guardat al nostre programa els coeficients dels polinomis de grau 2, 3 i 4 generats per la descomposició en valors singulars en un vector i hem creat un programa que anés evaluant (mitjançant la regla de Horner) aquests tres polinomis al llarg de l'interval $[0, 12]$ (havent agafat 1200 punts equiespaiats) i escrivint el resultat en un arxiu *.txt* que després hem passat al programa GNUPLOT. Aquest és el resultat:

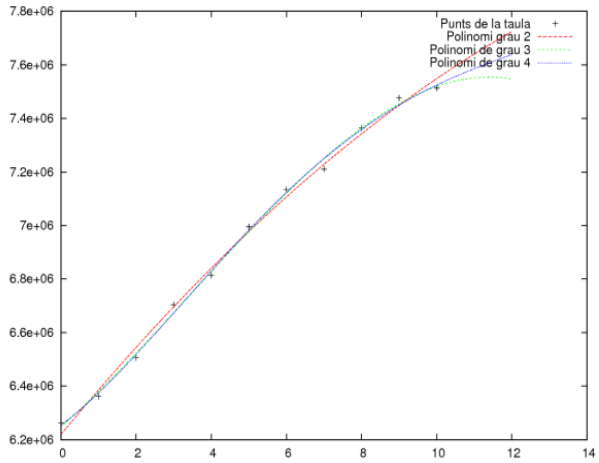

Figura 7: Gràfica dels punts de la taula i els polinomis de grau 2, 3 i 4 que l'aproximen

Veiem com les nostres prediccions es compleixen bé: aquest polinomis fan una interpolació acceptable dels punts i extrapolen molt bé. No adjuntem el mateix gràfic però amb els polinomis generats per la factorització QR donat que és totalment idèntica, per motius deduibles de la discusió feta al primer apartat.

(iv) Considereu el cas de $p_{10}(x)$, polinomi de grau 10 obtingut per mínims quadrats. Representeu-lo juntament amb els punts de la taula (aquests amb una rodona). Que observeu? Comenteu-lo.

Agafarem els dos polinomis de grau 10 donats per la descomposició en valors singulars i la descomposició QR (hem vist al primer apartat que hi havia discrepàncies a l'error entre un i l'altre) i construirem la gràfica tal i com ho hem fet a l'apartat anterior, també amb GNUPLOT. Aquest és el resultat:

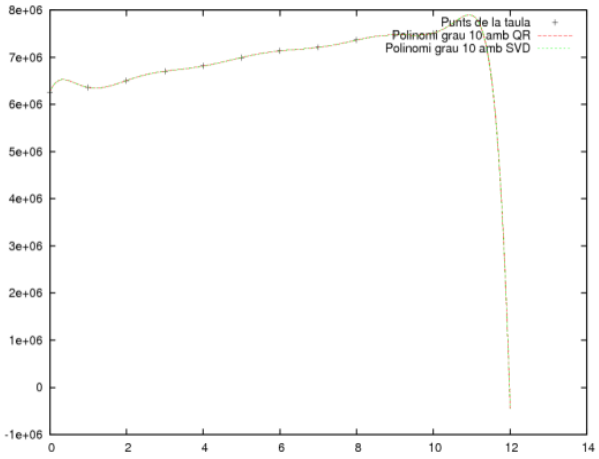

Figura 8: Gràfica dels punts de la taula i els polinomis de grau 2, 3 i 4 que l'aproximen

Potser pensàvem que veuriem les gràfiques dels dos polinomis diferents, pero vist el gràfic adjunt és obvi que no es així donat que la nostra gràfica tracta amb números de població molt grans, i encara que l'algorisme QR calculi el polinomi interpolador amb un error 409233.4494 vegades millor que el del polinomi interpolador calculat amb la factorització SVD, l'error d'aquest últim polinomi no deixa de ser de 3.5235 persones, una xifra ínfima en una gràfica que varia "a ritme" de decenes de milers de persones.

L'apreciació important i comuna a les dues gràfiques és el conegut fenòmen de Runge: la gràfica interpola bé als punts, però fora de l'intèrval $[0, 10]$ marxa ràpidament cap als negatius, pel que si fem una predicció per a l'any 2012 (com vam fer a l'apartat número dos d'aquest exercici) ens trobarem amb un número de població negativa, tenint un error colosal. Recordem que l'error pel polinomi de grau 3 extrapolat a l'any 2012 és, com hem vist abans, de 18936 persones. Per tant, com a conclusió, podem dir que no sempre agafar un polinomi de grau elevat ens portarà a millors resultats: aquest estudi demostra que els polinomis adients per a extrapolar taules de dades són els que tenen un grau moderat, encara que tinguin un error més elevat que els polinomis de grau gran (també la forma d'escollir com mesurar l'error afavoreix als polinomis que interpolen, no donant aquest error una mostra significativa de com pot ser de bo o dolent un polinomi fora d'aquest $m + 1$ punts.

3.2 Compresió d'imatges

Per a entendre la compresió d'imatges mitjançant aproximacions de matrius de diferents rangs, s'ha utilitzat una interessant propietat que recordem:

Propietat: *Considereu la descomposició SVD d'una matriu* $A \in M_{m,n}(\mathbb{R})$, $A = U\Sigma V^T$ *i escriviu* $U = [u_1, u_2, ..., u_n]$ *i* $V = [v_1, v_2, ..., v_n]$. *Observeu que podem escriure, equivalentment, que* $A = U\Sigma V^T = \sum_{i=1}^{n} \sigma_i u_i v_i^T$, *essent* $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$, $\sigma_{r+1} = \sigma_{r+2} = \cdots = \sigma_n = 0$. *Aleshores* $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T = U\Sigma_k V^T$, *amb* $\Sigma_k = diag(\sigma_1, \sigma_2, ..., \sigma_k, 0, ..., 0)$, *és la matriu de rang k que millor aproxima A en norma euclídea. A més,* $||A - A_k||_2 = \sigma_{k+1}$.

Demostració: Inclosa a l'enunciat de la pràctica.

Amb les comandes especificades a l'informe de la pràctica, el que fem és passar la nostra fotografia a escala de grisos i després treure una matriu d'intensitats que ens diu quan de clar o fosc es cada píxel de la nostra fotografia. La idea és que podem fer que la matriu d'intensitats sigui més lleugera a canvi de perdre qualitat de la imatge. Això significarà afagar una matriu amb rang més petit del que té la matriu d'intensitats inicial i tal que sigui una bona aproximació d'aquesta matriu d'intensitats. L'aproximació òptima ens la dona, precisament, la propietat anterior. Anomenem A a la matriu d'intensitats inicial de la fotografia escollida en escala de grisos. Si calculem la seva descomposició $A = U\Sigma V^T$, com a la nostra matriu Σ els valors singulars estàn ordenats de més gran a més petit la intuició ens sembla dir que no hi haurà quasi diferència entre els A_k i A a partir d'uns certs k tals que $\sigma_{k+1}, \sigma k + 2, ..., \sigma n$ són **quasi** zero (per la construcció de la nostra matriu Σ_k). Així dons, si això fos cert, estariem tractant amb una matriu A de rang massa gran, podent agafar una matriu A_k adient que només dista σ_{k+1} d'A en norma sub-2 (un número molt petit si la k és encertada). I, per comprovar que diem la veritat, si

recuperem la foto d'on surt la matriu d'intensitats A a partir dels diferents A_k, tindriem que hi ha molt poca diferència entre la foto original i les generades pels A_k a partir d'un cert k. Comprovarem que tot això és cert, experimentalment, amb una fotografia dels dos integrants del grup:

Figura 9: Fotografia de Ramon Celma Cercadillo i David Olivé Farga, de dimensions 640x480

Una vegada passada a escala de grisos i enmagatzemada la seva matriu d'intensitats A en un fitxer extern, calculem la descomposició en valors singulars d'aquesta matriu i generem, tal i com se'ns indica a la propietat abans comentada, les matrius $A_1, A_2, A_5, A_{10}, A_{25}, A_{100}$ i A_{298}. Les seves fotografies associades són aquestes:

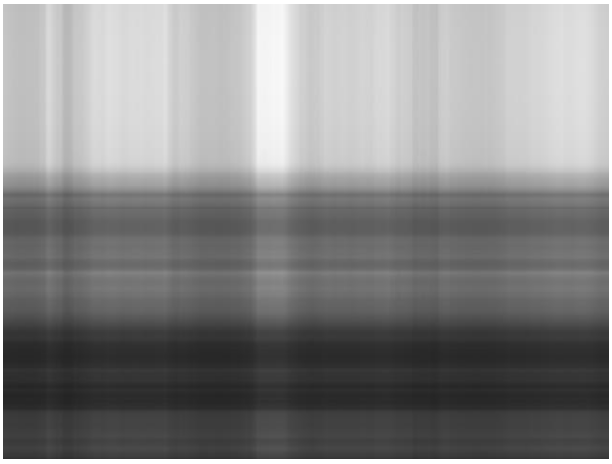

Figura 10: Aproximació de la fotografia inicial utilitzant el primer valor singular

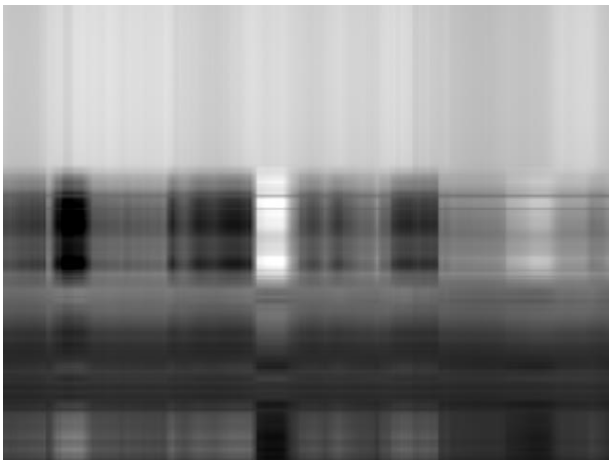

Figura 11: Aproximació de la fotografia inicial utilitzant els dos primers valors singulars

Figura 12: Aproximació de la fotografia inicial utilitzant els cinc primers valors singulars

Figura 13: Aproximació de la fotografia inicial utilitzant els deu primers valors singulars

Figura 14: Aproximació de la fotografia inicial utilitzant els vint-i-cinc primers valors singulars

Figura 15: Aproximació de la fotografia inicial utilitzant els cent primers valors singulars

Figura 16: Aproximació de la fotografia inicial utilitzant els dos-cent noranta-vuit primers valors singulars

Observem que, tal i com vam predir, a partir de l'aproximació A_{100} ja tenim una representació bastant fideligna de la fotografia original. Només amb una matriu de rang 100 ja tenim un resultat acceptable (sent la matriu d'intensitats original A, a més a més, de dimensions 640x480)! Com a conseqüència d'això, les aproximacions a la fotografia original a partir d'A_k amb $k \geq 100$ són bastant semblants a A_{100} (només milloren una mica el "soroll" de la fotografia, una diferència que no és fàcil d'apreciar visualment). Un exemple d'això és la comparació que podem fer entre l'aproximació de la fotografia inicial utilitzant els cent i els dos-cents noranta-vuit primers valors singulars: quasi no hi ha cap diferència, excepte que la segona fotografia es una mica més nítida (si ens fixem bé, a simple vista sembla la mateixa).

A nivell de pes de la fotografia, la original té un pes de **75,0 kB (75.016 bytes)** i la aproximació amb els cent primers valors singulars té un pes de **27,9 kB (27.916 bytes)**. Si algú pot pensar que aquesta diferència tan gran està motivada per ser la foto original en color i l'aproximació en escala de grisos, hem passat la fotografia original a escala de grisos sense fer cap aproximació i aquesta té un pes de **50,5 kB (50.477 bytes)**, quasi el doble que el de l'aproximació bona que hem trobat.

Referències

[1] Missouri State University. Singular Value Decomposition in Digital Signal Processing

[2] Wikipedia. The singular value decomposition on the construction of the Moore–Penrose pseudoinverse

[3] Rasmus Elsborg Madsen, Lars Kai Hansen and Ole Winther. Singular Value Decomposition and Principal Component Analysis

[4] D. M. Christopher, K. Eugenia, and M. Takemasa. Estimating and correcting global weather model error. Monthly weather review, 135(2):281:299, 2007.

[5] Nishith Kumar, Mohammed Nasser, Subaran Chandra Sarker. A New Singular Value Decomposition Based Robust Graphical Clustering Technique and Its Application in Climatic Data

[6] Pàgines 111-113. Demmel, James W. Applied Numerical Linear Algebra, SIAM, 1997.

[7] Pàgines 7-10. Nota: a l'arxiu hi ha una numeració de pàgines incorrecta, en realitat son un total de 9 diapositives. Peter Blomgren. Dynamical Systems Group Computational. Department of Mathematics and Statistics. San Diego State University. Sciences Research Center Numerical Matrix Analysis. Lecture Notes 23 — Eigenvalues. Computing the Singular Value Decomposition

[8] Pàgina 241. Demmel, James W. Applied Numerical Linear Algebra, SIAM, 1997.

[9] National Institute of Standards and Technology. United States Department of Commerce. Läuchli Matrix

[10] G.Golub, W. Kahan. Calculating the singular values and pseudo-inverse of a matrix, J. SIAM. Numer. Anal. Ser. B., 2, 205-224 (1965)

[11] Pàgines 248-254. Demmel, James W. Applied Numerical Linear Algebra, SIAM, 1997.

[12] Gene Golub, Christian Reinsch .Singular value decomposition and least squares solutions. Numerische Mathematik 14 (5): 403–420. doi:10.1007/BF02163027. MR 1553974.

[13] Wikipedia. Singular Value Descomposition. History

[14] Pàgines 119-121. Demmel, James W. Applied Numerical Linear Algebra, SIAM, 1997.

[15] Wikipedia alemanya. Autor: Dundanox. Llicència Creative Commons. Enllaç

[16] Greg Fasshauer. Department of Applied Mathematics. Illinois Institute of Technology, Chicago. How to Compute the SVD

[17] Peter Blomgren. Dynamical Systems Group Computational. Department of Mathematics and Statistics. San Diego State University. Sciences Research Center Numerical Matrix Analysis. Lecture Notes 23 — Eigenvalues. Computing the Singular Value Decomposition

[18] Pàgines 227-228. Demmel, James W. Applied Numerical Linear Algebra, SIAM, 1997.

[19] Benedikt Großer and Bruno Lang. Bergische Universität GH Wuppertal. Fachbereich Mathematik. An $O(n^2)$ algorithm for the bidiagonal SVD

[20] Raymond J. Spiteri. Department of Computer Science. University of Saskatchewan, Saskatoon, Canada. Unshifted QR Algorithm

[21] Peter Blomgren. Dynamical Systems Group Computational. Department of Mathematics and Statistics. San Diego State University. Sciences Research Center Lecture Notes 21 — Eigenvalues. The QR-Algorithm

[22] Peter Blomgren. Dynamical Systems Group Computational. Department of Mathematics and Statistics. San Diego State University. Sciences Research Center Numerical Matrix Analysis. Lecture Notes 22 — Eigenvalues. The QR-Algorithm with Shifts

[23] Seyed Roholah Ghodsi i Mohammad Taeibi-Rahni. Mechanical and Aerospace Engineering Department. Islamic Azad University (IAU). Tehran, Iran. A Novel Parallel Algorithm Based on the Gram-Schmidt Method for Tridiagonal Linear Systems of Equations