Àlgebra lineal computacional

Francesc Aràndiga i Pep Mulet

Índex

Introducció

Aquest text és fruit de la tasca docent dels autors en diversos camps de l'anàlisi numèrica. El seu contingut representa, d'una banda, una ampliació dels continguts del llibre bàsic *Mètodes numèrics per a l'àlgebra lineal* [2], també escrit pels autors, i de l'altra, un tractament més avançat del tema de l'àlgebra lineal numèrica, especialment enfocat per a estadístics.

Motivem aquest text presentant en el tema 2 un problema que apareix en la resolució numèrica d'equacions en derivades parcials de tipus el·líptic (vegeu, per exemple, [15] per a més detalls). Aquests sistemes obtinguts condueixen a matrius amb una estructura que il·lustra la importància d'utilitzar mètodes iteratius de manera òptima, tal com es tracta en el tema 4.

En el tema 3 veiem, de la mateixa manera, que la matriu obtinguda en el tema 2 permet també motivar la utilització dels mètodes directes per a matrius a bandes i matrius a blocs. Aquests mètodes poden aprofitar l'arquitectura jeràrquica de la memòria dels processadors actuals per obtenir algorismes amb velocitats d'execució elevades.

En el tema 5 es veuen mètodes per a calcular valors i vectors propis més avançats que els que es tracten en el llibre *Mètodes numèrics per a l'àlgebra lineal* [2].

Seguidament, en el tema 6, s'estudia el tema de la descomposició en valors singulars (SVD) donant-ne exemples i aplicacions, com ara la compressió de dades i l'anàlisi de components principals. A partir de la relació d'aquesta descomposició amb la descomposició espectral de les matrius simètriques s'obtenen extensions dels algorismes de càlcul de vectors i valors propis al càlcul dels valors i vectors singulars.

El text finalitza amb els temes 7 i 8, que tracten les inverses generalitzades i la seua relació amb la descomposició en valors singulars i la diferenciació matricial. Qui vulga aprofundir en aquests dos últims temes, els quals són particularment interessants per a estadístics, poden consultar els llibres [18, 4, 14, 10].

Tema 1

Preliminars

1.1 Notació

Els escalars es representaran habitualment amb lletres gregues minúscules: α, β, γ, $\cdots$. Interpretarem els vectors com a vectors columna quan no s'especifique el contrari, i els representarem habitualment amb lletres romanes minúscules: x, y, a, b, $\cdots$. Les matrius les representarem amb lletres romanes majúscules: A, B, X, $\cdots$. El conjunt de les matrius $m \times n$ el representem amb mat(m, n).

L'element (i, j) d'una matriu A es representa per A_{ij} o $A(i, j)$ o a_{ij}, si hem especificat prèviament que $A = (a_{ij})$. De manera similar, l'element i d'un vector v es representa per v_i o $v(i)$.

La notació que utilitzarem per referir-nos a les files, les columnes i les submatrius d'una matriu és de tipus *matlab*: per exemple

- $A(i, :) \equiv$ fila i-èsima de A.
- $A(:, i) \equiv$ columna i-èsima de A.

1.2 Operacions amb vectors i matrius

1.2.1 Producte com a combinació lineal

La multiplicació d'una matriu per un vector s'efectua habitualment mitjançant:

$$(A * x)(i) = A(i, :) * x = \sum_{k=1}^{n} A(i, k) * x(k), \quad i = 1, \ldots, m,$$

on A és $m \times n$ i x és $n \times 1$. La següent igualtat sol resultar molt útil:

$$A * x = x(1) * A(:, 1) + x(2) * A(:, 2) + \cdots + x(n) * A(:, n),$$

és a dir,

> el producte d'una matriu per un vector es calcula com la combinació lineal de les columnes de la matriu amb les components del vector com a coeficients.

Formalment, aquesta igualtat es pot provar de la manera següent:

$$\begin{aligned}
&A * x = x(1) * A(:, 1) + x(2) * A(:, 2) + \cdots + x(n) * A(:, n) \Leftrightarrow \\
&(A * x)(i) = (x(1) * A(:, 1) + x(2) * A(:, 2) + \cdots + x(n) * A(:, n))(i), \quad i = 1, \ldots, m \Leftrightarrow \\
&(A * x)(i) = x(1) * A(i, 1) + x(2) * A(i, 2) + \cdots + x(n) * A(i, n), \quad i = 1, \ldots, m
\end{aligned}$$

La multiplicació de dues matrius A $m \times l$ i B $l \times n$ s'efectua habitualment de la manera següent:

$$(A * B)(i,j) = A(i,:) * B(:,j) = \sum_{k=1}^{l} A(i,k) * B(k,j), \quad i = 1,\dots,m, j = 1,\dots,n.$$

Les igualtats següents descriuen com calcular les files i columnes del producte de les dues matrius

$$(A * B)(:,j) = A * B(:,j) = \sum_{k=1}^{l} A(:,k) * B(k,j), j = 1,\dots,n.$$
$$(A * B)(i,:) = A(i,:) * B = \sum_{k=1}^{l} A(i,k) * B(k,:), \quad i = 1,\dots,m,$$

és a dir,

- les *columnes* del producte es calculen com a combinació lineal de les *columnes* de la matriu de *l'esquerra*, agafant com a coeficients els elements de la *columna* corresponent de la matriu de la *dreta*.
- les *files* del producte es calculen com a combinació lineal de les *files* de la matriu de la *dreta*, agafant com a coeficients els elements de la *fila* corresponent de la matriu de *l'esquerra*.

Exemple 1.1. Considerem les matrius

$$A = \begin{bmatrix} 0 & 1 & -1 \\ 2 & -2 & 0 \\ -2 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad B = \begin{bmatrix} -2 & 2 \\ 1 & 0 \\ 0 & 0 \end{bmatrix} \quad A * B = \begin{bmatrix} 1 & 0 \\ -6 & 4 \\ 3 & -4 \\ 1 & -2 \end{bmatrix} \tag{1.1}$$

Calculem la primera columna de $A * B$ com a combinació lineal de les columnes de A:

$$(-2) * \begin{bmatrix} 0 \\ 2 \\ -2 \\ -1 \end{bmatrix} + 1 * \begin{bmatrix} 1 \\ -2 \\ -1 \\ -1 \end{bmatrix} + 0 * \begin{bmatrix} -1 \\ 0 \\ -1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ -6 \\ 3 \\ 1 \end{bmatrix}.$$

Calculem la primera fila de $A * B$ com a combinació lineal de les files de B:

$$0 * \begin{bmatrix} -2 & 2 \end{bmatrix} + 1 * \begin{bmatrix} 1 & 0 \end{bmatrix} + (-1) * \begin{bmatrix} 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

1.2.2 Invertibilitat

L'operació de "divisió" de matrius sols és possible quan el "divisor" és una matriu A *invertible* o *regular*, és a dir, quan existeix una altra matriu A^{-1}, necessàriament única, tal que $A^{-1}A = A^{-1}A = I$, per la qual cosa A és obligatòriament quadrada.

Proposició 1.2. Una matriu A $n \times n$ és invertible si i només si l'única solució del sistema homogeni $Ax = 0$ és la trivial $x = 0$.

1.3 Matrius per blocs

Podem considerar una matriu com a *matriu de matrius*, efectuant una partició de les files i columnes de la matriu en diversos grups. Per il·lustrar aquest fet considerarem particions de dos grups, encara que, en general, el nombre pot ser arbitrari.

Per exemple, la matriu A anterior es pot *particionar* de la següent manera:

$$\left[\begin{array}{cc|c} 0 & 1 & -1 \\ 2 & -2 & 0 \\ -2 & -1 & -1 \\ \hline -1 & -1 & -1 \end{array}\right] = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

$$A_{11} = \begin{bmatrix} 0 & 1 \\ 2 & -2 \\ -2 & -1 \end{bmatrix} \; A_{12} = \begin{bmatrix} -1 \\ 0 \\ -1 \end{bmatrix} \; A_{21} = \begin{bmatrix} -1 & -1 \end{bmatrix} \; A_{22} = \begin{bmatrix} -1 \end{bmatrix}$$

Operacions per blocs

Considerem la matriu $m \times n$

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

on A_{ij} és una matriu $m_i \times n_j$, on $m_1 + m_2 = m$, $n_1 + n_2 = n$.

La transposició de A es pot calcular com:

$$A^T = \begin{bmatrix} A_{11}^T & A_{21}^T \\ A_{12}^T & A_{22}^T \end{bmatrix}$$

Si α és un escalar, aleshores es verifica:

$$\alpha * A = \begin{bmatrix} \alpha * A_{11} & \alpha * A_{12} \\ \alpha * A_{21} & \alpha * A_{22} \end{bmatrix}$$

Si B és una altra matriu $m \times n$ amb una partició de dimensions idèntiques a la de A

$$B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

on B_{ij} és una matriu $m_i \times n_j$, on $m_1 + m_2 = m$, $n_1 + n_2 = n$, es verifica

$$A + B = \begin{bmatrix} A_{11} + B_{11} & A_{12} + B_{12} \\ A_{21} + B_{21} & A_{22} + B_{22} \end{bmatrix}.$$

Si B és una matriu $n \times l$ amb una partició de dimensions *compatibles pel producte* per A, és a dir,

$$B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

on B_{ij} és una matriu $n_i \times l_j$, on $n_1 + n_2 =$, $l_1 + l_2 = l$, es verifica

$$A * B = \begin{bmatrix} A_{11} * B_{11} + A_{12} * B_{21} & A_{11} * B_{12} + A_{12} * B_{22} \\ A_{21} * B_{11} + A_{22} * B_{21} & A_{21} * B_{12} + A_{22} * B_{22} \end{bmatrix}.$$

En resum:

> Les operacions amb matrius per blocs s'efectuen com si els blocs foren escalars, amb la precaució que cal assegurar que les operacions resultants entre els blocs es puguen efectuar i que el producte matricial no és commutatiu.

Tema 2

Resolució numèrica d'EDP el·líptiques

2.1 Equacions el·líptiques 1D

2.1.1 Equació de la calor unidimensional

Mitjans homogenis i termes font

Si $u(x,t)$ designa la temperatura (a determinar) en el punt x d'un cert mitjà homogeni unidimensional en l'instant t, *l'equació de la calor* unidimensional és:

$$\frac{\partial u(x,t)}{\partial t} = \kappa \frac{\partial^2 u(x,t)}{\partial x^2} + f(x,t), \tag{2.1}$$

on se suposa que $x \in (a,b)$, $t \in (0,t_1)$ (t_1 potser $+\infty$) i es prescriuen:

condicions de frontera: $u(a,t) = u(b,t) = 0$, $t \in (0,t_1)$ (per simplicitat, hem considerat condicions de frontera *Dirichlet homogènies*, encara que podríem haver-ne agafat d'altres)

condicions inicials: $u(x,0) = u_0(x)$, $x \in (a,b)$, on u_0 és una funció donada que verifica també les condicions de frontera: $u_0(a) = u_0(b) = 0$.

El coeficient $\kappa > 0$ s'anomena *difusivitat termal* del mitjà (amb unitats m^2/s). La funció (coneguda) $f(x,t)$ modela una font de calor que pot variar tant en l'espai com en el temps, amb unitats K/s (és a dir, si, per simplicar, f val 2 en un determinat punt x al llarg d'un temps Δt segons, es produeix un increment de temperatura de $2\Delta t$ graus Kelvin en el punt x).

Aquesta és una equació en derivades parcials (EDP) lineal parabòlica prototípica que descriu el procés de transferència (per difusió) de temperatura de parts calentes a les més fredes d'una distribució inicial de temperatura $u_0(x)$, on se suposa que les "parets" del recinte (els punts a,b) actuen com a "refredants" instantanis i el mitjà és homogeni (el coeficient de difusivitat termal és constant).

Difusivitat variable

Quan el coeficient de difusivitat termal $\kappa(x)$ és variable (però conegut), l'equació de la calor pren la forma:

$$\frac{\partial u(x,t)}{\partial t} = \frac{\partial}{\partial x}\left(\kappa(x)\frac{\partial u(x,t)}{\partial x}\right) + f(x,t), \tag{2.2}$$

Cal adonar-se que (2.1) és un cas particular de (2.2) quan $\kappa(x)$ és constant.

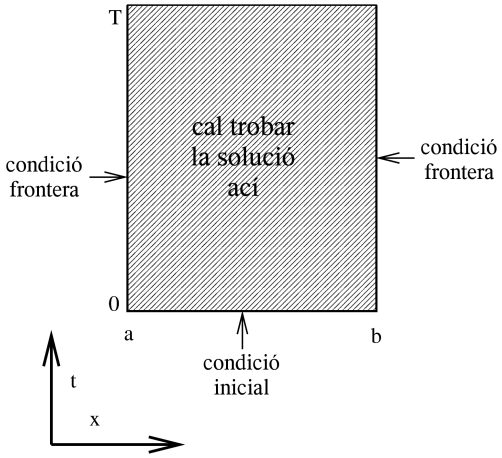


Figura 2.1: Domini per a l'equació de la calor unidimensional

2.1.2 Estat estacionari

Nota 2.1. Encara que en alguns casos les equacions que apareixeran en aquest context unidimensional siguen equacions diferencials ordinàries i/o fàcilment resolubles, això no serà així per a dimensió ≥ 2 o amb condicions de frontera diferents. Per aquest motiu, s'ha optat per no mostrar tècniques que s'apliquen sols al cas unidimensional.

L'estat estacionari de la solució $u(x,t)$ d'una de les equacions anteriors és una funció u_∞ que verifica:

$$\lim_{t\to\infty} u(x,t) = u_\infty(x).$$

Sota hipòtesis favorables de convergència (uniforme, per exemple),

$$\lim_{t\to\infty} \frac{\partial u(x,t)}{\partial t} = 0,$$

doncs u_∞ verifica l'equació resultant d'eliminar el terme $\frac{\partial}{\partial t}$. És a dir, en el cas de l'equació de la calor, el seu estat estacionari verifica

$$0 = \kappa \frac{\partial^2 u(x)}{\partial x^2} + f(x),$$

la qual és equivalent a *l'equació de Poisson*:

$$-\frac{\partial^2 u(x)}{\partial x^2} = \frac{f(x)}{\kappa}. \tag{2.3}$$

Aquesta equació és un exemple prototípic d'EDP el·líptica lineal amb coeficients constants.

En general, l'estat estacionari verifica l'equació el·líptica lineal, amb coeficients no constants

$$-\frac{\partial}{\partial x}\left(\kappa(x)\frac{\partial u(x)}{\partial x}\right) = f(x), \tag{2.4}$$

on se suposa que es verifiquen les condicions de frontera $u(a) = u(b) = 0$.

Cal adonar-se que (2.3) és un cas particular de (2.4).

Nota 2.2. Per simplificar la notació, designarem les derivades parcials amb subíndex. Així, per exemple, l'equació (2.4) quedaria:

$$-\left(\kappa(x)u_x(x)\right)_x = f(x),$$

2.1.3 Discretització per diferències finites

En casos favorables (una dimensió, coeficients constants i condicions de frontera periòdiques, per exemple) es pot trobar la solució exacta de (2.3). Però, en general, no existeixen fórmules tancades per a la solució de (2.4), per la qual cosa cal emprar mètodes numèrics per a la seua aproximació.

El primer pas en la discretització d'una EDP consisteix a "discretitzar" les possibles funcions solució, és a dir, a seleccionar una quantitat *finita* d'informació a partir de la qual es puga *aproximar* la solució buscada. En el cas de les equacions el·líptiques anteriors substituïm la funció u per valors u_i ($i = 0, \ldots, n+1$), aproximació de $u(a + ih)$, ($h = (b - a)/(n + 1)$). Els punts $x_i = a + ih$, ($i = 0, \ldots, n + 1$) formen una *malla equiespaiada* per a l'interval $[a, b]$, amb n punts al seu interior i $x_0 = a, x_{n+1} = b$. Cal adonar-se que, en aquest cas, les condicions de frontera prescriuen els valors $u_0 = u_{n+1} = 0$.

El segon pas en aquest procés de discretització és veure quines condicions (equacions) sobre els valors $u(a+ih)$ ($i = 1, \ldots, n$) es dedueixen de l'EDP en qüestió i les condicions de frontera. Aquestes equacions que "aproximen" l'EDP original se solen obtenir substituint les derivades parcials per aproximacions numèriques basades en diferències finites (d'ací el nom d'aquestes discretitzacions per *diferències finites*). Les fórmules de diferenciació numèrica que emprarem seran:

$$u'(x) = \frac{u(x + h) - u(x)}{h} + \mathcal{O}(h) \tag{2.5}$$

$$u'(x) = \frac{u(x) - u(x - h)}{h} + \mathcal{O}(h) \tag{2.6}$$

$$u''(x) = \frac{u(x + h) - 2u(x) + u(x - h)}{h^2} + \mathcal{O}(h^2) \tag{2.7}$$

Les dues primeres aproximacions es diuen de primer ordre perquè l'error és $\mathcal{O}(h)$; l'última es diu de segon ordre perquè l'error és $\mathcal{O}(h^2)$.

L'equació discretitzada s'obté a partir d'aquestes equacions "aproximades", substituint els valors $u(a + ih)$ pels u_i (a determinar) i l'aproximació per igualtat.

Exemple 2.3. Trobarem una discretització (no és única) en diferències finites de l'equació de Poisson

$$-u_{xx} = f(x), x \in (0, 1), \tag{2.8}$$

amb les condicions de frontera anteriors.

Fixada la *malla computacional* $x_i = ih$, $i = 0, \ldots, n + 1$, $h = 1/(n + 1)$, aproximem la derivada $u_{xx}(x_i)$, $x_i \in (0, 1)$, és a dir, $1 \leq i \leq n$, per la fórmula (2.7) en funció dels valors $u(x_j)$:

$$\begin{aligned} u_{xx}(x_i) &\approx \frac{u(x_i + h) - 2u(x_i) + u(x_i - h)}{h^2} \\ &= \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2}. \end{aligned}$$

La substitució d'aquesta fórmula en (2.8) (per a $x = x_i$) dóna:

$$\frac{-u(x_{i+1}) + 2u(x_i) - u(x_{i-1})}{h^2} \approx f(x_i),$$

ja que la *discretització en diferències finites* de l'equació de Poisson queda:

$$\frac{-u_{i+1} + 2u_i - u_{i-1}}{h^2} = f(x_i), i = 1, \ldots, n, \tag{2.9}$$

on $u_i \approx u(x_i)$. Cal adonar-se que en aquest sistema lineal de n equacions amb les n incògnites $u_1, \ldots, u_n$, apareixen, a més, u_0 (sols en la primera) i u_{n+1} (sols en l'última). Aquestes expressions, com a aproximacions de la solució u de (2.8) en $x_0 = 0$ i $x_{n+1} = 1$, prenen el valor 0, doncs, per simplificar la notació, cal considerar que no hi apareixen (en el cas general en què les condicions de frontera no fóren homogènies, caldria passar els valors de u_0 i u_{n+1} al segon membre).

La forma matricial de (2.9) és:

$$\frac{1}{h^2}\begin{bmatrix} 2 & -1 & 0 & & \ldots & 0 \\ -1 & 2 & -1 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & 0 & -1 & 2 & -1 \\ 0 & \ldots & & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ u_n \end{bmatrix} = \begin{bmatrix} f(x_1) \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ f(x_n) \end{bmatrix}. \tag{2.10}$$

La matriu Δ_n d'aquest sistema s'anomena *laplaciana unidimensional*. Cal adonar-se de la simetria d'aquesta matriu i de la disposició dels elements no zero en les diagonals -1,0,1.

Exemple 2.4. Trobarem ara una discretització de l'equació

$$-\left(\kappa(x)u_x(x)\right)_x = f(x), \tag{2.11}$$

amb la mateixa malla computacional i notació anteriors.

Per raons que aclarirem després, l'aproximació no la farem desenvolupant la derivada exterior (la que afecta a $\kappa(x)u_x(x)$), és a dir, no utilitzarem la fórmula anterior i similars per aproximar

$$\left(\kappa(x)u_x(x)\right)_x = \kappa_x(x)u_x(x) + \kappa(x)u_{xx}(x),$$

sinó que utilitzarem fórmules *diferents* per aproximar les derivades parcials que apareixen (utilitzarem (2.6) per a la derivada "exterior" i (2.5) per a les "interiors"):

$$\begin{aligned} &\left(\kappa(x)u_x(x)\right)_x(x_i) \approx \quad \text{(usem (2.6))} \\ &\frac{\kappa(x_i)u_x(x_i) - \kappa(x_i-h)u_x(x_i-h)}{h} \approx \quad \text{(usem (2.5))} \\ &\frac{1}{h}\left(\kappa(x_i)\frac{u(x_i+h)-u(x_i)}{h} - \kappa(x_{i-1})\frac{u(x_i)-u(x_i-h)}{h})\right) = \\ &\frac{1}{h^2}\left(\kappa(x_i)u(x_{i+1}) - (\kappa(x_i)+\kappa(x_{i-1}))u(x_i) + \kappa(x_{i-1})u(x_{i-1})\right) \end{aligned}$$

Si substituïm en (2.11), introduïm les aproximacions u_i en compte de $u(x_i)$ i l'aproximació per igualtat, aleshores obtenim:

$$\frac{1}{h^2}\left(-\kappa_i u_{i+1} + (\kappa_i + \kappa_{i-1})u_i - \kappa_{i-1}u_{i-1}\right) = f(x_i), \tag{2.12}$$

on $\kappa_i = \kappa(x_i)$.

La forma matricial de (2.12) és:

$$\frac{1}{h^2}\begin{bmatrix} \kappa_0+\kappa_1 & -\kappa_1 & 0 & & \ldots & 0 \\ -\kappa_1 & \kappa_1+\kappa_2 & -\kappa_2 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & 0 & -\kappa_{n-2} & \kappa_{n-2}+\kappa_{n-1} & -\kappa_{n-1} \\ 0 & \ldots & & 0 & -\kappa_{n-1} & \kappa_{n-1}+\kappa_n \end{bmatrix} \begin{bmatrix} u_1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ u_n \end{bmatrix} = \begin{bmatrix} f(x_1) \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ f(x_n) \end{bmatrix}. \tag{2.13}$$

Cal adonar-se de la simetria d'aquesta matriu (aquest és el motiu de la discretització triada) i de la disposició dels elements no zero en les diagonals -1,0,1 i del fet que aquesta matriu es redueix a la de (2.10) quan $\kappa(x) = 1$. En general, i a diferència de (2.10), la matriu del sistema no té les diagonals constants.

Proposició 2.5. La matriu de (2.13), amb $\kappa_i > 0$, és una matriu definida positiva.

Demostració. Designem

$$B = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & -1 & 1 \\ 0 & \dots & 0 & 0 & -1 \end{bmatrix} \quad K = \begin{bmatrix} \kappa_1 & 0 & 0 & \dots & 0 \\ 0 & \kappa_2 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & \kappa_{n-1} & 0 \\ 0 & \dots & 0 & 0 & \kappa_n \end{bmatrix} \quad E = \begin{bmatrix} \kappa_0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & 0 & 0 \\ 0 & \dots & 0 & 0 & 0 \end{bmatrix}.$$

Es comprova que la matriu A de (2.13) admet la descomposició següent:

$$A = B^T K B + E.$$

Siga $x \in \mathbb{R}^n$ i calculem

$$x^T A x = x^T (B^T K B + E) x = x^T B^T K B x + x^T E x = (Bx)^T K (Bx) + \kappa_0 x_1^2.$$

Si designem $y = Bx$, aquesta darrera equació queda:

$$x^T A x = y^T K y + \kappa_0 x_1^2 = \sum_{i=1}^{n} \kappa_i y_i^2 + \kappa_0 x_1^2.$$

Com que $\kappa_i > 0$, es dedueix aleshores que

1. $x^T A x \geq 0$, $\forall x$.
2. $x^T A x = 0$ si i sols si $y_i = 0$, $i = 1, \dots, n$ i $x_1 = 0$ si i sols si $x_1 = \cdots = x_n$ i $x_1 = 0$ si i sols si $x_1 = \cdots = x_n = 0$

la qual cosa clou la demostració. □

Corol·lari 2.6. La matriu de (2.10) és una matriu definida positiva.

2.2 Equacions el·líptiques 2D

2.2.1 Equació de la calor bidimensional

Mitjans homogenis i termes font

Si $u(x, y, t)$ denota la temperatura (a determinar) en el punt (x, y) d'un cert mitjà homogeni bidimensional en l'instant t, *l'equació de la calor* bidimensional és:

$$\frac{\partial u(x,y,t)}{\partial t} = \kappa \frac{\partial^2 u(x,y,t)}{\partial x^2} + \kappa \frac{\partial^2 u(x,y,t)}{\partial y^2} + f(x,y,t), \tag{2.14}$$

on se suposa que $(x, y) \in \Omega = (a, b) \times (c, d)$, $t \in (0, t_1)$ (t_1 potser $+\infty$) i es prescriuen:

condicions de frontera: $u(x,y,t) = 0$, $(x,y,t) \in \partial\Omega \times (0,t_1)$.

condicions inicials: $u(x,y,0) = u_0(x,y)$, $(x,y) \in \Omega$, on u_0 és una funció donada que verifica també les condicions de frontera: $u_0(x,y) = 0$, $(x,y) \in \partial\Omega$.

La funció (coneguda) $f(x,y,t)$ modela una font de calor que pot variar tant en l'espai com en el temps. Amb la notació habitual per a les derivades parcials i amb $\Delta u = u_{xx} + u_{yy}$, aquesta equació queda:

$$u_t(x,y,t) = \kappa \Delta u(x,y,t) + f(x,y,t), \tag{2.15}$$

Difusivitat variable

Quan el coeficient de difusivitat termal és variable, l'equació de la calor pren la forma:

$$\frac{\partial u(x,y,t)}{\partial t} = \frac{\partial}{\partial x}\left(\kappa(x,y)\frac{\partial u(x,y,t)}{\partial x}\right) + \frac{\partial}{\partial y}\left(\kappa(x,y)\frac{\partial u(x,y,t)}{\partial y}\right) + f(x,y,t). \tag{2.16}$$

Cal adonar-se que (2.14) és un cas particular de (2.16) quan $\kappa(x,y)$ és constant.

2.2.2 Estat estacionari

De la mateixa manera que s'ha arribat a les equacions el·líptiques per als estats estacionaris de les diverses equacions de la calor unidimensionals, en el cas bidimensional obtindríem les equacions següents:

$$\begin{aligned} 0 &= \kappa\frac{\partial}{\partial x}\left(\frac{\partial u(x,y)}{\partial x}\right) + \kappa\frac{\partial}{\partial y}\left(\frac{\partial u(x,y)}{\partial y}\right) + f(x,y) \\ 0 &= \frac{\partial}{\partial x}\left(\kappa(x,y)\frac{\partial u(x,y)}{\partial x}\right) + \frac{\partial}{\partial y}\left(\kappa(x,y)\frac{\partial u(x,y)}{\partial y}\right) + f(x,y), \end{aligned}$$

les quals donen:

$$-\frac{\partial}{\partial x}\left(\frac{\partial u(x,y)}{\partial x}\right) - \frac{\partial}{\partial y}\left(\frac{\partial u(x,y)}{\partial y}\right) = \frac{f(x,y)}{\kappa} \tag{2.17}$$

$$-\frac{\partial}{\partial x}\left(\kappa(x,y)\frac{\partial u(x,y)}{\partial x}\right) - \frac{\partial}{\partial y}\left(\kappa(x,y)\frac{\partial u(x,y)}{\partial y}\right) = f(x,y). \tag{2.18}$$

Aquestes equacions són EDP lineals el·líptiques, la primera amb coeficients constants (*equació de Poisson bidimensional*) i la segona variables.

2.2.3 Discretització per diferències finites

Equació de Poisson bidimensional

El primer pas en la discretització per diferències finites de l'equació (2.17) consisteix a establir una *malla computacional* sobre el domini Ω.

Agafem sobre $[a,b]$ una malla equiespaiada amb m punts al seu interior ($x_i = a + ih$, $h = (b-a)/(m+1)$, $i = 0, \cdots m+1$) i sobre $[c,d]$ una malla equiespaiada amb n punts al seu interior ($y_j = c + jk$, $k = (d-c)/(n+1)$, $j = 0, \cdots n+1$). La malla triada en $\overline{\Omega} = [a,b] \times [c,d]$ és producte cartesià de les malles triades en $[a,b]$ i $[c,d]$: (x_i, y_j), $i = 0, \ldots, m+1$, $j = 0, \ldots, n+1$, dels quals aquells amb índex $1 \leq i \leq m$, $i \leq j \leq n$ estan en l'interior (un total de mn) i la resta en la frontera $\partial\Omega$.

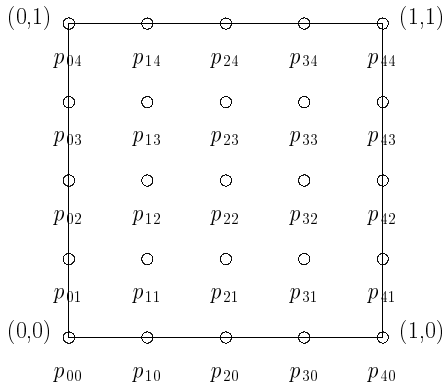


Figura 2.2: Malla sobre domini rectangular

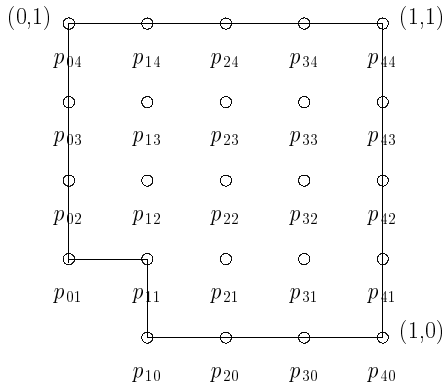


Figura 2.3: Malla sobre domini no rectangular

El segon pas en la discretització consisteix a aproximar les derivades parcials que apareixen en (2.17) per fórmules de diferenciació numèrica. Per la definició mateixa de les derivades parcials, tenim que $\frac{\partial}{\partial x}\left(\frac{\partial u(x_i,y_j)}{\partial x}\right)$ és la derivada segona de la funció $x \mapsto u(x, y_j)$, ja que es pot aproximar per:

$$\begin{aligned}\frac{\partial}{\partial x}\left(\frac{\partial u(x_i,y_j)}{\partial x}\right) &\approx \frac{u(x_i+h,y_j)-2u(x_i,y_j)+u(x_i-h,y_j)}{h^2}\\ &= \frac{u(x_{i+1},y_j)-2u(x_i,y_j)+u(x_{i-1},y_j)}{h^2}.\end{aligned} \tag{2.19}$$

De la mateixa manera, tenint en compte que $\frac{\partial}{\partial y}\left(\frac{\partial u(x_i,y_j)}{\partial y}\right)$ és la derivada segona de la funció $y \mapsto u(x_i, y)$,

$$\begin{aligned}\frac{\partial}{\partial y}\left(\frac{\partial u(x_i,y_j)}{\partial y}\right) &\approx \frac{u(x_i,y_j+h)-2u(x_i,y_j)+u(x_i,y_j-h)}{h^2}\\ &= \frac{u(x_i,y_{j+1})-2u(x_i,y_j)+u(x_i,y_{j-1})}{h^2}.\end{aligned} \tag{2.20}$$

Per tant, l'equació aproximada a (2.17) obtinguda a partir d'aquestes fórmules és:

$$\begin{aligned}
&-\frac{1}{h^2}\big(u(x_{i+1},y_j)-2u(x_i,y_j)+u(x_{i-1},y_j)\\
&+u(x_i,y_{j+1})-2u(x_i,y_j)+u(x_i,y_{j-1})\big)\approx\frac{f(x_i,y_j)}{\kappa}\\
&\Rightarrow\frac{1}{h^2}\big(-u_{i+1,j}-u_{i-1,j}-u_{i,j+1}-u_{i,j-1}+4u_{i,j}\big)=\frac{f_{i,j}}{\kappa},
\end{aligned}\tag{2.21}$$

$i=1,\cdots m, j=1,\cdots,n.$

Difusivitat variable

De la mateixa manera que hem demostrat en el cas unidimensional que

$$\left(\kappa(x)u_x(x)\right)_x(x_i)\approx\frac{1}{h^2}\left(\kappa(x_i)u(x_{i+1})-(\kappa(x_i)+\kappa(x_{i-1}))u(x_i)+\kappa(x_{i-1})u(x_{i-1})\right)$$

i per la definició de les derivades parcials, tenim que:

$$\begin{aligned}
\frac{\partial}{\partial x}\left(\kappa(x,y)\frac{\partial u}{\partial x}\right)(x_i,y_j)\approx&\frac{1}{h^2}(\kappa(x_i,y_j)u(x_{i+1},y_j)\\
&-(\kappa(x_i,y_j)+\kappa(x_{i-1},y_j))u(x_i,y_j)\\
&+\kappa(x_{i-1},y_j)u(x_{i-1},y_j))\\
\frac{\partial}{\partial y}\left(\kappa(x,y)\frac{\partial u}{\partial y}\right)(x_i,y_j)\approx&\frac{1}{h^2}(\kappa(x_i,y_j)u(x_i,y_{j+1})\\
&-(\kappa(x_i,y_j)+\kappa(x_i,y_{j-1}))u(x_i,y_j)\\
&+\kappa(x_i,y_{j-1})u(x_i,y_{j-1})),
\end{aligned}\tag{2.22}$$

per la qual cosa la discretització de l'equació (2.18) queda:

$$\begin{aligned}
&\frac{1}{h^2}(-\kappa_{i,j}u_{i+1,_j}-\kappa_{i,j}u_{i,j+1}\\
&+(2\kappa_{i,j}+\kappa_{i-1,j}+\kappa_{i,j-1})u_{i,j}\\
&-\kappa_{i-1,j}u_{i-1,j}-\kappa_{i,j-1}u_{i,j-1})=f_{i,j},
\end{aligned}\tag{2.23}$$

on $\kappa_{i,j}=\kappa(x_i,y_j)$ i $u_{i,j}\approx u(x_i,y_j)$ i $f_{i,j}=f(x_i,y_j)$, $i=1,\cdots m, j=1,\cdots,n.$

2.2.4 Expressió matricial

Els valors $u_{i,j}$ ($i=1,\cdots m, j=1,\cdots,n$) que apareixen en la discretització bidimensional formen una matriu U $m\times n$. Cal adonar-se que aquesta disposició fa correspondre les *línies horitzontals* de la malla (j fix) a les columnes i les *línies verticals* de la malla (i fix) a les files.

En àlgebra lineal, per trobar l'expressió matricial d'un operador cal treballar amb vectors amb índex naturals (suposarem que l'inici dels índexs és 0, per simplificar algunes expressions posteriors). Així doncs, per exemple, per expressar matricialment les accions dels operadors (2.19) i (2.20), caldria veure quina és l'expressió matricial de l'operador quan identifiquem la matriu U i la matriu resultant de l'operador amb vectors.

Una manera d'identificar matrius amb vectors és l'isomorfisme (bijecció lineal)

$$J\colon \text{mat}(m,n) \to \mathbb{R}^{mn} \tag{2.24}$$

tal que $J(U)(i) = U(i\%m, i/m)$, $0 \le i < mn$, on $i\%m$ és la resta de la divisió entera i/m o, alternativament,

$$J(U)(mj+i) = U(i,j), 0 \le j < n, 0 \le i < m. \tag{2.25}$$

Intuïtivament, J forma un vector per juxtaposició de les columnes de la matriu.

Trobar una expressió matricial-vectorial d'una aplicació lineal

$$M\colon \text{mat}(m,n) \to \text{mat}(m,n),$$

consisteix, doncs, a calcular l'expressió matricial-vectorial de l'aplicació lineal

$$\tilde{M} = J \circ M \circ J^{-1}\colon \mathbb{R}^{mn} \to \mathbb{R}^{mn}. \tag{2.26}$$

Producte de Kronecker

El producte de Kronecker d'una matriu $A = (a_{i,j})$ $n \times n$ amb una matriu $B = (b_{i,j})$ $m \times m$ [1] és una matriu $A \otimes B$ $mn \times mn$ donada per blocs per

$$A \otimes B = \begin{bmatrix} a_{0,0}B & \dots & a_{0,n-1}B \\ \dots & \dots & \dots \\ a_{n-1,0}B & \dots & a_{n-1,n-1}B \end{bmatrix}$$

o, també,

$$(A \otimes B)(mj+i, mp+q) = A_{jp}B_{iq} \quad 0 \le j,p < n, 0 \le i,q < m, \tag{2.27}$$

on usem (i usarem més endavant) el fet següent:

$$\{k\colon 0 \le k < mn\} = \{m \cdot k_1 + k_2\colon 0 \le k_1 < n, 0 \le k_2 < m\}.$$

L'actuació de $A \otimes B$ sobre un vector $J(U)$, en la component $mj+i$, $0 \le j < n$, $0 \le i < m$, és:

$$\begin{aligned}((A \otimes B)J(U))(mj+i) &= \sum_{p=1}^{n}\sum_{q=1}^{m}(A \otimes B)(mj+i, mp+q)J(U)(mp+q) \\ &= \sum_{p=1}^{n}\sum_{q=1}^{m} A_{jp}B_{iq}U_{qp} \\ &= \sum_{p=1}^{n}\big(\sum_{q=1}^{m} B_{iq}U_{qp}\big)(A^T)_{pj} \\ &= \sum_{p=1}^{n}\big((BU)_{ip}\big)\big)(A^T)_{pj} \\ &= (BUA^T)_{ij} = J(BUA^T)(mj+i),\end{aligned}$$

així doncs obtenim

$$(A \otimes B)J(U) = J(BUA^T) \tag{2.28}$$

[1] El producte de Kronecker es pot donar per a matrius no quadrades, però s'ha optat per mantenir les matrius quadrades per no introduir-hi més paràmetres.

L'operador $\Phi : U \mapsto BUA^T = (BU)A^T$ és composició de

$$U \mapsto BU, \quad V \mapsto VA^T,$$

és a dir, podem veure'l com l'actuació primer de B i després de A^T. Vegem quina és la interpretació de l'actuació d'aquests darrers operadors, en termes de l'actuació de $x \mapsto Ax$ i $y \mapsto By$:

- Com que $(BU)(:, j) = B(U(:, j))$, el primer operador actua "transformant columnes" (és a dir, les columnes de BU són el resultat de l'actuació de B sobre cadascuna d'elles).
- Com que $VA^T = (AV^T)^T$, les files de VA^T es calculen:

  $$(VA^T)(i,:) = (AV^T)^T(i,:) = (AV^T(:,i))^T = (A(V(i,:)^T)^T,$$

  és a dir, les files de VA^T s'obtenen transformant les files de V per producte per A (amb pas previ a columna).

D'aquests raonaments deduïm que l'operador Φ ("fer primer transformacions a les columnes segons B seguit de fer transformacions a les files segons A") és $U \mapsto BUA^T$ i la seua matriu, segons (2.28) i (2.26), és $A \otimes B$, ja que, com veiem tot seguit, la seua expressió matricial-vectorial és $\tilde{\Phi}(x) = (A \otimes B)x$, $x \in \mathbb{R}^{mn}$, on $\tilde{\Phi} = J \circ \Phi \circ J^{-1} \colon \mathbb{R}^{mn} \to \mathbb{R}^{mn}$:

$$\tilde{\Phi}(x) = J(\Phi(\underbrace{J^{-1}(x)}_{U})) = J(\Phi(U)) = J(BUA^T) = (A \otimes B)J(U) = (A \otimes B)x.$$

Proposició 2.7 (propietats del producte de Kronecker)**.**

1. $(\alpha A + \beta B) \otimes C = \alpha(A \otimes C) + \beta(B \otimes C)$
2. $(A \otimes B) \otimes C = A \otimes (B \otimes C)$.
3. $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$
4. Si A i B són diagonals $A \otimes B$ és diagonal, amb els $(A \otimes B)(mi + j, mi + j) = A(i,i)B(j,j)$.
5. $I_m \otimes I_n = I_{mn}$
6. $(A \otimes B)^{-1} = (A^{-1}) \otimes (B^{-1})$.

Demostració. Per demostrar 2, siga A $m \times m$, B $n \times n$ i C $p \times p$. Designem $X = A \otimes B$ $m \cdot n \times m \cdot n$ i $Y = B \otimes C$ $n \cdot p \times n \cdot p$, cal demostrar $X \otimes C = A \otimes Y$, per a la qual cosa, com que totes dues matrius tenen les mateixes dimensions ($m \cdot n \cdot p$) cal agafar índex $0 \leq i, j < m \cdot n \cdot p$ i demostrar $(X \otimes C)(i,j) = (A \otimes Y)(i,j)$.

Per calcular el primer membre, escrivim $i = p \cdot i_1 + i_2$, $j = p \cdot j_1 + j_2$, $0 \leq i_1, j_1 < m \cdot n$, $0 \leq i_2, j_2 < p$. La relació (2.27) dóna en aquest cas:

$$(X \otimes C)(p \cdot i_1 + i_2, p \cdot j_1 + j_2) = X(i_1, j_1)C(i_2, j_2)$$

Si ara $i_1 = n \cdot i_4 + i_3$, $j_1 = n \cdot j_4 + j_3$, $0 \leq i_4, j_4 < m$, $0 \leq i_3, j_3 < n$, aleshores

$$X(i_1, j_1) = (A \otimes B)(n \cdot i_4 + i_3, n \cdot j_4 + j_3) = A(i_4, j_4)B(i_3, j_3),$$

per la qual cosa

$$(X \otimes C)(i,j) = A(i_4, j_4)B(i_3, j_3)C(i_2, j_2), \tag{2.29}$$

on la relació entre els diversos índexs és:

$$\begin{aligned} i &= p\cdot i_1 + i_2 = p(n\cdot i_4 + i_3) + i_2 = n\cdot p\cdot i_4 + p\cdot i_3 + i_2 \\ j &= p\cdot j_1 + j_2 = p(n\cdot j_4 + j_3) + j_2 = n\cdot p\cdot j_4 + p\cdot j_3 + j_2 \\ & 0\le i_2, j_2 < p, \quad 0\le i_3, j_3 < n, \quad 0\le i_4, j_4 < m. \end{aligned} \tag{2.30}$$

Per calcular el segon membre, escrivim $i = n\cdot p\cdot i'_1 + i'_2$, $j = n\cdot p\cdot j'_1 + j'_2$, $0\le i'_1, j'_1 < m$, $0\le i'_2, j'_2 < n\cdot p$. La relació (2.27) dóna en aquest cas:

$$(A\otimes Y)(n\cdot p\cdot i'_1 + i'_2, n\cdot p\cdot j'_1 + j'_2) = A(i'_1, j'_1)Y(i'_2, j'_2)$$

Si ara $i'_2 = p\cdot i'_4 + i'_3$, $j'_2 = p\cdot j'_4 + j'_3$, $0\le i'_4, j'_4 < n$, $0\le i'_3, j'_3 < p$, aleshores

$$Y(i'_2, j'_2) = (B\otimes C)(p\cdot i'_4 + i'_3, p\cdot j'_4 + j'_3) = B(i'_4, j'_4)C(i'_3, j'_3)$$

per la qual cosa

$$(A\otimes Y)(i,j) = A(i'_1, j'_1)B(i'_4, j'_4)C(i'_3, j'_3) \tag{2.31}$$

on la relació entre els diversos índexs és:

$$\begin{aligned} i &= n\cdot p\cdot i'_1 + i'_2 = n\cdot p\cdot i'_1 + p\cdot i'_4 + i'_3 \\ j &= n\cdot p\cdot j'_1 + j'_2 = n\cdot p\cdot j'_1 + p\cdot j'_4 + j'_3 \\ & 0\le i'_1, j'_1 < m, \quad 0\le i'_4, j'_4 < n, \quad 0\le i'_3, j'_3 < p \end{aligned} \tag{2.32}$$

Comparant (2.30) i (2.32) deduïm:

$$*'_1 = *_4 \quad *'_4 = *_3 \quad *'_3 = *_2,$$

on $*$ representa i o j; d'ací, (2.29) i (2.31) es dedueix la igualtat buscada.

Per demostrar 3, siguen A, C matrius $n\times n$, B, D matrius $m\times m$. Com que les matrius en ambdós membres són de la mateixa dimensió $m\cdot n$, cal demostrar la igualtat entre els seus elements corresponents.

$$\begin{aligned} &\big((A\otimes B)(C\otimes D)\big)(m\cdot i + j, m\cdot p + q) \\ &= \sum_{k=1}^{n}\sum_{l=1}^{m}(A\otimes B)(m\cdot i + j, m\cdot k + l)(C\otimes D)\big)(m\cdot k + l, m\cdot p + q) \\ &= \sum_{k=1}^{n}\sum_{l=1}^{m} A(i,k)B(j,l)C(k,p)D(l,q) \\ &= \sum_{k=1}^{n}\big(A(i,k)C(k,p)(\sum_{l=1}^{m} B(j,l)D(l,q))\big) \\ &= \big(\sum_{k=1}^{n} A(i,k)C(k,p)\big)\big(\sum_{l=1}^{m} B(j,l)D(l,q)\big) \\ &= (AC)(i,p)\cdot(BD)(j,q) \\ &= ((AC)\otimes(BD))(m\cdot i + j, m\cdot p + q) \end{aligned}$$

La resta de propietats es proposen com a exercici.

□

Forma matricial

Amb la notació de la secció (2.2.3), si $U = (u_{i,j})$ és una matriu $m \times n$ la discretització de u_{xx} aplicada a aquesta matriu és:

$$\frac{1}{h^2}(u_{i-1,j} - 2u_{i,j} + u_{i+1,j} \quad i = 1, \dots m, j = 1, \dots, n.$$

Per agilitar la notació, cal interpretar aquesta expressió com que els termes $u_{p,q}$ que corresponen a punts $(x_p, y_q) \in \partial\Omega$ **no apareixen**. Alternativament, caldria escriure:

$$\begin{aligned}
&\frac{1}{h^2}(-2u_{1,j} + u_{2,j}) \quad j = 1, \dots, n \\
&\frac{1}{h^2}(u_{i-1,j} - 2u_{i,j} + u_{i+1,j}) \quad i = 2, \dots m-1, j = 1, \dots, n \\
&\frac{1}{h^2}(u_{m-1,j} - 2u_{m,j}) \quad j = 1, \dots, n.
\end{aligned} \tag{2.33}$$

En qualsevol cas, si designem Δ_m l'operador sobre vectors de $\mathbb{R}^m$ tal que $w = \Delta_m v$ verifica:

$$\begin{aligned}
w_1 &= \frac{1}{h^2}(-2v_1 + v_2) \\
w_j &= \frac{1}{h^2}(v_{i-1} - 2v_i + v_{i+1}) \quad i = 2, \dots m-1 \\
w_n &= \frac{1}{h^2}(v_{m-1} - 2v_m)
\end{aligned}$$

aleshores (2.33) consisteix a fer actuar Δ_m sobre les columnes de U. Com que en (2.33) l'actuació per files és trivial (no hi ha modificació posterior de les files), si I_n denota la identitat $n \times n$ aleshores, pel que hem vist en la secció anterior, la forma matricial de (2.33) és $I_n \otimes \Delta_m$.

De la mateixa manera si la discretització de u_{yy} aplicada a U és

$$\begin{aligned}
&\frac{1}{h^2}(-2u_{i,1} + u_{i,2}) \quad i = 1, \dots, m \\
&\frac{1}{h^2}(u_{i,j-1} - 2u_{i,j} + u_{i,j+1}) \quad i = 1, \dots, m, j = 2, \dots n-1 \\
&\frac{1}{h^2}(u_{i,n-1} - 2u_{i,n}) \quad i = 1, \dots, m.
\end{aligned} \tag{2.34}$$

i designem Δ_n l'operador sobre vectors de $\mathbb{R}^n$ tal que $w = \Delta_n v$ verifica:

$$\begin{aligned}
w_1 &= \frac{1}{h^2}(-2v_1 + v_2) \\
w_j &= \frac{1}{h^2}(v_{j-1} - 2v_j + v_{j+1}) \quad j = 2, \dots n-1 \\
w_n &= \frac{1}{h^2}(v_{n-1} - 2v_n)
\end{aligned}$$

i I_m la matriu identitat $m \times m$, aleshores la forma matricial de (2.34) és $\Delta_n \otimes I_m$.

Deduïm dels darrers arguments que la matriu del sistema (2.21) és $-\Delta_{m,n}$, on

$$\Delta_{m,n} = I_n \otimes \Delta_m + \Delta_n \otimes I_m, \tag{2.35}$$

amb Δ_n i Δ_m matrius *laplacianes unidimensionals*.

Exemple 2.8. Els comandaments de matlab:

```
>> A=kron(eye(3), spdiags(ones(4,1)*[-1 2 -1], [-1 0 1], 4, 4));
>> B=kron(spdiags(ones(3,1)*[-1 2 -1], [-1 0 1], 3, 3), speye(4));
>> C=A+B;
```

donen les matrius $A = I_3 \otimes \Delta_4$, $B = \Delta_3 \otimes I_4$ i $C = \Delta_{3,4}$:

Proposició 2.9. La matriu $-\Delta_{m,n}$ és una matriu simètrica i definida positiva.

Demostració. Per (2.35) i les propietats del producte de Kronecker: $-\Delta_{m,n} = I_n \otimes (-\Delta_m) + (-\Delta_n) \otimes I_m$. Com que la suma de matrius definides positives ho és també i $-\Delta_m, -\Delta_n$ són matrius definides positives pel corol·lari (2.6), per concloure, sols cal demostrar el resultat següent. □

Proposició 2.10. Si A i B són matrius simètriques i definides positives, $A \otimes B$ també ho és.

Demostració. Utilitzarem el fet que una matriu A és simètrica i definida positiva si i només si existeixen matrius P ortogonal i D diagonal, $D_{i,i} > 0$, tal que $A = PDP^T$ (valors propis positius, vectors propis ortonormals).

Siguen aleshores P, Q ortogonals i D, E diagonals "positives" (de les dimensions adequades) tals que $A = PDP^T$ i $B = QEQ^T$. Per les propietats del producte de Kronecker:

1. $P \otimes Q$ és ortogonal:

$$(P \otimes Q)^T = P^T \otimes Q^T = (P^{-1}) \otimes (Q^{-1}) = (P \otimes Q)^{-1}$$

2. $D \otimes E$ és diagonal.

3. $A \otimes B$ té valors propis positius i vectors propis ortonormals:

$$\begin{aligned}&(P \otimes Q)(D \otimes E)(P \otimes Q)^T = (P \otimes Q)(D \otimes E)(P^T \otimes Q^T)\\ &= (PDP^T) \otimes (QEQ^T) = A \otimes B.\end{aligned}$$

Deduïm que $A \otimes B$ és simètrica i definida positiva. □

Tema 3

Mètodes directes per a sistemes lineals compatibles determinats

3.1 Introducció

Estudiem en aquest tema com resoldre els sistemes $Ax = b$ amb solució única, que és equivalent del fet que A siga una matriu invertible.

3.1.1 Eliminació de Gauss

L'algorisme en el qual es basen la majoria de mètodes directes per a la solució de sistemes lineals (mètodes que amb operacions exactes donen la solució del sistema en un nombre finit de passos) és el de l'eliminació de Gauss, el qual utilitza transformacions elementals (essencialment sumar a una equació un múltiple d'un altra) per convertir el problema inicial en un altre equivalent amb matriu triangular superior, senzill de resoldre utilitzant substitució cap endarrere. Resulta útil interpretar l'algorisme de l'eliminació de Gauss com el càlcul d'una descomposició de la matriu de coeficients A com a producte $A = LU$ de la matriu triangular superior resultant, U, i d'una matriu triangular triangular inferior, formada col·locant els *multiplicadors* (emprats en les transformacions elementals) sota una diagonal formada per uns. Amb aquesta interpretació, el procediment per resoldre un sistema $Ax = b$ consisteix a:

1. Calcular la descomposició $A = LU$ amb l'algorisme 1 (a). Després d'aplicar aquest algorisme la matriu d'entrada A queda modificada de manera que U és a la part triangular superior i (la part no trivial de) L a la part estrictament triangular inferior.

2. Resoldre $Ly = b$ amb l'algorisme de substitució cap endavant 1 (b). Cal notar que aquest algorisme sols accedeix a la part de la matriu sota la diagonal i assumeix implícitament que aquesta conté uns.

3. Resoldre $Ux = y$ amb l'algorisme de substitució cap endarrere 1 (c). Cal notar que aquest algorisme sols accedeix a la part de la matriu sobre la diagonal (incloent-la).

Input: A matriu $n \times n$
for $k = 1, \ldots, n-1$
  for $i = k+1, \ldots, n$
    $A(i,k) = A(i,k)/A(k,k)$
    for $j = k+1, \ldots, n$
      $A(i,j) = A(i,j) - A(i,k)A(k,j)$
    end
  end
end
Output: part estrictament triangular inferior de L sota la diagonal, part triangular superior de U sobre la diagonal, incloent-la.

(a)

Input: $A\ n \times n$, $b\ n \times 1$
for $i = n, \ldots, 1$
  for $j = i+1, \ldots, n$
    $b(i) = b(i) - A(i,j)b(j)$
  end
  $b(i) = b(i)/A(i,i)$
end
Output: b

(b)

Input: $A\ n \times n$, $b\ n \times 1$
for $i = 1, \ldots, n$
  for $j = 1, \ldots, i-1$
    $b(i) = b(i) - A(i,j)b(j)$
  end
end
Output: b

(c)

Algorisme 1: (a) Algorisme de l'eliminació de Gauss: calcula els factors L triangular inferior amb uns a la diagonal i U triangular superior tal que $A = LU$. (b) Algorisme de substitució cap enrere. Després de l'execució d'aquest algorisme, la solució queda en la variable b. (c) Algorisme de substitució per a sistemes amb matrius amb uns en la diagonal. Després de l'execució d'aquest algorisme, la solució queda en la variable b.

El cost computacional del pas 1 és de l'ordre de $\frac{2}{3}n^3$ (n la dimensió de la matriu) i els dels passos 2 i 3 de n^2 cadascun. Se'n dedueix que un algorisme eficient per a resoldre diversos sistemes amb la mateixa matriu de coeficients consisteix a aplicar el pas 1 inicialment i després els passos 2 i 3 per a cada vector de termes independents.

L'objectiu d'aquest tema és refinar aquest algorisme per obtenir millores en el temps de computació i, quan siga possible, en l'ocupació de memòria. Per aconseguir reduir el temps de computació utilitzem dos principis: primer, fer que les operacions aritmètiques s'executen a la major velocitat possible i, segon, evitar fer operacions innecessàries (sumes i productes amb zeros).

Distingirem dos tipus de matrius, segons la possibilitat d'aplicació del segon principi:

- Matrius denses, les quals tenen una petita proporció d'entrades nul·les.
- Matrius disperses, les quals tenen una petita proporció d'entrades *no* nul·les.

Les matrius denses no permeten reduir el nombre d'operacions, però els algorismes que presentem permeten una execució més ràpida, en aprofitar l'existència de memòries cau molt més ràpides que les memòries principals (*RAM*), encara que molt més petites.

En canvi, per a una matriu dispersa la major part de les operacions de l'algorisme 1 1 involucren sumes i productes amb zeros, les quals es poden evitar. Per exemple, considerem el cas d'una matriu tridiagonal A. Els factors L, U tenen la mateixa estructura que A, és a dir, son bidiagonals i es poden obtenir en aproximadament $2n$ operacions amb l'algorisme 2. Aquest algorisme mostra una tècnica que permet estalviar memòria i, per tant, resoldre problemes més grans: sols emmagatzemem les tres diagonals no nul·les de la matriu en tres vectors $n \times 1$, l, d, u. L'ús d'una *estructura de dades* (disposició concreta de les dades en memòria i procediments per accedir a aquestes dades) com aquesta, diferent d'una estructura de matriu quadrada, és típica en algorismes eficients per a resoldre sistemes amb matrius disperses. Però per a matrius disperses generals, no tot serà tan fàcil com per a les tridiagonals, ja que, a diferència d'aquestes, els factors L i U poden tenir més entrades no nul·les que la matriu A inicial (fenomen de *fill-in*).

for $i = 1, \ldots, n$
  $l(i) = l(i)/d(i)$
  $d(i+1) = d(i+1) - l(i) * u(i+1)$
end

$$A = \begin{bmatrix} d(1) & u(2) & & & & \\ l(1) & d(2) & u(3) & & & \\ & \ddots & \ddots & \ddots & & \\ & & & & & u(n) \\ & & & & l(n-1) & d(n) \end{bmatrix}$$

Algorisme 2: Algorisme per calcular la descomposició LU d'una matriu tridiagonal A.

3.1.2 Pivotatge

La utilització de *pivots* ($A(k,k)$ en l'algorisme 1 1) petits introdueix entrades grans en els factors resultants (o ∞ en cas de ser zero), la qual cosa desestabilitza l'algorisme, i li fa perdre precisió en la solució obtinguda. La tècnica de *pivotatge* intenta evitar aquest fenomen, permutant de manera adequada les files i, si cal, les columnes per aconseguir un pivot suficientment gran. L'estratègia més àmpliament usada és la de *pivotatge parcial*, que consisteix a triar com a pivot en el pas k de l'eliminació de Gauss l'element de major valor absolut de $A(k:n,k)$, la part de la columna k sota la diagonal, incloent-hi aquesta. D'aquesta manera ens assegurem que els multiplicadors (els elements de L) queden afitats per part de 1. Encara que es poden trobar matrius per a les quals aquesta estratègia no es suficient per garantir que els elements de U no cresquen, aquests exemples són rars i no es troben en les aplicacions usuals.

3.1.3 Descomposició de Choleski

Quan la matriu A és simètrica i definida positiva l'algorisme de la *descomposició de Choleski* permet calcular una descomposició $R^T R = A$ (R triangular superior) en aproximadament $\frac{1}{3}n^3$ operacions (la meitat que l'eliminació de Gauss), aprofitant la simetria de la matriu. Una particularitat molt apreciable d'aquest algorisme és que no cal utilitzar pivotatge, ja que els elements de la columna i de R no poden superar $\sqrt{A(i,i)}$ (exercici).

3.2 Matrius denses

3.2.1 Estructura de dades

Una estructura de dades per a un problema consisteix en la disposició d'aquestes d'una determinada manera en la memòria de l'ordinador, i també en els algorismes emprats per accedir a aquestes dades, donada aquesta disposició concreta.

Exemple 3.1. La manera habitual d'emmagatzemar una matriu A 10×20 i accedir a l'element $A(1,2)$ en C és utilitzar el codi següent (els índexs comencen per 0):

```
float A[10][30]; A[1][2]=1;
```

El que estem fent amb la primera instrucció (realment, el compilador ho fa per nosaltres) és reservar un espai de $10 \times 30 \times 4 = 1200$ bytes contigus en memòria (un `float` ocupa 4 bytes) per als $10 \times 30 = 300$ elements de A. Tenint en compte que les files de la matriu s'emmagatzemen contiguament, la segona instrucció calcula a quina distància en bytes del principi (on és `A[0][0]`) es troba l'element `A[1][2]`, distància$= (1 \times 30 + 2) \times 4 = 32 \times 4 = 128$ i emmagatzema en 4 bytes, començant a aquesta distància, la representació de 1 en punt flotant en simple precisió. Aquest codi anterior resulta essencialment equivalent a

```
float A[300]; A[32]=1; /* 30*1+2=32 */
```

L'estructura de dades per emmagatzemar *per columnes* una matriu com la d'abans, és a dir, de manera que les seues columnes siguen contigües, i per accedir al seu element (1, 2), consisteix en el codi següent:

```
float A[300]; A[21]=1; /* 10*2+1=21 */
```

Fortran assumeix emmagatzemament per columnes, de manera que en Fortran les instruccions anàlogues a les anteriors serien:

```
real*4 A(10, 30)
A(2, 3)=1
```

Recordeu que, per defecte, els índexs de les matrius i vectors en Fortran comencen en 1 i en C en 0.

La figura 3.1 mostra gràficament aquestes estructures de dades.

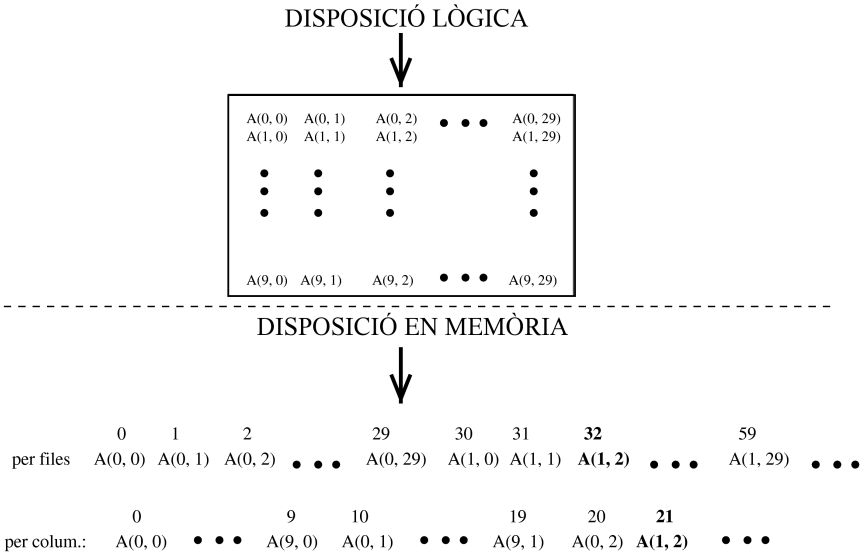


Figura 3.1: Emmagatzemament per files i per columnes.

Probablement aquesta és l'única estructura de dades per a matrius coneguda fins ara. En aquest tema presentarem altres estructures que ens serviran per estalviar memòria i per facilitar l'accés a les dades.

3.2.2 Multiplicació de matrius

La multiplicació de matrius és, sense cap mena de dubte, una operació fonamental en àlgebra lineal. Constitueix la base de molts altres algorismes, entre ells l'eliminació de Gauss, tal com veurem més endavant. És ben conegut que la multiplicació $C = AB$ de dues matrius A $m \times n$ i B $n \times p$ s'efectua multiplicant les files de A per les columnes de B, de manera que l'algorisme 3 1 implementa aquesta operació. Tanmateix, podem plantejar 5 algorismes alternatius, considerant les altres permutacions dels bucles i, j, k. Totes aquestes variants es poden interpretar des del punt de vista de l'àlgebra lineal: per exemple, l'algorisme 3 2 calcula en cada pas pel bucle exterior j la columna $C(:, j)$, fent la combinació lineal de les columnes $A(:, k)$ amb coeficients $B(k, j)$, $k = 1, \ldots, n$.

Estudiem les prestacions d'aquests algorismes. Executem cada algorisme per a matrius quadrades A, B, C, $n \times n$, $n = 100, 200, \ldots, 1000$. Mesurem la "velocitat" dels algorismes en *megaflops* (*mflops*),

```
for i = 1,...,m
  for j = 1,...,p
    for k = 1,...,n
      C(i,j) = C(i,j) + A(i,k)B(k,j)
    end
  end
end
```
(1)

```
for j = 1,...,p
  for k = 1,...,n
    for i = 1,...,m
      C(i,j) = C(i,j) + A(i,k)B(k,j)
    end
  end
end
```
(2)

```
for i = 1,...,m
  for k = 1,...,n
    for j = 1,...,p
      C(i,j) = C(i,j) + A(i,k)B(k,j)
    end
  end
end
```
(3)

Algorisme 3: Diversos algorismes per calcular el producte de dues matrius.

és a dir, en milions de operacions en punt flotant per segon. Obtenim aquesta xifra dividint el nombre d'operacions $2n^3$ entre el temps de CPU utilitzat per als càlculs. Finalment representem la xifra de mflops contra la dimensió de les matrius per a cada algorisme en la gràfica de la figura 3.2. Deduïm d'aquest experiment que la versió jki (és a dir, aquesta és la disposició dels bucles de l'exterior cap a l'interior) és la que millors prestacions ofereix, gairebé el doble que la versió usual ijk, basada en productes interiors. Sorprenentment, la versió ikj és 4 vegades més lenta que la seua versió dual jki. Per què?

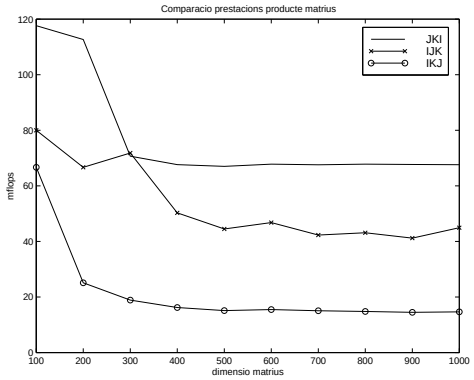


Figura 3.2: Comparació de les prestacions en mflops dels algorismes de 3 per a diferents dimensions de les matrius.

3.2.3 Multiplicació de matrius a blocs

Suposem que volem multiplicar dues matrius quadrades A, B, C $n \times n$ i que les tenim descompostes per blocs $m \times m$:

$$A = \begin{bmatrix} A_{11} & \dots & A_{1l} \\ \vdots & & \vdots \\ A_{l1} & \dots & A_{ll} \end{bmatrix} \quad B = \begin{bmatrix} B_{11} & \dots & B_{1l} \\ \vdots & & \vdots \\ B_{l1} & \dots & B_{ll} \end{bmatrix} \quad C = \begin{bmatrix} C_{11} & \dots & C_{1l} \\ \vdots & & \vdots \\ C_{l1} & \dots & C_{ll} \end{bmatrix} \qquad n = m \times l.$$

És a dir, A_{11} és la submatriu de A formada per les files $1, \dots m$ i les columnes $1, \dots, m$, A_{12} és la submatriu de A formada per les files $1, \dots, m$ i les columnes $m+1, \dots, 2m$, etc. La multiplicació de les dues matrius es pot efectuar fent operacions amb els blocs de manera anàloga a l'emprada en els algorismes 3.

En la figura 3.3 comparem la velocitat de l'algorisme 4 amb la versió més ràpida de l'algorisme 3. La conclusió que obtenim és que la velocitat de la multiplicació per blocs és quasi constant respecte de la

```
for j = 1,...,l
  for k = 1,...,l
    for i = 1,...,l
      C_ij = C_ij + A_ik B_kj
    end
  end
end
```

Algorisme 4: Algorisme de multiplicació de matrius a blocs.

dimensió, i que la velocitat asimptòtica de la versió sense blocs ($\approx 67mflops$) és poc més de la meitat de la velocitat asimptòtica del producte per blocs (112mflops) i 5 vegades inferior a una versió del producte de matrius altament optimitzat (297 mflops).

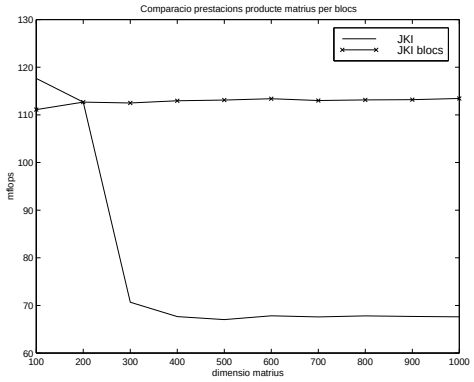


Figura 3.3: Comparació de les prestacions en mflops de l'algorisme jki de 3 amb la seua versió per blocs 4 per a diferents dimensions de les matrius.

Veurem més endavant la motivació de considerar algorismes com aquest per obtenir millors prestacions.

3.2.4 LU a blocs

Suposem que tenim la matriu A $n \times n$ descomposta per blocs $m \times m$ (per simplificar idees, els suposem tots quadrats i de la mateixa dimensió, encara que l'únic que caldria suposar és que els de la diagonal ho són). A més, suposarem que els blocs de la diagonal $A_{11}, \ldots, A_{ll}$ són tots invertibles. En cas que no poguérem assegurar aquesta invertibilitat *a priori* s'hi poden incorporar tècniques de pivotatge, no entrem en detalls. Els exercicis 1 i 2 indiquen casos on aquesta invertibilitat es pot assegurar a priori.

L'objectiu d'aquesta secció és descriure com podem calcular la descomposició LU de A fent operacions matricials amb els blocs A_{ij}. Concretament, veurem que l'algorisme que obtenim és totalment anàleg a l'algorisme clàssic de la figura 1, només que operacions matricials entre blocs substitueixen les escalars. Plantegem descomposicions per blocs de L i U compatibles amb la de A, és a dir

$$L = \begin{bmatrix} L_{11} & 0 & \ldots & 0 \\ L_{21} & L_{22} & \ldots & 0 \\ \vdots & \vdots & & \vdots \\ L_{l1} & \ldots & \ldots & L_{ll} \end{bmatrix} \qquad U = \begin{bmatrix} U_{11} & U_{12} & \ldots & U_{1l} \\ 0 & U_{22} & \ldots & U_{2l} \\ \vdots & \vdots & & \vdots \\ 0 & \ldots & \ldots & U_{ll} \end{bmatrix} \qquad L_{ij}, U_{ij}\ m \times m$$

Els blocs de la diagonal de L han de ser de tipus triangular inferior amb uns a la diagonal, i els de U de tipus triangular superior. Plantegem l'equació matricial $LU = A$ i la seua descomposició per blocs:

$$\begin{bmatrix} L_{11} & 0 & \dots & 0 \\ L_{21} & L_{22} & \dots & 0 \\ \vdots & \vdots & & \vdots \\ L_{l1} & \dots & \dots & L_{ll} \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} & \dots & U_{1l} \\ 0 & U_{22} & \dots & U_{2l} \\ \vdots & \vdots & & \vdots \\ 0 & \dots & \dots & U_{ll} \end{bmatrix} = \begin{bmatrix} L_{11}U_{11} & L_{11}U_{12} & \dots & L_{11}U_{1l} \\ L_{21}U_{11} & & & \\ & & & \\ L_{l1}U_{11} & & & \end{bmatrix} = \begin{bmatrix} A_{11} & \dots & A_{1l} \\ \vdots & & \\ A_{l1} & & \end{bmatrix}$$

Igualant les primeres files de blocs dels dos membres d'aquesta equació obtenim

$$L_{11}U_{11} = A_{11}, L_{11}U_{12} = A_{12}, \dots, L_{11}U_{1l} = A_{1l}$$

Veiem que $L_{11}U_{11} = A_{11}$, i com que L_{11} i U_{11} són com es requereix en la descomposició LU, aleshores són els factors d'aquesta descomposició per a A_{11}. Aquests factors es poden calcular amb l'algorisme 1 1.

Una vegada obtingut L_{11} podem obtenir la resta de la primera fila de blocs de U:

$$U_{12} = L_{11}^{-1}A_{12}, \dots, U_{1l} = L_{11}^{-1}A_{1l}$$

Com que L_{11} és triangular inferior amb uns a la diagonal, els productes anteriors es poden implementar aplicant l'algorisme de substitució cap endavant 1 2 a cadascuna de les columnes de $U_{12}, \dots, U_{1l}$.

De la mateixa manera, podem obtenir la resta de la primera columna de L a partir de les igualtats

$$L_{21}U_{11} = A_{21}, \dots, L_{l1}U_{11} = A_{l1} \Rightarrow$$
$$L_{21} = A_{21}U_{11}^{-1}, \dots, L_{l1} = A_{l1}U_{11}^{-1}$$

Els productes $A_{i1}U_{11}^{-1}$ es poden implementar amb substitució cap endavant, ja que

$$A_{i1}U_{11}^{-1} = ((U_{11}^T)^{-1}(A_{i1}^T))^T$$

i U_{11}^T és triangular inferior.

Una vegada coneixem la primera columna de L i la primera fila de U ens plantegem trobar la resta. Plantegem ara la descomposició per blocs següent:

$$\left[\begin{array}{c|ccc} L_{11} & 0 & \dots & 0 \\ \hline L_{21} & L_{22} & \dots & 0 \\ \vdots & \vdots & & \vdots \\ L_{l1} & \dots & \dots & L_{ll} \end{array}\right] \left[\begin{array}{c|ccc} U_{11} & U_{12} & \dots & U_{1l} \\ \hline 0 & U_{22} & \dots & U_{2l} \\ \vdots & \vdots & & \vdots \\ 0 & \dots & \dots & U_{ll} \end{array}\right] = \left[\begin{array}{c|ccc} A_{11} & \dots & A_{1l} \\ \hline \vdots & & \vdots \\ A_{l1} & \dots & A_{ll} \end{array}\right]$$

Igualant els blocs (2:n,2:n) dels dos membres tenim:

$$\begin{bmatrix} L_{21} \\ \vdots \\ L_{l1} \end{bmatrix} \begin{bmatrix} U_{12} & \dots & U_{1l} \end{bmatrix} + \begin{bmatrix} L_{22} & \dots & 0 \\ \vdots & & \vdots \\ L_{l2} & \dots & L_{ll} \end{bmatrix} \begin{bmatrix} U_{22} & \dots & U_{2l} \\ \vdots & & \vdots \\ 0 & \dots & U_{ll} \end{bmatrix} = \begin{bmatrix} A_{22} & \dots & A_{2l} \\ \vdots & & \vdots \\ A_{l2} & \dots & A_{ll} \end{bmatrix} \Rightarrow$$
$$\begin{bmatrix} L_{22} & \dots & 0 \\ \vdots & & \vdots \\ L_{l2} & \dots & L_{ll} \end{bmatrix} \begin{bmatrix} U_{22} & \dots & U_{2l} \\ \vdots & & \vdots \\ 0 & \dots & U_{ll} \end{bmatrix} = \begin{bmatrix} A_{22} & \dots & A_{2l} \\ \vdots & & \vdots \\ A_{l2} & \dots & A_{ll} \end{bmatrix} - \begin{bmatrix} L_{21} \\ \vdots \\ L_{l1} \end{bmatrix} \begin{bmatrix} U_{12} & \dots & U_{1l} \end{bmatrix} \Rightarrow$$

$$\begin{bmatrix} L_{22} & \dots & 0 \\ \vdots & & \vdots \\ L_{l2} & \dots & L_{ll} \end{bmatrix} \begin{bmatrix} U_{22} & \dots & U_{2l} \\ \vdots & & \vdots \\ 0 & \dots & U_{ll} \end{bmatrix} = \begin{bmatrix} A_{22} - L_{21}U_{12} & \dots & A_{2l} - L_{21}U_{1l} \\ \vdots & & \vdots \\ A_{l2} - L_{l1}U_{12} & \dots & A_{ll} - L_{l1}U_{1l} \end{bmatrix} \tag{3.1}$$

Aquesta equació ens dóna una descomposició LU per blocs de la matriu de la dreta, calculada restant a cada bloc A_{ij} $i, j = 2, \ldots, l$ el producte $L_{i1}U_{1j}$. Sota la suposició que la matriu de la dreta de (3.1) té blocs invertibles en la diagonal, podem establir un algorisme de càlcul de la descomposició LU per blocs d'una matriu:

Input: A matriu $l \times l$ a blocs $m \times m$
for $k = 1, \ldots, l$
  $A_{kk} = [L_{kk} \backslash U_{kk}]$ % resultat eliminació de Gauss per a A_{kk}
  for $j = k+1, \ldots, l$
    $A_{kj} = L_{kk}^{-1} A_{kj}$ % $A_{kj} = U_{kj}$
  end
  for $i = k+1, \ldots, l$
    $A_{ik} = A_{ik} U_{kk}^{-1}$ % $A_{ik} = L_{ik}$
  end
  for $i = k+1, \ldots, l$
    for $j = k+1, \ldots, l$
      $A_{ij} = A_{ij} - A_{ik} A_{kj}$ %$A_{ij} = A_{ij} - L_{ik} U_{kj}$
    end
  end
end

Algorisme 5: Algorisme LU per blocs.

En la figura 3.4 comparem la velocitat de l'algorisme 5 amb totes les versions de l'algorisme 1. La conclusió que obtenim és que la velocitat del càlcul de la descomposició LU per blocs és quasi constant respecte de la dimensió i gairebé el doble (105 mflops) que la velocitat asimptòtica de la versió sense blocs més ràpida ($\approx 67 mflops$). Una versió altament optimitzada de la descomposició LU per blocs permet millorar la velocitat unes 4 vegades (265 mflops).

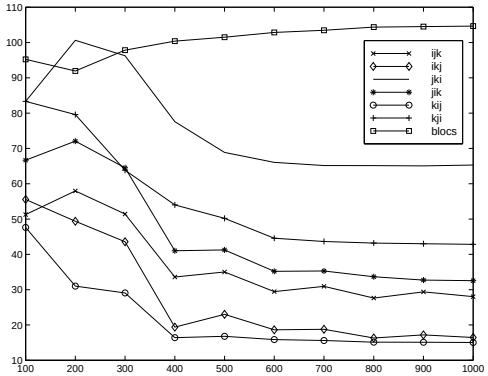


Figura 3.4: Comparació de les prestacions en mflops dels algorismes per calcular la descomposició LU amb l'algorisme 5 per a diferents dimensions de les matrius.

3.2.5 Aprofitament cau per algorismes a blocs

La velocitat a la qual *poden* fer operacions les unitats de punt flotant dels processadors actuals sol ser bastant major que la de la memòria RAM d'on agafa les dades per fer les operacions. La qüestió que hom pot fer-se davant d'aquest fet és: per a què serveix que el processador puga fer 100 milions d'operacions

per segon si les dades sobre les quals ha de fer-les sols li arriben a ritme de 10 milions per segon? En absència de més raons, sembla que la resposta és que sols podrà fer 10 milions d'operacions.

Vegem què passa si hi ha repeticions en les dades. Donem-ne un exemple hipotètic, amb nombres d'operacions més petits perquè resulten més manejables: suposarem que la unitat de punt flotant fa una operació per segon si ha d'agafar els operands des de la RAM. Suposem que tenim 10 nombres reals $a_1, \ldots, a_{10}$ i ens proposem fer les 45 operacions:

$$a_1 + a_2, a_1 + a_3, \ldots, a_1 + a_{10}, a_2 + a_3, \ldots, a_9 + a_{10} \tag{3.2}$$

Per fer la primera, cal agafar a_1 i a_2 des de memòria RAM i fer-ne la suma, la qual cosa tarda un segon; per fer la següent operació $a_1 + a_3$, cal agafar a_3 des de RAM, per tant tardem un altre segon a efectuar-la. Seguint aquest raonament, tardaríem 45 segons a fer les 45 operacions.

Tanmateix, suposem que hi ha una petita part del processador on es poden guardar 10 dades de manera que la unitat de punt flotant puga accedir-hi ràpidament i fer operacions al ritme de 10 per segon. Després d'haver calculat $a_1 + a_{10}$, si el processador guardara les últimes dades carregades des de RAM en aquesta petita memòria ràpida anomenada *memòria cau*[1], aleshores, com que tenim $a_1, \ldots a_{10}$ en la memòria cau, les 36 operacions restants es poden fer a ritme de 10 per segon. L'existència d'aquesta petita memòria cau ens ha permès reduir el temps d'execució d'aquest exemple de 45 segons a 9+3,6=12,6, la qual cosa suposa fer aproximadament 3,6 operacions per segon. En casos més realistes, [2] el temps d'execució d'aquest algorisme estaria molt prop del màxim possible, sols *pagaríem el preu* de carregar inicialment les dades a la memòria cau, per poder fer després moltes operacions amb les dades en la memòria cau a la velocitat màxima.

El que hom podria plantejar-se, en vista dels avantatges de la memòria cau, és: per què no substituïm la memòria RAM per memòria cau? La resposta és que les relacions preu/capacitat, grandària/capacitat, temperatura/capacitat de les memòries cau fan que sols siga viable tenir memòries cau d'alguns centenars de kb, enfront dels centenars de Mb de RAM de què solen disposar els processadors actuals.

Una vegada establerts els avantatges de gaudir de memòria cau, estudiarem l'algorisme anterior sota la suposició que certes restriccions sobre les relacions */capacitat han fet que disposem d'una memòria cau que pot contenir únicament 6 dades. Una representació gràfica del que exposem tot seguit la podeu trobar a la figura 3.5.

En arribar a fer l'operació $a_1 + a_7$, la memòria cau ja conté $a_1, \ldots, a_6$. El processador carrega a_7 i el posa en la memòria cau en el lloc de la dada que fa més temps que no s'ha utilitzat, a_2 en aquest cas. Continuant d'aquesta manera, quan s'arriba a fer $a_2 + a_3$, cap d'aquestes dades no és a la memòria cau, ja que cal carregar-les-hi i expulsar-ne a_5 i a_6. Continuant d'aquesta manera, la primera vegada que trobarem les dades en la memòria cau és quan haurem de calcular $a_5 + a_{10}$ i d'ací fins a $a_9 + a_{10}$ no caldrà carregar les dades des de RAM. El cost d'execució d'aquest algorisme ha estat:

$$1 \times 34 + \frac{1}{10} \times 11 = 35,1 \text{ segons}$$

és a dir 1,3 operacions per segon, prop del ritme d'execució sense memòria cau, i més prop com més gran és el nombre de dades.

Sembla que es puga deduir d'ací que la memòria cau sols resulta útil quan les dades d'un problema són suficientment reduïdes per a cabre en la memòria cau. Tanmateix, aquesta conclusió és equivocada, tal com veurem en l'exemple tot seguit. Aquest exemple il·lustra el següent principi general:

> Si un algorisme fa moltes més operacions que dades, aleshores es pot trobar alguna reformulació de l'algorisme que aprofite la memòria cau.

[1] La qual cosa fan els processadors actuals.

[2] En la memòria cau dels Intel Pentium II caben al voltant de 4.000 reals en simple precisió.

El nostre algorisme és un d'aquests, ja que fa $\approx \frac{1}{2}n^2$ operacions sobre n dades. Farem una reordenació de les operacions

$$\begin{array}{l} \overbrace{a_1+a_2,\ldots a_1+a_6,a_2+a_3,\ldots,a_5+a_6,}^{\text{10 ops cau/5 ops RAM, } 10*0.1+5*1=6 \text{ segons,}} \\ \overbrace{a_1+a_7,\ldots a_1+a_{10},a_2+a_7,\ldots,a_2+a_{10},\ldots a_6+a_7,\ldots,a_6+a_{10},}^{\text{15 ops cau/9 ops RAM, } 15*0.1+9*1=10.5 \text{ segons}} \\ \overbrace{a_7+a_8,\ldots,a_7+a_{10},\ldots a_9+a_{10}}^{\text{6 ops cau/0 ops RAM, } 6*0.1=0.6 \text{ segons}} \end{array} \tag{3.3}$$

El temps d'execució és de 17,1 segons, o 2,6 operacions per segon. Amb una capacitat de la memòria cau i un nombre de dades major el ritme s'aproparia al màxim de 10 operacions per segon.

i	operació	t	cache	
1	a_1+a_2	1.0	1 1 ? ? ? ? a_1 a_2 ? ? ? ?	i càrrega dades
2	a_1+a_3	2.0	2 1 2 ? ? ? a_1 a_2 a_3 ? ? ?	
⋮				
5	a_1+a_6	5.0	5 1 2 3 4 5 a_1 a_2 a_3 a_4 a_5 a_6	
6	a_1+a_7	6.0	6 6 2 3 4 5 a_1 a_7 a_3 a_4 a_5 a_6	
7	a_1+a_8	7.0	7 6 7 3 4 5 a_1 a_7 a_8 a_4 a_5 a_6	
⋮				
34	a_5+a_9	34.0	32 33 34 30 31 34 a_7 a_8 a_9 a_{10} a_6 a_5	
35	a_5+a_{10}	34.1	32 33 34 35 31 35 a_7 a_8 a_9 a_{10} a_6 a_5	
36	a_6+a_7	34.2	36 33 34 35 36 35 a_7 a_8 a_9 a_{10} a_6 a_5	
⋮				
44	a_8+a_{10}	35.0	42 44 43 44 39 35 a_7 a_8 a_9 a_{10} a_6 a_5	
45	a_9+a_{10}	35.1	42 44 45 45 39 35 a_7 a_8 a_9 a_{10} a_6 a_5	

i	operació	t	cache	
1	a_1+a_2	1.0	1 1 ? ? ? ? a_1 a_2 ? ? ? ?	i càrrega dades
2	a_1+a_3	2.0	2 1 2 ? ? ? a_1 a_2 a_3 ? ? ?	
⋮				
16	a_1+a_7	7.0	16 16 12 14 15 15 a_1 a_7 a_3 a_4 a_5 a_6	
17	a_1+a_8	8.0	17 16 17 14 15 15 a_1 a_7 a_8 a_4 a_5 a_6	
18	a_1+a_9	9.0	18 16 17 18 15 15 a_1 a_7 a_8 a_9 a_5 a_6	
⋮				
39	a_6+a_{10}	16.5	35 36 37 38 39 39 a_5 a_7 a_8 a_9 a_{10} a_6	
40	a_7+a_8	16.6	35 40 40 38 39 39 a_5 a_7 a_8 a_9 a_{10} a_6	
41	a_7+a_9	16.7	35 41 40 41 39 39 a_5 a_7 a_8 a_9 a_{10} a_6	
42	a_7+a_{10}	16.8	35 42 40 41 42 39 a_5 a_7 a_8 a_9 a_{10} a_6	
43	a_8+a_9	16.9	35 42 43 43 42 39 a_5 a_7 a_8 a_9 a_{10} a_6	
44	a_8+a_{10}	17.0	35 42 44 43 44 39 a_5 a_7 a_8 a_9 a_{10} a_6	
45	a_9+a_{10}	17.1	35 42 44 45 45 39 a_5 a_7 a_8 a_9 a_{10} a_6	

Figura 3.5: A l'esquerra, alguns detalls de la utilització de la memòria cau de l'algorisme 3.2, i a la dreta els de 3.3. La primera columna de cadascuna d'aquestes taules indica l'índex de l'operació efectuada, la segona aquesta operació efectuada, la tercera el temps en segons transcorreguts, i l'última, l'estat de la memòria cau. En la primera fila de l'estat de la memòria cau posem quan s'ha accedit per última vegada a les dades que apareixen en la segona fila.

Analitzem ara un exemple més interessant. Suposem que tenim tres matrius A, B, C $n \times n$, les quals caben en la memòria cau, i que volem fer l'operació $C = C + A * B$, per a la qual emprem l'algorisme 3 1. Designem com a t_m els temps per fer una operació agafant les dades de memòria RAM, i com a t_c el temps per fer una operació agafant les dades de la memòria cau.

Després de passar pel bucle extern quan $i = 1$ hem carregat a la memòria cau tota la matriu B, les primeres files de A i C, a les quals no s'accedeix posteriorment. En el pas segon, tercer, etc. pel bucle extern sols cal que carreguem les files segones, terceres, etc. de A i C, ja que B ja és a la memòria cau i les carregues posteriors de les files de A i C no expulsen B fora, perquè hi ha lloc per a les tres matrius a la memòria cau. El compte del temps és:

- operacions que requereixen càrrega des de RAM: $2n^2 + \overbrace{2n + \cdots + 2n}^{n-1} \approx 4n^2$
- operacions que *no* requereixen càrrega des de RAM: $\approx 2n^3 - 4n^2$

El temps de computació és $\approx 4n^2t_m + (2n^3 - 4n^2)t_c$. El mínim absolut seria $2n^3t_c$, i la relació que existeix entre aquests dos temps és

$$\frac{4n^2t_m + (2n^3 - 4n^2)t_c}{2n^3t_c} = 1 + \frac{2}{n}(\frac{t_m}{t_c} - 1)$$

Si $n = 100$ i $\frac{t_m}{t_c} = 10$, aleshores aquesta fracció és $1 + 0,02 * 9 = 1,18$, és a dir, que el temps d'execució és sols un 18% superior al mínim, en comptes de $10 = \frac{t_m}{t_c}$ vegades superior si no existira la memòria cau.

3.2.6 LU a bandes

Les matrius a bandes, a mig camí entre les matrius denses i les disperses, suposen una altra situació que permet millores substancials en l'eliminació de Gauss. Suposem que els elements no nuls de la matriu A estan localitzats en les diagonals $-l, \ldots, 0, \ldots, l$. Es comprova (vegeu l'exercici 7) que els factors L i U de la descomposició LU de A tenen zeros on A els té, dit altrament: els únics elements no nuls de L es troben en les diagonals $-l, \ldots, 0$ i els de U en les diagonals $0, \ldots, l$. Dit en poques paraules:

La descomposició LU d'una matriu a bandes és a bandes.

La primera millora que ens permet aquest fet és estalviar operacions amb l'algorisme següent:

Input: A matriu $n \times n$
for $k = 1, \ldots, n-1$
  for $i = k+1, \ldots, \min(n, k+l)$
    $A(i,k) = A(i,k)/A(k,k)$
    for $j = k+1, \ldots, \min(n, k+l)$
      $A(i,j) = A(i,j) - A(i,k)A(k,j)$
    end
  end
end
Output: part estrictament triangular inferior de L sota la diagonal, part triangular superior de U sobre la diagonal, incloent-la.

Algorisme 6: Algorisme LU per a matrius a bandes, amb estructura de dades de matriu densa.

El nombre de passos pel bucle j és $\leq l$ i el nombre d'operacions és $\leq 2l$. El nombre de passos pel bucle i és $\leq l$, ja que el nombre d'operacions per cada pas és $\leq l \times 2l = 2l^2$. El nombre total d'operacions de l'algorisme és $\leq 2nl^2$.

L'estalvi d'operacions d'aquest algorisme respecte a l'algorisme usual és espectacular: tal com veurem més endavant, les matrius que apareixen en moltes aplicacions son $n \times n$ i tenen una amplada de banda (inferior i superior) de $\sqrt{n}$. Fins i tot per a volums moderats $n = 10000$, el nombre d'operacions del l'algorisme usual és $\frac{\frac{2}{3}n^3}{2nl^2} = \frac{n}{3} \approx 3300$ vegades superior al d'aquest últim.

L'altra millora consisteix en l'estalvi de memòria que comporta sols guardar els elements no nuls. Mostrem l'estructura de dades que permet aquest estalvi en la figura següent:

3.3 Matrius disperses

Quan considerem que la proporció d'elements no nuls d'una matriu és suficientment petita com per considerar la matriu dispersa? Des d'un punt de vista teòric, la definició que tindria sentit plantejar-se per

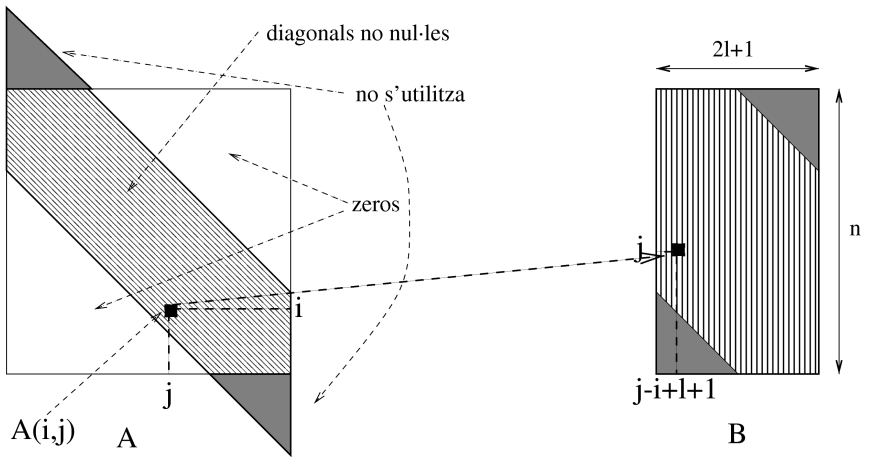


Figura 3.6: Estructura de dades per emmagatzemar eficientment una matriu a bandes. B és una matriu $n \times (2l+1)$ i $B(j, j-i+l+1) = A(i,j)$, si $|i-j| \leq l$.

a matrius disperses seria que una família parametritzada de matrius $(A(n))_{n\in\mathbb{N}}$, $A(n)$ matriu $n \times n$, és dispersa quan

$$\lim_n \frac{\text{nombre d'elements no nuls } (A(n))}{n^2} = 0.$$

Però aquesta hipotètica definició no té utilitat practica. Des d'un punt de vista pràctic, es pot dir que una matriu és dispersa quan és avantatjós utilitzar algorismes i/o estructures de dades diferents dels de les matrius denses per millorar substancialment el temps de computació o l'ús de memòria. Així doncs, sota aquest principi, no consideraríem una matriu tridiagonal 3×3 com a dispersa, ja que no obtindríem millores substancials. El concepte de matriu dispersa és, en resum, un tant subjectiu i dependent del problema.

De les matriu conegudes fins ara, podem considerar com a disperses les diagonals i tridiagonals, les quals permeten resoldre sistemes en $\mathcal{O}(n)$ operacions; també considerem disperses les matrius amb una amplada de banda petita l en comparació amb la dimensió de la matriu, amb les quals podem resoldre sistemes en $\mathcal{O}(l^2 n)$ operacions.

3.3.1 Estalvi de memòria

Tot seguit presentem dues estructures de dades per estalviar memòria en l'emmagatzemament de matrius disperses.

Emmagatzemament per diagonals

L'emmagatzemament per diagonals s'utilitza quan la matriu té poques diagonals no nul·les i és molt semblant a l'estructura de dades que hem utilitzat per a les matrius a bandes, només que per a aquestes matrius les diagonals són contigües, i en el nostre cas poden no ser-ho, per la qual cosa caldrà també emmagatzemar els índexs de les diagonals no nul·les. Vegem-ho en un exemple.

Exemple 3.2.

$$A = \underbrace{\begin{bmatrix} 1 & 0 & 5 & 0 \\ 0 & 3 & 0 & 6 \\ 2 & 0 & 7 & 0 \\ 9 & 4 & 0 & 8 \end{bmatrix}}_{\text{matriu original}} \Rightarrow B = \underbrace{\begin{bmatrix} 9 & 2 & 1 & 0 \\ 0 & 4 & 3 & 0 \\ 0 & 0 & 7 & 5 \\ 0 & 0 & 8 & 6 \end{bmatrix}}_{\text{diagonals}} \quad d = \underbrace{\begin{bmatrix} -3 \\ -2 \\ 0 \\ 2 \end{bmatrix}}_{\text{índex diagonals}}$$

Emmagatzemament CRS

L'emmagatzemament CRS (*Compressed Row Storage*) és un tipus d'emmagatzemament més general que l'anterior, ja que no requereix que hi hagen poques diagonals no nul·les perquè resulte eficient. Consisteix a utilitzar tres vectors: en el primer guardem el valor de cada element no nul de la matriu, ordenats per files, en el segon guardem les columnes on es troben aquests elements no nuls, i en el tercer, els índexs dins d'aquests dos vectors on comença cada fila. Ho veurem millor amb un exemple

Exemple 3.3.

$$A = \begin{bmatrix} 1 & 0 & 5 & 0 \\ 0 & 3 & 0 & 6 \\ 2 & 0 & 7 & 0 \\ 9 & 4 & 0 & 8 \end{bmatrix} \Rightarrow \begin{array}{l} v = \left[\begin{array}{c|cc|cc|cc|ccc|} & 1 & 5 & 3 & 6 & 2 & 7 & 9 & 4 & 8 & \end{array} \right] \\ \qquad\quad 1 \qquad\quad 3 \qquad\quad 5 \qquad\quad 7 \qquad\quad 10 \\ ja = \left[\begin{array}{c|cc|cc|cc|ccc|} & 1 & 3 & 2 & 4 & 1 & 3 & 1 & 2 & 4 & \end{array} \right] \\ \qquad\quad 1 \qquad\quad 3 \qquad\quad 5 \qquad\quad 7 \qquad\quad 10 \\ ia = \left[\begin{array}{ccccc} 1 & 3 & 5 & 7 & 10 \end{array} \right] \end{array}$$

3.3.2 Estalvi d'operacions

A l'hora d'eliminar les entrades de la part subdiagonal d'una certa columna d'una matriu dispersa, l'estalvi d'operacions ve per dos llocs:

1. Sols eliminem aquelles entrades no nul·les i per tant el bucle i s'executa sols quan $A(i,k) \neq 0$.
2. A l'hora de afegir a una fila la fila pivotal multiplicada pel factor adequat, sols caldrà que fem l'operació per a aquelles entrades de la fila pivotal que no siguen nul·les.

Tanmateix, just en aquest punt pot ocórrer el fenomen del *fill-in*, en el qual algunes entrades nul·les d'una fila esdevenen no nul·les en sumar-los un múltiple no nul de l'entrada no nul·la de la mateixa columna de la fila pivotal; és a dir, quan fem l'operació $A(i,j) = A(i,j) - A(i,k) * A(k,j)$ pot passar que $A(i,j) = 0$, però $A(i,k), A(k,j) \neq 0$, amb la qual cosa $A(i,j)$ passa a ser no nul·la. Aquest fenomen pot ser especialment greu si no prenem mesures, tal com ens ho il·lustra l'exemple següent.

Exemple 3.4. La matriu representada en la figura 3.7, a l'esquerra, és una matriu dispersa. Però té una particularitat: la primera fila i la primera columna contenen molts elements no nuls (de fet, tots). Perquè el primer pas de l'eliminació de Gauss produeix "l'ompliment" de les zones no nul·les de la matriu. En el segon pas, l'algorisme es troba ja amb una matriu densa!

Si no fem res per evitar-ho, el fenomen del *fill-in* tindrà les següents conseqüències:

1. Es necessitarà més memòria per contenir els factors de la descomposició LU que per contenir la matriu original. En el cas anterior, els factors són densos.
2. Si es creen entrades no nul·les addicionals, caldrà eliminar-les en passos posteriors, la qual cosa comporta més treball i pot crear encara més *fill-in*, el qual duu més treball. ...

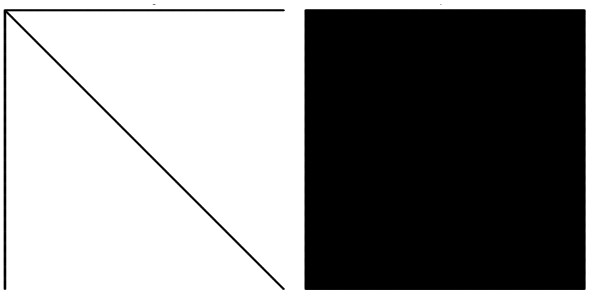


Figura 3.7: Patró de dispersitat de la matriu original i de la factorització LU. La part negra simbolitza entrades no nul·les i la part blanca entrades nul·les.

Reordenacions

El cas de l'exemple anterior il·lustra una de les tècniques utilitzades per reduir el *fill-in* i per tant la memòria i temps de computació necessaris. Si reordenem les files i columnes de la matriu en ordre invers, la solució del sistema essencialment no canvia (les entrades la solució obtinguda apareixen en ordre invers). La matriu reordenada la trobem en la figura 3.8. Veiem que podem aplicar-li l'eliminació de Gauss en $\mathcal{O}(n)$ operacions.

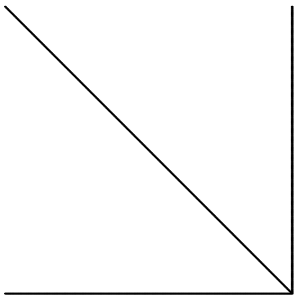


Figura 3.8: La matriu de la figura 3.7 amb files i columnes ordenades inversament.

El que suggereix aquest exemple és que caldria reordenar files i columnes de la matriu de menor a major en funció dels elements que continguen; és a dir, per a $i = 1, \ldots, n$ definim s_i com la suma del nombre d'elements no nuls en la fila i i en la columna i. Ordenem ara aquestes quantitats de menor a major:

$$s_{i_1} \leq s_{i_1} \leq \cdots \leq s_{i_n}$$

La permutació de les files i columnes de A segons $(i_1, \ldots, i_n)$ s'anomena *reordenació per columnes* i tendeix a donar una matriu amb amplària de banda creixent. No és la millor reordenació, però és senzilla de calcular. La *reordenació per mínim grau* és més sofisticada, però utilitza principis similars i sol donar molt bons resultats.

Altres reordenacions cerquen obtenir amplàries de banda més o menys constants i el més petit possible. Entre aquestes citem la de Cuthill-McKee inversa.

Exemple 3.5. Generem aleatòriament una matriu simètrica i definida positiva, el patró de dispersitat de la qual mostrem en la figura 3.9 de dalt a l'esquerra. La proporció d'entrades no nul·les és aproximadament un 2%. En calculem les reordenacions per mínim grau i per Cuthill-McKee inversa, i apliquem l'algorisme LU a totes tres matrius per obtenir els següents resultats:

reordenació	cap	mínim grau	Cuthill-McKee inversa
no zeros LU	30558	9258	18596
segons de CPU per LU	0,39	0,08	0,18

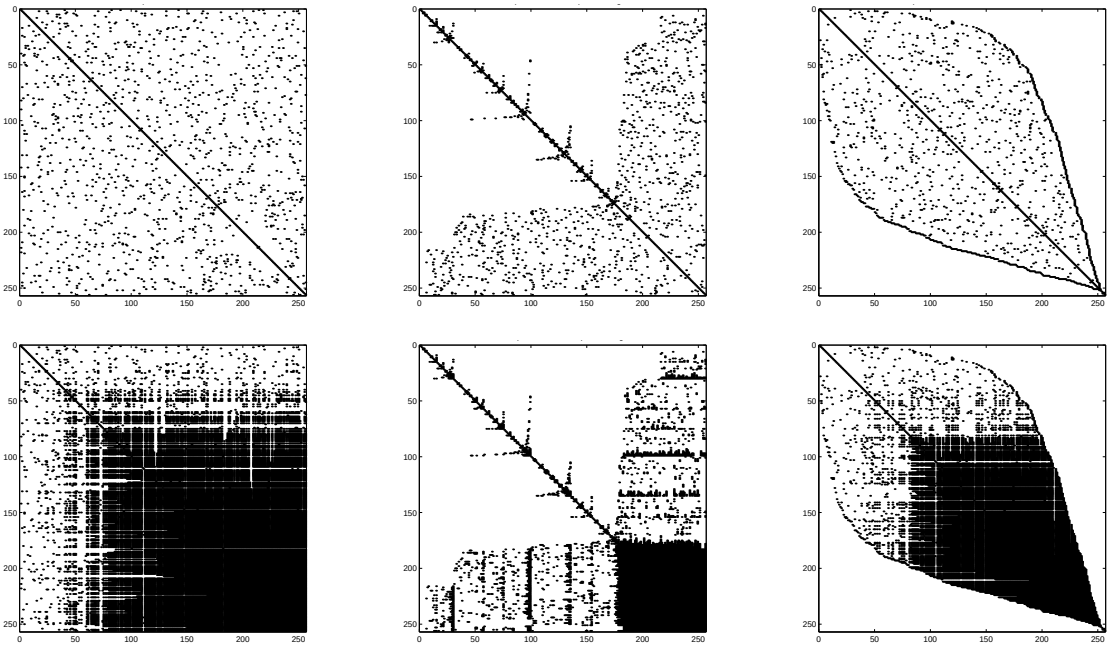


Figura 3.9: D'esquerra a dreta, en la filera de dalt, matriu original, després de fer permutació de mínim grau i després de permutació Cuthill-McKee inversa; en la filera de baix, descomposició LU de la matriu original, descomposició LU de la matriu després de fer permutació de mínim grau, descomposició LU de la matriu després de fer permutació de Cuthill-McKee.

3.4 Problemes

Problema 1. Si A és simètrica definida positiva, aleshores aquesta suposició és certa. Per provar-ho, demostreu:

1. Els blocs de la diagonal d'una matriu simètrica definida positiva ho són també.
2. Si una matriu A és simètrica i definida positiva i designem $\overline{A}$ el resultat del primer pas $(k = 1)$ de l'algorisme 3 1, aleshores la *submatriu de treball* $\overline{A}(2 : n, 2 : n)$ és simètrica i definida positiva.
3. Demostreu que la matriu resultant de fer el pas $k = 1$ de l'algorisme 5 és la mateixa que la resultant de fer els passos $k = 1, \ldots, m$ de l'algorisme 1 1.

Problema 2. El mateix si A és diagonalment dominant (per files o per columnes).

Problema 3. Interpreteu les altres variants de la multiplicació matricial que no apareixen en 3.

Problema 4. Interpreteu les altres variants de l'algorisme per calcular la descomposició LU que no apareixen en 1.

Problema 5. Què passa amb l'algorisme 3 si A, B, C no caben en la memòria cau? Analitzeu el temps d'execució dels algorismes 3 i 4 sota aquesta suposició.

Problema 6. Ídem per a l'algorisme 5.

Problema 7. Si els elements no nuls de la matriu A estan localitzats en les diagonals $-l, \ldots, 0, \ldots, l$ i $A = LU$ és una descomposició LU de A, aleshores la matriu L té els seus elements no nuls en les diagonals $-l, \ldots, 0$ i els de U en les diagonals $0, \ldots, l$.

Problema 8. Escriviu l'algorisme 6 utilitzant l'estructura de dades de la figura 3.6

.

Problema 9. Escriviu una rutina que torne l'element (i, j) d'una matriu A emmagatzemada per diagonals.

Problema 10. Escriviu una rutina que torne l'element (i, j) d'una matriu A emmagatzemada en format CRS.

Problema 11. Escriviu un algorisme eficient per calcular la factorització LU de la matriu de la figura 3.8.

Problema 12. Trobeu la discretització per diferències finites de l'equació de Poisson en el domini de la figura 2.3.

Tema 4

Mètodes iteratius per a sistemes lineals

4.1 Introducció

4.1.1 Els mètodes de Jacobi, Gauss-Seidel i SOR

Els mètodes iteratius bàsics són de tipus lineal i s'obtenen de la manera següent:

1. es descompon $A = M - N$, on M és una aproximació de A,
2. es planteja l'equació $x = M^{-1}(Nx + b)$, equivalent a $Ax = b$
3. es dedueix d'ací la iteració
$$x_{i+1} = M^{-1}(Nx_i + b), \tag{4.1}$$
on $x_{(0)}$ és una aproximació inicial donada.

La multiplicació $M^{-1}(Nx_i + b)$ s'implementa resolent el sistema amb matriu M i terme independent $Nx_i + b$. En deduïm que resoldre sistemes amb matriu M cal que siga més senzill que amb A perquè aquest mètode siga efectiu.

Si $A = L + D + U$ és una descomposició de A com en la figura 4.1, aleshores l'elecció que mostrem en la taula següent dóna els mètodes iteratius bàsics ja coneguts:

M	**Mètode**
D	Jacobi
$D + L$	Gauss Seidel
$\frac{1}{\omega}D + L$	SOR

El paràmetre ω del mètode de *sobrerelaxació successiva* (SOR) pertany a l'interval $(0, 2)$. Cal adonar-se que aquestes tres matrius són fàcilment invertibles, la primera és diagonal i les altres dues són triangulars inferiors. Observeu també que el mètode de Gauss-Seidel és un cas particular de SOR per a l'elecció $\omega = 1$.

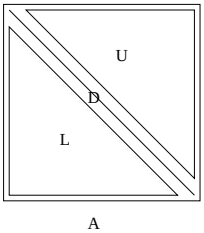


Figura 4.1: Partició $A = L + D + U$ per a l'obtenció de mètodes iteratius.

Tal com veurem més endavant en aquest tema, és important tenir matrius precondicionadores simètriques i definides positives quan la matriu A ho siga. La de Jacobi ho és (per què?), però les altres dues no. "Simetritzem" SOR (i, com a conseqüència, Gauss-Seidel). El *SOR invers* es defineix, de manera dual a la del SOR, com el mètode iteratiu lineal que té per matriu precondicionadora $M = \frac{1}{\omega}D + U$ (vegeu l'exercici 1 per a una justificació del nom). Suposem ara que A és simètrica i definida positiva i plantegem la iteració consistent a fer un pas de SOR seguit d'un pas de SOR invers (cal tenir en compte que $U = L^T$ perquè A és simètrica):

$$
\begin{aligned}
\tilde{x}_{i+1} &= (\frac{1}{\omega}D + L)^{-1}(-((1-\frac{1}{\omega})D + L^T)x_i + b) \quad \text{(ja que } A - M = (1-\frac{1}{\omega})D + L^T) \\
x_{i+1} &= (\frac{1}{\omega}D + L^T)^{-1}(-((1-\frac{1}{\omega})D + L)\tilde{x}_{i+1} + b) \quad \text{(ja que } A - M = (1-\frac{1}{\omega})D + L) \\
&= (\frac{1}{\omega}D + L^T)^{-1}\Big(- \big((1-\frac{1}{\omega})D + L\big)(\frac{1}{\omega}D + L)^{-1}\Big(- \big((1-\frac{1}{\omega})D + L^T\big)x_i + b\Big) + b\Big) \\
&= (\frac{1}{\omega}D + L^T)^{-1}\Big(\big(\cdots\big)x_i + \Big(- \big((1-\frac{1}{\omega})D + L\big)(\frac{1}{\omega}D + L)^{-1} + I\Big)b\Big) \\
&= (\frac{1}{\omega}D + L^T)^{-1}\Big(\big(\cdots\big)x_i + \Big(- \big((1-\frac{1}{\omega})D + L\big)(\frac{1}{\omega}D + L)^{-1} + I\Big)b\Big) \\
&= (\frac{1}{\omega}D + L^T)^{-1}\big(\cdots\big)x_i + (\frac{1}{\omega}D + L^T)^{-1}\frac{2-\omega}{\omega}D(\frac{1}{\omega}D + L)^{-1}b
\end{aligned}
$$

Per la forma que tenen els mètodes iteratius, deduïm que la inversa de la matriu precondicionadora que defineix aquest mètode és

$$M^{-1} = (\frac{1}{\omega}D + L^T)^{-1}\frac{2-\omega}{\omega}D(\frac{1}{\omega}D + L)^{-1},$$

d'on obtenim la matriu precondicionadora del mètode de *sobrerelaxació successiva simètrica* (SSOR)

$$M = (\frac{1}{\omega}D + L)\frac{\omega}{2-\omega}D^{-1}(\frac{1}{\omega}D + L^T). \tag{4.2}$$

Aquesta matriu és simètrica i definida positiva sempre que A ho és.

4.1.2 Convergència dels mètodes iteratius lineals

Un mètode iteratiu es diu que convergeix quan la successió generada pel mètode convergeix (obligatòriament a la solució del sistema) per a qualsevol aproximació inicial $x_{(0)}$. Els valors propis de la matriu $Q = M^{-1}N$ donen la clau de la convergència del mètode iteratiu. Donem tot seguit un recull dels fets més importants sobre la convergència d'aquests mètodes:

1. El mètode és convergent si i només si el radi espectral de Q (el màxim dels mòduls dels valors propis de Q) verifica $\rho(Q) < 1$.
2. Si A és estrictament diagonalment dominant aleshores Jacobi i Gauss-Seidel convergeixen.
3. Si A és simètrica i definida positiva i $\omega \in (0, 2)$ aleshores SOR convergeix.
4. Com a conseqüencia, si A és simètrica i definida positiva, aleshores Gauss-Seidel convergeix.

En molts casos d'interès, entre ells la resolució numèrica d'algunes EDPs com la del problema de Poisson, el mètode de Gauss-Seidel és més ràpid que el de Jacobi i, si el paràmetre ω es tria de certa manera òptima, es pot aconseguir una millora substancial de la convergència de SOR respecte a la de Gauss-Seidel (és a dir, SOR per a $\omega = 1$).

4.1.3 Avantatges i inconvenients dels mètodes iteratius

Hi ha aplicacions que requereixen resoldre sistemes lineals amb una precisió baixa. Per exemple, la discretització de l'equació de Poisson que hem plantejat abans és de segon ordre, és a dir, *l'error de discretització*, la diferència entre la solució del problema continu (l'equació de Poisson) i el discretitzat és $\mathcal{O}(h^2)$. Si $h = 0,01$, aleshores cometem un error de l'ordre de 10^{-4} en aproximar la solució de l'equació de Poisson per la solució del sistema resultant de la discretització. Si cometem un error del mateix ordre en la resolució d'aquest sistema, l'error global resulta ser del mateix ordre que si el resolem exactament.

Si el cost de la resolució dels sistemes fóra proporcional respecte de la precisió requerida, aquestes aplicacions es podrien dur a terme més ràpidament. L'eliminació de Gauss és molt lluny de complir aquesta propietat, ja que si aturem l'algoritme a la meitat no sols no obtenim la meitat dels dígits de la solució, sinó que no n'obtenim cap!

Un altre avantatge és que un mètode iteratiu com els que hem presentat permet aprofitar el fet que una matriu dispersa tinga molts zeros d'una manera clara: en els casos anteriors, multiplicar per N i resoldre sistemes amb matriu M es pot efectuar en $\mathcal{O}(\text{no zeros de } A)$, sense problemes de *fill-in* perquè la matriu M sol ser triangular. A més, sovint no cal ni emmagatzemar les matrius M i N per fer-ne ús en mètodes iteratius, amb la qual cosa la memòria requerida per a resoldre un sistema és $\mathcal{O}(n)$. Aquesta quantitat, que en principi pot parèixer modesta, no ho és tant en algunes aplicacions on n és de l'ordre de $10^6, 10^7$. Un mètode directe pot ser senzillament inaplicable en aquests casos, perquè podríem tenir memòria sols per emmagatzemar una desena de vectors de grandària n, però no tenir prou memòria per emmagatzemar la matriu, encara que fóra utilitzant estructures de dades sofisticades. Per tots aquests motius, un mètode iteratiu de convergència ràpida aplicat a una matriu dispersa pot significar una reducció en el cost computacional molt important sobre un mètode directe o, senzillament, ser l'única alternativa.

Tammateix, els mètodes iteratius no són la solució universal, ja que perquè siguen d'utilitat cal tenir garantida la convergència a priori, la qual cosa en limita la capacitat d'utilització als casos esmentats abans. Així i tot, aquests casos són suficientment significatius com perquè els mètodes iteratius siguen una eina insustituïble per mètodes directes en nombroses aplicacions.

4.2 El mètode del gradient conjugat

Com que la convergència del mètode de Gauss-Seidel no sol ser massa ràpida, presentem ací un mètode iteratiu amb propietats de convergència superiors. No es tracta d'un mètode lineal, basat en una partició de la matriu $A = M - N$. El mètode dels *gradients conjugats* és més sofisticat que els mètodes iteratius lineals i està relacionat amb problemes de minimització de polinomis, encara que no entrarem a analitzar aquest aspecte.

Abans de començar a desenvolupar el mètode proposem dues definicions i un teorema.

Definició 4.1. Siga $x, y \in \mathbb{R}^n$ definim el producte interior com:

$$(x, y) = x^T y = \sum_{i=1}^{n} x_i y_i$$

Proposició 4.2. Siguen $x, y, z \in \mathbb{R}^n$ i siga $\alpha \in \mathbb{R}$, aleshores:

1. $(x, y) = (y, x)$
2. $(\alpha x, y) = \alpha(x, y)$
3. $(x + z, y) = (x, y) + (z, y)$

4. $(x, x) \geq 0$

5. $(x, x) = 0 \Leftrightarrow x = 0$

Definició 4.3. Direm que $A \in \mathcal{M}^{n\times n}$ és definida positiva si $x \in \mathbb{R}^n \backslash \{0\}$:

$$(x, Ax) = x^T Ax > 0$$

Observem que si A és simètrica aleshores es compleix que, si $x, y \in \mathbb{R}^n$:

$$(x, Ay) = x^T Ay = x^T A^T y = (Ax)^T y = y^T (Ax) = (y, Ax).$$

En aquesta secció suposarem que $A \in \mathcal{M}^{n\times n}$ és una matriu simètrica i definida positiva (SPD).
Ara definim la funció quadràtica $g\colon \mathbb{R}^n \to \mathbb{R}$

$$g(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i \sum_{j=1}^{n} A_{ij} x_j - 2\sum_{i=1}^{n} b_i x_i$$

o, vectorialment,

$$g(x) = x^T Ax - 2x^T b = (x, Ax) - 2(x, b)$$

que és fonamental per al desenvolupament del mètode del gradient conjugat.

El següent resultat és una eina bàsica per al desenvolupament del mètode del gradient conjugat.

Proposició 4.4. El vector $x \in \mathbb{R}^n$ és solució del sistema lineal simètric i definit positiu $Ax = b$ si minimitza:

$$g(x) = (x, Ax) - 2(x, b)$$

Prova. En primer lloc tenim que si A és simètrica i x_* és solució del sistema $Ax_* = b$, aleshores per a tot $y \in \mathbb{R}^n$

$$g(y) = g(x_*) + (y - x_*)^T A(y - x_*). \tag{4.3}$$

A més, com que A és definida positiva tenim que si $y \neq x_*$, aleshores $y - x_* \neq 0$ i per tant $g(y) > g(x_*)$ amb la qual cosa la x solució de $Ax = b$, x_*, és un mínim del funcional $g(x)$.

Provarem ara l'altra implicació; per fer-ho, donats $x, p \neq 0 \in \mathbb{R}^n$ vectors fixos i $t \in \mathbb{R}$ un nombre variable, definim:

$$\begin{aligned} g(x + tp) &= (x + tp, A(x + tp)) - 2(x + tp, b) \\ &= (x, Ax) + t(p, Ax) + t(p, Ax) + t^2(p, Ap) - 2(x, b) - 2t(p, b) \\ &= (x, Ax) - 2(x, b) + 2t(p, Ax) - 2t(p, b) + t^2(p, Ap) \\ &= g(x) + 2t(p, Ax - b) + t^2(p, Ap) \end{aligned} \tag{4.4}$$

Si definim la funció quadràtica en la variable t com $h(t) = g(x + tp)$ tenim que $h'(t) = 2(p, Ax - b) + 2t(p, Ap)$ i es minimitza quan

$$\hat{t} = -\frac{(p, Ax - b)}{(p, Ap)} = \frac{(p, b - Ax)}{(p, Ap)}. \tag{4.5}$$

De l'equació (4.4) obtenim:

$$\begin{aligned} h(\hat{t}) &= g(x) + 2\hat{t}(p, Ax - b) + \hat{t}^2(p, Ap) \\ &= g(x) + 2\frac{(p, b - Ax)}{(p, Ap)}(p, Ax - b) + (\frac{(p, b - Ax)}{(p, Ap)})^2(p, Ap) \\ &= g(x) - 2\frac{(p, b - Ax)^2}{(p, Ap)} + \frac{(p, b - Ax)^2}{(p, Ap)} \\ &= g(x) - \frac{(p, b - Ax)^2}{(p, Ap)}. \end{aligned} \tag{4.6}$$

Ara, si x_* minimitza g, per a tot $p \in \mathbb{R}^n$ tenim que:

$$g(x_*) \leq g(x_* + \hat{t}p) = g(x_*) - \frac{(p, b - Ax_*)^2}{(p, Ap)}.$$

Això només pot passar si $(p, b - Ax_*) = 0, \quad \forall p \in \mathbb{R}^n$, és a dir, si $Ax_* - b = 0$.

Per començar amb el mètode del gradient conjugat, agafem una solució aproximada de $Ax_* = b$ i una direcció de cerca, $p \neq 0$, per intentar aconseguir una millor aproximació a x_*. Si $r = b - Ax \neq 0$ és el vector resta associat amb x, i si p i r no són ortogonals, aleshores si definim

$$t = \frac{(p, b - Ax)}{(p, Ap)} = \frac{(p, r)}{(p, Ap)}$$

aleshores, $g(x + tp) < g(x)$ i possiblement està més prop de x_* que x. Així, suggerim el següent mètode:

Suposem x_0 una aproximació inicial a x_*, siga $p_0 \neq 0$ una direcció de cerca inicial. Per a $i = 1, 2, \ldots,$ calculem:

$$t_i = \frac{(p_{i-1}, b - Ax_{i-1})}{(p_{i-1}, Ap_{i-1})}$$

$$x_i = x_{i-1} + t_i p_{i-1}$$

i agafem una nova direcció de cerca p_i. L'objectiu és fer una elecció de manera que la successió d'aproximacions convergisca ràpidament a x_*.

4.2.1 El mètode del màxim pendent.

Una manera de triar les direccions de cerca seria la següent. Com que

$$g(x) = (x, Ax) - 2(x, b) = \sum_{i,j=1}^{n} a_{i,j}x_i x_j - 2\sum_{i=1}^{n} x_i b_i,$$

si calculàvem les derivades parcials respecte a la component x_i tindríem que:

$$\frac{\partial g(x)}{\partial x_l} = 2\sum_{i=1}^{n} a_{l,i}x_i - 2b_l$$

per tant, el gradient de g és:

$$\nabla g = (\frac{\partial g(x)}{\partial x_1}, \frac{\partial g(x)}{\partial x_2}, \ldots, \frac{\partial g(x)}{\partial x_n})^T = 2(Ax - b) = -2r. \tag{4.7}$$

Utilitzant anàlisis de diverses variables (regla de la cadena):

$$g(x+tp) = g(x) + t\nabla g(x)p + o(t^2) \quad t > 0$$

com que $\nabla g(x)p = \| \nabla g(x) \| \| p \| cos(\theta)$, tindrem que obtindrem el màxim pendent quan $p = -\nabla g(x)$, és a dir, quan $cos(\theta) = -1$.

En aquest cas agafaríem $p_i = r_i = b - Ax_i$ i el mètode s'anomenaria del màxim pendent (algorisme 1).

inici: A, b, x_0
$r_0 = b - Ax_0$
$p_0 = r_0$
for $i = 1, \ldots$
  $t_i = (p_{i-1}, r_{i-1})/(p_{i-1}, Ap_{i-1})$
  $x_i = x_{i-1} + t_i p_{i-1}$
  $r_i = b - Ax_i$
  $p_i = r_i$
end

Algorisme 1: Algorisme del màxim pendent. Per simplificar-ho, hem omès el criteri de parada, el qual sol estar basat en la quantitat $\|r_i\|/\|b\|$.

El problema del mètode del màxim pendent és que ens dóna el descens més ràpid localment, que no és el més ràpid globalment. De fet, es pot provar que $(p_i, p_{i-1}) = 0, \quad \forall i$. És a dir, pren direccions ortogonals en cada pas i, òbviament, eixa no és la millor manera de baixar.

Exercici 4.5. Amb la notació utilitzada anteriorment, proveu que per al mètode del màxim pendent les direccions successives són perpendiculars.

Prova. Com que

$$p_i = r_i = b - Ax_i = b - Ax_{i-1} - t_i Ap_{i-1},$$

tenim

$$(p_i, p_{i-1}) = (b - Ax_{i-1}, p_{i-1}) - t_i(Ap_{i-1}, p_{i-1}) = 0, \quad \forall i$$

4.2.2 El mètode de les direccions conjugades

Plantegem un mètode alternatiu usant un conjunt de vectors de direccions no nul·les $\{p_0, p_1, \ldots, p_{n-1}\}$ que són A-ortogonals, és a dir, compleixen:

$$(p_i, Ap_j) = 0, \quad i \neq j.$$

No es difícil provar que un conjunt de vectors A-ortogonals associats amb la matriu definida positiva A és linealment independent.

La següent proposició mostra que aquesta elecció de direccions de cerca dóna la convergència en n passos sempre que assumim que l'aritmètica siga exacta.

Proposició 4.6. Siga $\{p_0, p_1, \ldots, p_{n-1}\}$ un conjunt A-ortogonal de vectors no nuls associats amb la matriu A SDP i siga x_0 arbitrari; aleshores, si definim

$$t_i = \frac{(p_{i-1}, b - Ax_{i-1})}{(p_{i-1}, Ap_{i-1})} \quad i \quad x_i = x_{i-1} + t_i p_{i-1} \quad i = 1, 2, \ldots, n$$

Aleshores, assumint aritmètica exacta, $Ax_n = b$

Prova. Com per a cada $i = 1, 2, \ldots, n, \quad x_i = x_{i-1} + t_i p_{i-1}$ tenim que:

$$\begin{aligned} Ax_n &= Ax_{n-1} + t_n Ap_{n-1} \\ &= Ax_{n-2} + t_{n-1}Ap_{n-2} + t_n Ap_{n-1} \\ &\vdots \\ &= Ax_0 + t_1 Ap_0 + \ldots + t_n Ap_{n-1} \end{aligned} \tag{4.8}$$

Si restem b i multipliquem per p_{i-1} tenim que:

$$\begin{aligned} (Ax_n - b, p_{i-1}) &= (Ax_0 + t_1 Ap_0 + \ldots + t_n Ap_{(n)} - b, p_{i-1}) \\ &= (Ax_0 - b, p_{i-1}) + t_1(Ap_0, p_{i-1}) + \ldots + t_n(Ap_n, p_n) \\ &= (Ax_0 - b, p_{i-1}) + t_i(Ap_{i-1}, p_{i-1}) \end{aligned} \tag{4.9}$$

per ser A-ortogonals. Per altra part, tenim

$$\begin{aligned} t_i(Ap_{i-1}, p_{i-1}) &= (p_{i-1}, b - Ax_{i-1}) \\ &= (p_{i-1}, b - Ax_0 + Ax_0 \ldots - Ax_{i-2} + Ax_{i-2} - Ax_{i-1}) \\ &= (p_{i-1}, b - Ax_0) + (p_{i-1}, Ax_0 - Ax_1) + \ldots + (p_{i-1}, Ax_{i-2} - Ax_{i-1}) \\ &= (p_{i-1}, b - Ax_0) + (p_{i-1}, Ax_0 - Ax_1) + \ldots + (p_{i-1}, Ax_{i-2} - Ax_{i-1}) \\ &= (p_{i-1}, b - Ax_0) - t_1(p_{i-1}, Ap_0) - \ldots - t_{i-1}(p_{i-1}, Ap_{i-2}) \\ &= (p_{i-1}, b - Ax_0) \end{aligned} \tag{4.10}$$

De (4.9) i de (4.10) obtenim que $(Ax_n - b, p_{i-1}) = 0$, i. e., que $Ax_n - b$ és ortogonal al conjunt de vectors A-ortogonals i, per tant, vectors independents (vegeu el problema 6). Això implica que $Ax_n - b = 0$.

Exercici 4.7. Els vectors residuals r_i, $i = 1, \ldots, n$, per a un mètode de direcció conjugada satisfan les següents equacions:

$$(r_i, p_j) = 0, \quad j = 0, \ldots, i-1$$

4.2.3 El mètode del gradient conjugat

El mètode del gradient conjugat tria les direccions de cerca p_{i-1} de manera que els vectors r_i siguen mútuament ortogonals. Vegem com es podrien construir els vectors de direcció $\{p_0, p_1, \ldots, p_{n-1}\}$.

Proposició 4.8. Siga x_0 una aproximació inicial aleatòria. Siga la primera direcció de cerca $p_0 = r_0 = b - Ax_0$. Si calculem, per i=1, ...

$$\begin{aligned} t_i &= \frac{(p_{i-1}, r_{i-1})}{(p_{i-1}, Ap_{i-1})} \\ x_i &= x_{i-1} + t_i p_{i-1} \\ r_i &= b - Ax_i \\ s_i &= -\frac{(p_{i-1}, Ar_i)}{(p_{i-1}, Ap_{i-1})} \\ p_i &= r_i + s_i p_{i-1} \end{aligned}$$

aleshores, $\forall i$, $(p_i, Ap_j) = 0$ i $(r_i, r_j) = 0$ per a $j = 0, \ldots, i-1$.

Prova. Ho provarem per inducció.

Per a $i = 1$, $(r_1, r_0) = (r_1, p_0) = 0$ per l'exercici 4.7 i $(p_1, Ap_0) = (r_1 + s_1 p_0, Ap_0) = (r_1, Ap_0) + s_1(p_0, Ap_0) = 0$.

Suposem que és cert fins i i vegem que també ho és per a $i + 1$.

D'una banda, tenim que per a $j = 0, \ldots, i$, $(r_{i+1}, r_j) = (r_{i+1}, p_j - s_j p_{j-1}) = (r_{i+1}, p_j) - s_j(r_{i+1}, p_{j-1}) = 0$, per l'exercici 4.7.

D'altra banda, queda provar que $(p_{i+1}, Ap_j) = 0$ per $j = 1, \ldots, i-1$. En efecte,

$$\begin{aligned}(p_{i+1}, Ap_j) &= (p_j, Ap_{i+1}) = \frac{1}{t_{i+2}}((p_j, Ax_{i+2} - Ax_{i+1}) \\ &= \frac{1}{t_{i+2}}((p_j, -b + Ax_{i+2} + b - Ax_{i+1}) \\ &= \frac{1}{t_{i+2}}((p_j, r_{i+1}) - (p_j, r_{i+2})) = 0 \quad j = 1, \ldots, i.\end{aligned}$$

Nota 4.9. Com a motivació del nom de gradient conjugat podem dir que, donat que $(r_j, r_i) = 0$ per a $i \neq j$, per (4.7) tenim que els gradients de g en les iteracions x_i de l'algorisme són *conjugats* (sinònim d'ortogonals).

Així, el mètode del gradient conjugat es podria escriure de la següent manera (algorisme 2).

inici: A, b, x_0
$r_0 = b - Ax_0$
$p_0 = r_0$
for $i = 1, \ldots$
  $t_i = (p_{i-1}, r_{i-1})/(p_{i-1}, Ap_{i-1})$
  $x_i = x_{i-1} + t_i p_{i-1}$
  $r_i = b - Ax_i$
  $s_i = -(p_{i-1}, Ar_i)/(p_{i-1}, Ap_{i-1})$
  $p_i = r_i + s_i p_{i-1}$
end

Algorisme 2: Algorisme dels gradients conjugats sense optimitzar. Versió 1. Per simplificar-ho, hem omès el criteri de parada, el qual sol estar basat en la quantitat $\|r_i\|/\|b\|$.

En l'algorisme tal com està escrit cal fer, per cada iteració, tres productes de matriu per vector (en realitat). Això es pot reduir fent uns canvis.

En primer lloc, com que

$$\begin{aligned}t_i &= \frac{(p_{i-1}, r_{i-1})}{(p_{i-1}, Ap_{i-1})} = \frac{(r_{i-1} + s_{i-1}p_{i-2}, r_{i-1})}{(p_{i-1}, Ap_{i-1})} \\ &= \frac{(r_{i-1}, r_{i-1})}{(p_{i-1}, Ap_{i-1})} + s_{i-1}\frac{(p_{i-2}, r_{i-1})}{(p_{i-1}, Ap_{i-1})},\end{aligned}$$

Per l'exercici (4.7) deduïm que:

$$t_i = \frac{(r_{i-1}, r_{i-1})}{(p_{i-1}, Ap_{i-1})} \tag{4.11}$$

També, de $x_i = x_{i-1} + t_i p_{i-1}$, obtenim $Ax_i - b = Ax_{i-1} - b + t_i Ap_{i-1}$, és a dir, $r_i = r_{i-1} - t_i Ap_{i-1}$.

Així,

$$(r_i, r_i) = (r_{i-1}, r_i) - t_i(Ap_{i-1}, r_i) = -t_i(r_i, Ap_{i-1}).$$

Si ajuntem això amb l'equació (4.2.3), obtenim

$$\begin{aligned} s_i &= -\frac{(p_{i-1}, Ar_i)}{(p_{i-1}, Ap_{i-1})} = -\frac{(r_i, Ap_{i-1})}{(p_{i-1}, Ap_{i-1})} \\ &= \frac{(1/t_i)(r_i, r_i)}{(1/t_i)(r_{i-1}, r_{i-1})} = \frac{(r_i, r_i)}{(r_{i-1}, r_{i-1})} \end{aligned}$$

Així, l'algorisme del gradient conjugat es pot escriure de la següent manera (algorisme 3)

inici: A, b, x_0
$r_0 = b - Ax_0$
$p_0 = r_0$
for $i = 1, \ldots$
  $t_i = (r_{i-1}, r_{i-1})/(p_{i-1}, Ap_{i-1})$
  $x_i = x_{i-1} + t_i p_{i-1}$
  $r_i = r_{i-1} - t_i Ap_{i-1}$
  $s_i = (r_i, r_i)/(r_{i-1}, r_{i-1})$
  $p_i = r_i + s_i p_{i-1}$
end

Algorisme 3: Algorisme dels gradients conjugats sense optimitzar. Versió 2. Per simplificar-ho, hem omès el criteri de parada, el qual sol estar basat en la quantitat $\|r_i\|/\|b\|$.

Adoneu-vos que per cada iteració sols cal calcular un producte Ap_{i-1} per la matriu A i emmagatzemar-lo en un vector q_i que es pot utilitzar posteriorment per calcular r_{i+1}. Si designem com a *axpy* l'operació $y = y + ax$ i per *dot* $(x, y) = x_T y$, el cost d'una iteració és aproximadament

$$\text{cost(producte matriu} \times \text{vector)} + 3\text{cost}(axpy) + 2\text{cost}(dot) \approx \text{cost}(A\times) + 10n$$

Respecte a la memòria necessària per a una implementació eficient, a més de A, b, x_i, cal utilitzar 3 vectors addicionals r_i, q_i, p_{i-1}, suposant que en cada iteració cada $*_{i+1}$ s'escriu en comptes de $*_i$.

Si anomenem $\alpha_i = t_{i-1}$ i $\beta_i = s_{i-1}$, l'algorisme 3 és equivalent a l'algorisme 4.

4.2.4 Propietats fonamentals

Si A és una matriu simètrica i definida positiva, aleshores $(x, y)_A = (x, Ay)$ és un producte escalar.

Si $p = p(x) = p_n x^n + p_{n-1}x^{n-1} + \cdots + p_1 x + p_0$ és un polinomi i A és una matriu, definim $p(A) = p_n A^n + p_{n-1}A^{n-1} + \cdots + p_1 A + p_0 I$.

Proposició 4.10.

1. Per a $i \geq 0$, hi ha polinomis f_i, g_i, h_i de grau i, tal que
   (a) $x_{i+1} = x_0 + f_i(A)r_0 \in x_0 + K_i$
   (b) $p_i = g_i(A)r_0 \in K_i$
   (c) $r_i = h_i(A)r_0 \in K_i$, $h_i 0 = 1$

inici: A, b, x_0
$r_0 = b - Ax_0$
$p_0 = r_0$
for $i = 0, \ldots$
$\alpha_i = (r_i, r_i)/(p_i, Ap_i)$
$x_{i+1} = x_i + \alpha_i p_i$
$r_{i+1} = r_i - \alpha_i A p_i$
$\beta_i = (r_{i+1}, r_{i+1})/(r_i, r_i)$
$p_{i+1} = r_{i+1} + \beta_i p_i$
end

Algorisme 4: Algorisme dels gradients conjugats. Per simplificar, hem omès el criteri de parada, el qual sol estar basat en la quantitat $\|r_i\|/\|b\|$.

2. $(r_i, r_j) = 0$, $(p_j, p_j)_A = 0$, $\forall i \neq j$.
3. Si $r_i \neq 0$, i designem $K_i = \{p(A)r_0 \colon \text{grau}(p) \leq i\}$ aleshores
   (a) $\{r_0, \ldots, r_i\}$ és base ortogonal de K_i respecte al producte escalar $(\cdot, \cdot)$.
   (b) $\{p_0, \ldots, p_i\}$ és base ortogonal de K_i respecte al producte escalar $(\cdot, \cdot)_A$.

4.2.5 Convergència

L'anàlisi de la convergència del mètode dels gradients conjugats és d'una complexitat relativament alta, de manera que sols anomenem els fets més importants sense donar-ne la demostració.

Proposició 4.11. Gradient conjugat Utilitzant la notació: $0 < \lambda_1 \leq \cdots \leq \lambda_n$, valors propis de A, $\kappa = \frac{\lambda_n}{\lambda_1}$ la condició de A (en norma euclidiana), $e_i = x_* - x_i$, x_* solució del sistema $Ax = b$, $\|x\|_Z = \sqrt{(x, Zx)}$ ($Z = A, A^{-1}$) tenim:

1. $\|e_i\|_A = \|r_i\|_{A^{-1}}$
2. Si h_i ve donat per la proposició 4.10 aleshores
$$\|r_i\|_{A^{-1}} = \|h_i(A)r_0\|_{A^{-1}} = \min_{\text{grau}(h) \leq i, h_i(0)=1} \|h(A)r_0\|_{A^{-1}}.$$
3. $\|e_i\|_A \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^i \|e_0\|_A$
4. $\|e_i\| \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^i \sqrt{\kappa}\|e_0\|$

Proposició 4.12. Màxim pendent

1. $\|e_i\|_A \leq \left(\frac{\kappa-1}{\kappa+1}\right)^i \|e_0\|_A$

Nota 4.13. La base de la potència que apareix en l'apartat 4 de la proposició 4.11 és

$$R = \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} < 1, \tag{4.12}$$

doncs $\lim_i \|e_i\| = 0$, és a dir, la iteració x_i convergeix a la solució x_* i a més a més tenim assegurada una fita sobre el decreixement de $\|e_i\|$. Com que

$$\lim_{\kappa\to 1} \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} = 0 \quad \lim_{\kappa\to\infty} \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} = 1$$

tenim que la convergència serà més ràpida com menor siga la condició de la matriu. En general, podrem assegurar que

$$\frac{\|e_i\|}{\|e_0\|} \leq \varepsilon, \forall i > \log(\varepsilon/(2\sqrt{\kappa}))/\log(R). \tag{4.13}$$

Per a una matriu amb $\kappa = 100$, tenim que $R = \frac{10-1}{10+1} \approx 0.8$. Així, (4.13) indica que caldrà fer $\log(10^{-6}/20)/\log(0.8) \approx 76$ iteracions per garantir una disminució de 10^{-6} en $\|e_i\|$ en comparació amb $\|e_0\|$.

En canvi, si $\kappa = 10^8$, obtenim $R = \frac{10^4-1}{10^4+1} \approx 0,9998$ i caldrà fer $\log(10^{-6}/(2\cdot 10^4))/\log(0,9998) \approx 118.580$ iteracions per garantir la mateixa reducció anterior!

Corol·lari 4.14. Siga p el polinomi mínim de A, és a dir, el polinomi mònic p de grau mínim tal que $p(A) = 0$, aleshores $i = \text{grau}(p) \leq n$ verifica $r_i = 0$, és a dir, $x_i = x_*$.

Prova. Com que A és invertible, $p(0) \neq 0$ (comproveu-ho). Siga $h(x) = p(x)/p(0)$, de grau i i amb $h(0) = 1$. Aleshores per la proposició 4.11 2, $\|r_i\|_{A^{-1}} \leq \|h(A)r_0\|_{A^{-1}} = 0$, d'on $0 = r_i = b - Ax_i$ i $x_i = x_*$.

Nota 4.15. Des d'un punt de vista pràctic aquest resultat és més aviat una curiositat, per dos motius principals:

1. No es verifica per a aritmètica inexacta.
2. Si n és molt gran (de l'ordre de 10^6, per exemple) i el grau del polinomi mínim és $n-1$, aleshores no podem esperar a fer $n-1$ iteracions per obtenir la solució.

L'ús fonamental del resultat 4.11 4 és quan κ és relativament petit, és a dir, per a matrius ben condicionades. L'algorisme dels gradients conjugats per a matrius mal condicionades pot tenir una convergència excessivament lenta per a resultar d'utilitat. En aquest cas, les tècniques que exposem tot seguit resulten crucials.

4.3 Gradient conjugat precondicionat

El resultat 4 de la proposició 4.11 suggereix que la condició de la matriu és el paràmetre fonamental que controla la convergència de l'algorisme dels gradients conjugats. Aquest algorisme aplicat a una matriu ben condicionada (κ petit) resulta que és un mètode extremadament ràpid, però és poc més que inútil quan la condició és gran. El que es pot fer en aquesta (malauradament freqüent) situació és cercar un sistema "equivalent" amb millor condició. Vegem com podem trobar un tal sistema. Si M és una matriu simètrica definida positiva que aproxima d'alguna manera A, aleshores podem plantejar el sistema equivalent

$$M^{-1}Ax = M^{-1}b. \tag{4.14}$$

L'aplicació de l'algorisme dels gradients conjugats a aquesta matriu no és possible en general, perquè la matriu $M^{-1}A$ no és simètrica i definida positiva, malgrat ser-ho totes dues.

4.3.1 Precondicionament simètric

Per resoldre el problema anterior, separem la matriu M en dos factors $M = C^T C$ on C és triangular superior i "dividim" A per aquests factors, un per cada costat. Específicament, podem calcular una descomposició $M = C^T C$ perquè M és simètrica i definida positiva (per exemple, la descomposició de Choleski) i després plantejar el sistema

$$C^{-T} A C^{-1} C x = C^{-T} b$$

el qual s'escriu

$$\overline{A}\overline{x} = \overline{b}, \quad \overline{A} = C^{-T} A C^{-1}, \overline{b} = C^{-T} b, \overline{x} = Cx \tag{4.15}$$

Si $\overline{x}_*$ és solució del sistema (4.15) aleshores $x_* = C^{-1}\overline{x}_*$ és solució de $Ax = b$, ja que, si $\overline{x}$ és aproximació a $\overline{x}_*$, aleshores $C^{-1}\overline{x}$ és aproximació a x_*. La diferència amb el sistema (4.14) és que la matriu $\overline{A} = C^{-T} A C^{-1}$ del sistema (4.15) és simètrica i definida positiva:

$$\overline{A}^T = (C^{-1})^T A^T (C^{-T})^T = C^{-T} A C^{-1} = \overline{A}, \text{ per ser } A \text{ simètrica}$$
$$\overline{x} \neq 0 \Rightarrow \overline{x}^T \overline{A} \overline{x} = (C^{-1}\overline{x})^T A (C^{-1}\overline{x}) > 0, \text{ per ser } A \text{ definida positiva i } C^{-1}x \neq 0$$

La matriu M s'anomena *matriu precondicionadora*, i la tècnica de substituir el sistema $Ax = b$ pel sistema (4.15) rep el nom de *precondicionament simètric*. Es dedueix que l'eficiència d'aquesta tècnica depèn del fet que el càlcul de la descomposició $M = CC^T$ i la resolució de sistemes amb matrius C i C^T siguen senzills.

Designem els valors propis de $M^{-1}A$ per $\overline{\lambda}_1 \leq \cdots \leq \overline{\lambda}_n$ i suposem que M és una bona aproximació a A, en el sentit que els valors propis anteriors són prop de 1. El cas extrem $M = A$ dóna que $M^{-1}A = I$ té tots els valors propis 1, però aquesta tria de matriu precondicionadora no té massa sentit. Vegem quina és la condició de la matriu $\overline{A}$. Com que $\overline{A}$ és semblant a $M^{-1}A$, ja que

$$C^{-1}\overline{A}C = C^{-1}C^{-T}A\overline{C}^{-1}C = M^{-1}A,$$

aleshores $\overline{A}$ té els mateixos valors propis que $M^{-1}A$ i la seua condició és el quocient del major entre el menor: $\frac{\lambda_n}{\lambda_1}$. Com que λ_1 i λ_n són prop de 1, aleshores tenim que $\kappa(\overline{A}) \approx 1$ i l'algorisme dels gradients conjugats resultarà amb una convergència molt ràpida.

4.3.2 L'algorisme del gradient conjugat precondicionat

Plantegem l'algorisme dels gradients conjugats per a $\overline{A}\overline{x} = \overline{b}$ (equació (4.15)), amb aproximació inicial $\overline{x}_0 = Cx_0$. Designarem com a $\overline{r}_i$ el residual del sistema (4.15) per a la iteració i-èsima $\overline{x}_i$ obtinguda per l'algorisme. Així mateix, designarem com a $\overline{p}_i$ els vectors auxiliars que apareixen en l'algorisme.

L'única dificultat que té la implementació pràctica d'aquest algorisme és el producte per la matriu $\overline{A} = C^{-T} A C^{-1}$. No és recomanable formar explícitament la matriu $\overline{A}$, perquè això podria suposar perdre precisió en calcular-la i perquè no tenim cap mena de garantia que la matriu $\overline{A}$ siga dispersa encara que A ho siga. Però si analitzem bé l'estructura de $\overline{A}$ ens adonem que un producte $\overline{q} = \overline{A}\overline{p}$ es pot efectuar de la manera següent:

$$s = C^{-1}\overline{p} \tag{4.16}$$
$$t = As \tag{4.17}$$
$$\overline{q} = (C^T)^{-1} t \tag{4.18}$$

inici: $\overline{A}$, $\overline{b}$, $\overline{x}_0$
$\overline{r}_0 = \overline{b} - \overline{A}\overline{x}_0$
$\overline{p}_0 = \overline{r}_0$
for $i = 0, \ldots$
$\alpha_i = (\overline{r}_i, \overline{r}_i)/(\overline{p}_i, \overline{A}\overline{p}_i)$
$\overline{x}_{i+1} = \overline{x}_i + \alpha_i \overline{p}_i$
$\overline{r}_{i+1} = \overline{r}_i - \alpha_i \overline{A}\overline{p}_i$
$\beta_i = (\overline{r}_{i+1}, \overline{r}_{i+1})/(\overline{r}_i, \overline{r}_i)$
$\overline{p}_{i+1} = \overline{r}_{i+1} + \beta_i \overline{p}_i$
end
$x_t = C^{-1}\overline{x}_t$

Algorisme 5: Algorisme dels gradients conjugats amb precondicionament simètric. La iteració $\overline{x}_t$ és l'obtinguda després d'haver executat el bucle, amb una possible eixida prematura per haver-hi satisfet algun criteri de parada. A partir d'aquesta aproximació a $\overline{x} = C^{-1}x$ obtenim una aproximació $x_t = C^{-1}\overline{x}_t$ a x_*.

Si la matriu A és dispersa, l'aplicació del pas (4.17) es pot efectuar ràpidament. Deduïm que l'eficiència del mètode dependrà de l'eficiència de la resolució de sistemes amb matriu C (pel pas (4.16)) i C^T (pel pas (4.18)). Aquestes matrius solen ser triangulars i disperses, ja que l'aplicació d'aquests dos passos es pot fer de manera eficient amb modificacions dels algorismes de substitució adequades per aprofitar la dispersitat de les matrius.

Nota 4.16. Es pot deduir que el resultat de l'algorisme 5 depèn només de M i no dels factors del producte $M = C^T C$, vegeu l'exercici 5.

4.3.3 Precondicionament diagonal

El precondicionament diagonal consisteix a triar la diagonal D de A com precondicionador. Aquesta tria està justificada des del punt de vista teòric, ja que les entrades de la diagonal d'una matriu definida positiva són positives, però des del punt de vista pràctic no ho és tant, si no és que la diagonal de la matriu té molt de "pes", és a dir, que la matriu és "molt" diagonalment dominant.

Cal adonar-se que $x = D^{-1}b$ es pot obtenir com la primera iteració del mètode de Jacobi aplicat al sistema $Ax = b$ amb $x^0 = 0$:

$$x^{(1)} = D^{-1}(-(A - D)0 + b) = D^{-1}b$$

4.3.4 Precondicionament SSOR

La matriu precondicionadora

$$M = \frac{\omega}{2-\omega}(\frac{1}{\omega}D + L)D^{-1}(\frac{1}{\omega}D + L^T)$$

que defineix el mètode SSOR és una matriu simètrica i definida positiva si A ho és i triem $\omega \in (0, 2)$. Podem descompondre $M = C^T C$, on

$$C = \sqrt{\frac{\omega}{2-\omega}} D^{-\frac{1}{2}}(\frac{1}{\omega}D + L^T),$$

on $D^{-\frac{1}{2}}$ és la matriu diagonal formada per les arrels quadrades dels elements de D (els quals són positius). La matriu C és triangular superior i és dispersa si A ho és, la qual cosa implica que la matriu precondicionadora és fàcilment invertible.

De la mateixa manera que en el precondicionament diagonal, cal adonar-se que $x = M^{-1}b$ es pot obtenir com la primera iteració del mètode de SSOR aplicat al sistema $Ax = b$ amb $x_0 = 0$.

4.3.5 Altres precondicionadors

Hi ha moltes altres tècniques de precondicionament, a banda de les que hem citat ací. Algunes d'aquestes tècniques són de propòsit general, és a dir, que es poden aplicar satisfactòriament per a qualsevol problema, encara que no són tècniques òptimes en la majoria dels casos. Entre aquestes citem una tècnica que consisteix a calcular una *descomposició de Choleski incompleta* de la matriu A. Aquesta s'obté modificant lleugerament l'algorisme original, perquè no produïsca *fill-in*, i obtenir així factors que tenen el mateix patrò de dispersitat que la matriu original.

Hi ha altres precondicionadors que sols funcionen en casos molt concrets i solen obtenir-se tenint en compte el problema original. Així, sovint una equació el·líptica (com la de Poisson) amb coeficients que varien molt suaument pot aproximar-se per una equació amb coeficients constants. La matriu corresponent a aquesta última és ràpidament invertible utilitzant el que s'anomena FFT (transformada ràpida de Fourier), ja que pot servir com a bon precondicionador pel problema amb coeficients variables.

4.4 Experiments numèrics

Considerem les matrius A resultants de discretitzar l'equació de Poisson en malles de $m \times m$ punts, $m = 128, 256, 512$. Aquestes matrius són de dimensions $m^2 \times m^2$, amb $m^2 = 16384, 65536, 262144$, i són disperses. Formem sistemes lineals $Ax = b$, $b_i = 1$, $\forall i$ i els resolem utilitzant els mètodes iteratius següents amb $x^0 = 0$:

1. Mètode dels gradients conjugats (CG).
2. Mètode dels gradients conjugats amb precondicionament per SSOR ($\omega = 1$) (SSOR PCG).
3. Mètode dels gradients conjugats amb precondicionament per descomposició de Choleski incompleta (IC(2) PCG).
4. Mètode de Gauss-Seidel (GS)

Exposem els resultats en la figura 4.2. Deduïm d'aquest experiment que el mètode de Gauss-Seidel té una convergència extremadament lenta i que el mètode dels gradients conjugats amb precondicionament per IC(2) és el més ràpid.

4.5 Problemes

Problema 1. Escriviu els algorismes de SOR i SOR invers i trobeu una justificació al nom d'aquest últim.

Problema 2. Comproveu que la matriu M de (4.2) és simètrica i definida positiva sempre que A ho siga.

Problema 3. † Demostreu la proposició 4.10. Pista: per demostrar l'apartat (2) utilitzeu inducció sobre i per provar que $(r_i, r_j) = 0$ i $(p_i, p_j)_A = 0$ si $j < i$. A l'hora de demostrar que aquestes igualtats són vàlides per a $i + 1$ a partir de i, distingiu els casos $j = i$ i $j < i$.

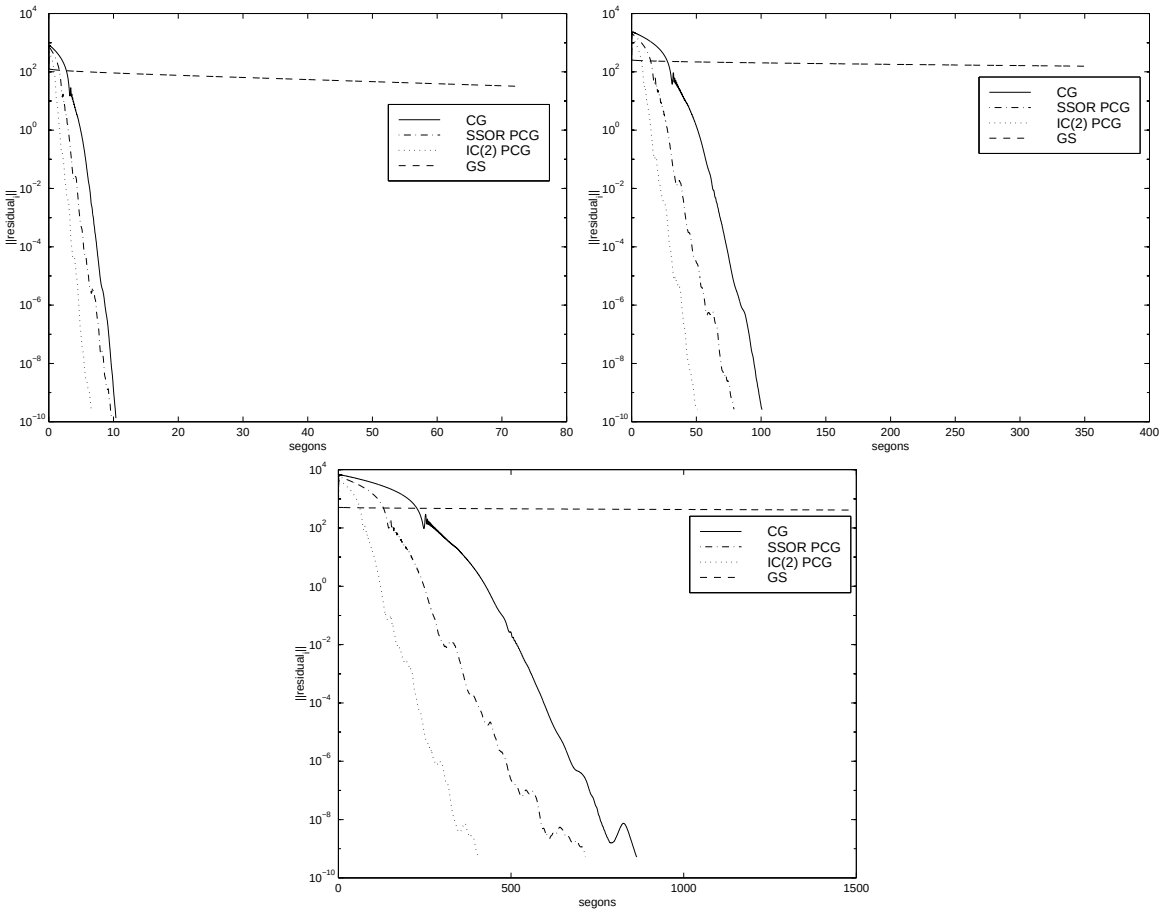


Figura 4.2: Resultats de l'experiment de la secció 4.4. D'esquerra a dreta i de dalt a baix: història de la convergència per a $m = 128, 256, 512$. En l'eix vertical representem el temps en segons transcorregut des de l'inici de l'algorisme i en el vertical la norma del residual de la iteració aconseguida en eixe moment.

Problema 4. Partint del fet que una matriu simètrica A verifica $\lambda_{\min}(A)\|x\|^2 \leq (x, Ax) \leq \lambda_{\max}(A)\|x\|^2$ demostreu que, en la proposició 4.11, l'apartat 3 implica l'apartat 4.

Problema 5. Demostreu que l'algorisme 6 calcula directament les aproximacions $x_i = C^{-1}\overline{x}_i$ de l'algorisme 5.

Pista: introduïu les variables

1. $x_i = C^{-1}\overline{x}_i$,
2. $p_i = C^{-1}\overline{p}_i$,
3. $r_i = b - Ax_i = C^T(C^{-T}b - C^{-T}AC^{-1}Cx_i) = C^T(\overline{b} - \overline{A}\overline{x}_i) = C^T\overline{r}_i$,
4. $z_i = M^{-1}r_i$,

deduïu recurrències per a les noves variables i comproveu que α_i i β_i de l'algorisme 5 es poden escriure en funció de les noves variables com $\alpha_i = (z_i, r_i)/(p_i, Ap_i)$, $\beta_i = (z_{i+1}, r_{i+1})/(z_i, r_i)$

Problema 6. Proveu que si el conjunt de vectors $\{p_0, \ldots, p_{n-1}\}$ són A-ortogonals, aleshores son linealment independents.

inici: A, b, x_0
$r^0 = b - Ax_0$
$z_0 = M^{-1}r^0$
$p_0 = z_0$
for $i = 0, \ldots$
  $\alpha_i = (z_i, r_i)/(p_i, Ap_i)$
  $x_{i+1} = x_i + \alpha_i p_i$
  $r_{i+1} = r_i - \alpha_i A p_i$
  $z_{i+1} = M^{-1} r_{i+1}$
  $\beta_i = (z_{i+1}, r_{i+1})/(z_i, r_i)$
  $p_{i+1} = z_{i+1} + \beta_i p_i$
end

Algorisme 6: Versió equivalent de l'algorisme dels gradients conjugats amb precondicionament simètric per una matriu precondicionadora simètrica i definida positiva M.

Tema 5

Mètodes per a valors i vectors propis

5.1 Introducció

Donem tot seguit alguns resultats i definicions, molts d'ells coneguts. Designem A una matriu $n \times n$.

Un parell (x, λ) ($x \in \mathbb{R}^n$, $\lambda \in \mathbb{R}$) s'anomena *parell (vector/valor) propi* de A quan:

$$Ax = \lambda x, x \neq 0$$

El problema de determinar els parells propis d'una matriu és un problema no lineal (tenim el producte λx de les incògnites), intrínsecament més complicat que els lineals que hem vist fins ara. Tammateix, sabem que les arrels del *polinomi característic* de A, $\det(A - tI)$, són exactament els valors propis de A i que els vectors propis es poden obtenir calculant el nucli de $A - \lambda I$, on λ és un dels valors propis prèviament calculats. Aquesta caracterització ens permet reduir l'equació $Ax - \lambda x = 0$, $x \neq 0$ a resoldre un polinomi per trobar els valors propis i, després, trobar els nuclis d'una sèrie de matrius per trobar els vectors propis. Aquesta caracterització, junt amb el fet que tot polinomi real té arrels complexes, ens permet deduir que tota matriu real té valors i vectors propis *complexos*, però aquests no són necessàriament reals, com ocorre, per exemple, amb la matriu

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix},$$

els valors propis de la qual son $\pm i$. Com que els polinomis de grau major o igual que 5 no són resolubles directament deduïm que no podem obtenir mètodes directes per al càlcul dels valors propis.

L'obtenció separada dels valors i vectors propis no sol ser satisfactòria des del punt de vista numèric, sobretot si aquest procediment es basa a calcular prèviament el polinomi característic de la matriu. La majoria dels mètodes que proposarem estan basats en altres dos principis:

- transformacions de semblança,
- potències de la matriu

Dues matrius A, B es diu que són *semblants* si existeix una matriu P regular tal que $B = PAP^{-1}$. La transformació $A \to PAP^{-1}$ s'anomena *transformació de semblança* de matriu P. Les transformacions de semblança són especialment apropiades per a tractar el problema dels parells propis, ja que dues matrius semblants tenen els mateixos valors propis i x és un vector propi de B per al valor propi λ si i només si $P^{-1}x$ és vector propi de A per al mateix valor propi. Una matriu A s'anomena *diagonalitzable* si és semblant a una diagonal. És fàcil comprovar que si A és diagonalitzable, $A = PDP^{-1}$, amb D diagonal

i P regular aleshores els elements de la diagonal de D són els valors propis de A i les columnes de P són els vectors propis corresponents. Aquesta descomposició

$$A = PDP^{-1} \tag{5.1}$$

s'anomena *descomposició en valors propis*.

El *quocient de Rayleigh* es defineix com

$$\frac{x^T Ax}{\|x\|_2^2} \tag{5.2}$$

Es tracta d'una funció contínua en $\mathbb{R}^n \setminus \{0\}$, perquè és quocient de contínues, sense anul·larse el denominador. A més, si $x \neq 0$ és un vector propi de A associat al valor propi λ, aleshores el quocient de Rayleigh de x és

$$\frac{x^T \lambda x}{\|x\|_2^2} = \lambda.$$

Com a conseqüència d'aquest fet i de la continuïtat del quocient de Rayleigh tenim que, si x és aproximació d'un vector propi de A associat a λ, aleshores el quocient de Rayleigh de x aproxima λ.

Suposem que la matriu A $n \times n$ és diagonalitzable i siguen $\lambda_1, \ldots, \lambda_n$ els valors propis de A de manera que

$$|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|,$$

i $v_1, \ldots, v_n$ els vectors propis corresponents. Si $|\lambda_1| > |\lambda_2|$ aleshores λ_1 s'anomena *valor propi dominant*. Presentem el *mètode de la potència* com l'algorisme 1.

Triar $z^{(0)}$, $\|z^{(0)}\|_2 = 1$
for $i = 1, \ldots$
  $p^{(i)} = Az^{(i-1)}$
  $\lambda^{(i-1)} = (z^{(i-1)})^T p^{(i)}$
  $z^{(i)} = \frac{p^{(i)}}{\|p^{(i)}\|_2}$
end

Algorisme 1: Mètode de la potència. Hem omès un criteri de parada basat en $\lambda^{(i)}$.

Sota les hipòtesis que λ_1 és dominant i $z^{(0)} = \alpha_1 v_1 + \cdots + \alpha_n v_n$, $\alpha_1 \neq 0$, els fets fonamentals d'aquest algorisme són:

1. $z^{(i)} = \frac{A^i z^{(0)}}{\|A^i z^{(0)}\|_2}$
2. $\pm z^{(i)} \to$ vector propi associat a λ_1
3. $\lambda^{(i)}$ és el quocient de Rayleigh de $z^{(i)}$ ($\|z^{(i-1)}\|_2 = 1$), ja que $\lambda^{(i)} \to \lambda_1$ (les aproximacions d'un vector propi donen quocients de Rayleigh que són aproximacions del valor propi associat).
4. La convergència d'aquest métode és lineal, amb ratio de convergència $|\lambda_2|/|\lambda_1|$.

Si $|\lambda_1| \approx |\lambda_2|$ aleshores la convergència pot resultar extremadament lenta i donar un mètode poc eficaç. Tot seguit veiem modificacions per resoldre aquesta circumstància. La possibilitat que $\alpha_1 = 0$ es pot considerar poc freqüent, sobretot si els càlculs són inexactes.

El *mètode de la potència inversa* consisteix a aplicar el mètode de la potència a A^{-1}. Com a conseqüència, aquest mètode aproxima el valor propi de A de menor valor absolut: si suposem

$$|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_{n-1}| > |\lambda_n|$$

aleshores els valors propis de A^{-1} són:

$$|\lambda_1^{-1}| \leq |\lambda_2^{-1}| \leq \cdots \leq |\lambda_{n-1}^{-1}| < |\lambda_n^{-1}|,$$

i λ_n^{-1} és valor propi dominant de A^{-1}.

Mostrem el mètode de la potència inversa en l'algorisme 2. Hem tingut en compte que: $p = A^{-1}z$ si i només si p és solució del sistema $Ap = z$

Triar $z^{(0)}$, $\|z^{(0)}\|_2 = 1$
for $i = 1, \ldots$
  trobar $p^{(i)}$ solució de $Ap^{(i)} = z^{(i-1)}$
  $\lambda^{(i-1)} = (z^{(i-1)})^T p^{(i)}$
  $z^{(i)} = \frac{p^{(i)}}{\|p^{(i)}\|_2}$
end

Algorisme 2: Algorisme de la potència inversa. Hem omès un criteri de parada basat en $\lambda^{(i)}$.

La variable $\lambda^{(i-1)}$ aproxima $\frac{1}{\lambda_n}$ ja que $\frac{1}{\lambda^{(i-1)}}$ aproxima λ_n. La velocitat de convergència és $\frac{\lambda_n}{\lambda_{n-1}}$ Així doncs, la convegència serà ràpida si $|\lambda_{n-1}|$ i $|\lambda_n|$ estan ben separats.

Quant a la implementació, adoneu-vos que sols cal fer la factorització LU de A inicialment, per resoldre després els sistemes $Ap^{(i)} = z^{(i-1)}$ fent sols substitucions, la qual cosa té com a resultat un cost computacional molt inferior.

Si apliquem el mètode de la potència inversa a la matriu $A - \mu I$ aleshores obtindrem aproximacions al valor propi λ_i de A més a prop de μ (el qual correspon al valor propi $\lambda_i - \mu$ més petit de $A - \mu I$).

5.2 Aplicacions

5.2.1 Sistemes d'ODEs

La solució analítica de sistemes d'equacions diferencials lineals amb coeficients constants es pot obtenir utilitzant els valors i vectors propis de la matriu de coeficients.

Siguen $u_i = u_i(t)\ i = 1, \ldots, n$ les funcions desconegudes i siga

$$\begin{aligned} u_1'(t) &= a_{11}u_1(t) + \cdots + a_{1n}u_n(t) \\ &\vdots \\ u_n'(t) &= a_{n1}u_1(t) + \cdots + a_{nn}u_n(t) \\ u_i(0) &= \alpha_i, i = 1, \ldots, n \end{aligned} \tag{5.3}$$

un problema de valors inicials donat per un sistema d'equacions diferencials lineals de primer ordre i amb coeficients constants a_{ij} i pel coneixement dels valors inicial $u_i(0)$. Si anomenem A la matriu formada per aquests coeficients, aleshores el sistema anterior es pot escriure abreviadament com a

$$u'(t) = A\,u(t), \forall t, \tag{5.4}$$

entenent que la derivada d'una funció vectorial és la derivada de cadascuna de les seues components.

Suposem que A siga diagonalitzable i siga $A = PDP^{-1}$ una descomposició tal que D és diagonal, és a dir, D té els valors propis de A en la diagonal i les columnes de P són els vectors propis corresponents a aquests valors propis. Denotem $P^{-1} = (q_{ij})$ i fem el canvi de variables $v(t) = P^{-1}u(t)$, és a dir, la funció v_i és una certa combinació lineal de les funcions $u_1, \ldots, u_n$:

$$v_i(t) = q_{i1}u_1(t) + \cdots + q_{in}u_n(t), \forall t. \tag{5.5}$$

També tenim que

$$u(t) = Pv(t). \tag{5.6}$$

Derivem l'equació (5.5) respecte a la variable independent t:

$$v_i'(t) = q_{i1}u_1'(t) + \cdots + q_{in}u_n'(t), \forall t.$$

És a dir,

$$v'(t) = P^{-1}u'(t), \forall t$$

D'ací i de les relacions (5.4) i (5.6) obtenim:

$$v'(t) = P^{-1}Au(t) = P^{-1}APv(t) = Dv(t). \tag{5.7}$$

Com que D és diagonal, si anomenem $\lambda_i = D(i,i)$, aleshores (5.7) s'escriu com:

$$v_i'(t) = \lambda_i v_i(t). \tag{5.8}$$

Cadascuna de les equacions del sistema (5.8) sols depèn d'una variable, aleshores es poden resoldre independentment:

$$v_i(t) = v_i(0)e^{\lambda_i t}, \tag{5.9}$$

on $v_i(0) = q_{i1}u_1(0) + \cdots + q_{in}u_n(0)$ és conegut. D'ací i de (5.6):

$$u(t) = v_1(0)e^{\lambda_1 t}p_1 + \cdots + v_n(0)e^{\lambda_n t}p_n \tag{5.10}$$

és la solució del problema de valors inicials (5.3), on $p_i = P(:,i)$.

El comportament asimptòtic de la solució del sistema és estable si $Re(\lambda_i) \leq 0\ \forall i$ i inestable en cas contrari:

Suposem que tenim ordenats els valors propis de A tal que

$$Re(\lambda_1) \geq \cdots \geq Re(\lambda_n).$$

Podem reescriure (5.10) com:

$$u(t) = e^{\lambda_1 t}(v_1(0)p_1 + \cdots + v_n(0)e^{(\lambda_n - \lambda_1)t}p_n) = e^{\lambda_1 t}w(t)$$

Si prenem mòduls als dos membres d'aquesta relació:

$$|u_i(t)| = |e^{\lambda_1 t}||w_i(t)| = e^{Re(\lambda_1)t}|w_i(t)|. \tag{5.11}$$

D'altra banda, $|w_i(t)|$ és una quantitat fitada:

$$|w_i(t)| \leq |v_1(0)||p_{i1}| + \cdots + |v_n(0)||e^{(\lambda_n - \lambda_1)t}||p_{in}| \leq |v_1(0)||p_{i1}| + \cdots + |v_n(0)||p_{in}|, \tag{5.12}$$

ja que $|e^{(\lambda_i - \lambda_1)t}| = e^{Re(\lambda_i - \lambda_1)t} \leq 1$. Deduïm d'ací i de (5.11) que

$$\lim_{t\to\infty} u(t) = \begin{cases} 0(\text{estable}) & \text{si } Re(\lambda_1) < 0 \\ \infty(\text{inestable}) & \text{si } Re(\lambda_1) > 0 \text{ (sempre que } w(t) \not\to 0) \\ \text{oscil·lació} & \text{si } Re(\lambda_1) = 0 \end{cases}$$

Nota 5.1. Quan el sistema d'EDOs correspon al comportament dinàmic d'una estructura (un edifici, un pont, un avió, etc.) és crucial conèixer l'estabilitat del sistema (és a dir, el signe de les parts reals dels valors propis) ja que, en aquests casos, els sistemes inestables són ben poc desitjables i els oscil·latoris poden ser molests!

5.3 Condicionament

El problema de la determinació de valors propis múltiples és mal condicionat, tal com ens ho mostra l'exemple següent.

Considerem la matriu

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix},$$

la qual té com a valor propi l'1 (doble), i la pertorbació de A:

$$\begin{bmatrix} 1 & 1 \\ \varepsilon & 1 \end{bmatrix},$$

aleshores els valors propis són $1 \pm \sqrt{\varepsilon}$, amb la qual cosa una petita variació de ε en la matriu produeix una variació $\sqrt{\varepsilon}$ en els valors propis. La condició de la determinació d'aquest valor propi és:

$$\lim_{\varepsilon \to 0} \sup_{\|\Delta A\| = \varepsilon} \frac{\text{valor propi}(A + \Delta A) - \text{valor propi}(A)}{\|\Delta A\|} \geq \lim_{\varepsilon \to 0} \frac{\sqrt{\varepsilon}}{\varepsilon} = \infty.$$

En general, si λ és un valor propi de multiplicitat m d'una matriu A, aleshores hi ha pertorbacions de A de grandària ε que donen pertorbacions en λ aproximadament de $\varepsilon^{\frac{1}{m}}$, amb la qual cosa podem afirmar que la determinació de valors propis múltiples és un problema mal condicionat.

Lema 5.2. Si A és una matriu invertible i $\|\Delta A\| < \frac{1}{\|A^{-1}\|}$ aleshores $A + \Delta A$ és invertible.

Proposició 5.3. Siga A una matriu diagonalitzable, $A = PDP^{-1}$, amb D diagonal. Si λ és un valor propi de $A + E$ aleshores existeix un valor propi μ de A tal que $|\lambda - \mu| \leq \|E\|\kappa(P)$, on $\kappa(P) = \|P\|\|P^{-1}\|$ és la condició respecte de la norma $\|\cdot\| = \|\cdot\|_p$, $p \in [1, \infty]$.

Prova. Si λ és valor propi de A, podem prendre $\mu = \lambda$ i el resultat és clar en aquest cas. Suposem, per tant, que λ no és valor propi de A. Com que λ és valor propi de $A + E$, aleshores $A + E - \lambda I$ no és invertible. Multiplicant per P i per P^{-1}, tenim que la matriu

$$P^{-1}(A + E - \lambda I)P = D - \lambda I + P^{-1}EP \tag{5.13}$$

no és invertible. Com que λ no apareix en la diagonal de D, la matriu $D - \lambda I$ és invertible. Aplicant a (5.13) el lema 5.2 tenim que

$$\frac{1}{\|(D - \lambda I)^{-1}\|} \leq \|P^{-1}EP\| \leq \|P^{-1}\|\|E\|\|P\| = \kappa(P)\|E\|.$$

Com que

$$\|(D - \lambda I)^{-1}\| = \max_i(|\mu_i - \lambda|^{-1}) = \frac{1}{\min_i(|\mu_i - \lambda|)},$$

deduïm que

$$\min_i(|\mu_i - \lambda|) \leq \kappa(P)\|E\|.$$

Aquest resultat ens suggereix que la condició del càlcul dels valors propis està fitada per la condició de la matriu formada pels vectors propis; així doncs, com menor siga la condició d'aquesta matriu de vectors propis menor serà la condició del càlcul dels valors propis de la matriu. El cas extrem és quan la condició és 1. Aquest mínim l'assoleixen, per exemple, matrius ortogonals (en el cas complex, hermitianes) quan la norma és l'euclidiana. Com a conseqüència, matrius diagonalitzables per matrius ortogonals (o hermitianes) donen problemes de valors propis perfectament condicionats, i per tant ideals des del punt de vista numèric. Aquestes matrius s'anomenen matrius *normals*. Vegeu el problema 7 per a una caracterització d'aquestes matrius.

5.4 Matrius simètriques

La majoria dels algorismes i resultats que exposem en la resta d'aquest tema són vàlids per a matrius simètriques i, amb modificacions convenients, per a matrius no simètriques. Com que la simetria ens permetrà simplificacions importants, en la resta d'aquest tema suposarem que la matriu A és simètrica i donarem els algorismes i resultats sota aquesta suposició.

Els problemes de valors propis amb matrius simètriques són molt convenients des del punt de vista numèric, per dues raons almenys:

1. Els seus valors i vectors propis són reals, així doncs no cal utilitzar aritmètica complexa, amb el conseqüent estalvi de temps de computació i memòria.

2. Tota matriu simètrica és diagonalitzable per matrius ortogonals, de manera que és normal i, per tant, la determinació dels seus valors propis és un problema ben condicionat.

Un altre motiu menys aparent és que podem construir una transformació de semblança entre la matriu A simètrica i una matriu tridiagonal, amb la qual cosa el problema dels valors i vectors propis general queda reduït a matrius tridiagonals. Aquesta reducció significarà un estalvi crucial en la majoria dels procediments que segueixen, ja que permetrà passar de fer moltes iteracions amb cost per iteració $\mathcal{O}(n^3)$ a fer la reducció inicial, amb un cost $\mathcal{O}(n^3)$, i després fer les iteracions que calguen amb un cost $\mathcal{O}(n)$.

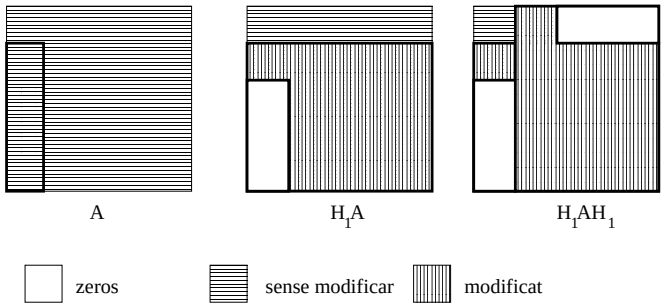
Podem trobar una transformació de Householder H_1 tal que

$$H_1 = \begin{bmatrix} 1 & 0 \\ 0 & \tilde{H}_1 \end{bmatrix}, \quad \tilde{H}_1 \begin{bmatrix} A_{21} \\ \vdots \\ A_{n1} \end{bmatrix} = \begin{bmatrix} A'_{21} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Si designem $B = H_1 A$, com que la primera fila de A no la modifica el producte per H_1, aleshores $B(1,:) = A(1,:) = A(:,1)^T$, per la qual cosa

$$B(1,:)H_1 = A(1,:)H_1 = (H_1 A(:,1))^T = \begin{bmatrix} A_{11} \\ A'_{21} \\ \vdots \\ 0 \end{bmatrix}^T = \begin{bmatrix} A_{11} & A'_{21} & \dots & 0 \end{bmatrix}$$

Cal adonar-se que el producte BH_1 sols modifica les columnes $2, \dots, n$; per tant, no modifiquem els zeros ja obtinguts en la primera columna. Esquemàticament, l'obtenció de la matriu transformada $H_1 A H_1$ ($H_1^T = H_1$) és la següent:



Si continuem aquest procediment de manera recursiva amb la submatriu A_1 formada per les $n-1$ últimes files i columnes de H_1AH_1, aleshores obtenim una matriu tridiagonal semblant a A. El cost computacional d'aquest algorisme és aproximadament de $\frac{8}{3}n^3$ (vegeu el problema 2).

5.5 Iteració del quocient de Rayleigh

La *iteració del quocient de Rayleigh* s'assembla al mètode de la potència inversa aplicat a la matriu $A-\mu I$ perquè aproxima el valor propi de A més a prop de μ. La diferència és que, en la iteració del quocient de Rayleigh, μ varia d'una iteració a l'altra

Triar $z^{(0)}$, $\|z^{(0)}\|_2 = 1$, $\lambda^{(0)} = (z^{(0)})^T A z^{(0)}$
for $i = 1, \ldots$
  trobar $p^{(i)}$ solució de $(A - \lambda^{(i-1)} I)p^{(i)} = z^{(i-1)}$
  $z^{(i)} = \frac{p^{(i)}}{\|p^{(i)}\|_2}$
  $\lambda^{(i)} = (z^{(i)})^T A z^{(i)}$
end

Algorisme 3: Algorisme de la iteració del quocient de Rayleigh. Hem omès un criteri de parada basat en $\lambda^{(i)}$.

No podem utilitzar la mateixa estratègia que en la potència inversa de calcular sols la descomposició LU inicialment, ja que ací la matriu de coeficients és $(A - \lambda^{(i-1)} I)$, la qual varia d'iteració en iteració. Per qüestions d'eficiència, és convenient que A siga tridiagonal per aplicar aquest mètode, perquè així la matriu $(A - \lambda^{(i-1)} I)$ és tridiagonal també, i el cost de la resolució del sistema és $\mathcal{O}(n)$.

La iteració del quocient de Rayleigh té unes propietats de convergència excel·lents:

1. Convergeix per a quasi tota aproximació inicial $z^{(0)}$.
2. La convergència és cúbica, és a dir, si $\lambda^{(i)} \to \lambda$ aleshores, per a i suficientment gran, $|\lambda - \lambda^{(i+1)}| = \mathcal{O}(|\lambda - \lambda^{(i)}|^3)$

La convergència cúbica és extremadament ràpida, ja que a partir d'una certa iteració el nombre de xifres exactes en $\lambda^{(*)}$ es triplica en cada iteració!

5.6 Algorisme QR

L'algorisme QR, com el seu nom indica, utilitza la descomposició $A = QR$, on Q és ortogonal i R és triangular superior, en una iteració matricial que convergeix (sota condicions adequades) en una matriu

diagonal semblant a la original i, per tant, amb els valors propis en la diagonal. L'algorisme 4 mostra la versió bàsica, la qual és sorprenentment senzilla.

$A_0 = A$
for $i = 0, \ldots$
  $A_i = Q_i R_i$ (descomposició QR)
  $A_{i+1} = R_i Q_i$
end

Algorisme 4: Algorisme QR.

Com que $Q_i^T A_i Q_i = Q_i^T Q_i R_i Q_i = R_i Q_i = A_{i+1}$, A_i i A_{i+1} són semblants. Inductivament justifiquem que

$$\overline{Q}_i^T A \overline{Q}_i = \overline{Q}_i^T A_0 \overline{Q}_i = A_i, \tag{5.14}$$

on $\overline{Q}_i = Q_0 \ldots Q_{i-1}$, per tant A i A_i són semblants, amb matriu de pas $\overline{Q}_i$. Si designem $\overline{R}_i = R_{i-1} \ldots R_0$, aleshores

$$A^i = \overline{Q}_i \overline{R}_i \tag{5.15}$$

és una descomposició QR de A^i: si $i = 1$ el resultat és $A = Q_0 R_0$, la qual cosa és certa. Si $A^i = \overline{Q}_i \overline{R}_i$, aleshores, utilitzant la relació $A_j Q_j = Q_j A_{j+1}$ $(j = 0, \ldots, i-1)$, obtenim

$$\begin{aligned} A^{i+1} &= A A^i = A_0 Q_0 \ldots Q_{i-1} \overline{R}_i \\ &= Q_0 A_1 Q_1 \ldots Q_{i-1} \overline{R}_i = \cdots = Q_0 \ldots Q_{i-1} A_i \overline{R}_i \\ &= \ldots Q_0 \ldots Q_{i-1} Q_i R_i \overline{R}_i = \overline{Q}_{i+1} \overline{R}_{i+1} \end{aligned}$$

5.6.1 Algorisme QR per a matrius tridiagonals

El cost computacional del càlcul d'una descomposició QR d'una matriu genèrica $n \times n$ és $\mathcal{O}(n^3)$, ja que, si A és una matriu $n \times n$ simètrica genèrica aleshores el cost computacional de cada iteració de l'algorisme és $\mathcal{O}(n^3)$, la qual cosa suposa un cost prohibitiu si hem de fer moltes iteracions. Tammateix si la matriu A és tridiagonal, aleshores el resultat següent ens assegura que A_i és tridiagonal, per a qualsevol i. Veurem també que la descomposició QR d'una matriu simètrica tridiagonal es pot fer en $\mathcal{O}(n)$ operacions, ja que el cost per iteració pot reduir-se a $\mathcal{O}(n)$ quan la matriu és tridiagonal.

Lema 5.4. Si A és una matriu tridiagonal simètrica i invertible i $A = QR$ és una descomposició QR aleshores $A_1 = RQ$ també és tridiagonal i simètrica.

Prova. Si A és invertible, aleshores R també ho és, R^{-1} és triangular superior i $Q = AR^{-1}$. Si apliquem el problema 3 amb $X_1 = A$, $X_2 = R^{-1}$, $l_1 = -1$, $u_1 = 1$, $l_2 = 0$, $u_2 = n-1$ obtenim que $Q(i,j) = 0$ si $j - i < -1$. Si tornem a aplicar aquest problema a $A_1 = RQ$, amb $X_1 = Q$ i $X_2 = R$ $l_1 = -1$, $u_1 = n-1$, $l_2 = 0$, $u_2 = n-1$, obtenim que $A_1(i,j) = 0$ si $j - i < -1$. D'altra banda $A_1 = Q^T QRQ = Q^T AQ$, doncs $A_1^T = Q^T A^T Q = A_1$, d'on deduïm que també es verifica $A_1(i,j) = 0$ si $j - i > 1$, és a dir, A_1 és tridiagonal.

Per calcular la descomposició QR d'una matriu tridiagonal és convenient utilitzar transformacions de Givens, ja que hi ha exactament una entrada no nul·la en cada columna per sota la diagonal. Vegem com

actua un pas d'aquest algorisme per a una matriu 4×4

$$A = \begin{bmatrix} a_1 & \times & 0 & 0 \\ b_1 & \times & \times & 0 \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix}$$

Per aconseguir un zero on és b_1 utilitzem la transformació de Givens

$$G_1 = \begin{bmatrix} \hat{a}_1 & \hat{b}_1 & 0 & 0 \\ -\hat{b}_1 & \hat{a}_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{on} \quad \hat{a}_1 = \frac{a_1}{\sqrt{a_1^2 + b_1^2}} \quad \text{i} \quad \hat{b}_1 = \frac{b_1}{\sqrt{a_1^2 + b_1^2}}.$$

$$G_1 A = \begin{bmatrix} \hat{a}_1 & \hat{b}_1 & 0 & 0 \\ -\hat{b}_1 & \hat{a}_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{a}_1 & \hat{b}_1 & 0 & 0 \\ -\hat{b}_1 & \hat{a}_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \times & \times & * & 0 \\ 0 & a_2 & \times & 0 \\ 0 & b_2 & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix}$$

Indiquem amb $*$ una nova entrada creada on hi havia un zero. Per anul·lar l'entrada b_2 utilitzem la transformació de Givens

$$G_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \hat{a}_2 & \hat{b}_2 & 0 \\ 0 & -\hat{b}_2 & \hat{a}_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{on} \quad \hat{a}_2 = \frac{a_2}{\sqrt{a_2^2 + b_2^2}} \quad \text{i} \quad \hat{b}_2 = \frac{b_2}{\sqrt{a_2^2 + b_2^2}}.$$

$$G_2 G_1 A = \begin{bmatrix} \times & \times & * & 0 \\ 0 & \times & \times & * \\ 0 & 0 & a_3 & \times \\ 0 & 0 & b_3 & \times \end{bmatrix}$$

La transformació de Givens per anul·lar l'entrada b_3 és

$$G_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \hat{a}_3 & \hat{b}_3 \\ 0 & 0 & -\hat{b}_3 & \hat{a}_3 \end{bmatrix} \quad \text{on} \quad \hat{a}_3 = \frac{a_3}{\sqrt{a_3^2 + b_3^2}} \quad \text{i} \quad \hat{b}_3 = \frac{b_3}{\sqrt{a_3^2 + b_3^2}}.$$

Deduïm aleshores que

$$G_3 G_2 G_1 A = \begin{bmatrix} \times & \times & * & 0 \\ 0 & \times & \times & * \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix} = R,$$

d'on $A = (G_1^T G_2^T G_3^T) R$ és una descomposició QR, amb $Q = G_1^T G_2^T G_3^T$.

$$A_1 = RQ = \begin{bmatrix} \times & \times & * & 0 \\ 0 & \times & \times & * \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix} \begin{bmatrix} \hat{a}_1 & -\hat{b}1 & 0 & 0 \\ \hat{b}1 & \hat{a}_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \hat{a}_2 & -\hat{b}2 & 0 \\ 0 & \hat{b}2 & \hat{a}_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \hat{a}_3 & -\hat{b}3 \\ 0 & 0 & \hat{b}3 & \hat{a}_3 \end{bmatrix} = \begin{bmatrix} \times & \times & * & * \\ \times & \times & \times & * \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix}$$

Com que A_1 és simètrica, aleshores les entrades marcades amb $*$ són nul·les.

En general, per a matrius $n \times n$ aquest algorisme calcula $n - 1$ transformacions de Givens que s'apliquen tant per l'esquerra com per la dreta. Cada transformació té un cost computacional de $\mathcal{O}(1)$ operacions, per a una total de $\mathcal{O}(n)$ operacions.

5.6.2 Convergència de la iteració QR per a matrius tridiagonals

Si $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$ són els valors propis ordenats de major a menor valor absolut i la matriu P formada pels vectors propis corresponents satisfà que $P(1:j, 1:j)$ és invertible $j = 1, \ldots, n$, aleshores

1. $A_i \to D = \text{diag}(\lambda_1, \ldots, \lambda_n)$, $\overline{Q}_i \to P$.
2. per a i suficientment gran

$$\begin{aligned} |A_{i+1}(j+1, j)/A_i(j+1, j)| &\approx |\lambda_{j+1}/\lambda_j|, j = 1, \ldots, n-1 \\ |A_{i+1}(j, j) - \lambda_j|/|A_i(j, j) - \lambda_j| &\approx |\lambda_{j+1}/\lambda_j|, j = 1, \ldots, n \end{aligned} \tag{5.16}$$

La possibilitat que no es verifiquen les hipòtesis d'aquest resultat és molt baixa, especialment en aritmètica inexacta. Tammateix, si hi ha dos valors propis amb valors absoluts molt similars, aleshores la relació (5.16) indica que la convergència serà extremadament lenta.

5.6.3 Deflació

Si tenim una matriu simètrica que es descompon d'aquesta manera:

$$A = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix} \quad A_1 = A(1:m, 1:m), A_2 = A(m+1:n, m+1:n) \tag{5.17}$$

i coneixem descomposicions en valors propis de A_1 i A_2:

$$P_1^T A_1 P_1 = D_1 \quad P_2^T A_2 P_2 = D_2$$

amb P_i ortogonals i D_i diagonals ($i = 1, 2$), aleshores

$$\begin{bmatrix} P_1 & 0 \\ 0 & P_2 \end{bmatrix}^T \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix} \begin{bmatrix} P_1 & 0 \\ 0 & P_2 \end{bmatrix} = \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix}.$$

Deduïm que, si tenim una descomposició de la matriu A com en (5.17), aleshores el problema de trobar els valors i vectors propis es redueix a trobar els valors i vectors propis de dues matrius de dimensió inferior, la qual cosa significa una simplificació important.

Des d'un punt de vista pràctic, una descomposició com (5.17) és rar que es done exactament. Si la matriu A és tridiagonal, segons (5.16), el que sol passar en alguna iteració de l'algorisme QR és que

$$|A_i(m+1, m)| \approx \varepsilon_{\text{mach}} \|A\|$$

Si substituïm aquesta entrada per zero estem en la situació simplificadora que hem descrit abans. Des del punt de vista numèric, l'error que cometem en aquest pas és de l'ordre $\varepsilon_{\text{mach}}\|A\|$, comparable al que es produeix en els càlculs en punt flotant i no ampliat per les transformacions posterior perquè aquestes són ortogonals. Aquesta tècnica s'anomena *deflació*. L'algorisme 5 implementa recursivament aquesta tècnica.[1]

Vegem que la deflació no influeix en la estabilitat cap enrere: si designem la matriu resultant d'anul·lar $A_i(m, m+1)$ i $A_i(m+1, m)$:

$$\tilde{A}_i = \begin{bmatrix} A_i(1:m, 1:m) & 0 \\ 0 & A_i(m+1:n, m+1:n) \end{bmatrix},$$

[1] Adoneu-vos que la recursivitat es pot evitar de manera relativament senzilla.

la qual verifica $\tilde{A}_i = A_i + \Delta A_i$, $\|\Delta A_i\| = \mathcal{O}(\varepsilon_{\text{mach}}\|A\|)$ i calculem descomposicions en valors propis dels blocs de la diagonal:

$$A_i(1:m,1:m) = Z_1 D_1 Z_1^T, A_i(m+1:n,m+1:n) = Z_2 D_2 Z_2^T,$$

amb Z_i matrius ortogonals i D_i matrius diagonals ($i = 1, 2$), aleshores podem construir aproximadament una descomposició en valors propis de A:

$$A = \overline{Q}_i A_i \overline{Q}_i^T = \overline{Q}_i(\tilde{A}_i + \Delta A_i)\overline{Q}_i^T = \underbrace{\overline{Q}_i \begin{bmatrix} Z_1 & 0 \\ 0 & Z_2 \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix}}_{D} \underbrace{(\overline{Q}_i \begin{bmatrix} Z_1 & 0 \\ 0 & Z_2 \end{bmatrix})^T}_{U^T} + \overline{Q}_i \Delta A_i \overline{Q}_i^T$$

Si anomenen $\Delta A = -\overline{Q}_i \Delta A_i \overline{Q}_i^T$ tenim que $\|\Delta A\| = \|\Delta A_i\| = \mathcal{O}(\varepsilon_{\text{mach}}\|A\|)$, perquè $\overline{Q}_i$ és ortogonal, i $A + \Delta A = \overline{Q}_i D \overline{Q}_i^T$, de manera que calculem la descomposició en valors propis d'una matriu $A + \Delta A$ que és una pertorbació relativa de A de l'ordre de $\varepsilon_{\text{mach}}$, és a dir, que l'algorisme amb deflació calcula la descomposició en valors propis de A d'una manera estable cap enrere.

```
function [D, P] = qr(A)
A_0 = A
P = I
for i = 0, ...
  A_i = Q_i R_i (descomposició QR)
  A_{i+1} = R_i Q_i
  P = P Q_i
  for m = n - 1 : -1 : 1
    if |A_{i+1}(m+1, m)| ≤ ε
      A_{i+1}(m+1, m) = A_{i+1}(m, m+1) = 0
      [D_1, P_1] = qr(A(1 : m, 1 : m))
      [D_2, P_2] = qr(A(m+1 : n, m+1 : n))
      D = [D_1, 0; 0, D_2]
      P = P[P_1, 0; 0, P_2]
      return
    end
  end
end
```

Algorisme 5: Algorisme QR amb deflació per a matrius tridiagonals.

La utilització de la deflació permet treballar amb matrius tridiagonals *irreductibles*, és a dir, amb matrius tridiagonals sense zeros en la subdiagonal (i superdiagonal, per simetria), és a dir

$$A(i+1,i) \neq 0 \quad i = 1, \ldots, n-1.$$

La suposició que A és irreductible permet resoldre el problema de la unicitat de la descomposició QR: podem assegurar que la descomposició QR és única sols quan la matriu és invertible. El problema és que no podem assegurar la invertibilitat de A_i. Però si A_i és tridiagonal irreductible i no invertible, el que sí que podem assegurar és que les primeres $n-1$ columnes són linealment independents, ja que la submatriu

$A_i(2:n, 1:n-1)$ és $(n-1)\times(n-1)$ i és invertible, perquè és triangular superior i té la subdiagonal de A_i com a diagonal, ja que rang$(A_i) = n-1$. Si

$$A_i = Q_i R_i \tag{5.18}$$

és una descomposició QR de A_i, aleshores, pel problema 4, $Q_i(:, 1:n-1)$ i R_i són unics i $R_i(n,:) = 0$. D'ací que

$$A_{i+1} = R_i Q_i = \begin{bmatrix} R_i(1:n-1,:)Q_i(:,1:n-1) & * \\ 0 & 0 \end{bmatrix}.$$

Com que A_{i+1} és simètrica, tenim que

$$A_{i+1} = \begin{bmatrix} R_i(1:n-1,:)Q_i(:,1:n-1) & 0 \\ 0 & 0 \end{bmatrix}, \tag{5.19}$$

ja que està unívocament determinada per A_i, perquè l'únic que no és unívocament determinat per A_i és $Q_i(:,n)$ i aquest vector no apareix en (5.19).

5.7 Translacions

El resultat (5.16) sobre la convergència del mètode QR ens indica que aquesta convergència és lenta si hi ha dos valors propis amb valors absoluts semblants. Suposem que un d'aquests valors propis semblants és λ_j. Si μ és una bona aproximació a λ_j, aleshores el menor valor propi de $A-\mu I$ serà $\lambda_j - \mu$ (els valors propis de $A-\mu I$ són $\lambda-\mu$, on λ és valors propi de A). Si el segon menor valor propi de $A-\mu I$ és $\lambda_l - \mu$ aleshores $R = (\lambda_j-\mu)/(\lambda_l-\mu) \ll 1$ és petit, si suposem que μ és molt millor aproximació a λ_j que a cap altre valor propi. En definitiva, si apliquem la iteració QR a $B = A-\mu I$ tindrem que

$$|B_i(n,n) - (\lambda_j - \mu)| = \mathcal{O}(R^i) \quad |B_i(n,n-1)| = \mathcal{O}(R^i).$$

És a dir,

$$B_i(n,n) + \mu \to \lambda_j \quad B_i(n,n-1) \to 0 \quad i \to \infty$$

molt ràpidament. Es pot comprovar que la iteració següent:

$$\begin{aligned} A_i - \mu I &= Q_i R_i, \\ A_{i+1} &= R_i Q_i + \mu I \end{aligned} \tag{5.20}$$

és equivalent a aplicar la iteració QR a $B = A-\mu I$ i després sumar μ als valors propis obtinguts. Deduïm, doncs, que la convergència a λ_j aquesta iteració serà molt més ràpida en aquesta iteració que en el mètode original. La tècnica d'utilitzar aquest desplaçaments de A a $A-\mu I$ s'anomena *traslació (shift)*.

La qüestió fonamental és com triar μ tal que $\mu \approx \lambda_j$, sense conèixer λ_j. Suposem que la iteració QR ha arribat a un moment en què la matriu A_i té l'element $A_i(n,n-1) = \varepsilon$ petit. Triem $\mu = A_i(n,n)$ i siga C la matriu resultant d'anul·lar les entrades $(n,n-1)$ i $(n-1,n)$ de A_i, $\|C - A_i\|_2 = \varepsilon$. Com que μ és valor propi de C i C és una pertorbació de A_i, aleshores la proposició 5.3 ens assegura que A_i té un valor propi λ (com que A i A_i són semblants, també és valor propi de A) tal que $|\lambda-\mu| \le \varepsilon$ (la matriu que diagonalitza A_i és ortogonal, en ser A_i simètrica, per tant té condició 1). En definitiva, $\mu = A_i(n,n)$ és una bona aproximació d'un valor propi de A (probablement el més petit) quan $A_i(n,n-1)$ és petit. Aquesta reflexió ens permet formular l'algorisme 6, el qual consisteix en la iteració (5.20), en la qual μ varia d'una iteració a l'altra, $\mu_i = A_i(n,n)$.

Hi ha altres possibles tries de translacions, però per a matrius simètriques la que hem presentat ací és molt satisfactòria.

function $[D, P] = \text{qr}(A)$
$A_0 = A$
$P = I$
for $i = 0, \ldots$
  $\mu_i = A_i(n, n)$
  $A_i - \mu_i I = Q_i R_i$ (descomposició QR)
  $A_{i+1} = R_i Q_i + \mu_i I$
  $P = PQ_i$
end

Algorisme 6: Algorisme QR amb translacions per a matrius tridiagonals.

Anàlogament a (5.14) es pot comprovar que A i A_i són semblants amb matriu de pas

$$\overline{Q}_i = Q_0 \ldots Q_{i-1}. \tag{5.21}$$

També, anàlogament a (5.15), es pot provar que

$$(A - \mu_0 I) \ldots (A - \mu_{i-1} I) = \overline{Q}_i \overline{R}_i, \tag{5.22}$$

on $\overline{R}_i = R_{i-1} \ldots R_0$. Es pot comprovar que la iteració plantejada en l'algorisme 6 està relacionada amb la iteració del quocient de Rayleigh (vegeu el problema 5) i la velocitat de convergència de $A_i(n, n)$ a un valor propi (probablement el menor, encara que ara no tenim garanties de l'ordre) és cúbica. Si combinem aquest algorisme amb deflació, obtindrem un mètode extremadament ràpid, el qual implementem en l'algorisme 7.

5.8 Problemes

Problema 1. En el context de la iteració de Rayleigh, si denotem

$$y^{(i)} = (A - \lambda^{(i-1)})^{-1} \ldots (A - \lambda^{(0)})^{-1} z^{(0)}, \tag{5.23}$$

demostreu que $z^{(i)} = y^{(i)} / \|y^{(i)}\|_2$.

Problema 2.

1. Escriviu en matlab l'algorisme de reducció d'una matriu simètrica a tridiagonal utilitzant transformacions de Householder: `function [D, P]=tridiag(A)`

2. Comproveu que el nombre d'operacions requerides per aquest algorisme és aproximadament de $\frac{8}{3}n^3$.

Problema 3. Siga $l_k \leq u_k$ $k = 1, 2$ sencers i X_1 i X_2 matrius $n \times n$ tal que $X_k(i, j) = 0$ si $j - i < l_k$ o $j - i > u_k$ $k = 1, 2$. Aleshores $X_1 X_2(i, j) = 0$ si $j - i < l_1 + l_2$ o $j - i > u_1 + u_2$.

El que ens indica aquest problema és:

> la suma de les amplàries de banda inferiors (respectivament, superiors) dels factors dóna l'amplària de banda inferior (respectivament, superior) del producte.

function $[D, P] = \text{qr}(A)$

$[T, P] = \text{tridiag}(A)$ % $P^T AP = T$ tridiagonal, P ortogonal
$A_0 = T$
for $i = 0, \ldots$
  $\mu_i = A_i(n, n)$
  $A_i - \mu_i I = Q_i R_i$ (descomposició QR)
  $A_{i+1} = R_i Q_i + \mu_i I$
  $P = PQ_i$
  for $m = n - 1 : -1 : 1$
    if $|A_{i+1}(m + 1, m)| \leq \varepsilon$
      $A_{i+1}(m + 1, m) = 0$
      $A_{i+1}(m, m + 1) = 0$
      $[D_1, P_1] = \text{qr}(A(1 : m, 1 : m))$
      $[D_2, P_2] = \text{qr}(A(m + 1 : n, m + 1 : n))$
      $D = [D_1, 0; 0, D_2]$
      $P = P[P_1, 0; 0, P_2]$
      return
    end
  end
end

Algorisme 7: Algorisme QR pràctic per resoldre problemes de valors i vectors propis per a matrius simètriques.

Problema 4. Si A és una matriu $m \times n$ amb $\text{rang}(A) = n$ i $Q_1 R_1 = Q_2 R_2 = A$ són dues descomposicions QR de A, amb les columnes de les matrius $m \times n$ Q_i ortonormals ($Q_i^T Q_i = I_n$) i amb R_i triangular superior amb entrades positives en la diagonal ($i = 1, 2$), aleshores $Q_1 = Q_2$ i $R_1 = R_2$.

Problema 5. Siga $\overline{Q}_i$ la matriu donada en (5.21), obtinguda en l'algorisme 6. Demostreu que

$$\overline{Q}_i(:, n) = \pm z^{(i)},$$

on $z^{(i)}$ s'obté amb la iteració del quocient de Rayleigh amb $z^{(0)} = e_n$.

Pista:

1. demostreu, a partir de (5.22) i la simetria de A, que

$$(A - \mu_{i-1} I)^{-1} \ldots (A - \mu_0 I)^{-1} = \overline{Q}_i \overline{R}_i^{-T},$$

   i, d'ací, que

$$\pm \overline{Q}_i(:, n) = \tilde{y}^{(i)} / \|\tilde{y}^{(i)}\|_2, \tag{5.24}$$

   on

$$\tilde{y}^{(i)} = (A - \mu_{i-1} I)^{-1} \ldots (A - \mu_0 I)^{-1} e_n.$$

2. utilitzeu inducció sobre i per obtenir a partir del problema 1 i (5.24) que $\mu_i = \lambda^{(i)}$ i, d'ací, $\tilde{y}^{(i)} = y^{(i)}$ de (5.23).

Problema 6. Demostreu per inducció sobre la dimensió de la matriu A amb entrades complexes que existeixen una matriu unitària P i una matriu triangular superior complexa T tals que

$$A = PTP^*, \tag{5.25}$$

on la conjugada transposada és $P^* = P^{-1}$, perquè és unitària. La descomposició (5.25) s'anomena *descomposició de Schur* de A i és la base dels algorismes per al càlcul de vectors i valors propis en matrius no simètriques, perquè es pot obtenir a partir de A amb transformacions unitàries (ortogonals complexes).

Problema 7. Demostreu que una matriu complexa A és normal si i només si $A^*A = AA^*$.

Pista: la implicació complicada és que $A^*A = AA^*$ dóna que A és normal, és a dir, que existeixen una matriu unitària P i una matriu diagonal D tals que $A = PDP^*$. Per provar això, utilitzeu l'exercici 6 per concloure que $A = PTP^*$ i que, per hipòtesi, T i T^* commuten. Demostreu després que si T és triangular superior i $T^*T = TT^*$ aleshores T és diagonal.

Tema 6

La descomposició SVD

6.1 Existència i interpretació geomètrica de la SVD

Les matrius simètriques es poden caracteritzar com les matrius ortogonalment semblants a matrius diagonals, ja que les transformacions lineals amb aquest tipus de matriu actuen d'una manera molt senzilla en algun sistema ortonormal. Concretament, si A és simètrica, aleshores existeixen una matriu ortogonal P (formada per vectors propis) i una matriu diagonal D (amb els valors propis corresponents en la diagonal) tal que

$$D = P^T AP. \tag{6.1}$$

Si x és un vector que té coordenades $(x_1, \ldots, x_n)$ en la base ortonormal $\{P(:,1), \ldots, P(:,n)\}$, aleshores

$$x = x_1 P(:,1) + \ldots x_n P(:,n) = P \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

d'on deduïm que l'acció de A sobre x, $y = Ax$ té coordenades en la base anterior:

$$P^T y = P^T Ax = P^T AP \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = D \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} D(1,1)x_1 \\ \vdots \\ D(n,n)x_n \end{bmatrix}.$$

És a dir, la transformació lineal que té matriu A en la base canònica actua en la base ortonormal $\{P(:,1), \ldots, P(:,n)\}$ allargant o escurçant les coordenades dels vectors en aquesta base. Tenir una descripció d'aquest tipus ens permet simplificar molts problemes que involucren la matriu A.

Amb molta probabilitat el resultat anàleg per a matrius generals, no simètriques i fins i tot no quadrades, no ens resulta familiar. Si A és una matriu $m \times n$, el resultat conegut més a prop de (6.1) és que existeixen P, Q regulars tal que

$$PAQ = \begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix}. \tag{6.2}$$

Aquest resultat no sol ser d'utilitat numèrica, perquè l'algorisme per obtenir (6.2) no és estable cap enrere, perquè no són ortogonals les transformacions que donen P i Q. La novetat que aporta la *descomposició en valors singulars* (*singular value decomposition, SVD*) és que P i Q es poden triar ortogonals (però els elements de la diagonal ja no seran uns), amb la qual cosa l'estabilitat cap endarrere d'algorismes basats en aquesta descomposició està garantida.

Teorema 6.1. Siga A una matriu $m \times n$, aleshores existeixen matrius ortogonals U $m \times m$ i V $n \times n$ i una matriu diagonal Σ $m \times n$ amb elements no negatius tal que

$$A = U\Sigma V^T. \tag{6.3}$$

Prova. La SVD d'una matriu A $m \times n$ es pot obtenir a partir d'una descomposició en valors i vectors propis de la matriu simètrica A^TA: si V és una matriu ortogonal amb vectors propis en les columnes, aleshores:

$$V^TA^TAV = D = \text{diag}(\lambda_1, \ldots, \lambda_r, 0, \ldots, 0), \lambda_i > 0 \tag{6.4}$$

on $r = \text{rang}(A^TA) = \text{rang}(A)$. Si anomenem $X = AV$, aleshores (6.4) queda com

$$X^TX = \text{diag}(\lambda_1, \ldots, \lambda_r, 0, \ldots, 0), \tag{6.5}$$

la que les r primeres columnes de X són ortogonals amb norma $\sqrt{\lambda_i}$ i les $n - r$ últimes són 0. Si dividim les columnes $1, \ldots, r$ per la seua norma $\sqrt{\lambda_i}$, aleshores obtenim una matriu U_1 tal que $U_1^TU_1 = I_r$ i tal que

$$X = [U_1\text{diag}(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_r})\ 0] \quad 0 \text{ matriu de zeros } n \times (n - r). \tag{6.6}$$

Si U és una matriu ortogonal tal que $U = [U_1\ *]$ i designem $\Sigma = \text{diag}(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_r}, 0, \ldots, 0)$ matriu diagonal $m \times n$, aleshores

$$U\Sigma = [U_1\ *] \begin{bmatrix} \text{diag}(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_r}) & 0 \\ 0 & 0 \end{bmatrix} = [U_1\text{diag}(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_r})\ 0] = X,$$

ja que podem formar una SVD de A:

$$AV = X = U\Sigma \Rightarrow A = U\Sigma V^T. \tag{6.7}$$

Exemple 6.2. Considerem la matriu

$$A = \begin{bmatrix} 4 & 4 \\ 1 & -1 \end{bmatrix}$$

i calculem, segons la demostració anterior, una SVD de A.

$$A^TA = \begin{bmatrix} 17 & 15 \\ 15 & 17 \end{bmatrix}$$

Aleshores els valors propis són $2, 32$, la matriu Σ és

$$\Sigma = \begin{bmatrix} \sqrt{32} & 0 \\ 0 & \sqrt{2} \end{bmatrix}$$

i la matriu ortogonal V formada pels vectors propis de A^TA és

$$V = \begin{bmatrix} \sqrt{2}/2 & \sqrt{2}/2 \\ \sqrt{2}/2 & -\sqrt{2}/2 \end{bmatrix}.$$

Per calcular U, calculem primer

$$X = AV = \begin{bmatrix} \sqrt{32} & 0 \\ 0 & \sqrt{2} \end{bmatrix}$$

Si dividim la primera columna de X per $\sqrt{32}$ i la segona per $\sqrt{2}$, tenim

$$U = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Aleshores es comprova que

$$U\Sigma V^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{32} & 0 \\ 0 & \sqrt{2} \end{bmatrix} \begin{bmatrix} \sqrt{2}/2 & \sqrt{2}/2 \\ \sqrt{2}/2 & -\sqrt{2}/2 \end{bmatrix} = \begin{bmatrix} 4 & 4 \\ 1 & -1 \end{bmatrix} = A$$

Exemple 6.3. Considerem la matriu

$$A = \begin{bmatrix} 3 & 6 \\ 2 & 4 \\ 4 & 8 \end{bmatrix}$$

i calculem, segons la demostració anterior, una SVD de A.

$$A^T A = \begin{bmatrix} 29 & 59 \\ 58 & 116 \end{bmatrix}$$

Aleshores els valors propis són $145, 0$, la matriu Σ és

$$\Sigma = \begin{bmatrix} \sqrt{145} & 0 \\ 0 & 0 \end{bmatrix}$$

i la matriu ortogonal V formada pels vectors propis de $A^T A$ és

$$V = \frac{1}{\sqrt{5}} \begin{bmatrix} 1 & -2 \\ 2 & 1 \end{bmatrix}.$$

Per calcular U, calculem primer

$$X = AV = \frac{1}{\sqrt{5}} \begin{bmatrix} 15 & 0 \\ 10 & 0 \\ 20 & 0 \end{bmatrix}$$

Si dividim la primera columna de X per $\sqrt{145}$, tenim

$$U_1 = \frac{1}{\sqrt{5 * 145}} \begin{bmatrix} 15 \\ 10 \\ 20 \end{bmatrix} = \frac{1}{5\sqrt{29}} \begin{bmatrix} 15 \\ 10 \\ 20 \end{bmatrix}$$

Si U és una matriu tal que $U = [U_1 *]$,aleshores es comprova que

$$U\Sigma V^T = \begin{bmatrix} 3 & 6 \\ 2 & 4 \\ 4 & 8 \end{bmatrix} = A$$

Aquesta descomposició és una *descomposició en valors singulars* de A (no és única). Els elements de la diagonal de Σ s'anomenen els *valors singulars* de A. Utilitzarem com a notació

$$\begin{aligned} U &= \begin{bmatrix} u_1 & \dots & u_m \end{bmatrix} \\ V &= \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix} \\ \Sigma &= \text{diag}(\sigma_1, \dots, \sigma_r, \dots, 0) \quad \sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0 \end{aligned}$$

on r és el rang de A (vegeu l'exercici 1 per justificar que Σ es pot escriure així).

Com que la SVD es pot escriure també com

$$AV = U\Sigma V^T V = U\Sigma$$

aleshores la interpretació vectorial de la SVD és

$$\begin{aligned} Av_i &= \sigma_i u_i, \quad i = 1, \ldots, r \\ Av_i &= 0, \quad i = r+1, \ldots, n \end{aligned}$$

Podem deduir d'ací que $N(A) = \langle v_{r+1}, \ldots, v_n \rangle$ i Im$A = \langle u_1, \ldots, u_r \rangle$.

El resultat anterior ens permet construir una SVD de A a partir d'una descomposició de $A^T A$ amb valors i vectors propis. Recíprocament, una SVD de A permet obtenir una descomposició de $A^T A$ amb valors i vectors propis:

$$AA^T = U\Sigma(V^T V)\Sigma^T U^T = U\Sigma\Sigma^T U^T = UDU^T \Rightarrow D = U^T(AA^T)U, \tag{6.8}$$

on $D = \Sigma\Sigma^T$ és una matriu diagonal amb $D(i,i) = \sigma_i^2$. Deduïm d'aquesta equivalència que un algorisme directe per obtenir la SVD d'una matriu A donaria un algorisme directe per obtenir els valors i vectors propis de AA^T, la qual cosa no és possible. Tammateix, aquesta relació entre els valors singulars de A i els valors propis de AA^T ens permetrà obtenir algorismes per a la SVD a partir d'algorismes per a valors propis. En el tema següent veurem que hi ha algorismes estables, necessàriament iteratius, per a calcular la SVD.

Una altra propietat fonamental de la SVD és la caracterització de la condició de la matriu en funció dels valors singulars. Sabem que la condició (en la norma euclidiana) d'una matriu invertible A és $\kappa(A) = \|A\|_2\|\|A^{-1}\|_2$. Pel problema 2, $\|A\|_2 = \sigma_1$. Pel problema 3 $\|A^{-1}\|_2 = \frac{1}{\sigma_n}$, doncs

$$\kappa(A) = \frac{\sigma_1}{\sigma_n}. \tag{6.9}$$

Deduïm d'ací que una matriu invertible és mal condicionada (és a dir, $\kappa(A)$ és gran) si i només si $\sigma_1 \gg \sigma_n$. El cas extrem es dóna quan $\sigma_n = 0$, és a dir, quan la matriu és no invertible, i en aquest cas es defineix $\kappa(A) = \infty$. Vegeu la secció 6.3 per a una interpretació geomètrica del mal condicionament de la matriu.

El fet anterior també es pot observar si intentem resoldre el sistema $Ax = b$, on A és quadrada i invertible, utilitzant la descomposició SVD. En efecte, com $A = U\Sigma V^T$, on $\sigma_1 \geq \ldots \geq \sigma_n > 0$, tenim

$$x = A^{-1}b = V\Sigma^{-1}U^T b = \sum_{k=1}^{n} \frac{u_k^T b}{\sigma_k} v_k.$$

Així, si σ_n és petit, petites variacions de b i A ens donaran grans variacions d'x.

Quan la matriu no és quadrada (i per tant no invertible) es pot definir la seua condició a partir de (6.9), com $\kappa(A) =$ quocient del major entre el menor valor singular. Si aquest últim és zero, aleshores $\kappa(A) = \infty$.

6.2 Aplicacions

6.2.1 Aproximació per matrius de rang petit

La SVD serveix per a aproximar òptimament matrius per altres de rang petit, tal com estableix el següent teorema.

Teorema 6.4. Siga $A = U\Sigma V^T$ una SVD amb els valors singulars ordenats de major a menor i designem $A_k = U(:, 1:k)\Sigma(1:k, 1:k)V(:, 1:k)^T$, $k = 1, \ldots, \text{rang}(A)$. Aleshores

1. $\text{rang}(A_k) = k$.
2. $\sigma_{k+1} = \|A - A_k\|_2 = \min_{\text{rang}(B)=k} \|A - B\|_2$.

Prova. La matriu A_k es pot expressar alternativament com

$$A_k = U\Sigma_k V^T, \quad \Sigma_k = \begin{bmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_k & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix} \tag{6.10}$$

Com que el rang es conserva en fer productes per matrius invertibles, tenim que $\text{rang}(A_k) = \text{rang}(\Sigma_k) = k$, ja que $\sigma_1 \geq \cdots \geq \sigma_k \geq \cdots \geq \sigma_r > 0$.

De la SVD de A i de la relació (6.10) obtenim

$$A - A_k = U(\Sigma - \Sigma_k)V^T = U\Sigma'_k V^T, \quad \Sigma'_k = \begin{bmatrix} 0 & & & & \\ & \ddots & & & \\ & & \sigma_{k+1} & & \\ & & & \ddots & \end{bmatrix}, \tag{6.11}$$

doncs $U\Sigma'_k V^T = A - A_k$ és una SVD de $A - A_k$, doncs, pel problema 2, $\|A - A_k\|_2 = \sigma_{k+1}$, el major element de la diagonal de Σ'_k. Deduïm que

$$\min_{\text{rang}(B)=k} \|A - B\|_2 \leq \sigma_{k+1}.$$

Per acabar, caldrà demostrar que $\|A - B\|_2 \geq \sigma_{k+1}$ per a qualsevol matriu B amb $\text{rang}(B) = k$. Considerem els subespais $\langle B\rangle$ generats per les columnes de B i $N(B) = \{x \in \mathbb{R}^n : Bx = 0\}$. Per la definició del rang d'una matriu, $\dim\langle B\rangle = k$, i, com $n = \dim\langle B\rangle + \dim N(B)$, aleshores $\dim N(B) = n - k$.

D'altra banda, siga $\langle V(:, 1:k+1)\rangle$ el subespai generat per les columnes de $V(:, 1:k+1)$, com que aquestes són linealment independents, perquè V és invertible, tenim que $\dim\langle V(:, 1:k+1)\rangle = k + 1$. Utilitzem ara la relació entre les dimensions de la suma i la intersecció de subespais per obtenir:

$$\begin{aligned} &\dim(N(B) \cap \langle V(:, 1:k+1)\rangle) \\ &= \dim N(B) + \dim\langle V(:, 1:k+1)\rangle - \dim(N(B) + \langle V(:, 1:k+1)\rangle) \\ &\geq n - k + k + 1 - n = 1 \end{aligned}$$

Per tant podem triar $x \in N(B) \cap \langle V(:, 1:k+1)\rangle \neq 0$, $\|x\|_2 = 1$. Com $x \in \langle V(:, 1:k+1)\rangle$, $x = \sum_{i=1}^{k+1} x_i v_i$, on v_i designa la columna i-èsima de V. Com que els vectors v_i pertanyen a una base ortonormal, $\|x\|_2 = \sqrt{\sum_{i=1}^{k+1} x_i^2}$. A més a més, $Ax = \sum_{i=1}^{k+1} x_i A v_i = \sum_{i=1}^{k+1} \sigma_i x_i u_i$, doncs, perquè u_i pertany a una base ortonormal i $\sigma_{k+1} \leq \sigma_i$, $i = 1, \ldots, k$,

$$\|Ax\|_2 = \sqrt{\sum_{i=1}^{k+1} \sigma_i^2 x_i^2} \geq \sigma_{k+1} \sqrt{\sum_{i=1}^{k+1} x_i^2} = \sigma_{k+1}$$

Tenint en compte tot això i que $Bx = 0$, calculem

$$\|(A - B)x\|_2 = \|Ax\|_2 \geq \sigma_{k+1}.$$

Com que $\|x\|_2 = 1$, tenim que $\|A - B\|_2 \geq \sigma_{k+1}$.

Aquest resultat indica que A_k és la millor aproximació a A entre les matrius de rang k i que si σ_{k+1} és petit en comparació amb $\sigma_1 = \|A\|_2$, aquesta és una bona aproximació.

Com $\sigma_n = \min_{\text{rang}(B)=n-1} \|A - B\|_2$. tenim que si σ_n és petit, A està molt prop de les matrius no invertibles. Així doncs, petites pertorbacions poden produir molta inestabilitat.

El resultat també pot ser utilitzar per "comprimir" matrius. Per obtenir una matriu A $m \times n$ necessitem mn nombres. En canvi, per obtenir $A_k = U(:, 1 : k)\Sigma(1 : k, 1 : k)V(:, 1 : k)^T$ sols necessitem $\Sigma(1 : k, 1 : k)V(:, 1 : k)^T$ i $U(:, 1 : k)$ (i multiplicar-les), és a dir, $(m+n)k$ nombres. Si $k \ll \min(m, n)$, aleshores $(m + n)k \ll mn$. Si, a més a més, σ_{k+1}/σ_1 és petit, aleshores hem aproximat la matriu A (mn nombres) per A_k ($(m + n)k$ nombres) amb un error relatiu $\|A - A_k\|_2/\|A\|_2 = \sigma_{k+1}/\sigma_1$ petit.

Exemple 6.5. Suposem que $A \in R^{10000\times10000}$ és densa. Aleshores, calcular el producte matriu-vector Ax és computacionalment car: 10^8 multiplicacions.

Però si A té com a valors singulars:

$$\sigma_1 = 100, \sigma_2 = 35, \sigma_3 = 10, \sigma_4 = 2$$

i $\sigma_k \leq 0.001$ per a $k \geq 5$, aleshores l'aproximant de rang 4 és

$$A_4 = \sum_{i=1}^{4} \sigma_i u_i v_i^T.$$

Aleshores, si fem

$$b = A_4 x = 100(v_1^T x)u_1 + 35(v_2^T x)u_2 + 10(v_3^T x)u_3 + 2(v_4^T x)u_4$$

tenim

$$\|Ax - b\| \leq \|A - A_4\|\|x\| \leq 0.001\|x\|$$

amb la qual cosa obtenim una aproximació a Ax amb un error relatiu menor que 0.001 en 4×10^4 multiplicacions.

6.2.2 Anàlisi de components principals

Suposem que $x_1, \ldots, x_n$ són variables aleatòries i X és una matriu $m \times n$ tal que les seues files estan formades per mostres de cada variable x_i. Es defineix la *matriu de covariància* de X com la matriu $C = conv(X)$ tal que

$$C(i, j) = \frac{1}{m-1} \sum_{k=1}^{m} (X(k, i) - \overline{X}(i))(X(k, j) - \overline{X}(j))$$

$$\overline{X}(i) = \frac{1}{m} \sum_{k=1}^{m} X(k, i), \quad i, j = 1, \ldots, n,$$

on $\overline{X}(i)$ és la mitjana de les mostres de x_i. Noteu que

$$\begin{bmatrix} \overline{X}(1) & \ldots & \overline{X}(n) \end{bmatrix} = \frac{1}{m} \begin{bmatrix} 1 & \ldots & 1 \end{bmatrix} X$$

La *variància* de la mostra de x_i és el i-èsim element de la diagonal de C:

$$C(i,i) = \frac{1}{m-1}\sum_{k=1}^{m}(X(k,i) - \overline{X}(i))^2.$$

Noteu que podem expressar:

$$C = \frac{1}{m-1}\widetilde{X}^T\widetilde{X}$$

$$\widetilde{X} = X - \begin{bmatrix}1\\ \vdots \\ 1\end{bmatrix}\begin{bmatrix}\overline{X}(1) & \dots & \overline{X}(n)\end{bmatrix} = X - \begin{bmatrix}1\\ \vdots \\ 1\end{bmatrix}\frac{1}{m}\begin{bmatrix}1 & \dots & 1\end{bmatrix}X =$$

$$\underbrace{(I_m - \frac{1}{m}\begin{bmatrix}1\\ \vdots \\ 1\end{bmatrix}\begin{bmatrix}1 & \dots & 1\end{bmatrix})}_{F_m}X$$

ja que la matriu C és simètrica i semidefinida positiva.

La *matriu de correlació* es defineix com la matriu R tal que

$$R(i,j) = \frac{C(i,j)}{\sqrt{C(i,i)}\sqrt{C(j,j)}}.$$

Noteu que $R(i,i) = 1$ i que $R(i,j)$ és la *correlació* entre x_i i x_j deduïda de la mostra X.

L'anàlisi de components principals consisteix a trobar altres variables aleatòries $y_1, \dots, y_n$, relacionades amb $x_1, \dots, x_n$ mitjançant un canvi de base ortonormal, tal que la matriu de covariància siga diagonal, és a dir, tal que les noves variables y_i tinguen correlació nul·la entre elles.

Si formem la matriu P (a determinar) a partir dels coeficients de les combinacions lineals de y_i en funció dels x_i:

$$\begin{aligned} y_1 &= P(1,1)x_1 + \dots + P(n,1)x_n \\ &\dots \\ y_n &= P(1,n)x_1 + \dots + P(n,n)x_n \end{aligned}$$

la matriu de mostres que deduïm de X i d'aquestes combinacions lineals és

$$\begin{aligned} Y(i,1) &= P(1,1)X(i,1) + \dots + P(n,1)X(i,n) \\ &\dots \\ Y(i,n) &= P(1,n)X(i,1) + \dots + P(n,n)X(i,n) \end{aligned}$$

la qual es pot expressar com $Y = XP$. La matriu de covariància de Y és

$$cov(Y) = \frac{1}{m-1}(F_mY)^T(F_mY) = \frac{1}{m-1}(F_mXP)^T(F_mXP) =$$

$$\frac{1}{m-1}P^TX^TF_m^TF_mXP = P^Tcov(X)P.$$

Si $cov(Y)$ té ha de ser diagonal, aleshores les covariàncies (els elements de la diagonal de $cov(Y)$) han de ser els valors propis de $cov(X)$ i les columnes de P han de ser els vectors propis associats. Es dedueix d'ací i de la demostració del teorema 6.1 que P és la matriu dels vectors singulars de la matriu $\widetilde{X} = F_m X$ i que s'hauria de calcular a partir d'aquesta i no a partir de $cov(X)$.

La utilitat d'aquesta anàlisi sol ser la detecció de *redundància* en les dades. Per exemple, si, com a cas extrem, tenim que $\widetilde{Y}(:,i)^T\widetilde{Y}(:,i) = cov(Y)(i,i) = 0$, i $\widetilde{Y}(k,i) = Y(k,i) - \overline{Y}(i)$, aleshores

$$\widetilde{Y}(:,i) = 0 \Rightarrow Y(k,i) = \overline{Y}(i) \quad \forall k,$$

$$\begin{aligned} P(1,i)X(k,1) + \cdots + P(n,i)X(k,n) = Y(k,i) = \overline{Y}(i) = P(1,i)\overline{X}(1) + \cdots + P(n,i)\overline{X}(n) \Rightarrow \\ P(1,i)(X(k,1) - \overline{X}(1)) + \cdots + P(n,i)(X(k,n) - \overline{X}(n)) = 0 \quad \forall k, \end{aligned}$$

és a dir, la combinació lineal

$$P(1,i)(x_1 - \overline{X}(1)) + \cdots + P(n,i)(x_n - \overline{X}(n))$$

dóna un mostreig 0, per la qual cosa podem eliminar una de les variables, expressant-la com a combinació lineal de les altres.

6.2.3 Sistemes lineals

Si $Ax = b$ és un sistema lineal amb matriu $m \times n$ i $A = U\Sigma V^T$ és una SVD, aleshores podem obtenir un sistema "equivalent" senzill de resoldre utilitzant aquesta descomposició:

$$U^T A V \underbrace{(V^T x)}_{\overline{x}} = \underbrace{U^T b}_{\overline{b}} \Rightarrow \Sigma\overline{x} = \overline{b} \tag{6.12}$$

on $\overline{x} = V^T x$ designa les coordenades de x en la base ortonormal (formada per les columnes de) V per i $\overline{b} = U^T b$ les coordenades de b en la base ortonormal U. Si $r = \text{rang}(A)$ aleshores el sistema (6.12) es pot escriure com

$$\begin{aligned} \sigma_1\overline{x}_1 &= \overline{b}_1 \\ &\vdots \\ \sigma_r\overline{x}_r &= \overline{b}_r \\ 0 &= \overline{b}_{r+1} \\ &\vdots \\ 0 &= \overline{b}_m \end{aligned} \tag{6.13}$$

Deduïm que si $r < m$ i $\overline{b}_j \neq 0$, $r+1 \leq j \leq m$ aleshores el sistema és incompatible. Tractarem aquest cas en la secció següent.

Suposem que el sistema és compatible, és a dir $\overline{b}_j = 0$ $j = r+1, \ldots, m$. Obtenim de (6.13) que $\overline{x}_i = \overline{b}_i/\sigma_i$ $i = 1, \ldots, r$ i que no hi ha cap relació que lligue els $\overline{x}_i$, $i = r+1, \ldots, n$. Com que la solució del sistema original la dóna $x = V\overline{x}$ (per (6.12) tenim que la solució és

$$x = \frac{\overline{b}_1}{\sigma_1}v_1 + \cdots + \frac{\overline{b}_r}{\sigma_r}v_r + \lambda_{r+1}v_{r+1} + \cdots + \lambda_n v_n,$$

on $\lambda_{r+1}, \ldots, \lambda_n$ són paràmetres. Deduïm també que la solució del sistema homogeni $Ax = 0$ (el nucli de A) és el subespai generat per $v_{r+1}, \ldots, v_n$.

6.2.4 Mínims quadrats

Quan un sistema lineal $m \times n$ $Ax = b$ no té solució, el millor que es pot fer per "resoldre'l" és trobar el vector x que estiga més a prop de verificar les m equacions, és a dir, fer el residual $b - Ax$ el menor possible. Quan la mesura aplicada al residual és la norma euclídia, es planteja el problema de *mínims quadrats*: trobar x_* tal que

$$\|b - Ax_*\|_2 = \min_x \|b - Ax\|_2 \tag{6.14}$$

El problema de mínims quadrats és equivalent a les *equacions normals*

$$A^T Ax = A^T b. \tag{6.15}$$

Si rang$(A) = n$, és a dir, les columnes de A són linealment independents, aleshores rang$(A^T A) = n$, la matriu $A^T A$ és invertible i les equacions normals tenen solució única, ja que també té solució única el problema dels mínims quadrats. Si rang$(A) < n$ la matriu s'anomena *defectiva*; aleshores hi ha múltiples solucions de les equacions normals i, per tant, del problema de mínims quadrats. Tractarem aquest cas en la secció següent.

A banda d'exigir més operacions que els algorismes usuals, quan la matriu és prop de ser defectiva (mal condicionada) resoldre el problema de mínims quadrats mitjançant les equacions normals no és el mètode adequat des d'un punt de vista numèric. Els algorismes usuals per a resoldre el problema del mínims quadrats utilitzen transformacions ortogonals (les quals preserven la norma euclidiana) per reduir el problema original a un altre equivalent de resolució directa. Molts d'aquests algorismes estan basats en la descomposició $A = QR$, Q ortogonal, R triangular superior, la qual existeix quan la matriu no és defectiva, encara que es poden aplicar tècniques de pivotatge per columnes per resoldre la deficiència de rang. No entrarem en aquests detalls, perquè la SVD ens permet un algorisme alternatiu per resoldre problemes de mínims quadrats amb qualsevol tipus de matriu.

6.2.5 Solució de problemes de mínims quadrats per SVD

Considerem el problema de mínims quadrats (6.14) i una SVD $A = U\Sigma V^T$. Designem les coordenades d'un vector genèric x en la base ortonormal V per $\overline{x} = V^T x$ i les coordenades de b en la base ortonormal U per $\overline{b} = U^T b$. Aleshores, utilitzant aquesta notació i que el producte per matrius ortogonals preserva la norma euclidiana, obtenim

$$\|Ax - b\|_2 = \|U\Sigma V^T x - U(U^T b)\|_2 = \|U(\Sigma V^T x - U^T b)\|_2 = \|\Sigma\overline{x} - \overline{b}\|_2 \tag{6.16}$$

Per tant

$$\min_x \|Ax - b\|_2 = \min_{\overline{x}} \|\Sigma\overline{x} - \overline{b}\|_2 \tag{6.17}$$

amb les solucions dels dos problemes relacionades per $\overline{x} = V^T x$.

Conseqüentment, hem passat d'un problema de mínims quadrats amb una matriu general a un problema de mínims quadrats amb una matriu diagonal, el qual és trivial de resoldre, tal com veurem tot seguit. Siga $r = \text{rang}(A)$ i introduïm la notació

$$\begin{aligned} \Sigma^1 &= \Sigma(1:r, 1:r) \\ \overline{x}^1 &= \overline{x}(1:r) \\ \overline{x}^2 &= \overline{x}(r+1:n) \\ \overline{b}^1 &= \overline{b}(1:r) \\ \overline{b}^2 &= \overline{b}(r+1:m) \end{aligned} \tag{6.18}$$

aleshores

$$\Sigma = \begin{bmatrix} \Sigma^1 & 0 \\ 0 & 0 \end{bmatrix}$$

i, per tant,

$$\|\Sigma\overline{x} - \overline{b}\|_2^2 = \| \begin{bmatrix} \Sigma^1\overline{x}^1 \\ 0 \end{bmatrix} - \begin{bmatrix} \overline{b}^1 \\ \overline{b}^2 \end{bmatrix} \|_2^2 = \| \begin{bmatrix} \Sigma^1\overline{x}^1 - \overline{b}^1 \\ -\overline{b}^2 \end{bmatrix} \|_2^2 = \|\Sigma^1\overline{x}^1 - \overline{b}^1\|_2^2 + \|\overline{b}^2\|_2^2 \tag{6.19}$$

Deduïm que

$$\min_{\overline{x}} \|\Sigma\overline{x} - \overline{b}\|_2^2 \geq \|\overline{b}^2\|_2^2,$$

amb igualtat si i només si $\|\Sigma^1\overline{x}^1 - \overline{b}^1\|_2 = 0$, és a dir, si i només si $\Sigma^1\overline{x} = \overline{b}^1$, la qual cosa és equivalent a

$$\overline{x}_i = \overline{b}_i/\sigma_i, i = 1, \ldots, r. \tag{6.20}$$

Tal com hem deduït en (6.16), el conjunt de solucions del problema original és $V\overline{x}$ tal que $\overline{x}$ és solució del problema (6.17), és a dir, tal que $\overline{x}$ verifica (6.20). Deduïm que la solució del problema de mínims quadrats original és

$$x = \frac{\overline{b}_1}{\sigma_1}v_1 + \cdots + \frac{\overline{b}_r}{\sigma_r}v_r + \lambda_{r+1}v_{r+1} + \cdots + \lambda_n v_n, \tag{6.21}$$

on $\overline{b} = U^T b$ i $\lambda_{r+1}, \ldots, \lambda_n$ són paràmetres, els quals no apareixen si $r = n$, és a dir, la solució és única quan el rang de la matriu és el màxim possible (matriu no defectiva).

Quan la matriu és defectiva, se solen imposar condicions addicionals per triar una solució adequada entre les de la relació (6.21). Habitualment es demana que la solució tinga la menor norma euclidiana possible. Com que V és una matriu ortogonal, es dedueix que la norma de x de (6.21) és

$$\|x\|_2^2 = \left(\frac{\overline{b}_1}{\sigma_1}\right)^2 + \cdots + \left(\frac{\overline{b}_r}{\sigma_r}\right)^2 + \lambda_{r+1}^2 + \cdots + \lambda_n^2,$$

i és mínima quan $\lambda_{r+1} = \cdots = \lambda_n = 0$, és a dir, la solució del problema de mínims quadrats amb norma mínima és

$$x = \frac{\overline{b}_1}{\sigma_1}v_1 + \cdots + \frac{\overline{b}_r}{\sigma_r}v_r. \tag{6.22}$$

6.3 Interpretació geomètrica

Suposem que la matriu A no és defectiva. En particular, tenim que $m \geq n$. La interpretació geomètrica de la SVD és que la matriu A transforma la bola unitat en $\mathbb{R}^n$ en un hiperel·lipsoide en $\mathbb{R}^m$, contingut en el subespai lineal generat per $u_1, \ldots, u_n$, $n = \text{rang}(A)$. El semieix i-èsim , té longitud σ_i, està en la direcció de u_i i és la imatge de v_i per A. Vegem-ho. L'equació d'un hiperel·lipsoide amb semieixos en la direcció u_i i longitud σ_i, $i = 1, \ldots, n$ és

$$\left(\frac{\overline{y}_1}{\sigma_1}\right)^2 + \cdots + \left(\frac{\overline{y}_n}{\sigma_n}\right)^2 = 1, \overline{y}_i = 0, i = n+1, \ldots, m \tag{6.23}$$

on $\overline{y} = U^T y \in \mathbb{R}^m$ són les coordenades del vector genèric $y \in \mathbb{R}^m$ en la base U. El que volem veure és que $y = Ax$ està en l'hiperel·lipsoide d'equacions (6.23) si $\|x\|_2 = 1$. Tenint en compte la SVD $A = U\Sigma V^T$ tenim que

$$\overline{y} = U^T Ax = \Sigma V^T x = \Sigma\overline{x}, \overline{x} = V^T x. \tag{6.24}$$

D'aquesta relació obtenim

$$\overline{y}_i = \overline{x}_i \sigma_i, i = 1, \ldots, n, \overline{y}_i = 0 i = n+1, \ldots, m. \tag{6.25}$$

Substituint aquestes relacions en (6.23) obtenim

$$\begin{aligned} &\left(\frac{\sigma_1 \overline{x}_1}{\sigma_1}\right)^2 + \cdots + \left(\frac{\sigma_2 \overline{x}_n}{\sigma_n}\right)^2 \\ &= (\overline{x}_1)^2 + \cdots + (\overline{x}_n)^2 \\ &= \|\overline{x}\|_2 = \|V^T x\|_2 = \|x\|_2 = 1, \\ &\overline{y}_i = 0, i = n+1, \ldots, m \end{aligned} \tag{6.26}$$

La figura 6.1 mostra aquest fet pel cas $m = n = 2$.

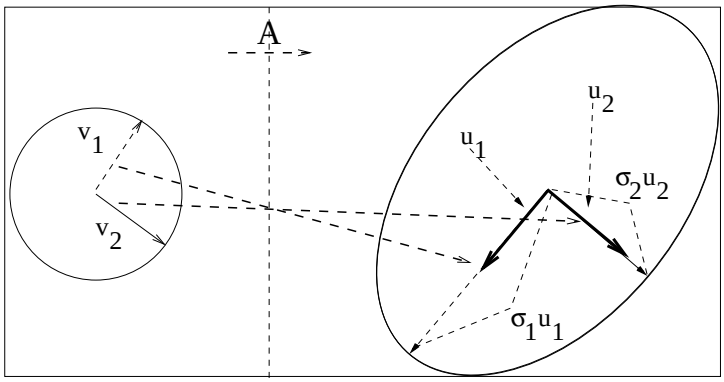


Figura 6.1: Interpretació geomètrica de la SVD.

Utilitzant les intuïcions exposades abans, les figures 6.2, 6.3, 6.4 donen interpretacions geomètriques de la resolució de sistemes lineals, del bon condicionament dels sistemes i del mal condicionament dels sistemes. Adoneu-vos que per l'equació (6.9) una matriu ben condicionada correspon a un el·lipsoide quasi circular i una matriu mal condicionada a un el·lipsoide molt estirat.

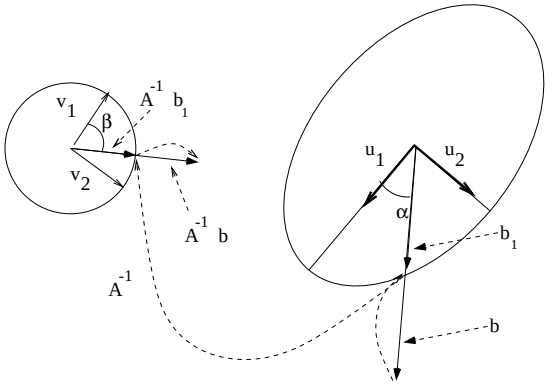


Figura 6.2: Interpretació geomètrica de la resolució de sistemes a partir de la SVD: el mateix factor λ que s'aplica per obtenir $b_1 = \lambda b$ sobre l'el·lipsoide s'aplica després de calcular $A^{-1}b_1$ per obtenir $A^{-1}b = \lambda A^{-1}b_1$.

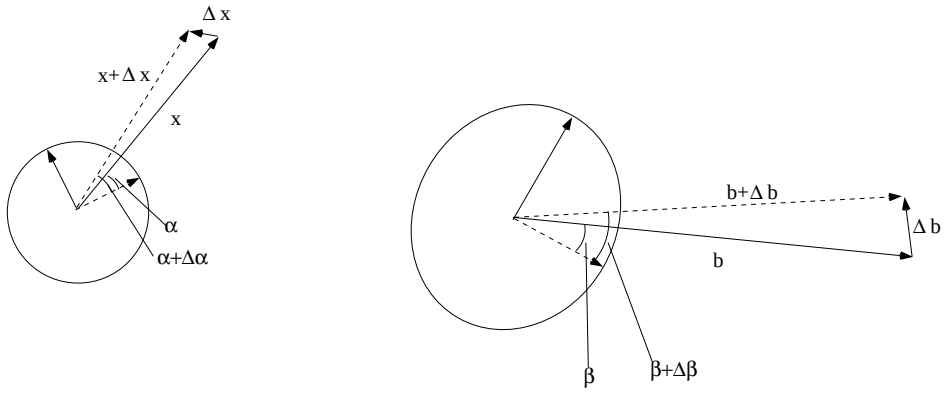


Figura 6.3: Interpretació geomètrica del bon condicionament de la resolució de sistemes a partir de la SVD.

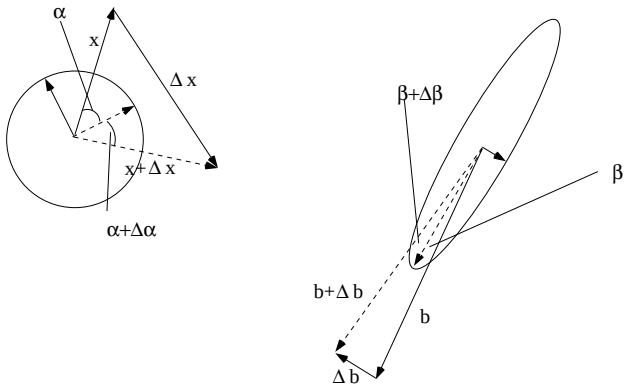


Figura 6.4: Interpretació geomètrica del mal condicionament de la resolució de sistemes a partir de la SVD.

6.4 Algorisme QR per al càlcul de la SVD

De la mateixa manera que ocorre amb les equacions normals per solucionar problemes de mínims quadrats, en el càlcul de la SVD no és convenient calcular A^TA, ja que pot suposar una pèrdua de precisió fatal i augmentar la condició del problema. L'algorisme que presentem aplica implícitament l'algorisme QR a A^TA, però sense formar-la.

El primer pas és reduir la matriu A a bidiagonal (superior) mitjançant transformacions ortogonals. L'algorisme és semblant al de la transformació d'una matriu simètrica en tridiagonal per a posteriorment aplicar mètodes per calcular valors/vectors propis. La diferència fonamental és que ací tenim un poc més de llibertat, ja que les transformacions per l'esquerra són independents de les de la dreta. Aquest fet és el que permet reduir la matriu a sols dues diagonals (per què no podem reduir-la a una?).

Sempre podem suposar que $m \geq n$ a l'hora de calcular la SVD d'una matriu $m \times n$ (vegeu el problema 3 del tema 6). Mostrem com funciona aquest algorisme per a una matriu 5×4. Esquemàticament, la matriu és

$$A = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}$$

1. Calculem una matriu de Householder U_1 que actua sobre les entrades $(1:5, 1:4)$ i anul·la les entrades $(2:5, 1)$:

$$U_1 A = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix}$$

2. Calculem una matriu de Householder V_1 que actua sobre les entrades $(1:5, 2:4)$ i anul·la les entrades $(1, 3:4)$

$$U_1 A V_1 = \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix}$$

   Adoneu-vos que, en no actuar sobre la columna 1, els zeros aconseguits en el pas anterior no es modifiquen.

3. Calculem una matriu de Householder U_2 que actua sobre les entrades $(2:5, 2:4)$ i anul·la les entrades $(3:5, 2)$:

$$U_2 U_1 A V_1 = \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix}$$

4. Calculem una matriu de Householder V_2 que actua sobre les entrades $(2:5, 3:4)$ i anul·la les entrades $(2, 4)$:

$$U_2 U_1 A V_1 V_2 = \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix}$$

5. Calculem una matriu de Householder U_3 que actua sobre les entrades $(3:5, 3:4)$ i anul·la les entrades $(4:5, 3)$:

$$U_3 U_2 U_1 A V_1 V_2 = \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ 0 & 0 & 0 & \times \end{bmatrix}$$

6. Calculem una matriu de Householder U_4 que actua sobre les entrades $(4:5, 4)$ i anul·la les entrades $(5, 4)$:

$$U_3U_2U_1AV_1V_2 = \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

A partir d'aquest algorisme obtenim matrius ortogonals X_1, Y_1 $m \times m$ i $n \times n$ i una matriu bidiagonal superior B $n \times n$ tal que

$$X_1AY_1 = \begin{bmatrix} B \\ 0 \end{bmatrix}$$

Si $B = \overline{U\Sigma V}^T$ és una SVD de B aleshores tenim la SVD de A:

$$A = \underbrace{X_1^T \begin{bmatrix} \overline{U} & 0 \\ 0 & I_{m-n} \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} \overline{\Sigma} \\ 0 \end{bmatrix}}_{\Sigma} \underbrace{\overline{V}^T Y_1^T}_{V^T}.$$

Per tant, a l'hora de calcular la SVD podem suposar que la matriu A és bidiagonal superior i $n \times n$.

6.4.1 Algorisme QR-SVD per a matrius bidiagonals

Suposem que A és bidiagonal superior. La primera iteració de l'algorisme QR per a la matriu A^TA consisteix a:

1. calcular la descomposició QR de $B_0 = A^TA$: $B = QR$
2. calcular $B_1 = RQ$.

Com que A ja és triangular superior i el producte de triangulars superiors també ho és, per complir el pas 1 bastarà calcular Q^T ortogonal tal que $Q^TA^T = R_1$ siga triangular superior o, el que és el mateix, tal que $L_1 = R_1^T = AQ$ siga triangular inferior. Aleshores tindrem que

$$A^TA = QQ^TA^TA = Q\underbrace{(R_1A)}_{R}$$

és una descomposició QR de A^TA. En el pas 2 calculem

$$B_1 = RQ = Q^TA^TAQ = L_1^TL_1$$

La iteració següent consisteix a calcular la descomposició QR de B_1, però ara no podem utilitzar la mateixa tècnica que abans, perquè L_1 no és bidiagonal superior (de fet es pot veure que és bidiagonal *inferior*). Ara bé, si calculem P ortogonal tal que $A_1 = PL_1$ és bidiagonal superior tindrem que

$$A_1^TA_1 = L_1^TP^TPL_1 = L_1^TL_1 = B_1,$$

és a dir, B_1 té la mateixa estructura que B, producte d'una transposició d'una bidiagonal superior per aquesta sense transposar. L'algorisme 1 implementa la idea de generar A_{i+1} a partir de A_i tal que $B_i = A_i^TA_i$ és la iteració i-èsima de l'algorisme QR aplicat a $B_0 = A^TA$. Sabem que B_i convergeix en una matriu diagonal, aleshores, pel problema 6, A_i també convergeix a una matriu diagonal Σ. L'algorisme s'atura quan

$$\|A_i - \text{diag}(A_i)\| < \varepsilon$$

on $\varepsilon = \mathcal{O}(\varepsilon_{\text{mach}}\|A\|)$, amb la qual cosa assegurem l'estabilitat cap enrere.

A aquest algorisme se li pot incorporar deflació: quan

$$|A_i(j, j+1)| < \varepsilon = \mathcal{O}(\varepsilon_{\text{mach}}\|A\|)$$

podem fer $A_i(j, j+1) = 0$ sense influir en l'estabilitat cap enrere. Si designem la matriu resultant d'anul·lar $A_i(j, j+1)$

$$\tilde{A}_i = \begin{bmatrix} A_i(1:j, 1:j) & 0 \\ 0 & A_i(j+1:n, j+1:n) \end{bmatrix},$$

la qual verifica $\tilde{A}_i = A_i + \Delta A_i$, $\|\Delta A_i\| = \mathcal{O}(\varepsilon_{\text{mach}}\|A\|)$ i calculem SVDs dels blocs de la diagonal:

$$A_i(1:j, 1:j) = Z_1\Sigma_1 X_1^T, A_i(j+1:n, j+1:n) = Z_2\Sigma_2 X_2^T,$$

amb Z_i, X_i matrius ortogonals i Σ_i matrius diagonals amb entrades no negatives ($i = 1, 2$), aleshores podem construir aproximadament una SVD de A:

$$A = U_i A_i V_i^T = U_i(\tilde{A}_i + \Delta A_i)V_i^T = \underbrace{U_i \begin{bmatrix} Z_1 & 0 \\ 0 & Z_2 \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix}}_{\Sigma} \underbrace{(V_i \begin{bmatrix} X_1 & 0 \\ 0 & X_2 \end{bmatrix})^T}_{V^T} + U_i\Delta A_i V_i^T$$

Si anomenem $\Delta A = -U_i\Delta A_i V_i^T$ tenim que $\|\Delta A \leq \|\Delta A_i\| = \mathcal{O}(\varepsilon_{\text{mach}}\|A\|)$ i $A + \Delta A = U\Sigma V^T$, ja que calculem la SVD d'una matriu $A + \Delta A$ que és una pertorbació relativa de A de l'ordre de $\varepsilon_{\text{mach}}$, és a dir, calculem la SVD de A d'una manera estable cap enrere.

$A_0 = A, U_0 = I, V_0 = I$
for $i = 0, \ldots$
  calcular Q_i tal que AQ_i triangular inferior
  calcular P_i tal que PAQ_i bidiagonal superior
  $A_{i+1} = P_i A_i Q_i$
  $U_{i+1} = U_i P_i^T$
  $V_{i+1} = V_i Q_i$
end

Algorisme 1: Algorisme QR per calcular la SVD d'una matriu bidiagonal superior.

L'últim detall que ens falta per completar la descripció és mostrar com calcular les matrius P_i i Q_i de l'algorisme 1. Mostrem un exemple amb una matriu 3×3: siga la matriu

$$A_i = \begin{bmatrix} a_1 & b_1 & 0 \\ 0 & \times & \times \\ 0 & 0 & \times \end{bmatrix}$$

1. Siga la transformació de Givens

$$G_1 = \frac{1}{\sqrt{a_1^2 + b_1^2}} \begin{bmatrix} a_1 & -b_1 & 0 \\ b_1 & a_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

que verifica

$$AG_1 = \begin{bmatrix} \times & 0 & 0 \\ \times & a_2 & b_2 \\ 0 & 0 & \times \end{bmatrix}$$

2. Siga la transformació de Givens

$$G_2 = \frac{1}{\sqrt{a_2^2 + b_2^2}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & a_2 & -b_2 \\ 0 & b_2 & a_2 \end{bmatrix}$$

que verifica

$$AG_1G_2 = \begin{bmatrix} \times & 0 & 0 \\ \times & \times & 0 \\ 0 & \times & \times \end{bmatrix}.$$

Aleshores $Q_i = G_1G_2$ verifica que AQ_i és triangular inferior (adoneu-vos que, de fet, és bidiagonal inferior).

3. Siga ara

$$AQ_i = \begin{bmatrix} a_3 & 0 & 0 \\ b_3 & \times & 0 \\ 0 & \times & \times \end{bmatrix}$$

i la transformació de Givens

$$G_1 = \frac{1}{\sqrt{a_3^2 + b_3^2}} \begin{bmatrix} a_3 & b_3 & 0 \\ -b_3 & a_3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

que verifica

$$G_3AQ_i = \begin{bmatrix} \times & \times & 0 \\ 0 & a_4 & 0 \\ 0 & b_4 & \times \end{bmatrix}$$

4. Finalment, siga G_4 la transformació de Givens

$$G_4 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & a_4 & b_4 \\ 0 & -b_4 & a_4 \end{bmatrix}$$

que verifica

$$G_4G_3AQ_i = \begin{bmatrix} \times & \times & 0 \\ 0 & \times & \times \\ 0 & 0 & \times \end{bmatrix}$$

Aleshores $P_i = G_4G_3$ verifica que $P_iA_iQ_i$ és bidiagonal superior.

També es poden incorporar translacions implícites, però aquesta tècnica és molt més complicada que la que hem presentat ací.

6.5 Problemes

Problema 1. Si $U\Sigma V^T = A$ és una SVD de A demostreu que podeu construir-ne una altra $\tilde{U}\tilde{\Sigma}\tilde{V}^T = A$ tal que els elements de la diagonal de $\tilde{\Sigma}$ siguen els elements de la diagonal de Σ reordenats de major a menor i que hi haja sols $r = \text{rang}(A)$ no nuls.

Problema 2. Si $U\Sigma V^T = A$ és una SVD aleshores $\|A\|_2$ és el major valor singular de A.

Problema 3.

1. Si $U\Sigma V^T = A$ és una SVD d'una matriu invertible A $n \times n$ aleshores $V\Sigma^{-1}U^T = A^{-1}$ és una SVD de A^{-1} i $\|A^{-1}\|_2 = \frac{1}{\sigma_n}$.

2. Si $U\Sigma V^T = A$ és una SVD d'una matriu A $m \times n$ aleshores $V\Sigma^T U^T = A^T$ és una SVD de A^T.

Problema 4. Calculeu la SVD de la matriu

$$A = \begin{bmatrix} 3 & 2 \\ -1 & 2 \\ 2 & 0 \end{bmatrix}$$

Solució.

$$U\Sigma V^T = \begin{bmatrix} 2/\sqrt{5} & 1/\sqrt{30} & * \\ 0 & 5/\sqrt{30} & * \\ 1/\sqrt{5} & -2/\sqrt{30} & * \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 0 & \sqrt{6} \end{bmatrix} \begin{bmatrix} 2/\sqrt{5} & 1/\sqrt{5} \\ -1/\sqrt{5} & 2/\sqrt{5} \end{bmatrix} = \begin{bmatrix} 3 & 2 \\ -1 & 2 \\ 2 & 0 \end{bmatrix} = A$$

Problema 5. Donada la matriu A, $m \times n$ $m \geq n$, siga $A = U\Sigma V^T$ la seua SVD. Suposem $\sigma_1 \geq \ldots \sigma_n > 0$. Es defineix $K_2(A) = \|A\|\|A^+\|$ on $A^+ = V\Sigma^+ U^T$, $n \times m$ i

$$\Sigma^+ = \begin{bmatrix} 1/\sigma_1 & & 0 & 0 & 0 \\ & \ddots & & & \\ & & 1/\sigma_n & 0 & 0 \end{bmatrix}$$

Proveu

1. $AA^+A = A$
2. $A^+AA^+ = A^+$
3. $A^+ = A^{-1}$ si A és quadrada i no singular.
4. $K_2(A) = \|A\|_2\|(A^TA)^{-1}A^T\|_2$
5. $K_2(A)^2 = \|A\|_2^2\|(A^TA)^{-1}\|_2$
6. $(r(A^TA))^{1/2} = \sigma_1$

Problema 6. Siguen A_i matrius bidiagonals superiors tals que $A_i^TA_i \to E$ matriu diagonal amb entrades positives en la diagonal. Demostreu que $A_i \to \sqrt{E}$.

Pista:

1. Escriviu

$$A_i = \begin{bmatrix} d_1^{(i)} & u_1^{(i)} & & \\ & \ddots & \ddots & \\ & & \ddots & u_{n-1}^{(i)} \\ & & & d_n^{(i)} \end{bmatrix}$$

i comproveu que

$$A_i^T A_i = \begin{bmatrix} (d_1^{(i)})^2 & u_1^{(i)} d_1^{(i)} & & \\ u_1^{(i)} d_1^{(i)} & (u_1^{(i)})^2 + (d_2^{(i)})^2 & & \\ & \ddots & \ddots & \ddots \\ & & \ddots & u_{n-1}^{(i)} d_{n-1}^{(i)} \\ & & u_{n-1}^{(i)} d_{n-1}^{(i)} & (u_{n-1}^{(i)})^2 + (d_n^{(i)})^2 \end{bmatrix}$$

2. Si $A_i^T A_i \to \text{diag}(e_1, \ldots, e_n)$, amb $e_i \neq 0$, demostreu per inducció sobre j que $\lim_i d_j^{(i)} = \sqrt{e_j}$ i que $\lim_i u_j^{(i)} = 0$, és a dir, $\lim_i A_i^T A_i = \text{diag}(\sqrt{e_1}, \ldots, \sqrt{e_n})$.

Tema 7

Inverses generalitzades

7.1 Introducció

L'interès d'aquest capítol és d'estudiar els sistemes amb matriu no invertible, bé perquè no té rang màxim o bé perquè no és quadrada. Intentarem trobar relacions que han de complir certes matrius que fan "parcialment" el paper de la matriu inversa, les *inverses generalitzades*.

Com a primer exemple, suposem que la matriu A $m \times n$ té rang n i $m > n$, aleshores tenim que les columnes de A són linealment independents i per tant, part d'una base de $\mathbb{R}^m$. Si designem P la matriu que té per columnes els elements d'aquesta base, és a dir, tal que $P(:, 1:n) = A$, aleshores

$$I_m(:, 1:n) = P^{-1}(P(:, 1:n)) = P^{-1}A \Rightarrow I_n = P^{-1}(1:n, :)A,$$

ja que $C = P^{-1}(1:n, :)$ verifica $CA = I_n$. Una matriu C que verifique $CA = I_n$ s'anomena *inversa per l'esquerra* de C. Cal adonar-se que si A té inversa per l'esquerra, aleshores rang$A = n$, ja que $n =$ rang$I_n \leq$ rang$A \leq n$. A més a més, cal notar que parlem *d'una* inversa generalitzada, perquè no és única (la matriu P anterior no és única, tret que $m = n$).

De manera dual, es pot definir el concepte *d'inversa per la dreta* i provar que una matriu té inversa per la dreta si i només si rang$A = m$.

Si A té una inversa per l'esquerra C i $Ax = b$ és un sistema compatible (i determinat), aleshores $x = Cb$ és (la) solució del sistema. Aquest tipus de relacions seran els temes que tractarem en aquest capítol.

7.2 Inverses generalitzades

Definició 7.1. Una matriu C és *inversa generalitzada* de A si per a qualsevol b que faça compatible $Ax = b$, el vector Cb és solució de $Ax = b$.

Teorema 7.2. C és inversa generalitzada de A si i només si $ACA = A$.

Prova. Suposem que C és inversa generalitzada, per demostrar $ACA = A$ veiem que $ACAy = Ay$ per a qualsevol $y \in \mathbb{R}^n$. Siga $y \in \mathbb{R}^n$, definim $b = Ay$, aleshores el sistema $Ax = b$ és compatible, ja que Cb és solució de $Ax = b$, és a dir,

$$ACb = b \Rightarrow ACAy = Ay \quad \forall y \in \mathbb{R}^n \Rightarrow ACA = A.$$

Suposem, per contra, que $ACA = A$ i siga $Ax = b$ un sistema compatible, és a dir, tal que $b = Ay$ per algun y. Vegem que Cb és solució de $Ax = b$:

$$ACb = ACAy = Ay = b.$$

La notació que emprarem per a les inverses generalitzades és A^-. Amb això i el teorema anterior tenim l'equació que defineix les inverses generalitzades:

$$AA^-A = A.$$

En cas que el rang de la matriu siga el màxim possible, el concepte d'inversa generalitzada coincideix amb el de les inverses "laterals" (per l'esquerra en cas que rang$A = n \leq m$ o per la dreta en cas que rang$A = m \leq n$):

Si una matriu té rang n aleshores $AX = AY$ implica $X = Y$ i si té rang m aleshores $XA = YA$ implica $X = Y$. En el primer cas, podem eliminar la A de l'esquerra de l'equació $AA^-A = AI$, per obtenir $A^-A = I$. De manera similar, en el segon cas $AA^- = I$.

Deduïm que el camp d'aplicació d'aquesta noció és quan el rang no és màxim.

SVD i existència d'inverses generalitzades

Siga $A = U\Sigma V^T$ una SVD i particionem Σ com

$$\Sigma = \begin{bmatrix} \Sigma_{11} & 0 \\ 0 & 0 \end{bmatrix},$$

on Σ_{11} és la matriu diagonal $r \times r$ formada pels valors singulars de A (adoneu-vos que la segona columna de zeros no hi apareix si $r = n$ i/o la segona fila de zeros no apareix si $r = m$). Transformem l'equació $ACA = A$ mitjançant U i V:

$$ACA = A \Leftrightarrow (U\Sigma V^T)C(U\Sigma V^T) = U\Sigma V^T \Leftrightarrow \Sigma \underbrace{V^TCU}_{\widetilde{C}} \Sigma = \Sigma \Leftrightarrow$$

$$\begin{bmatrix} \Sigma_{11} & 0 \\ 0 & 0 \end{bmatrix} \underbrace{\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}}_{\widetilde{C}} \begin{bmatrix} \Sigma_{11} & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \Sigma_{11}C_{11}\Sigma_{11} & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \Sigma_{11} & 0 \\ 0 & 0 \end{bmatrix} \Leftrightarrow$$

$\Sigma_{11}C_{11}\Sigma_{11} = \Sigma_{11} \Leftrightarrow C_{11} = \Sigma_{11}^{-1}$ ja que els valors singulars son $\neq 0$.

Deduïm que, en termes de la SVD de A, una matriu C és inversa generalitzada si i només si

$$C = V\widetilde{C}U^T, \quad \widetilde{C} = \begin{bmatrix} \Sigma_{11}^{-1} & * \\ * & * \end{bmatrix}. \tag{7.1}$$

Adoneu-vos que aquesta caracterització permet molts graus de llibertat (les entrades de la matriu $\widetilde{C}$ fora de la submatriu $\widetilde{C}(1:r, 1:r)$ queden lliures). L'elecció més "simplificadora" per a les entrades lliures és prendre-les 0, amb la qual cosa tenim una SVD de la matriu:

$$C = V \begin{bmatrix} \Sigma_{11}^{-1} & 0 \\ 0 & 0 \end{bmatrix} U^T. \tag{7.2}$$

Aquesta matriu s'anomena *pseudoinversa de Moore-Penrose*, i ens la trobarem més endavant de manera natural en estudiar un altre tipus de condicions.

7.3 Inverses generalitzades reflexives

Quan la matriu A és invertible es verifica $(A^{-1})^{-1} = A$. De la mateixa manera, si $C = A^-$ ens preguntem si $A = C^-$, és a dir, si

$$ACA = A$$
$$CAC = C$$

Una inversa generalitzada d'aquest tipus s'anomena *inversa generalitzada reflexiva*. Exigir aquesta condició permet relacionar els blocs "lliures" de $\widetilde{C}$.

Teorema 7.3. Si $ACA = A$ aleshores són equivalents:

1. $CAC = C$
2. rang$C$ = rang$A$

Com que A és una inversa generalitzada, de (7.1) deduïm que

$$\text{rang}C = \text{rang}V\widetilde{C}U^T = \text{rang}\widetilde{C} \geq r,$$

ja que la submatriu $\widetilde{C}(1:r, 1:r)$ és invertible. Amb això, el que ens diu aquest resultat és que una inversa generalitzada reflexiva és una inversa generalitzada de rang mínim.

Prova. Transformem la igualtat $CAC = C$ amb les matrius U i V:

$$CAC = C \Rightarrow V^TCU\Sigma V^TCU = V^TCU \Rightarrow \widetilde{C}\Sigma\widetilde{C} = \widetilde{C} \Rightarrow$$
$$\begin{bmatrix}\Sigma_{11}^{-1} & C_{12}\\ C_{21} & C_{22}\end{bmatrix}\begin{bmatrix}\Sigma_{11} & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}\Sigma_{11}^{-1} & C_{12}\\ C_{21} & C_{22}\end{bmatrix} = \begin{bmatrix}\Sigma_{11}^{-1} & C_{12}\\ C_{21} & C_{22}\end{bmatrix} \Rightarrow$$
$$\begin{bmatrix}\Sigma_{11}^{-1} & C_{12}\\ C_{21} & C_{21}\Sigma_{11}C_{12}\end{bmatrix} = \begin{bmatrix}\Sigma_{11}^{-1} & C_{12}\\ C_{21} & C_{22}\end{bmatrix}$$

Deduïm que C és una inversa generalitzada si i només si $C_{21}\Sigma_{11}C_{12} = C_{22}$.

També és clar que

$$r = \text{rang}C = \text{rang}\widetilde{C} = \text{rang}\begin{bmatrix}\Sigma_{11}^{-1} & C_{12}\\ C_{21} & C_{22}\end{bmatrix}$$

si i només si les files $r+1, \ldots, n$ de $\widetilde{C}$ són combinació lineal de les files $1, \ldots, r$, és a dir,[1] si i només si existeix X tal que

$$X\begin{bmatrix}\Sigma_{11}^{-1} & C_{12}\end{bmatrix} = \begin{bmatrix}C_{21} & C_{22}\end{bmatrix} \Leftrightarrow X\Sigma_{11}^{-1} = C_{21}, XC_{12} = C_{22} \Leftrightarrow C_{21}\Sigma_{11}C_{12} = C_{22} \quad (X = C_{21}\Sigma_{11})$$

7.4 Projectors ortogonals

Si S és un subespai vectorial de $\mathbb{R}^n$, aleshores qualsevol $x \in \mathbb{R}^n$ es pot descompondre de manera única com

$$x = y + z \quad y \in S, z \in S^\perp \tag{7.3}$$

Aleshores, aquesta unicitat assegura que l'aplicació

$$P : x \in \mathbb{R}^n \mapsto P(x) = y \in S \subseteq \mathbb{R}^n$$

[1] Recordeu que les files d'un producte són combinació lineal de les files de la matriu de la dreta amb coeficients agafats de les files de l'esquerra.

és ben definida i lineal (perquè S i $S^\perp$ són subespais vectorials). Tal com és habitual, identifiquem P amb la seua matriu respecte de la base canònica, de manera que podem escriure $P(x) = Px$. Amb aquesta notació, l'equació (7.3) queda

$$x = \underbrace{Px}_{\in S} + \underbrace{(x - Px)}_{\in S^\perp}. \tag{7.4}$$

La matriu P rep el nom de *projector ortogonal* sobre S. Adoneu-vos que si $x \in S$ aleshores $Px = x$, ja que

$$x = \underbrace{x}_{\in S} + \underbrace{0}_{\in S^\perp}.$$

Amb això tenim que $S = \text{Im}P$. Aleshores tenim que P és un projector ortogonal (sobre Im P) si i només si $x - Px \in \text{Im}P^\perp \ \forall x \in \mathbb{R}^n$.

Teorema 7.4. Siga P una matriu $n \times n$, aleshores són equivalents:

1. P és un projector ortogonal (sobre Im P).
2. P és idempotent i simètrica:

$$P^2 = P$$
$$P^T = P$$

3. $P^2 = P$ i $N(P)^\perp = \text{Im}P$

Prova. Anomenen $S = \text{Im}P$. Aleshores tenim que:

$$\begin{aligned}
&P \text{ és projector ortogonal sobre } S \Leftrightarrow \\
&x - Px \in S^\perp, \ \forall x \in \mathbb{R}^n \Leftrightarrow \\
&y^T(x - Px) = 0, \ \forall x \in \mathbb{R}^n, y \in S \Leftrightarrow \\
&(Py)^T(x - Px) = 0, \ \forall x, y \in \mathbb{R}^n (\text{ ja que Im } P = S) \Leftrightarrow \\
&y^T P^T(I - P)x = 0, \ \forall x, y \in \mathbb{R}^n \Leftrightarrow \\
&P^T(I - P) = 0 \Leftrightarrow \\
&P^T = P^T P
\end{aligned}$$

Ara bé, com que $P^T P$ és sempre simètrica (ja que $(P^T P)^T = P^T P$), tenim que

$$P^T = P^T P \Leftrightarrow P = P^T \& P = P^2$$

Amb això provem $1 \Leftrightarrow 2$.

D'altra banda,

$$\begin{aligned}
&N(P)^\perp = \text{Im}P \Leftrightarrow \\
&N(P) = \text{Im}P^\perp \Leftrightarrow \\
&Px = 0 \Rightarrow z^T x = 0 \ \forall z \in \text{Im}P \Leftrightarrow \\
&Px = 0 \Rightarrow y^T P^T x (Py)^T x = 0 \ \forall y \in \mathbb{R}^n (z = Py) \Leftrightarrow \\
&Px = 0 \Rightarrow P^T x = 0
\end{aligned} \tag{7.5}$$

Aleshores, si suposem que $P = P^2$ i $P^T = P$, aleshores (7.5) és cert, amb la qual cosa $2 \Rightarrow 3$. Suposem ara que $P^2 = P$ i $N(P)^\perp = \text{Im}P$. De $P^2 = P$ deduïm que $P(Px - x) = 0\ \forall x \in \mathbb{R}^n$, amb la qual cosa $Px - x \in N(P)\ \forall x \in \mathbb{R}^n$. Per hipòtesi :

$$N(P) \perp \text{Im}P \Rightarrow Px - x \perp Py\ \forall x, y \in \mathbb{R}^n,$$

per la qual cosa, igual que abans, deduïm que $P^T = P^T P$ i d'ací que $P = P^T$ i, per tant $3 \Rightarrow 2$.

7.5 Inversa generalitzada minimitzadora de norma

Per definició, si $C = A^-$ i $Ax = b$ és compatible, aleshores Cb és solució de $Ax = b$, però pot ser-ho qualsevol si aquest sistema té múltiples solucions. Entre totes aquestes solucions, sabem que n'hi ha exactament una que té norma euclidiana mínima (vegeu el tema de la SVD, equació (6.22)). Una inversa generalitzada C s'anomena *minimitzadora de norma* (IGMN) quan Cb és la solució de norma mínima per a qualsevol sistema compatible $Ax = b$.

Teorema 7.5. Si A és una matriu, $C = A^-$ i $P = CA$ aleshores són equivalents:

1. C és IGMN de A
2. $N(A)^\perp = \text{Im}(P)$
3. P és projector ortogonal
4. P és una matriu simètrica.

Prova. Com a pas previ, vegem que si C és inversa generalitzada de A aleshores

- $P^2 = P$
- $N(A) = N(CA)$.

  Vegem la primera igualtat:

  $$P^2 = C(ACA) = CA = P$$

  Per veure la segona qüestió, el que és clar és $N(A) \subseteq N(CA)$ ($Ax = 0 \Rightarrow CAx = 0$); d'altra banda

  $$\begin{array}{l} x \in N(CA) \Rightarrow \\ CAx = 0 \Rightarrow \\ Ax = ACAx = 0 \text{ (perquè } C \text{ és inversa generalitzada de } A) \Rightarrow \\ x \in N(A) \end{array}$$

amb la qual cosa tenim establert que $N(A) = N(CA)$ quan $C = A^-$. Amb això, 2 és equivalent a

2'. $N(P)^\perp = \text{Im}(P)$.

Si x_0 és una solució particular d'un sistema compatible $Ax = b$, aleshores el conjunt de solucions d'aquest sistema és el subespai *afí* $x_0 + N(A)$. L'element del conjunt de solucions $x_0 + N(A)$ de norma mínima és $x_* = x_0 + y$ ($y \in N(A)$) tal que $x_* \in N(A)^\perp$. Deduïm que

$$\begin{array}{ll} C \text{ és IGMN } \Leftrightarrow & (7.6) \\ Cb \perp N(A)\ \forall z \in \mathbb{R}^n, b = Az \Leftrightarrow & \\ CAz \perp N(A)\ \forall z \in \mathbb{R}^n \Leftrightarrow & \\ \text{Im}(CA) \subseteq N(A)^\perp \Leftrightarrow & \\ \text{Im}(CA) \subseteq N(CA)^\perp \Leftrightarrow & \\ \text{Im}(CA) = N(CA)^\perp = N(A)^\perp & (7.7) \end{array}$$

(hem utilitzat $\dim N(CA)^\perp = n - \dim N(CA) = \dim \mathrm{Im}(CA)$ per obtenir (7.7)). Amb això establim l'equivalència entre 1 i 2.

L'equivalència entre 2', 3 i 4 és clara a partir del teorema 7.4.

Teorema 7.6. Siga $A = U\Sigma V^T$ una SVD de A $m \times n$ i C $m \times n$; aleshores són equivalents:

1. C és IGMN de A
2. $C = V\widetilde{C}U^T$, on $\widetilde{C} = \begin{bmatrix} \Sigma_{11}^{-1} & C_{12} \\ 0 & C_{22} \end{bmatrix}$.

Prova. Sabem que C és inversa generalitzada si i només si $\widetilde{C}(1:r,1:r) = \Sigma_{11}^{-1}$ on $\Sigma_{11} = \Sigma(1:r,1:r)$, $r = \mathrm{rang} A$. Aleshores, pel teorema 7.5 C és IGMN si i només si CA és simètrica i $\widetilde{C}(1:r,1:r) = \Sigma_{11}^{-1}$. Analitzem ara la matriu CA utilitzant la SVD de A: la matriu CA és simètrica si i només si V^TCAV ho és

$$V^T(CA)V = V^TCU\Sigma V^TV = \widetilde{C}\Sigma =$$
$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} \Sigma_{11} & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} C_{11}\Sigma_{11} & 0 \\ C_{21}\Sigma_{11} & 0 \end{bmatrix}$$

Tenim que

C és IGMN $\Leftrightarrow$
$C_{11} = \widetilde{C}(1:r,1:r) = \Sigma_{11}^{-1}$ i CA és simètrica $\Leftrightarrow$
$C_{11} = \widetilde{C}(1:r,1:r) = \Sigma_{11}^{-1}$ i V^TCAV és simètrica $\Leftrightarrow$
$C_{11} = \widetilde{C}(1:r,1:r) = \Sigma_{11}^{-1}$ i $C_{21}\Sigma_{11} = 0 \Leftrightarrow$
$C_{11} = \widetilde{C}(1:r,1:r) = \Sigma_{11}^{-1}$ i $C_{21} = 0$ (ja que Σ_{11} és invertible) $\Leftrightarrow$
$C = V \begin{bmatrix} \Sigma_{11}^{-1} & C_{12} \\ 0 & C_{22} \end{bmatrix} U^T$.

Comprovem ara aquest teorema, aprofitant que coneixem l'expressió de la solució de norma mínima d'un sistema compatible $Ax = b$, utilitzant la SVD de la matriu de coeficients:

$$\frac{\overline{b}_1}{\sigma_1}v_1 + \cdots + \frac{\overline{b}_r}{\sigma_r}v_r, \quad \overline{b} = U^Tb \tag{7.8}$$

Siga b un vector que fa compatible el sistema amb matriu A, és a dir, tal que $b \in \mathrm{Im}A$. Com que $\mathrm{Im}A = \langle u_1, \ldots, u_r \rangle$, tenim que $b = \overline{b}_1u_1 + \cdots + \overline{b}_ru_r$, $\overline{b}_i = 0$, $i = r+1, \ldots, m$, on $\overline{b} = U^Tb$ són les coordenades de b en la base ortonormal U. Podem escriure, doncs,

$$\overline{b} = \begin{bmatrix} c \\ 0 \end{bmatrix}$$

on c és un vector $r \times 1$.

Suposem ara que $C = V\widetilde{C}U^T$, amb $\widetilde{C} = \begin{bmatrix} \Sigma_{11}^{-1} & C_{12} \\ 0 & C_{22} \end{bmatrix}$, anomenem

$$V_1 = V(:, 1:r),$$
$$V_2 = V(:, r+1:n)$$

i calculem Cb:

$$Cb = V\widetilde{C}U^Tb = \begin{bmatrix} V_1 & V_2 \end{bmatrix} \begin{bmatrix} \Sigma_{11}^{-1} & C_{12} \\ 0 & C_{22} \end{bmatrix} \begin{bmatrix} c \\ 0 \end{bmatrix} = V_1\Sigma_{11}^{-1}c = \frac{\overline{b}_1}{\sigma_1}v_1 + \cdots + \frac{\overline{b}_r}{\sigma_r}v_r,$$

la qual és solució de norma mínima per (7.8).

7.6 Inversa generalitzada per a mínims quadrats

Definició 7.7. Donada la matriu A $m \times n$, una matriu C $n \times m$ s'anomena *inversa generalitzada per a resoldre mínims quadrats* (IGMQ) si Cb és solució del problema de mínims quadrats associat a $Ax = b$, $\forall b \in \mathbb{R}^m$.

Com que les solucions per mínims quadrats de problemes compatibles són exactament les solucions clàssiques, ens adonem que tota C IGMQ és inversa generalitzada, ja que si $Ax = b$ és compatible aleshores Cb és solució del problema de mínims quadrats associat, és a dir, Cb és solució clàssica del sistema lineal.

Lema 7.8.

$$A^TAX = 0 \Leftrightarrow AX = 0$$

Prova. Evidentment $AX = 0 \Rightarrow A^TAX = 0$. Ara, si $A^TAX = 0$ i y és un vector qualsevol, aleshores

$$A^TAXy = 0 \Rightarrow \|AXy\|_2 = y^TX^TA^TAXy = 0 \Rightarrow AXy = 0\ \forall y \Rightarrow AX = 0.$$

Teorema 7.9. Són equivalents:

1. C és IGMQ.
2. $A^TAC = A^T$.
3. $\begin{aligned} ACA &= A \\ (AC)^T &= AC \end{aligned}$

Prova. Com que les solucions del problema de mínims quadrats són exactament les solucions de les equacions normals, tenim que

$$\begin{aligned} &C \text{ és IGMQ} \Leftrightarrow \\ &Cb \text{ és solució de} A^TAx = A^Tb\ \forall b \Leftrightarrow \\ &A^TACb = A^Tb\ \forall b \Leftrightarrow \\ &A^TAC = A^T \end{aligned}$$

doncs $1 \Leftrightarrow 2$.

Per veure $3 \Rightarrow 2$:

$$\begin{aligned} &(CA)^T = C^TA^T = AC \Rightarrow \\ &C^TA^TA = ACA = A \Rightarrow \\ &A^TAC = (C^TA^TA)^T = A^T \end{aligned}$$

Suposem ara que $A^TAC = A^T$. Vegem primer que $ACA = A$: multipliquem per la dreta per A, ho passem tot al primer membre i apliquem el lema 7.8:

$$A^TA(CA - I) = A^TACA - A^TA = A^TA - A^TA = 0 \Rightarrow ACA - A = 0 \Rightarrow ACA = A$$

Vegem ara que $(AC)^T = AC$: multipliquem per C^T per l'esquerra:

$$\begin{aligned} C^TA^TAC = C^TA^T &\Rightarrow \\ (AC)^T(AC) = (AC)^T &\Rightarrow \\ AC \text{ simètrica.} & \end{aligned}$$

Caracteritzem ara les IGMQ amb la SVD.

Teorema 7.10. Són equivalents:

1. $\begin{matrix} ACA = A \\ (AC)^T = AC \end{matrix}$
2. $C = V \underbrace{\begin{bmatrix} \Sigma_{11}^{-1} & 0 \\ C_{21} & C_{22} \end{bmatrix}}_{\widetilde{C}} U^T.$

Prova. Sabem ja que $ACA \Leftrightarrow \widetilde{\Sigma}(1:r, 1:r) = \Sigma_{11}^{-1}$.

Calculem ara AC en funció de la SVD:

$$AC = U\Sigma V^T V\widetilde{C}U^T =$$
$$U\begin{bmatrix} \Sigma_{11} & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} \Sigma_{11}^{-1} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}U^T = U\begin{bmatrix} \Sigma_{11}\Sigma_{11}^{-1} & \Sigma_{11}C_{12} \\ 0 & 0 \end{bmatrix}U^T = U\begin{bmatrix} I_r & \Sigma_{11}C_{12} \\ 0 & 0 \end{bmatrix}U^T$$

Aleshores AC és simètrica si i només si $\Sigma_{11}C_{12} = 0$ si i només si $C_{12} = 0$.

7.7 Inversa generalitzada per a solucions de norma mínima dels mínims quadrats

Aquest últim tipus d'inversa generalitzada (IGMNMQ) es defineix com aquella matriu C tal que Cb és solució de norma mínima del problema de mínims quadrats associat a qualsevol sistema $Ax = b$.

Com que la solució del problema de mínims quadrats coincideix amb la de les equacions normals $A^TAx = A^Tb$, deduïm que

C és IGMNMQ $\Leftrightarrow$
Cb és solució de norma mínima de $A^TAx = A^Tb\ \forall b \Leftrightarrow$
$A^TACb = A^Tb$ i $Cb \perp N(A^TA)\ \forall b \Leftrightarrow$
$A^TAC = A^T$ i Im$C \perp N(A^TA) \Leftrightarrow$
$A^TAC = A^T$ i Im$C \subseteq N(A^TA)^\perp$

Tenim que $N(A^TA) = N(A)$ ($Ax = 0 \Leftrightarrow A^TAx = 0$) i $N(A) = N(CA)$ (ja que $ACA = A$).

Com que $A^TAC = A^T$ aleshores C és IGMQ, en particular és inversa generalitzada, doncs sabem que rang$C \geq$ rang$A$. En aquest cas, la relació Im$C \subseteq N(A)^\perp$ dóna la desigualtat de les dimensions respectives:

$$\text{rang}(A) \leq \dim \text{Im}C \leq \dim N(A)^\perp = n - \dim N(A) = \text{rang}(A).$$

Aleshores

C és IGMNMQ $\Leftrightarrow$
$A^TAC = A^T$ i Im$C = N(CA)^\perp$.

D'altra banda, per la propietat minimitzadora sabem que CA és projector ortogonal, per la qual cosa $N(CA) = \text{Im}(I - CA)$. Per tant

C és IGMNMQ $\Leftrightarrow$
$A^TAC = A^T$ i Im$C = \text{Im}(I - CA)^\perp \Leftrightarrow$
$A^TAC = A^T$ i $x^TC^T(I - CA)y = (Cx)^T(I - CA)y = 0\ \forall x, y \Leftrightarrow$
$A^TAC = A^T$ i $C^T(I - CA) = 0 \Leftrightarrow$
$A^TAC = A^T$ i $C^T = C^TCA \Leftrightarrow$
C és IGMQ de A i A és IGMQ de C (pel teorema 7.9)$\Leftrightarrow$

$$ACA = 0 \tag{7.9}$$
$$CAC = C \tag{7.10}$$
$$(CA)^T = CA \tag{7.11}$$
$$(AC)^T = AC \tag{7.12}$$

Per a la caracterització d'aquest tipus d'inversa generalitzada en termes de la SVD, utilitzem el que hem vist abans. Per (7.9) sabem que $C_{11} = \Sigma_{11}^{-1}$, per (7.11) $C_{21} = 0$, per (7.10) $C_{12} = 0$ i per (7.10) $C_{22} = C_{21}\Sigma_{11}C_{12} = 0$. Deduïm que C és IGMNMQ si i només si

$$C = A^\dagger = V \begin{bmatrix} \Sigma_{11}^{-1} & 0 \\ 0 & 0 \end{bmatrix} U^T \tag{7.13}$$

Aleshores existeix una única IGMNMQ, donada per l'equació (7.13), la qual s'anomena *pseudoinversa de Moore-Penrose* de A.

Comprovem, per acabar, que $A^\dagger b$ és la solució de norma mínima del problema de mínims quadrats associat a $Ax = b$, per a qualsevol b. Sabem que aquesta solució és:

$$\frac{\overline{b}_1}{\sigma_1}v_1 + \cdots + \frac{\overline{b}_r}{\sigma_r}v_r$$

D'altra banda:

$$A^\dagger b = V \begin{bmatrix} \Sigma_{11}^{-1} & 0 \\ 0 & 0 \end{bmatrix} U^T b = \begin{bmatrix} V_1 & V_2 \end{bmatrix} \begin{bmatrix} \Sigma_{11}^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = V_1\Sigma_{11}^{-1}c_1 = \frac{\overline{b}_1}{\sigma_1}v_1 + \cdots + \frac{\overline{b}_r}{\sigma_r}v_r,$$

on $V = \begin{bmatrix} V_1 & V_2 \end{bmatrix}$, $\overline{b} = U^T b = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$.

7.8 Problemes

Problema 1. Si A és una matriu $m \times n$ amb rang$A = n$ demostreu de dues maneres diferents que $C = (A^TA)^{-1}A^T$ és la pseudoinversa de Moore-Penrose de A, demostrant prèviament que C està ben definida (és a dir, que A^TA és invertible).

Problema 2. Siga A una matriu $m \times n$ i P matriu invertible de dimensió $m \times m$. Demostreu:

1. C és inversa generalitzada de A si i només si CP^{-1} és inversa generalitzada de PA (simbòlicament: $(PA)^- = A^-P^{-1}$)
2. C és inversa generalitzada reflexiva de A si i només si CP^{-1} és inversa generalitzada reflexiva de PA.
3. C és IGMN de A si i només si CP^{-1} és IGMN de PA.
4. Sota quina hipòtesi sobre P seria cert que C és IGMQ de A si i només si CP^{-1} és IGMQ de PA.

Tema 8

Diferenciació matricial

En aquest tema obtindrem fórmules algebraiques per a la diferenciació de funcions escalars/vectorials de variables vectorials/matricials.

8.1 Notació

Siga una funció $\alpha\colon \mathbb{R}^m \to \mathbb{R}$ escalar de m variables. Designem les variables com a $x = (x_1, \ldots, x_m)$. Definim

$$\alpha' = \frac{\partial \alpha}{\partial x} = \begin{bmatrix} \frac{\partial \alpha}{\partial x_1} \\ \vdots \\ \frac{\partial \alpha}{\partial x_m} \end{bmatrix}.$$

Siga una funció $y\colon \mathbb{R}^m \to \mathbb{R}^n$ vectorial de m variables. Designem les variables com a $x = (x_1, \ldots, x_m)$. Definim

$$y' = \frac{\partial y}{\partial x} = \begin{bmatrix} \frac{\partial y^1}{\partial x_1} & \cdots & \frac{\partial y^n}{\partial x_1} \\ \vdots & & \\ \frac{\partial y^1}{\partial x_m} & \cdots & \frac{\partial y^n}{\partial x_m} \end{bmatrix}.$$

Siga una funció $\alpha\colon \text{matrius}(m,n) \to \mathbb{R}$ escalar amb variables matricials. Designem les variables com a

$$X = \begin{bmatrix} X_{1,1} & \ldots & X_{1,n} \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ X_{m,1} & \ldots & X_{m,n} \end{bmatrix}.$$

Definim

$$\alpha' = \frac{\partial \alpha}{\partial X} = \begin{bmatrix} \frac{\partial \alpha}{\partial X_{1,1}} & \cdots & \frac{\partial \alpha}{\partial X_{1,n}} \\ \vdots & & \\ \frac{\partial \alpha}{\partial X_{m,1}} & \cdots & \frac{\partial \alpha}{\partial X_{m,n}} \end{bmatrix}.$$

8.2 Fórmules per a la diferenciació matricial

Proposició 8.1. Siguen α, β funcions escalars de m variables; aleshores

$$(\alpha\beta)'(x) = \beta(x)\alpha'(x) + \alpha(x)\beta'(x).$$

Prova. Per la regla de la diferenciació del producte de dues funcions:

$$((\alpha\beta)'(x))(i) = \frac{\partial \alpha\beta}{\partial x_i} = \frac{\partial \alpha}{\partial x_i}\beta + \alpha\frac{\partial \beta}{\partial x_i} = \beta(x)\alpha'(x)(i) + \alpha(x)\beta'(x)(i) \Rightarrow$$
$$(\alpha(x)\beta(x))' = \beta(x)\alpha'(x) + \alpha(x)\beta'(x)$$

Proposició 8.2. Siga a vector $m \times 1$ i $\alpha(x) = a^T x$. Aleshores $\alpha' = a$.

Prova.

$$\alpha(x) = a^T x = a_1 x_1 + \cdots + a_m x_m \Rightarrow \frac{\partial \alpha}{\partial x_i} = a_i \Rightarrow \alpha' = \begin{bmatrix} a_1 \\ \vdots \\ a_m \end{bmatrix} = a$$

Proposició 8.3. Siga $b\colon \mathbb{R}^m \to \mathbb{R}^n$, a vector $n \times 1$ i $\alpha(x) = a^T b(x)$. Aleshores $\alpha' = b'(x)a$.

Prova. Adoneu-vos que $b'(x)$ és una matriu $m \times n$.

$$\alpha(x) = a^T b(x) = a_1 b_1(x) + \cdots + a_n b_n(x) \Rightarrow$$
$$(\alpha'(x))(i) = \frac{\partial \alpha}{\partial x_i} = a_1\frac{\partial b_1}{\partial x_i} + \cdots + a_n\frac{\partial b_n}{\partial x_i} = (b'(x))(i,:)a = (b'(x)a)(i) \Rightarrow$$
$$\alpha'(x) = b'(x)a$$

Proposició 8.4. Si A és una matriu $n \times m$ i $y\colon \mathbb{R}^m \to \mathbb{R}^n$, $y(x) = Ax$, aleshores $y'(x) = A^T$.

Prova. Per definició

$$(y'(x))(i,j) = \frac{\partial y_j}{\partial x_i}.$$

$$y_j = (Ax)(j) = A(j,:)x = A(j,1)x_1 + \cdots + A(j,m)x_m \Rightarrow$$
$$\frac{\partial y_j}{\partial x_i} = A(j,i) \Rightarrow$$
$$(y'(x))(i,j) = A(j,i) = (A^T)(i,j) \Rightarrow$$
$$y'(x) = A^T$$

Proposició 8.5. Siga $a, b\colon \mathbb{R}^m \to \mathbb{R}^n$ i $\alpha(x) = a(x)^T b(x)$. Aleshores $\alpha' = a'b + b'a$.

Prova.

$$\begin{aligned}
\alpha &= a_1(x)b_1(x) + \cdots + a_n(x)b_n(x) \Rightarrow \\
(\alpha'(x))(i) &= \frac{\partial \alpha}{\partial x_i} \\
&= \frac{\partial a_1}{\partial x_i}b_1(x) + \cdots + \frac{\partial a_n}{\partial x_i}b_n(x) + a_1(x)\frac{\partial b_1}{\partial x_i} + \cdots + a_n(x)\frac{\partial b_n}{\partial x_i} \\
&= a'(i,1)b_1(x) + \cdots + a'(i,n)b_n(x) + a_1(x)b'(i,1) + \cdots + a_n(x)b'(i,n) \\
&= a'(i,:) * b + b'(i,:)a = (a'b + b'a)(i) \Rightarrow \\
\alpha' &= a'b + b'a
\end{aligned}$$

Exemple 8.6.

1. Per la proposició 8.4: $y(x) = x = I * x \Rightarrow y' = I^T = I$.
2. Per la proposició 8.5: $\alpha(x) = \|x\|_2^2 = (\underbrace{x}_{a(x)})^T \underbrace{x}_{b(x)} \Rightarrow \alpha' = a'b + b'a = Ix + Ix = 2x$
3. Per la proposició 8.5:

$$\alpha(x) = (\underbrace{x}_{a(x)})^T \underbrace{Ax}_{b(x)} \Rightarrow \alpha'(x) = a'b + b'a = IAx + A^T x = (A + A^T)x.$$

   Si A és simètrica, aleshores $\alpha'(x) = 2Ax$.

Proposició 8.7. Siga $\alpha(X) = \text{tr}(X) = \sum_{i=1}^n X_{i,i}$, on X és $n \times n$. Aleshores $\alpha'(X) = I_n$.

Prova. Per definició:

$$\alpha'(X)(i,j) = \frac{\partial \alpha}{\partial X_{ij}} = \delta_{ij} = (I_n)(i,j) \Rightarrow \alpha'(X) = I_n$$

Proposició 8.8. Siga $\alpha(X) = \det(X)$, on X és $n \times n$. Aleshores

$$\alpha'(X) = \text{adj}(X) = \det(X)X^{-T},$$

on adj(X) és la matriu adjunta de X.

Prova. Per calcular $\alpha'(X)(i,j) = \frac{\partial \det(X)}{\partial X_{ij}}$, desenvolupem el determinant pels adjunts de la fila i:

$$\det(X) = X_{i1}\text{adj}(X)_{i1} + \cdots + X_{ij}\text{adj}(X)_{ij} + \cdots + X_{in}\text{adj}(X)_{in}$$

Com que en el segon membre d'aquesta expressió els adjunts no depenen de X_{ij}, en derivar respecte de X_{ij} obtenim

$$\frac{\partial \det(X)}{\partial X_{ij}} = \text{adj}(X)_{ij} \Rightarrow \frac{\partial \det(X)}{\partial X} = \text{adj}(X) = \det(X)X^{-T},$$

on hem tingut en compte que $X^{-1} = \text{adj}(X)^T / \det(X)$.

8.3 Problemes

Problema 1. Siga D una matriu $n \times n$ diagonal. Mitjançant càlcul diferencial matricial deduïu quins són el mínim i el màxim de la funció *quocient de Rayleigh*

$$\lambda(x) = \frac{x^T Dx}{x^T x}$$

Problema 2. Siga A una matriu $m \times n$ i b un vector $m \times 1$. Mitjançant càlcul diferencial matricial deduïu que les equacions normals són condició necessària per a la solució del problema dels mínims quadrats:

$$\min_x \|Ax - b\|_2^2.$$

Problema 3. Siga A una matriu $m \times n$ i b un vector $m \times 1$. Mitjançant càlcul diferencial matricial trobeu condicions necessàries per a la solució del problema d'optimització

$$\min_{Ax=b} \|x\|_2^2$$

Índex de termes

Bibliografia

[1] ANDERSON, E., Z. BAI, ET AL., *Lapack Users Guide*, SIAM, 1995.

[2] ARÁNDIGA, F., R. DONAT I P. MULET, *Mètodes numèrics per a l'àlgebra lineal*, Universitat de València, 2000.

[3] AXELSSON, O., *Iterative Solution Methods*, Cambridge University Press, NY, 1994.

[4] BASILEVSKY, A., *Applied Matrix Algebra in the Statistical Sciences*, Dover Publications, 2005.

[5] BURDEN, R., J. FAIRES, *Análisis Numérico*, Thomson Learning, México, 2002.

[6] CHATELIN, F., *Eigenvalues of matrices*, Jonh Willey and Sons, 1993.

[7] COLEMAN, T., C. V. LOAN, *Handbook for Matrix Computations*, SIAMPub, Philadelphia, PA, 1988.

[8] DEMMEL, J. W., *Applied Numerical Linear Algebra*, SIAM, 1997.

[9] FORSYTHE, G., C. MOLER, *Computer Solution of Linear Algebraic Systems*, PrenticeHall, Englewood Cliffs, NJ, 1967.

[10] GENTLE, J. E., *Matrix Algebra: Theory, Computations, and Applications in Statistics*, Springer, 2000.

[11] GOLUB, G., J. ORTEGA, *Scientific Computing and Differential Equations*, Academic Press, 1992.

[12] GOLUB, G., C. VAN LOAN, *Matrix Computations*, JohnsHopkinsPress, Baltimore, MD, segunda ed., 1989.

[13] HAGEMAN, L., D. YOUNG, *Applied Iterative Methods*, Academic Press, New York, NY, 1981.

[14] HARVILLE, D. A., *Matrix Algebra From a Statistician's Perspective*, Springer, 2000.

[15] LEVEQUE, R. J., *Finite Difference Methods for Ordinary and Partial Differential Equations*, SIAMPub, Philadelphia, PA, 2007.

[16] PARLET, B., *The Symetric Eigenvalue Problem*, PrenticeHall, Englewood Cliffs, NJ, 1980.

[17] SAAD, Y., *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, Boston, MA, 1996.

[18] SEARLE, S. R., *Matrix Algebra Useful for Statistics*, Wiley-Interscience, 2006.

[19] STRANG, G., *Linear Algebra and Its Applications*, Academic Press, Orlando, second ed., 1976.

[20] ——, *Introduction to Applied Mathematics*, Cambridge Press, Wellesley, MA, 1986.

[21] TREFETHEN, L. N., I. DAVID BAU, *Numerical Linear Algebra*, SIAM, 1997.

[22] VARGA, R., *Matrix Iterative Analysis*, PrenticeHall, Englewood Cliffs, NJ, 1962.

[23] WILKINSON, J., *The Algebraic Eigenvalue Problem*, Oxford University Press, (Clareton), London and New York, 1965.

[24] YOUNG, D., *Iterative Solutions of Large Linear Systems*, Academic Press, New York, NY, 1971.