DEPARTAMENTO DE MATEMÁTICA APLICADA

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Métodos iterativos en s-pasos para a resolución de grandes sistemas dispersos de ecuacións e a súa implementación paralela

GERARDO CASAL URCERA

Xaneiro, 2012

Don José Antonio Álvarez Dios, profesor titular do Departamento de Matemática Aplicada da Universidade de Santiago de Compostela, e Don José Carlos Cabaleiro Domínguez, profesor titular do Departamento de Electrónica e Computación da Universidade de Santiago de Compostela, informan que a memoria titulada:

Métodos iterativos en s-pasos para a resolución de grandes sistemas dispersos de ecuacións e a súa implementación paralela

foi realizada baixo a súa dirección por Don Gerardo Casal Urcera, estimando que o interesado atópase en condicións de optar ao grao de Doutor en Ciencias Matemáticas, polo que solicitan que sexa admitida a trámite para a súa lectura e defensa pública.

En Santiago de Compostela, a 29 de decembro de 2011

Os directores:

José Antonio Álvarez Dios

José Carlos Cabaleiro Domínguez

O doutorando:

Gerardo Casal Urcera

Ás persoas maiores gústanlles as cifras.
Cando lles falades dun novo amigo, nunca
vos preguntan nada esencial del.
Non vos din: "Como é a súa voz?,
Que xogos lle gustan máis?, Colecciona
bolboretas?". Non, as persoas maiores preguntan:
"Cal é a súa idade?, Cantos irmáns ten?,
Canto pesa?, Canto gaña o seu pai?".
Pensan que soamente así o poden coñecer.

O Principiño.
Antoine de Saint-Exupéry.

A Veva e a Xoana

Agradecementos

Quixera mostrar o meu agradecemento aos profesores José Antonio Álvarez Dios, do Departamento de Matemática Aplicada, e José Carlos Cabaleiro Domínguez, do Departamento de Electrónica e Computación, ámbolos dous da Universidade de Santiago de Compostela e directores deste traballo. O tempo que lle dedicaron, as súas directrices e correccións fixeron posible a consecución do mesmo. Ao profesor Tomás Fernández Pena polas súas aportacións e interese mostrado neste tema. Así mesmo gustaríame tamén agradecer ao CESGA (Centro de Supercomputación de Galicia) por permitirme e facilitarme o uso das computadoras e software do centro. Este agradecemento fágoo extensible ao persoal do mesmo polo asesoramento técnico ofrecido en momentos de necesidade, sempre de xeito inmediato, eficaz e amable.

Debo dar as grazas ao Departamento de Matemática Aplicada da Universidade de Santiago de Compostela por posibilitar o meu traballo nesta tese ao longo destes anos, en particular á directora e á secretaria do mesmo, as profesoras Dolores Gómez e Patricia Barral, polo apoio recibido. De xeito tamén especial aos meus compañeiros do Departamento no Campus de Lugo, Manuel, Juan, Isabel, Pili, Duarte, Miguel Ernesto, Miguel Vilar e Jose Ramón, que xa nos abandonou por terras máis cálidas. Con eles atopei o mellor ambiente de traballo, ademais de compañeiros inmellorables e amigos de ceas e saídas varias. A miña chegada a este xélido Lugo foi doada grazas a eles e a outros moitos compañeiros e compañeiras da Escola Politécnica Superior e doutras facultades do Campus de Lugo (Ana, a outra Ana, Mar, Marcelo, Loli, Bernardo, Xose Manuel, Montse, Luís e un largo etcétera). Todos eles teñen a culpa de que acabara sentindo esta terra como a miña.

Aínda que xa pasaron uns cantos anos non podo esquecer o tempo que traballei como profesor do Departamento de Economía Financiera e Actuarial da Universidade Complutense de Madrid e ao seu director nos últimos anos que eu pasei alí,

o catedrático Sinesio Gutierrez Valdeón, un dos mellores profesores que coñecín e mellor persoa. A miña admiración por el serviume de guía no meu labor docente en todo este tempo. Souben tristemente do seu falecemento uns anos despois de deixar o seu Departamento. Quero tamén dar as grazas ao profesor Javier Lafuente López, do Departamento de Geometría y Topología da Universidad Complutense de Madrid, el foi quen me iniciou no traballo de investigación e do que aprendín moito do que sei agora.

Para rematar, grazas Veva, por aguantar sen queixas, por preocuparte, por coidarme e por quererme así, sen máis, e deixar que eu poida facer o mesmo. E á nosa pequena Xoana, que agora non entende, mais espero que algún día o faga e saiba perdoar que esta foi a causa de que o seu pai non pasara nestes meses máis tempo xogando con ela.

Lugo, 29 de decembro de 2011

Este traballo foi financiado pola Xunta de Galicia a través dos proxectos de investigación PGIDIT02PXIA20701AF e PGIDIT06RMA23501PR, e contou co soporte informático do Centro de Supercomputación de Galicia (CESGA).

Índice xeral

Introdución

Na formulación matemática das leis que rixen moitos dos problemas físicos figuran, con frecuencia, ecuacións en derivadas parciais de orde igual ou superior a dous. A resolución analítica destas ecuacións é moi difícil ou imposible. Esta imposibilidade creou a necesidade do desenvolvemento de métodos para obter unha solución numérica aproximada por procedementos axeitados a cada problema concreto.

A solución numérica dun problema con formulación en derivadas parciais pasa por un proceso de discretización (con diferenzas finitas, elementos finitos, elementos de contorno, volumes finitos,...) que nos permita avaliar a solución nun número finito de puntos do dominio. Esta discretización a miúdo conduce á resolución dun gran sistema de ecuacións lineares onde as incógnitas son precisamente estes valores numéricos puntuais da solución aproximada do problema físico que orixina o sistema. En gran parte destes problemas a matriz de coeficientes do sistema de ecuacións é de orde moi elevada, pero á vez con moitos máis ceros que elementos non nulos (matrices dispersas ou *sparse matrices*).

O uso do cálculo vectorial e paralelo, a aparición de matrices dispersas e os problemas asociados aos erros de redondeo e o efecto de recheo que se producen na aplicación dos métodos directos, fan máis axeitados os métodos iterativos para a resolución destes grandes sistemas. Entre estes métodos, teñen especial relevancia os algoritmos baseados nos subespazos de Krylov.

Os métodos apropiados para aproximar a solución foron desenvolvéndose simultáneamente á evolución dos computadores, de xeito que a eficiencia dun método depende en gran parte da súa facilidade de implementación e a súa adaptación á arquitectura da computadora utilizada. Deste xeito métodos como o do Gradente Conxugado, publicado por Hestenes e Stiefel en 1952 [48, 10, 79, 73], adquiriron re-

levancia nos últimos trinta años, que é cando atopou o soporte informático axeitado nos computadores vectoriais e paralelos. O principal inconveniente destes métodos é que no caso dun sistema con matriz non simétrica son mal condicionados.

Por outra banda, o gran tamaño destes sistemas fixo necesario o uso de novas arquitecturas en computación como a vectorial ou a paralela. A idea da computación paralela é simple: consiste en diminuír o tempo de execución dun programa ou aplicación repartindo o traballo computacional de xeito que varias instrucións poidan ser executadas á vez. Cada unha destas instrucións execútase nunha unidade de computación, que pode ser un fío (*thread*) ou un proceso, e que corre nun dos núcleos dun procesador ou mesmo nun procesador. Para iso, ademais do hardware, foron desenvolvéndose diversos proxectos para a creación do software necesario na programación paralela, como son HPF [54] (*High Performance Fortran*), PVM [41] (*Parallel Virtual Machine*), MPI [1, 38, 62, 66] (*Message Passing Interface*), OpenMP [1, 20, 21, 63], UPC [37] e outros.

Entre os métodos de resolución de sistemas de ecuacións lineares, os primeiros candidatos á implementación en arquitecturas paralelas son os métodos iterativos clásicos [64], como o método de *Jacobi*, usado na resolución da ecuación de Laplace polo método de diferenzas finitas e doadamente paralelizable, ou esquemas de relaxación de Jacobi como o *Gauss-Seidel* e a *ordenación en vermello e negro*. Outros métodos que tamén se encontran entre estes primeiros candidatos á implementación en paralelo son os baseados en subespazos de Krylov. Traballouse ademais na busca de técnicas para acelerar a execución destes métodos, xa sexa incluíndo precondicionadores [42, 72], ou o uso de *wavelets* para modificar o sistema linear noutro máis disperso [43], ou modificando os métodos para obter un mellor rendimento no procesamento paralelo [22, 23, 24, 28, 31]. Entre estes últimos estarían as variantes en s-pasos obxecto de xeneralización e estudo nesta tese.

En todos estes métodos iterativos, ademais de produtos matriz dispersa por vector, a maioría das operacións utilizadas son produtos vector-vector e combinacións lineares de vectores. Na librería de programas de álxebra linear BLAS (*Basic Linear Algebra Subprograms* [57]), escrita en Fortran e posteriormente traducida a C, referímosnos a estas rutinas comunmente polos seus nomes SDOT e SAXPY, pertencentes ao nivel 1 de BLAS (BLAS 1). En programación paralela resultan máis

eficientes as operacións entre matriz-vector e aínda máis matriz-matriz, pertencentes estas aos niveis BLAS 2 e BLAS 3, respectivamente. Isto é debido a que, cos últimos niveis de BLAS, aumenta a razón entre o número de operacións e os accesos á memoria do computador e/ou as comunicacións. Polo tanto, nun computador paralelo con núcleos BLAS paralelizados resultará tanto máis eficiente a implementación en paralelo cantas máis operacións haxa pertencentes ao maior nivel de BLAS posible. Co recente auxe e desenvolvemento das arquitecturas multinúcleo (*multicore*), nestes últimos anos publicáronse traballos [9, 31] que seguen esta mesma liña consistente en diminuír as comunicacións fronte ao número de operacións.

Un intento por substituír unhas operacións de BLAS por outras dun nivel máis alto constitúeno as diversas variantes en s-pasos de métodos iterativos coñecidos ou métodos iterativos en s-pasos (*s-step iterative methods*) descritos por Chronopoulos e outros autores en [24] e [23] basicamente. A idea principal consiste en, a partir de métodos clásicos de subespazos de Krylov como o Gradente Conxugado, o Orthomin(m), etc, xerar en cada iteración un bloque de s direccións que corresponderán, en certo modo, as s iteracións xeradas polo método orixinal. A eficiencia que se gaña na implementación paralela é corroborada en [25] e [29]. Nun traballo recente [28], o mesmo Chronopoulos en colaboración con Kucherov, retoman estas técnicas para propoñer métodos iterativos en s-pasos para sistemas lineares con varios termos independentes, coñecidos como métodos en bloques.

O obxectivo desta tese é construír un marco teórico que xeneralice os métodos iterativos en s-pasos coñecidos. A partir deste marco teórico obtéñense novas variantes doutros métodos que amplían o abano dos tipos de matrices nos que converxen. En particular propóñense novos métodos iterativos en s-pasos converxentes para calquera matriz cadrada non singular, e polo tanto, non necesariamente simétrica. Ofrécense tamén resultados numéricos obtidos na programación paralela destes métodos nas computadoras do Centro de Supercomputación de Galicia (CESGA).

No primeiro capítulo desta tese preséntanse os antecedentes que deron lugar a este traballo, comenzando cuns preliminares nos que se establece a notación necesaria. Lémbranse tamén algúns conceptos básicos e resultados coñecidos que deben ser tidos en conta ao longo deste traballo. Posteriormente, nunha sección deste primeiro capítulo, resúmense os métodos iterativos en s-pasos propostos por Chronopoulos

e outros autores nas súas publicacións. Na última parte deste capítulo descríbese o Algoritmo Xeral de Ortogonalización (AXO) seguindo a referencia [51], onde aparece proposto, ou [4] onde tamén se desenvolve. Este algoritmo xeneraliza os métodos tipo Gradente Conxugado, e poden obterse, mediante a axeitada substitución de dúas matrices parámetro, métodos coñecidos como o Gradente Conxugado, Residuo Conxugado, Erro Minimal, Ecuación Normal, Residuo Minimal, etc.

No capítulo 2 proponse unha variante en s-pasos do Algoritmo Xeral de Ortogonalización, que se denotará por s-AXO. Este método en s-pasos consiste en aplicar as técnicas dos métodos en s-pasos de Chronopoulos ao Algoritmo Xeral de Ortogonalización que foron expostos no capítulo anterior. Do mesmo xeito que o AXO xeneraliza os métodos tipo Gradente Conxugado, o s-AXO xeneraliza as variantes en s-pasos destes mesmos métodos. Constrúese tamén o marco teórico onde se encadra esta variante, demostrando algúns lemas de ortogonalidade e un teorema de converxencia, así como da estimación do erro. Así mesmo dáse unha variante en s-pasos para a versión Orthomin deste algoritmo xunto cun teorema de converxencia. Na última sección deste capítulo proponse unha versión do s-AXO no que se aplica o método modificado de Gram-Schmidt para ortonormalizar, respecto dunha determinada matriz, os vectores calculados en cada iteración do método. Isto faise co obxectivo de gañar estabilidade nestas variantes en s-pasos, e baseándose no traballo realizado en [29].

No capítulo 3 expóñese os métodos iterativos en s-pasos particulares que se obteñen substituíndo no s-AXO as dúas matrices parámetro por matrices concretas. Algúns destes métodos foron xa propostos anteriormente, como a variante en s-pasos do Gradente Conxugado, a do Gradente Conxugado Precondicionado, a do Residuo Conxugado, a da Ecuación Normal ou a do Residuo Conxugado Xeneralizado (e a do Residuo Minimal como a versión Orthomin(1) deste método). Se ben as versións que neste capítulo se presentan son aquelas nas que se aplica o método modificado de Gram-Schmidt para ortonormalizar os vectores calculados en cada iteración. Entre os casos particulares obtidos do s-AXO, xorde unha variante en s-pasos non coñecida ata o momento que é a do Erro Minimal, e que resulta converxente para calquera tipo de matriz cadrada non singular e non necesariamente simétrica.

No capítulo 4 proponse unha variante en s-pasos dunha segunda forma do Algo-

ritmo Xeral de Ortogonalización que tamén vén descrito nas referencias [51] e [4]. Esta segunda forma é equivalente á primeira, mais dela pódese obter un método coñecido como o Algoritmo da Dobre Serie Ortogonal, proposto en [5], e que é converxente para calquera matriz cadrada non singular. En consecuencia, neste capítulo proponse unha nova variante en s-pasos dun método concreto que é o da Dobre Serie Ortogonal, e que é tamén converxente para calquera matriz cadrada non singular.

No capítulo 5 proponse unha nova variante en s-pasos, a do Gradente Biconxugado que, se non dexenera, converxe para matrices cadradas non singulares.

No capítulo 6 expóñense os resultados numéricos obtidos coa programación paralela destes métodos iterativos en s-pasos na máquina Finis Terrae do Centro de Supercomputación de Galicia (CESGA). Os códigos dos programas son realizados en Fortran 95 e usando directivas OpenMP para a súa paralelización. Na súa execución utilizáronse matrices exemplo obtidas da Harwell-Boeing Collection na páxina web [61].

A memoria finaliza cunha exposición das conclusións, facendo fincapé nas aportacións orixinais incluídas na mesma. Trátase neste capítulo tamén de indicar as posibles liñas de investigación futuras que derivan deste traballo, nas que haberá que abordar algunhas das cuestións pendentes que quedan por completar ou resolver.

Capítulo 1

Preliminares

Neste capítulo establécese a notación utilizada nesta memoria así como os conceptos básicos e resultados necesarios para abordar os contidos dos seguintes capítulos. Na primeira sección introdúcese a notación e expoñense algúns resultados básicos. Nunha segunda sección preséntanse, de xeito resumido, os métodos iterativos en s-pasos publicados por Chronopoulos [23, 24, 28]. Finalmente, tamén de xeito resumido, descríbese na terceira sección o Algoritmo Xeral de Ortogonalización publicado en [51].

1.1. Notacións e resultados básicos

A partir de agora suporase que A é en xeral unha matriz real cadrada e non singular de orde n e $\|\cdot\|$ denota a norma 2 en $\mathbb{R}^n$ [18, 72]. Dise que A é dispersa se ten a maior parte dos seus termos nulos. Denótase por $\mathcal{M}_{n\times s}(\mathbb{R})$ a álxebra das matrices reais de orde $n \times s$ coas operacións habituais. A parte simétrica e antisimétrica de A denótase por A^S e A^{aS} respectivamente e defínese como:

$$A^S = \frac{1}{2}(A + A^t) \tag{1.1}$$

$$A^{aS} = \frac{1}{2}(A - A^t). \tag{1.2}$$

Se M é unha matriz cadrada de orde n calquera, entón a norma matricial de M inducida pola norma 2 de $\mathbb{R}^n$ é

$$\|M\| = \max_{v\neq 0} \frac{\|Mv\|}{\|v\|}$$

e o condicionamento de M respecto da norma 2

$$\mathrm{cond}(M) = \|M\| \left\|M^{-1}\right\|.$$

Cúmprese ademais que, para todo $v \in \mathbb{R}^n$

$$< Mv, v >= (Mv)^t v = v^t M^t v =< v, M^t v > . \tag{1.3}$$

Se ademais M é simétrica definida positiva, entón todos os autovalores de M son reais. Denótase por $\lambda_{\text{máx}}(M)$ e $\lambda_{\text{mín}}(M)$ ao máximo e mínimo valor propio de M, respectivamente. O condicionamento de M neste caso é

$$\text{cond}(M) = \frac{\lambda_{\text{máx}}(M)}{\lambda_{\text{mín}}(M)}.$$

Lémbrese que, para todo $v \in \mathbb{R}^n$

$$\lambda_{\text{mín}}(M)) \left\|v\right\|^2 \leq < v, Mv > \leq \lambda_{\text{máx}}(M) \left\|v\right\|^2 . \tag{1.4}$$

A matriz de orde $n \times s$ que forman os vectores columna de $\mathbb{R}^n$, $v_1, \ldots, v_s$, denotarase por

$$(v_1 \,|v_2 \ldots| \, v_s)$$

e

$$\pounds\{v_1, \ldots, v_s\}$$

indicará o subespazo vectorial que xeran. Do mesmo xeito, se $A_1, \ldots, A_s$ son matrices reais de orde $n \times p$, a matriz de orde $n \times ps$ que forman escribirase

$$(A_1 \,|A_2 \ldots| \, A_s)$$

e

$$\pounds\{A_1, \ldots, A_s\}$$

indicará o subespazo vectorial xerado polos vectores columna de todas as matrices. Asimesmo, chámase espazo rango dunha matriz ao subespazo vectorial xerado polos seus vectores columna.

Sexa N unha matriz cadrada de orde $n \times n$ simétrica e definida positiva, e u, v dous vectores de $\mathbb{R}^n$. Dise que u e v son N-ortogonais se

$$< u, Nv >= 0$$

e son N-ortonormais se ademais

$$< u, Nu >= 1 \quad \text{e} \quad < v, Nv >= 1.$$

Se $N = I$, a matriz identidade de orde $n \times n$, dise que os vectores son ortogonais ou ortonormais, respectivamente. Analogamente, se P e Q son dúas matrices de orde $n \times s$ calesquera, dise que P e Q son N-ortogonais se

$$P^t N Q = 0,$$

o que equivale a que os vectores columna de P sexan N-ortogonais cos vectores columna de Q. Se $N = I$, a matriz identidade de orde $n \times n$, dise que P e Q son ortogonais.

Sexa $b \in \mathbb{R}^n$ un vector columna. Un sistema de ecuacións lineares con matriz de coeficientes A é

$$Ax = b, \tag{1.5}$$

onde $x = (x_1, \dots, x_n)^t$ é o vector columna de incógnitas. A solución exacta do sistema vaise denotar por $c \in \mathbb{R}^n$. Se a matriz A é dispersa dise que o sistema é disperso. Para a resolución numérica destes sistemas existen métodos numéricos directos e métodos iterativos, máis convenientes na resolución de grandes sistemas dispersos. Entre os métodos iterativos están os métodos clásicos como Jacobi, Gauss-Seidel, Relaxación ([72, 73]), etc, e os baseados en subespazos de Krylov. Estes últimos son os que van ser obxecto de estudo neste traballo.

Recórdase a continuación o concepto de subespazo de Krylov.

Definición 1.1 *Para cada $v \in \mathbb{R}^n$, $v \neq 0$ e $s \in \mathbb{N}$, o subespazo vectorial*

$$\pounds\{v, Av, A^2v, \dots, A^{s-1}v\}$$

chámase subespazo de Krylov de orde s, e denotarase por $\mathcal{K}_s(A, v)$.

En xeral a dimensión de $\mathcal{K}_s(A, v)$ é menor ou igual que s se $s < n$, mais hai casos prácticos onde é igual a s.

Lémbrese que un polinomio mónico é o que ten o coeficiente do termo de maior grao igual a 1.

Definición 1.2 *Sexa $v \in \mathbb{R}^n$. Defínese o grao de v respecto da matriz A, como o grao mínimo dun polinomio mónico $p(\lambda)$ tal que $p(A)v = 0$.*

Observación 1.1 *Se A é unha matriz non singular, entón o polinomio $p(\lambda)$ debe ser de termo independente non nulo, pois do contrario $\lambda^{-1}p(\lambda)$ sería tamén un polinomio mónico tal que $A^{-1}p(A)v = 0$, de grao unha unidade menor que $p(\lambda)$.*

Denotarase ao grao de v respecto da matriz A como $\text{grao}_A(v)$. Como consecuencia do teorema de Cayley-Hamilton que asegura que toda matriz é raíz da súa ecuación característica (por exemplo [72]), o grao de v respecto de A nunca será maior que n. Nos seguintes lemas verase a relación entre este concepto e a dimensión do subespazo de Krylov $\mathcal{K}_s(A, v)$. As demostracións dos mesmos seguen as ideas expostas en [72].

Lema 1.1 *Sexa $v \in \mathbb{R}^n$, $v \neq 0$. Se $grao_A(v) = m$ e $s \geq m$, entón $\mathcal{K}_s(A, v)$ é A-invariante, é dicir, $A\mathcal{K}_s(A, v)$=$\mathcal{K}_s(A, v)$. Ademais,*

$$\mathcal{K}_s(A, v) = \mathcal{K}_m(A, v). \tag{1.6}$$

Demostración:

Para ver que $\mathcal{K}_s(A, v)$ é A-invariante bastará ver que $A^s v \in \mathcal{K}_s(A, v)$ e que $v \in A\mathcal{K}_s(A, v)$, pois os outros xeradores coinciden. A demostración faise por indución sobre s:

Se $s = m = \text{grao}_A(v)$ entón existen $\lambda_0, \dots, \lambda_{s-1} \in \mathbb{R}$ tales que

$$A^s v = \lambda_{s-1}A^{s-1}v + \cdots + \lambda_0 v \tag{1.7}$$

e como consecuencia $A^s v \in \mathcal{K}_s(A, v)$. Ademais, como A é non singular, da observación 1.1 tense que $\lambda_0 \neq 0$ do que se deduce que $v \in A\mathcal{K}_s(A, v)$. Ademais, $\mathcal{K}_s(A, v) = \mathcal{K}_m(A, v)$ trivialmente.

Supóñase agora o lema certo para $s > m = \text{grao}_A(v)$ e próbase que entón é certo para $s + 1$. Dado que $A^{s+1}v = A(A^s v)$ e $A^s v \in A\mathcal{K}_s(A, v)$, por ser $\mathcal{K}_s(A, v)$ un subespazo A-invariante pola hipótese de indución, tense que

$$A^{s+1}v \in A\mathcal{K}_s(A, v), \tag{1.8}$$

o cal demostra a primeira parte, pois $A\mathcal{K}_s(A, v) \subset \mathcal{K}_{s+1}(A, v)$ de xeito inmediato.

Por outra banda, tense que $v \in A\mathcal{K}_{s+1}(A, v)$ por ser $\text{grao}_A(v) = m < s$ e verificarse como consecuencia da observación 1.1

$$A^m v = \lambda_{m-1}A^{m-1}v + \cdots + \lambda_0 v, \text{ con } \lambda_0 \neq 0. \tag{1.9}$$

Para demostrar que $\mathcal{K}_s(A,v) = \mathcal{K}_m(A,v)$ basta ver que

$$A^m v, A^{m+1}v, \dots, A^{s-1}v \in \mathcal{K}_m(A,v).$$

Da ecuación 1.9 tense que $A^m v \in \mathcal{K}_m(A,v)$. Multiplicando sucesivamente por A na ecuación 1.9 vaise deducindo a pertenza do resto dos vectores a $\mathcal{K}_m(A,v)$. ■

Estase agora en condicións de determinar a dimensión dun subespazo de Krylov de xeito xeral.

Lema 1.2 *Verifícase:*

$$\dim(\mathcal{K}_s(A,v)) = \text{mín}\{s, grao_A(v)\}. \tag{1.10}$$

Demostración:

Sexa $s < \text{grao}_A(v)$. Entón os s xeradores de $\mathcal{K}_s(A,v)$, $\{v, Av, \dots, A^{s-1}v\}$, son linearmente independentes se e só se, para calquera conxunto de escalares $\alpha_i \in \mathbb{R}$, $i = 0, \dots, s-1$, a igualdade $\alpha_0 v + \alpha_1 Av + \cdots + \alpha_{s-1}A^{s-1}v = 0$ implica $\alpha_i = 0$, $i = 0, \dots, s-1$, o cal é certo pola definición 1.2, posto que en caso contrario sería $\text{grao}_A(v) < s$. En consecuencia $\dim(\mathcal{K}_s(A,v)) = s$. Para $s \geq \text{grao}_A(v)$ o resultado é consecuencia directa da segunda parte do lema 1.1. ■

Dados dous subespazos vectoriais $\mathcal{K}_m$ e $\mathcal{L}_m$ de dimensión m en $\mathbb{R}^n$, unha proxección P de $\mathbb{R}^n$ no subespazo $\mathcal{K}_m$ e ortogonal a $\mathcal{L}_m$ é unha aplicación linear tal que $P^2 = P$, a súa imaxe é $\mathcal{K}_m$ e o seu núcleo o subespazo ortogonal a $\mathcal{L}_m$. Cando $\mathcal{L}_m = \mathcal{K}_m$ dise que a proxección é ortogonal, en caso contrario é unha proxección oblícua.

Os métodos iterativos baseados en subespazos de Krylov resolven numericamente un sistema linear mediante un proceso de proxección onde $\mathcal{K}_m = \mathcal{K}_m(A, r_0)$ e $\mathcal{L}_m$ é outro subespazo de Krylov que pode ou non coincidir co propio $\mathcal{K}_m$. En cada iteración obtense unha solución aproximada x_m no subespazo afín $x_0 + \mathcal{K}_m(A, r_0)$ impoñendo a condición de que o residuo $r_m = b - Ax_m$ sexa ortogonal a un determinado subespazo $\mathcal{L}_m$ de dimensión m. Esta condición é equivalente a que $P(r_m) = 0$ e coñécese como a condición de Petrov-Galerkin.

No desenvolvemento destes métodos é fundamental que os subespazos de Krylov sexan de dimensión máxima, é dicir, que $\dim(\mathcal{K}_s(A, r_i)) = s$ para todo $i = 0, 1, \dots,$

pois do contrario o método pode dexenerar sen ter converxido. A ruptura do método por esta razón é, en teoría, pouco probable. Aínda no caso no que se dera, teríase conseguido dar coa solución exacta do sistema. É por isto que a esta circunstancia se lle atribúe o termo en lingua inglesa *lucky breakdown*. O lema que segue a continuación é o que permite facer esta afirmación:

Lema 1.3 *Aplíquese un método iterativo baseado en subespazos de Krylov para a resolución numérica do sistema de ecuacións lineares* $Ax = b$ *e sexa* $r_i = b - Ax_i$, $i = 0, 1, \dots, m$, *o residuo do* i*-ésimo iterante. Entón, se* $\dim(\mathcal{K}_s(A, r_0)) = m < s$, *a solución do sistema é exacta na iteración* m *do método.*

Demostración:

Sexa x_m a solución aproximada obtida polo método iterativo na iteración m. Sexa P a proxección de $\mathbb{R}^n$ en $\mathcal{K}_m(A, r_0)$ asociada ao método. Tense entón que

$$x_m \in x_0 + \mathcal{K}_m(A, r_0) \tag{1.11}$$

e

$$P(r_m) = P(b - Ax_m) = 0. \tag{1.12}$$

Utilizando a linearidade de P, pódese expresar

$$P(r_m) = P(b - Ax_0) + P(Ax_0 - Ax_m). \tag{1.13}$$

Como

$$r_0 = b - Ax_0 \in \mathcal{K}_m(A, r_0) \tag{1.14}$$

e

$$Ax_0 - Ax_m \in \mathcal{K}_m(A, r_0) \tag{1.15}$$

por ser $\mathcal{K}_m(A, r_0)$ A-invariante e $x_0 - x_m \in \mathcal{K}_m(A, r_0)$, tense que

$$P(r_m) = b - Ax_0 + Ax_0 - Ax_m = r_m \tag{1.16}$$

e polo tanto

$$r_m = 0 \tag{1.17}$$

e entón a solución x_m é exacta. ■

1.2. Métodos iterativos en s-pasos de Chronopoulos

As variantes en s-pasos (s-step iterative methods) de métodos iterativos baseados en subespazos de Krylov foron introducidos por Chronopoulos en [22], [23] e [24] coa intención de substituír operacións tipo BLAS 1 ou BLAS 2 por outras tipo BLAS 2 ou BLAS 3 de xeito que a programación en arquitecturas paralelas destas variantes fora máis eficiente que a dos métodos orixinais. As primeiras variantes en s-pasos publicadas foron a do Gradente Conxugado e a do Residuo Conxugado en [24] por Chronopulos e Gear [22]. Ao igual que os métodos orixinais, estas variantes converxen para matrices simétricas e definidas positivas. Posteriormente, en [23], Chronopoulos xeneraliza estas variantes para os algoritmos Xeneralizado do Residuo Conxugado (XCR), o do Mínimo Residuo (MR) e o Orthomin(m) e polo tanto métodos converxentes para algúns tipos de matrices non necesariamente simétricas e non necesariamente definidas positivas.

Os métodos iterativos en s-pasos consisten en construír en cada iteración unha base dun subespazo de Krylov do tipo

$$\mathcal{K}_s(A, v) = \pounds\{v, Av, A^2v, \dots, A^{s-1}v\} \tag{1.18}$$

de dimensión s, o cal é equivalente a que o grao de v respecto da matriz A sexa maior ou igual a s. Despois, mediante un procedemento axeitado de proxección sobre dito subespazo, calcúlase o seguente iterante de xeito que minimice, segundo o método considerado (GC, XCR, Orthomin(m), etc...), ou ben o erro ou ben a norma do residuo.

Se $c \in \mathbb{R}^n$ denota a solución exacta do sistema, o Gradente Conxugado minimiza o funcional

$$E(x) =< c - x, A(c - x) > .$$

O algoritmo clásico é o seguinte:

Algoritmo 1.1 (Gradente Conxugado (GC)).
Sexa $x_0 \in \mathbb{R}^n$
$p_0 = r_0 = b - Ax_0$
Para $i = 0, 1, 2, \dots$ *ata converxencia*

$$\alpha_i = \frac{< r_i, r_i >}{< Ap_i, p_i >} \tag{1.19}$$

$$x_{i+1} = x_i + \alpha_i p_i \tag{1.20}$$

$$r_{i+1} = r_i - \alpha_i A p_i \tag{1.21}$$

$$\beta_i = \frac{< r_{i+1}, r_{i+1} >}{< r_i, r_i >} \tag{1.22}$$

$$p_{i+1} = r_{i+1} + \beta_i p_i \tag{1.23}$$

Fin

Lémbranse os seguintes resultados [4, 51, 72]:

1. Cada residuo r_k é ortogonal aos anteriores $r_0, \dots, r_{k-1}$.

2. As direccións de descenso p_i son A-ortogonais entre sí, é dicir:

$$< Ap_i, p_j >= 0 \quad se \;\; i \neq j.$$

3. Cúmprense as siguientes igualdades entre os espazos xerados polos residuos, as direccións de descenso, e o subespazo de Krylov $\mathcal{K}_k(A, r_0)$

$$\mathcal{K}_k(A, r_0) = \pounds\{r_0, r_1, \dots, r_{k-1}\} = \pounds\{p_0, \dots, p_{k-1}\}. \tag{1.24}$$

Polo tanto, da definición de x_{i+1} próbase por indución que

$$x_{k+1} = x_0 + \sum_{i=0}^{k} \alpha_i p_i,$$

e de (1.24) séguese que

$$x_{k+1} = x_0 + \sum_{i=0}^{k} a_i A^i r_0,$$

onde $\alpha_i, a_i \in \mathbb{R}$, $i = 0, \dots, k$, son escalares.

Entón, se $s < n$ e o GC non converxe en menos de s iteracións, a dimensión do subespazo $\pounds\{r_0, r_1, \dots, r_{s-1}\}$ é máxima por ser os residuos $r_0, r_1, \dots, r_{s-1}$ distintos de cero e ortogonais entre si. Deste xeito, se, para cada $i = 0, 1, 2, \dots$ ata a converxencia, o grao de r_i respecto de A é maior ou igual que s, a variante en s-pasos do GC obtense xerando en cada iteración os s vectores linearmente independentes $\{r_i, Ar_i, A^2 r_i, \dots, A^{s-1} r_i\}$ e facendo que sexan A-ortogonais ás s direccións calculadas na iteración anterior.

Finalmente, se c denota a solución exacta do sistema, calcúlase o novo iterante e o novo residuo de xeito que o funcional

$$E(x) =< c - x, A(c - x) >,$$

sexa mínimo sobre a variedade afín

$$x_i + \mathcal{K}_s(A, r_i).$$

Deste xeito a variante en s-pasos do Gradente Conxugado que se propón en [24] é a seguinte:

Algoritmo 1.2 (Gradente Conxugado en s-pasos (s-GC)).
Sexa $x_0 \in \mathbb{R}^n$
$p_0^1 = r_0 = b - Ax_0, \ldots, p_0^s = A^{s-1}r_0$
Para $i = 0, 1, 2, \ldots$ *ata converxencia*
calcúlanse os coeficientes a_i^j *resolvendo*

$$\left.\begin{array}{rcl} a_i^1 < p_i^1, Ap_i^1 > + \cdots + a_i^s < p_i^s, Ap_i^s > & = & < r_i, Ap_i^1 > \\ \ldots\ldots\ldots\ldots\ldots & & \\ \ldots\ldots\ldots\ldots\ldots & & \\ a_i^1 < p_i^s, Ap_i^1 > + \cdots + a_i^s < p_i^s, Ap_i^s > & = & < r_i, Ap_i^s > \end{array}\right\}$$

$$x_{i+1} = x_i + a_i^1 p_i^1 + a_i^2 p_i^2 + \cdots + a_i^s p_i^s \tag{1.25}$$

$$r_{i+1} = b - Ax_{i+1} \tag{1.26}$$

calcúlanse os coeficientes $b_i^{(j,l)}$ *de xeito que* $< p_{i+1}^j, Ap_i^l >= 0$ *para* $1 \le j, l \le s$

$$\left.\begin{array}{rcl} p_{i+1}^1 & = & r_i + b_i^{(1,1)} p_i^1 + \cdots + b_i^{(1,s)} p_i^s \\ p_{i+1}^2 & = & Ar_i + b_i^{(2,1)} p_i^1 + \cdots + b_i^{(2,s)} p_i^s \\ \ldots\ldots & \ldots & \ldots\ldots \\ \ldots\ldots & \ldots & \ldots\ldots \\ p_{i+1}^s & = & A^{s-1}r_i + b_i^{(s,1)} p_i^1 + \cdots + b_i^{(s,s)} p_i^s \end{array}\right\}$$

Fin

Para cada $i = 0, 1, 2, \ldots$, denótase por W_i á matriz simétrica de orde $s \times s$ de xeito que, para $1 \le j, l \le s$, o termo da fila j-ésima e columna l-ésima é $< p_i^j, Ap_i^l >$.

A matriz W_i é non singular xa que $p_i^1, ..., p_1^s$ son linearmente independentes. Para obter os coeficientes $b_i^{(j,l)}$ é preciso resolver os s sistemas lineares

$$W_i b_i^j = c_i^j$$

onde

$$b_i^j = (b_i^{(1,1)}, ..., b_i^{(1,1)})$$

e

$$c_i^j = (< A^{j-1} r_i, Ap_i^1 >, ..., < A^{j-1} r_i, Ap_i^s >)$$

con $1 \leq j \leq s$. En [24] demóstrase que este método converxe ao sumo na parte enteira de n/s iteracións se A é unha matriz cadrada non singular simétrica e definida positiva.

En [23] proponse a variante en s-pasos do algoritmo do Mínimo Residuo. Supoñendo que, para cada $i = 0, 1, ...$ ata a converxencia, o grao de r_i respecto de A é maior ou igual que s, constrúese en cada iteración o subespazo de Krylov

$$\mathcal{K}_s(A, r_i) = \pounds\{r_i, Ar_i, ..., A^{s-1} r_i\}$$

e despois obtense

$$x_{i+1} = x_i + a_i^1 r_i + \cdots + a_i^s A^{s-1} r_i$$

calculando os coeficientes $\{a_i^1, ..., a_i^s\}$ de xeito que se minimice o funcional

$$E(x) = \|b - Ax\| = \|r\|$$

sobre a variedade afín

$$x_i + \mathcal{K}_s(A, r_i),$$

sendo o novo residuo $r_{i+1} = b - Ax_{i+1}$. Isto é equivalente a que o residuo r_{i+1} sexa ortogonal a $A\mathcal{K}_s(A, r_i)$. Entón o algoritmo resulta o seguinte:

Algoritmo 1.3 (Mínimo Residuo en s-pasos (s-MR)).
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0$
Para $i = 0, 1, 2, ...$ *ata converxencia*
calcúlanse os coeficientes a_i^j *resolvendo*

$$\left.\begin{array}{lcl} a_i^1 < Ar_i, Ar_i > + \cdots + a_i^s < Ar_i, A^s r_i > & = & < r_i, Ar_i > \\ \cdots\cdots\cdots\cdots\cdots & & \\ \cdots\cdots\cdots\cdots\cdots & & \\ a_i^1 < A^s r_i, Ar_i > + \cdots + a_i^s < A^s r_i, A^s r_i > & = & < r_i, A^s r_i > \end{array}\right\}$$

$$x_{i+1} = x_i + a_i^1 r_i + a_i^1 A r_i + \cdots + a_i^s A^{s-1} r_i \tag{1.27}$$

$$r_{i+1} = b - A x_{i+1} \tag{1.28}$$

Fin

A partir da variante en s-pasos do Mínimo Residuo pódese construír a variante en s-pasos do algoritmo Xeneralizado do Residuo Conxugado (s-XRC) e a do s-Orthomin(m), impoñendo en cada iteración que as s direccións do subespazo de Krylov $\mathcal{K}_s(A, r_i)$ sexan A^tA-ortogonais simultaneamente aos m subespazos de Krylov anteriores, no caso do s-Orthomin(m), ou a todos os subespazos de Krylov anteriores, no caso do s-XRC. O seguinte algoritmo engloba entón os dous métodos:

Algoritmo 1.4 (s-XRC ou s-Orthomin(m)).
Sexa $x_0 \in \mathbb{R}^n$
$p_0^1 = r_0 = b - Ax_0, \ldots, p_0^s = A^{s-1} r_0$
Para $i = 0, 1, 2, \ldots$ *ata converxencia*
calcúlanse os coeficientes a_i^j *resolvendo*

$$\left.\begin{array}{rcl} a_i^1 < Ap_i^1, Ap_i^1 > + \cdots + a_i^s < Ap_i^s, Ap_i^s > & = & < r_i, Ap_i^1 > \\ \ldots\ldots\ldots\ldots\ldots & & \\ \ldots\ldots\ldots\ldots\ldots & & \\ a_i^1 < Ap_i^s, Ap_i^1 > + \cdots + a_i^s < Ap_i^s, Ap_i^s > & = & < r_i, Ap_i^s > \end{array}\right\}$$

$$x_{i+1} = x_i + a_i^1 p_i^1 + a_i^2 p_i^2 + \cdots + a_i^s p_i^s \tag{1.29}$$

$$r_{i+1} = b - A x_{i+1} \tag{1.30}$$

Para $k = h, \ldots, i$ ($h = 0$ *se s-XRC ou* $h = i - m + 1$ *se Orthomin(m)),*
calcúlanse os coeficientes $b_k^{(j,l)}$ *de xeito que* $< Ap_{i+1}^j, Ap_k^l >= 0$ *para* $1 \le j, l \le s$

$$\left.\begin{array}{rcl} p_{i+1}^1 & = & r_i + \sum_{k=h}^{i} \left(b_k^{(1,1)} p_k^1 + \cdots + b_k^{(1,s)} p_k^s \right) \\ p_{i+1}^2 & = & A r_i + \sum_{k=h}^{i} \left(b_k^{(2,1)} p_k^1 + \cdots + b_k^{(2,s)} p_k^s \right) \\ \ldots\ldots & \ldots & \ldots\ldots \\ \ldots\ldots & \ldots & \ldots\ldots \\ p_{i+1}^s & = & A^{s-1} r_i + \sum_{k=h}^{i} \left(b_k^{(s,1)} p_k^1 + \cdots + b_k^{(s,s)} p_k^s \right) \end{array}\right\}$$

Fin

A variante en s-pasos do Orthomin(0) é a variante en s-pasos do Mínimo Residuo e a variante en s-pasos do Orthomin(1) coincide coa variante en s-pasos do Residuo Conxugado cando a matriz A é simétrica definida positiva.

Non obstante a converxencia destes métodos non está garantida para todo tipo de matriz cadrada non singular. En [23], por exemplo, demóstrase que estes métodos converxen para toda matriz cadrada non singular e simétrica, nonsimétrica definida e algunhas nonsimétricas indefinidas.

Ademais das variantes dos métodos mencionados, en [22] Chronopoulos presenta unha variante en s-pasos do método da Ecuación Normal, converxente para toda matriz cadrada non singular e que consiste en aplicar o Gradente Conxugado ao sistema

$$A^tAx = A^tb. \tag{1.31}$$

Mais este método pode estar comprometido cualitativamente polo gran condicionamento do sistema, pois a tasa de converxencia é da orde de $\mathrm{cond}(A^tA)$.

En [2] preséntase unha variante en s-pasos do algoritmo da Dobre Serie Ortogonal. A versión orixinal deste algoritmo pode verse en [5]. Esta variante vólvese a presentar mellorada no capítulo 5 desta tese e ten como vantaxe a converxencia do método para toda matriz cadrada non singular.

A pesar das propiedades de converxencia para as variantes en s-pasos demostradas nestes artigos e a maior eficiencia no cálculo paralelo corroborada polos resultados numéricos publicados, o tamaño do parámetro s está limitado ($s \leq 5$) por problemas de estabilidade numérica. Chronopoulos e Swanson introducen en [29] unha modificación nas variantes do XCR e o Orthomin(m) coa intención de resolver este problema. A modificación consiste en A^tA-ortonormalizar as s direccións calculadas en cada iteración do algoritmo usando o método Modificado de Gram-Schmidt. Deste xeito obtéñense resultados sen problemas de estabilidade numérica para valores de s ata $s = 14$.

Recentemente Chronopulos e Kucherov introduciron en [27] e [28] variantes ortonormalizadas en s-pasos dos métodos en bloques do XCR e do Orthomin(m). Estes métodos en bloques son algoritmos para a resolución de sistemas lineares con múlti-

ples termos independentes, os cales aparecen, por exemplo, na resolución numérica de problemas de control óptimo de ecuacións en derivadas parciais.

1.3. Algoritmo Xeral de Ortogonalización (AXO)

Se a matriz A non é necesariamente simétrica nin definida positiva, existe unha xeneralización do método do Gradente Conxugado, o Algoritmo Xeral de Ortogonalización (AXO), que se describe seguidamente de xeito resumido [4, 51].

Partindo do sistema linear $Ax = b$, con A matriz cadrada non singular de orde n e $b \in \mathbb{R}^n$, sexan H e K matrices cadradas de orde n coa súa parte simétrica definida positiva. Introdúcense as seguintes matrices:

$$N = A^t H^S A \tag{1.32}$$

$$M = L^t N L \tag{1.33}$$

onde $K^S = LL^t$, é dicir, LL^t é a factorización de Cholesky da parte simétrica de K.

Para todo $r \in \mathbb{R}^n$ defínese $E(r) =< r, Hr >$. Xa que

$$E(r) =< H^t r, r >=< r, Hr >,$$

verifícase tamén

$$E(r) =< r, \frac{1}{2}(H + H^t)r >=< r, H^S r > . \tag{1.34}$$

Posto que H^S é definida positiva, verifícase que $E(r)$ é unha función estrictamente convexa. O seguinte algoritmo é un algoritmo de ortogonalización que tamén minimiza a función $E(r)$ sobre $\mathbb{R}^n$:

Algoritmo 1.5 (Xeral de Ortogonalización (AXO)).
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0 = A(x - x_0)$
$g_0 = A^t H^S r_0 = A^t H^S A(x - x_0) = N(x - x_0)$
$p_0 = K g_0$
Para $i = 0, 1, 2, ...$ *ata converxencia*

$$\alpha_i = \frac{< g_i, p_i >}{< p_i, N p_i >} \tag{1.35}$$

$$x_{i+1} = x_i + \alpha_i p_i \tag{1.36}$$

$$g_{i+1} = g_i - \alpha_i N p_i = A^t H^S r_{i+1} \tag{1.37}$$

$$\beta_{i+1}^l = -\frac{< Kg_{i+1}, Np_l >}{< p_l, Np_l >}, \quad 0 \leq l \leq i \tag{1.38}$$

$$p_{i+1} = Kg_{i+1} + \sum_{l=0}^{i} \beta_{i+1}^l p_l \tag{1.39}$$

Fin

Denomínanse *residuos xeneralizados* aos vectores g_i, e *direccións de descenso xeneralizados* aos vectores p_i. As seguintes propiedades están demostradas en [51] ou [4]:

1. $r_i = b - Ax_i$, $i = 0, 1, 2, ...$
2. $< p_i, Np_j >= 0$ para todo $i \neq j$.
3. $< g_i, p_j >= 0$ para todo $0 \leq j \leq i$.
4. $< g_i, Kg_j >= 0$ para todo $0 \leq j \leq i$.
5. Se fosen $g_0 \, ... \, g_{n-1} \neq 0$ entó $g_n = 0$.

Unha propiedade importante do Algoritmo Xeral de Ortogonalización, e que pode verse en [51], é que o subespazo que xeran as k primeiras direccións de descenso xeneralizados é o mesmo que o subespazo de Krylov $\mathcal{K}_k(KN, Kg_0)$. No seguinte lema enúnciase esta propiedade.

Lema 1.4 *No Algoritmo Xeral de Ortogonalización cúmprese que:*

$$\begin{gathered} \pounds\{p_0, ..., p_k\} = \\ = \pounds\{Kg_0, Kg_1, ..., Kg_k\} = \pounds\{Kg_0, (KN)Kg_0, ..., (KN)^k Kg_0\}. \end{gathered} \tag{1.40}$$

Ademais, o residuo r_{i+1} minimiza o funcional $E(r)$ sobre o subespazo afín $x_0 + \pounds\{p_0, ..., p_{i-1}\}$.

Demostración:

Desenvólvese aquí de xeito máis detallado a demostración deste lema que vén dada en [51]. A inclusión

$$\pounds\{p_0, ..., p_k\} \subset \pounds\{Kg_0, Kg_1, ..., Kg_k\} \tag{1.41}$$

demóstrase por indución sobre k, xa que $p_0 = Kg_0$ e se $k = 1$,

$$p_1 = Kg_1 + \beta_1^0 p_0 = Kg_1 + \beta_1^0 Kg_0. \tag{1.42}$$

Supónse agora certo o resultado para $k > 1$, entón

$$p_{k+1} = Kg_{k+1} + \sum_{l=0}^{k} \beta_{k+1}^l p_l \tag{1.43}$$

e pola hipótese de indución tense a inclusión:

$$\mathcal{L}\{p_0, ..., p_{k+1}\} \subset \mathcal{L}\{Kg_0, Kg_1, \dots, Kg_{k+1}\}. \tag{1.44}$$

A igualdade conclúese da independencia linear de $\{p_0, ..., p_k\}$, xa que

$$\dim \mathcal{L}\{p_0, ..., p_k\} = k + 1 \leq \dim \mathcal{L}\{Kg_0, Kg_1, \dots, Kg_k\} \leq k + 1. \tag{1.45}$$

A igualdade

$$\mathcal{L}\{Kg_0, Kg_1, \dots, Kg_k\} = \mathcal{L}\{Kg_0, (KN)Kg_0, \dots, (KN)^k Kg_0\} \tag{1.46}$$

é certa trivialmente para $k = 0$. A inclusión

$$\mathcal{L}\{Kg_0, Kg_1, \dots, Kg_k\} \subset \mathcal{L}\{Kg_0, (KN)Kg_0, \dots, (KN)^k Kg_0\} \tag{1.47}$$

próbase por indución sobre k:
Se $k = 1$,

$$g_1 = g_0 - \alpha_0 N p_0 = g_0 - \alpha_0 NKg_0, \tag{1.48}$$

o cal demostra a inclusión para $k = 1$. Suponse a inclusión certa para $k \in \mathbb{N}$. Como

$$g_{k+1} = g_k - \alpha_k N p_k \tag{1.49}$$

e $p_k \in \mathcal{L}\{Kg_0, Kg_1, \dots, Kg_k\}$, conclúese pola hipótese de indución que

$$Kg_{k+1} \in \mathcal{L}\{Kg_0, (KN)Kg_0, \dots, (KN)^{k+1} Kg_0\}. \tag{1.50}$$

Para probar a igualdade, razóase de xeito similar á igualdade anterior, tendo en conta que $\{Kg_0, Kg_1, \dots, Kg_k\}$ son tamén linearmente independentes como consecuencia da igualdade anterior. ■

En consecuencia, o AXO converxe en n iteracións ao sumo. Ademais, a seguinte estimación do erro está demostrada en [51] ou [4]:

$$E_i \leq E_0 \left(1 - \frac{\lambda_{\text{mín}}(L^t (K^{-1})^S L)}{\text{cond}(M)}\right)^i \tag{1.51}$$

e, se a matriz K é simétrica:

$$E_i \leq E_0 \left(\frac{\text{cond}(M) - 1}{\text{cond}(M) + 1} \right)^{2i}. \tag{1.52}$$

Debe observarse que se a matriz K é simétrica entón, pola propiedade 4 e a definición de g_{i+1}, tense que $\beta_{i+1}^l = 0$ para todo $0 \leq l < i$ e, deste xeito, para o cálculo de p_{i+1} non faría falta ter en conta todos os p_i anteriores. Porén, no caso de que a matriz K non sexa simétrica, fanse necesarios todos os p_i anteriores para o cálculo de p_{i+1}, o cal supón un problema de almacenamento cando o número de iteracións é grande. Este problema resólvese en parte utilizando para o cálculo dos p_{i+1} unicamente as m direcciones anteriores, sendo m un parámetro previamente escollido. Obtense así unha nova variante do método chamado Orthomin(m) que se menciona máis adiante.

No algoritmo 1.5 poden facerse distintas eleccións das matrices H e K, obtendo como casos particulares algúns dos métodos iterativos coñecidos baseados en subespazos de Krylov:

- Se $H = A^{-1}$ e $K = I$, entón $N = A$ e tense o algoritmo 1.1 do Gradente Conxugado, converxente para A simétrica definida positiva.
- Se $H = I$ e $K = A^{-1}$, entón $N = A^t A$ e o algoritmo é o do Residuo Conxugado, converxente para A simétrica non singular:

  Algoritmo 1.6 (Residuo Conxugado (RC)).
  Sexa $x_0 \in \mathbb{R}^n$
  $p_0 = r_0 = b - Ax_0$
  Para $i = 0, 1, 2, ...$ *ata converxencia*

$$\alpha_i = \frac{< r_i, Ap_i >}{< Ap_i, Ap_i >}$$
$$x_{i+1} = x_i + \alpha_i p_i$$
$$r_{i+1} = r_i - \alpha_i Ap_i$$
$$\beta_i = \frac{< Ar_{i+1}, Ap_i >}{< Ap_i, Ap_i >}$$
$$p_{i+1} = r_{i+1} + \beta_i p_i$$

  Fin

- Se $H = I$ e $K = I$, entón $N = A^tA$ e o algoritmo que resulta é o da Ecuación Normal, converxente para A cadrada non singular:

Algoritmo 1.7 (Ecuación Normal).
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0$
$p_0 = A^t r_0$
Para $i = 0, 1, 2, ...$ *ata converxencia*

$$\alpha_i = \frac{< A^t r_i, A^t r_i >}{< Ap_i, Ap_i >}$$
$$x_{i+1} = x_i + \alpha_i p_i$$
$$r_{i+1} = r_i - \alpha_i A p_i$$
$$\beta_i = \frac{< A^t r_{i+1}, A^t r_{i+1} >}{< A^t r_i, A^t r_i >}$$
$$p_{i+1} = A^t r_{i+1} + \beta_i p_i$$

Fin

- Se $H = (AA^t)^{-1}$ e $K = A^tA$, entón $N = I$ e tense o algoritmo do Erro Minimal, converxente para A cadrada non singular:

Algoritmo 1.8 (Erro Minimal).
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0$
$p_0 = A^t r_0$
Para $i = 0, 1, 2, ...$ *ata converxencia*

$$\alpha_i = \frac{< r_i, r_i >}{< p_i, p_i >}$$
$$x_{i+1} = x_i + \alpha_i p_i$$
$$r_{i+1} = r_i - \alpha_i A p_i$$
$$\beta_i = \frac{< r_{i+1}, r_{i+1} >}{< r_i, r_i >}$$
$$p_{i+1} = A^t r_{i+1} + \beta_i p_i$$

Fin

- No caso en que A non sexa simétrica e se escolle $H = I$ e $K = A^{-1}$, como no Residuo Conxugado, entón $N = A^tA$ mais K non é simétrica. O algoritmo que

se obtén é o do Residuo Conxugado Xeneralizado, converxente se A^S é definida positiva:

Algoritmo 1.9 (Residuo Conxugado Xeneralizado (RCX)).
Sexa $x_0 \in \mathbb{R}^n$
$p_0 = r_0 = b - Ax_0$
Para $i = 0, 1, 2, ...$ *ata converxencia*

$$\alpha_i = \frac{< r_i, Ap_i >}{< Ap_i, Ap_i >}$$
$$x_{i+1} = x_i + \alpha_i p_i$$
$$r_{i+1} = r_i - \alpha_i Ap_i$$
$$\beta_i^l = \frac{< Ar_{i+1}, Ap_l >}{< Ap_l, Ap_l >}, \quad 0 \leq l \leq i$$
$$p_{i+1} = r_{i+1} + \sum_{l=0}^{i} \beta_i^l p_l$$

Fin

- A variante Orthomin(1) do Residuo Conxugado Xeneralizado coñécese como o algoritmo do Residuo Minimal, converxente se A^S é definida positiva:

Algoritmo 1.10 (Residuo Minimal).
Sexa $x_0 \in \mathbb{R}^n$
$p_0 = r_0 = b - Ax_0$
Para $i = 0, 1, 2, ...$ *ata converxencia*

$$\alpha_i = \frac{< r_i, Ap_i >}{< Ap_i, Ap_i >}$$
$$x_{i+1} = x_i + \alpha_i p_i$$
$$r_{i+1} = r_i - \alpha_i Ap_i$$
$$\beta_i = \frac{< Ar_{i+1}, Ap_i >}{< Ap_i, Ap_i >}$$
$$p_{i+1} = r_{i+1} + \beta_i^l p_i$$

Fin

Capítulo 2

Variante en s-pasos do AXO e Orthomin(m)

O AXO é un método de Krylov e, polo tanto, as direccións xeradas en cada iteración forman unha base dun subespazo de Krylov. Este feito permite neste capítulo construír unha variante en s-pasos do Algoritmo Xeral de Ortogonalización (a partir de agora s-AXO). Próbanse ademais algunhas propiedades desta variante e obtense un teorema de converxencia. O capítulo complétase cunha variante en s-pasos do método Orthomin e a inclusión nestas variantes da ortonormalización dos subespazos de Krylov co fin de obter unha mellora na estabilidade.

2.1. Variante en s-pasos do AXO

A seguinte definición simplificará a notación que se vai usar ao longo deste traballo:

Definición 2.1 *Sexa $n, s \in \mathbb{N}$ $(s < n)$, $M \in \mathcal{M}_{n\times n}(\mathbb{R})$. Defínese a aplicación*

$$\Delta_M : \mathbb{R}^n \longrightarrow \mathcal{M}_{n\times s}(\mathbb{R})$$

por:

$$\Delta_M(v) = (v\,|Mv|\,M^2v\,|...|\,M^{s-1}v) \text{ para todo } v \in \mathbb{R}^n. \tag{2.1}$$

Respecto á aplicación Δ_M definida, hai que ter en conta as seguintes observacións de inmediata comprobación:

1. $\Delta_M(\alpha u + \beta v) = \alpha\Delta_M(u) + \beta\Delta_M(v)$ para todo $\alpha, \beta \in \mathbb{R}$ e $u, v \in \mathbb{R}^n$, é dicir, Δ_M é unha aplicación linear.

2. $\Delta_M(M^k v) = M^k \Delta_M(v)$ para todo $v \in \mathbb{R}^n$ and $k \in \mathbb{N}$.

En concordancia co que se fai no AXO, sexa o sistema linear $Ax = b$, con A matriz cadrada non singular de orde n e $b \in \mathbb{R}^n$, e sexan H e K matrices cadradas de orde n coa súa parte simétrica definida positiva. Introdúcense tamén as seguintes matrices:

$$N = A^t H^S A \tag{2.2}$$

$$M = L^t N L \tag{2.3}$$

onde $K^S = LL^t$, é dicir, LL^t é a factorización de Cholesky da parte simétrica de K.

Do mesmo xeito, para todo $r \in \mathbb{R}^n$ defínese $E(r) =< r, Hr >$, e verifícase tamén

$$E(r) =< r, \frac{1}{2}(H + H^t)r >=< r, H^S r > . \tag{2.4}$$

Posto que H^S é definida positiva, tense que $E(r)$ é unha función estrictamente convexa.

Así, se para cada $i = 0, 1, 2, ...$ ata a converxencia, $\text{grao}_{KN}(Kg_i) \geq s$, entón $\mathcal{K}_s(KN, Kg_i)$ é o espazo rango da matriz $\Delta_{KN}(Kg_i)$. Polo tanto, se $\text{grao}_{KN}(Kg_i) \geq s$ para todo $i = 0, 1, 2, ...$ ata a converxencia, a variante en s-pasos do Algoritmo Xeral de Ortogonalización (s-AXO) é a seguinte:

Algoritmo 2.1 (s-AXO).
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0$
$g_0 = A^t H^S r_0$
$P_0 = \Delta_{KN}(Kg_0) = Q_0$
Para $i = 0, 1, 2, ...$ *ata converxencia*

$$W_i = (P_i)^t N P_i \tag{2.5}$$

$$z_i = (P_i)^t g_i \tag{2.6}$$

$$y_i = (W_i)^{-1} z_i \tag{2.7}$$

$$x_{i+1} = x_i + P_i y_i \tag{2.8}$$

$$g_{i+1} = g_i - N P_i y_i = A^t H^S r_{i+1} \tag{2.9}$$

$$Q_{i+1} = \Delta_{KN}(Kg_{i+1}) \tag{2.10}$$

Para $j = 0, \ldots, i$:

$$B_{i+1}^{j} = -W_j^{-1}(P_j)^t N Q_{i+1} \tag{2.11}$$

$$P_{i+1} = Q_{i+1} + \sum_{j=0}^{i} P_j B_{i+1}^{j} \tag{2.12}$$

Fin

Observación 2.1 *Por indución sobre i pódese obter na ecuación* (2.9) *a seguinte para o residuo:*

$$r_{i+1} = r_i - AP_i y_i. \tag{2.13}$$

En cada iteración do algoritmo s-AXO calcúlase na ecuación (2.12) a matriz P_{i+1} de orde $n \times s$. Nela faise uso dunha matriz de coeficientes B_{i+1}^{j}, calculada coa fórmula (2.11), que consegue, como se verá máis adiante, que a matriz P_{i+1} sexa N-ortogonal a todas as matrices P_j calculadas nas iteracións anteriores e, polo tanto, $(P_{i+1})^t N P_j = 0$ para todo $j \leq i$. Comprobarase que os vectores columna das matrices P_i son linearmente independentes e, polo tanto, forman unha base do subespazo de Krylov $\mathcal{K}_s(KN, Kg_i)$. Este feito, xunto co de que a matriz N sexa simétrica definida positiva, implica que a matriz W_i, definida na ecuación (2.5), é unha matriz non singular de orde s, co que ten sentido o cálculo da súa inversa.

Nos catro lemas que se enuncian a continuación, expóñense as propiedades do s-AXO que conclúen nun teorema de converxencia. As demostracións destes lemas e o teorema de converxencia inspíranse nas demostracións das propiedades análogas para a variante en s-pasos do Residuo Conxugado Xeneralizado que aparecen desenvoltas en [23].

No seguinte lema próbanse as propiedades máis relevantes de ortogonalidade que xeneralizan as do Algoritmo Xeral de Ortogonalización, que poden verse en [51] e [4]. Entre elas está a referida anteriormente de N-ortogonalidade entre as matrices P_i calculadas en distintas iteracións:

Lema 2.1 *Sexan* $i, s \in \mathbb{N}$ *tales que* $s(i+1) \leq n$*, e suponse que* $g_i \neq 0$ *para todo* $i = 0, 1, 2, \ldots$ *ata a converxencia. Se* $grao_{KN}(Kg_0) \geq s(i+1)$*, no s-AXO:*

a) $(P_i)^t N P_j = 0$ *para todo* $i \neq j$ (2.14)

b) $(P_j)^t g_i = 0$ *para todo* $i > j$ (2.15)

Demostración:

a) Xa que a matriz N é simétrica, bastará comprobar que $(P_j)^t N P_i = 0$ para $i > j$. Procederemos por indución sobre i:
Se $i = 1$ entón $j = 0$. Usando (2.12) e multiplicando por $(P_0)^t N$, tense:

$$(P_0)^t N P_1 = (P_0)^t N Q_1 + (P_0)^t N P_0 B_1^0, \tag{2.16}$$

e por (2.5) e (2.11):

$$(P_0)^t N P_0 B_1^0 = W_0 B_1^0 = -W_0 W_0^{-1} (P_0)^t N Q_1 = -(P_0)^t N Q_1 \tag{2.17}$$

e así

$$(P_0)^t N P_1 = 0. \tag{2.18}$$

Suponse agora certa a propiedade ata $i-1$ e comprobemos que é certa para i. Usando de novo (2.12) e a hipótese de indución:

$$(P_j)^t N P_i = (P_j)^t N Q_i + (P_j)^t N \sum_{k=0}^{i-1} P_k B_i^k = (P_j)^t N Q_i + (P_j)^t N P_j B_i^j \tag{2.19}$$

e, tendo en conta (2.5) e (2.11), tense que:

$$(P_j)^t N P_j B_i^j = W_j B_i^j = -W_j W_j^{-1} (P_j)^t N Q_i = -(P_j)^t N Q_i \tag{2.20}$$

co que finalmente tense probado que

$$(P_j)^t N P_i = 0. \tag{2.21}$$

b) Fixado $j \in \mathbb{N}$ procédese por indución sobre i. Se $i = j + 1$, multiplicando (2.9) por $(P_j)^t$ e utilizando (2.5)-(2.7) tense:

$$(P_j)^t g_{j+1} = (P_j)^t g_j - ((P_j)^t N P_j) y_j = (P_j)^t g_j - W_j y_j = 0. \tag{2.22}$$

Suponse agora certo que $(P_j)^t g_i = 0$ para $i > j$. Entón, de (2.9)

$$(P_j)^t g_{i+1} = (P_j)^t g_i - (P_j)^t N P_i y_i, \tag{2.23}$$

da hipótese de indución e *a)* dedúcese que $(P_j)^t g_{i+1} = 0$. ■

Como consecuencia do lema anterior, obtéñense outras propiedades que se enuncian a continuación e que tamén xeneralizan as súas versións análogas para o AXO [4, 51]:

Lema 2.2 *Sexan $i, s \in \mathbb{N}$ tales que $s(i+1) \leq n$, e suponse que $g_i \neq 0$ para todo $i = 0, 1, 2, ...$ ata a converxencia. Se $grao_{KN}(Kg_0) \geq s(i+1)$, no s-AXO:*

a) $(P_i)^t g_i = (Q_i)^t g_i$ *para todo* i.

b) $(Q_j)^t g_i = 0$ *para todo* $i > j$.

c) $(P_i)^t N Q_j = 0$ *para todo* $i > j$.

d) $(P_i)^t N P_i = (P_i)^t N Q_i$.

e) $(P_i)^t g_j = (P_i)^t g_0$ *para todo* $i \geq j$.

Demostración:

a) Esta igualdade obtense multiplicando por g_i en (2.12) trasposta e usando (2.15).

b) Dedúcese directamente despexando Q_j en (2.12) e usando (2.15).

c) Este apartado obtense tamén despexando Q_j en (2.12), multiplicando á esquerda por $(P_i)^t N$ e usando (2.14).

d) De (2.12) e (2.14) obtense esta igualdade.

e) Por indución, e tendo en conta (2.14) e a igualdade

$$g_j = g_{j-1} - N P_{j-1} y_{j-1},$$

demóstrase este último apartado. ■

Lembremos que $\{p_0, p_1, ..., p_{(i+1)s-1}\}$ son as direccións de descenso xeneralizadas calculadas no Algoritmo Xeral de Ortogonalización segundo (1.39). Denotamos por $p_i^1, ..., p_i^s$ aos s vectores dirección calculados en cada iteración da variante en s-pasos do Algoritmo Xeral de Ortogonalización, é dicir, $P_i = (p_i^1 \,|...|\, p_i^s)$. Tendo en conta as notacións e lemas precedentes, enunciamos o seguinte lema que relaciona os subespazos de Krylov e os xerados polos vectores dirección dos dous algoritmos:

Lema 2.3 *Sexan $i, s \in \mathbb{N}$ tales que $s(i+1) \leq n$, e suponse que $g_i \neq 0$ para todo $i = 0, 1, 2, ...$ ata a converxencia. Se $grao_{KN}(Kg_0) \geq s(i+1)$ entón:*

$$\begin{aligned} \pounds\{P_0, ..., P_i\} = \bigoplus_{j=0}^{i} \mathcal{K}_s(KN, Kg_j) = \mathcal{K}_{s(i+1)}(KN, Kg_0) = \\ = \pounds\{p_0, p_1, ..., p_{(i+1)s-1}\}, \end{aligned} \tag{2.24}$$

onde $\bigoplus$ denota a suma directa de espazos vectoriais.

Demostración:

Para demostrar a igualdade

$$\pounds\{P_0, \dots, P_i\} = \bigoplus_{j=0}^{i} \mathcal{K}_s(KN, Kg_j) \tag{2.25}$$

procedemos por indución sobre i:
Para $i = 0$ o resultado é trivial da definición dos vectores $p_0^1, \dots, p_0^s$. Suponse agora certa a igualdade para $(i-1)$. Para $k, l \in \{1, \dots, s\}$ denótase por $b_j^{k,l}$ ao elemento da fila k e columna j da matriz B_{i+1}^j definida na ecuación (2.11). Deste xeito, se $k \in \{1, \dots, s\}$, da igualdade matricial (2.12), tense a seguinte ecuación para a correspondente columna k-ésima:

$$p_i^k = (KN)^{k-1} Kg_i + \sum_{j=0}^{i-1} \left(b_j^{(k,1)} p_j^1 + \cdots + b_j^{(k,s)} p_j^s \right) \tag{2.26}$$

e entón

$$P_i \in \bigoplus_{j=0}^{i} \mathcal{K}_s(KN, Kg_j), \tag{2.27}$$

xa que $(KN)^{k-1} Kg_i \in \mathcal{K}_s(KN, Kg_i)$ e pola hipótese de indución. Como consecuencia, verifícase a inclusión

$$\pounds\{P_0, \dots, P_i\} \subset \bigoplus_{j=0}^{i} \mathcal{K}_s(KN, Kg_j). \tag{2.28}$$

A outra inclusión, e polo tanto a igualdade, obtense despexando $(KN)^{k-1} Kg_i$ na fórmula (2.26).

A inclusión

$$\bigoplus_{j=0}^{i} \mathcal{K}_s(KN, Kg_j) \subset \mathcal{K}_{s(i+1)}(KN, Kg_0) \tag{2.29}$$

demóstrase tamén por indución sobre i:
Se $i = 0$, a inclusión é trivial. Suponse agora que a inclusión (2.29) é certa para $i-1$. Se $k \in \{1, \dots, s\}$, denotando por y_i^k a coordenada k-ésima do vector y_i definido en (2.7), entón, por (2.9) tense que

$$(KN)^{k-1} Kg_i = (KN)^{k-1} (Kg_{i-1} - y_{i-1}^1 KN p_{i-1}^1 - \cdots - y_{i-1}^s KN p_{i-1}^s). \tag{2.30}$$

Tendo en conta que

$$(KN)^k Kg_{i-1} \in \mathcal{K}_{si}(KN, Kg_0) \tag{2.31}$$

pola hipótese de indución, e que

$$\pounds\{P_0,\dots,P_{i-1}\} = \bigoplus_{j=0}^{i-1} \mathcal{K}_s(KN, Kg_j) \subset \mathcal{K}_{si}(KN, Kg_0), \tag{2.32}$$

como se acaba de demostrar, e polo tanto

$$(KN)^k p_{i-1}^1, \dots, (KN)^k p_{i-1}^s \in \mathcal{K}_{s(i+1)}(KN, Kg_0), \tag{2.33}$$

pois $k + (is-1) \leq (i+1)s - 1$, cúmprese que

$$(KN)^{k-1} Kg_i \in \mathcal{K}_{s(i+1)}(KN, Kg_0). \tag{2.34}$$

Así $\bigoplus_{j=0}^{i} \mathcal{K}_s(KN, Kg_j) \subset \mathcal{K}_{s(i+1)}(KN, Kg_0)$.

Para demostrar a inclusión

$$\mathcal{K}_{s(i+1)}(KN, Kg_0) \subset \bigoplus_{j=0}^{i} \mathcal{K}_s(KN, Kg_j) \tag{2.35}$$

débese comprobar previamente que

$$g_i = g_0 + \sum_{j=0}^{is-1} \lambda_j N(KN)^j Kg_0, \text{ con } \lambda_j \in \mathbb{R}, \tag{2.36}$$

o cal demóstrase por indución sobre i na fórmula

$$g_i = g_{i-1} - y_{i-1}^1 N p_{i-1}^1 - \cdots - y_{i-1}^s N p_{i-1}^s \tag{2.37}$$

e, tendo en conta, pola igualdade (2.25) e a inclusión (2.29) demostradas anteriormente, que

$$\pounds\{P_0,\dots,P_{i-1}\} \subset \mathcal{K}_{si}(KN, Kg_0), \tag{2.38}$$

e deste xeito pódense poñer os vectores $Np_{i-1}^1, \dots, Np_{i-1}^s$ como combinación linear de $N(KN)^0 Kg_0, \dots, N(KN)^{is-1} Kg_0$.

Agora, demóstrase a inclusión (2.35) por indución sobre i:
Para $i = 0$ a inclusión

$$\mathcal{K}_s(KN, Kg_0) \subset \bigoplus_{j=0}^{0} \mathcal{K}_s(KN, Kg_j) \tag{2.39}$$

é trivial. Pola fórmula (2.36), se $i = 1$ entón, para $k \in \{1, \dots, s\}$, tense que

$$(KN)^{k-1} Kg_1 = (KN)^{k-1} Kg_0 + \sum_{j=0}^{s-1} \lambda_j (KN)^{j+k} Kg_0. \tag{2.40}$$

Se se proba que $\lambda_{s-1} \neq 0$ obtense, despexando na fórmula anterior para $k = 1, \dots, s$, que

$$(KN)^s Kg_0, \dots, (KN)^{2s-1} Kg_0 \in \bigoplus_{j=0}^{1} \mathcal{K}_s(KN, Kg_j) \tag{2.41}$$

e deste xeito tense que $\mathcal{K}_{2s}(KN, Kg_0) \subset \bigoplus_{j=0}^{1} \mathcal{K}_s(KN, Kg_j)$. Mais $\lambda_{s-1} \neq 0$ pois, polo apartado *b)* do lema 2.1 e a igualdade (2.25),

$$\begin{aligned} < g_1, Kg_1 > = < g_1, Kg_0 + \sum_{j=0}^{s-1} \lambda_j (KN)^{j+1} Kg_0 > = \\ = \lambda_{s-1} < g_1, (KN)^s Kg_0 >, \end{aligned} \tag{2.42}$$

e se $g_1 \neq 0$, entón $< g_1, Kg_1 > \neq 0$, pois K é unha matriz coa parte simétrica definida positiva.

Por último, supoñamos que a inclusión

$$\mathcal{K}_{si}(KN, Kg_0) \subset \bigoplus_{j=0}^{i-1} \mathcal{K}_s(KN, Kg_j) \tag{2.43}$$

é certa para $i - 1$. Entón, se $k \in \{1, \dots, s\}$, pola ecuación (2.36),

$$(KN)^{k-1} Kg_i = (KN)^{k-1} Kg_0 + \sum_{j=0}^{is-1} \lambda_j (KN)^{j+k} Kg_0, \tag{2.44}$$

e bastará comprobar que $\lambda_{is-1} \neq 0$ para concluír que

$$\mathcal{K}_{s(i+1)}(KN, Kg_0) \subset \bigoplus_{j=0}^{i} \mathcal{K}_s(KN, Kg_j). \tag{2.45}$$

Mais isto é certo pois se, polo contrario, fora $\lambda_{is-1} = 0$ na ecuación (2.36), teríase que

$$Kg_i \in \mathcal{K}_{is}(KN, Kg_0) \tag{2.46}$$

e, pola hipótese de indución e a igualdade (2.25),

$$Kg_i \in \pounds\{P_0, \dots, P_{i-1}\}, \tag{2.47}$$

o que leva, polo apartado *b)* do lema 2.1, a que

$$< g_i, Kg_i > = 0, \tag{2.48}$$

o cal, se $g_i \neq 0$, é unha contradicción, pois a parte simétrica da matriz K é definida positiva.

A última igualdade

$$\mathcal{K}_{s(i+1)}(KN, Kg_0) = \pounds\{p_0, p_1, \dots, p_{(i+1)s-1}\} \tag{2.49}$$

é a demostrada no lema 1.4. ■

O s-AXO minimiza tamén o funcional $E(r) =< r, H^S r >$, como se demostra no seguinte resultado:

Lema 2.4 *Para cada $i = 0, 1, 2, \dots$, o residuo do Algoritmo s-AXO, r_{i+1}, minimiza o funcional*

$$E(r) =< r, H^S r > \tag{2.50}$$

sobre o subespazo afín

$$x_0 + \pounds\{P_0, \dots, P_i\}. \tag{2.51}$$

Demostración:

Na liña do que se fai na demostración do lema análogo para o AXO, que se pode ver en [51] ou [4], sexa r_{i+1} o residuo correspondente ao iterante x_{i+1}. Posto que $r_{i+1} = r_i - AP y_i$, pódese comprobar sen dificultade por indución que

$$r_{i+1} = r_0 - \sum_{k=0}^{i} (AP_k y_k)\,. \tag{2.52}$$

Substituíndo entón o valor de r_{i+1} en $E(r)$ e desenvolvendo tense que

$$\begin{aligned} E(r_{i+1}) =< r_0, H^S r_0 > -2 < \sum_{k=0}^{i} (AP_k y_k)\,, H^S r_0 > + \\ < \sum_{k=0}^{i} (AP_k y_k)\,, H^S \sum_{k=0}^{i} (AP_k y_k) > \end{aligned} \tag{2.53}$$

mais, usando a definición (2.9) e o apartado *e)* do lema 2.2, o segundo sumando do membro dereito da ecuación (2.53) pode simplificarse tendo en conta que

$$< \sum_{k=0}^{i} (AP_k y_k)\,, H^S r_0 >= \sum_{k=0}^{i} < g_0, P_k y_k, >= \sum_{k=0}^{i} < (P_k)^t g_k, y_k > . \tag{2.54}$$

E o terceiro sumando do membro dereito da ecuación (2.53) tamén pode simplificarse tendo en conta que $N = A^t H^S A$, a definición (2.5) e usando o apartado *a)* do lema

2.1, co que se ten

$$\begin{array}{c} < \sum_{k=0}^{i} (AP_k y_k) \, , H^S \sum_{k=0}^{i} (AP_k y_k) >= \\ \sum_{k=0}^{i} < (P_k)^t N P_k y_k, y_k >= \sum_{k=0}^{i} < W_k y_k, y_k > . \end{array} \tag{2.55}$$

Deste xeito, queda

$$E(r_{i+1}) =< r_0, H^S r_0 > -2 \sum_{k=0}^{i} < (P_k)^t g_k, y_k > + \sum_{k=0}^{i} < W_k y_k, y_k > . \tag{2.56}$$

Dado que $E(r)$ é convexo, o feito de que r_{i+1} sexa o mínimo de $E(r)$ sobre $x_0 + \pounds\{P_0, \ldots, P_i\}$ equivale a que os vectores de coeficientes y_k, con $k = 0, \ldots, i$, sexan as solucións dos sistemas lineares

$$W_k y_k = (P_k)^t g_k, \tag{2.57}$$

mais estes sistemas son precisamente os que definen aos coeficientes y_k no s-AXO, como pode deducirse das definicións (2.6) e (2.7). ■

Observación 2.2 *Como consecuencia do lema 2.3, para cada $i = 0, 1, 2, \ldots$, as matrices P_i teñen rango s. Da definición de W_i e o feito de que N sexa definida positiva, tense que $v^t W_i v =< P_i v_i, N P_i v_i >$ é estrictamente positivo para todo $v \in \mathbb{R}^s$, $v \neq 0$. Polo tanto, a matriz W_i é definida positiva e, en consecuencia, non singular.*

Observación 2.3 *Sexan $\tilde{r}_i$ e r_i os residuos calculados na i-ésima iteración do AXO e do s-AXO, respectivamente. Posto que $E(r)$ é unha función convexa e grazas aos lemas 2.3 e 2.4, se x_0 é o mesmo vector inicial nos algoritmos AXO e s-AXO entón $\tilde{r}_{is} = r_i$ en aritmética exacta.*

Dos lemas que se acaban de demostrar, pódese deducir o seguinte resultado de converxencia para o s-AXO:

Teorema 2.1 *Sexan $i, s \in \mathbb{N}$ tales que $s(i+1) \leq n$, e suponse que $g_i \neq 0$ para todo $i = 0, 1, 2, \ldots$ ata a converxencia. Se $grao_{KN}(Kg_0) \geq s(i+1)$, tense que a variante en s-pasos do Algoritmo Xeral de Ortogonalización converxe en ao sumo $[n/s]$ iteracións.*

Demostración:

Sexa $i \in \mathbb{N}$, como consecuencia dos lemas anteriores, tense que, se $g_i \neq 0$, entón g_i é ortogonal a $\mathcal{K}_{is}(KN, Kg_0)$. Ademais

$$\dim \mathcal{K}_{is}(KN, Kg_0) = is, \tag{2.58}$$

polo tanto, se $is \geq n$, conclúese necesariamente que debe ser $g_i = 0$, o cal implica que $r_i = 0$, xa que $g_i = A^t H^S r_i$ e $A^t H^S$ é non singular. ■

Ademais, enúnciase tamén un teorema de estimación do erro, obtendo fórmulas análogas as ecuacións (1.51) e (1.52).

Teorema 2.2 *Nas mesmas hipóteses que o teorema 2.1, se r_i é o residuo da i-ésima iteración do algoritmo s-AXO e $E_i = E(r_i)$, verifícase:*

$$E_i \leq E_0 \left(1 - \frac{\lambda_{\text{mín}}(L^t(K^{-1})^S L)}{cond(M)}\right)^{is}. \tag{2.59}$$

Ademais, se a matriz K é simétrica, tense:

$$E_i \leq E_0 \left(\frac{cond(M) - 1}{cond(M) + 1}\right)^{2is}. \tag{2.60}$$

Demostración:

A demostración é obvia a partir das ecuacións (1.51) e (1.52) e da observación 2.3. ■

De xeito análogo ao que ocurre no AXO, no s-AXO necesítanse todas as matrices P_j, $j = 0, \ldots, i$ obtidas nas iteracións anteriores para o cálculo de B_{i+1}^j. Este feito causa un crecemento da memoria requerida polo sistema a medida que aumenta o número de iteracións. En consecuencia, se o número de iteracións requerido é alto, o método pode orixinar un erro por rebosamento da memoria. Grazas ao seguinte lema, no caso en que a matriz K sexa simétrica, tan só será necesario almacenar a matriz P_i do iterante anterior para o cómputo de B_{i+1}^j.

Lema 2.5 *Nas hipóteses do teorema 2.1, se a matriz K é simétrica, entón, para $j = 0, \ldots, i-1$*

$$(P_j)^t N Q_{i+1} = 0. \tag{2.61}$$

Demostración:

A demostración deste lema xeneraliza a da propiedade análoga para o AXO, que se pode ver en [51] ou [4], e segue un razoamento similar baseado nas propiedades de ortogonalidade do mesmo.

Sexa $j \in \{0, \dots, i-1\}$ fixo. Entón $(P_j)^t N Q_{i+1}$ é unha matriz cadrada de orde s tal que o elemento kl é $< (KN)^{k-1} K g_{i+1}, N p_j^l >$, con $k, l \in \{1, \dots, s\}$. Se K é simétrica, entón

$$< (KN)^{k-1} K g_{i+1}, N p_j^l > = < (g_{i+1}, (KN)^k p_j^l > . \tag{2.62}$$

Polo lema 2.3 tense que $p_j^l \in \mathcal{K}_{s(j+1)}(KN, Kg_0)$ para todo $l = 1, \dots, s$. Así, se $j \leq i-1$ e $k \in \{1, \dots, s\}$, entón

$$(KN)^k p_j^l \in \mathcal{K}_{s(i+1)}(KN, Kg_0), \tag{2.63}$$

pois $k + (j+1)s - 1 \leq s + is - 1 = s(i+1) - 1$. Mais, polo apartado *b)* do lema 2.1 e o lema 2.3, g_{i+1} é ortogonal a $P_0, \dots, P_i$. As columnas destas matrices xeran $\mathcal{K}_{s(i+1)}(KN, Kg_0)$, polo que g_{i+1} é ortogonal a $\mathcal{K}_{s(i+1)}(KN, Kg_0)$ e, deste xeito, conclúese que, se $0 \leq j \leq i-1$, o termo da dereita da ecuación (2.62) é cero. ■

Polo tanto, se K é simétrica, as ecuacións (2.11) e (2.12) do s-AXO son:

$$B_{i+1} = -W_i^{-1} (P_i)^t N Q_{i+1} \tag{2.64}$$

e

$$P_{i+1} = Q_{i+1} + P_i B_{i+1}. \tag{2.65}$$

E o Algoritmo Xeral de Ortogonalización en s-pasos, no caso de que K sexa simétrica, queda como segue:

Algoritmo 2.2 (s-AXO, K simétrica).
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0$
$g_0 = A^t H^S r_0$
$P_0 = \Delta_{KN}(Kg_0) = Q_0$
Para $i = 0, 1, 2, \dots$ *ata converxencia*

$$W_i = (P_i)^t N P_i \tag{2.66}$$

$$z_i = (P_i)^t g_i \tag{2.67}$$

$$y_i = (W_i)^{-1} z_i \tag{2.68}$$

$$x_{i+1} = x_i + P_i y_i \tag{2.69}$$

$$g_{i+1} = g_i - NP_i y_i = A^t H^S r_{i+1} \tag{2.70}$$

$$Q_{i+1} = \Delta_{KN}(Kg_{i+1}) \tag{2.71}$$

$$B_{i+1} = -W_i^{-1}(P_i)^t N Q_{i+1} \tag{2.72}$$

$$P_{i+1} = Q_{i+1} + P_i B_{i+1} \tag{2.73}$$

Fin

No caso máis xeral, cando a matriz K non é simétrica, pódese resolver o problema de almacenamento usando un número de direccións anteriores predeterminado nunha variante do algoritmo coñecida por método Orthomin(m) [80]. A falta de N-ortogonalidade con todas as direccións anteriores conleva a converxencia do método non necesariamente nun número finito de iteracións.

2.2. Variante en s-pasos do método Orthomin(m)

Se a matriz K non é simétrica, o Algoritmo Xeral de Ortogonalización e a súa variante en s-pasos requiren do almacenamento de todas as direccións anteriores a cada iteración. Para evitar este feito, a variante Orthomin(m), proposta en [80], ten en conta unicamente as m direccións anteriores, no caso do Algoritmo Xeral de Ortogonalización, ou os m subespazos de Krylov anteriores, no caso da súa variante en s-pasos. A variante Orthomin(m) do AXO é:

Algoritmo 2.3 (Orthomin(m)).
Sexa $x_0 \in \mathbb{R}^n$ un vector inicial distinto da solución do sistema e $m \in \mathbb{N}$:

1. ***Inicio:***
   - $r_0 = b - Ax_0 = A(x - x_0)$
   - $g_0 = A^t H^S r_0 = A^t H^S A(x - x_0) = N(x - x_0)$
   - $p_0 = Kg_0$
2. ***Iteraciones:*** *Para $i = 0, 1, ...,$*
   - $\alpha_i = \dfrac{< g_i, p_i >}{< p_i, Np_i >}$
   - $x_{i+1} = x_i + \alpha_i p_i$
   - $g_{i+1} = g_i - \alpha_i N p_i = A^t H^S r_{i+1}$

- $\beta_{i+1}^{l} = -\dfrac{< Kg_{i+1}, Np_l >}{< p_l, Np_l >}, \quad i-m+1 \leq l \leq i$
- $p_{i+1} = Kg_{i+1} + \displaystyle\sum_{l=i-m+1}^{i} \beta_{i+1}^{l} p_l$

Fin

De xeito análogo á variante en s-pasos ao Algoritmo Xeral de Ortogonalización, se para cada $i = 0, 1, 2, ...$ ata a converxencia $\text{grao}_{KN}(Kg_i) \geq s$, proponse unha variante en s-pasos do Orthomin(m):

Algoritmo 2.4 (**s-Orthomin(m)**).
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0$
$g_0 = A^t H^S r_0$
$P_0 = \Delta_{KN}(Kg_0) = Q_0$
Para $i = 0, 1, 2, ...$ *ata converxencia*

$$W_i = (P_i)^t N P_i \tag{2.74}$$
$$z_i = (P_i)^t g_i \tag{2.75}$$
$$y_i = (W_i)^{-1} z_i \tag{2.76}$$
$$x_{i+1} = x_i + P_i y_i \tag{2.77}$$
$$g_{i+1} = g_i - NP_i y_i = A^t H^S r_{i+1} \tag{2.78}$$
$$Q_{i+1} = \Delta_{KN}(Kg_{i+1}) \tag{2.79}$$

Para $j = i-m+1, ..., i$:

$$B_{i+1}^{j} = -W_j^{-1} (P_j)^t N Q_{i+1} \tag{2.80}$$

$$P_{i+1} = Q_{i+1} + \sum_{j=i-m+1}^{i} P_j B_{i+1}^{j} \tag{2.81}$$

Fin

Para a variante en s-pasos do Orthomin(m) tense o seguinte lema de demostración análoga ás demostracións dos lemas 2.1 e 2.4:

Lema 2.6 *Sexan $i, s, m \in \mathbb{N}$ tales que $s(i+1) \leq n$, e suponse que $g_i \neq 0$ para todo $i = 0, 1, 2, ...$ ata a converxencia. Se $grao_{KN}(Kg_0) \geq s(i+1)$ na variante en s-pasos do Orthomin(m), entón:*

a) *P_i é N-ortogonal a P_j para todo $i - m \leq j \leq i - 1$, $i \geq m$.*

b) *$(P_j)^t g_i = 0$ para todo $i - m \leq j \leq i - 1$, $i \geq m$.*

c) *r_{i+1} minimiza $E(r)$ sobre $x_{i-m+1} + \pounds\{P_{i-m+1}, ..., P_i\}$.*

A converxencia do s-AXO nun número finito de iteracións é consecuencia da ortogonalidade do residuo xeneralizado con todas as matrices P_j calculadas nas iteracións anteriores. Mais na variante en s-pasos do Orthomin(m), o residuo xeneralizado é ortogonal soamente coas matrices P_j calculadas nas últimas iteracións. Debido a isto, non podemos deducir a converxencia da variante en s-pasos do Orthomin(m) cun argumento similar ao utilizado no s-AXO. Para obter un teorema de converxencia deste método demóstranse antes os seguintes lemas, nos que se supón que se cumpren as hipóteses do lema 2.6:

Lema 2.7 *Se se denota por $E_i = E(r_i) =< r_i, H^S r_i >$ para cada $i = 0, 1, ...$, entón cúmprese que:*

$$E_{i+1} = E_i - (P_i^t g_i)^t W_i^{-1} P_i^t g_i \tag{2.82}$$

e así:

$$\lim_{i \to \infty} (P_i^t g_i)^t W_i^{-1} P_i^t g_i = 0. \tag{2.83}$$

Demostración:

Tense que

$$E_{i+1} =< r_{i+1}, H^S r_{i+1} >=< r_i - AP_i y_i, H^S (r_i - AP_i y_i) > \tag{2.84}$$

desenvolvendo o último termo, e tendo en conta que $N = A^t H^S A$, obtense

$$E_{i+1} = E_i - 2 < AP_i y_i, H^S r_i > + < P_i y_i, NP_i y_i > . \tag{2.85}$$

Pero, usando que $W_i = (P_i)^t N P_i$, e substituíndo $y_i = W_i^{-1}(P_i)^t g_i$

$$\begin{aligned} < P_i y_i, NP_i y_i >= y_i^t W_i y_i = (W_i^{-1}(P_i)^t g_i)^t W_i (W_i^{-1}(P_i)^t g_i) = \\ = g_i^t P_i W_i^{-1} (P_i)^t g_i. \end{aligned} \tag{2.86}$$

Ademais, tendo en conta tamén que $g_i = A^t H^S r_i$, obtense:

$$< AP_i y_i, H^S r_i >= (P_y W_i^{-1}(P_i)^t g_i)^t g_i = g_i^t P_i W_i^{-1} (P_i)^t g_i. \tag{2.87}$$

Substituíndo agora na ecuación (2.85) conclúese que:

$$E_{i+1} = E_i - (P_i^t g_i)^t W_i^{-1} P_i^t g_i. \tag{2.88}$$

A segunda parte do lema é consecuencia directa de que E_i é unha sucesión decrecente, acotada inferiormente ao ser non negativa, e polo tanto converxente. Deste xeito

$$\lim_{i\to\infty} E_i = \lim_{i\to\infty} E_{i+1} \tag{2.89}$$

e así

$$\lim_{i\to\infty} (P_i^t g_i)^t W_i^{-1} P_i^t g_i = \lim_{i\to\infty} E_i - \lim_{i\to\infty} E_{i+1} = 0. \tag{2.90}$$

■

Lema 2.8 *Lémbrese que $\lambda_{\text{mín}}(K^S)$ é o mínimo autovalor da parte simétrica da matriz K. Entón cúmprese:*

$$\left\| P_i^t g_i \right\| \geq \lambda_{\text{mín}}(K^S) \left\| g_i \right\|^2. \tag{2.91}$$

Demostración:

Multiplicando por g_{i+1} en (2.81) e tendo en conta o apartado *b)* do lema 2.6, tense que

$$P_i^t g_i = Q_i^t g_i. \tag{2.92}$$

Deste xeito

$$\left\| P_i^t g_i \right\|^2 = \left\| Q_i^t g_i \right\|^2 = \sum_{k=0}^{s-1} \left(< (KN)^k K g_i, g_i >^2 \right). \tag{2.93}$$

Así:

$$\left\| P_i^t g_i \right\|^2 \geq < K g_i, g_i >^2 = < K^S g_i, g_i >^2 \tag{2.94}$$

e, dado que K^S é simétrica definida positiva, conclúese que

$$\left\| P_i^t g_i \right\|^2 \geq \left(\lambda_{\text{mín}}(K^S) \left\| g_i \right\|^2 \right)^2 \tag{2.95}$$

e, posto que $\lambda_{\text{mín}}(K^S) > 0$, tense o resultado que se quere demostrar. ■

Lema 2.9 *Lémbrese que $\lambda_{\text{máx}}(W_i)$ e $\lambda_{\text{máx}}(N)$ son os autovalores máximos de W_i e N, respectivamente, para cada $i \in \mathbb{N}$. Entón cúmprese a seguinte desigualdade:*

$$\lambda_{\text{máx}}(W_i) \leq \lambda_{\text{máx}}(N) \left(\|K\| + \|KNK\| + \cdots + \left\|(KN)^{(s-1)}K\right\| \right)^2 \|g_i\|^2 . \tag{2.96}$$

Demostración:

Sexa $v \in \mathbb{R}^s$ autovector de W_i asociado ao autovalor $\lambda_{\text{máx}}(W_i)$ e tal que $\|v\| = 1$, e polo tanto:

$$\lambda_{\text{máx}}(W_i) = v^t W_i v = v^t (P_i)^t N P_i v. \tag{2.97}$$

Por outra banda, por (2.81)

$$(P_i)^t N P_i = \left(Q_i + \sum_{j=i-m}^{i-1} P_j B_i^j \right)^t N \left(Q_i + \sum_{j=i-m}^{i-1} P_j B_i^j \right). \tag{2.98}$$

Usando as propiedades de ortogonalidade do lema 2.6, obtense que

$$\begin{aligned} (P_i)^t N P_i = Q_i^t N Q_i + \sum_{j=i-m}^{i-1} Q_i^t N P_j B_i^j + \sum_{j=i-m}^{i-1} (P_j B_i^j)^t N Q_i + \\ + \sum_{j=i-m}^{i-1} (P_j B_i^j)^t N P_j B_i^j, \end{aligned} \tag{2.99}$$

mais, tendo en conta (2.80) e a simetría das matrices N e W_j:

$$Q_i^t N P_j B_i^j = -Q_i^t N P_j W_j^{-1} (P_j)^t N Q_i, \tag{2.100}$$

$$\begin{aligned} (P_j B_i^j)^t N Q_i = -(W_j^{-1}(P_j)^t N Q_i)^t (P_j)^t N Q_i = \\ = -Q_i^t N P_j W_j^{-1} (P_j)^t N Q_i \end{aligned} \tag{2.101}$$

e

$$\begin{aligned} (P_j B_i^j)^t N P_j B_i^j = (W_j^{-1}(P_j)^t N Q_i)^t (P_j)^t N P_j (W_j^{-1}(P_j)^t N Q_i) = \\ = Q_i^t N P_j W_j^{-1} (P_j)^t N Q_i, \end{aligned} \tag{2.102}$$

xa que $(P_j)^t N P_j W_j^{-1} = W_j W_j^{-1} = I$, pódese substituír na ecuación (2.99) e obter

$$(P_i)^t N P_i = Q_i^t N Q_i - \sum_{j=i-m}^{i-1} Q_i^t N P_j W_j^{-1} (P_j)^t N Q_i. \tag{2.103}$$

Posto que as matrices $Q_i^t N P_j W_j^{-1} (P_j)^t N Q_i$ son simétricas e definidas positivas, pódese afirmar que:

$$v^t (P_i)^t N P_i v \leq v^t Q_i^t N Q_i v \leq \lambda_{\text{máx}}(N) \|Q_i v\|^2 . \tag{2.104}$$

Por outra banda, pola definición de Q_i, se $(v_1, \ldots, v_s)$ son as coordenadas do vector v respecto da base canónica de $\mathbb{R}^s$, pódese escribir

$$\|Q_i v\| = \left\| v_1 K g_i + v_2 (NK) K g_i + \cdots + v_s (NK)^{s-1} K g_i \right\| . \tag{2.105}$$

Ademais, tendo en conta a desigualdade triangular e que $\|v\| = 1$,

$$\left\|v_1 K g_i + \cdots + v_s (NK)^{s-1} g_i\right\| \leq \left(\|K\| + \cdots + \left\|(NK)^{s-1} K\right\|\right) \|g_i\| \tag{2.106}$$

e deste xeito, conclúese que

$$\begin{gathered} \lambda_{\text{máx}}(W_i) = v^t (P_i)^t N P_i v \leq \\ \leq \lambda_{\text{máx}}(N) \left(\|K\| + \|(NK)K\| + \cdots + \|(NK)^{s-1} K\|\right)^2 \|g_i\|^2 . \end{gathered} \tag{2.107}$$

■

Teorema 2.3 *Baixo as hipóteses do lema 2.6, na variante en s-pasos do método Orthomin(m) verifícase que*

$$\lim_{i \to \infty} r_i = 0. \tag{2.108}$$

Demostración:

Polo lema 2.7 tense que

$$E_{i+1} = E_i - (P_i^t g_i)^t W_i^{-1} P_i^t g_i. \tag{2.109}$$

Obviamente, a sucesión E_i é non negativa e decrecente, pois

$$E_i =< r_i, H^S r_i >\geq 0 \tag{2.110}$$

e

$$(P_i^t g_i)^t W_i^{-1} P_i^t g_i \geq 0. \tag{2.111}$$

Deste xeito poden darse dous casos:

a) Existe $i \in \mathbb{N}$ tal que $E_i = E_{i+1}$. Entón $(P_i^t g_i)^t W_i^{-1} P_i^t g_i = 0$, o que implica que $P_i^t g_i = 0$, pois W_i^{-1} é definida positiva. Polo apartado *b)* do lema 2.6 e a definición de P_i, tense entón que

$$Q_i^t g_i = 0, \tag{2.112}$$

en particular, $< K g_i, g_i >= 0$ e, polo tanto, $g_i = 0$, co que o método converxe nun número finito de iteracións.

b) Para todo $i \in \mathbb{N}$ verifícase que $E_i < E_{i+1}$. Entón E_i é estrictamente decrecente e acotada inferiormente polo que converxe e así:

$$\lim_{i \to \infty} (E_i - E_{i+1}) = 0 = \lim_{i \to \infty} (P_i^t g_i)^t W_i^{-1} P_i^t g_i. \tag{2.113}$$

Deste xeito, para cada $\epsilon > 0$, existe $k \in \mathbb{N}$ tal que

$$(P_k^t g_k)^t W_k^{-1} P_k^t g_k < \epsilon. \tag{2.114}$$

Por outra banda,

$$(P_k^t g_k)^t W_k^{-1} P_k^t g_k \geq \lambda_{\text{mín}}(W_k^{-1}) \left\| P_k^t g_k \right\|^2 = \frac{1}{\lambda_{\text{máx}}(W_k)} \left\| P_k^t g_k \right\|^2 \tag{2.115}$$

e, aplicando os lemas anteriores 2.8 e 2.9, obtense que

$$\begin{gathered} \frac{1}{\lambda_{\text{máx}}(W_k)} \left\| P_k^t g_k \right\|^2 \geq \\ \geq \frac{\lambda_{\text{mín}}(K^S)}{\lambda_{\text{máx}}(N) \left(\|K\| + \|KNK\| + \cdots + \|(KN)^{(s-1)}K\| \right)^2} \|g_k\|^2 . \end{gathered} \tag{2.116}$$

Tense probado pois que, para cada $\epsilon > 0$, existe $k \in \mathbb{N}$ tal que

$$\|g_k\|^2 < \epsilon \frac{\lambda_{\text{máx}}(N) \left(\|K\| + \|KNK\| + \cdots + \left\|(KN)^{(s-1)}K\right\| \right)^2}{\lambda_{\text{mín}}(K^S)}, \tag{2.117}$$

o que implica que

$$\lim_{i \to \infty} g_i = 0 \tag{2.118}$$

e, como consecuencia,

$$\lim_{i \to \infty} r_i = 0. \tag{2.119}$$

■

2.3. N-Ortonormalización das direccións de descenso

En cada iteración da variante s-pasos do Algoritmo Xeral de Ortogonalización é necesario resolver un pequeno sistema linear de orde s con matriz de coeficientes W_i. Se s é relativamente grande ($s \geq 6$) [29], a resolución numérica en cada iteración destes sistemas pode provocar a perda de estabilidade numérica do algoritmo principal. Baseándose no mesmo artigo, proponse a continuación unha alternativa para evitar este problema, que consiste en N-ortonormalizar, na iteración i-ésima, as direccións $\{p_i^1, \ldots, p_i^s\}$ segundo a definición que se deu anteriormente no capítulo 1 para dous vectores, e que se pode xeneralizar para un conxunto de k vectores coa seguinte definición.

Definición 2.2 *Sexa N unha matriz simétrica definida positiva de orde $n \times n$. Dícese que o conxunto de vectores $\{v_1, \dots, v_k\} \subset \mathbb{R}^n$, con $k \leq n$, son N-ortonormais se:*

- $< v_i, Nv_i >= 1$ *para todo* $i = 1, \dots, k$.
- $< v_i, Nv_j >= 0$ *para todo* $i \neq j$ *con* $i, j \in \{1, \dots, k\}$.

Na liña do que se fai co método de Gram-Schmidt ([72], por exemplo), a partir dun conxunto de vectores linearmente independente pode construírse outro conxunto de vectores N-ortonormais coa propiedade de ir xerando os mesmos subespazos vectoriais que os orixinais, no senso que mostra o seguinte lema:

Lema 2.10 *Sexan $\{p_1, \dots, p_s\} \subset \mathbb{R}^n$, con $s \leq n$, un conxunto de vectores linearmente independente, e $\{v_1, \dots, v_s\} \subset \mathbb{R}^n$ tales que:*

1. $v_1 = \dfrac{p_1}{< p_1, Np_1 >^{1/2}}$.
2. *Para cada* $j = 2, \dots, s$, *se* $q_j^1 = p_j$, *entón para* $k = 1, \dots, j-1$,

$$q_j^{k+1} = p_j - \sum_{m=1}^{k} \left(< q_j^m, Nv_m > v_m \right) \tag{2.120}$$

3. $v_j = \dfrac{q_j^j}{< q_j^j, Nq_j^j >^{1/2}}$.

Entón:

a) *Para cada* $j = 1, \dots, s$, $\pounds\{v_1, \dots, v_j\} = \pounds\{p_1, \dots, p_j\}$.

b) *Os vectores* $\{v_1, \dots, v_s\}$ *son N-ortonormais.*

Demostración:

a) Da definición de v_1 é obvio que $\pounds\{v_1\} = \pounds\{p_1\}$. Sexa $j \in \{1, \dots, s\}$ e suponse agora certo que

$$\pounds\{v_1, \dots, v_{j-1}\} = \pounds\{p_1, \dots, p_{j-1}\},$$

entón, usando a definición de v_j e despexando p_j en (2.120), tense

$$p_j =< q_j^j, Nq_j^j >^{1/2} v_j + \sum_{m=1}^{j-1} \left(< q_j^m, Nv_m > v_m \right) \tag{2.121}$$

e $p_j \in \pounds\{v_1, \dots, v_j\}$. Por outra banda, despexano v_j na mesma ecuación (2.121) e usando a hipótese de indución, dedúcese que $v_j \in \pounds\{p_1, \dots, p_j\}$. Co que se pode concluír que

$$\pounds\{v_1, \dots, v_j\} = \pounds\{p_1, \dots, p_j\}.$$

b) Da definición de $v_1, \dots, v_s$ dedúcese de xeito inmediato que $< v_i, Nv_i >= 1$, para todo $i \in 1, \dots, s$. Posto que N é simétrica, para probar que $< v_i, Nv_j >= 0$ para todo $i \neq j$, bastará facelo para $i > j$. Por simplicidade na notación, defínese $q_1^1 = p_1$. Así, tendo en conta que

$$v_j = \frac{q_j^j}{< q_j^j, Nq_j^j >^{1/2}} \text{ para todo } j \in 1, \dots, s, \tag{2.122}$$

demostrarase que

$$< q_i^i, Nq_j^j >= 0 \text{ para } i > j. \tag{2.123}$$

A demostración faise por indución en i. Dado que $i > j \geq 1$, próbase primeiro que a (2.123) é certa para $i = 2$ e $j = 1$. Da definición de q_2^2 en (2.122), e tendo en conta que $q_2^1 = p_2$, queda

$$\begin{gathered} < q_2^2, Nq_1^1 >=< p_2 - < p_2, Nv_1 > v_1, Np_1 >= \\ =< p_2, Np_1 > - < p_2, Nv_1 >< v_1, Np_1 > \end{gathered} \tag{2.124}$$

e utilizando a definición de v_1, o último sumando do segundo membro de (2.124) pódese escribir como

$$< p_2, Nv_1 >< v_1, Np_1 >= \frac{< p_2, Np_1 >< p_1, Np_1 >}{< p_1, Np_1 >} =< p_2, Np_1 > \tag{2.125}$$

e así a expresión (2.124) é nula.

Agora suponse que (2.123) é certa para $i - 1 > j$, entón

$$\begin{gathered} < q_i^i, Nq_j^j >=< p_i - \sum_{m=1}^{i-1} \left(< q_i^m, Nv_m > v_m \right), Nq_j^j >= \\ =< p_i, Nq_j^j > - \sum_{m=1}^{i-1} \left(< q_i^m, Nv_m >< v_m, Nq_j^j > \right). \end{gathered} \tag{2.126}$$

Tendo en conta a definición de v_m e a hipótese de indución,

$$\begin{gathered} \sum_{m=1}^{i-1} \left(< q_i^m, Nv_m >< v_m, Nq_j^j > \right) =< q_i^j, Nv_j >< v_j, Nq_j^j >= \\ = \frac{< q_i^j, Nq_j^j >< q_j^j, Nq_j^j >}{< q_j^j, Nq_j^j >} =< q_i^j, Nq_j^j >, \end{gathered} \tag{2.127}$$

polo que a ecuación (2.126) queda

$$< q_i^i, Nq_j^j >=< p_i, Nq_j^j > - < q_i^j, Nq_j^j >=< p_i - q_i^j, Nq_j^j > . \tag{2.128}$$

E de (2.120) e a hipótese de indución:

$$< p_i - q_i^j, Nq_j^j >=< \sum_{m=1}^{j-1} \left(< q_i^m, Nv_m > v_m\right), Nq_j^j >= 0 \qquad (2.129)$$

e, deste xeito, a expresión (2.126) é 0 e queda probado (2.123). ■

Dada entón a matriz simétrica definida positiva N e un conxunto de vectores linearmente independentes $\{p_1, \dots, p_s\}$, escríbese o seguinte algoritmo de N-ortonormalización para ditos vectores $\{p_1, \dots, p_s\}$, baseado no algoritmo modificado de Gram-Schmidt ([72], por exemplo):

Algoritmo 2.5 (De N-ortonormalización).
$\alpha_1^1 =< p_1, Np_1 >^{1/2}$
Se $\alpha_1^1 = 0$ *entón o algoritmo para. No caso contrario,* $v_1 = p_1/\alpha_1^1$.
Para $j = 2, \dots, s$, *calcular:*

- $q_j^1 = p_j$
- ***Para*** $k = 1, \dots, j-1$,
  - $\alpha_j^k =< q_j^k, Nv_k >$
  - $q_j^{k+1} = q_j^k - \alpha_j^k v_k$

  Fin do bucle en k.
- $q_j = q_j^j$
- $\alpha_j^j =< q_j, Nq_j >^{1/2}$
- *Se* $\alpha_j^j = 0$ *entón o algoritmo acábase. No caso contrario,* $v_j = q_j/\alpha_j^j$.

Fin do bucle en j.
Fin

Se $P = (p_1|p_2|\dots|p_s)$ e $V = (v_1|v_2|\dots|v_s)$, a ecuación (2.121) pode escribirse de xeito matricial:

$$P = VR, \qquad (2.130)$$

onde R é unha matriz cadrada de orde $s \times s$ triangular superior e non singular formada polos coeficientes α_j^k, con $1 \le k \le j \le s$, do algoritmo 2.5. Esta forma matricial é análoga á coñecida como descomposición QR dunha matriz de orde $n \times s$

e rango s ([72], por exemplo).

En cada iteración do método de ortogonalización en s-pasos aplicaremos o algoritmo de N-ortonormalización sobre as s columnas da matriz P_i, obtendo unha nova matriz V_i de orde $n \times s$ cos vectores columna N-ortonormais. Deste xeito, a matriz de coeficientes do sistema de orde s que había que resolver en cada iteración, $W_i = (P_i)^t N P_i$, convértese na matriz identidade, $I = V_i^t N V_i$, evitándose deste xeito o cálculo de W^{-1} que era necesario no s-AXO. A nova versión do s-AXO N-ortonormalizada queda como segue:

Algoritmo 2.6 (s-AXO N-ortonormalizado).
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0$
$g_0 = A^t H^S r_0$
$P_0 = \Delta_{KN}(Kg_0) = Q_0$
Para $i = 0, 1, 2, ...$ *ata converxencia*

$$N\text{-ortonormalízanse as columnas de } P_i \tag{2.131}$$

$$y_i = (AP_i)^t H^S r_i = P_i^t g_i \tag{2.132}$$

$$x_{i+1} = x_i + P_i y_i \tag{2.133}$$

$$r_{i+1} = r_i - AP_i y_i \tag{2.134}$$

$$g_{i+1} = A^t H^S r_{i+1} \tag{2.135}$$

$$Q_{i+1} = \Delta_{KN}(Kg_{i+1}) \tag{2.136}$$

Para $j = 0, \ldots, i$*:*

$$B_{i+1}^j = -(P_j)^t N Q_{i+1} \tag{2.137}$$

$$P_{i+1} = Q_{i+1} + \sum_{j=0}^{i} P_j B_{i+1}^j \tag{2.138}$$

Fin

O s-AXO N-ortonormalizado que se acaba de propoñer é equivalente en aritmética exacta á variante en s-pasos do Algoritmo Xeral de Ortogonalización, o algoritmo 2.1. Para probar esta afirmación, basta ver que iniciando ambos os dous algoritmos no mesmo iterante inicial x_0, os iterantes e os residuos resultan ser os mesmos en cada iteración. No seguinte lema demóstrase a igualdade dos iterantes e residuos.

Lema 2.11 *Sexan $\tilde{x}_i$ e $\tilde{r}_i$ os iterantes e residuos do s-AXO orixinal (algoritmo 2.1), e x_i e r_i os iterantes e residuos do s-AXO N-ortonormalizado (algoritmo 2.6). Inícianse os dous algoritmos no mesmo iterante inicial x_0, entón, para cada $i = 1, 2, ...$, $\tilde{x}_i = x_i$ e $\tilde{r}_i = r_i$.*

Demostración:

Para diferenciar, úsase o til enriba das variables calculadas en cada iteración do s-AXO orixinal. Deste xeito, no s-AXO orixinal

$$\tilde{x}_{i+1} = \tilde{x}_i + \tilde{P}_i\tilde{y}_i \tag{2.139}$$

e

$$\tilde{r}_{i+1} = \tilde{r}_i - A\tilde{P}_i\tilde{y}_i \tag{2.140}$$

e no s-AXO N-ortonormalizado

$$x_{i+1} = x_i + P_iy_i \tag{2.141}$$

e

$$r_{i+1} = r_i - AP_iy_i. \tag{2.142}$$

A demostración faise por indución sobre i. Inícianse ambos os dous algoritmos no mesmo iterante inicial x_0, e de xeito inmediato $\tilde{r}_0 = b - Ax_0 = r_0$. Ademais, $\tilde{g}_0 = A^tH^Sr_0 = g_0$ e $\tilde{P}_0 = \Delta_{KN}(Kg_0) = P_0$, polo que $\tilde{y}_0 = (P_0)^tg_0 = y_0$ e entón $\tilde{x}_1 = x_0 + P_0y_0 = x_1$ e $\tilde{r}_1 = r_0 - AP_0y_0 = r_1$.

Supóñase agora que as igualdades $\tilde{x}_i = x_i$ e $\tilde{r}_i = r_i$ son certas para certo $i \in \mathbb{N}$. Para demostrar que $\tilde{x}_{i+1} = x_{i+1}$ e $\tilde{r}_{i+1} = r_{i+1}$, basta probar que $\tilde{P}_i\tilde{y}_i = P_iy_i$. De (2.130) dedúcese que existe unha matriz R cadrada, triangular superior e non singular, de orde s, tal que $\tilde{P}_i = P_iR$. Por outra banda, da hipótese de indución $\tilde{g}_i = A^tH^Sr_i = g_i$. Deste xeito, e, tendo en conta da definición de $\tilde{y}_i$ que

$$\tilde{y}_i = (\tilde{W}_i)^{-1}(R_i)^t(P_i)^tg_i, \tag{2.143}$$

tense que

$$\tilde{P}_i\tilde{y}_i = P_iR_i(\tilde{W}_i)^{-1}(R_i)^t(P_i)^tg_i = P_iR_i(\tilde{W}_i)^{-1}(R_i)^ty_i. \tag{2.144}$$

E para concluír que $\tilde{P}_i\tilde{y}_i = P_iy_i$, basta probar que

$$R_i(\tilde{W}_i)^{-1}(R_i)^t = I_{s\times s} \tag{2.145}$$

ou, o que é o mesmo,

$$\left((R_i)^t\right)^{-1}\tilde{W}_i(R_i)^{-1} = I_{s\times s}, \tag{2.146}$$

onde $I_{s\times s}$ é a matriz identidade de orde $s \times s$. Mais, usando a definición de $\tilde{W}_i$,

$$(R_i^t)^{-1}\tilde{W}_i(R_i)^{-1} = (R_i^t)^{-1}(\tilde{P}_i)^t N\tilde{P}_i(R_i)^{-1} = (P_i)^t NP_i, \tag{2.147}$$

que é a identidade por ser as columnas de P_i N-ortonormais. Deste xeito queda demostrado o lema. ■

No caso en que a matriz K sexa simétrica tense o seguinte algoritmo:

Algoritmo 2.7 (s-AXO N-ortonormalizado, K simétrica).
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0$
$g_0 = A^t H^S r_0$
$P_0 = \Delta_{KN}(Kg_0) = Q_0$
Para $i = 0, 1, 2, ...$ *ata converxencia*

$$N\text{-ortonormalízanse as columnas de } P_i \tag{2.148}$$
$$y_i = (AP_i)^t H^S r_i = P_i^t g_i \tag{2.149}$$
$$x_{i+1} = x_i + P_i y_i \tag{2.150}$$
$$r_{i+1} = r_i - AP_i y_i \tag{2.151}$$
$$g_{i+1} = A^t H^S r_{i+1} \tag{2.152}$$
$$Q_{i+1} = \Delta_{KN}(Kg_{i+1}) \tag{2.153}$$
$$B_{i+1} = -(P_i)^t NQ_{i+1} \tag{2.154}$$
$$P_{i+1} = Q_{i+1} + P_i B_{i+1} \tag{2.155}$$

Fin

Analogamente, se a matriz K non é simétrica, o algoritmo Orthomin(m) pode escribirse, N-ortonormalizando as direccións dos subespazos de Krylov, da seguinte maneira:

Algoritmo 2.8 (s-Orthomin(m) N-ortonormalizado).
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0$
$g_0 = A^t H^S r_0$

$P_0 = \Delta_{KN}(Kg_0) = Q_0$
Para $i = 0, 1, 2, ...$ *ata converxencia*

$$N\text{-}ortonormalízanse\ as\ columnas\ de\ P_i \tag{2.156}$$
$$y_i = (AP_i)^t H^S r_i = P_i^t g_i \tag{2.157}$$
$$x_{i+1} = x_i + P_i y_i \tag{2.158}$$
$$r_{i+1} = r_i - AP_i y_i \tag{2.159}$$
$$g_{i+1} = A^t H^S r_{i+1} \tag{2.160}$$
$$Q_{i+1} = \Delta_{KN}(Kg_{i+1}) \tag{2.161}$$

Para $j = i - m + 1, ..., i$*:*

$$B_{i+1}^j = -(P_j)^t N Q_{i+1} \tag{2.162}$$

$$P_{i+1} = Q_{i+1} + \sum_{j=i-m+1}^{i} P_j B_{i+1}^j \tag{2.163}$$

Fin

Observación 2.4 *Os lemas de ortogonalidade 2.1 e 2.6 son válidos igualmente para estas versións coas direccións N-ortonormalizadas.*

Capítulo 3

Casos particulares do s-AXO

Neste capítulo elixiranse convenientemente as matrices H e K para obter as variantes en s-pasos de métodos coñecidos como casos particulares do s-AXO. En concreto,

a) Se a matriz K elixida é simétrica o algoritmo que se usará é o s-AXO N-ortonormalizado para K simétrica (algoritmo 2.7).

b) Se a matriz K elixida non é simétrica, entón o algoritmo que se utilizará é o s-Orthomin(m) N-ortonormalizado (algoritmo 2.8), obténdose á súa vez métodos concretos para distintos valores de m.

Destes métodos, foron publicados previamente o s-Gradente Conxugado e o s-Residuo Conxugado en [24], o s-Residuo Conxugado Xeneralizado e o s-Residuo Minimal de Axelsson que aparecen propostos en [23], o s-Gradente Conxugado precondicionado en [25] e o s-Ecuación Normal en [22]. A nosa innovación é proporcionar un marco unificado de onde todos eles se derivan, así como versións N-ortonormalizadas dos devanditos métodos con mellores propiedades de estabilidade numérica, o que posibilita a utilización de maiores valores de s para maior velocidade de execución. Así mesmo, proponse un método novo, o s-Erro Minimal ([3]).

3.1. Variante en s-pasos do Gradente Conxugado

O método do Gradente Conxugado foi orixinalmente proposto en [48]. É probablemente o método máis coñecido entre os métodos iterativos baseados en subespazos de Krylov. O método converxe cando a matriz de coeficientes do sistema é simétrica e definida positiva. A variante en s-pasos do Gradente Conxugado (s-GC) foi a

primeira variante en s-pasos publicada por Chronopoulos e Gear [24].

Nesta sección vólvese a escribir o algoritmo 1.2 (s-GC) como caso particular do algoritmo 2.7, de xeito que se obtén unha versión A-ortonormalizada do s-GC.

Suponse que a matriz A é simétrica e definida positiva. Para derivar este algoritmo do s-AXO proposto, escóllese $H = A^{-1}$ e $K = I$. Entón $N = A$, e posto que a matriz K é simétrica neste caso, substituíndo no algoritmo 2.7, obtense a variante en s-pasos do Gradente Conxugado:

Algoritmo 3.1 (s-Gradente Conxugado).
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0$
$P_0 = \Delta_A(r_0)$
Para $i = 0, 1, 2, ...$ *ata converxencia*

$$A\text{-ortonormalízanse as columnas de } P_i \tag{3.1}$$
$$y_i = P_i^t r_i \tag{3.2}$$
$$x_{i+1} = x_i + P_i y_i \tag{3.3}$$
$$r_{i+1} = r_i - AP_i y_i \tag{3.4}$$
$$Q_{i+1} = \Delta_A(r_{i+1}) \tag{3.5}$$
$$B_{i+1} = -(P_i)^t A Q_{i+1} \tag{3.6}$$
$$P_{i+1} = Q_{i+1} + P_i B_{i+1} \tag{3.7}$$

Fin

A partir dos lemas 2.1 e 2.4 obtéñense as seguintes propiedades de ortogonalidade para este algoritmo:

1. $P_i^t A P_j = 0$ para todo $i \neq j$.

2. $(P_j)^t r_i = 0$ para todo $i > j$. En particular, $< r_i, r_j >= 0$ para todo $i > j$.

3. r_{i+1} minimiza $E(r) =< r, A^{-1}r >$ sobre $x_0 + \pounds\{P_0, ..., P_i\}$.

3.2. Variante en s-pasos do Gradente Conxugado Precondicionado

A variante en s-pasos do Gradente Conxugado Precondicionado proponse en [25]. Nesta ocasión tómase $H = A^{-1}$ e K unha matriz calquera simétrica e definida positiva. Entón $N = A$ e o algoritmo 2.7 escríbese:

Algoritmo 3.2 (s-Gradente Conxugado precondicionado).
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0$
$P_0 = \Delta_{KA}(Kr_0)$
Para $i = 0, 1, 2, ...$ *ata converxencia*

$$A\text{-ortonormalízanse as columnas de } P_i \tag{3.8}$$
$$y_i = P_i^t r_i \tag{3.9}$$
$$x_{i+1} = x_i + P_i y_i \tag{3.10}$$
$$r_{i+1} = r_i - AP_i y_i \tag{3.11}$$
$$Q_{i+1} = \Delta_{KA}(Kr_{i+1}) \tag{3.12}$$
$$B_{i+1} = -(P_i)^t AQ_{i+1} \tag{3.13}$$
$$P_{i+1} = Q_{i+1} + P_i B_{i+1} \tag{3.14}$$

Fin

Neste caso os lemas 2.1 e 2.4 proporcionan as seguintes propiedades:

1. $(P_i)^t AP_j = 0$ para todo $i \neq j$.
2. $(P_j)^t r_i = 0$ para todo $i > j$. En particular, $< r_i, Kr_j >= 0$ para todo $i > j$.
3. r_{i+1} minimiza $E(r) =< r, A^{-1}r >$ sobre $x_0 + \pounds\{P_0, ..., P_i\}$.

3.3. Variante en s-pasos do Residuo Conxugado

O método do Residuo Conxugado publicouse orixinalmente en [60] como un método similar ao Gradente Conxugado. Ten maiores requerimentos computacionais máis a vantaxe sobre o Gradente Conxugado de converxer en sistemas con

matrices de coeficientes simétricas e non singulares, mais non necesariamente definidas positivas.

No mesmo artigo no que se propón o s-Gradente Conxugado [24], introdúcese a variante en s-pasos do método do Residuo Conxugado. Suponse neste caso que a matriz A é simétrica e non singular. Elíxese $H = I$ e $K = A^{-1}$, entón $N = A^2$. Substituíndo no algoritmo 2.7, obtense a variante en s-pasos do Residuo Conxugado:

Algoritmo 3.3 (s-Residuo Conxugado).
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0$
$P_0 = \Delta_A(r_0)$
Para $i = 0, 1, 2, \ldots$ *ata converxencia*

$$A^2\text{-ortonormalízanse as columnas de } P_i \tag{3.15}$$
$$y_i = (AP_i)^t r_i \tag{3.16}$$
$$x_{i+1} = x_i + P_i y_i \tag{3.17}$$
$$r_{i+1} = r_i - AP_i y_i \tag{3.18}$$
$$Q_{i+1} = \Delta_A(r_{i+1}) \tag{3.19}$$
$$B_{i+1} = -(AP_i)^t A Q_{i+1} \tag{3.20}$$
$$P_{i+1} = Q_{i+1} + P_i B_{i+1} \tag{3.21}$$

Fin

Hai que ter en conta que, posto que A é simétrica, no algoritmo 3.3 non é necesario calcular A^2 para A^2-ortonormalizar as columnas da matriz P_i. As propiedades que se obteñen agora dos lemas 2.1 e 2.4 son:

1. $(AP_i)^t(AP_j) = 0$ para todo $i \neq j$.

2. $(AP_j)^t r_i = 0$ para todo $i > j$. En particular, $< r_i, Ar_j >= 0$ para todo $i > j$.

3. r_{i+1} minimiza $E(r) =< r, r >$ sobre $x_0 + \pounds\{P_0, \ldots, P_i\}$.

3.4. Variante en s-pasos do método da Ecuación Normal

Se a matriz de coeficientes do sistema A é non singular mais non é simétrica, o método da Ecuación Normal consiste en aplicar o método do Gradente Conxugado sobre o sistema equivalente

$$A^tAx = A^tb,$$

xa que A^tA é simétrica e definida positiva. Aínda que non é necesario calcular a matriz A^tA, este método ten unha converxencia lenta cando a matriz A está mal condicionada.

No noso coñecemento, a variante en s-pasos do método da Ecuación Normal non se atopa publicado en ningún artigo, mais vén proposto por Chronopoulos en [22]. Neste método A é unha matriz non singular calquera e tómase $H = I$ e $K = I$. Entón $N = A^tA$, a matriz K segue sendo simétrica e, deste xeito, o algoritmo 2.7 escríbese:

Algoritmo 3.4 (s-Ecuación Normal).
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0$
$g_0 = A^tr_0$
$P_0 = \Delta_{A^tA}(g_0)$
Para $i = 0, 1, 2, ...$ *ata converxencia*

$$A^tA\text{-}ortonormalízanse\ as\ columnas\ de\ P_i \tag{3.22}$$
$$y_i = P_i^tg_i \tag{3.23}$$
$$x_{i+1} = x_i + P_iy_i \tag{3.24}$$
$$r_{i+1} = r_i - AP_iy_i \tag{3.25}$$
$$g_{i+1} = A^tr_{i+1} \tag{3.26}$$
$$Q_{i+1} = \Delta_{A^tA}(g_{i+1}) \tag{3.27}$$
$$B_{i+1} = -(AP_i)^tAQ_{i+1} \tag{3.28}$$
$$P_{i+1} = Q_{i+1} + P_iB_{i+1} \tag{3.29}$$

Fin

Do mesmo xeito que no caso do s-Residuo Conxugado, non é necesario realizar o cálculo de A^tA para A^tA-ortonormalizar as columnas da matriz P_i. As propiedades consecuencia dos lemas 2.1 e 2.4 son:

1. $(AP_i)^t(AP_j) = 0$ para todo $i \neq j$.
2. $(AP_j)^t r_i = 0$ para todo $i > j$. En particular, $< Ar_i, Ar_j >= 0$ para todo $i > j$.
3. r_{i+1} minimiza $E(r) =< r, r >$ sobre $x_0 + \pounds\{P_0, \dots, P_i\}$.

3.5. Variante en s-pasos do método do Erro Minimal

O método do Erro Minimal foi proposto por King en [53] como unha modificación do Gradente Conxugado que busca minimizar a norma do erro en lugar da norma do residuo e que converxe en sistemas con matrices de coeficientes non singulares e non necesariamente simétricas. En [3] o autor e directores deste traballo propoñen unha variante en s-pasos deste método. A diferenza das variantes publicadas por Chronopoulos e outros autores ata o momento, e exceptuando a variante do método da Ecuación Normal, este método supón un algoritmo converxente para calquera matriz A non singular.

Sexa polo tanto A unha matriz cadrada e non singular. Tómase $H = (AA^t)^{-1}$ e $K = A^tA$. Entón K é simétrica e $N = I$.

Para obter entón a variante en s-pasos do método do Erro Minimal, substitúense as matrices anteriores no algoritmo 2.7, resultando entre os cálculos a realizar en cada iteración os seguintes

$$g_i = A^{-1} r_i \tag{3.30}$$

e

$$y_i = V_i^t g_i. \tag{3.31}$$

Neste caso faise necesario para o cálculo do vector y_i o do vector g_i, e para este último é imprescincible calcular a inversa de A. Isto último faría innecesaria a aplicación do algoritmo, xa que a solución do sistema viría dada polo produto $A^{-1}b$.

Para solucionar esta dependencia da matriz A^{-1} haberá que facer algunhas pequenas modificacións que evitarán a necesidade do cálculo de g_i. Estas modificacións faranse sobre a versión orixinal (non N-ortonormalizada) do s-AXO, o algoritmo 2.2, na que se substitúen as matrices H, K e N definidas, e obtendo unha primeira variante en s-pasos do Erro Minimal. Posteriormente, obterase unha segunda versión desta variante na que se N-ortonormalizan as columnas da matriz P_i. Deste xeito, substituíndo as matrices H, K e N antes citadas no algoritmo 2.2, obtense o seguinte algoritmo:

Algoritmo 3.5 .
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0$
$g_0 = A^{-1}r_0$
$P_0 = \Delta_{A^tA}(A^tr_0) = Q_0$
Para $i = 0, 1, 2, ...$ *ata converxencia*

$$W_i = (P_i)^tP_i \tag{3.32}$$
$$z_i = (P_i)^tg_i \tag{3.33}$$
$$y_i = (W_i)^{-1}z_i \tag{3.34}$$
$$x_{i+1} = x_i + P_iy_i \tag{3.35}$$
$$g_{i+1} = A^{-1}r_{i+1} \tag{3.36}$$
$$Q_{i+1} = \Delta_{A^tA}(A^tr_{i+1}) \tag{3.37}$$
$$B_{i+1} = -W_i^{-1}(P_i)^tQ_{i+1} \tag{3.38}$$
$$P_{i+1} = Q_{i+1} + P_iB_{i+1} \tag{3.39}$$

Fin

Para evitar a necesidade do cálculo de g_i en cada iteración úsase a definición de P_i e o apartado ***b)*** do lema 2.1 para escribir, para cada $i = 0, 1, 2, ...$,

$$z_i = (P_i)^tg_i = (Q_i)^tA^{-1}r_i = ((A^{-1})^t(Q_i)^t)^tr_i, \tag{3.40}$$

mais, pola definición de Q_i

$$(A^{-1})^t(Q_i)^t = (\Delta_{AA^t}(r_i)). \tag{3.41}$$

Deste xeito, se se define a matriz

$$R_i = \Delta_{AA^t}(r_i), \tag{3.42}$$

tense que

$$Q_i = A^t R_i \tag{3.43}$$

e

$$z_i = R_i r_i, \tag{3.44}$$

co que a variante en s-pasos do algoritmo do Erro Minimal queda:

Algoritmo 3.6 (s-Erro Minimal, primeira versión).
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0$
$R_0 = \Delta_{AA^t}(r_0)$
$P_0 = A^t R_0$
Para $i = 0, 1, 2, ...$ *ata converxencia*

$$W_i = (P_i)^t P_i \tag{3.45}$$
$$z_i = (R_i)^t r_i \tag{3.46}$$
$$y_i = (W_i)^{-1} z_i \tag{3.47}$$
$$x_{i+1} = x_i + P_i y_i \tag{3.48}$$
$$r_{i+1} = r_i - AP_i y_i \tag{3.49}$$
$$R_{i+1} = \Delta_{AA^t}(r_{i+1}) \tag{3.50}$$
$$B_{i+1} = -W_i^{-1}(AP_i)^t R_{i+1} \tag{3.51}$$
$$P_{i+1} = A^t R_{i+1} + P_i B_{i+1} \tag{3.52}$$

Fin

Coa intención de ortonormalizar as columnas da matriz P_i en cada iteración do s-Erro Minimal, para cada $i = 0, 1, 2, ...$ defínense as seguintes matrices:

$$Q_0 = R_0, \tag{3.53}$$

$$Q_{i+1} = R_{i+1} + Q_i B_i. \tag{3.54}$$

É obvio que, se $i = 0, 1, 2, ...$,

$$P_{i+1} = A^t Q_{i+1} \tag{3.55}$$

e, pola propiedade ***b)*** do lema 2.1, para $i = 1, 2, ...$,

$$(Q_{i-1})^t r_i = ((A^{-1})^t P_{i-1})^t r_i = (P_{i-1})^t A^{-1} r_i = 0 \tag{3.56}$$

polo que, para $i = 1, 2, ...,$

$$z_i = (R_i)^t r_i = (Q_i)^t r_i - (B_i)^t (Q_{i-1})^t r_i = (Q_i)^t r_i \tag{3.57}$$

e, obviamente

$$z_0 = (R_0)^t r_0 = (Q_0)^t r_0. \tag{3.58}$$

Deste xeito, pódese reescribir a variante en s-pasos do algoritmo do Erro Minimal introducindo en cada iteración a AA^t-ortonormalización das columnas da matriz Q_i, o que fai que as columnas da matriz P_i sexan ortonormais, e polo tanto $W_i = I$. O algoritmo resultante é:

Algoritmo 3.7 (s-Erro Minimal, segunda versión).
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0$
$Q_0 = R_0 = \Delta_{AA^t}(r_0)$
Para $i = 0, 1, 2, ...$ *ata convergencia*

$$AA^t\text{-ortonormalízanse as columnas de } Q_i \tag{3.59}$$
$$P_i = A^t Q_i \tag{3.60}$$
$$y_i = (Q_i)^t r_i \tag{3.61}$$
$$x_{i+1} = x_i + P_i y_i \tag{3.62}$$
$$r_{i+1} = r_i - AP_i y_i \tag{3.63}$$
$$R_{i+1} = \Delta_{AA^t}(r_{i+1}) \tag{3.64}$$
$$B_{i+1} = -(AP_i)^t R_{i+1} \tag{3.65}$$
$$Q_{i+1} = R_{i+1} + Q_i B_{i+1} \tag{3.66}$$

Fin

As propiedades de ortogonalidade consecuencia dos lemas 2.1 e 2.4 son neste caso:

1. $(P_i)^t (P_j) = 0$ para todo $i \neq j$.
2. $(P_j)^t A^{-1} r_i = 0$ para todo $i > j$. En particular, $< r_i, r_j >= 0$ para todo $i > j$.
3. r_{i+1} minimiza $E(r) =< A^{-1}r, A^{-1}r >=< x-c, x-c >$ sobre $x_0 + \pounds\{P_0, ..., P_i\}$, onde $c \in \mathbb{R}^n$ é a solución exacta do sistema.

3.6. Variante en s-pasos do Residuo Conxugado Xeneralizado

Escollendo $H = I$ e $K = (A^{-1})^t$ obtense a variante en s-pasos do Residuo Conxugado Xeneralizado, que aparece proposto en [23]. Así, $N = A^tA$ e a matriz A non ten porque ser simétrica, mais a súa parte simétrica debe ser definida positiva. Deste xeito atopámonos co primeiro exemplo no que a matriz K non é en xeral simétrica, mais K^S é definida positiva pois, se se fai $x = Ay$, tense que

$$< x, K^S x >=< A^{-1}x, x >=< y, Ay >=< y, A^S y >\geq 0, \tag{3.67}$$

xa que A^S é definida positiva. Xa que a matriz K non é simétrica, neste caso débese substituír no algoritmo 2.6:

Algoritmo 3.8 (s-Residuo Conxugado Xeneralizado).
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0$
$g_0 = A^t r_0$
$P_0 = \Delta_A(r_0) = Q_0$
Para $i = 0, 1, 2, \ldots$ *ata converxencia*

$$A^tA\text{-}ortonormalízanse\ as\ columnas\ de\ P_i \tag{3.68}$$
$$y_i = P_i^t g_i \tag{3.69}$$
$$x_{i+1} = x_i + P_i y_i \tag{3.70}$$
$$r_{i+1} = r_i - AP_i y_i \tag{3.71}$$
$$g_{i+1} = A^t r_{i+1} \tag{3.72}$$
$$Q_{i+1} = \Delta_A(r_{i+1}) \tag{3.73}$$

Para $j = 0, \ldots, i$*:*

$$B_{i+1}^j = -(AP_j)^t AQ_{i+1} \tag{3.74}$$

$$P_{i+1} = Q_{i+1} + \sum_{j=0}^{i} P_j B_{i+1}^j \tag{3.75}$$

Fin

As propiedades de ortogonalidade derivados dos lemas 2.1 e 2.4 para este algoritmo son:

1. $(AP_i)^t(AP_j) = 0$ para todo $i \neq j$.
2. $(AP_j)^t r_i = 0$ para todo $i > j$. En particular, $< r_i, Ar_j >= 0$ para todo $i > j$.
3. r_{i+1} minimiza $E(r) =< r, r >$ sobre $x_0 + \pounds\{P_0, \ldots, P_i\}$.

3.7. Variante en s-pasos do Residuo Minimal de Axelsson

Xa se viu anteriormente que, cando a matriz K non é simétrica, como pasa na variante en s-pasos do Residuo Conxugado Xeneralizado, pode resultar problemático o feito de ter que almacenar todas as matrices P_j das iteracións anteriores para o cálculo da matriz B_{i+1}^j. O problema resólvese en parte coa variante Orthomin(m), onde m é un número enteiro que indica o número de matrices P_j inmediatamente anteriores á iteración i-ésima que se usan para o cálculo da matriz B_{i+1}^j. Se se ten en conta a versión Orthomin(1) da variante en s-pasos do Residuo Conxugado Xeneralizado, obtense a variante en s-pasos dun método coñecido, o Método do Residuo Minimal de Axelsson [6]. Esta nova variante en s-pasos aparece tamén proposta en [23]. A parte simétrica da matriz de coeficientes A^S debe seguir sendo definida positiva e $H = I$ e $K = (A^{-1})^t$, co cal $N = A^t A$:

Algoritmo 3.9 (s-Residuo Minimal de Axelsson).
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0$
$g_0 = A^t r_0$
$P_0 = \Delta_A(r_0) = Q_0$
Para $i = 0, 1, 2, \ldots$ *ata convergencia*

$$A^t A\text{-ortonormalízanse as columnas de } P_i \tag{3.76}$$
$$y_i = P_i^t g_i \tag{3.77}$$
$$x_{i+1} = x_i + P_i y_i \tag{3.78}$$
$$r_{i+1} = r_i - AP_i y_i \tag{3.79}$$
$$g_{i+1} = A^t r_{i+1} \tag{3.80}$$

$$Q_{i+1} = \Delta_A(r_{i+1}) \tag{3.81}$$

$$B_{i+1} = -(AP_i)^t AQ_{i+1} \tag{3.82}$$

$$P_{i+1} = Q_{i+1} + P_i B_{i+1} \tag{3.83}$$

Fin

Este algoritmo coincide coa variante en s-pasos do Residuo Conxugado cando a matriz A é simétrica definida positiva. Neste caso as propiedades de ortogonalización son como consecuencia do lema 2.6:

1. $(AP_i)^t(AP_{i-1}) = 0$ para todo $i \geq 1$.
2. $(AP_{i-1})^t r_i = 0$ para todo $i \geq 1$. En particular, $< r_i, Ar_{i-1} >= 0$.
3. r_{i+1} minimiza $E(r) =< r, r >$ sobre $x_i + \pounds\{P_i\}$.

Os métodos obtidos como casos particulares da variante en s-pasos do Algoritmo Xeral de Ortogonalización poden resumirse na seguinte táboa:

Algoritmo	**Condición converxencia**	H	K
s-Gradente Conxugado	A s. d. p.	A^{-1}	I
s-Gradente Conxugado Precondicionado	A s. d. p.	A^{-1}	calquera
s-Residuo Conxugado	A simétrica non singular	I	A^{-1}
s-Ecuación normal	A non singular	I	I
s-Erro Minimal	A non singular	$(AA^t)^{-1}$	A^tA
s-Residuo Conxugado Xeneralizado	A^S s. d. p.	I	$(A^{-1})^t$
s-Residuo Minimal	A^S s. d. p.	I	$(A^{-1})^t$

Resumindo, entre estes métodos o s-Erro minimal é proposto por primeira vez polos autores deste traballo. O resto dos métodos aparecen en publicacións doutros autores. Neste traballo proponse un marco teórico que xeneraliza estes métodos, que permite obtelos como casos particulares e na súa versión N-ortonormalizada. Esta versión N-ortonormalizada é tamén novedosa para todos eles excepto para o s-Residuo Conxugado Xeneralizado e o s-Orthomin(m).

3.8. Outras variantes en s-pasos

Supoñamos que a matriz A é simétrica e definida positiva. Entón pódense obter dous novos métodos. Para o primeiro escollendo $H = A^{-1}$ e $K = A$, entón $N = A$. A matriz K é simétrica definida positiva e o algoritmo é:

Algoritmo 3.10 .
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0$
$P_0 = \Delta_{A^2}(Ar_0)$
Para $i = 0, 1, 2, ...$ *ata converxencia*

$$A\text{-ortonormalízanse as columnas de } P_i \tag{3.84}$$
$$y_i = P_i^t r_i \tag{3.85}$$
$$x_{i+1} = x_i + P_i y_i \tag{3.86}$$
$$r_{i+1} = r_i - AP_i y_i \tag{3.87}$$
$$Q_{i+1} = \Delta_{A^2}(Ar_{i+1}) \tag{3.88}$$
$$B_{i+1} = -(P_i)^t AQ_{i+1} \tag{3.89}$$
$$P_{i+1} = Q_{i+1} + P_i B_{i+1} \tag{3.90}$$

Fin

As súas propiedas de ortogonalización son:

1. $(P_i)^t A(P_j) = 0$ para todo $i \neq j$.
2. $(P_j)^t r_i = 0$ para todo $i > j$. En particular, $< r_i, Ar_j >= 0$ para todo $i > j$.
3. r_{i+1} minimiza $E(r) =< r, A^{-1}r >$ sobre $x_0 + \pounds\{P_0, ..., P_i\}$.

O segundo método a considerar consiste en escoller $H = (AA^t)^{-1}$ e $K = A$, entón $N = I$. A matriz K segue sendo simétrica definida positiva e o algoritmo resultante:

Algoritmo 3.11 .
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0$
$P_0 = \Delta_A(r_0)$
Para $i = 0, 1, 2, ...$ *ata converxencia*

$$ortonormalízanse\ as\ columnas\ de\ P_i \tag{3.91}$$

$$y_i = P_i^t r_i \tag{3.92}$$

$$x_{i+1} = x_i + P_i y_i \tag{3.93}$$

$$r_{i+1} = r_i - AP_i y_i \tag{3.94}$$

$$Q_{i+1} = \Delta_A(r_{i+1}) \tag{3.95}$$

$$B_{i+1} = -(P_i)^t Q_{i+1} \tag{3.96}$$

$$P_{i+1} = Q_{i+1} + P_i B_{i+1} \tag{3.97}$$

Fin

E as súas propiedades de ortogonalización son:

1. $(P_i)^t(P_j) = 0$ para todo $i \neq j$.
2. $(P_j)^t A^{-1} r_i = 0$ para todo $i > j$. En particular, $< r_i, r_j >= 0$ para todo $i > j$.
3. r_{i+1} minimiza $E(r) =< x - c, x - c >$ sobre $x_0 + \pounds\{P_0, \ldots, P_i\}$.

Capítulo 4

Variante en s pasos da segunda forma do AXO

Considérase unha segunda formulación do Algoritmo Xeral de Ortogonalización, coñecida como segunda forma do AXO, que é equivalente ao algoritmo 1.5 xa visto no capítulo 2. A diferenza está no cálculo das direccións p_{i+1} e dos coeficientes β_{i+1}^l. A partir desta segunda forma, e coa conveniente elección das matrices H e K, obtense un novo método converxente para toda matriz de coeficientes non sigular, o método da Dobre Serie Ortogonal [5]. Neste capítulo proponse unha variante en s-pasos desta segunda forma do AXO, que analogamente se denotará por segunda forma do s-AXO, e derívase na variante en s-pasos do algoritmo da Dobre Serie Ortogonal, que aparece proposto en [2].

4.1. Segunda forma do s-AXO

As matrices H e K supóñense, de igual xeito que nos algoritmos 1.5 e 2.1, cadradas coa súa parte simétrica definida positiva e $N = A^t H^S A$. O algoritmo, que pode verse en [4] ou [50], é o seguinte:

Algoritmo 4.1 (segunda forma do AXO).
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0 = A(x - x_0)$
$g_0 = A^t H^S r_0 = A^t H^S A(x - x_0) = N(x - x_0)$
$p_0 = Kg_0$

Para $i = 0, 1, 2, ...$ *ata converxencia*

$$\alpha_i = \frac{< g_i, p_i >}{< p_i, Np_i >} \tag{4.1}$$

$$x_{i+1} = x_i + \alpha_i p_i \tag{4.2}$$

$$g_{i+1} = g_i - \alpha_i N p_i = A^t H^S r_{i+1} \tag{4.3}$$

$$\beta_{i+1}^l = -\frac{< KNg_{i+1}, Np_l >}{< p_l, Np_l >}, \quad 0 \leq l \leq i \tag{4.4}$$

$$p_{i+1} = KNg_{i+1} + \sum_{l=0}^{i} \beta_{i+1}^l p_l \tag{4.5}$$

Fin

Este algoritmo permite obter novos métodos para eleccións axeitadas de H e K, que se poden ver en [51] ou [4]. A continuación, proponse unha variante en s-pasos da segunda forma do AXO que dará lugar, mediante as eleccións axeitadas de H e K, as correspondentes variantes en s-pasos dos métodos que se obteñen da segunda forma do AXO. Como xa se indicou anteriormente, as matrices H e K supóñense cadradas coa parte simétrica definida positiva e $N = A^t H^S A$. Denótase por p_i^s o vector correspondente á última columna da matriz P_i. Entón, se $\text{grao}_{KN}(KNp_i^s) \geq s$ para todo $i = 0, 1, 2, ...$ ata a converxencia, a segunda forma do s-AXO é:

Algoritmo 4.2 (segunda forma do s-AXO).
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0$
$g_0 = A^t H^S r_0$
$P_0 = \Delta_{KN}(Kg_0) = Q_0$
Para $i = 0, 1, 2, ...$ *ata converxencia*

$$W_i = (P_i)^t N P_i \tag{4.6}$$

$$z_i = (P_i)^t g_i \tag{4.7}$$

$$y_i = (W_i)^{-1} z_i \tag{4.8}$$

$$x_{i+1} = x_i + P_i y_i \tag{4.9}$$

$$r_{i+1} = r_i - AP_i y_i \tag{4.10}$$

$$g_{i+1} = g_i - NP_i y_i = A^t H^S r_{i+1} \tag{4.11}$$

$$Q_{i+1} = \Delta_{KN}(KNp_i^s) \tag{4.12}$$

Para $j = 0, \dots, i$*:*

$$B_{i+1}^{j} = -W_j^{-1}(P_j)^t N Q_{i+1} \tag{4.13}$$

$$P_{i+1} = Q_{i+1} + \sum_{j=0}^{i} P_j B_{i+1}^{j} \tag{4.14}$$

Fin

Pode comprobarse que a única diferenza co s-AXO orixinal, ao que se denominará a partir de agora primeira forma do s-AXO, está no xeito de calcular a matriz cos vectores xeradores do subespazo de Krylov en cada iteración, Q_{i+1}. Non obstante, os dous algoritmos, primeira e segunda forma, son equivalentes no senso de que, se se parte do mesmo vector inicial x_0, ambos os dous proporcionan os mesmos iterantes x_i en aritmética exacta. Para probar este feito, o primeiro que se comprobará é que os subespazos de Krylov que van xerando os dous algoritmos en cada iteración son coincidentes.

Lema 4.1 *Sexan P_i e $\tilde{P}_i$ as matrices calculadas nas ecuacións* (2.12) *e* (4.14) *dos algoritmos 2.1 e 4.2 respectivamente. Entón, se partimos do mesmo vector inicial x_0 nos dous algoritmos, verifícase:*

$$\pounds\{P_0, \dots, P_i\} = \pounds\{\tilde{P}_0, \dots, \tilde{P}_i\} \text{ para todo } i = 0, 1, 2, \dots \tag{4.15}$$

Demostración:

A demostración inspírase na demostración do lema análogo para a segunda forma do AXO que se pode ver en [51] ou [4]. Demóstrase por indución:

Para $i = 0$ é trivial que $P_0 = \tilde{P}_0$. Suponse agora que

$$\pounds\{P_0, \dots, P_{i-1}\} = \pounds\{\tilde{P}_0, \dots, \tilde{P}_{i-1}\} \tag{4.16}$$

para certo $i \in \mathbb{N}$. Por indución probarase que

$$\pounds\{P_0, \dots, P_i\} = \pounds\{\tilde{P}_0, \dots, \tilde{P}_i\}. \tag{4.17}$$

Para isto basta demostrar que

$$\pounds\{P_i\} \subset \pounds\{\tilde{P}_0, \dots, \tilde{P}_i\} \tag{4.18}$$

e que

$$\pounds\{\tilde{P}_i\} \subset \pounds\{P_0, \dots, P_i\}. \tag{4.19}$$

Sexa $k \in \{1, \dots, s\}$ e p_i^k o vector da columna k-ésima da matriz P_i e $b_j^{(k,l)}$ o termo da fila k e columna l da matriz B_j definida na primeira forma do s-AXO, entón:

$$p_i^k = (KN)^{k-1} K g_i + \sum_{j=0}^{i-1} \left(\sum_{l=1}^{s} b_j^{(k,l)} p_j^l \right) \tag{4.20}$$

e, pola hipótese de indución, $p_j^l \in \pounds\{\tilde{P}_0, \dots, \tilde{P}_{i-1}\}$ para $j \in \{0, \dots, i-1\}$ e $l \in \{1, \dots, s\}$.
Por outra banda, e posto que $g_i = g_{i-1} - NP_{i-1}y_i - 1$,

$$(KN)^{k-1} K g_i = (KN)^{k-1} K (g_{i-1} - NP_{i-1}y_{i-1}) \tag{4.21}$$

e como

$$(KN)^{k-1} K g_{i-1} \in \tilde{P}_{i-1} \tag{4.22}$$

e

$$(KN)^k P_{i-1} y_{i-1} \in \pounds\{\tilde{P}_0, \dots, \tilde{P}_i\} \text{ para } k \in \{1, \dots, s\} \tag{4.23}$$

tense entón que

$$(KN)^{k-1} K g_i \in \pounds\{\tilde{P}_0, \dots, \tilde{P}_i\}, \tag{4.24}$$

co que queda probado que

$$p_i^k \in \pounds\{\tilde{P}_0, \dots, \tilde{P}_i\} \text{ para } k \in \{1, \dots, s\}. \tag{4.25}$$

Dado $k \in \{1, \dots, s\}$, sexa agora $\tilde{p}_i^k$ o vector da columna k-ésima da matriz $\tilde{P}_i$ e $\tilde{b}_j^{(k,l)}$ o termo da fila k e columna l da matriz $\tilde{B}_j$ definida na segunda forma do s-AXO, entón:

$$\tilde{p}_i^k = (KN)^{k-1} KN \tilde{p}_{i-1}^s + \sum_{j=0}^{i-1} \left(\sum_{l=1}^{s} \tilde{b}_j^{(k,l)} \tilde{p}_j^l \right). \tag{4.26}$$

De novo, pola hipótese de indución, $\tilde{p}_j^l \in \pounds\{P_0, \dots, P_{i-1}\}$ para $j \in \{0, \dots, i-1\}$ e $l \in \{1, \dots, s\}$. Ademais,

$$\pounds\{P_0, \dots, P_{i-1}\} = \mathcal{K}_{is}(KN, Kg_0) \tag{4.27}$$

polo lema 2.3, polo que $\tilde{p}_{i-1}^s \in \mathcal{K}_{is}(KN, Kg_0)$. Logo

$$(KN)^{k-1} KN \tilde{p}_{i-1}^s \in \mathcal{K}_{s(i+1)}(KN, Kg_0) = \pounds\{P_0, \dots, P_i\}, \tag{4.28}$$

e así queda tamén probado que

$$\tilde{p}_i^k \in \mathcal{L}\{P_0, \ldots, P_i\} \text{ para } k \in \{1, \ldots, s\}. \tag{4.29}$$

■

As propiedades de ortogonalidade dos lemas 2.1 e 2.2 para a primeira forma do s-AXO tamén son certas para a segunda forma, que se probarían con demostracións completamente análogas ás de ditos lemas.

Propiedades 4.1 *Sexan $i, s \in \mathbb{N}$ tales que $s(i+1) \leq n$, e suponse que $g_i \neq 0$ para todo $i = 0, 1, 2, \ldots$ ata a converxencia. Se $grao_{KN}(Kg_0) \geq s(i+1)$, na segunda forma do s-AXO cúmprense:*

a) $(P_i)^t N P_j = 0$ *para todo* $i \neq j$.

b) $(P_j)^t g_i = 0$ *para todo* $i > j$.

c) $(P_i)^t g_i = (Q_i)^t g_i$ *para todo* i, j, *e polo tanto* $(Q_j)^t g_i = 0$ *para todo* $i > j$.

d) $(P_i)^t N Q_j = 0$ *para todo* $i > j$.

e) $(P_i)^t N P_i = (P_i)^t N Q_i$.

f) $(P_i)^t g_j = (P_i)^t g_0$ *para todo* $i \geq j$.

E do mesmo xeito que para a primeira forma do s-AXO, a segunda forma minimiza tamén o funcional $E(r) =< r, H^S r >$, o cal se enuncia no seguinte lema con demostración tamén completamente análoga á do lema 2.4, correspondente á primeira forma.

Lema 4.2 *Para cada $i = 0, 1, 2, \ldots$, o residuo da segunda forma do algoritmo s-AXO, r_{i+1}, minimiza o funcional*

$$E(r) =< r, H^S r > \tag{4.30}$$

sobre o subespazo afín

$$x_0 + \mathcal{L}\{P_0, \ldots, P_i\}. \tag{4.31}$$

Deste xeito, e posto que o funcional $E(r)$ é estrictamente convexo e, polo tanto, ten un único mínimo sobre o subespazo $x_0 + \mathcal{L}\{P_0, \ldots, P_i\}$, dedúcese que os iterantes x_i son os mesmos en aritmética exacta na primeira e segunda forma do s-AXO.

Se a matriz K é simétrica, na segunda forma do s-AXO estase ante unha situación semellante á primeira forma do s-AXO cando a matriz K é tamén simétrica. Na

primeira forma do s-AXO anúlanse todas as matrices B_{i+i}^j excepto a última, mentras que na segunda forma, cando K é simétrica, anúlanse todas as matrices B_{i+i}^j excepto as dúas últimas. O lema que demostra esta afirmación é o seguinte:

Lema 4.3 *Nas mesmas hipóteses que o lema 4.1, se a matriz K é simétrica na segunda forma do s-AXO entón, para todo $j < i-1$, cúmprese que*

$$B_{i+1}^j = 0. \tag{4.32}$$

Demostración:

Inspírase na demostración do lema análogo para a segunda forma do AXO que se pode ver en [51] ou [4]. O termo na fila l e columna k da matriz B_{i+1}^j é

$$< p_j^l, N(KN)^k p_i^s > . \tag{4.33}$$

Posto que as matrices N e K son simétricas, tense que

$$< p_j^l, N(KN)^k p_i^s > = < (KN)^k p_j^l, N p_i^s > \tag{4.34}$$

e como $k \in \{1, \dots, s\}$,

$$(KN)^k p_j^l \in \pounds\{\tilde{P}_0, \dots, \tilde{P}_{j+1}\}. \tag{4.35}$$

Así que, se $j+1 < i$, pola propiedade ***a)*** anterior, tense que

$$< (KN)^k p_j^l, N p_i^s > = 0 \tag{4.36}$$

e, deste xeito, se $j < i-1$, a matriz B_{i+1}^j ten todos os seus termos nulos. ■

Convimos que $B_1^{-1} = 0$. Así pois a segunda forma do s-AXO para K simétrica quedaría:

Algoritmo 4.3 (segunda forma do s-AXO, K simétrica).
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0$
$g_0 = A^t H^S r_0$
$P_0 = \Delta_{KN}(Kg_0) = Q_0$
Para $i = 0, 1, 2, \dots$ *ata converxencia*

$$W_i = (P_i)^t N P_i \tag{4.37}$$

$$z_i = (P_i)^t g_i \tag{4.38}$$

$$y_i = (W_i)^{-1} z_i \tag{4.39}$$

$$x_{i+1} = x_i + P_i y_i \tag{4.40}$$

$$r_{i+1} = r_i - A P_i y_i \tag{4.41}$$

$$g_{i+1} = g_i - N P_i y_i = A^t H^S r_{i+1} \tag{4.42}$$

$$Q_{i+1} = \Delta_{KN}(K N p_i^s) \tag{4.43}$$

$$Se\ i = 0, B_1^{-1} = 0 \tag{4.44}$$

$$B_{i+1}^{i-1} = -W_{i-1}^{-1} (P_{i-1})^t N Q_{i+1} \tag{4.45}$$

$$B_{i+1}^{i} = -W_{i}^{-1} (P_{i})^t N Q_{i+1} \tag{4.46}$$

$$P_{i+1} = Q_{i+1} + P_{i-1} B_{i+1}^{i-1} + P_i B_{i+1}^{i} \tag{4.47}$$

Fin

Coa segunda forma do s-AXO pódense obter novas variantes en s-pasos de métodos non publicadas, no noso coñecemento, por outros autores ata o momento. En concreto, proponse unha segunda forma do s-Erro Minimal e a variante en s-pasos do Algoritmo da Dobre Serie Ortogonal, este último proposto polo autor e directores deste traballo en [2].

4.2. Variante en s-pasos do Algoritmo da Dobre Serie Ortogonal

O algoritmo do Erro Minimal pódese reescribir a partir da segunda forma do AXO. Basta escoller na segunda forma do AXO as mesmas matrices que foron escollidas para obter o Erro Minimal a partir do AXO [4, 51]. Definindo unha sucesión axeitada de vectores ortogonais, pode construírse un algoritmo equivalente coñecido como o algoritmo da Dobre serie Ortogonal, proposto por Amara e Nedelec en [5].

Na mesma liña, nesta sección reescríbese a variante en s-pasos do Erro Minimal a partir da segunda forma do s-AXO. A partir desta primeira versión faranse algunhas modificacións obtendo como resultado unha nova versión que se chamará a variante en s-pasos do algoritmo da Dobre Serie Ortogonal.

Deste xeito, escóllese $H = (AA^t)^{-1}$ e $K = A^t A$, polo que $N = I$ e K é simétrica. E substituíndo na segunda forma do s-AXO o algoritmo 4.3 queda como segue:

Algoritmo 4.4 .

Sexa $x_0 \in \mathbb{R}^n$

$r_0 = b - Ax_0$

$g_0 = A^{-1}r_0$

$P_0 = \Delta_{A^tA}(A^tr_0)$

Para $i = 0, 1, 2, ...$ *ata converxencia*

$$W_i = (P_i)^t P_i \tag{4.48}$$

$$z_i = (P_i)^t g_i \tag{4.49}$$

$$y_i = (W_i)^{-1} z_i \tag{4.50}$$

$$x_{i+1} = x_i + P_i y_i \tag{4.51}$$

$$r_{i+1} = r_i - AP_i y_i \tag{4.52}$$

$$g_{i+1} = A^{-1} r_{i+1} \tag{4.53}$$

$$Q_{i+1} = \Delta_{A^tA}(A^t A p_i^s) \tag{4.54}$$

$$Se\ i = 0, B_1^{-1} = 0 \tag{4.55}$$

$$B_{i+1}^{i-1} = -W_{i-1}^{-1}(P_{i-1})^t Q_{i+1} \tag{4.56}$$

$$B_{i+1}^{i} = -W_{i}^{-1}(P_{i})^t Q_{i+1} \tag{4.57}$$

$$P_{i+1} = Q_{i+1} + P_{i-1}B_{i+1}^{i-1} + P_i B_{i+1}^i \tag{4.58}$$

Fin

Faise necesario evitar o cómputo de g_i, pois aparece en función da inversa da matriz A. Para iso, usando a definición de P_i e g_i e o apartado ***b)*** das propiedades 4.1, tense que

$$z_0 = (P_0)^t g_0 = ((A^{-1})^t (P_0)^t)^t r_0 \tag{4.59}$$

e, se $i = 1, 2, ...$

$$z_i = (P_i)^t g_i = (Q_i)^t A^{-1} r_i = ((A^{-1})^t (Q_i)^t)^t r_i \tag{4.60}$$

e, tendo en conta que

$$(A^{-1})^t (P_0)^t = (\Delta_{AA^t}(r_0)) \tag{4.61}$$

e que, para $i = 1, 2, ...$

$$(A^{-1})^t (Q_i)^t = (\Delta_{AA^t}(Ap_i^s)), \tag{4.62}$$

pódense definir as matrices

$$R_0 = \Delta_{AA^t}(r_0) \tag{4.63}$$

e, para $i = 1, 2, ...$

$$R_i = \Delta_{AA^t}(Ap_i^s). \tag{4.64}$$

Deste xeito, para $i = 1, 2, ...$

$$Q_i = A^t R_i \tag{4.65}$$

e, para $i = 0, 1, 2, ...$

$$z_i = R_i r_i. \tag{4.66}$$

Agora pódese escribir unha nova versión do algoritmo 4.4, que é a variante en s-pasos do algoritmo da Dobre Serie Ortogonal (s-DSO):

Algoritmo 4.5 (s-DSO).
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0$
$R_0 = \Delta_{AA^t}(r_0)$
$P_0 = A^t R_0$
Para $i = 0, 1, 2, ...$ *ata converxencia*

$$W_i = (P_i)^t P_i \tag{4.67}$$
$$z_i = (R_i)^t r_i \tag{4.68}$$
$$y_i = (W_i)^{-1} z_i \tag{4.69}$$
$$x_{i+1} = x_i + P_i y_i \tag{4.70}$$
$$r_{i+1} = r_i - AP_i y_i \tag{4.71}$$
$$R_{i+1} = \Delta_{AA^t}(Ap_i^s) \tag{4.72}$$
$$Q_{i+1} = A^t R_{i+1} \tag{4.73}$$
$$Se\ i = 0, B_1^{-1} = 0 \tag{4.74}$$
$$B_{i+1}^{i-1} = -W_{i-1}^{-1}(P_{i-1})^t Q_{i+1} \tag{4.75}$$
$$B_{i+1}^{i} = -W_{i}^{-1}(P_{i})^t Q_{i+1} \tag{4.76}$$
$$P_{i+1} = Q_{i+1} + P_{i-1}B_{i+1}^{i-1} + P_i B_{i+1}^i \tag{4.77}$$

Fin

De xeito análogo ao que se fixo coa variante en s-pasos do algoritmo do Erro Minimal, é necesario definir en cada iteración unha matriz intermedia para poder introducir a ortonormalización das columnas da matriz P_i. Neste caso o que se vai facer é redefinir as matrices Q_i, de xeito que

$$Q_0 = R_0 \tag{4.78}$$

e, para cada $i = 0, 1, 2, ...$

$$Q_{i+1} = R_{i+1} + Q_{i-1}B_{i+1}^{i-1} + Q_iB_{i+1}^{i}, \tag{4.79}$$

tendo en conta que se $i = 0$, entón $B_1^{-1} = 0$ e o segundo sumando do segundo membro na ecuación (4.79) é nulo, co que non se fai necesario definir Q_{-1}.

A partir da definición da nova matriz Q_i, pódese comprobar sen dificultede por indución que, se $i = 0, 1, 2, ...,$

$$P_{i+1} = A^tQ_{i+1}. \tag{4.80}$$

Ademais, polo apartado ***b)*** das propiedades 4.1 para a segunda forma do s-AXO, se $i = 1, 2, ...,$

$$(Q_{i-1})^tr_{i+1} = ((A^{-1})^tP_{i-1})^tr_{i+1} = (P_{i-1})^tA^{-1}r_{i+1} = 0 \tag{4.81}$$

e, analogamente

$$(Q_i)^tr_{i+1} = 0. \tag{4.82}$$

Polo que

$$z_0 = (Q_0)^tr_0 \tag{4.83}$$

e, se $i = 1, 2, ...,$

$$z_i = (R_i)^tr_i = (Q_i)^tr_i - (B_i^{i-2})^t(Q_{i-2})^tr_i - (B_i^{i-1})^t(Q_{i-1})^tr_i = (Q_i)^tr_i, \tag{4.84}$$

tendo en conta de novo que se $i = 1$, entón $B_1^{-1} = 0$.

Deste xeito, pódese escribir unha nova versión ortonormalizada do s-DSO introducindo en cada iteración a AA^t-ortonormalización das columnas da matriz Q_i, o que fai que as columnas da matriz P_i sexan ortonormais, e polo tanto $W_i = I$. Así, $y_i = z_i = (Q_i)^tr_i$ e o algoritmo resultante é:

Algoritmo 4.6 (s-DSO ortonormalizado).
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0$
$Q_0 = R_0 = \Delta_{AA^t}(r_0)$
Para $i = 0, 1, 2, ...$ *ata convergencia*

$$AA^t - \textit{ortonormalízanse as columnas de } Q_i \tag{4.85}$$

$$P_i = A^t Q_i \tag{4.86}$$

$$y_i = (Q_i)^t r_i \tag{4.87}$$

$$x_{i+1} = x_i + P_i y_i \tag{4.88}$$

$$r_{i+1} = r_i - A P_i y_i \tag{4.89}$$

$$R_{i+1} = \Delta_{AA^t}(A p_i^s) \tag{4.90}$$

$$Se\ i = 0 : B_1^{-1} = 0 \tag{4.91}$$

$$B_{i+1}^{i-1} = -(AP_{i-1})^t R_{i+1} \tag{4.92}$$

$$B_{i+1}^{i} = -(AP_{i})^t R_{i+1} \tag{4.93}$$

$$Q_{i+1} = R_{i+1} + Q_{i-1} B_{i+1}^{i-1} + Q_i B_{i+1}^{i} \tag{4.94}$$

Fin

Capítulo 5

Variante en s-pasos do BiCG. Métodos derivados.

O método do Gradente Biconxugado [39, 72, 79] (BiCG) é un método iterativo para a resolución de sistemas nos que a matriz non é simétrica. Implicitamente, o algoritmo non só resolve o sistema orixinal $Ax = b$, senón que tamén resolve un sistema $A^t x^* = b^*$ coa matriz A^t. A idea da construción do BiCG consiste en simetrizar o sistema construíndo un novo sistema de orde $2n$, obtendo dúas sucesións de vectores tipo Gradente Conxugado, unha baseada no sistema orixinal coa matriz A e outra coa matriz A^t. O algoritmo é o seguinte:

Algoritmo 5.1 BiCG).
Sexa $x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0$
$r_0^* \in \mathbb{R}^n$ *tal que* $< r_0, r_0^* > \neq 0$
$p_0 = r_0$
$p_0^* = r_0^*$
Para $i = 0, 1, 2, ...$ *ata converxencia*

$$\alpha_i = \frac{< r_i, r_i^* >}{< Ap_i, p_i^* >}$$

$$x_{i+1} = x_i + \alpha_i p_i$$

$$r_{i+1} = r_i - \alpha_i A p_i$$

$$r_{i+1}^* = r_i^* - \alpha_i A^t p_i^*$$

$$\beta_i = \frac{< r_{i+1}, r_{i+1}^* >}{< r_i, r_i^* >}$$

$$p_{i+1} = r_{i+1} + \beta_i p_i$$

$$p^*_{i+1} = r^*_{i+1} + \beta_i p^*_i$$

Fin

O método do Gradente Biconxugado pode dexenerar, xa que non se pode garantir a non nulidade dos produtos escalares $< r_i, r^*_i >$ e $< Ap_i, p^*_i >$ para todo i. Non obstante, se non dexenera, pódese comprobar que converxe para calquera matriz cadrada non singular [39, 4].

Do Gradente Biconxugado derívanse os métodos Gradente Conxugado Cadrado [72, 76, 79] (GCC) e o Gradente Biconxugado Estabilizado [72, 78, 79] (BiCGSTAB). Estes dous últimos métodos conseguen evitar o uso da matriz A^t á vez que presentan una converxencia máis rápida con practicamente o mesmo coste computacional. A idea consiste en observar que, no BiCG, os residuos e vectores dirección poden ser expresados como

$$r_i = \phi_i(A)r_0, \quad r^*_i = \phi_i(A^t)r^*_0 \tag{5.1}$$

e

$$p_i = \pi_i(A)p_0, \quad p^*_i = \pi_i(A^t)p^*_0, \tag{5.2}$$

onde $\phi_i(t)$ e $\pi_i(t)$ son polinomios en t de grao i, coa condición $\phi_i(0) = 1$. A partir de (5.1) pódese escribir

$$< r_i, r^*_i > = < \phi_i(A)r_0, \phi_i(A^t)r^*_0 > = < \phi^2_i(A)r_0, r^*_0 > \tag{5.3}$$

e, de (5.2),

$$< Ap_i, p^*_i > = < A\pi_i(A)p_0, \pi_i(A^t)p^*_0 > = < A\pi^2_i(A)p_0, p^*_0 > . \tag{5.4}$$

Deste xeito, o Gradente Conxugado Cadrado constrúese obtendo fórmulas de recurrencia para $\phi_i(A)^2 r_0$ e $\pi_i(A)^2 p_0$, e definindo uns novos residuos e vectores dirección como

$$\tilde{r}_i = \phi_i(A)^2 r_0 \tag{5.5}$$

e

$$\tilde{p}_i = \pi_i(A)^2 p_0 \tag{5.6}$$

A partir de aquí é inmediato expresar os escalares α_i e β_i en función de $\tilde{r}_i$ e $\tilde{p}_i$. Non obstante o GCC ten o problema de que, cando a converxencia non é moi regular, pode haber problemas de estabilidade numérica.

O BiCGSTAB é unha variante do GCC que foi desenvolta para evitar o problema da estabilidade numérica. A diferenza do GCC, para construír o BiCGSTAB, defínense os residuos e os vectores dirección

$$\tilde{r}_i = \phi_i(A)\psi_i(A)r_0 \tag{5.7}$$

e

$$\tilde{p}_i = \psi_i(A)\pi_i(A)p_0, \tag{5.8}$$

onde $\phi_i(t)$ e $\pi_i(t)$ son os mesmos polinomios das ecuacións (5.1) e (5.2), e $\psi_i(t)$ é outro polinomio en t de grao i definido pola recurrencia

$$\psi_{i+1}(t) = (1 - \omega_i t)\psi_i(t). \tag{5.9}$$

O escalar ω determínase de xeito que se minimice a norma do vector

$$\tilde{r}_{i+1} = (1 - \omega_i A)\psi_i(A)\phi_{i+1}(A)r_0. \tag{5.10}$$

Neste capítulo proponse unha variante en s-pasos do Gradente Biconxugado (s-BiCG), que se construirá seguindo técnicas análogas á construción do Gradente Biconxugado orixinal.

No caso dos métodos Gradente Conxugado Cadrado e BiCGSTAB non é posible obter as correspondentes variantes en s-pasos coas mesmas técnicas que se usan para obter o GCC e o BiCGSTAB orixinais. O problema é que, se ben os residuos do s-BiCG poden expresarse como polinomios en A polo residuo inicial, non ocorre o mesmo coas matrices P_i, que conteñen os vectores dirección nas súas columnas. Isto é debido á non conmutatividade do produto matricial que impide, na ecuación que definirá as matrices P_i no s-BiCG, o desenvolvemento de P_i como un polinomio en A pola matriz inicial Q_0.

5.1. Variante en s-pasos do BiCG

Nesta sección obtense a partir do s-AXO a variante en s-pasos do algoritmo do Gradente Biconxugado (s-BiCG), que xa foi proposta en [3]. Para obter a variante en s-pasos do BiCG (s-BiCG) usarase unha técnica análoga á utilizada para obter o BiCG orixinal.

As hipóteses sobre as matrices H e K non serán certas neste caso (a súa parte simétrica non ten porque ser definida positiva), polo que a converxencia debería obterse con outros argumentos distintos aos que se usaron para obter a do s-AXO. Non obstante é de esperar que o s-BiCG se comporte de xeito similar ao BiCG en canto á converxencia.

Dada A unha matriz regular de orde $n \times n$, defínense entón as seguintes matrices:

$$\mathbb{A} = \begin{pmatrix} 0 & A \\ A^t & 0 \end{pmatrix}, \; X = \begin{pmatrix} x^* \\ x \end{pmatrix}, \; B = \begin{pmatrix} b \\ b^* \end{pmatrix}. \tag{5.11}$$

E sexan

$$\mathbb{H} = (\mathbb{A}^{-1})^t = \begin{pmatrix} 0 & (A^t)^{-1} \\ A^{-1} & 0 \end{pmatrix} \text{ e } \mathbb{K} = \begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix}, \tag{5.12}$$

entón $\mathbb{N} = \mathbb{A}^t\mathbb{H}\mathbb{A} = \mathbb{A}$ é simétrica mais non é definida positiva.

Deste xeito pódese obter o s-BiCG escribindo o algoritmo s-AXO na súa versión para $\mathbb{K}$ simétrica:

Algoritmo 5.2 (s-BiCG).
Sexa $X_0 \in \mathbb{R}^{2n}$
$R_0 = B - \mathbb{A}X_0$
$\mathbb{P}_0 = \mathbb{Q}_0 = \Delta_{\mathbb{K}\mathbb{A}}(\mathbb{K}R_0)$
Para $i = 0, 1, 2, \ldots$ *ata converxencia*

$$W_i = \mathbb{P}_i^t \mathbb{A} \mathbb{P}_i \tag{5.13}$$

Se W_i *singular entón o algoritmo para* (5.14)

$$y_i = W_i^{-1} \mathbb{P}_i^t R_i \tag{5.15}$$

$$X_{i+1} = X_i + \mathbb{P}_i y_i \tag{5.16}$$

$$R_{i+1} = R_i - \mathbb{A}\mathbb{P}_i y_i \tag{5.17}$$

$$\mathbb{Q}_{i+1} = \Delta_{\mathbb{K}\mathbb{A}}(\mathbb{K}R_{i+1}) \tag{5.18}$$

$$B_{i+1} = -(W_i)^{-1} \mathbb{P}_i^t \mathbb{A} \mathbb{Q}_{i+1} \tag{5.19}$$

$$\mathbb{P}_{i+1} = \mathbb{Q}_{i+1} + \mathbb{P}_i B_{i+1} \tag{5.20}$$

Fin

Dado que $\mathbb{A}$ non é definida positiva, a matriz W_i podería resultar singular para algunha iteración i. Neste caso non tería lugar o cálculo de W^{-1} e o algoritmo para.

É por isto que é necesario introducir no algoritmo a liña (5.14). Se o algoritmo chega a parar por este motivo, dise que dexenera. No que segue suporase que o algoritmo non dexenera antes da súa converxencia.

Neste algoritmo non é necesario o cálculo do residuo xeneralizado, xa que, se se denota por G_i, viría definido pola seguinte ecuación:

$$G_i = \mathbb{A}^t \mathbb{H}^S R_i = R_i. \tag{5.21}$$

Coa intención de poder escribir o s-BiCG en termos de vectores n dimensionais, denótanse as matrices $\mathbb{P}_i$, $\mathbb{Q}_i$ e R_i do seguinte xeito:

$$\mathbb{P}_i = \begin{pmatrix} P_i^* \\ P_i \end{pmatrix}, \ \mathbb{Q}_i = \begin{pmatrix} Q_i^* \\ Q_i \end{pmatrix} \text{ e } \ R_i = \begin{pmatrix} r_i \\ r_i^* \end{pmatrix} \tag{5.22}$$

e enúnciase o seguinte lema:

Lema 5.1 *No s-BiCG tense para todo $i, k \in \{0, 1, 2, ...\}$ que:*

a) $(Q_i^*)^t A^k r_i = Q_i^t \left(A^t\right)^k r_i^*$.

b) $(Q_i^*)^t A^k Q_i = Q_i^t \left(A^t\right)^k Q_i^*$.

Demostración:

Da definición de $\mathbb{Q}_i$ en (5.18) e a súa expresión en (5.22) tense que

$$Q_i^* = \Delta_{A^t}(r_i^*) \text{ e } Q_i = \Delta_A(r_i). \tag{5.23}$$

Entón os apartados ***a)*** e ***b*** do lema son obvios a partir do feito de que para cada $i, k \in \{0, 1, 2, ...\}$,

$$(r_i^*)^t A^k r_i = r_i^t \left(A^t\right)^k r_i^*. \tag{5.24}$$

■

O seguinte lema pode enunciarse como consecuencia dos lemas 2.1 e 5.1:

Lema 5.2 *No s-BiCG cúmplese para todo $i, k \in \{0, 1, 2, ...\}$ que:*

a) $(P_i^*)^t A^k P_i = P_i^t \left(A^t\right)^k P_i^*$

b) $(P_i^*)^t A^k r_{i+1} = P_i^t \left(A^t\right)^k r_{i+1}^*$

c) $(P_i^*)^t r_{i+1} = P_i^t r_{i+1}^* = 0$

d) $(P_i^*)^t r_i = P_i^t r_i^* = (Q_i^*)^t r_i = Q_i^t r_i^*$.

Demostración:

Os apartados ***a)*** e ***b)*** demóstranse por indución sobre i:

Para $i = 0$ os apartados ***a)*** e ***b)*** son certos polo lema 5.1 posto que $\mathbb{P}_0 = \mathbb{Q}_0$. Suponse agora que ***a)*** e ***b)*** son certos para $i-1$ con $i \geq 1$, entón, tendo en conta a definición de $\mathbb{P}_i$ en (5.20) e a súa expresión en (5.22),

$$P_i^* = Q_i^* + P_{i-1}^* B_i \text{ e } P_i = Q_i + P_{i-1} B_i, \tag{5.25}$$

e entón

$$\begin{gathered}(P_i^*)^t A^k P_i = \\ = (Q_i^*)^t A^k Q_i + (Q_i^*)^t A^k P_{i-1} B_i + B_i^t (P_{i-1}^*)^t A^k Q_i + B_i^t (P_{i-1}^*)^t A^k P_{i-1} B_i\end{gathered} \tag{5.26}$$

e

$$\begin{gathered}P_i^t \left(A^t\right)^k P_i^* = \\ = Q_i^t (A^t)^k Q_i^* + Q_i^t (A^t)^k P_{i-1}^* B_i + B_i^t P_{i-1}^t (A^t)^k Q_i^* + B_i^t P_{i-1}^t (A^t)^k P_{i-1}^* B_i.\end{gathered} \tag{5.27}$$

Os primeiros sumandos dos segundos membros das ecuacións (5.26) e (5.27) son iguais como consecuencia do lema 5.1. O resto de sumandos tamén o son pola hipótese de indución en ***a)*** e ***b)***. Isto demostra o apartado ***a)*** do lema.

Por outra banda, da ecuación (5.17) e a expresión de $\mathbb{R}_i$ en (5.22), tense que

$$r_{i+1} = r_i - A P_i y_i \tag{5.28}$$

e

$$r_{i+1}^* = r_i^* - A^t P_i^* y_i. \tag{5.29}$$

Deste xeito,

$$(P_i^*)^t A^k r_{i+1} = (P_i^*)^t A^k r_i - (P_i^*)^t A^{k+1} P_i y_i \tag{5.30}$$

e

$$P_i^t \left(A^t\right)^k r_{i+1}^* = P_i^t \left(A^t\right)^k r_i^* - P_i^t \left(A^t\right)^{k+1} (P_i)^* y_i. \tag{5.31}$$

Os segundos sumandos dos segundos membros das ecuacións (5.30) e (5.31) son iguais como consecuencia do apartado ***a)*** que se acaba de demostrar. Tendo tamén en conta as expresións de P_i e P_i^* da ecuación (5.25), a igualdade

$$(P_i^*)^t A^k r_i = P_i^t \left(A^t\right)^k r_i^* \tag{5.32}$$

obtense polo lema 5.1 e a hipótese de indución. Deste xeito os primeiros membros das ecuacións (5.30) e (5.31) son equivalentes e o apartado ***b)*** queda demostrado.

O apartado ***c)*** demóstrase polo apartado ***b)*** deste lema, para $k = 0$ e o ***b)*** do lema 2.1, tendo en conta que $\mathbb{H} = (\mathbb{A}^{-1})^t$ e entón $g_i = \mathbb{A}^t\mathbb{H}^S r_i = r_i$.

Por último, da definición de $\mathbb{P}_{i+1}$ en (5.20) e, polo tanto, das ecuacións en (5.25), tense que

$$(P_i^*)^t r_i = (Q_i^*)^t r_i + (B_i)^t (P_{i-1}^*)^t r_i \tag{5.33}$$

e

$$(P_i)^t r_i = (Q_i)^t r_i + (B_i)^t (P_{i-1})^t r_i, \tag{5.34}$$

e o apartado ***d)*** é certo entón polo apartado ***c)*** anterior e o apartado ***a)*** do lema 5.1 para $k = 0$. ■

Agora, polo apartado ***a)*** do lema 5.2, pódese escribir

$$W_i = (P_i^*)^t A P_i + P_i^t A^t P_i^* = 2(P_i^*)^t A P_i. \tag{5.35}$$

Por outra banda, polo apartado ***d)*** do lema 5.2, a ecuación (5.15) pode volverse a escribir como

$$W_i y_i = 2(P_i^*)^t r_i. \tag{5.36}$$

Como consecuencia da sección ***b)*** do lema 5.2, tense que

$$(P_i^*)^t A Q_{i+1} = P_i^t A^t Q_{i+1}^*, \tag{5.37}$$

e entón, da ecuación (5.19), obtense

$$B_{i+1} = -(W_i)^{-1}\left(P_i^t A^t Q_{i+1}^* + (P_i^*)^t A Q_{i+1}\right) = -2(W_i)^{-1}\left(P_i^t A^t Q_{i+1}^*\right). \tag{5.38}$$

Finalmente pódese agora reescribir o s-BiCG do seguinte xeito:

Algoritmo 5.3 (s-BiCG).
Sexa $x_0, x_0^* \in \mathbb{R}^n$
$r_0 = b - Ax_0$
$r_0^* = b^* - A^t x_0^*$
$P_0 = \Delta_A(r_0)$
$P_0^* = \Delta_{A^t}(r_0^*)$
Para $i = 0, 1, 2, \ldots$ *ata converxencia*

$$W_i = (P_i^*)^t A P_i \tag{5.39}$$

$$y_i = W_i^{-1}(P_i^*)^t r_i \tag{5.40}$$

$$x_{i+1} = x_i + P_i y_i \tag{5.41}$$
$$r_{i+1} = r_i - AP_i y_i \tag{5.42}$$
$$r^*_{i+1} = r^*_i - A^t P^*_i y_i \tag{5.43}$$
$$Q_{i+1} = \Delta_A(r_{i+1}) \tag{5.44}$$
$$Q^*_{i+1} = \Delta_{A^t}(r^*_{i+1}) \tag{5.45}$$
$$B_{i+1} = -(W_i)^{-1}(P^*_i)^t A Q_{i+1} \tag{5.46}$$
$$P_{i+1} = Q_{i+1} + P_i B_{i+1} \tag{5.47}$$
$$P^*_{i+1} = Q^*_{i+1} + P^*_i B_{i+1} \tag{5.48}$$

Fin

No caso do s-BiCG, a $\mathbb{A}$-ortonormalización das columnas da matriz $\mathbb{P}_i$ complica moito a posibilidade de dar unha versión deste método en termos de vectores n dimensionais, polo que neste caso non se usará esta técnica, quedando o algoritmo anterior, no que se fai necesario o cálculo en cada iteración da inversa da matriz cadrada W_i de orde s. Como xa se comentou anteriormente, pode ocurrir que algunha destas matrices W_i sexa singular, co que o algoritmo dexeneraría.

As propiedades de ortogonalidade que se poden obter dos lemas 2.1 e 2.2 non son neste caso inmediatas en termos de vectores n dimensionais. Enúncianse e demóstranse no seguinte lema algunhas propiedades de biortogonalidade:

Lema 5.3 *No s-BiCG cúmplese que:*
a) $(Q^*_j)^t r_i = Q^t_j r^*_i = 0$ *para todo* $i > j$.
b) $(P^*_i)^t A P_j y_j = P^t_i A^t P^*_j y_j = 0$ *para todo* $i \neq j$.
c) $P^t_j r^*_i = (P^*_j)^t r_i = 0$ *para todo* $i > j$.

Demostración:

Do apartado ***b)*** do lema 2.2 tense:

$$\mathbb{Q}^t_j R_i = 0 \text{ para todo } i > j, \tag{5.49}$$

do apartado ***a)*** do lema 2.1:

$$\mathbb{P}^t_i \mathbb{A} \mathbb{P}_j = 0 \text{ para todo } i \neq j \tag{5.50}$$

e, do apartado ***b*** do lema 2.1

$$\mathbb{P}^t_j R_i = 0 \text{ para todo } i > j. \tag{5.51}$$

Desenvolvendo a ecuación (5.49)

$$(Q_j^*)^t r_i + (Q_j)^t r_i^* = 0 \text{ para todo } i > j, \tag{5.52}$$

a ecuación (5.50)

$$(P_i)^t A^t P_j^* + (P_i^*)^t A P_j = 0 \text{ para todo } i \neq j, \tag{5.53}$$

polo que

$$(P_i)^t A^t P_j^* y_y + (P_i^*)^t A P_j y_j = 0 \text{ para todo } i \neq j, \tag{5.54}$$

e, desenvolvendo a ecuación (5.51),

$$(P_j^*)^t r_i + (P_j)^t r_i^* = 0 \text{ para todo } i > j. \tag{5.55}$$

Por isto, será suficiente demostrar unha das igualdades en cada un dos tres apartados do lema e a outra será automaticamente certa. Posto que $((P_i^*)^t A P_j)^t = P_j^t A^t P_i^*$, o apartado ***b)*** basta demostralo para $i > j$. Deste xeito, demóstranse os tres apartados por indución para $i > j$.

Se $i = 1$, entón só pode ser $j = 0$. O apartado ***a)*** é certo para $i = 1$ e $j = 0$, xa que $Q_0 = P_0$, $Q_0^* = P_0^*$ e polo apartado ***c)*** do lema 5.2. Para probar o apartado ***b)*** para $i = 1$ e $j = 0$, úsase a definición (5.47) para desenvolver P_1, quedando

$$(P_1)^t A^t P_0^* = Q_1^t A^t P_0^* + B_1^t (P_0)^t A^t P_0^*. \tag{5.56}$$

E agora, usando as definicións (5.46) e (5.39) obtense

$$B_1 = -((P_0^*)^t A P_0)^{-1} (P_0^*)^t A Q_1, \tag{5.57}$$

que ao substituír na ecuación (5.56), queda

$$(P_1)^t A^t P_0^* = 0. \tag{5.58}$$

Para demostrar que o apartado ***c)*** é certo para $i = 1$ e $j = 0$, basta ter en conta que $r_1 = r_0 - A P_0 y_0$. Deste xeito,

$$(P_0^*)^t r_1 = (P_0^*)^t r_0 - (P_0^*)^t A P_0 y_0, \tag{5.59}$$

e, posto que $y_0 = W_0^{-1} (P_0^*)^t r_0$ e $W_0 = (P_0^*)^t A P_0$, chégase a que

$$(P_0^*)^t r_1 = 0. \tag{5.60}$$

Supóñanse certos agora os apartados ***a)***, ***b)*** e ***c)*** para $i-1$ e para todo $j < i-1$. Entón, para comprobar que o apartado ***a)*** é certo para i, pola definición de r_i pódese escribir

$$(Q_j^*)^t r_i = (Q_j^*)^t r_{i-1} - (Q_j^*)^t A P_{i-1} y_{i-1}. \tag{5.61}$$

Agora, se $j < i-1$, entón

$$(Q_j^*)^t r_{i-1} = 0 \tag{5.62}$$

pola hipótese de indución e, usando a definición (5.48) para P_j^*

$$(Q_j^*)^t A P_{i-1} y_{i-1} = (P_j^*)^t A P_{i-1} y_{i-1} - B_j^t (P_{j-1}^*)^t A P_{i-1} y_{i-1} = 0 \tag{5.63}$$

pola hipótese de indución para o apartado ***b)***.

Se $j = i-1$, tense que $Q_{i-1}^* r_{i-1} = P_{i-1}^* r_{i-1}$ polo apartado ***d)*** do lema 5.2, o cal pódese substituír na ecuación (5.61) e, usando de novo a definición (5.48) para P_{i-1}, tense que

$$((Q_{i-1}^*)^t A P_{i-1}) y_{i-1} = (P_{i-1}^*)^t A P_{i-1} y_{i-1} - B_{i-1}^t (P_{i-2}^*)^t A P_{i-1} y_{i-1}. \tag{5.64}$$

Pero $(P_{i-2}^*)^t A P_{i-1} y_{i-1} = 0$ pola hipótese de indución e, tendo en conta a definición (5.40) para y_{i-1} e (5.39) para W_{i-1},

$$(P_{i-1}^*)^t A P_{i-1} y_{i-1} = W_{i-1} (W_{i-1})^{-1} P_{i-1}^* r_{i-1} = P_{i-1}^* r_{i-1}, \tag{5.65}$$

co que, substituíndo na ecuación (5.61), queda probado o apartado ***a)***.

Para demostrar a certeza do apartado ***b)*** para todo $i > j$, úsase a definición (5.48) para desenvolver P_i^*, quedando

$$(P_i^*)^t A P_j y_j = (Q_i^*)^t A P_j y_j + (B_i)^t (P_{i-1}^*)^t A P_j y_j. \tag{5.66}$$

Se $j = i-1$ entón, das definicións (5.46) e (5.39), tense que

$$B_i = -((P_{i-1}^*)^t A P_{i-1})^{-1} (P_{i-1}^*)^t A Q_i \tag{5.67}$$

e, sacando factor común y_{i-1} e substituíndo na ecuación (5.66), chégase a que

$$(P_i)^t A^t P_{i-1}^* y_{i-1} = 0. \tag{5.68}$$

Por outra banda, se $j < i-1$, o segundo sumando da ecuación (5.66) é cero pola hipótese de indución. E usando a definición (5.42) para substituír

$$A P_j y_j = r_{j+1} - rj, \tag{5.69}$$

tense que o primeiro sumando da ecuación (5.66) tamén é cero polo apartado ***a)*** demostrado anteriormente. Deste xeito queda demostrado o apartado ***b)***.

Por último, a certeza do apartado ***c)*** para todo $i > j$ demóstrase usando a definición (5.42) para desenvolver r_i de xeito que

$$(P_j^*)^t r_i = (P_j^*)^t r_{i-i} - (P_j^*)^t A P_{i-i} y_{i-1} \tag{5.70}$$

e, se $j = i-1$, a ecuación (5.70) é cero pola definición (5.40) de y_{i-1}. E, se $j < i-1$, o primeiro sumando do segundo termo da ecuación (5.70) é cero pola hipótese de indución e o segundo sumando tamén polo apartado ***b)*** demostrado anteriormente. ■

Capítulo 6

Resultados numéricos

O obxectivo deste capítulo é presentar e analizar os resultados numéricos obtidos na implementación paralela dos métodos propostos nos capítulos 3, 4 e 5. Antes de presentar estes resultados introdúcense algúns conceptos relevantes sobre a programación paralela en xeral e outros máis concretos relacionados cos resultados numéricos presentados neste capítulo. En concreto, nun primeiro apartado defínese un sistema paralelo e dáse algunha das clasificacións máis habituais destes sistemas especificando finalmente o tipo de sistema paralelo onde se executan estas probas. Nunha segunda sección trátase a noción de matriz dispersa e, entre os diferentes formatos de almacenamento, o que se usa na programación destes métodos. Na seguinte sección preséntanse os distintos paradigmas de programación paralela e indícase cal é o paradigma escollido neste caso, así como a especificación máis habitual asociada a este paradigma e usada para a paralelización dos códigos, OpenMP. Nun cuarto apartado defínense os conceptos de aceleración e eficiencia, que se usarán como medida da escalabilidade dos métodos propostos. Finalmente móstranse os resultados obtidos coa implementación paralela dos métodos en s-pasos.

6.1. Sistemas paralelos

A computación paralela é unha técnica de programación na que varias instrucións execútanse simultaneamente. Baséase no principio de que os problemas grandes pódense dividir en partes máis pequenas que poden resolverse de xeito concurrente (en paralelo). Durante moitos anos, a computación paralela aplicouse na computación de altas prestacións, mais o interese nela aumentou nos últimos anos debido, entre outros factores, ás restricións de consumo enerxético dos microprocesadores

que derivaron na integración de múltiples núcleos nun único procesador (sistemas multinúcleo) [46, 70].

Un sistema paralelo pode definirse como unha colección de elementos de procesamento independentes que cooperan e se comunican entre sí, e que son capaces de resolver de xeito conxunto un único problema. Esta definición abrangue dende os recentes sistemas multinúcleo ata as contornas Grid ou Cloud [70]. Existen varios xeitos de clasificar os sistemas paralelos. Pode facerse considerando a organización interna dos procesadores, a estrutura de interconexión entre os procesadores ou o fluxo de información a través do sistema. Unha primeira clasificación presentada por Flynn [70] está baseada no fluxo de instrucións e de datos e distingue entre catro tipos de sistemas computacionais:

- Unha instrución, un dato (SISD). É o caso dun computador secuencial que non explota o paralelismo.
- Múltiples instrucións, un dato (MISD). Pouco común debido ao feito de que a efectividade dos múltiples fluxos de instrucións sole precisar de múltiples fluxos de datos.
- Unha instrución, múltiples datos (SIMD). Un computador que explota varios fluxos de datos dentro dun único fluxo de instrucións para realizar operacións que poden ser paralelizadas de xeito natural. Por exemplo, un procesador vectorial.
- Múltiples instrucións, múltiples datos (MIMD). Varios procesadores autónomos que executan simultaneamente instrucións diferentes sobre datos diferentes. Os sistemas distribuídos adoitan clasificarse como arquitecturas MIMD; ben sexa explotando un único espazo compartido de memoria, ou un distribuído.

Esta clasificación é moi utilizada porque é moi sinxela e proporciona unha primeira aproximación academicamente correcta. Non obstante non é moi axeitada para clasificar os sistemas paralelos actuais pois, a pesar das diferenzas que presentan, case todos eles coincidirían dentro da mesma categoría. A principal clasificación dos sistemas paralelos actuais baséase no paradigma de programación destes sistemas. Esta distingue basicamente entre tres grandes familias: memoria compartida, procesamento paralelo de datos e arquitecturas distribuídas (ou de paso de mensaxes).

Esta clasificación é tamén moi xenérica e existen outras aproximacións diferentes, pero é xeralmente considerada como unha boa aproximación para a clasificación da maioría dos sistemas utilizados en contornas da computación de altas prestacións.

6.1.1. Sistemas de memoria compartida

Nos sistemas de memoria compartida, tamén coñecidos como multiprocesadores, todos os procesadores poden acceder a calquera dirección de memoria do sistema utilizando as instrucións convencionais de acceso á memoria. O mecanismo máis habitual para programar estes sistemas é mediante OpenMP [20, 21] que, a través de directivas de compilación, proporciona unha interfaz sinxela para xerar código paralelo. Estes sistemas, á súa vez, poden clasificarse en procesadores simétricos (*Symmetric MultiProcessor*, SMP) tamén denominados UMA (*Uniform Memory Access*) e sistemas de acceso a memoria non uniforme (*Non Uniform Memory Access*, NUMA) [40, 46, 47, 70]. A arquitectura SMP utiliza un bus de comunicacións para conectar os procesadores coa memoria, garantindo que o custo de acceso a memoria é o mesmo para todos os procesadores, independentemente da posición de memoria solicitada. Os problemas de contención no bus do sistema condicionan, en gran medida, a escalabilidade destes sistemas. Pola contra, nos sistemas NUMA a distribución da memoria non é simétrica, polo que o custo de acceso á memoria dependerá da distancia entre o procesador e a posición de memoria solicitada. O esquema máis sinxelo dos sistemas NUMA asocia unha sección de memoria a cada procesador, de tal xeito que o acceso de cada procesador á súa memoria asociada presenta unha baixa latencia mentres que o acceso a posicións de memoria que estean asociadas a outro procesador implica unha maior penalización en tempo. Esta estrutura xerárquica pode ter máis niveis na arquitectura.

6.1.2. Procesamento paralelo de datos

A característica principal destes sistemas é que unha operación se executa en paralelo en cada elemento dunha estrutura regular, actuando sobre diferentes datos. Este tipo de sistemas encaixa dentro da categoría SIMD e é a base dos procesadores vectoriais. Actualmente, no ámbito da computación de altas prestacións, os sistemas baseados unicamente neste paradigma de programación están a desaparecer. Mais este tipo de técnicas seguen vixentes noutros sectores como, por exemplo, no procesamento de imaxes mediante tarxetas gráficas (*Graphics Processing Units*, GPUs)

ou os *systolic arrays*. Nos últimos anos, cobrou especial relevancia a programación de propósito xeral de GPUs (*General-Purpose computation on Graphics Processing Units*, GPGPU), na que se utiliza a GPU como un coprocesador matemático para realizar operacións sobre grandes estruturas de datos [40, 46, 74]. En certas aplicacións, este tipo de técnicas obteñen un rendemento moi superior ao obtido con microprocesadores de propósito xeral.

6.1.3. Arquitecturas paralelas distribuídas

Os sistemas paralelos distribuídos están formados por elementos computacionais independentes (nodos) unidos mediante unha rede de interconexión. A principal característica destes sistemas é que os accesos a memorias non locais, é dicir, a datos almacenados na memoria local doutros nodos, esixen unha comunicación explícita entre os nodos implicados. O paradigma de programación de paso de mensaxes é o mecanismo de programación habitual neste tipo de sistemas. Aínda que existen outras alternativas, a libraría MPI (*Message Passing Interface*) [38, 66] converteuse no estándar de facto do paradigma de paso de mensaxes en arquitecturas paralelas distribuídas. Os sistemas paralelos distribuídos son altamente escalables e, ademais, permiten introducir novos niveis de arquitectura dentro de cada nodo computacional. As arquitecturas máis comúns en computación de altas prestacións dos últimos anos foron arquitecturas paralelas distribuídas: clusters, constelacións e computadoras masivamente paralelas [40, 46, 70].

- Un cluster pode definirse como un sistema paralelo formado por nodos independentes conectados a través dunha rede de interconexión dedicada. En xeral, os clusters son sistemas pouco acoplados, o que permite que sexan moi robustos e, ao mesmo tempo, facilita a administración e desenvolvemento dos propios sistemas. Ademais da súa flexibilidade e versatilidade, o éxito destes sistemas débese á excelente relación prezo-rendemento. A configuración máis sinxela de cluster, coñecida como Beowulf, consiste en múltiples PCs conectados mediante unha rede de tipo Ethernet, xeralmente utilizando un sistema operativo GNU/Linux. Actualmente os sistemas máis habituais están formados por procesadores multinúcleo interconectados por redes InfiniBand ou Gigabit Ethernet. A aparición de nodos formados por múltiples núcleos produce dous niveis de comunicación diferentes: internodo e intranodo. Debido ás diferentes características que presentan estes dous niveis de comunicación, a adaptación

dos programas a esta distribución xerárquica dos elementos computacionais é fundamental para aproveitar eficazmente estes sistemas.

- As constelacións considéranse unha subclase de cluster na que o número de núcleos por nodo é superior ao número de nodos do sistema completo. É dicir, o nivel de paralelismo dominante nas constelacións é o nivel intranodo. En xeral, os nodos destes sistemas están formados por multiprocesadores (SMP ou sistemas NUMA), cun gran número de núcleos. Para obter alto rendemento deste tipo de sistemas é necesario utilizar un paradigma de programación híbrido, que utilice paso de mensaxes nas comunicacións internodo, mentres que dentro de cada nodo pódense executar diferentes fíos baixo o paradigma SMP.

- Os computadores masivamente paralelos (*Massively Parallel Processing*, MPP), ao contrario que os clusters, son sistemas distribuídos cun nivel de acoplamento moi alto. Nos sistemas MPP os nodos son independentes (é dicir, conteñen a súa propia memoria e unha copia do sistema operativo) e están conectados a través dunha rede de alta velocidade, pero xestiónanse como un único sistema, de maneira similar a un multiprocesador. En xeral, o rendemento destes sistemas é maior que o dos clusters, xa que algúns compoñentes adoitan deseñarse especificamente para estes sistemas.

A implementación dos métodos probados neste capítulo foi realizada no sistema Finis Terrae do Centro de Supercomputación de Galicia (CESGA) [17], contando tamén coa axuda puntual dos técnicos de sistemas e outro persoal deste centro. O sistema Finis Terrae está formado por 142 nodos de computación HP Integrity rx7640, cada un deles con 16 núcleos Itanium Montvale a 1,6 GHz e 128 GB de memoria principal, e 1 nodo HP Integrity Superdome con 128 núcleos Itanium Montvale e 1024 GB de memoria.

6.2. Matrices dispersas

As probas numéricas executáronse sobre matrices exemplo da páxina web Matrix Market [61]. Escolléronse matrices reais dispersas, cadradas, non singulares e de gran tamaño, coas características particulares necesarias para a converxencia do método executado. Unha matriz dispersa é aquela cun gran número de termos nulos, no senso de que sexa suficiente como para que este feito poida ser aproveitado en

termos de memoria e tempo de execución.

Se n é a orde da matriz (e polo tanto o número de filas ou columnas) e nnz o número de elementos non nulos, nunha matriz dispersa cúmprese que $nnz << n^2$. Normalmente almacénanse as entradas non nulas e certa información acerca da súa posición específica dentro da estrutura da matriz. Deste xeito afórrase espazo de almacenamento en memoria. Pódense citar, entre os esquemas de almacenamento de matrices dispersas, algúns dos máis usados [10]: SKS (*Sky-line Storage*), JDS (*Jagged Diagonal Storage*), BCRS (*Block Compressed Row Storage*), CDS (*Compressed Diagonal Storage*), CCS (*Compressed Column Storage*) e CRS (*Compressed Row Storage*). A elección dun esquema de almacenamento ou outro soe estar guiada polo tipo de patrón de accesos do código que vaia realizarse sobre a matriz dispersa en cada aplicación.

Nas probas numéricas que se presentan neste capítulo utilizouse o formato CRS. Este formato é equivalente ao CCS salvo que o CRS percorre a matriz por filas e o CCS faino por columnas. A razón do seu uso é que resultou ser o formato máis eficiente na paralelización dos produtos matriz dispersa vector e matriz dispersa matriz, xa que ambas as dúas operacións fanse percorrendo a primeira matriz (neste caso a dispersa) por filas.

O formato CRS consiste en almacenar os datos da matriz e a información sobre a súa posición nela nos tres vectores seguintes:

- O vector real *values* de dimensión nnz que contén os datos non nulos da matriz ordeados seguindo a orde imposta polas filas.

- O vector enteiro *colind* de dimensión nnz cos índices das columnas á que pertencen os elementos almacenados no vector *values*.

- O vector enteiro *rowptr* de dimensión $n+1$, que contén o punteiro (ou posición) do primeiro elemento non nulo de cada fila da matriz no vector *values*. O último elemento de *rowptr* é $nnz + 1$.

Por exemplo, a matriz

$$A = \begin{pmatrix} 1. & 0. & 0. & 2. & 0. \\ 3. & 4. & 0. & 5. & 0. \\ 6. & 0. & 7. & 8. & 9. \\ 0. & 0. & 10. & 11. & 0. \\ 0. & 0. & 0. & 0. & 12. \end{pmatrix}$$

almacénase, no formato CRS, nos vectores:

- *values*= $(1., 2., 3., 4., 5., 6., 7., 8., 9., 10., 11., 12.)$
- *colind*= $(1, 4, 1, 2, 4, 1, 3, 4, 5, 3, 4, 5)$
- *rowptr*= $(1, 3, 6, 10, 12, 13)$

A linguaxe utilizada na programación dos métodos é Fortran 95. O compilador de Fortran usado, dos dispoñibles no CESGA, é o propio de Intel. Entre outras vantaxes, con esta linguaxe de programación simplifícase o código nas operacións vectoriais e matriciais. Fortran 95 dispón das súas propias funcións intrínsecas para as operacións con vectores e matrices. Non obstante, as operacións con matrices dispersas deben realizarse cun código realizado ao efecto. No cadro 6.1 móstrase o código para o produto dunha matriz dispersa A, cadrada de orde n, en formato CRS, por un vector $x \in \mathbb{R}^n$. O resultado é o vector $y \in \mathbb{R}^n$.

DO $i = 1, n$
    $k1 = rowptr(i)$
    $k2 = rowptr(i+1) - 1$
    $y(i) =$Dot_Product$(values(k1 : k2), x(colind(k1 : k2)))$
END DO

Cadro 6.1: Produto matriz dispersa por vector

O código do produto dunha matriz dispersa A, cadrada de orde n, en formato CRS, por unha matriz B de orde $n \times s$ pode verse no cadro 6.2. O resultado é a matriz C de orde $n \times s$.

DO $k = 1, s$
  DO $i = 1, n$
    $k1 = rowptr(i)$
    $k2 = rowptr(i+1) - 1$
    $C(i,k)$ =Dot_Product$(values(k1:k2), B(colind(k1:k2), k))$
  END DO
END DO

Cadro 6.2: Produto matriz dispersa por matriz

6.3. Programación Paralela

Existen diversos paradigmas na programación paralela, en moitos casos ligados á arquitectura ou sistema paralelo onde se implementa. Actualmente sobresaen os seguintes modelos de programación.

- **Programación mediante o paradigma de paso de mensaxes.** Este paradigma é o máis axeitado en sistemas paralelos con memoria distribuída, como os clústers. Os procesos comunícanse enviando e recibindo mensaxes e facendo chamadas a bibliotecas as cales administran o intercambio de datos entre os procesos. A biblioteca estándar neste momento, e portable, para o paso de mensaxes é MPI (*Message Passing Interface*), dispoñible para Fortran, C e C++. Neste modelo de programación todo o paralelismo é explícito, é dicir, o programador é o responsable de extraer o paralelismo, distribuíndo os datos e computacións, e de establecer as comunicacións e sincronizacións.

- **Programación mediante directivas de memoria compartida.** Neste modelo a memoria é común a todos os fíos e as variables poden ser compartidas por todos os fíos, é o modelo máis axeitado ás arquitecturas do tipo multinúcleo. A coordinación e cooperación entre os fíos realízase a través da lectura e escritura das variables compartidas e a través de variables de sincronización. O estándar neste caso é OpenMP, que permite ao programador engadir ao programa secuencial directivas de compilación que indican á computadora onde e como hai que paralelizar. A vantaxe deste modelo é que é moito máis sinxelo paralelizar á vez que se consegue un código portable. Non obstante é menos flexible e eficiente, xa que non ten tanto control por parte do programador.

- **Modelo de programación híbrida.** Este é o modelo que combina os dous anteriores, implementaríase nunha constelación onde existe una xerarquía de memoria en varios niveis. A constelación ten varios nodos cada un coa súa memora local e unha serie de núcleos que comparten dita memoria. Dentro de cada nodo o modelo sería o da memoria compartida e pasaríase a un modelo híbrido no caso de usar varios nodos. Para este modelo existe algunha linguaxe recentemente desenvolvida, como UPC [37], mais o habitual é o uso conxunto de OpenMP e MPI.

As probas numéricas que se presentan neste traballo relizáronse no Finis Terrae do CESGA que, como xa se comentou, é un sistema con 142 nodos e cada un deles con 16 núcleos. Decidiouse escoller un modelo de programación mediante directivas de memoria compartida usando un único nodo en cada execución e ata 16 núcleos. Para iso úsase OpenMP.

OpenMP (Open Multi-Processing) [20, 63] é unha API (Application Program Interface) que nace do traballo conxunto de importantes fabricantes de hardware e software como AMD, IBM, Intel, Cray, HP, Fujitsu, NVIDIA, NEC, Microsoft, Texas Instruments, Oracle Corporation e algunha máis [63], ofrecendo ao usuario un modelo de programación de paralelismo explícito. Esta API está composta por un conxunto de directivas do compilador que o usuario utiliza para especificar que rexións de código van ser paralelizadas, e unha libraría de rutinas. A implementación de OpenMP utiliza fíos de execución paralelos (multithreading) para obter o paralelismo, mediante un modelo coñecido como fork-join, onde unha tarefa divídese en varios fíos (*fork*), para logo recolectar os seus resultados ao final e unilos nun só resultado (*join*). Deste xeito, ao atoparse unha directiva paralela que así o indique, o fío maestro divídese nun número determinado de fíos esclavos que se executan concurrentemente e as tarefas distribúense entre eles. En OpenMP todos os fíos acceden á mesma memoria, aínda que é posible xestionar estes accesos e xerar espazos de memoria privada, entre outras operacións. Diso encárganse algunhas das súas directivas, que se ven complementadas polas cláusulas, con diversos usos dependendo da directiva á que acompañen. En xeral, estas cláusulas úsanse para modificar o carácter das variables (privadas ou compartidas), para levar a cabo operacións de sincronización, inicialización, copia de datos, redución, control de fluxo, etc.

O paralelismo con OpenMP faise principalmente a nivel de bucle, introducindo directivas naqueles bucles que sexan paralelizables. Para iso hai que evitar, ás veces modificando o código, a existencia de dependencias no bucle, e procurar, cando existan bucles anidados, paralelizar o bucle máis externo posible. No cadro 6.3 móstrase o código paralelo para o produto matriz dispersa por vector.

```
!$OMP PARALLEL DO
DO i = 1, n
        k1 = rowptr(i)
        k2 = rowptr(i + 1) − 1
        y(i) =Dot_Product(values(k1 : k2), x(colind(k1 : k2)))
END DO
```

Cadro 6.3: Produto matriz dispersa por vector con OpenMP

E no cadro 6.4 pode verse o código paralelo do produto dunha matriz dispersa por matriz. Neste caso intercambiouse a orde dos bucles anidados, facendo que o bucle externo sexa o bucle en n, xa que $n >> s$, e polo tanto, o que interesa paralelizar é o bucle en n.

```
!$OMP PARALLEL DO
DO i = 1, n
        DO k = 1, s
                k1 = rowptr(i)
                k2 = rowptr(i + 1) − 1
                C(i, k) =Dot_Product(values(k1 : k2), B(colind(k1 : k2), k))
        END DO
END DO
```

Cadro 6.4: Produto matriz dispersa por matriz con OpenMP

A directiva WORKSHARE de OpenMP paraleliza as operacións vectoriais e matriciais en Fortran 95. Non obstante, as probas feitas no Finis Terrae mostraron que o uso desta directiva non producía ningunha mellora sobre o código orixinal, o cal aconsellou non usar as operacións sobre vectores e matrices nin as funcións de Fortran 95. Reescribíronse, desenvolvendo en bucles, todas as operacións vectorias e matricias e paralelizáronse os bucles coas correspondentes directivas.

6.4. Aceleración e eficiencia

O obxectivo principal da computación paralela é reducir o tempo de execución dun programa repartindo o traballo computacional entre varias CPUS que executan o código en paralelo. Para medir a ganancia en rendemento ou eficiencia dun programa paralelo respecto ao programa en secuencial, defínense algunhas medidas como a aceleración ou a eficiencia. Se un mesmo programa se executa nun fío e en Nth fíos e se denota por T_1 e T_{Nth} aos respectivos tempos de execución, podería esperarse que se cumprise nun hipotético caso ideal que $T_{Nth} = T_1/Nth$. Isto non se cumpre en xeneral pois, entre outras cousas, as comunicacións e sincronizacións entre fíos consomen o seu tempo na execución dun programa paralelo. Unha medida que indica o beneficio relativo de resolver un determinado problema en paralelo é a aceleración (*speedup*).

A aceleración defínese como o cociente entre o tempo de execución dun algoritmo en secuencial (e polo tanto nun único fío) e o tempo de execución do mesmo algoritmo en paralelo con Nth fíos [56]. É dicir:

$$S = \frac{T_1}{T_{Nth}}. \tag{6.1}$$

O caso óptimo no que se obtería un maior beneficio coa programación en paralelo debería ser cando $S = Nth$. En teoría, a aceleración non debería superar nunca o valor de Nth. Tense así que

$$0 < S \leq Nth, \tag{6.2}$$

e canto máis se achegue a aceleración a Nth, mellor será o rendemento que obtemos da programación en paralelo. Porén, na práctica dáse $S > Nth$ nalgunos casos (superaceleración ou aceleración superlinear). En xeral, este feito débese a un mellor aproveitamento da xerarquía de memoria. Pola natureza do estudo que se fai coas variantes en s-pasos, onde o interese está na comparación da variante co método orixinal, medirase neste traballo a aceleración respecto ao tempo de execución do método orixinal en secuencial. É dicir, T_1 é o tempo que tarda o método orixinal ($s = 1$) en executarse en secuencial ($Nth = 1$).

Por outra banda, unha medida que indica a porcentaxe de tempo que cada procesador, por termo medio, dedica á execución do programa (e polo tanto ao tempo que non se perde en comunicacións e sincronizacións) é a *eficiencia*, que se

define como o cociente entre a aceleración e o número de fíos utilizados na execución do algoritmo. Polo tanto:

$$E = \frac{S}{Nth}. \tag{6.3}$$

Por suposto, ante un comportamento ideal teríase que o valor da aceleración é igual a Nth e polo tanto $E = 1$, deste xeito

$$0 < E \leq 1, \tag{6.4}$$

e teríase superaceleración naqueles casos en que $E > 1$.

6.5. Resultados

Nesta sección preséntanse os resultados numéricos das diferentes variantes en s-pasos propostos e a comparación cos respectivos métodos orixinais. Aínda que dalgunhas das variantes xa existen resultados numéricos publicados anteriormente [24, 25, 77], os resultados numéricos que se presentan neste traballo son das súas versións N-ortonormalizadas. Destas últimas tan só existen resultados numéricos publicados do s-Residuo Conxugado Xeneralizado e s-Orthomin(m) en [29].

Cada método foi executado sobre varias matrices. Mostránse os resultados obtidos cunha das matrices, tendo en conta que, sempre que o método orixinal converxa, os resultados sobre outras matrices son similares.

Na execución dos métodos, o proceso iterativo finalizará cando se cumpra unha determinada condición que se denomina test de parada. Na programación destes métodos o test de parada utilizado é $\frac{\|r_i\|}{\|r_0\|} < \epsilon$, con $\epsilon \in \mathbb{R}$ un valor prefixado en cada método. Se existe algún número natural k tal que o test de parada se cumpre con $i = k$ e non se cumpre con $i < k$, entón o algoritmo para e dise que o método converxe en k iteracións. Posto que o obxectivo é o estudo da escalabilidade na implementación paraela, para cada método axústaxe o test de parada fixando o ϵ, de xeito que o número de iteracións ata a converxencia sexa suficientemente grande. Canto maior sexa o número de iteracións ata a converxencia máis se notará a mellora da escalabilidade nos métodos en s-pasos, se esta existe.

6.5.1. s-Gradente Conxugado

As primeiras probas realízanse co s-Gradente Conxugado. Da variante en s-pasos do Gradente Conxugado xa existen resultados numéricos publicados en [24], o cal fai esperar resultados positivos neste primeiro caso. Non obstante a variante probada nesta sección é a que contén a N-ortonormalización dos vectores directores calculados en cada iteración, da cal non existen resultados publicados ata a data.

Os resultados que se presentan nesta sección obtivéronse coa matriz s3dkt3m2 da Matrix Martket [61]. Esta matriz é cadrada simétrica, definida positiva, de orde 90449 e 3753461 elementos non nulos. Ao igual que o resto das matrices que se usan, un dos formatos no que se pode descargar a matriz é o CCS, mais non o RCS, polo que o primeiro paso que se fai necesario é o uso dunha rutina que cambie o formato CCS ao RCS, que é o que se decide usar polas razóns xa explicadas. Na páxina do proxecto *BeBOP Sparse Matrix Converter* [11] ofrécese unha librería de rutinas en C que realizan o cambio entre distintos formatos de matrices dispersas.

O test de parada que se escolle nestas probas concretas é $\epsilon = 10^{-6}$. Por unha banda executóuse o método orixinal, que equivale á variante para $s = 1$ e pola outra a variante para distintos valores de s. Decidiuse ir collendo valores pares para s. A medida que aumenta o valor de s o número de iteracións en que converxe o método vai diminuíndo. O valor da s máis grande escollido é aquel a partir do cal o número de iteracións en que converxe comenza a crecer. Neste caso particular os valores de s son $s = 1, 2, 4, 6, 8, 10, 12$ e 14.

O cadro 6.5, mostra o número de iteracións en que converxe o método para os distintos valores dados de s. Na segunda liña do cadro móstrase o cociente entre o número de iteracións en que converxe o método para $s = 1$ e o número de iteracións en que converxe o método para o correspondente valor de s. Deste xeito pode comprobarse en que factor se reduce o número de iteracións para os distintos valores de s, verificándose, como era de esperar, que o número de iteracións no que converxe o método para cada valor de s ($Iter(s)$) é próximo ao cociente entre o número de iteracións no que converxe o método orixinal ($Iter(s = 1)$) e o valor concreto de s. Este cociente non é exacto debido a que nos cálculos úsase aritmética con precisión finita.

	s=1	**s=2**	**s=4**	**s=6**	**s=8**	**s=10**	**s=12**	**s=14**
Iteracións	15474	7744	3874	2584	1939	1553	1295	1111
$\frac{Iter(s=1)}{Iter(s)}$	1	2	3.99	5.99	7.98	9.96	11.96	13.93

Cadro 6.5: Iteracións s-GC

A seguinte variable que se mide nestas probas é o tempo de execución en segundos, que se mostran no cadro 6.6. Resáltanse en cor vermello os tempos mínimos de entre todas as variantes executadas nun mesmo número de fíos. Estes son os valores mínimos de cada fila no cadro 6.6. Analizando estes resultados pode comprobarse como, a partir de 4 fíos, os tempos de execución melloran en todos os casos (excepto $s = 2$ e 4 fíos e $s = 14$ e 8 fíos) da variante en s-pasos respecto ao método orixinal. Non obstante, esta mellora non é máis pronunciada a medida que aumenta s, incluso pode comprobarse como para 16 fíos os tempos permanecen todos en valores moi similares para os distintos valores de s. Deste xeito os mellores tempos de execución obtéñense para valores como $s = 4$ ou $s = 6$, nalgún caso para $s = 2$.

Tempo	**s=1**	**s=2**	**s=4**	**s=6**	**s=8**	**s=10**	**s=12**	**s=14**
Nth=1	688.27	803.97	790.18	784.64	876.43	942.75	992.34	1095.68
Nth=2	398.29	396.89	386.93	470.86	492.06	401.80	432.12	460.28
Nth=4	216.88	346.69	218.06	195.36	198.25	204.71	212.32	208.92
Nth=8	112.77	77.77	79.45	85.02	82.39	81.79	86.13	114.62
Nth=16	74.88	40.61	39.71	40.75	40.45	40.15	41.44	44.26

Cadro 6.6: Tempos de execución s-GC

A aceleración proporciona unha medida da escalabilidade do método e a súa variante en s-pasos para os distintos valores de s. No cadro 6.7 pódense comparar as aceleracións medidas en todos os casos. Sinálase neste caso en vermello aqueles valores máximos entre os que son executados cun mesmo número de fíos. Neste caso, son os máximos de cada fila no cadro 6.7 que, loxicamente, coinciden no mesmo lugar que os tempos mínimos resaltados no cadro 6.6. O salientable neste cadro é comprobar que para 8 e 16 fíos coa variante en s-pasos para $s = 2, 4, 6, 8$ e 10 obtéñense acelereracións moi superiores as que se obteñen co método orixinal ($s = 1$), superando incluso o límite teórico establecido polo número de fíos (superaceleración). Deste xeito danse nestes casos exemplos de superaceleración.

Aceleración	s=1	s=2	s=4	s=6	s=8	s=10	s=12	s=14
Nth=1	1	0.86	0.87	0.88	0.79	0.73	0.69	0.63
Nth=2	1.73	1.73	1.78	1.46	1.40	1.71	1.59	1.50
Nth=4	3.17	1.99	3.16	3.52	3.47	3.36	3.24	3.29
Nth=8	6.10	8.85	8.66	8.10	8.35	8.42	7.99	6.00
Nth=16	9.19	16.95	17.33	16.89	17.02	17.14	16.61	15.55

Cadro 6.7: Aceleración s-GC

Cunha representación gráfica das aceleracións no gráfico 6.1 pode apreciarse mellor como mellora a aceleración en paralelo na variante en s-pasos en todos os casos representados con respecto ao método orixinal ($s = 1$). Escolléronse para este gráfico unicamente os valores de $s = 1, 2, 4, 6, 8$ e 10. Coa intención de obter un gráfico máis claro elimínanse os valores $s = 12$ e $s = 14$, para os que os valores da aceleración tampouco crecen significativamente. O único caso no que se pode apreciar que a variante en s-pasos non mellora a aceleración respecto ao orixinal neste exemplo é para $s = 2$ e 4 procesos.

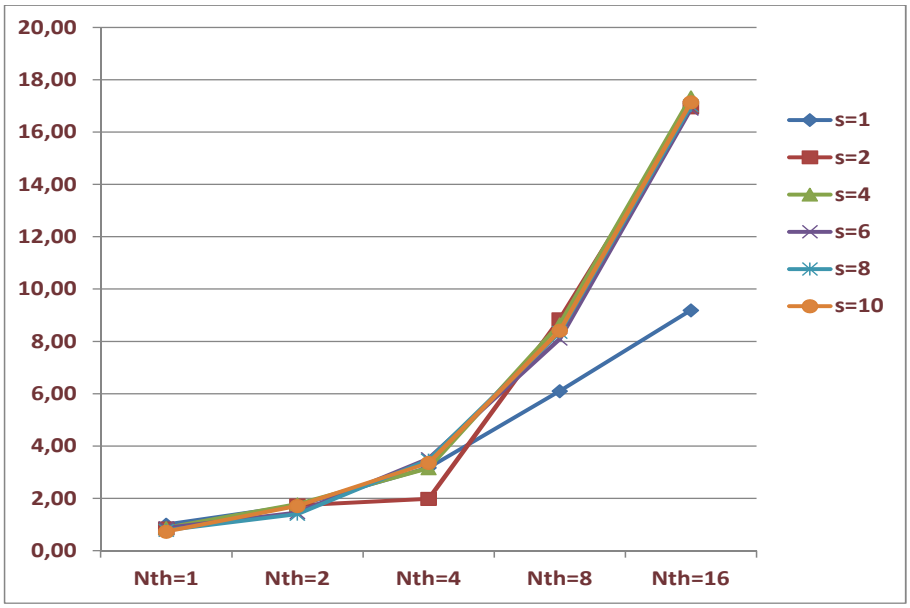

Figura 6.1: Aceleración para s-CG.

Neste exemplo preséntase un último cadro, o cadro 6.8 coas eficiencias correspondentes a cada execución. De igual xeito que nos cadros anteriores, resáltese en vermello as eficiencias máximas para cada fío, é dicir, en cada fila do cadro. Estes máximos coinciden tamén nos mesmos lugares que os sinalados nos cadros anteriores. Os datos presentados no cadro 6.8 permiten comparar a eficiencia da implementación paralela entre todos os casos, incluso executados con distinto número de fíos. A eficiencia medida nas execucións para 1 fío, primeira fila ($Nth = 1$), mostra o valor da eficiencia da variante en s-pasos, para cada s, executada en secuencial, con respecto ao método orixinal en secuencial. Aínda que se pode apreciar en secuencial que, para valores de $s > 1$ a eficiencia baixa bastante, cando se utilizan varios fíos contrarréstase este efecto, chegando incluso a sobrepasar eficiencias maiores que 1. Estas corresponden cos datos do cadro 6.7 nos que se aprecia superaceleración.

Eficiencia	**s=1**	**s=2**	**s=4**	**s=6**	**s=8**	**s=10**	**s=12**	**s=14**
Nth=1	1	0.86	0.87	0.88	0.79	0.73	0.69	0.63
Nth=2	0.86	0.87	0.89	0.73	0.70	0.86	0.80	0.75
Nth=4	0.79	0.50	0.79	0.88	0.87	0.84	0.81	0.82
Nth=8	0.76	1.11	1.08	1.01	1.04	1.05	1.00	0.75
Nth=16	0.57	1.06	1.08	1.06	1.06	1.07	1.04	0.97

Cadro 6.8: Eficiencia s-GC

Aínda que a información que ofrece a medida da eficiencia ten algún matiz diferente á aceleración, nos seguintes métodos prescíndese dos cadros con esta medida xa que é doadamente computable a partir dos valores da aceleración, e polo tanto, vén dando información redundante coa obtida a partir da aceleración.

6.5.2. s-Residuo Conxugado

O método de proba neste caso é o Residuo Conxugado e a súa correspondente variante en s-pasos. O método do Residuo Conxugado converxe para matrices simétricas e non singulares. Aínda que a matriz non ten porque ser definida positiva preséntanse os resultados obtidos coa mesma matriz que se usou para as probas no s-Gradente Conxugado, a s3dkt3m2, de orde 90449 e 3753461 elementos non nulos. Para o test de parada escóllese tamén $\epsilon = 10^{-6}$, do mesmo xeito que se fixo co s-Gradente Conxugado. Execútase o método orixinal ($s = 1$) e a correspondente

variante en s-pasos para $s = 2, 4, 6, 8, 10, 12$ e 14.

No cadro 6.9, co número de iteracións no que converxe, compróbase o esperado en canto a que o número de iteracións decrece cando o valor de s aumenta, de xeito que é aproximadamete igual ao número de iteracións do método orixinal divido polo valor de s. Non obstante, o que destaca neste caso é que, aínda que a matriz e o test de parada son os mesmos, o número de iteracións para a converxencia do método é moito máis pequeno que o correspondente para o s-Gradente Conxugado, polo que, neste exemplo, o s-Residuo Conxugado ten maior velocidade de converxencia que o s-Gradente Conxugado.

	s=1	**s=2**	**s=4**	**s=6**	**s=8**	**s=10**	**s=12**	**s=14**
Iteracións	4626	2313	1157	772	579	463	386	331
$\frac{Iter(s=1)}{Iter(s)}$	1	2	4	5.99	7.99	9.99	11.98	13.98

Cadro 6.9: Iteracións s-RC

No cadro 6.10 móstranse os tempos de execución. Estes son bastante menores que no s-Gradente Conxugado en todos os casos. Pode comprobarse tamén como en secuencial ($Nth = 1$), a diferenza do que acontece co s-Gradente Conxugado, os tempos de execución da variante s-pasos, con $s = 2, 4, 6, 8$ e 10, son menores que o tempo de execución do método orixinal ($s = 1$), é dicir, neste caso, a variante en s-pasos xa resulta algo máis eficiente que o método orixinal incluso en secuencial. Este feito poderíase explicar pola localidade, as matrices de orde $n \times s$ caberían na cache ata certos valores de s, e para valores maiores xa non caberían.

Tempo	**s=1**	**s=2**	**s=4**	**s=6**	**s=8**	**s=10**	**s=12**	**s=14**
Nth=1	224.91	210.24	189.10	203.05	214.41	220.47	240.41	266.11
Nth=2	129.67	101.69	93.86	117.22	117.18	121.66	129.31	135.59
Nth=4	62.11	60.63	56.13	51.70	51.82	57.37	53.04	57.25
Nth=8	22.11	20.77	19.68	20.25	19.65	22.09	22.79	24.07
Nth=16	11.33	10.80	10.14	9.25	10.72	10.91	11.10	11.48

Cadro 6.10: Tempos de execución s-RC

Os valores mínimos do tempo de execución para cada valor do número de fíos danse neste caso para maiores valores de s, $s = 4, 6$ e incluso $s = 8$ con 8 fíos. Aínda

que para $s = 14$ segue reducíndose o número de iteracións para a converxencia, os tempos de execución empeoran e, de feito, son maiores incluso que os correspondentes do método orixinal (excepto para $Nth = 4$).

O cadro de aceleracións 6.11 mostra valores moi grandes da aceleración tamén para o método orixinal, habendo superaceleración en 8 e 16 fíos. Non obstante, a aceleración da variante en s-pasos ten valores por enriba da aceleración do método orixinal en todos os casos excepto para $s = 14$, obténdose superaceleración, ademais de para 8 e 16 fíos, para algúns valores de s con 4 fíos.

Aceleración	**s=1**	**s=2**	**s=4**	**s=6**	**s=8**	**s=10**	**s=12**	**s=14**
Nth=1	1	1.07	1.19	1.11	1.05	1.02	0.94	0.85
Nth=2	1.73	2.21	2.40	1.92	1.92	1.85	1.74	1.66
Nth=4	3.62	3.71	4.01	4.35	4.34	3.92	4.24	3.93
Nth=8	10.17	10.83	11.43	11.11	11.45	10.18	9.87	9.34
Nth=16	19.85	20.83	22.17	24.31	20.97	20.61	20.26	19.59

Cadro 6.11: Aceleración s-RC

Móstrase a continuación a gráfica 6.2 coas aceleracións do método orixinal ($s = 1$) e a variante en s-pasos para $s = 2, 4, 6$. Non se representa a aceleración do resto dos valores de s xa que son resultados moi similares e sobrecargan en exceso a gráfica. Nela pode comprobarse como os valores da aceleración da variante en s-pasos están por enriba da aceleración do método orixinal, se ben esta diferenza non é tan pronunciada como é a do s-Gradente Conxugado.

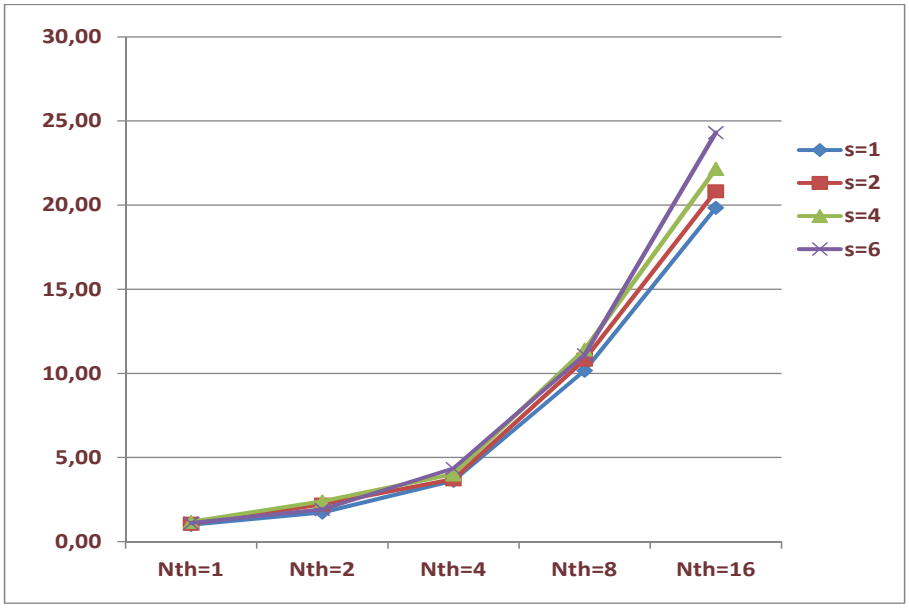

Figura 6.2: Aceleración para s-Residuo Conxugado.

6.5.3. s-Ecuación Normal

A variante en s-pasos da Ecuación Normal aparece proposto en [22], mais non existen, no noso coñecemento, resultados numéricos desta variante na literatura. O método da Ecuación Normal é, na teoría, un método converxente para calquera matriz cadrada singular. Os resultados que se mostran son os obtidos coa matriz af23560, cadrada, non simétrica mais non singular, de orde 23560 e 484256 elementos non nulos. É sabido que o método da Ecuación Normal ten unha converxencia máis lenta que os anteriores, polo que para o test de parada tómase un valor menor, $\epsilon = 10^{-4}$.

Execútase o método orixinal e a variante en s-pasos para $s = 2, 4, 6, 8$ e 10, pois para $s > 10$ o número de iteracións no que converxe comenza a aumentar. O número de iteracións en que converxen as distintas execucións preséntanse no cadro 6.12.

	s=1	**s=2**	**s=4**	**s=6**	**s=8**	**s=10**
Iteracións	21371	11641	5520	3565	2726	2240
$\frac{Iter(s=1)}{Iter(s)}$	1	1.84	3.87	5.99	7.84	9.54

Cadro 6.12: Iteracións s-Ecuación Normal

Os tempos de execución móstranse no cadro 6.13. Pode verse que a variante en s-pasos, tamén neste caso, rebaixa os tempos de execución en secuencial para $s = 4, 6, 8$ e 10. Ademais, obsérvanse dous grupos de resultados, o primeiro correspondente aos resultados en 2 e 4 fíos ($Nth = 2$ e 4), onde se presentan diminucións no tempo de execución en todos os casos excepto $s = 2$ e o tempo mínimo en $s = 6$. E un segundo grupo, cos resultados correspondentes a 8 e 16 fíos ($Nth = 8$ e 16), onde os tempos de execución só decrecen para $s = 4$ e 6 e $s = 2$ en 16 fíos, téndose os valores mínimos para $s = 4$.

Tempo	**s=1**	**s=2**	**s=4**	**s=6**	**s=8**	**s=10**
Nth=1	297.26	303.74	284.44	268.71	278.64	292.47
Nth=2	110.68	113.51	99.67	96.66	101.19	105.84
Nth=4	57.54	60.11	52.70	51.23	52.98	56.27
Nth=8	33.23	34.02	30.46	31.13	33.41	34.63
Nth=16	22.40	21.40	20.02	20.34	23.16	23.59

Cadro 6.13: Tempos de execución s-Ecuación Normal

Os valores da aceleración no cadro 6.14 mostran que, no caso da variante en s-pasos da Ecuación Normal, a mellora na eficiencia desta variante respecto ao orixinal en paralelo non é moita, se ben para $s = 4$ e $s = 6$ obtéñense mellores aceleracións que co método orixinal ($s = 1$).

Aceleración	**s=1**	**s=2**	**s=4**	**s=6**	**s=8**	**s=10**
Nth=1	1	0.98	1.05	1.11	1.07	1.02
Nth=2	2.69	2.62	2.98	3.08	2.94	2.81
Nth=4	5.17	4.95	5.64	5.80	5.61	5.28
Nth=8	8.95	8.74	9.76	9.55	8.90	8.58
Nth=16	13.27	13.89	14.85	14.61	12.83	12.60

Cadro 6.14: Aceleración s-Ecuación Normal

Na gráfica 6.3 pode visualizarse mellor o comentado anteriormente. Tan só se representan os casos $s = 1, 2, 4$ e 6 e pode verse que non existen moitas diferenzas, se ben as curvas de aceleracións dos casos $s = 4$ e $s = 6$ van por enriba da curva correspondente a $s = 1$.

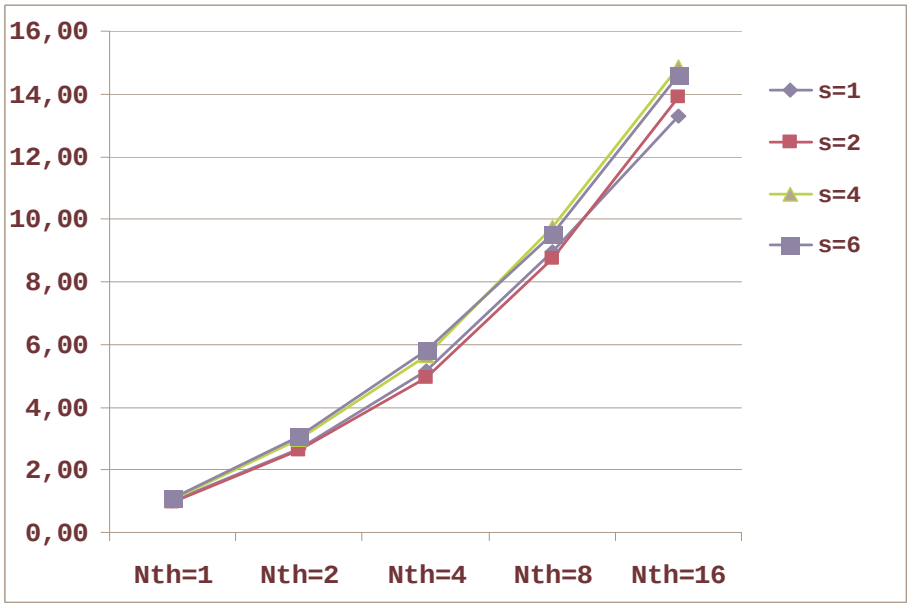

Figura 6.3: Aceleración para s-Ecuación Normal.

6.5.4. s-Erro Minimal

A variante en s-pasos do Erro Minimal é unha das variantes que se aportan como novidade nesta tese, proposta anteriormente en [3] polo autor e directores da mesma. Neste mesmo artigo preséntanse tamén, de xeito resumido, os resultados numéricos sobre esta variante que se mostran nesta sección.

Dado que é un método converxente para calquera matriz cadrada non singular, escóllese neste caso a matriz fidapm11, que é cadrada non simétrica de orde 22294 e con 623554 elementos non nulos. Despois de varias probas decídese escoller para o test de parada $\epsilon = 10^{-3}$, co que obtemos un número grande de iteracións ata a converxencia. Execútase a variante en s-pasos para $s = 1, 2, 4, 6, 8, 10, 12$ e 14. No cadro 6.15 móstrase o número de iteracións no que converxe cada programa executado.

	s=1	**s=2**	**s=4**	**s=6**	**s=8**	**s=10**	**s=12**	**s=14**
Iteracións	15526	8305	3897	2590	1570	1550	1294	1111
$\frac{Iter(s=1)}{Iter(s)}$	1	1.87	3.98	5.99	9.89	10.02	12.00	13.97

Cadro 6.15: Iteracións s-Erro Minimal

No cadro 6.16, correspondente aos tempos de execución para o s-Erro Minimal, prescíndese dos resultados para $s > 8$. Aínda que o número de iteracións continuaba diminuíndo máis aló de $s = 8$, non resultou así cos tempos de execución. Neste cadro pode comprobarse como os tempos nas execucións en paralelo con 2, 4, 8 e 16 fíos decrecen a medida que aumenta o valor de s, obténdose os valores mínimos en todos os casos para $s = 8$.

Tempo	**s=1**	**s=2**	**s=4**	**s=6**	**s=8**
Nth=1	190.22	198.99	188.77	197.04	186.59
Nth=2	157.40	118.91	107.92	107.36	87.14
Nth=4	90.99	83.72	72.28	68.45	55.56
Nth=8	83.64	66.84	56.24	54.32	44.06
Nth=16	83.49	61.53	54.01	49.85	40.24

Cadro 6.16: Tempo de execución en segundos do s-Erro Minimal

As aceleracións poden verse no cadro 6.17. O método orixinal presenta aceleracións moi baixas, non obstante a súa variante en s-pasos chega a duplicala para $s = 8$.

Aceleración	**s=1**	**s=2**	**s=4**	**s=6**	**s=8**
Nth=1	1.00	0.96	1.01	0.97	1.02
Nth=2	1.21	1.60	1.76	1.77	2.18
Nth=4	2.09	2.27	2.63	2.78	3.42
Nth=8	2.27	2.85	3.38	3.50	4.32
Nth=16	2.28	3.09	3.52	3.82	4.73

Cadro 6.17: Aceleración s-Erro Minimal

No gráfico 6.4 pode visualizarse como a curva da aceleración crece máis rápidamente canto maior é o valor de s na variante en s-pasos.

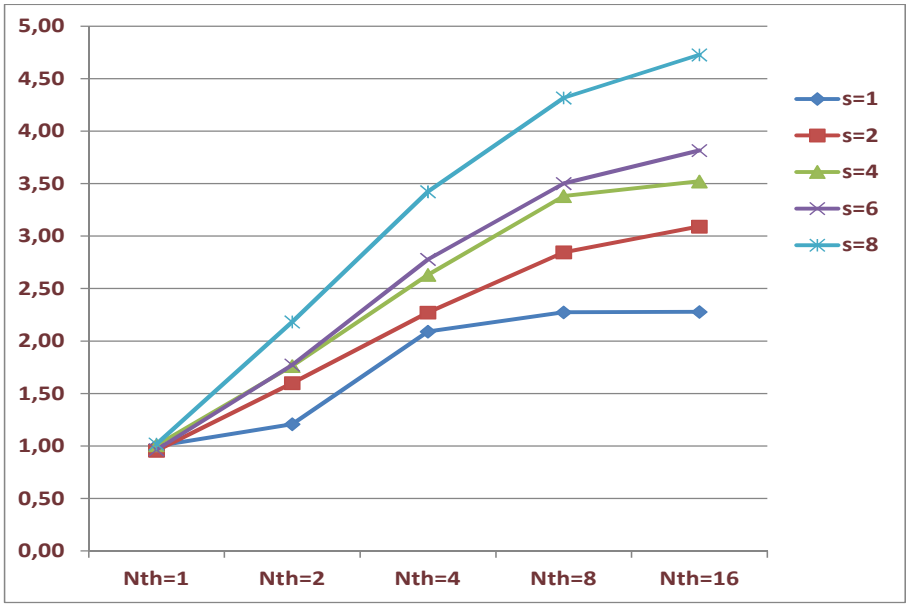

Figura 6.4: Aceleración para s-Erro Minimal.

6.5.5. s-Residuo Minimal

A variante en s-pasos do Residuo Minimal foi proposta en [23]. Aparecen ademais resultados numéricos da súa versión A^tA-ortonormalizada en [29], utilizando unha matriz de coeficientes que provén da discretización, polo método de diferenzas finitas, dun problema de ecuacións en derivadas parciais con valores na fronteira e que os autores se propoñen. Nesta sección preséntanse tamén resultados obtidos con este método en s-pasos na súa versión A^tA-ortonormalizada, neste caso sobre unha matriz exemplo da Matrix Market [61]. O método do Residuo Minimal converxe para matrices cadradas non singulares e non necesariamente simétricas, mais coa súa parte simétrica definida positiva. Esta última característica non aparece na información que acompaña as matrices da Matrix Market. Os resultados que se mostran nesta sección obtivéronse cunha das matrices non simétricas coas que o método orixinal converxía. A matriz en concreto é a matriz fidapm37, que é unha matriz cadrada non simétrica de orde 9152 e con 765944 elementos non nulos.

Executouse o método e a súa variante en s-pasos para os valores de $s = 2, 4, 6, 8$ e 10. O test de parada nesta ocasión é $\epsilon = 5 \times 10^{-8}$. Do mesmo xeito que nas anteriores, o primeiro cadro que se mostra é o 6.18 coas iteracións nas que converxe o método orixinal ($s = 1$) e a súa variante en s-pasos para os correspondentes valores de s.

	s=1	**s=2**	**s=4**	**s=6**	**s=8**	**s=10**
Iteracións	8272	4134	2133	1359	1035	840
$\frac{Iter(s=1)}{Iter(s)}$	1	2	3.88	6.09	7.99	9.85

Cadro 6.18: Iteracións s-Residuo Minimal

No cadro 6.19 móstranse os tempos de execución en cada un dos casos executados. Os tempos de execución son menores en paralelo con 2, 4, 8 e 16 fíos, en todos os casos da variante en s-pasos, excepto para $s = 10$ e 16 fíos. Os valores mínimos danse para $s = 6$, excepto en 16 fíos que se da en $s = 2$.

Tempo	**s=1**	**s=2**	**s=4**	**s=6**	**s=8**	**s=10**
Nth=1	71.19	57.44	73.57	57.70	63.76	62.42
Nth=2	27.28	25.36	23.28	22.68	23.01	24.94
Nth=4	15.09	13.65	13.57	12.94	13.57	14.04
Nth=8	8.81	7.75	7.81	7.62	8.28	8.49
Nth=16	6.61	5.52	5.88	6.13	6.60	7.48

Cadro 6.19: Tempo de execución en segundos do s-Residuo Minimal

Os valores da aceleración poden verse no cadro 6.20. Para 2, 4 e 8 fíos consíguese superaceleración en todos os casos, sendo máis pronunciada na variante en s-pasos.

Aceleración	**s=1**	**s=2**	**s=4**	**s=6**	**s=8**	**s=10**
Nth=1	1	1.24	0.97	1.23	1.12	1,14
Nth=2	2.61	2.81	3.06	3.14	3.09	2.86
Nth=4	4.72	5.22	5.25	5.50	5.24	5.07
Nth=8	8.08	9.19	9.12	9.35	8.60	8.38
Nth=16	10.77	12.90	12.11	11.61	10.79	9.51

Cadro 6.20: Aceleración s-Residuo Minimal

Na gráfica das aceleracións 6.5, na que se representan unicamente os casos $s = 1, 2, 4$ e 6, poden verse as curvas das aceleracións para $s = 2, 4$ e 6 por enriba da curva da aceleración para $s = 1$.

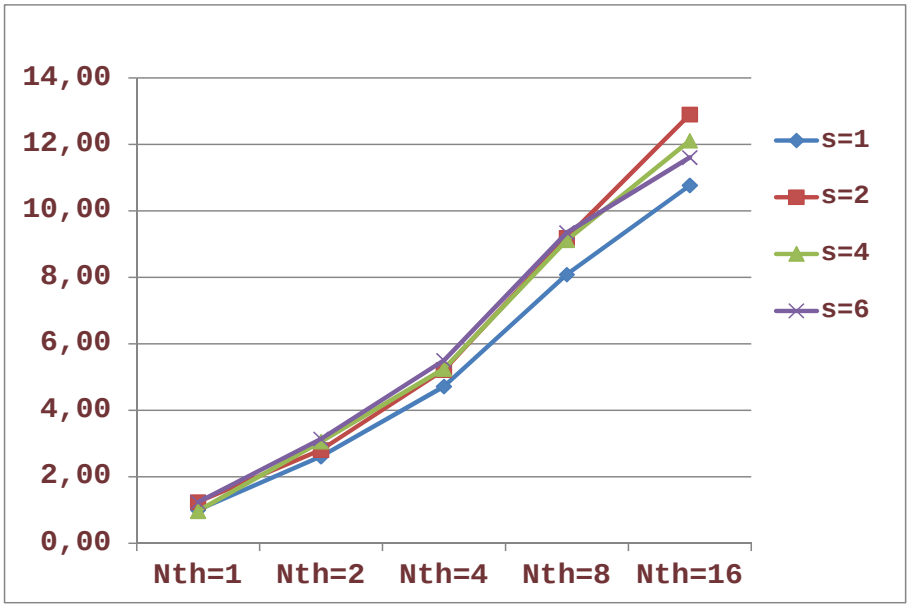

Figura 6.5: Aceleración para s-Residuo Minimal.

6.5.6. s-Orthomin(m) do Residuo Conxugado Xeneralizado

O Residuo Conxugado Xeneralizado require do almacenamento de todos os vectores calculados nas iteracións anteriores, de igual xeito que a súa variante en s-pasos require do almacenamento de todas as matrices dirección calculadas nas iteracións anteriores. Como xa se viu, isto pode provocar problemas de almacenamento en memoria. O método Orthomin(m) fai uso unicamente das m direccións anteriores, no método orixinal, ou das m matrices anteriores, na variante en s-pasos. No apartado anterior presentáronse resultados do s-Residuo Minimal, que é o s-Orthomin(1) do Residuo Conxugado Xeneralizado. Co mesmo problema que se obteñen resultados para o s-Residuo Minimal en [29] preséntanse no mesmo artigo resultados numéricos para o Orthomin(4). Neste apartado móstranse os resultados obtidos co s-Orthomin(m) do Residuo Conxugado Xeneralizado para os valores de $m = 3$ e $m = 5$.

Executáronse os métodos orixinais ($s = 1$) e as súas variantes en s-pasos para os valores de $s = 2, 4, 6, 8$ e 10. O test de parada para o Orthomin(3) é $\epsilon = 5 \times 10^{-7}$ e para o Orthomin(5) é $\epsilon = 10^{-6}$, xa que este último mostrou unha velocidade de converxencia máis lenta. A matriz utilizada en ambos os dous algoritmos foi a mesma que para o s-Residuo Minimal, a matriz fiadpm37.

As iteracións nas que converxe cada algoritmo para os distintos valores de s preséntanse nos cadros 6.21 e 6.22.

	s=1	**s=2**	**s=4**	**s=6**	**s=8**	**s=10**
Iteracións	9153	4738	2002	1268	923	744
$\frac{Iter(s=1)}{Iter(s)}$	1	1.93	4.57	7.22	9.92	12.30

Cadro 6.21: Iteracións s-Orthomin(3)

	s=1	**s=2**	**s=4**	**s=6**	**s=8**	**s=10**
Iteracións	9064	4490	1793	1118	835	628
$\frac{Iter(s=1)}{Iter(s)}$	1	2.02	5.06	8.11	10.86	14.43

Cadro 6.22: Iteracións s-Orthomin(5)

Nos cadros 6.23 e 6.24 preséntanse os tempos de execución para os dous algoritmos. Pode verse nos dous algoritmos que os tempos de execución son máis baixos en todas as variantes en s-pasos que no respectivo método orixinal. Os mínimos valores para Orthomin(3) atópanse para $s = 8$, $s = 6$ e incluso $s = 4$ en 16 fíos, e no Orthomin(5) para $s = 8$ e $s = 6$ en 8 e 16 fíos.

Tempo	**s=1**	**s=2**	**s=4**	**s=6**	**s=8**	**s=10**
Nth=1	87.40	75.98	58.82	55.46	52.45	54.37
Nth=2	37.67	32.72	25.77	24.47	24.30	24.65
Nth=4	20.57	18.37	14.31	13.59	13.30	13.63
Nth=8	12.71	11.32	8.57	8.22	8.38	8.66
Nth=16	9.12	8.78	6.55	6.85	7.24	7.81

Cadro 6.23: Tempo de execución en segundos do s-Orthomin(3)

Tempo	s=1	s=2	s=4	s=6	s=8	s=10
Nth=1	101.93	84.40	60.83	55.18	56.65	53.41
Nth=2	44.29	37.07	26.83	24.93	24.62	24.01
Nth=4	23.48	19.97	14.47	13.07	13.88	12.90
Nth=8	13.53	11.70	8.47	7.84	8.09	8.06
Nth=16	9.63	8.38	6.57	6.00	6.52	6.61

Cadro 6.24: Tempo de execución en segundos do s-Orthomin(5)

Os valores da aceleración para o Orthomin(3) e o Orthomin(5) poden verse nos cadros 6.25 e 6.26 respectivamente. Como pode verse, resultan valores moi altos, habendo superaceleración nos dous algoritmos en case todos os casos para 2, 4 e 8 fíos.

Aceleración	s=1	s=2	s=4	s=6	s=8	s=10
Nth=1	1	1.15	1.49	1.58	1.67	1.61
Nth=2	2.32	2.67	3.39	3.57	3.60	3.55
Nth=4	4.25	4.76	6.11	6.43	6.57	6.41
Nth=8	6.88	7.72	10.20	10.63	10.43	10.09
Nth=16	9.58	9.96	13.34	12.76	12.06	11.20

Cadro 6.25: Aceleración s-Orthomin(3)

Aceleración	s=1	s=2	s=4	s=6	s=8	s=10
Nth=1	1	1.21	1.68	1.85	1.80	1.91
Nth=2	2.30	2.75	3.80	4.09	4.14	4.24
Nth=4	4.34	5.10	7.04	7.80	7.35	7.90
Nth=8	7.53	8.71	12.03	12.99	12.61	12.65
Nth=16	10.59	12.16	15.52	16.98	15.64	15.41

Cadro 6.26: Aceleración s-Orthomin(5)

Nas gráficas das aceleracións 6.6 e 6.7 represéntanse os casos de $s = 1, 2, 4$ e 6. Nelas evidéncianse aceleracións máis pronunciadas para as variantes en s-pasos nos valores $s = 2, 4, 6$.

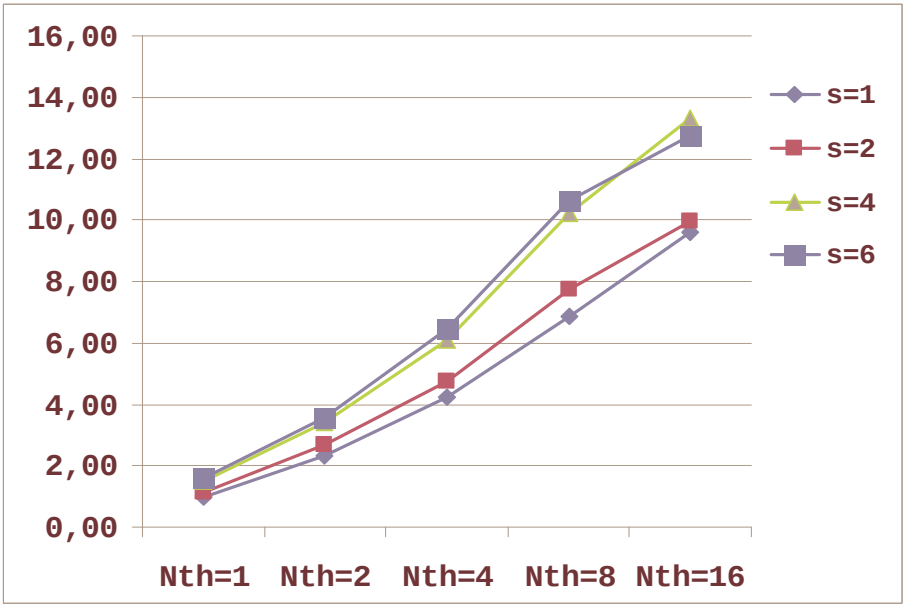

Figura 6.6: Aceleración para s-Orthomin(3).

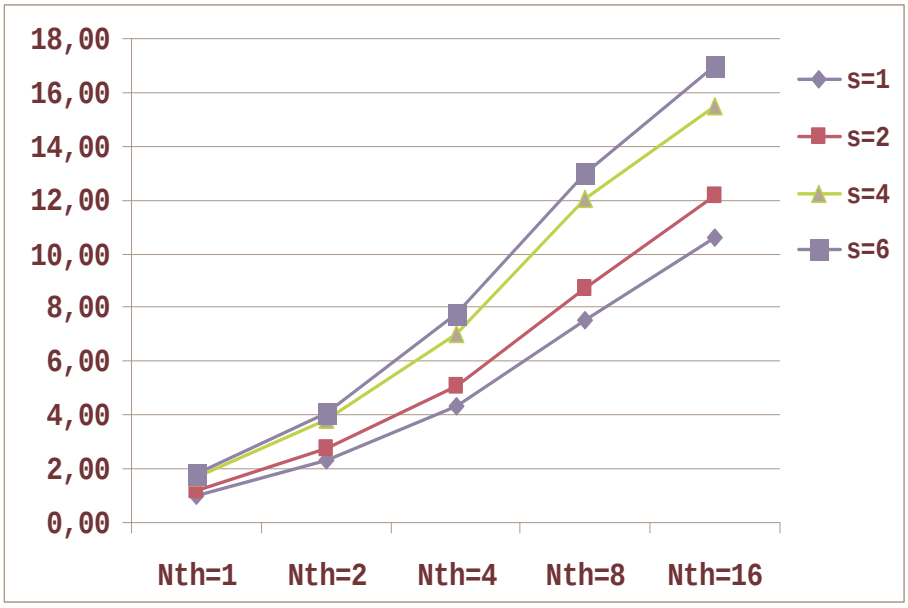

Figura 6.7: Aceleración para s-Orthomin(5).

6.5.7. s-Dobre Serie Ortogonal

A variante en s-pasos do método da Dobre Serie Ortogonal é unha das aportacións deste traballo. O método orixinal, proposto en [5], converxe na teoría para calquera matriz cadrada non singular. Non obstante, nas probas realizadas, com-

probouse que non tiña moi boa converxencia coa maioría das matrices coas que se estaba traballando. Finalmente, escolleuse a matriz de proba UTM5940, cadrada, non simétrica, de orde 5940 e 83842 elementos non nulos.

A variante en s-pasos execútase para $s = 2, 3, 4, 5$ e 6 pois para $s > 6$ o número de iteracións no que converxe comenza a aumentar. O test de parada utilizado é $\epsilon = 5 \times 10^{-4}$. No cadro 6.27 pode verse o número de iteracións no que converxe cada unha das execucións realizadas. Neste caso sorprende o factor tan alto de diminución do número de iteracións para $s = 5$ e $s = 6$.

	s=1	**s=2**	**s=3**	**s=4**	**s=5**	**s=6**
Iteracións	2263	1138	752	571	339	275
$\frac{Iter(s=1)}{Iter(s)}$	1	1.99	3.01	3.96	6.68	8.23

Cadro 6.27: Iteracións s-Dobre Serie Ortogonal

Como pode verse no cadro 6.28, os tempos de execución da variante en s-pasos están por debaixo dos do método orixinal en todos os casos, incluso en secuencial ($Nth = 1$). Os valores mínimos obtéñense en todos os casos para $s = 6$.

Tempo	**s=1**	**s=2**	**s=3**	**s=4**	**s=5**	**s=6**
Nth=1	9.12	7.61	7.06	6.98	5.15	5.10
Nth=2	4.91	4.31	4.32	4.34	3.25	2.91
Nth=4	2.82	2.52	2.58	2.68	1.93	1.74
Nth=8	1.89	1.65	1.64	1.68	1.21	1,15
Nth=16	1.76	1.43	1.55	1.43	1.15	1.07

Cadro 6.28: Tempos de execución s-Dobre Serie Ortogonal

O cadro 6.29 mostra as aceleracións, que se manteñen na variante en s-pasos con valores por enriba do método orixinal, atopándose os máximos para $s = 6$.

Aceleración	s=1	s=2	s=3	s=4	s=5	s=6
Nth=1	1	1.20	1.29	1.31	1.77	1.79
Nth=2	1.86	2.12	2.11	2.10	2.81	3.13
Nth=4	3.23	3.61	3.54	3.41	4.73	5.24
Nth=8	4.83	5.51	5.56	5.44	7.56	7.92
Nth=16	5.19	6.37	5.87	6.40	7.96	8.52

Cadro 6.29: Aceleración s-Dobre Serie Ortogonal

Xa que hai menos valores de s, esta vez represéntanse todos os casos na gráfica de aceleracións 6.8. Nela poden verse todas as curvas, correspondentes aos distintos valores de s, da aceleración da variante en s-pasos por enriba da curva de aceleración do método orixinal. Distínguense os casos $s = 5$ e $s = 6$ con aceleracións algo máis elevadas co resto.

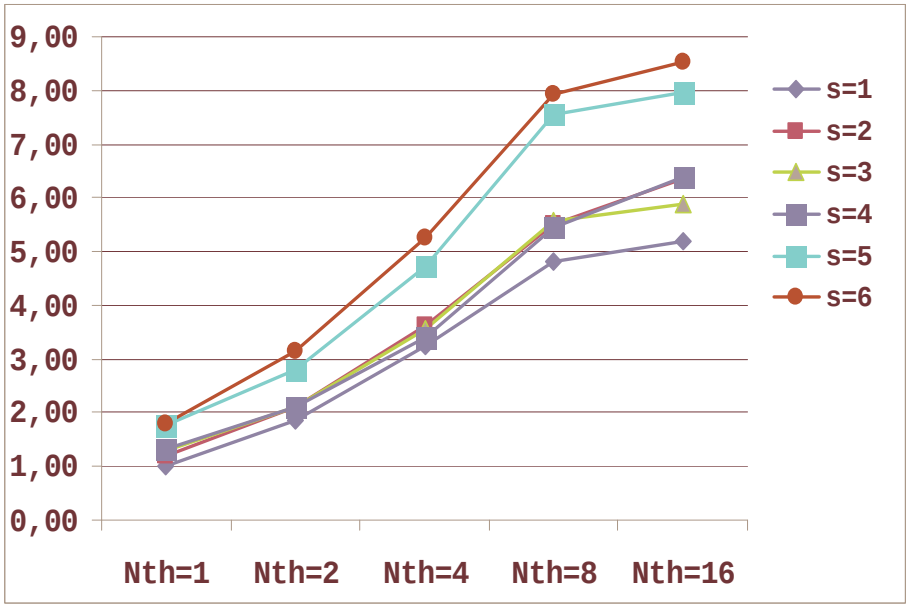

Figura 6.8: Aceleración para s-Dobre Serie Ortogonal.

6.5.8. s-Gradente Biconxugado

É coñecido que o método do Gradente Biconxugado pode dexenerar. Aparte deste problema, deuse o problema engadido da falla de estabilidade en moitos casos da variante en s-pasos. Hai que lembrar que nesta variante non se puideron

N-ortonormalizar as direccións calculadas en cada iteracción. Esta falla de estabilidade, cuestión que debería ser abordada nun futuro, provocou que chegara a suceder situacións como que o mesmo programa (mesma variante co mesmo valor de s) converxera ou non dependendo do número de fíos no que se executara. Non obstante, é de esperar que este problema se poida mitigar coa utilización de aritmética en 128 ou 256 bits.

Entre as matrices probadas con este método escóllese a matriz fidapm37, cadrada, non simétrica, de orde 9152 e 765944 elementos non nulos. Executase o método para $s = 1, 2, 4, 6$. O test de parada utilizado é $\epsilon = 5 \times 10^{-4}$. No cadro 6.30 móstrase o número de iteracións en que converxe cada programa executado.

	s=1	**s=2**	**s=4**	**s=6**
Iteracións	277	140	94	49
$\frac{Iter(s=1)}{Iter(s)}$	1	1.98	2.95	5.65

Cadro 6.30: Iteracións s-BiCG

Mostránse os tempos de execución no cadro 6.31. Dado que o número de iteracións no que converxe o método non é moi alto, tampouco o é o tempo de execución, polo que a ganancia en paralelo de tempo de execución non é moita co respecto ao orixinal en secuencial. Aínda así a variante en s-pasos mellora estes tempos en todos os casos, obténdose o valor mínimo en paralelo para $s = 6$.

Tempo	**s=1**	**s=2**	**s=4**	**s=6**
Nth=1	3.23	2.87	2.71	2.72
Nth=2	1.42	1.22	1.18	1.08
Nth=4	0.74	0.65	0.64	0.53
Nth=8	0.38	0.35	0.34	0.31
Nth=16	0.41	0.41	0.37	0.28

Cadro 6.31: Tempo de execución en segundos do s-BiCG

Aínda que os tempos de execución aparentemente no ofrecen moita ganancia, no cadro de aceleracións 6.32 preséntanse aceleracións elevadas nalgúns casos, por exemplo para 4 e 8 fíos hai superaceleración en todos os casos. Os valores da ace-

leración da variante en s-pasos son superiores en todos os casos que os do método orixinal, chegándose á máxima para $s = 6$.

Aceleración	**s=1**	**s=2**	**s=4**	**s=6**
Nth=1	1.00	1.12	1.19	1.19
Nth=2	2.26	2.64	2.74	2.98
Nth=4	4.38	4.94	5.06	6.08
Nth=8	8.43	9.33	9.45	10.49
Nth=16	7.94	7.86	8.82	11.69

Cadro 6.32: Aceleración s-BiCG

Na gráfica 6.9 das curvas de aceleración pode comprobarse como as correspondentes á variante en s-pasos están por enriba da correspondente ao método orixinal. Non obstante, nesta gráfica particular, resalta o feito de que as curvas de aceleración decrecen de 8 a 16 fíos excepto no caso $s = 6$.

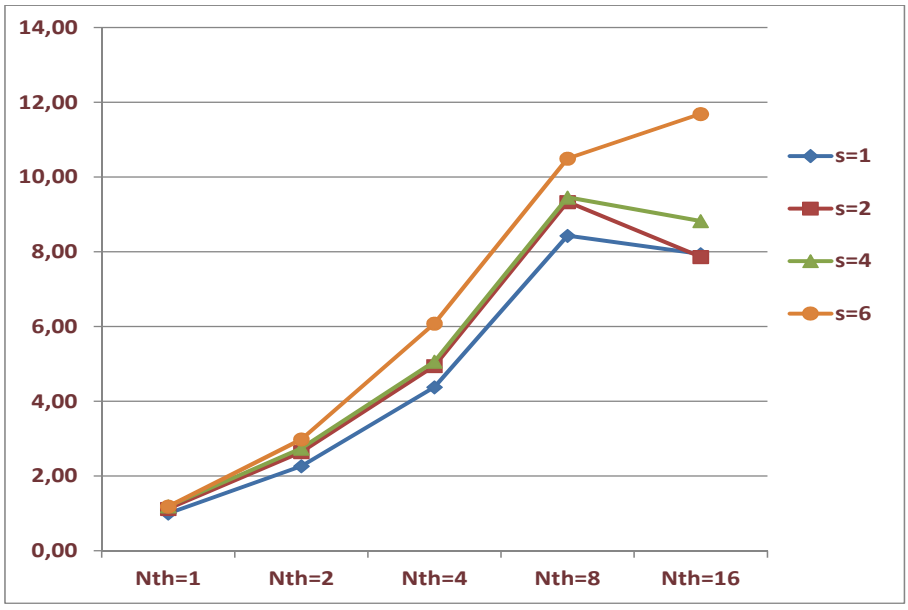

Figura 6.9: Aceleración para s-BiCG.

Conclusións e futuras liñas de traballo

Nesta última parte presentamos as conclusións do traballo realizado, centrándonos nas aportacións orixinais incluídas no mesmo. O obxectivo último dos métodos en s-pasos é gañar eficiencia na implementación en arquitecturas paralelas respecto dos métodos orixinais, isto é, unha mellor escalabilidade na implementación paralela, que se traduce nunha mellora dos tempos de execución cando se aumenta o número de fíos ou procesos cos que é executado o método.

Os comenzos deste traballo xurdiron da necesidade de facer máis eficientes para a programación paralela os métodos iterativos de resolución de grandes sistemas lineares. A idea era tratar de converter as operacións vector vector ou matriz vector, que había en cada iteración destes métodos, en operacións matriz vector e matriz matriz respectivamente, co fin de reducir o número de comunicacións entre procesos ou fíos e accesos á memoria con respecto ao número de operacións. Este foi precisamente o logro das variantes en s-pasos propostas por Chronopoulos en artigos anteriores, [23] e [24] basicamente. Por outra banda, coñecíamos o Algoritmo Xeral de Ortogonalización (AXO) publicado en [51] e que supón unha xeneralización dalgúns destes métodos, os do tipo Gradente Conxugado, de xeito que, coa substitución de dúas matrices parámetro neste algoritmo, poden obterse como casos particulares métodos coñecidos como o Gradente Conxugado, o do Residuo Conxugado, o do Erro Minimal, etc. O noso primeiro obxectivo foi obter unha variante en s-pasos deste Algoritmo Xeral de Ortogonalización.

Unha das principais aportacións desta tese é a variante en s-pasos do Algoritmo Xeral de Ortogonalización, que denotamos por s-AXO, proposta no capítulo 2 e que supón unha xeneralización dos métodos en s-pasos xa coñecidos e doutros

non coñecidos ata o momento. Inspirándonos nas demostracións das propiedades do AXO desenvoltas en [51] e [4], demostramos no mesmo capítulo propiedades análogas para o s-AXO. Estas derivan en dous resultados importantes, un de converxencia e outro de estimación do erro. Cando unha das matrices parámetro, a denotada por K, é simétrica, a versión correspondente evita ter que almacenar en memoria todas as matrices cos vectores dirección que se calculan en cada iteración. Pola contra, se a matriz K non é simétrica, o almacenamento de todas estas matrices pode evitarse coa versión Orthomin(m) deste método. Propoñemos entón a versión Orthomin(m) do s-AXO, que denotamos por s-Orthomin(m), e demostramos tamén un teorema de converxencia para o mesmo. Finaliza o capítulo cunha sección onde propoñemos unha modificación do s-AXO e do s-Orthomin(m), ortonormalizando respecto da matriz N (que depende dunha das matrices parámetro) os vectores dirección en cada iteración do método. A introdución desta modificación vén motivada pola procura dunha mellora na estabilidade dos métodos cando $s > 5$ e inspírase en [29], onde se utiliza esta técnica no Residuo Conxugado Xeneralizado e o Orthomin(m) con este obxectivo. Como resultado obtemos novas versións dos métodos s-AXO e s-Orthomin(m) que serán ademais as utilizadas para obter as variantes particulares en s-pasos.

No capítulo 3 substituímos as matrices parámetro H e K por valores concretos nos métodos s-AXO e s-Orthomin de xeito que imos obtendo as diferentes variantes en s-pasos de métodos particulares. Aínda que a maioría dos métodos en s-pasos expostos neste capítulo foron xa propostos en traballos anteriores, presentamos estes na súa versión N-ortonormalizada, o cal vén sendo unha aportación nesta tese, excepto para a variante en s-pasos do Residuo Conxugado Xeneralizado que xa aparece en [29]. Entre os métodos en s-pasos obtidos, hai un método non proposto anteriormente por outros autores, a variante en s-pasos do Erro Minimal, e que denotamos por s-Erro Minimal.

Existe unha segunda forma do Algoritmo Xeral de Ortogonalización dada en [51]. Desta segunda forma pode obterse o método da Dobre Serie Ortogonal proposto por Amara e Nedelec en [5]. No capítulo 4 propoñemos unha variante en s-pasos da segunda forma do AXO e demostramos que é equivalente á forma orixinal do s-AXO. Na segunda parte deste capítulo obtemos, a partir desta segunda forma do s-AXO, a variante en s-pasos do método da Dobre Serie Ortogonal. Propoñemos tamén unha

versión coa ortonormalización dos vectores dirección calculados en cada iteración.

No capítulo 5 propoñemos o último método en s-pasos desta memoria, a variante en s-pasos do Gradente Biconxugado (s-BiCG). É sabido que o Gradente Biconxugado pode dexenerar, se ben existe un teorema de converxencia no caso en que non dexenere. Neste capítulo conseguimos demostrar algún lema de ortogonalidade, mais non un resultado xeral de converxencia, quedando isto como un obxectivo nas liñas futuras de investigación que se propoñen.

Por último, no capítulo 6, mostramos os resultados numéricos obtidos na programación paralela dos métodos en s-pasos propostos nos capítulos anteriores. Executamos estes programas na computadora Finis Terrae do CESGA e utilizamos como matrices exemplo matrices da Matrix Market Collection. Os resultados obtidos corroboran unha mellora da eficiencia na implementación paralela dos métodos en s-pasos con respecto aos métodos orixinais.

A obtención de variantes en s-pasos doutros métodos iterativos podería ter interese. En particular as dos métodos derivados do Gradente Biconxugado, como o Gradente Conxugado Cadrado, o BiCGSTAB ou o TFQMR. Este traballo podería intentarse abordar nun futuro se se consegue resolver a dificultade que presenta a imposibilidade de poñer as matrices dos vectores dirección como polinomios en A polos vectores dirección iniciais.

No transcurso do traballo experimental atopamos problemas de estabilidade na execución de determinados métodos, a pesar da ortonormalización dos vectores dirección en cada iteración. Cremos que esta circunstancia é debida a que a velocidade de converxencia dalgúns dos métodos dependen do condicionamento de AA^t. Faise entón conveniente a utilización de precondicionadores para obter unha boa converxencia. Nesta tese non abordamos este aspecto, pois o obxectivo último é o de propoñer métodos iterativos en s-pasos e estudar a súa mellor eficiencia na implementación paralela, independentemente da velocidade de converxencia dos métodos orixinais. O estudo do precondicionamento destes métodos pode ser un aspecto abordable no futuro.

Bibliografía

[1] F. Almeida, D. Giménez, J.M. Mantas e A.M. Vidal: *Introducción a la programación paralela*, Paraninfo (2008).

[2] J.A. Alvarez Dios, J.C. Cabaleiro e G. Casal: *A s-step variant of the double orthogonal series algorithm*, Numerical Mathematics and Advanced Applications. ENUMATH 2005, Springer-Verlag (2006) 937-944.

[3] J.A. Alvarez Dios, J.C. Cabaleiro e G. Casal: *A generalization of s-step variants of Gradient Methods*, Journal of Computational and Applied Mathematics. (En revisión).

[4] J.A. Alvarez Dios e M. Calaza: *Notas sobre análisis numérico de grandes sistemas*, Publicacións Docentes do Departamento de Matemática Aplicada. Universidad de Santiago de Compostela (2005).

[5] M. Amara e J.C. Nedelec: *Résolution de systeme matriciel indéfini par une décomposition sur une double suite orthogonale*, C. R. Acad. Sc. Paris, t. 295 (1982) 309-312.

[6] O. Axelsson: *Conjugate gradient type methods for unsymmetric and inconsistent systems of equations*, Linear Algebra and its Applications, 29 (1980) 1-16.

[7] O. Axelsson: *Iterative Solution Methods*, Cambridge University Press. Cambridge (1994).

[8] O. Axelsson e I. Gustafsson: *A modified upwind scheme for convective transport equations and the use of a conjugate gradient method for the solution of non-symmetric systems of equations*, J. Inst. Math. Appl., 23 (1979) 321-337.

[9] G. Ballard, J. Demmel, O. Holtz e O. Schwartz: *Minimizing Communication in Numerical Linear Algebra*, EECS Department University of California, Berkeley. (2011) Technical Report No. UCB/EECS-2011-15.

[10] R. Barrett, M. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine e H. Van der Vorst: *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM (1994).

[11] BeBOP Sparse Matrix Converter home page.
http://bebop.cs.berkeley.edu/smc/.

[12] V. Blanco Pérez: *Análisis, predicción y visualización del rendimiento de métodos iterativos en HPF y MPI*, Tesis doctoral. Departamento de Electrónica y Computación. USC (2002).

[13] V. Blanco, J.C. Cabaleiro, P. Gonzalez, D.B. Heras, T.F. Pena, J.J. Pombo e F.F. Rivera: *HPI (HPF and MPI hybrid) implementation of Paraiso*, Technical Report No. GAC2000-i11, Dept. of Electronics and Computer Science, University Santiago de Compostela (2000).

[14] V. Blanco, J.C. Cabaleiro, P. Gonzalez, D.B. Heras, T.F. Pena, J.J. Pombo e F.F. Rivera: *Performance of Parallel Iterative Solvers: a Library, a Prediction Model, and a Visualization Tool*, Journal of Information Science and Engenering 18 (2002) 763-785.

[15] H. Bücker e M. Sauren: *A variant of the biconjugate gradient method suitable for massively parallel computing*, Solving Irregularly Structured Problems in Parallel. Lecture Notes in Computer Science 1253 (1997) 72-79.

[16] L. Carpintero Arias: *Métodos de Krylov para un esquema implícito de tipo Lax-Wendroff. Aplicaciones en aerodinámica estacionaria*, Tesis doctoral. Departamento de Matemática Aplicada. USC (1996).

[17] CESGA home page.
http://www.cesga.es/.

[18] P.G. Ciarlet: *Introducción á análise numérica matricial e á optimización*, Servicio de Publicacións da Universidade de Santiago de Compostela (1999).

[19] P. Concus e G.H. Golub: *A Generalized Conjugate Gradient Method for Nonsymetric Systems of Linear Equations*, Lecture Notes in Economics and Math. Systems, Springer-Verlag 134 (1976) 56-65.

[20] R. Chandra, R. Menon, L. Dagum, D. Kohr, D. Maydan e J. McDonald: *Parallel Programming in OpenMP*, Morgan Kaufmann Publishers. Academic Press (2001).

[21] B. Chapman, G. Jost e R. van der Pas: *Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation)*, The MIT Press (2007).

[22] A.T. Chronopoulos: *A class of parallel iterative methods implemented on multiprocessors,* PhD Thesis. University of Illinois at Urbana-Champaing (1987).

[23] A.T. Chronopoulos: *s-step iterative methods for (non)symmetric (in)definite linear systems*, Siam J. Numer. Anal. Vol. 28, No. 6 (1991) 1776-1789.

[24] A.T. Chronopoulos e C.W. Gear: *s-step iterative methods for symmetric linear systems*, Journal of Computational and Applied Mathematics 25 (1989) 153-168.

[25] A.T. Chronopoulos e C.W. Gear: *On the efficient implementation of preconditioned s-step conjugate gradient methods on multiprocessors with memory hierarchy*, Parallel Computing 11 (1989) 37-53.

[26] A.T. Chronopoulos e D. Kincaid: *On the Odir iterative method for nonsymmetric indefinite linear systems*, Numer. Linear Algebra Appl. 8 (2001) 71-82.

[27] A.T. Chronopoulos e A.B. Kucherov: *A Parallel Krylov-Type Method for Nonsymmetric Linear Systems*, Lect. notes comput. sci. High performance computing. International conference No. 8, Hyderabad, Springer-Verlag vol. 2228 (2001) 104-114.

[28] A.T. Chronopoulos e A. Kucherov: *Block S-step Krylov Iterative Methods*, Numer. Linear Algebra Appl. 17 (1) (2010) 3-15.

[29] A.T. Chronopoulos e C.D. Swanson: *Parallel iterative s-step methods for unsymmetric linear systems*, Parallel Computing 22 (1996) 623-641.

[30] J.W. Demmel: *Applied Numerical Linear Algebra*, SIAM (1997).

[31] J. Demmel, M. Hoemmen, M. Mohiyuddin e K. Yelick: *Communication-optimal iterative methods*, SciDAC 2009, Journal of Physics Conference Series 180 (2009) Article Number: UNSP 012040.

[32] J.J. Dongarra, I.S. Duff, D.C. Sorensen e H.A. van der Vorst: *Numerical Linear Algebra for High-Performance Computers*, SIAM (1998).

[33] I.S. Duff, R.G. Grimes, e J.G. Lewis: *Users guide for the Harwell-Boeing sparse matrix collection*, Technical Report No. TR-PA-92-96, CERFACS (1992).

[34] I.S. Duff e H.A. van der Vorst: *Developments and trends in the parallel solution of linear systems*, Parallel Computing 25, Issue 13-14 (1999).

[35] I.S. Duff, M.A. Heroux, D.Kincaid e R. Pozo: *An overview of the sparse basic linear algebra subprograms: The new standard from the BLAS technical forum*, ACM Trans. on Math. Soft. Vol. 28, No. 2 (2002) 239-267.

[36] S.C. Eisenstat, H.C. Elman, e M.H. Schultz: *Variational iterative methods for nonsymetric systems of linear equations*, Siam J. Numer. Anal. Vol. 20, No. 2 (1983) 345-357.

[37] T. El-Ghazawi, W. Carlson, T. Sterling e K. Yelick : *UPC: Distributed Shared Memory Programming*, Wiley (2005).

[38] G. Em Karniadakis e R.M. Kirby II: *Parallel Scientific Computing in C++ and MPI: A Seamless Approach to Parallel Algorithms and their Implementation*, Cambridge University Press (2003).

[39] R. Fletcher: *Conjugate Gradient Methods for Indefinte Systems*, Lecture Notes in Math., Springer-Verlag, 506 (1976) 73-89.

[40] F. Gebali: *Algorithms and Parallel Computing (Wiley Series on Parallel and Distributed Computing)*, Wiley (2011).

[41] A. Geist, A. Beguelin, J. Dongarra, R. Manchek, W. Jiang e V. Sunderam: *PVM: A Users' Guide and Tutorial for Networked Parallel Computing*, MIT Press (1994).

[42] P. González, J.C. Cabaleiro e T.F. Pena: *Parallel Sparse Approximate Preconditioners for the solution of Large Dense Linear Systems*, World Scientific and Engineering Society Press (2000) 50-56.

[43] P. González, J.C. Cabaleiro e T.F. Pena: *Iterative solution of large linear systems with non-smooth submatrices using partial wavelet transforms and split-matrix matrix-vector multiplication*, Int. Journal for Numerical Methods in Engineering, 59 (2004) 457-473.

[44] Guangye Li: *A block variant of the GMRES method on massively parallel processors*, Parallel Computing 23 (1997) 1005-1019.

[45] W.A. Hackbush: *A parallel variant of the conjugate gradient method*, Numer. Linear Algebra Appl. 1 (1992) 133-147.

[46] G. Hager e G. Wellein: *Introduction to High Performance Computing for Scientists and Engineers*, Chapman and Hall/CRC Computational Science. CRC Press (2010).

[47] J.L. Hennessy e D.A. Patterson: *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann (2006).

[48] M.R. Hestenes e E. Stiefel: *Methods of conjugate gradients for solving linear systems*, Jour. Res. Nat. Bur. Standards 49 (1952) 409-436.

[49] Z. Jia: *The convergence of Krylov subspace methods for large unsymmetric linear systems*, Numer. Linear Algebra Appl. 3 (6) (1996) 491-512.

[50] P. Joly: *Présentation de synthèse des méthodes de gradient conjugué*, RAIRO. Modélisation mathématique et analyse numérique 20, nº4 (1988) 639-665.

[51] P. Joly: *Méthodes de gradient conjugué*, Cours du 3ème cycle 1988. Université de Saint-Jacques de Compostelle. Publications du laboratoire d'analyse numérique. Université Pierre et Marie Curie. (1988).

[52] D.S. Kershaw: *The incomplete Cholesky-conjugate gradient method for the iterative solution of systems of linear equations*, J. Computational Physics 26 (1978) 43-65.

[53] J.T. King: *A minimal error conjugate gradient method for ill-posed problems*, J. Optim. Theory Appl. 60 (1989) 297-304.

[54] C.H. Koelbel, D.V. Lobeman, R.S. Schreiber, G.L. Steele e M.E. Zoesel: *The High Performance Fortran Handbook*, MIT Press (1993).

[55] A. Kossjanskij: *Parallel Block-Iterative Algorithms for the Solution of Linear Systems,* PhD Thesis, Institute for Scientific Computing, RWTH Aachen University, (2004).

[56] V. Kumar, A. Grama, A. Gupta e G. Karypis: *Introduction to parallel computing: design and analysis of parallel algorithms*, The Benjamin/Cummings Publishing Company, Inc. (1994).

[57] C. Lawson, R. Hanson, D.Kincaid e F. Krogh: *Basic Linear Algebra Subprograms for Fortran Usage*, ACM Trans. on Math. Soft. 5 (1979) 308-325.

[58] P. O'Leary e P. Whitman: *Parallel QR factorization by Householder and modified Gram-Schmidt algorithms*, Parallel Computing 16 (1990) 99-112.

[59] S. Liu, Y. Zhang, X. Sun e R. Qiu: *Performance Evaluation of Multithreaded Sparse Matrix-Vector Multiplication Using OpenMP*, 2009 11th IEEE International Conference on High Performance Computing and Communications (2009) 659-665.

[60] D.G Luenberger: *The Conjugate Residual Method for constrained minimization problems*, S.I.A.M. Journal Numerical Analysis 7 (1970) 390-398.

[61] Matrix Market home page.
http://math.nist.gov/MatrixMarket/.

[62] MPI home page.
http://www.mcs.anl.gov/research/projects/mpi/.

[63] OpenMP home page.
http://openmp.org/wp/.

[64] J.J. Ortega: *Introduction to Parallel and Vector Solution of Linear Systems*, Plenum Press (1988).

[65] C.C. Page e M.A. Saunders: *Solutions of sparse indefinite systems of linear equations*, S.I.A.M. Journal Numerical Analysis 12 (1975) 617-629.

[66] Peter Pacheco: *Parallel programing with MPI*, Morgan Kaufmann (1997).

[67] Peter Pacheco: *An Introduction to Parallel Programming*, Morgan Kaufmann (2011).

[68] PBLAS home page.
http://www.netlib.org/scalapack/pblas_qref.html.

[69] G. Radicati di Brozolo e Y. Robert: *Parallel conjugate gradient-like algorithms for sparse nonsymmetric systems on a vector multiprocessor*, Parallel Computing 11(2) (1989) 223-239.

[70] T. Rauber e G. Rünger: *Parallel Programming: for Multicore and Cluster Systems*, Springer (2010).

[71] J.K. Reid: *On the Method of Conjugate Gradient for the Solution of Large Sparse Systems of Linear Equations, Large Sparse Sets of Linear Equations*, Reid, éd., Academic Press (1971) 231-254.

[72] Yousef Saad: *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company (1996).

[73] Y. Saad, H.A. van der Vorst: *Iterative solution of linear systems in the 20th-century*, J. Comput. Appl. Math. 123 (2000) 1-33.

[74] J. Sanders e E. Kandrot: *CUDA by Example: An Introduction to General-Purpose GPU Programming*, Addison-Wesley Professional (2010).

[75] V. Simoncini e D.B. Syld: *Recent computational developments in Krylov subspace methods for linear systems*, Numer. Linear Algebra Appl. 14 (2007) 1-59.

[76] P. Sonneveld: *CGS, a fast Lanczos-type solver for nonsymmetric linear systems.*, S.I.A.M. Journal on Scientific and Statistical Computing 10 (1) (1989) 36-52.

[77] F. Toutounian: *The stable ATA-orthogonal s-step Orthomin(k) algorithm with the CADNA library.*, Numerical Algorithms. Springer Netherlands. Vol. 17, No. 1-2 (1998) 105-119.

[78] H.A. van der Vorst: *BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymetric linear systems*, SIAM Journal on Scientific and Statistical Computing 12 (1992) 631-644.

[79] H.A. van der Vorst: *Iterative Methods for Large Linear Systems*, Cambridge University Press. Cambridge (2003).

[80] P.K.W. Vinsome: *Orthomin, an Iterative Method for Solving Sparse Sets of Simultaneous Linear Equations*, Proceedings Fourth Symposium on Reservoir Simulation, SPE of AIME, Los Angeles (1976) 149-159.