LEONARDO SILVEIRA BORGES

MÉTODOS PARA PROBLEMAS MAL-POSTOS DISCRETOS DE GRANDE PORTE

CAMPINAS
2013

UNIVERSIDADE ESTADUAL DE CAMPINAS

INSTITUTO DE MATEMÁTICA, ESTATÍSTICA
E COMPUTAÇÃO CIENTÍFICA

LEONARDO SILVEIRA BORGES

MÉTODOS PARA PROBLEMAS MAL-POSTOS DISCRETOS DE GRANDE PORTE

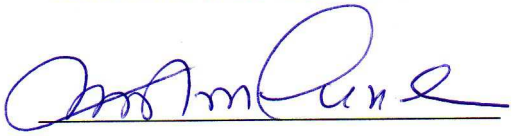
Orientadora: Prof.ª Dr.ª Maria Cristina de Castro Cunha

Coorientador: Prof. Dr. Fermín Sinforiano Viloche Bazán

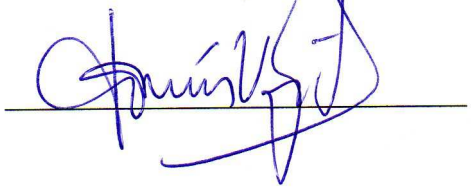
Tese de doutorado apresentada ao Instituto de Matemática, Estatística e Computação Científica da UNICAMP para obtenção do Título de Doutor em Matemática Aplicada.

ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DA TESE
DEFENDIDA PELO ALUNO LEONARDO SILVEIRA BORGES,
E ORIENTADA PELA PROF.A. DR.A. MARIA CRISTINA DE CASTRO CUNHA.

Assinatura do Orientador



Assinatura do Coorientador



CAMPINAS
2013

FICHA CATALOGRÁFICA ELABORADA POR
ANA REGINA MACHADO - CRB8/5467
BIBLIOTECA DO INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E
COMPUTAÇÃO CIENTÍFICA - UNICAMP

Borges, Leonardo Silveira, 1983-
B644m Métodos para problemas mal-postos discretos de grande porte / Leonardo Silveira Borges. – Campinas, SP : [s.n.], 2013.

Orientador: Maria Cristina de Castro Cunha.
Coorientador: Fermín Sinforiano Viloche Bazán.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1. Tikhonov, A. N. (Andrei Nikolaevich), 1906-1993. 2. Métodos iterativos (Matemática). 3. Análise numérica. I. Cunha, Maria Cristina de Castro,1945-. II. Viloche Bazán, Fermín Sinforiano. III. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. IV. Título.

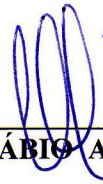
Informações para Biblioteca Digital

Título em inglês: Methods for large-scale discrete ill-posed problems
Palavras-chave em inglês:
Tikhonov, A. N. (Andrei Nikolaevich), 1906-1993
Iterative methods (Mathematics)
Numerical analysis
Área de concentração: Matemática Aplicada
Titulação: Doutor em Matemática Aplicada
Banca examinadora:
Fermín Sinforiano Viloche Bazán [Coorientador]
Fábio Antonio Dorini
Luiz Mariano Paes de Carvalho Filho
Márcia Aparecida Gomes Ruggiero
Eduardo Cardoso de Abreu
Data de defesa: 07-02-2013
Programa de Pós-Graduação: Matemática Aplicada

Tese de Doutorado defendida em 07 de fevereiro de 2013 e aprovada

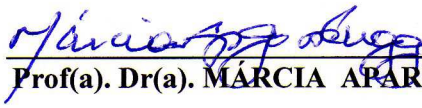
Pela Banca Examinadora composta pelos Profs. Drs.

Prof(a). Dr(a). FERMIN SINFORIANO VILOCHE BAZÁN



Prof(a). Dr(a). FÁBIO ANTONIO DORINI

Prof(a). Dr(a). LUIZ MARIANO PAES DE CARVALHO FILHO



Prof(a). Dr(a). MÁRCIA APARECIDA GOMES RUGGIERO

Prof(a). Dr(a). EDUARDO CARDOSO DE ABREU

Agradecimentos

Aos professores Maria Cristina e Fermín, pela orientação. Todas as conversas, dicas e sugestões foram fundamentais para a realização desta pesquisa e meu aperfeiçoamento profissional.

À FAPESP[1], pelo apoio financeiro e suporte necessário para a viabilização desta pesquisa.

À UNICAMP e ao IMECC, pela infraestrutura oferecida.

Aos demais colegas, professores e familiares que me acompanharam durante mais esta etapa.

E por fim, especialmente à minha esposa Maria Alice, por todo o apoio e incentivo dedicados.

[1]Processo Nº 2009/52193-1.

Resumo

A resolução estável de problemas mal-postos discretos requer o uso de métodos de regularização. Dentre vários métodos de regularização existentes na literatura, um dos mais utilizados é o método de regularização de Tikhonov, cuja eficiência depende da escolha do parâmetro de regularização. Existem vários métodos para selecionar um parâmetro apropriado tais como o princípio da discrepância de Morozov e métodos heurísticos como o critério da curva-L de Hansen, a Validação Cruzada Generalizada de Golub, Heath e Wahba e o método de ponto fixo de Bazán. Problemas mal-postos discretos de grande porte podem ser resolvidos por métodos iterativos como CGLS e LSQR desde que as iterações sejam interrompidas antes que a influência do ruído deteriore a qualidade das iteradas. Esta é uma tarefa difícil que ainda não foi abordada satisfatoriamente na literatura. Em uma tentativa de atenuar a dificuldade na escolha da iteração de parada, tais métodos podem ser combinados com o método de regularização de Tikhonov gerando os métodos híbridos como GKB-FP e W-GCV (ambos usam a matriz identidade como matriz de regularização).

As contribuições desta tese incluem primeiramente novas informações referentes ao algoritmo GKB-FP e como este pode ser eficientemente implementado para o método de regularização de Tikhonov com a matriz de regularização sendo diferente da matriz identidade. Como segunda contribuição tem-se o desenvolvimento de um critério de parada automático para métodos iterativos para problemas "de grande porte", incluindo meios para incorporar informações a priori da solução (como regularidade, por exemplo) no processo iterativo.

O método de regularização de Tikhonov usualmente está confinado apenas a um único parâmetro. Entretanto, alguns problemas apresentam soluções com distintas características que devem ser incorporadas na solução regularizada. Isso conduz ao método de regularização de Tikhonov com múltiplos parâmetros. A terceira contribuição desta tese é o desenvolvi-

mento de um método baseado em iterações de ponto fixo para a seleção destes parâmetros e um algoritmo do tipo GKB-FP para problemas de grande porte.

Por fim, os resultados teóricos obtidos nesta pesquisa são avaliados na construção de soluções numéricas para diversos problemas como restauração e super-resolução de imagens, problemas de espalhamento e outros obtidos de equações integrais de Fredholm.

Abstract

Discrete ill-posed problems need to be regularized in order to be stably solved. Amongst several regularization methods, perhaps the most used is the method of Tikhonov whose effectiveness depends on a proper choice of the regularization parameter. There are considerable amount of parameter choice rules in the literature; these include the Discrepancy Principle by Morozov and heuristic methods like the L-curve criterion by Hansen, Generalized Cross Validation by Golub, Heath and Wahba, and a fixed point method due to Bazán. Large-scale discrete ill-posed problems can be solved by iterative methods like CGLS and LSQR provided that the iterations are stopped before the noise starts deteriorating the quality of the iterates. This is a difficult task which has not yet been addressed satisfactorily in the literature. In an attempt to alleviate the difficulty associated with selecting the regularization parameter, iterative methods can be combined with Tikhonov regularization giving rise to the so-called hybrid methods such as GKB-FP and W-GCV (both using the identity matrix as regularization matrix).

The contributions of this thesis include further results concerning the theoretical properties of GKB-FP algorithm as well as the extension of GKB-FP to Tikhonov regularization using a general regularization matrix. Apart from this, as a second contribution, we propose an automatic stopping rule for iterative methods for large-scale problems, including the case where the methods are preconditioned via smoothing norms.

Tikhonov regularization has been widely applied to solve linear ill-posed problems, but almost always confined to a single regularization parameter. Nevertheless, some problems have solutions with distinctive characteristics that must be included in the regularized solution. This leads to multi-parameter Tikhonov regularization problems. The third contribution of the thesis is the development of a fixed point method to select the regularization parameters

in this multi-parameter case as well as a GKB-FP type algorithm which is well suited for large-scale problems.

The proposed algorithms are numerically illustrated by solving several problems such as reconstruction and super-resolution image problems, scattering problems and others from Fredholm integral equations.

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Sumário

Capítulo 1

Introdução

Inicialmente descreveremos o tipo de problema que será abordado neste estudo, isto é, suas origens, dificuldades e como contorná-las. Na sequência, apresentaremos os principais objetivos desta pesquisa e como os resultados estão organizados.

1.1 Motivação

Diversos processos e fenômenos físicos das ciências e engenharias são descritos por um conjunto de informações denominados de *entrada*, *sistema* e *saída*. O termo *entrada* se refere à incitação, ou estímulo, fornecido ao *sistema* que irá, através de suas propriedades, produzir uma *saída*, conforme Figura 1.1.

Figura 1.1: Modelagem de um processo físico.

A relação entre o processo físico e sua modelagem matemática é, usualmente, descrita através de equações do tipo

$$\mathcal{A}(f) = g, \tag{1.1}$$

em que $\mathcal{A} : \mathcal{F} \to \mathcal{G}$ é um operador, possivelmente não linear, $f \in \mathcal{F}$ é a função de *entrada*

do processo e $g \in \mathcal{G}$ é a função de *saída*, com $\mathcal{F}$ e $\mathcal{G}$ espaços apropriados. O problema de calcular a *resposta* (ou *saída*) de um *sistema*, dado o modelo e a função de *entrada*, é conhecido como *problema direto*. Já o *problema inverso*, consiste em determinar a *causa* (ou *entrada*) desconhecida a partir de efeitos desejados ou observados. Em outras palavras, sendo conhecida a função g e o operador $\mathcal{A}$, o problema inverso corresponde à inversão do operador, caso exista. Nesta tese trataremos apenas de problemas inversos lineares.

Em problemas práticos, a função g é, usualmente, obtida experimentalmente. Como consequência, em vez de g_{exato}, dispomos apenas de uma aproximação g que contém imprecisões oriundas do processo de aquisição de dados, isto é, $g = g_{\text{exato}} + \epsilon$, em que

$$\|\epsilon\| \leq \delta, \tag{1.2}$$

onde δ depende da precisão dos instrumentos e a norma escolhida depende das características do problema.

Na resolução de problemas inversos, uma análise prévia com relação a existência, unicidade e estabilidade de soluções deve ser realizada. Tal análise é de vital importância pois, se a função g não pertence ao espaço imagem do operador, então (1.1) não tem solução. Outra possibilidade é que o operador $\mathcal{A}$ seja não injetor, caso em que podem existir muitas soluções. A estabilidade é necessária se desejamos assegurar que pequenas variações nos dados produzem pequenas mudanças na solução. Estes três quesitos são a base do conceito de problemas bem-postos, cuja noção remonta ao início do século XX quando Hadamard [53] definiu que o problema (1.1) é bem-posto se as seguintes condições são satisfeitas:

i) existência: dado $g \in \mathcal{G}$, existe $f \in \mathcal{F}$ tal que $\mathcal{A}(f) = g$;

ii) unicidade: dado $g \in \mathcal{G}$, existe único $f \in \mathcal{F}$ que satisfaz (1.1);

iii) estabilidade: a solução f depende continuamente dos dados, isto é, o operador $\mathcal{A}^{-1}$ é contínuo.

Se alguma destas condições não for satisfeita então dizemos que o problema é mal-posto.

Problemas desta natureza aparecem na forma de problemas inversos [3, 31, 43, 87], e os encontramos nas mais diversas áreas como acústica [126], astronomia [37], tomografia com-

putadorizada [108], geofísica [101], biologia matemática [38], processamento de sinais [139], espalhamento de ondas acústicas e eletromagnéticas [33, 36, 74], problemas gravitacionais [67], inversão da equação do calor [29], restauração de imagens 1D e 2D [72, 129], estatística [132], sensoriamento remoto [5], ótica [17], dentre outros. O exemplo típico de problemas mal-postos são as equações integrais de Fredholm de primeira espécie [52]

$$\int_a^b K(s,t)f(t)dt = g(s), \qquad c \leq s \leq d, \tag{1.3}$$

com $K(s,t) : [c,d] \times [a,b] \to \mathbb{R}$, e $g(x) : [c,d] \to \mathbb{R}$, funções contínuas.

A resolução numérica [65, 68] de problemas lineares mal-postos geralmente conduz a um problema de minimização do tipo

$$f_{\text{LS}} = \underset{f \in \mathbb{R}^n}{\operatorname{argmin}} \|g - Af\|_2, \tag{1.4}$$

em que a matriz $A \in \mathbb{R}^{m \times n}$, $m \geq n = \text{posto}(A)$, tem seus valores singulares decaindo para zero sem que haja um salto notório neste decaimento. Além disso, o vetor de dados é da forma $g = g_{\text{exato}} + \epsilon \in \mathbb{R}^m$ em que g_{exato} é o vetor de dados livre de erros, e desconhecido, e ϵ é o ruído (ou vetor de erros) proveniente de medições, erros de truncamentos, etc.

A característica típica da discretização de problemas mal-postos é o elevado número de condição de A, denotado por $\kappa(A)$. Em virtude disso, pequenas alterações no vetor de dados g podem produzir grandes variações na solução. Hansen [63] definiu esse tipo de problema como sendo *Problema Mal-Posto Discreto*. Nesta tese, estamos interessados em construir soluções estáveis para problemas mal-postos discretos de grande porte. Por problemas "de grande porte" consideramos aqueles no qual o cálculo da SVD de A (ou a GSVD do par matricial (A, L)) é computacionalmente inviável e isso, naturalmente, depende do equipamento (processador, memória, etc.) disponível.

Como estamos lidando com problemas mal-postos discretos, a solução $f_{\text{LS}} = A^\dagger g$ (em que $A^\dagger$ denota a matriz Pseudo-Inversa de Moore-Penrose de A) raramente aproxima a solução desejada $f_{\text{exato}} = A^\dagger g_{\text{exato}}$ [60, 65, 68, 72, 87]. Na Figura 1.2 exibimos esta observação. Os dados foram obtidos através da discretização de uma equação integral do tipo (1.3) que modela a restauração de imagem unidimensional. A matriz do sistema satisfaz $A \in \mathbb{R}^{512 \times 512}$,

$\kappa(A) \approx 4,4 \times 10^{20}$, e o vetor de dados satisfaz $\|\epsilon\|_2 = 0,01\|g_{\text{exato}}\|_2$.

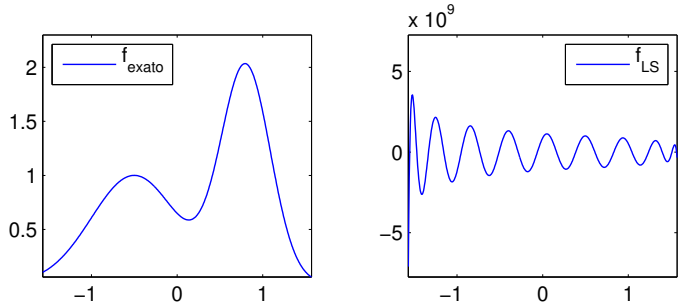


Figura 1.2: Soluções f_{exato} e f_{LS} para um problema mal-posto discreto.

Uma justificativa para este fenômeno vem em termos da Decomposição em Valores Singulares (SVD) [51, 77] da matriz A. Como bem conhecida, a SVD de A é dada por

$$A = \mathsf{U} \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} \mathsf{V}^T, \tag{1.5}$$

em que $\mathsf{U} = [\mathsf{u}_1, \ldots, \mathsf{u}_m] \in \mathbb{R}^{m\times m}$ e $\mathsf{V} = [\mathsf{v}_1, \ldots, \mathsf{v}_n] \in \mathbb{R}^{n\times n}$ são matrizes ortogonais e $\Sigma \in \mathbb{R}^{n\times n}$ é uma matriz diagonal, $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_n)$, com os valores singulares σ_i ordenados de modo não-crescente, $\sigma_1 \geq \cdots \geq \sigma_n > 0$. Assim, a solução f_{LS} é dada por

$$f_{\text{LS}} = A^\dagger g = A^\dagger g_{\text{exato}} + A^\dagger \epsilon = \sum_{i=1}^{n} \frac{\mathsf{u}_i^T g_{\text{exato}}}{\sigma_i} \mathsf{v}_i + \sum_{i=1}^{n} \frac{\mathsf{u}_i^T \epsilon}{\sigma_i} \mathsf{v}_i. \tag{1.6}$$

Assumamos que o vetor de ruídos tem entradas aleatórias, com distribuição normal, média zero e variância especificada. Assumamos também que a Condição Discreta de Picard (CDP) [63] é satisfeita, ou seja, que os coeficientes de Fourier $|\mathsf{u}_i^T g_{\text{exato}}|$, em média, decaem a zero mais rápido do que os valores singulares. Estas hipóteses garantem que existe um k^* tal que

$$|\mathsf{u}_i^T g| = |\mathsf{u}_i^T g_{\text{exato}} + \mathsf{u}_i^T \epsilon| \approx |\mathsf{u}_i^T \epsilon| \approx \text{“constante”}, \quad i > k^*. \tag{1.7}$$

Portanto, para valores singulares pequenos, temos que $|\mathsf{u}_i^T \epsilon|/\sigma_i \gg 1$, e como consequência, a solução f_{LS} não é de utilidade prática pois a influência do ruído é dominante.

Para calcular soluções estáveis, isto é, soluções em que o nível de ruído nos dados é mantido sob controle, algum método de regularização deve ser utilizado. Um dos métodos de

regularização mais conhecidos, e também um dos mais antigos, é devido a Tikhonov [52, 131]. Neste método, a solução f_{LS} é substituída pela solução regularizada f_λ

$$f_\lambda = \underset{f \in \mathbb{R}^n}{\operatorname{argmin}} \left\{ \|g - Af\|_2^2 + \lambda^2 \|Lf\|_2^2 \right\},$$

em que $L \in \mathbb{R}^{p \times n}$ é a matriz de regularização e $\lambda > 0$ é o parâmetro de regularização.

A escolha do parâmetro de regularização pode ser feita através de diversos métodos [9, 15, 49, 56, 57, 83, 86, 88, 95, 104, 123, 124, 138]. Estes métodos são separados em duas classes: métodos que exploram conhecimentos do ruído e métodos que não exploram estas informações, chamados de métodos heurísticos. Métodos da primeira classe incluem o princípio da discrepância (DP) por Morozov [104] que é, provavelmente, o método mais usado quando temos uma boa estimativa para a norma do ruído, o método do erro monótono [1, 56, 138] e o princípio do balanceamento [93]. Métodos da segunda classe incluem o critério da curva-L (LC) de Hansen e O'Leary [73], a Validação Cruzada Generalizada (GCV) de Golub, Heath e Wahba [49] e um método de ponto fixo (FP) de Bazán [9]. Contribuições mais recentes incluem o uso de *ribbons* [24, 26, 27] para aproximar a curva-L e a GCV com peso [34]. Outros métodos também podem ser encontrados em [7, 55, 62, 65, 85, 142].

O parâmetro de regularização escolhido pelo método DP depende da norma do ruído e, sob algumas condições, o erro $\|f_{\text{exato}} - f_\lambda\|$ converge para zero quando $\|\epsilon\| \to 0$ (veja [43]). Infelizmente isto não ocorre com métodos heurísticos e, como consequência, estes podem eventualmente falhar. Porém, isso está de acordo com um resultado devido a Bakushinski [4], que diz que métodos que não levam em consideração o nível de ruído estão sujeitos a falharem, pelo menos para alguns problemas. No entanto, estes métodos são amplamente usados em aplicações reais. Sugerimos [43, 86] para uma discussão sobre o uso de métodos heurísticos e [7, 119] para comparações numéricas entre diversos métodos. Todos os métodos têm suas vantagens e desvantagens, por exemplo, o método DP pode falhar caso a norma do ruído seja mal estimada e o critério da curva-L tem dificuldades quando a curva-L não apresenta o típico formato da letra L [103] ou quando existem múltiplos "cantos" [71, 127].

O método de regularização de Tikhonov é classificado como um método de penalização. Existem outras duas classes de métodos de regularização, a saber: os métodos de projeção

e os métodos híbridos, ou seja, a combinação de métodos de penalização com métodos de projeção. Os métodos de projeção são ideais para lidar com problemas de grande porte. Estes incluem a TSVD [61, 65], TGSVD [62] e métodos baseados em iterações de Lanczos/Arnoldi como os algoritmos CGLS [65], LSQR [112, 113] e os métodos de resíduo mínimo [22, 45, 60, 69, 78, 84, 125], em que o número de iterações, que substitui o parâmetro de regularização, pode ser escolhido pelo princípio da discrepância [54] ou pelo critério da curva-L [71].

Dois métodos híbridos bem sucedidos são GKB-FP [10] e W-GCV [34]. Estes não exigem conhecimentos a respeito da norma do ruído e combinam projeções sobre o subespaço de Krylov gerado pelo processo de bidiagonalização de Golub-Kahan (GKB) [50], com o método de regularização de Tikhonov usando $L = I_n$, ou seja, a matriz identidade de orden n. A principal diferença entre os dois métodos está na escolha do parâmetro de regularização, enquanto W-GCV utiliza o método GCV com peso [34], GKB-FP utiliza o método FP [9]. Um algoritmo que utiliza o critério da curva-L para a escolha do parâmetro de regularização é o LSQR-Tik [80, 81, 82]. Outras propostas podem ser encontradas em [18, 26]. Já para o caso em que $L \neq I_n$, uma abordagem que utiliza a bidiagonalização simultânea de duas matrizes aplicada ao par (A, L) foi proposta por Kilmer et al. [84], no entanto, este método pode ser inviável para problemas de grande porte, pois o procedimento descrito é muito exigente do ponto de vista computacional. Lampe et al. [89] e Reichel et al. [120] propuseram calcular soluções aproximadas minimizando o funcional de Tikhonov em subespaços de Krylov generalizado, em ambos os casos o parâmetro de regularização é escolhido pelo princípio da discrepância. Em [46, 76] temos outros métodos com esta mesma abordagem.

Métodos para a escolha do parâmetro de regularização como DP, LC e FP são facilmente implementados por intermédio da Decomposição em Valores Singulares Generalizados (GSVD) [51, 111] do par matricial (A, L). Entretanto, sabemos que calcular a GSVD é uma tarefa computacionalmente exigente e até proibitiva quando lidamos com problemas de grande porte, logo, para estes casos, os métodos iterativos e de projeção tornam-se bastante atrativos. Contudo, determinar a iteração de parada não é uma tarefa trivial, este problema pode ser parcialmente contornado caso nos seja fornecido a priori um parâmetro λ. Neste caso, para calcularmos a solução, podemos usar métodos iterativos para aproximar f_λ.

A seguir, elencamos as principais dificuldades concernentes ao método de regularização

de Tikhonov com $L \neq I_n$:

- problemas de grande porte inviabilizam o cálculo da GSVD do par (A, L);
- a escolha do parâmetro de regularização λ não é uma tarefa trivial;
- o cálculo da solução regularizada f_λ sem o uso da GSVD não é simples.

Existem situações em que a solução f_{exato} tem várias características distintas, como, por exemplo, em problemas de imagens (suavidade, movimento, etc), e é natural que queiramos incorporar na solução regularizada tais características. Assim, substituímos a solução f_{LS} pela solução f_λ obtida através do método de regularização de Tikhonov com múltiplos parâmetros

$$f_\lambda = \underset{f \in \mathbb{R}^n}{\operatorname{argmin}} \left\{ \|g - Af\|_2^2 + \lambda_1^2 \|L_1 f\|_2^2 + \lambda_2^2 \|L_2 f\|_2^2 + \cdots + \lambda_q^2 \|L_q f\|_2^2 \right\}, \tag{1.8}$$

em que $L_i \in \mathbb{R}^{p_i \times n}$, $i = 1, \ldots, q$, são as matrizes de regularização e $\lambda = [\lambda_1, \ldots, \lambda_q]^T$, $\lambda_i > 0$, é o vetor de parâmetros de regularização. Aplicações deste método tem aparecido em várias áreas como na determinação de geopotenciais a partir de órbitas precisas de satélites [135], reconstrução de imagens de alta resolução com erros de deslocamento [98], super-resolução de imagens [142] e estimação de parâmetros em processos de difusão de salto [41].

Assim como no caso com um parâmetro, os métodos para a escolha dos parâmetros podem ser separados em duas classes: métodos que exploram conhecimentos do ruído e métodos que não exploram estas informações. Métodos da primeira classe incluem o uso do princípio da discrepância (como em Lu et al. [97] e Lu e Pereverzev [96]) e os artigos [6, 8, 32], no qual os parâmetros de regularização dependem da estrutura do ruído. Outros métodos podem ser encontrados em [47, 140]. A segunda classe inclui a generalização do método da curva-L [14], a GCV generalizada [21], a abordagem proposta por Brezinski et al. [21] e a função de distância mínima por Belge et al. [15].

Não conhecemos na literatura uma decomposição do tipo GSVD para uma $(q+1)$-upla de matrizes $(A, L_1, \ldots, L_q)$, ou seja, um conjunto de matrizes ortogonais/não-singulares $U, V_1, \ldots, V_q, X$ e matrizes diagonais $D_A, D_1, \ldots, D_q$ tais que $XAU = D_A$ e $XL_iV_i = D_i$, $i = 1, \ldots, q$. Logo, a obtenção da solução regularizada não é trivial, especialmente para problemas de grande porte. Uma primeira tentativa foi feita em [120] no qual um algoritmo

baseado em iterações de Arnoldi generalizado é proposto. O algoritmo parece promissor, mas sua eficiência ainda precisa ser verificada.

A seguir, elencamos as principais dificuldades concernentes ao método de regularização de Tikhonov com múltiplos parâmetros:

- não conhecemos uma decomposição do tipo GSVD para uma $(q+1)$-upla de matrizes $(A, L_1, L_2, \ldots, L_q)$;
- escolha dos parâmetros de regularização $\lambda_1, \lambda_2, \ldots, \lambda_q$ não é uma tarefa trivial;
- cálculo da solução regularizada f_λ para problemas de pequeno e grande porte não é simples.

Deste modo, é importante o desenvolvimento de novos métodos para: *i*) escolha do parâmetro de regularização para o método de regularização de Tikhonov com $L \neq I_n$ para problemas de grande porte; *ii*) escolha dos parâmetros de regularização para o método de regularização de Tikhonov com múltiplos parâmetros para problemas de pequeno/grande porte. Além, é claro, de meios eficientes para o cálculo das respectivas soluções regularizadas.

1.2 Objetivos

Nossos principais objetivos com esta tese são:

1. apresentar novas informações a respeito das propriedades de convergência do algoritmo GKB-FP [10] e estender o seu uso para o método de regularização de Tikhonov com $L \neq I_n$;
2. utilizar métodos iterativos com um critério de parada apropriado e incorporar informações a priori da solução desejada nas soluções iteradas;
3. desenvolver um método de ponto fixo para a escolha dos parâmetros para o método de regularização de Tikhonov com múltiplos parâmetros e desenvolver extensões do algoritmo GKB-FP para problemas (1.8) de grande porte.

1.3 Organização da Tese

O texto deste trabalho está organizado de modo que os objetivos acima descritos são contemplados sequencialmente. A seguir, apresentamos um breve resumo sobre cada capítulo.

1.3.1 Capítulo 2

Neste capítulo revisamos o algoritmo GKB-FP, apresentamos novos resultados em relação às suas propriedades de convergência e nos concentramos em como aplicá-lo ao método de regularização de Tikhonov com $L \neq I_n$, em especial para problemas de grande porte, publicado em [12]. Duas abordagens são consideradas: a primeira baseia-se no fato de que podemos transformar, teoricamente, o método de regularização de Tikhonov com $L \neq I_n$ para o caso com $L = I_n$, no qual o GKB-FP pode ser aplicado; a segunda abordagem consiste em construirmos uma sequência de soluções regularizadas minimizando o funcional de Tikhonov no subespaço de Krylov gerado durante o processo de bidiagonalização de Golub-Kahan aplicado à matriz A, como sugerido em [76]. Em todos os casos a filosofia de GKB-FP é preservada, isto é, o método de ponto fixo de Bazán [9] é usado no problema projetado.

1.3.2 Capítulo 3

Para diminuirmos possíveis dificuldades associadas com a escolha do parâmetro de regularização de Tikhonov em cada iteração, como feito por GKB-FP ou W-GCV, propomos uma regularização iterativa baseada no algoritmo LSQR aplicado ao problema de mínimos quadrados $\operatorname{argmin} \|\overline{g} - \overline{A}\,\overline{f}\|_2$, em que o vetor $\overline{g}$ e a matriz $\overline{A}$ surgem da transformação para o caso $L = I_n$, juntamente com um critério de parada automático que não requer conhecimentos a respeito da norma do ruído, publicado em [12]. Teoricamente, seguimos um artigo por Hanke e Hansen [60] em que é mostrado como construir soluções regularizadas para (1.4) via CGLS/LSQR de tal modo que as propriedades da matriz L são incorporadas nas soluções iteradas. Além disso, apresentamos cotas para a qualidade da solução TSVD com relação às soluções iteradas e realizamos alguns experimentos numéricos para verificar a eficácia do critério proposto em conexão com outros algoritmos existentes, em particular com o *pruning algorithm* [71] que é baseado na curva-L discreta.

1.3.3 Capítulo 4

Neste capítulo apresentamos um método de ponto fixo, chamado de MFP, para a escolha dos parâmetros de regularização para o método de regularização de Tikhonov com múltiplos parâmetros, publicado em [11]. O algoritmo MFP pode ser visto como uma extensão natural do método de ponto fixo proposto por Bazán [9]. Além disso, propomos um algoritmo do tipo GKB-FP para problemas de grande porte.

1.3.4 Capítulo 5

Por fim, apresentamos os resultados numéricos obtidos pelos algoritmos propostos nesta pesquisa. Realizamos simulações numéricas em alguns problemas clássicos obtidos na literatura, problemas de espalhamento, restauração e super-resolução de imagens. Para fins de comparação, no caso em que tratamos o método de regularização de Tikhonov com $L \neq I_n$, utilizamos outros métodos como a curva-L e o princípio da discrepância para a escolha do parâmetro de regularização, bem como a GSVD do par matricial (A, L) e outro método para problemas de grande porte baseado em um processo de bidiagonalização simultânea [84]. Do mesmo modo, no caso com múltiplos parâmetros, comparamos nossos resultados com os obtidos por duas outras técnicas disponíveis na literatura [21, 96].

No final do trabalho apresentamos as considerações finais, as possibilidades de trabalhos futuros e cinco apêndices, a saber: o produto de Kronecker, ângulos e distâncias entre subespaços, o uso da fatoração LU no cálculo de soluções de norma mínima, um algoritmo baseado no processo de bidiagonalização simultânea do par (A, L) [84] e os códigos fontes (em linguagem Matlab) dos programas utilizados nesta tese.

Capítulo 2

Método de regularização de Tikhonov para problemas de grande porte e o algoritmo GKB-FP

O algoritmo GKB-FP, publicado recentemente por Bazán e Borges [10], foi desenvolvido para problemas mal-postos discretos de grande porte. Neste capítulo faremos uma breve revisão do método de regularização de Tikhonov e de alguns métodos para a escolha do parâmetro de regularização. Além disso, apresentamos o algoritmo GKB-FP juntamente com novas informações referentes às suas propriedades de convergência e, também, mostramos como este pode ser aplicado ao método de regularização de Tikhonov com $L \neq I_n$.

2.1 Método de regularização de Tikhonov

Conforme já comentamos no capítulo anterior, o método de regularização de Tikhonov é um dos métodos mais usados quando precisamos calcular soluções estáveis para o problema (1.4). Neste método, a solução f_{LS} é substituída pela solução regularizada, colocada aqui novamente por conveniência, dada por

$$f_\lambda = \underset{f\in\mathbb{R}^n}{\operatorname{argmin}} \left\{ \|g - Af\|_2^2 + \lambda^2 \|Lf\|_2^2 \right\}, \tag{2.1}$$

em que A é dada como em (1.4), $L \in \mathbb{R}^{p\times n}$ é a matriz de regularização, que tem por finalidade incorporar na solução regularizada propriedades da solução desejada, e $\lambda > 0$ é o parâmetro de regularização. Dizemos que (2.1) está na forma padrão caso $L = I_n$ ou na forma geral caso $L \neq I_n$. A equação (2.1) pode ser reescrita como

$$f_\lambda = \underset{f\in\mathbb{R}^n}{\operatorname{argmin}} \left\| \begin{pmatrix} g \\ 0 \end{pmatrix} - \begin{pmatrix} A \\ \lambda L \end{pmatrix} f \right\|_2^2. \tag{2.2}$$

Assim, encontrar a solução de (2.1) é equivalente a encontrar a solução das equações normais regularizadas

$$(A^TA + \lambda^2 L^TL)f = A^Tg, \tag{2.3}$$

cuja unicidade da solução garantimos com a condição $\mathcal{N}(A)\cap\mathcal{N}(L) = 0$, ou seja, a intersecção do núcleo das matrizes A e L é trivial. Ao longo deste trabalho vamos denotar o espaço nulo e o espaço coluna (ou imagem) de uma matriz A por $\mathcal{N}(A)$ e $\mathcal{R}(A)$, respectivamente.

Vamos considerar que $L \in \mathbb{R}^{p\times n}$ com $p \leq n$. Notemos que se $p \geq n$, então podemos calcular a decomposição QR de L, ou seja, $L = QR$ com R tendo mais colunas do que linhas e com posto linha completo. Assim, $\|Lf\|_2 = \|QRf\|_2 = \|Rf\|_2$. Deste modo, a decomposição GSVD do par (A, L) pode ser escrita como

$$A = \check{\mathsf{U}} \begin{bmatrix} \check{\Sigma} & 0 \\ 0 & I_{n-p} \end{bmatrix} \check{\mathsf{X}}, \quad L = \check{\mathsf{V}}[\check{\mathsf{M}}; 0]\check{\mathsf{X}}. \tag{2.4}$$

Aqui, $\check{\mathsf{U}} = [\check{\mathsf{u}}_1, \ldots, \check{\mathsf{u}}_n] \in \mathbb{R}^{m\times n}$ e $\check{\mathsf{V}} = [\check{\mathsf{v}}_1, \ldots, \check{\mathsf{v}}_p] \in \mathbb{R}^{p\times p}$ têm colunas ortonormais, $\check{\mathsf{X}} \in \mathbb{R}^{n\times n}$ é uma matriz não-singular, $\check{\Sigma}$ e $\check{\mathsf{M}}$ são matrizes diagonais: $\check{\Sigma} = \operatorname{diag}(\check{\sigma}_1, \ldots, \check{\sigma}_p)$, $\check{\mathsf{M}} = \operatorname{diag}(\check{\mu}_1, \ldots, \check{\mu}_p)$. Vamos definir $\delta_0 = \|(I_m - \check{\mathsf{U}}\check{\mathsf{U}}^T)g\|_2$ (o tamanho da componente de g que está fora de $\mathcal{R}(A)$). Assim, os valores singulares generalizados do par (A, L) são $\check{\gamma}_i = \check{\sigma}_i/\check{\mu}_i$. Então, se $x(\lambda) = \|g - Af_\lambda\|_2^2$ e $y(\lambda) = \|Lf_\lambda\|_2^2$, temos

$$x(\lambda) = \sum_{i=1}^{p} \frac{\lambda^4 |\check{\mathsf{u}}_i^T g|^2}{\left(\check{\gamma}_i^2 + \lambda^2\right)^2} + \delta_0^2, \quad y(\lambda) = \sum_{i=1}^{p} \frac{\check{\gamma}_i^2 |\check{\mathsf{u}}_i^T g|^2}{\left(\check{\gamma}_i^2 + \lambda^2\right)^2}, \tag{2.5}$$

e, para $\lambda > 0$, as derivadas com respeito a λ destas funções satisfazem

$$x'(\lambda) = 4\lambda^3 \sum_{i=1}^{p} \frac{\check{\gamma}_i^2 |\check{\mathsf{u}}_i^T g|^2}{\left(\check{\gamma}_i^2 + \lambda^2\right)^3} > 0, \quad y'(\lambda) = -4\lambda \sum_{i=1}^{p} \frac{\check{\gamma}_i^2 |\check{\mathsf{u}}_i^T g|^2}{\left(\check{\gamma}_i^2 + \lambda^2\right)^3} < 0. \tag{2.6}$$

Concluímos, então, que a norma (ou seminorma) da solução regularizada é uma função decrescente em λ e a norma do resíduo é uma função crescente em λ. Essas propriedades são exploradas frequentemente pelos métodos de escolha do parâmetro de regularização.

A solução regularizada f_λ, equação (2.1), é facilmente calculada desde que a GSVD do par matricial (A, L) esteja disponível. Entretanto, problemas de grande porte inviabilizam o cálculo desta decomposição, fazendo, assim, com que a GSVD seja apenas de interesse teórico.

2.2 Métodos para a escolha do parâmetro de regularização

Nesta seção faremos uma sucinta descrição de alguns dos métodos clássicos para a escolha do parâmetro de regularização. Além disso, veremos também o método de ponto fixo de Bazán uma vez que este é uma das bases do algoritmo GKB-FP. Para a simulação de dados, usamos problemas obtidos no pacote RegularizationTools [64], tais como heat, shaw, phillips e baart.

2.2.1 Princípio da Discrepância

O Princípio da Discrepância (DP) de Morozov [104, 105] de 1966 é, provavelmente, o método mais usado quando temos uma estimativa para o nível de ruído. A ideia é que devemos encontrar o parâmetro λ tal que a solução f_λ satisfaça a equação da discrepância

$$\|g - Af_\lambda\|_2 = c\|\epsilon\|_2 \equiv \delta, \quad c \geq 0, \tag{2.7}$$

o que geometricamente significa encontrar o ponto λ, se existir, da intersecção entre a curva $x(\lambda) = \|g - Af_\lambda\|_2^2$, que é crescente, com a reta $z = \delta^2$, conforme a Figura 2.1.

Apesar deste método ser um dos poucos que tem uma base teórica robusta, a dificuldade

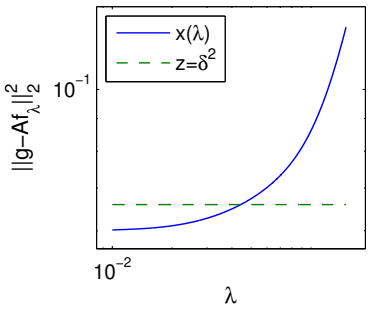


Figura 2.1: Interpretação geométrica do princípio da discrepância: encontrar a intersecção entre a curva $x(\lambda) = \|g - Af_\lambda\|_2^2$ e a reta $z = \delta^2$.

está no fato de que se a norma do ruído é mal estimada, então o método pode falhar. Na Figura 2.2 ilustramos uma situação em que utilizamos $\delta = 0,994\|\epsilon\|_2$, $\delta = 0,995\|\epsilon\|_2$ e $\delta = 1,01\|\epsilon\|_2$ para o problema shaw com $n = 512$ e NL $= \|\epsilon\|_2/\|g_{\text{exato}}\|_2 = 0,01$. Ao longo deste trabalho usaremos a sigla NL para representar o nível de ruído (*Noise Level*). Podemos perceber que para $\delta = 0,994\|\epsilon\|_2$ a solução f_λ não é uma boa aproximação para f_{exato}.

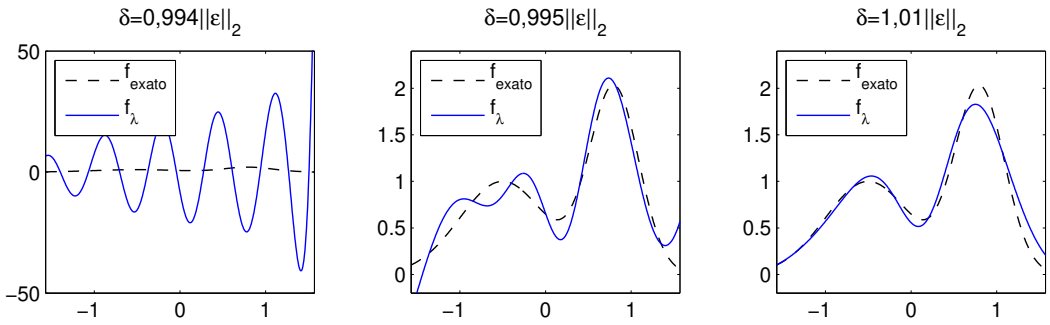


Figura 2.2: Soluções f_λ com alguns valores de δ para o problema shaw com $n = 512$ e NL $=$ 0,01.

2.2.2 Validação Cruzada Generalizada

Em 1979, Golub, Heath e Wahba [49] sugeriram o método da Validação Cruzada Generalizada (GCV) que é um método baseado em considerações estatísticas. O parâmetro escolhido por este método é o minimizador da função GCV, $G_{A,g} : [0, \infty) \subset \mathbb{R} \to \mathbb{R}$

$$G_{A,g}(\lambda) = \frac{\|g - Af_\lambda\|_2^2}{(\text{traço}(I_m - B(\lambda)))^2}, \quad B(\lambda) = A\left(A^TA + \lambda^2 L^TL\right)^{-1} A^T \in \mathbb{R}^{m\times m}. \tag{2.8}$$

Minimizar tal função é uma tarefa computacionalmente exigente, pois existe uma inversão matricial no cálculo da matriz $B(\lambda)$. Contudo, com a GSVD do par (A, L), a avaliação desta função torna-se mais simples. As dificuldades deste método ocorrem quando o minimizador da função GCV está em uma região quase plana, o que ocorre com certa frequência, ou quando existem vários minimizadores locais (veja a Figura 2.3).

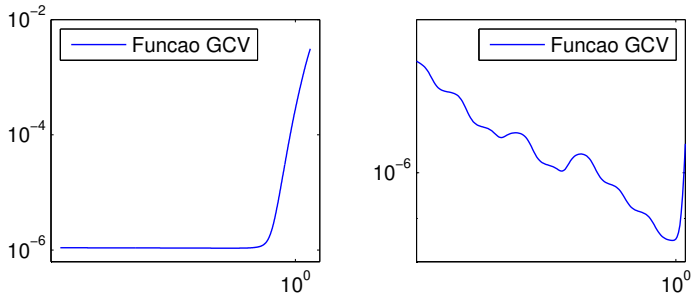


Figura 2.3: Função GCV para o problema shaw com $n = 512$ e $L = I_n$ (esquerda), e para o problema baart com $n = 32$ e L uma versão discretizada do operador diferencial de segunda ordem. Ambos com NL = 0,01.

2.2.3 Curva-L

O critério da curva-L por Hansen e O'Leary [73] (veja também [65]), sugere escolher o parâmetro de regularização como o "canto", ou seja, o ponto de máxima curvatura, da curva

$$\mathcal{L}(\lambda) = \left\{ (\log \|g - Af_\lambda\|_2, \log \|Lf_\lambda\|_2) \in \mathbb{R}^2 \ / \ \lambda > 0 \right\}. \tag{2.9}$$

Na Figura 2.4 (esquerda) temos uma típica curva-L e podemos perceber que a curva lembra a letra L. Este critério funciona bem, exceto quando a curva (2.9) não apresenta o típico formato da letra L [103] ou quando existem vários maximizadores locais da curvatura [71, 127] como na Figura 2.4 (direita). Outras limitações podem ser encontradas em [59, 66, 134].

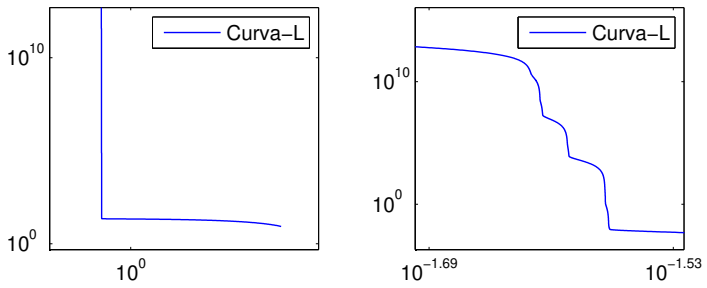


Figura 2.4: Curva-L para o problema shaw com $n = 512$ e $L = I_n$ (esquerda), e para o problema baart com $n = 32$ e L uma versão discretizada do operador diferencial de segunda ordem. Ambos com NL = 0,01.

2.2.4 O método de Regińska e sua realização via iterações de ponto fixo

Regińska [118] sugeriu escolher como parâmetro de regularização minimizadores da função $\Psi : [0, \infty) \subset \mathbb{R} \to \mathbb{R}$ definida por

$$\Psi(\lambda) = \|g - Af_\lambda\|_2^2 \|Lf_\lambda\|_2^{2\mu}, \quad \mu > 0. \tag{2.10}$$

Regińska provou que se a curvatura da curva-L é maximizada no ponto $\lambda = \lambda^*$ e se a reta tangente no ponto $(\log \|g - Af_{\lambda^*}\|_2, \log \|Lf_{\lambda^*}\|_2)$ tem inclinação $-1/\mu$, então a função $\Psi(\lambda)$ é minimizada em $\lambda = \lambda^*$. Para simplificar nossas argumentações, vamos considerar o método de regularização de Tikhonov na forma padrão. Logo, $\Psi(\lambda) = \|r_\lambda\|_2^2 \, \|f_\lambda\|_2^{2\mu}$ em que f_λ é a solução de (2.1), dada por $f_\lambda = (A^TA + \lambda^2 I_n)^{-1} A^T g$, e $r_\lambda = g - Af_\lambda$ é o resíduo correspondente. A motivação para usarmos este método vem da observação de que a norma da solução regularizada e a norma do resíduo são funções monótonas, conforme equações (2.5) e (2.6). Assim, é natural pensar que estas devem se equilibrar em algum momento. Para uma justificativa mais robusta vamos considerar a matriz $R_\lambda = (A^TA + \lambda^2 I_n)^{-1} A^T$. Então, podemos separar o erro na solução f_λ em

$$f_{\text{exato}} - f_\lambda = f_{\text{exato}} - R_\lambda g_{\text{exato}} - R_\lambda \epsilon. \tag{2.11}$$

Assim, obtemos a estimativa para o erro como a soma de dois termos

$$\|f_{\text{exato}} - f_\lambda\|_2 \leq \|f_{\text{exato}} - R_\lambda g_{\text{exato}}\|_2 + \|R_\lambda \epsilon\|_2 \equiv E_1(\lambda) + E_2(\lambda). \tag{2.12}$$

O termo $E_1(\lambda)$, que aumenta com λ, representa o erro na regularização enquanto que $E_2(\lambda)$, que diminui com λ, denota o erro causado pelo ruído (veja [87]). Assim, para obtermos um erro pequeno, devemos equilibrar estas quantidades para que (2.12) seja minimizada. Contudo, como não temos a nossa disposição $E_1(\lambda)$ e $E_2(\lambda)$, o melhor que podemos fazer é minimizar um modelo para a cota (2.12) e escolher o minimizador deste modelo como parâmetro de regularização. Logo, minimizar $\log \Psi(\lambda)$ é equivalente a minimizar a soma de dois termos, um crescente e outro decrescente. Na Figura 2.5 podemos apreciar a cota (2.12) (esquerda) e a função $\Psi(\lambda)$ com $\mu = 1$ (centro) para o problema shaw com $n = 512$ e NL = 0,01.

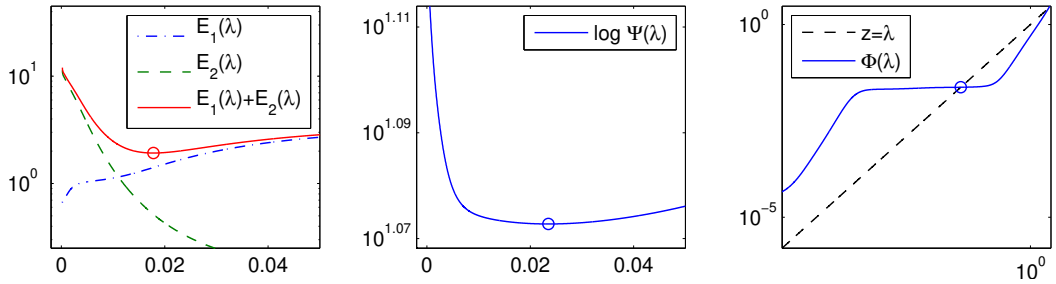


Figura 2.5: Cota (2.12) e funções $\Psi(\lambda)$ e $\phi(\lambda; 1)$. O círculo denota a localização do minimizador e do ponto fixo correspondente. Temos $\lambda_c = \text{argmin}\{E_1(\lambda) + E_2(\lambda)\} = 0,0178$ e $\lambda_\Psi = \text{argmin}\{\Psi(\lambda), \mu = 1\} = 0,0235$, com erros relativos $E_{\lambda_c} = 0,0639$ e $E_{\lambda_\Psi} = 0,0749$.

Do ponto de vista prático, os minimizadores de $\Psi(\lambda)$ podem ser calculados através de iterações de ponto fixo [9] que necessitam apenas da norma do resíduo e da norma da solução, isto é, os minimizadores de $\Psi(\lambda)$ são pontos fixos da função $\phi : [0, \infty) \subset \mathbb{R} \to \mathbb{R}$, definida por:

$$\phi(\lambda; \mu) = \sqrt{\mu}\frac{\|r_\lambda\|_2}{\|f_\lambda\|_2}, \quad \mu > 0. \tag{2.13}$$

Na Figura 2.5 (direita) temos a função $\phi(\lambda; 1)$ para o problema shaw. As ideias gerais do algoritmo de ponto fixo (FP) são:

- fornecemos uma aproximação inicial $\lambda_0 > 0$, definimos $\mu = 1$ e calculamos a sequência

$\lambda_{k+1} = \phi(\lambda_k; \mu)$, $k \geq 0$, até atingirmos algum critério de parada ou caso a divergência seja detectada;

- caso a divergência seja detectada, ajustamos μ conforme [9, 13] e as iterações recomeçam.

A curva-L (2.9) é convexa em uma vizinhança de um ponto λ_0 se, e somente se, para todo λ nesta vizinhança vale [13]

$$\phi'(\lambda; 1) \leq \frac{\phi(\lambda; 1)}{\lambda}. \tag{2.14}$$

Podemos apreciar um exemplo desta condição na Figura 2.6 em que optamos por usar o problema heat com dimensão $n = 64$ e NL = 0,05. No gráfico a esquerda temos a curva-L associada ao problema em questão e no gráfico a direita a condição (2.14). Assim, um ponto fixo λ^* é convexo (côncavo) se a curva-L em uma vizinhança de λ^* é convexa (côncava).

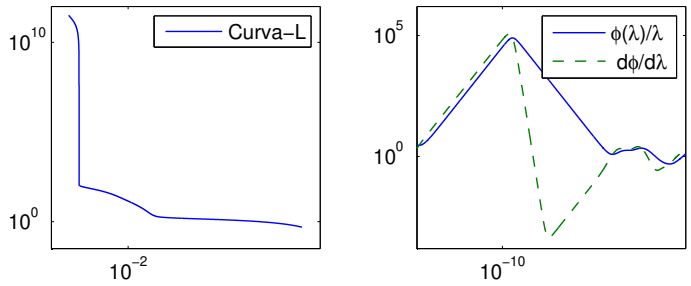


Figura 2.6: Curva-L para o problema heat com $n = 64$, $L = I_n$ e NL = 0,05 (esquerda) e curvas $\phi'(\lambda; 1)$ e $\phi(\lambda; 1)/\lambda$ em escala log-log (direita).

2.3 O algoritmo GKB-FP

Para problemas de tamanho pequeno a moderado, o método FP para a escolha do parâmetro de regularização de Tikhonov pode ser eficientemente implementado pela SVD da matriz A (ou pela GSVD do par (A, L)). Entretanto, problemas de grande porte inviabilizam tal decomposição. Com o propósito de implementar este método para problemas de grande porte com $L = I_n$, foi desenvolvido o algoritmo GKB-FP por Bazán e Borges [10] e Borges [20].

Seja λ^* o maior ponto fixo convexo da função $\phi(\lambda;\mu)$ associada ao problema de grande porte. A ideia central do GKB-FP consiste em aproximar λ^* através de uma sequência $(\lambda^{(k)^*})_{k\geq 2}$, em que $\lambda^{(k)^*}$ são pontos fixos de funções $\phi^{(k)}(\lambda;\mu)$ que satisfazem $\phi^{(k)}(\lambda;\mu) \to \phi(\lambda;\mu)$ quando $k \to n$. Estas funções $\phi^{(k)}(\lambda;\mu)$ são geradas através do processo GKB.

O processo GKB é um método iterativo que aplicado à matriz A com vetor inicial g gera, após $k < n$ iterações, as matrizes $U_{k+1} = [u_1, \ldots, u_{k+1}] \in \mathbb{R}^{m\times(k+1)}$ e $V_k = [v_1, \ldots, v_k] \in \mathbb{R}^{n\times k}$ com colunas ortonormais e a matriz bidiagonal inferior $B_k \in \mathbb{R}^{(k+1)\times k}$,

$$B_k = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_k \\ & & & \beta_{k+1} \end{pmatrix}, \tag{2.15}$$

tais que

$$\beta_1 U_{k+1} e_1 = g = \beta_1 u_1, \tag{2.16}$$

$$AV_k = U_{k+1} B_k, \tag{2.17}$$

$$A^T U_{k+1} = V_k B_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T, \tag{2.18}$$

em que e_k denota o k-ésimo vetor canônico de dimensão apropriada. No Algoritmo 2.1 temos o esquema do processo GKB.

Entrada: A, g
Saída: Matrizes U_{k+1}, V_k e B_k.
1. $\beta_1 u_1 = g, \quad \alpha_1 v_1 = A^T u_1$
2. **Para i=1,2,...**
$\beta_{i+1} u_{i+1} = A v_i - \alpha_i u_i$
$\alpha_{i+1} v_{i+1} = A^T u_{i+1} - \beta_{i+1} v_i$
$B_{i,i} = \alpha_i, \quad B_{i+1,i} = \beta_{i+1}$
Fim para

Algoritmo 2.1: Processo GKB.

Teoricamente, os vetores v_i e u_i são ortogonais, no entanto, numericamente é possível que

esta ortogonalidade seja perdida. Assim, implementações do processo GKB podem ser feitas sem ou com reortogonalização dos vetores v_i e u_i. No Capítulo 5 voltaremos com a questão da reortogonalização.

Um resultado com relação à matriz V_k é que suas colunas formam uma base ortonormal para o subespaço de Krylov $\mathcal{K}_k(A^TA, A^Tg) = \text{span}\left\{A^Tg, A^TAA^Tg, \ldots, (A^TA)^{k-1}A^Tg\right\}$ de dimensão k, para $k \leq n$, isto é,

$$\text{span}\{v_1, \ldots, v_k\} = \mathcal{K}_k(A^TA, A^Tg). \tag{2.19}$$

Em aplicações que envolvem problemas mal-postos discretos, este subespaço serve como aproximação para o subespaço $\mathcal{S}_k = \text{span}\{\mathsf{v}_1, \ldots, \mathsf{v}_k\}$ (veja, por exemplo, [65]). Logo, $\mathcal{K}_k(A^TA, A^Tg)$ é uma excelente escolha para a resolução de problemas mal-postos discretos. Portanto, podemos aproximar a solução f_λ por uma sequência de soluções $f_\lambda^{(k)}$ dadas por

$$f_\lambda^{(k)} = \underset{f \in \mathcal{K}_k(A^TA, A^Tg)}{\text{argmin}} \left\{\|g - Af\|_2^2 + \lambda^2\|f\|_2^2\right\}, \tag{2.20}$$

cujo resíduo associado é $r_\lambda^{(k)} = g - Af_\lambda^{(k)}$. Como V_k fornece uma base ortonormal para o subespaço $\mathcal{K}_k(A^TA, A^Tg)$, segue que dado $f \in \mathcal{K}_k(A^TA, A^Tg)$ existe $d \in \mathbb{R}^k$ tal que $f = V_kd$. Logo, usando as equações (2.16) a (2.18), temos

$$\begin{aligned}\left\|\begin{pmatrix} g \\ 0 \end{pmatrix} - \begin{pmatrix} A \\ \lambda I_n \end{pmatrix} f\right\|_2^2 &= \left\|\begin{pmatrix} \beta_1 U_{k+1} e_1 \\ 0 \end{pmatrix} - \begin{pmatrix} A \\ \lambda I_n \end{pmatrix} V_k d\right\|_2^2 \\ &= \left\|\begin{pmatrix} \beta_1 e_1 \\ 0 \end{pmatrix} - \begin{pmatrix} B_k \\ \lambda I_n \end{pmatrix} d\right\|_2^2.\end{aligned} \tag{2.21}$$

Portanto, resolver o problema restrito (2.20), é equivalente a resolver o problema irrestrito

$$d_\lambda^{(k)} = \underset{d \in \mathbb{R}^k}{\text{argmin}} \left\{\|\beta_1 e_1 - B_k d\|_2^2 + \lambda^2\|d\|_2^2\right\}, \quad f_\lambda^{(k)} = V_k d_\lambda^{(k)}. \tag{2.22}$$

Além disso, a norma da solução regularizada $f_\lambda^{(k)}$ e a norma do resíduo correspondente $r_\lambda^{(k)}$

satisfazem, respectivamente,

$$\|f_\lambda^{(k)}\|_2 = \|d_\lambda^{(k)}\|_2, \quad \|g - Af_\lambda^{(k)}\|_2 = \|\beta_1 e_1 - B_k d_\lambda^{(k)}\|_2. \tag{2.23}$$

Um resultado interessante com relação às normas das soluções $f_\lambda^{(k)}$ e dos resíduos $r_\lambda^{(k)}$ é que estas formam sequências monótonas, ou seja, para $\lambda > 0$ fixo e $k = 1, \ldots, n-1$ vale $\|f_\lambda^{(k+1)}\|_2 \geq \|f_\lambda^{(k)}\|_2$ e $\|r_\lambda^{(k+1)}\|_2 \leq \|r_\lambda^{(k)}\|_2$ (veja [10, 20]). Assim, considerando o problema irrestrito (2.22), definimos, para $k \geq 2$, a função $\phi^{(k)} : [0, \infty) \subset \mathbb{R} \to \mathbb{R}$, por

$$\phi^{(k)}(\lambda; \mu) = \sqrt{\mu} \frac{\|\beta_1 e_1 - B_k d_\lambda^{(k)}\|_2}{\|d_\lambda^{(k)}\|_2}, \quad \mu > 0. \tag{2.24}$$

Como a matriz B_k é bidiagonal inferior, a resolução do problema irrestrito (2.22) pode ser feita de modo eficiente através da decomposição QR de B_k via rotações de Givens, ou seja, o custo computacional é da ordem de $\mathcal{O}(k)$ operações. Logo, o esforço computacional associado ao problema (2.22) e, consequentemente, na avaliação da função $\phi^{(k)}(\lambda; \mu)$, é praticamente insignificante para n grande. Portanto, o principal custo computacional fica por conta do processo GKB para obtermos a matriz B_k. Duas propriedades importantes das funções $\phi^{(k)}(\lambda; \mu)$ é que estas satisfazem $\phi(\lambda; \mu) \leq \phi^{(k+1)}(\lambda; \mu) \leq \phi^{(k)}(\lambda; \mu)$, $k \geq 2$, e que a sequência $\phi^{(k)}(\lambda; \mu)$ rapidamente converge para $\phi(\lambda; \mu)$, pelo menos para valores de λ próximos de σ_1. Na Figura 2.7 exibimos algumas funções $\phi^{(k)}(\lambda; 1)$ para o problema shaw e podemos notar que, após $k = 6$ iterações, a função $\phi^{(6)}(\lambda; 1)$ praticamente coincide com a $\phi(\lambda; 1)$ do problema original no intervalo que contém o ponto fixo que minimiza $\Psi(\lambda)$.

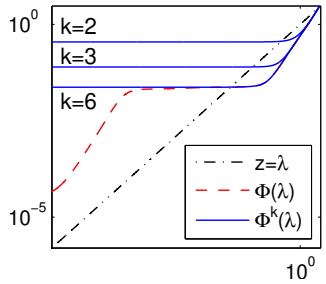


Figura 2.7: Algumas funções $\phi^{(k)}(\lambda; 1)$ e a função $\phi(\lambda; 1)$ para o problema shaw com $n = 512$ e NL = 0,01.

O resultado mais importante provado em [10, 20] é a estabilização da solução $f_\lambda^{(k)}$ com relação ao número de iterações.

Teorema 2.1. *Assuma que o maior ponto fixo convexo de* $\phi(\lambda;1)$ *tenha sido obtido com* k *passos GKB, i.e.,* $\phi(\lambda^*;1) = \lambda^* = \phi^{(k)}(\lambda^*;1)$. *Então,* $f_{\lambda^*}^{(k)} = f_{\lambda^*}^{(k+1)} = \cdots = f_{\lambda^*}^{(n)} = f_{\lambda^*}$ *e, como consequência, a norma do erro* $\|f_{exato} - f_{\lambda^*}^{(j)}\|_2$ *permanece constante para* $k \leq j \leq n$.

Demonstração. A demonstração pode ser encontrada em [10, 20].

□

Exibimos na Figura 2.8 uma ilustração deste teorema e podemos apreciar que o fenômeno da estabilização se inicia na mesma iteração em que os pontos fixos estabilizam.

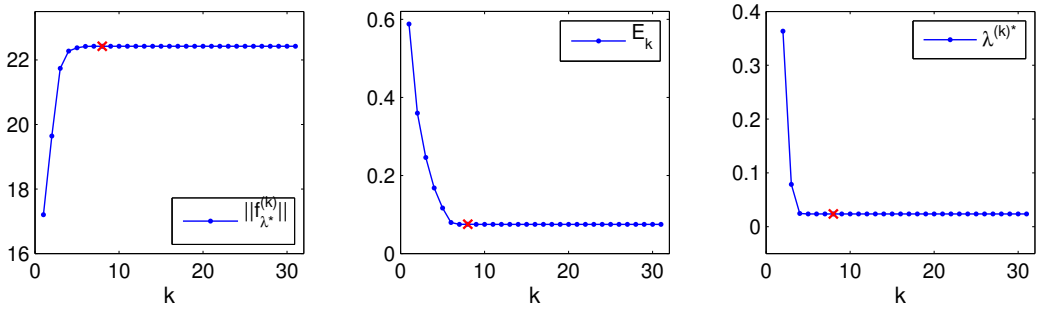


Figura 2.8: Evolução para $\|f_{\lambda^*}^{(k)}\|_2$ e $E_k = \|f_{\text{exato}} - f_{\lambda^*}^{(k)}\|_2 / \|f_{\text{exato}}\|_2$ com $\lambda^* = \lambda^{(8)^*}$ (esquerda e centro). Parâmetro $\lambda^{(k)^*}$ obtido em cada passo GKB (direita) para o problema shaw com $n = 512$ e NL = 0,01.

No Algoritmo 2.2 apresentamos a descrição do algoritmo GKB-FP. Como critério de parada, Bazán e Borges [10] e Borges [20], sugerem parar as iterações quando a variação relativa entre dois pontos fixos consecutivos é pequena

$$|\lambda^{(k+1)^*} - \lambda^{(k)^*}| < \varepsilon_1 |\lambda^{(k)^*}|, \tag{2.25}$$

em que ε_1 é um pequeno parâmetro de tolerância, por exemplo $\varepsilon_1 = 10^{-6}$. A desvantagem do critério de parada (2.25) é que a sequência de pontos fixos $\lambda^{(k)^*}$ pode diminuir muito lentamente. Para contornar estas dificuldades, outro critério de parada foi introduzido

$$|\lambda^{(k+1)^*} - \lambda^{(k)^*}| < \varepsilon_2 |\lambda^{(0)^*}|, \tag{2.26}$$

Entrada: A, g, $p_0 > 1$, $k_{\max}$, ε.
Saída: Solução regularizada $f_{\lambda^*}^{(k)}$

1. Aplicamos p_0 passos GKB em A com vetor inicial g e formamos a matriz B_{p_0}.
2. Definimos $k = p_0$, $\mu = 1$. Calculamos o ponto fixo ${\lambda^{(k)}}^*$ de $\phi^{(k)}(\lambda;\mu)$ e definimos$\lambda_0 = {\lambda^{(k)}}^*$, $\lambda_{\text{old}} = \lambda_0$, $k \leftarrow k+1$.
3. Realizamos mais um passo GKB e calculamos o ponto fixo ${\lambda^{(k)}}^*$ de $\phi^{(k)}(\lambda;\mu)$ considerando λ_0 como aproximação inicial.
Definimos $\lambda_{\text{old}} = \lambda_0$, $\lambda_0 = {\lambda^{(k)}}^*$.
4. **Se** critério de parada satisfeito **faça**
$\lambda^* = \lambda_{\text{old}}$
senão faça
$k \leftarrow k+1$
Vá para passo **3.**
Fim se
5. Calculamos a solução regularizada $f_{\lambda^*}^{(k)}$.

Algoritmo 2.2: Algoritmo GKB-FP.

em que ${\lambda^{(0)}}^*$ é o ponto fixo calculado no passo 2 e ε_2 é outro parâmetro de tolerância. Claramente, nada nos impede de usar $\varepsilon_2 = \varepsilon_1$.

Finalizamos esta seção com a observação de que precondicionadores [106] podem ser incorporados durante o processo GKB e em [10, 20], encontramos diversos resultados numéricos que contemplam este caso.

2.4 GKB-FP como método de projeção

Nosso objetivo é entender melhor as propriedades de convergência do algoritmo GKB-FP e, para isto, vamos nos concentrar na questão sobre quão bem a solução regularizada f_λ pode ser aproximada "projetando" o problema de grande porte sobre um subespaço de dimensão pequena. Assumiremos que $\{\mathcal{V}_k\}_{k\geq 1}$ é uma família de subespaços de dimensão finita tal que $\mathcal{V}_1 \subset \mathcal{V}_2 \subset \cdots \subset \mathcal{V}_k \subset \cdots$, e para $k > 1$, consideraremos aproximações $f_\lambda^{(k)}$ definidas por

$$f_\lambda^{(k)} = \underset{f\in\mathcal{V}_k}{\operatorname{argmin}} \left\{ \|g - Af\|_2^2 + \lambda^2 \|Lf\|_2^2 \right\}. \tag{2.27}$$

Podemos notar que para $L = I_n$ e $\mathcal{V}_k = \mathcal{K}_k(A^TA, A^Tg)$ temos o algoritmo GKB-FP.

Para a análise que desenvolveremos a seguir, utilizaremos o número $\gamma_k = \|A - AP_k\|_2$, em

que P_k denota o projetor ortogonal sobre o subespaço $\mathcal{V}_k$. Este número mede a qualidade da aproximação do operador AP_k com relação ao operador A, que é fundamental para estimar a qualidade da aproximação $f_\lambda^{(k)}$. O lema a seguir fornece uma simples expressão para γ_k.

Lema 2.2. *Seja $V_k = [v_1, \ldots, v_k] \in \mathbb{R}^{n\times k}$ uma matriz com colunas ortonormais tal que span$\{v_1, \ldots, v_k\} = \mathcal{V}_k$. Então existe uma matriz $M = [M_1 \;\; M_2] \in \mathbb{R}^{m\times n}$ com $M_1 \in \mathbb{R}^{m\times k}$ e $M_2 \in \mathbb{R}^{m\times(n-k)}$ tal que*

$$\gamma_k = \|M_2\|_2. \tag{2.28}$$

Demonstração. Como V_k tem k colunas, podemos encontrar uma matriz $V_\perp \in \mathbb{R}^{n\times(n-k)}$ tal que $V = [V_k \;\; V_\perp]$ seja ortogonal. Vamos definir a matriz $\overline{U} = AV$.

Podemos decompor $\overline{U}$ em $\overline{U} = UM$ em que $U \in \mathbb{R}^{m\times m}$ é uma matriz ortogonal e $M = [M_1 \;\; M_2] \in \mathbb{R}^{m\times n}$. Disto segue que $AV_k = UM_1$ e, portanto,

$$\gamma_k = \|A - AP_k\|_2 = \|A - AV_kV_k^T\|_2 = \|M - M_1V_k^T[V_k \;\; V_\perp]\|_2 = \|M_2\|_2. \tag{2.29}$$

□

No contexto do algoritmo GKB-FP em que os subespaços $\mathcal{V}_k$ são subespaços de Krylov $\mathcal{K}_k(A^TA, A^Tg)$, vamos considerar que o processo GKB possa ser executado até o fim, isto é, podemos executar n passos sem interrupções, pelo menos em aritmética exata. Isto é razoável em problemas mal-postos discretos pois os valores singulares decaem a zero gradualmente. Assim, temos o seguinte resultado.

Teorema 2.3. *Assuma que o processo GKB possa ser executado até o fim. Seja G_k a matriz $(n-k+1)\times(n-k)$ bidiagonal inferior dada por*

$$G_k = \begin{pmatrix} \alpha_{k+1} & & & \\ \beta_{k+2} & \alpha_{k+2} & & \\ & \ddots & \ddots & \\ & & \beta_n & \alpha_n \\ & & & \beta_{n+1} \end{pmatrix}, \tag{2.30}$$

em que α_k e β_k são gerados pelo processo de bidiagonalização de Golub-Kahan aplicado à matriz A. Então para $k = 1, \ldots, n-1$ temos

1. $\gamma_k = \|G_k\|_2$,

2. $\gamma_{k+1} \leq \gamma_k$.

Demonstração. O primeiro item segue como consequência do Lema 2.2 e o segundo item pelo fato de G_{k+1} ser uma submatriz de G_k.

□

Como usualmente os elementos α_k e β_k diminuem de valor quando o número de iterações aumenta, a estrutura especial de G_k nos leva a concluir:

1. o número γ_k pode ser aproximado usando a norma 2 da primeira coluna da matriz G_k, isto é, se e_1 denota o vetor canônico em $\mathbb{R}^{(n-k)}$, então

$$\gamma_k \gtrapprox \|G_k e_1\|_2 = \sqrt{\alpha_{k+1}^2 + \beta_{k+2}^2}; \tag{2.31}$$

2. se $\text{posto}(AP_k) = k$, então

$$\gamma_k \geq \sigma_{k+1}, \tag{2.32}$$

   e a igualdade é atingida se o subespaço $\mathcal{V}_k$ é gerado pelos k primeiros vetores singulares a direita da matriz A. Neste caso, AP_k é a matriz de posto k mais próxima de A na norma 2 (veja, por exemplo, Golub e Van Loan [51, Teo. 2.5.3]). Na prática, pelo fato do subespaço $\mathcal{V}_k$ ter informações relevantes dos k primeiros vetores singulares a direita, ambos γ_k e $\sqrt{\alpha_{k+1}^2 + \beta_{k+1}^2}$ aproximam o valor singular σ_{k+1}, e a qualidade desta aproximação melhora quando k aumenta. Na Figura 2.9 podemos ver uma ilustração desta observação para os problemas phillips e heat.

O lema a seguir fornece uma estimativa para a norma (ou seminorma) entre a solução do problema de grande porte (2.1) e a solução do sistema restrito (2.27).

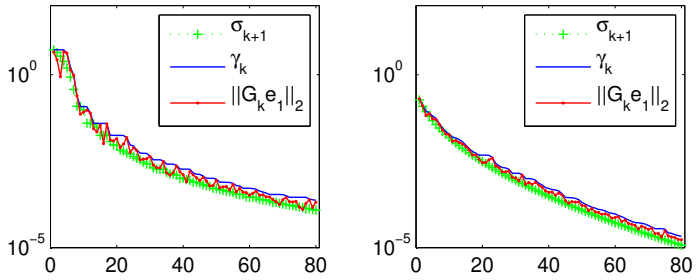


Figura 2.9: Estimativas (2.31) e (2.32) para os problemas phillips (esquerda) e heat (esquerda) com $n = 512$ e NL = 0,005.

Lema 2.4. *Sejam P_k o projetor ortogonal sobre $\mathcal{V}_k$, $\gamma_k = \|A - AP_k\|_2$ e $\delta_k = \|L - LP_k\|_2$. Então*

$$\|Lf_\lambda - Lf_\lambda^{(k)}\|_2 \leq \sqrt{\delta_k^2 + \frac{\gamma_k^2}{\lambda^2}}\|(I - P_k)f_\lambda\|_2. \tag{2.33}$$

Demonstração. Como $\mathcal{N}(A) \cap \mathcal{N}(L) = 0$, a função $<,>_\#: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, definida por

$$< u, v >_\# = < Au, Av > + \lambda^2 < Lu, Lv >, \tag{2.34}$$

é um produto interno em $\mathbb{R}^n$, em que $< \cdot, \cdot >$ denota o produto interno usual do $\mathbb{R}^n$. Agora, usando o fato de que

$$\left[\begin{pmatrix} g \\ 0 \end{pmatrix} - \begin{pmatrix} A \\ \lambda L \end{pmatrix} f_\lambda \right] \perp \mathcal{R} \begin{pmatrix} A \\ \lambda L \end{pmatrix}, \tag{2.35}$$

temos

$$< Af_\lambda - g, Af > + \lambda^2 < Lf_\lambda, Lf > = 0, \quad \forall f \in \mathbb{R}^n. \tag{2.36}$$

Similarmente, para o problema de minimização restrito (2.27) temos

$$< Af_\lambda^{(k)} - g, Af > + \lambda^2 < Lf_\lambda^{(k)}, Lf > = 0, \quad \forall f \in \mathcal{V}_k. \tag{2.37}$$

Podemos combinar estas duas últimas equações e obter

$$< f_\lambda - f_\lambda^{(k)}, f >_\# = 0, \quad \forall f \in \mathcal{V}_k. \tag{2.38}$$

Isto significa que $f_\lambda^{(k)} = \mathcal{P}_k f_\lambda$, em que $\mathcal{P}_k$ denota o projetor ortogonal sobre o subespaço $\mathcal{V}_k$ com respeito ao produto interno $< \cdot, \cdot >_\#$.

Seja $\| \cdot \|_\#$ a norma induzida pelo produto interno $< \cdot, \cdot >_\#$, então,

$$\begin{aligned} \|f_\lambda - f_\lambda^{(k)}\|_\#^2 &= \|f_\lambda - \mathcal{P}_k f_\lambda\|_\#^2 \\ &\leq \|f_\lambda - P_k f_\lambda\|_\#^2 \\ &= \|A(I_n - P_k)f_\lambda\|_2^2 + \lambda^2 \|L(I_n - P_k)f_\lambda\|_2^2 \\ &\leq \|A(I_n - P_k)\|_2^2 \|(I_n - P_k)f_\lambda\|_2^2 + \lambda^2 \|L(I_n - P_k)\|_2^2 \|(I_n - P_k)f_\lambda\|_2^2 \\ &= (\gamma_k^2 + \lambda^2 \delta_k^2) \|(I_n - P_k)f_\lambda\|_2^2. \end{aligned} \tag{2.39}$$

O resultado do lema segue usando a desigualdade

$$\lambda^2 \|L(f_\lambda - f_\lambda^{(k)})\|_2^2 \leq \|A(f_\lambda - f_\lambda^{(k)})\|_2^2 + \lambda^2 \|L(f_\lambda - f_\lambda^{(k)})\|_2^2 = \|f_\lambda - f_\lambda^{(k)}\|_\#^2. \tag{2.40}$$

□

Sejam u_i e v_i os vetores singulares da matriz A conforme equação (1.5). Vamos definir

$$\mathsf{U}_k = [\mathsf{u}_1, \ldots, \mathsf{u}_k], \quad \mathsf{V}_k = [\mathsf{v}_1, \ldots, \mathsf{v}_k], \quad \mathcal{S}_k = \text{span}\{\mathsf{v}_1, \ldots, \mathsf{v}_k\}. \tag{2.41}$$

Quando $L = I_n$, a solução regularizada f_λ, em termos da SVD de A, é dada por

$$f_\lambda = \sum_{i=1}^{n} \frac{\sigma_i^2}{\sigma_i^2 + \lambda^2} \frac{\mathsf{u}_i^T g}{\sigma_i} \mathsf{v}_i. \tag{2.42}$$

Com isto temos condições de provar o resultado mais importante desta seção.

Teorema 2.5. *Sejam P_k e P_k os projetores ortogonais sobre $\mathcal{V}_k$ e $\mathcal{S}_k$. Seja Ω_k o ângulo (veja apêndice B) entre os subespaços $\mathcal{V}_k$ e $\mathcal{S}_k$. Então, para $\lambda > 0$, vale*

$$\|(I_n - P_k)f_\lambda\|_2 \leq \text{sen}(\Omega_k)\|f_\lambda\|_2 + \gamma_k \frac{\|\tilde{g}_k\|_2}{\lambda^2}, \tag{2.43}$$

em que $\tilde{g}_k = g - \mathsf{U}_k \mathsf{U}_k^T g$. Como consequência, para $L = I_n$ e o parâmetro de regularização

escolhido pelo método de ponto fixo, denotado por λ_{FP}, temos

$$\frac{\|f_{\lambda_{FP}} - f_{\lambda_{FP}}^{(k)}\|_2}{\|f_{\lambda_{FP}}\|_2} \leq \sqrt{1 + \frac{\gamma_k^2}{\lambda_{FP}^2}} \left[\text{sen}(\Omega_k) + \frac{\gamma_k}{\lambda_{FP}} \frac{\|\tilde{g}_k\|_2}{\|r_{\lambda_{FP}}\|_2}\right]. \tag{2.44}$$

Demonstração. Podemos escrever a solução regularizada f_λ como $f_\lambda = \mathsf{P}_k f_\lambda + (I_n - \mathsf{P}_k) f_\lambda$. Multiplicando ambos os lados por $(I_n - P_k)$ temos

$$\|(I_n - P_k) f_\lambda\|_2 \leq \|(I_n - P_k)\mathsf{P}_k f_\lambda\|_2 + \|(I_n - P_k)(I_n - \mathsf{P}_k) f_\lambda\|_2. \tag{2.45}$$

O primeiro termo pode ser limitado por

$$\|(I_n - P_k)\mathsf{P}_k f_\lambda\|_2 \leq \text{sen}(\Omega_k)\|f_\lambda\|_2, \tag{2.46}$$

em que usamos o fato de que $\|(I_n - P_k)\mathsf{P}_k\|_2 = \text{sen}(\Omega_k)$ (veja, por exemplo, [51]). Para estimar o segundo termo, notemos que

$$(I_n - \mathsf{P}_k) f_\lambda = \sum_{i=k+1}^{n} \frac{\sigma_i(\mathsf{u}_i^T g)}{\sigma_i^2 + \lambda^2} \mathsf{v}_i = (A^T A + \lambda^2 I_n)^{-1} A^T \tilde{g}_k = A^T (AA^T + \lambda^2 I_m)^{-1} \tilde{g}_k, \tag{2.47}$$

em que usamos a SVD de A para obter $(A^T A + \lambda^2 I_n)^{-1} A^T = A^T (AA^T + \lambda^2 I_m)^{-1}$. Então

$$\|(I_n - P_k)(I_n - \mathsf{P}_k) f_\lambda\|_2 = \|(I_n - P_k) A^T (AA^T + \lambda^2 I_m)^{-1} \tilde{g}_k\|_2 \leq \gamma_k \frac{\|\tilde{g}_k\|_2}{\lambda^2}. \tag{2.48}$$

E assim, a primeira desigualdade segue das equações (2.45), (2.46) e (2.48).

Para provarmos a segunda parte, ou seja, para o caso do parâmetro de regularização ser escolhido pelo método FP, isto é, $\lambda_{\text{FP}} = \|r_{\lambda_{\text{FP}}}\|_2 / \|f_{\lambda_{\text{FP}}}\|_2$, temos

$$\frac{\|(I_n - P_k) f_{\lambda_{\text{FP}}}\|_2}{\|f_{\lambda_{\text{FP}}}\|_2} \leq \left(\text{sen}(\Omega_k) + \frac{\gamma_k}{\lambda_{\text{FP}}} \frac{\|\tilde{g}_k\|_2}{\|r_{\lambda_{\text{FP}}}\|_2}\right). \tag{2.49}$$

E o resultado do teorema segue imediatamente desta desigualdade.

□

O Teorema 2.5 mostra que se o subespaço $\mathcal{V}_k$ é uma boa aproximação para o subespaço $\mathcal{S}_k$,

o que significa que sen(Ω_k) é pequeno, então a solução para o problema (2.27) será uma boa aproximação para $f_{\lambda_{\text{FP}}}$ desde que a razão $\gamma_k/\lambda_{\text{FP}}$ seja suficientemente pequena. Ilustraremos isto calculando o erro relativo $\|(I_n - P_l)f_{\lambda_{\text{FP}}}\|_2/\|f_{\lambda_{\text{FP}}}\|_2$ e a cota (2.49) para o caso em que $\mathcal{V}_l$ coincide com $\mathcal{S}_l$, bem como aproximações para este subespaço. Para isto, vamos considerar, novamente, os problemas phillips e heat e, para k fixo, os subespaços $\widetilde{\mathcal{S}}_l$, $l = 1, \ldots, k$, gerados pelas primeiras l colunas da matriz $\widetilde{\mathsf{V}}_k = V_k\mathbf{v}$, em que a matriz V_k é gerada pelo processo GKB e $\mathbf{v} \in \mathbb{R}^{k\times k}$ é a matriz composta pelos vetores singulares a direita de B_k (também do processo GKB). Para os primeiros valores de l, os subespaços $\widetilde{\mathcal{S}}_l$ aproximam bem $\mathcal{S}_l$ e esta aproximação deteriora quando l se aproxima de k (veja, por exemplo, [65, cap. 6]). Na Figura 2.10 exibimos estas quantidades para $k = 80$.

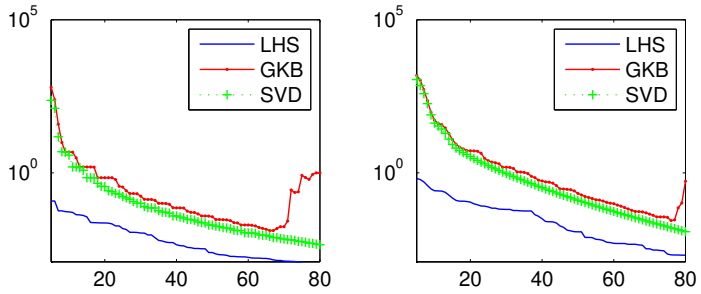


Figura 2.10: Erro relativo e estimativa (2.49) para os problemas phillips (esquerda) e heat (direita) com $n = 512$ e NL = 0,005. Aqui LHS (*left-hand side*) indica o lado esquerdo da equação (2.49).

Assim, para que $f_\lambda^{(k)}$ seja uma boa aproximação para f_λ, as seguinte condições devem ser atendidas:

1. o subespaço $\mathcal{V}_k$ tem que aproximar bem o subespaço $\mathcal{S}_k$ para que sen(Ω_k) seja pequeno;

2. o quociente γ_k^2/λ^2 deve ser o menor possível.

Portanto, a distância entre $Lf_\lambda^{(k)}$ e Lf_λ depende de quão pequenos δ_k e γ_k são e, claro, da escolha do parâmetro de regularização.

2.5 Extensão de GKB-FP para o método de regularização de Tikhonov na forma geral

Nesta seção propomos algumas extensões de GKB-FP para o método de regularização de Tikhonov na forma geral, concentrando, em particular, em problemas que envolvem seminormas $\|Lf\|_2$ em que a matriz L tem posto coluna incompleto e com muito mais linhas do que colunas, isto acontece frequentemente em problemas do tipo *deblurring*. Duas abordagens distintas são propostas:

i) o problema na forma geral pode ser, teoricamente, transformado para a forma padrão e ser resolvido eficientemente por GKB-FP e;

ii) construir uma sequência de soluções regularizadas resolvendo o problema restrito (2.27).

Teoricamente podemos transformar o problema na forma geral (2.1) para a forma padrão

$$\overline{f}_\lambda = \underset{\overline{f}\in\mathbb{R}^p}{\operatorname{argmin}} \left\{ \|\overline{g} - \overline{Af}\|_2^2 + \lambda^2 \|\overline{f}\|_2^2 \right\}, \tag{2.50}$$

sendo que esta transformação pode ser dividida em dois casos. Se a matriz L é não-singular, a transformação é simplesmente $\overline{f} = Lf \Leftrightarrow f = L^{-1}\overline{f}$, $\overline{g} = g$ e $\overline{A} = AL^{-1}$. Porém, se L é singular ou retangular, então a transformação é dada por

$$\overline{A} = AL_A^\dagger, \quad \overline{g} = g - Af_N, \quad f_\lambda = L_A^\dagger \overline{f}_\lambda + f_N, \tag{2.51}$$

em que a matriz

$$L_A^\dagger = \left(I_n - \left(A\left(I_n - L^\dagger L \right) \right)^\dagger A \right) L^\dagger, \tag{2.52}$$

é conhecida como Inversa Generalizada de L com peso A (veja [42]), e f_N é a componente da solução que pertence ao espaço nulo de L. Portanto, a ideia é aplicar o algoritmo GKB-FP no problema transformado (2.50) e, uma vez encontrada $\overline{f}_\lambda$, ou uma aproximação $\overline{f}_\lambda^{(k)}$, esta deve ser retornada para o cenário original, ou seja, $f_\lambda = L_A^\dagger \overline{f}_\lambda + f_N$, conforme Algoritmo 2.3.

Teoricamente podemos construir a matriz $\overline{A}$. Contudo, em problemas de grande porte, esta não é uma alternativa viável. Assim, a eficiência desta abordagem depende de como as

Entrada: A, L, g, $p_0 > 1$, $k_{\max}$, ε.
Saída: Solução regularizada $f_{\lambda^*}^{(k)}$

1. Aplicamos p_0 passos GKB em $\overline{A}$ com vetor inicial $\overline{g}$ e formamos a matriz B_{p_0}.
2. Definimos $k = p_0$, $\mu = 1$. Calculamos o ponto fixo $\lambda^{(k)^*}$ de $\phi^{(k)}(\lambda;\mu)$ e definimos $\lambda_0 = \lambda^{(k)^*}$, $\lambda_{\text{old}} = \lambda_0$, $k \leftarrow k+1$.
3. Realizamos mais um passo GKB e calculamos o ponto fixo $\lambda^{(k)^*}$ de $\phi^{(k)}(\lambda;\mu)$ considerando λ_0 como aproximação inicial.
Definimos $\lambda_{\text{old}} = \lambda_0$, $\lambda_0 = \lambda^{(k)^*}$.
4. **Se** critério de parada satisfeito **faça**
$\lambda^* = \lambda_{\text{old}}$
senão faça
$k \leftarrow k+1$
Vá para passo **3.**
Fim se
5. Calculamos a solução regularizada $\overline{f}_{\lambda^*}^{(k)}$.
6. Transformamos a solução $\overline{f}_{\lambda^*}^{(k)}$ para o cenário original, ou seja, $f_{\lambda^*}^{(k)} = L_A^\dagger \overline{f}_{\lambda^*}^{(k)} + f_N$.

Algoritmo 2.3: Algoritmo GKB-FP aplicado ao método de regularização de Tikhonov na forma geral.

multiplicações matriz-vetor envolvendo $\overline{A}$ são efetuadas.

2.5.1 GKB-FP aplicado ao problema transformado à forma padrão

Como a eficiência desta abordagem depende de como os produtos matriz-vetor envolvendo a matriz $\overline{A} = AL_A^\dagger$ são realizados, vamos assumir que seja dada uma matriz W tal que $\text{span}(W) = \mathcal{N}(L)$. Neste caso, $L_A^\dagger$ e f_N são dadas por

$$L_A^\dagger = \left(I_n - W\left(AW\right)^\dagger A\right) L^\dagger, \quad f_N = W\left(AW\right)^\dagger g. \tag{2.53}$$

Como geralmente $\mathcal{N}(L)$ é um subespaço de dimensão pequena, o custo para o cálculo de AW em (2.53) é relativamente pequeno. Sendo assim, o principal custo computacional fica por conta da estrutura de L e do modo como os produtos matriz-vetor envolvendo a matriz $L^\dagger$ são realizados. O produto matriz-vetor $\dot{u} = L^\dagger u$ é equivalente a

$$\dot{u} = \underset{t\in\mathbb{S}}{\text{argmin}} \|t\|_2, \quad \mathbb{S} = \left\{ t \in \mathbb{R}^n \,\middle/\, t = \underset{t\in\mathbb{R}^n}{\text{argmin}} \|u - Lt\|_2 \right\}, \tag{2.54}$$

ou seja, queremos a solução de norma mínima para o problema

$$t = \underset{t \in \mathbb{R}^n}{\operatorname{argmin}} \|u - Lt\|_2. \tag{2.55}$$

Uma observação análoga vale para $\dot{v} = L^{\dagger T} v$. O lema a seguir caracteriza o vetor $\dot{u} = L^{\dagger} u$ em termos de qualquer solução do problema de minimização (2.55) e do subespaço $\mathcal{N}(L)$. Este lema é necessário para a eficiência exigida para que o algoritmo GKB-FP seja competitivo quando aplicado ao método de regularização de Tikhonov na forma geral.

Lema 2.6. *Sejam a matriz de regularização $L \in \mathbb{R}^{p \times n}$ com $p \geq n \geq \mathsf{q} = posto(L)$, W uma matriz com colunas ortonormais tal que $span(W) = \mathcal{N}(L)$ e $u \in \mathbb{R}^p$. Então*

$$L^{\dagger} u = t_u - W W^T t_u, \tag{2.56}$$

em que t_u é qualquer solução de (2.55).

Demonstração. Seja $L = U_L \Sigma_L V_L^T$ a SVD da matriz L e $\mathsf{q} = \text{posto}(L)$, então

$$\begin{aligned} \|u - Lt\|_2^2 &= \|u - U_L \Sigma_L V_L^T t\|_2^2 = \|U_L^T u - \Sigma_L \tilde{t}\|_2^2 \\ &= \sum_{i=1}^{\mathsf{q}} \left(u_{L_i}^T u - \sigma_{L_i} \tilde{t}_i\right)^2 + \sum_{i=\mathsf{q}+1}^{p} \left(u_{L_i}^T u\right)^2. \end{aligned} \tag{2.57}$$

Assim, devemos ter $u_{L_i}^T u - \sigma_{L_i} \tilde{t}_i = 0$ para $i = 1, \ldots, \mathsf{q}$, para que $\|u - Lt\|_2^2$ seja minimizado, logo $\tilde{t}_i = u_{L_i}^T u / \sigma_{L_i}$. Para $i = \mathsf{q}+1, \ldots, n$ temos que $\tilde{t}_i \in \mathbb{R}$, portanto, a solução t_u é

$$t_u = V_L \tilde{t} = \sum_{i=1}^{n} \tilde{t}_i v_{L_i} = \sum_{i=1}^{\mathsf{q}} \tilde{t}_i v_{L_i} + \sum_{i=\mathsf{q}+1}^{n} \tilde{t}_i v_{L_i} = L^{\dagger} u + w, \tag{2.58}$$

no qual $w \in \mathcal{N}(L)$, pois sabemos que as colunas $i = \mathsf{q}+1, \ldots, n$, de V_L formam uma base para $\mathcal{N}(L)$. Então, se temos uma solução t_u, devemos remover a componente deste vetor que pertence ao $\mathcal{N}(L)$. Logo, como $\text{span}(W) = \mathcal{N}(L)$ temos que a solução de norma mínima é

$$L^{\dagger} u = t_u - W W^T t_u. \tag{2.59}$$

□

Vamos considerar novamente que $L \in \mathbb{R}^{p\times n}$ com $p \geq n$ mas com posto coluna completo, ou seja, posto$(L) = n$ e um vetor $v \in \mathbb{R}^n$. Assim, encontrar o vetor $\dot{v} = L^{\dagger^T} v$ é equivalente a encontrar a solução de norma mínima do problema

$$t_v = \underset{t\in\mathbb{R}^p}{\operatorname{argmin}} \|v - L^T t\|_2^2, \tag{2.60}$$

visto que $L^{\dagger^T} = L^{T^\dagger}$. Infelizmente não dispomos de uma base para $\mathcal{N}(L^T)$, logo não podemos utilizar o Lema 2.6, portanto, devemos encontrar outra alternativa para resolver (2.60). Como este sistema contém mais variáveis do que equações, podemos resolver (2.60) escrevendo a solução t_v como $t_v = Ly_v$ para algum $y_v \in \mathbb{R}^n$, logo

$$y_v = \underset{y\in\mathbb{R}^n}{\operatorname{argmin}} \|v - L^T L y\|_2^2, \tag{2.61}$$

e a solução t_v é obtida diretamente de $t_v = Ly_v$ após (2.61) ter sido resolvido.

Assim, para GKB-FP ser bem sucedido quando aplicado ao método de regularização de Tikhonov na forma geral, o problema (2.54) deve ser resolvido eficientemente. Na busca por esta eficiência, podemos tentar aplicar métodos como CGLS, LSQR ou sua versão precondicionada por subespaços [78]. Infelizmente, nossas experiências computacionais com estas técnicas com $\|Lf\|_2$ sendo uma norma de Sobolev têm sido insatisfatórias devido à baixa velocidade de convergência das iteradas, ou seja, foram necessárias muitas iterações para conseguirmos soluções de boa qualidade. No entanto, para aplicações envolvendo matrizes esparsas, matrizes de regularização com pequena largura de banda, entre outras, as seguintes abordagens podem ser consideradas:

Abordagem baseada na decomposição LU

Se a matriz L tem posto incompleto, digamos, $p \geq n > \mathsf{q} = \text{posto}(L)$, podemos encontrar matrizes $P_L \in \mathbb{R}^{p\times p}$ e $Q_L \in \mathbb{R}^{n\times n}$ tais que

$$P_L L Q_L = \hat{L}\hat{U}, \tag{2.62}$$

em que

$$\hat{L} = \begin{bmatrix} \hat{L}_1 \\ \hat{L}_2 \end{bmatrix} \in \mathbb{R}^{p\times \mathtt{q}}, \quad \hat{U} = \begin{bmatrix} \hat{U}_1 & \hat{U}_2 \end{bmatrix} \in \mathbb{R}^{\mathtt{q}\times n}, \tag{2.63}$$

com $\hat{L}_1 \in \mathbb{R}^{\mathtt{q}\times\mathtt{q}}$ triangular inferior e $\hat{U}_1 \in \mathbb{R}^{\mathtt{q}\times\mathtt{q}}$ triangular superior, ambas com posto $\mathtt{q}$. Então, resolver o problema de minimização (2.54) é equivalente a resolver dois subproblemas:

$$y_{\mathrm{LS}} = \operatorname{argmin} \|\hat{L}y - P_L u\|_2, \quad t_{\mathrm{LS}} = \operatorname{argmin} \|\hat{U}t - y_{\mathrm{LS}}\|_2. \tag{2.64}$$

O primeiro tem posto coluna completo, logo existe única solução e o segundo problema devemos resolver de acordo com as equações (2.60) e (2.61). Quando t_{LS} é determinado, temos $\dot{u} = L^{\dagger}u = Q_L t_{\mathrm{LS}}$. O raciocínio análogo vale para $\dot{v} = L^{\dagger^T}v$. Esta abordagem torna-se atrativa especialmente quando a matriz L é de banda e a largura da banda é pequena. Neste caso, os dois subproblemas podem ser tratados de modo muito eficaz como exigido para a eficiência de GKB-FP. No apêndice C apresentamos como esta abordagem pode ser eficientemente implementada através da fórmula de Sherman-Morrison-Woodbury para o caso em que $\dim(\mathcal{N}(L)) = 1$.

Abordagem baseada na matriz L_D

Neste caso, vamos considerar matrizes de regularização L da forma

$$L = \begin{bmatrix} L_{d_1} \otimes I_{\tilde{m}} \\ I_{\tilde{n}} \otimes L_{d_2} \end{bmatrix}, \tag{2.65}$$

em que $L_{d_1} \in \mathbb{R}^{(\tilde{n}-d_1)\times\tilde{n}}$, $L_{d_2} \in \mathbb{R}^{(\tilde{m}-d_2)\times\tilde{m}}$ e $(\cdot)\otimes(\cdot)$ denota o produto de Kronecker entre $(\cdot)$ e $(\cdot)$ (veja [77, Seção 4.2]). No apêndice A fornecemos algumas propriedades básicas referentes aos produtos de Kronecker. A ideia central é que se usarmos a SVD das matrizes pequenas L_{d_1} e L_{d_2}, $L_{d_1} = U_{d_1}\Sigma_{d_1}V_{d_1}^T$ e $L_{d_2} = U_{d_2}\Sigma_{d_2}V_{d_2}^T$, então a seminorma $\|Lf\|_2$ satisfaz a propriedade

$$\|Lf\|_2 = \|L_D f\|_2, \quad \text{com} \quad L_D = D(V_{d_1} \otimes V_{d_2})^T, \tag{2.66}$$

e D uma matriz diagonal, com os elementos da diagonal não-negativos, que satisfaz [69]

$$D^2 = \Sigma_{d_1}^T \Sigma_{d_1} \otimes I_{\tilde{m}} + I_{\tilde{n}} \otimes \Sigma_{d_2}^T \Sigma_{d_2}. \tag{2.67}$$

O subespaço $\mathcal{N}(L_D)$ é gerado pelas colunas da matriz $V_{d_1} \otimes V_{d_2}$ associadas aos elementos nulos de D (veja [69] novamente). Como consequência, podemos resolver o problema de minimização transformado para a forma padrão (2.50) associado a

$$f_\lambda = \underset{f \in \mathbb{R}^n}{\operatorname{argmin}} \left\{ \|g - Af\|_2^2 + \lambda^2 \|L_D f\|_2^2 \right\}, \tag{2.68}$$

aproveitando o fato de que a matriz $V_{d_1} \otimes V_{d_2}$ é ortogonal. A principal vantagem desta abordagem é que os produtos matriz-vetor $L_D^\dagger u$ e ${L_D^\dagger}^T v$ em (2.53) podem ser implementados de modo muito eficiente por

$$L_D^\dagger u = (V_{d_1} \otimes V_{d_2}) D^\dagger u, \quad {L_D^\dagger}^T v = D^\dagger (V_{d_1} \otimes V_{d_2})^T v. \tag{2.69}$$

Portanto, o custo computacional associado ao problema (2.54) reduz essencialmente a um produto matriz-vetor envolvendo o produto de Kronecker $V_{d_1} \otimes V_{d_2}$, e isto é importante para a eficiência do algoritmo GKB-FP. Ainda mais, se $A = A_1 \otimes A_2$, como ocorre em processamento de imagens e análise numérica (veja, por exemplo, [68]), podemos aumentar a eficiência do seguinte modo. Lembrando que a matriz $V_{d_1} \otimes V_{d_2}$ é ortogonal, vamos considerar a transformação

$$\check{f} = (V_{d_1} \otimes V_{d_2})^T f. \tag{2.70}$$

Então (2.68) reduz a

$$\check{f}_\lambda = \underset{\check{f} \in \mathbb{R}^n}{\operatorname{argmin}} \left\{ \|g - \check{A}\check{f}\|_2^2 + \lambda^2 \|D\check{f}\|_2^2 \right\}, \tag{2.71}$$

em que

$$\check{A} = (A_1 V_{d_1}) \otimes (A_2 V_{d_2}), \tag{2.72}$$

segue da propriedade dos produtos de Kronecker $(A_1 \otimes A_2)(V_{d_1} \otimes V_{d_2}) = (A_1 V_{d_1}) \otimes (A_2 V_{d_2})$. Assim, somente as matrizes $(A_1 V_{d_1})$ e $(A_2 V_{d_2})$ precisam ser armazenadas e, portanto, o problema (2.71) pode ser resolvido muito mais eficientemente do que o problema (2.68).

Veremos agora qual o custo computacional exigido pelo algoritmo GKB-FP e pela sua extensão para o método de regularização de Tikhonov na forma geral. O custo computacional por iteração para o GKB-FP é dominado pelos produtos matriz-vetor Av e $A^T u$ que aparecem no processo GKB. Assim, a maior parte do custo computacional está na realização dos produtos matriz-vetor envolvendo a matriz $\overline{A} = AL_A^\dagger$ que, de acordo com a equação (2.53), dependem de como os produtos matriz-vetor $L^\dagger u$ e ${L^\dagger}^T v$ são efetuados.

Vamos nos concentrar no caso em que L é dada por (2.65), $L_{d_1} = L_{d_2} \in \mathbb{R}^{(\tilde{n}-d_1)\times\tilde{n}}$ e, por simplicidade, $\dim(\mathcal{N}(L)) = 1$. Portanto, vamos considerar que a solução f_λ é obtida através do problema (2.68). Além disso, para simplificar nossa análise, vamos considerar que $A \in \mathbb{R}^{\tilde{n}^2\times\tilde{n}^2}$. Para

$$\overline{A}u = AL_A^\dagger u = A(I_n - W(AW)^\dagger A)L_D^\dagger u, \tag{2.73}$$

segue que o produto $\tilde{u} = L_D^\dagger u$, dado por $L_D^\dagger u = (V_{d_1} \otimes V_{d_1})D^\dagger u$, $V_{d_1} \in \mathbb{R}^{\tilde{n}\times\tilde{n}}$, tem um custo de $4\tilde{n}^3 + \tilde{n}^2$ operações (o custo de $\tilde{n}^2$ é para o cálculo de $D^\dagger u$ e $4\tilde{n}^3$ para a realização do produto matriz-vetor $(V_{d_1} \otimes V_{d_1})\tilde{u}$, veja Teorema A.11). Na continuação temos o cálculo de $\tilde{u}_1 = \tilde{u} - W(AW)^\dagger A\tilde{u}$. Como a matriz W contém apenas uma coluna, AW é um vetor e $(AW)^\dagger$ é simples de ser calculado. Por questões de eficiência, o cálculo do vetor linha $(AW)^\dagger A$ é realizado apenas uma única vez, antes do processo GKB, logo não há esforço computacional por iteração para $(AW)^\dagger A$. Como consequência, $(AW)^\dagger A\tilde{u}$ é um escalar obtido pelo produto interno entre $(AW)^\dagger A$ e $\tilde{u}$, custo de $2\tilde{n}^2$ operações e $\tilde{u} - W(AW)^\dagger A\tilde{u}$ precisa de mais $2\tilde{n}^2$ operações. Totalizando $4\tilde{n}^2$ operações para calcular $\tilde{u}_1$. Finalmente, $A\tilde{u}_1$ é um produto matriz-vetor que necessita de mais $2\tilde{n}^4$ operações. Logo, o custo computacional para $\overline{A}u$ é de, $2\tilde{n}^4 + 4\tilde{n}^3 + 5\tilde{n}^2$. O raciocínio análogo vale para $\overline{A}^T v$.

Vamos agora verificar qual o ganho com a mudança de variável quando $A = A_1 \otimes A_2$, dado pelas equações (2.71) - (2.72). Vamos realizar o produto $\overline{A}u = \check{A}D_A^\dagger u$. Para

$$\overline{A}u = \check{A}(I_n - W(\check{A}W)^\dagger \check{A})D^\dagger u, \tag{2.74}$$

com W uma matriz tal que $\text{span}(W) = \mathcal{N}(D)$, segue que o produto $\tilde{u} = D^\dagger u$ tem um custo de $\tilde{n}^2$ operações. Na continuação temos o cálculo de $\tilde{u}_1 = \tilde{u} - W(\check{A}W)^\dagger \check{A}\tilde{u}$. Aplicamos aqui a mesma técnica do caso anterior, isto é, como a matriz W contém apenas uma coluna, o

cálculo do vetor linha $(\check{A}W)^\dagger\check{A}$ é realizado apenas uma única vez, antes do processo GKB, logo não há esforço computacional por iteração para $(\check{A}W)^\dagger\check{A}$. Como consequência, $(\check{A}W)^\dagger\check{A}\tilde{u}$ é um escalar obtido pelo produto interno entre $(\check{A}W)^\dagger\check{A}$ e $\tilde{u}$, custo de $2\tilde{n}^2$ operações e $\tilde{u} - W(\check{A}W)^\dagger\check{A}\tilde{u}$ precisa de mais $2\tilde{n}^2$ operações. Totalizando $4\tilde{n}^2$ operações para calcularmos $\tilde{u}_1$. Finalmente, $\check{A}\tilde{u}_1 = (A_1V_{d_1}) \otimes (A_2V_{d_2})$ é um produto matriz-vetor que necessita de mais $4\tilde{n}^3$ operações. Logo, o custo computacional para $\overline{A}u$ é de $4\tilde{n}^3 + 5\tilde{n}^2$.

Na Tabela 2.1 temos um comparativo entre estas duas abordagens. Podemos perceber que o ganho com a mudança de variável (2.71) - (2.72) é considerável, ou seja, reduzimos $2\tilde{n}^4$ operações. Com isto, podemos concluir que esta abordagem é bastante atrativa. Se comparamos com o custo envolvido no algoritmo GKB-FP original em que $L = I_n$, o custo é de apenas $4\tilde{n}^3$ (deveria ser $2\tilde{n}^4$, mas estamos considerando que $A = A_1 \otimes A_2$), mas nem sempre $L = I_n$ produz bons resultados e por este motivo usamos outros regularizadores.

		L_D	D	GKB-FP ($L = I_{\tilde{n}^2}$)
$\tilde{u} = L^\dagger u$		$4\tilde{n}^3 + \tilde{n}^2$	$\tilde{n}^2$	-
$\tilde{u}_1 = \tilde{u} - W(AW)^\dagger A\tilde{u}$		$4\tilde{n}^2$	$4\tilde{n}^2$	-
$\overline{u} = A\tilde{u}_1$		$2\tilde{n}^4$	$4\tilde{n}^3$	$4\tilde{n}^3$
total		$2\tilde{n}^4 + 4\tilde{n}^3 + 5\tilde{n}^2$	$4\tilde{n}^3 + 5\tilde{n}^2$	$4\tilde{n}^3$

Tabela 2.1: Custo computacional para o produto matriz-vetor $\overline{u} = \overline{A}u = A(I_n - W(AW)^\dagger A)L^\dagger u$ usando a matriz L_D e a matriz D para imagens de tamanho $\tilde{n} \times \tilde{n}$.

2.6 PROJ-L: projetando no subespaço de Krylov associado ao processo GKB

Para evitarmos a transformação para a forma padrão, nosso propósito é construir aproximações para a solução f_λ dadas por (2.27). A extensão de GKB-FP que propomos, resolve o problema restrito (2.27) usando o subespaço de Krylov gerado pelo GKB aplicado à matriz A e segue os mesmos passos do algoritmo GKB-FP. Esta extensão difere da proposta original do algoritmo GKB-FP no sentido de que as soluções aproximadas são calculadas como

$$f_\lambda^{(k)} = V_k d_\lambda^{(k)}, \quad d_\lambda^{(k)} = \underset{d\in\mathbb{R}^k}{\operatorname{argmin}} \left\{ \|g - AV_k d\|_2^2 + \lambda^2 \|LV_k d\|_2^2 \right\}, \tag{2.75}$$

porém, a filosofia do método é preservada, pois, para cada $k > 1$, o ponto fixo de $\phi^{(k)}(\lambda;\mu)$ é sempre calculado. Para deixarmos esta proposta mais interessante, seja $LV_k = Q_k R_k$ a decomposição QR de LV_k. Assim, a solução do problema (2.75) se reduz a

$$d_\lambda^{(k)} = \underset{d\in\mathbb{R}^k}{\operatorname{argmin}} \left\{ \|\beta_1 e_1 - B_k d\|_2^2 + \lambda^2 \|R_k d\|_2^2 \right\}. \tag{2.76}$$

Portanto,

$$\|Lf_\lambda^{(k)}\|_2 = \|R_k d_\lambda^{(k)}\|_2, \quad \|r_\lambda^{(k)}\|_2 = \|\beta_1 e_1 - B_k d_\lambda^{(k)}\|_2, \tag{2.77}$$

e a função $\phi^{(k)}(\lambda;\mu)$ associada a este problema é

$$\phi^{(k)}(\lambda;\mu) = \sqrt{\mu}\frac{\|\beta_1 e_1 - B_k d_\lambda^{(k)}\|_2}{\|R_k d_\lambda^{(k)}\|_2}. \tag{2.78}$$

Deste modo, para $p_0 > 1$ e para $k \geq p_0$, calculamos o maior ponto fixo convexo de $\phi^{(k)}(\lambda;\mu)$ e o processo é repetido até algum critério de parada ser satisfeito.

Por questões de eficiência, os aspectos a seguir devem ser considerados:

1. a aproximação inicial no passo $k+1$ é escolhida como o ponto fixo do passo k;

2. a decomposição QR é feita uma única vez, isto é, no passo $k = p_0$ temos $LV_{p_0} = Q_{p_0} R_{p_0}$ e nos passos subsequentes esta é atualizada, ou seja, a decomposição QR de LV_{k+1} é atualizada a partir da decomposição QR de LV_k.

Para a atualização da QR, notemos que as k primeiras colunas da matriz LV_{k+1} são as mesmas colunas da matriz LV_k, logo se $LV_k = Q_k R_k$ e

$$LV_{k+1} = [LV_k \;\; Lv_{k+1}] = Q_{k+1}R_{k+1} = [Q_1 \; q_2] \begin{bmatrix} R_{11} & R_{12} \\ 0 & r_{22} \end{bmatrix} = [Q_1 R_{11} \;\; Q_1 R_{12} + r_{22} q_2], \tag{2.79}$$

então $LV_k = Q_k R_k = Q_1 R_{11}$ é a decomposição QR de LV_k, pois a matriz Q_1 tem colunas ortonormais e a matriz R_{11} é triangular superior. Além disso, $Q_k = Q_1$ e $R_k = R_{11}$. Finalmente, o vetor R_{12} é obtido por $R_{12} = Q_k^T Lv_{k+1}$ e $r_{22} q_2 = Lv_{k+1} - Q_k R_{12}$. O custo para este algoritmo é um pouco maior do que o processo GKB, ou seja, além das multiplicações matriz-vetor Au e $A^T v$, temos a atualização da decomposição QR que requer os produtos matriz-vetor

$\tilde{v} = Lv_{k+1}$ e $Q_k^T\tilde{v}$, o que significa $2pn$ e $2kp$ operações, respectivamente. O cálculo do escalar r_{22} requer $2p$ operações e o vetor q_2 precisa de $p + 2pk$ operações por iteração.

Terminamos este capítulo apresentando no Algoritmo 2.4 um esboço do algoritmo PROJ-L. Experimentos numéricos apresentamos no Capítulo 5.

Entrada: A, L, g, $p_0 > 1$, $k_{\max}$, ε.
Saída: Solução regularizada $f_{\lambda^*}^{(k)}$

1. Aplicamos p_0 passos GKB em A com vetor inicial g e formamos a matriz B_{p_0}.
2. Definimos $k = p_0$, $\mu = 1$. Calculamos a decomposição QR de LV_k, ou seja, $LV_k = Q_k R_k$.
3. Calculamos o ponto fixo ${\lambda^{(k)}}^*$ de $\phi^{(k)}(\lambda;\mu)$ e definimos $\lambda_0 = {\lambda^{(k)}}^*$, $\lambda_{\text{old}} = \lambda_0$, $k \leftarrow k + 1$.
4. Realizamos mais um passo GKB, atualizamos a decomposição QR de LV_{k+1} e calculamos o ponto fixo ${\lambda^{(k)}}^*$ de $\phi^{(k)}(\lambda;\mu)$ considerando λ_0 como aproximação inicial.
Definimos $\lambda_{\text{old}} = \lambda_0$, $\lambda_0 = {\lambda^{(k)}}^*$.
5. **Se** critério de parada satisfeito **faça**
$\lambda^* = \lambda_{\text{old}}$
senão faça
$k \leftarrow k + 1$
Vá para passo **3.**
Fim se
6. Calculamos a solução regularizada $f_{\lambda^*}^{(k)}$.

Algoritmo 2.4: Algoritmo PROJ-L.

Capítulo 3

Regularização iterativa baseada em métodos de subespaços

O algoritmo GKB-FP discutido no capítulo anterior, requer que um problema de projeção seja resolvido para distintos parâmetros de regularização λ. Contudo, em problemas de grande porte, isto pode se tornar caro caso a solução aproximada não seja encontrada logo nas primeiras iterações. Uma alternativa para o método de regularização de Tikhonov é usar regularização iterativa no qual o número de iterações k representa o papel do parâmetro de regularização. Contudo, apesar das propriedades regularizantes dos métodos iterativos serem relativamente bem compreendidas, determinar um número apropriado de iterações que produza uma boa aproximação para a solução desejada não é uma tarefa fácil. Métodos iterativos para a resolução de problemas de mínimos quadrados visam aproximar a solução f_{LS} através de uma sequência de soluções f_k, $k = 1, 2, \ldots$, geradas através de multiplicações do tipo matriz-vetor envolvendo a matriz A. Como exemplo, temos as iterações de Landweber [40, 91], as iterações de Cimmino [16, 35], a Técnica de Reconstrução Algébrica [40] e os métodos de subspaços como CGLS [75] e MINRES [110], dentre outros [19, 45, 58, 65, 69, 78, 84, 125, 136, 137]. Os métodos iterativos são mais atrativos do que os métodos diretos pois [65]:

- a matriz A só é acessada para a realização de produtos matriz-vetor. Se A é esparsa, qualquer decomposição matricial, especialmente as ortogonais, provavelmente irá destruir esta esparsidade ou qualquer outra estrutura [19];

- as operações envolvidas nos métodos iterativos geralmente são produtos matriz-vetor, operações do tipo saxpy ($f \leftarrow \alpha f + g$), norma de vetores e resolução de sistemas triangulares. Estas operações são simples, rápidas e paralelizáveis;

- quando o produto Af é realizado por uma *black-box*, ou seja, a matriz A não é conhecida, então os métodos iterativos são as únicas opções;

- os métodos precisam ser parados em algum momento, especialmente quando lidamos com problems mal-postos discretos. As soluções iteradas e os resíduos associados podem ser monitorados e usados para a escolha da iteração de parada.

Neste capítulo apresentamos um critério de parada automático para métodos iterativos baseados em subespaços que não necessita de nenhum conhecimento a respeito do nível de ruído.

3.1 Métodos iterativos baseados em subespaços

Dada uma família $\{\mathcal{V}_k\}_{k\geq 1}$ de subespaços tal que $\dim(\mathcal{V}_k) = k$ e $\mathcal{V}_k \subset \mathcal{V}_{k+1}$, a filosofia dos métodos iterativos baseados em subespaços é que cada solução iterada f_k é dada por

$$f_k = \underset{f\in\mathcal{V}_k}{\operatorname{argmin}} \|g - Af\|_2^2. \tag{3.1}$$

Como exemplos de subespaços que são usados em problemas mal-postos discretos temos:

- o subespaço $\mathcal{V}_k$ gerado pelos k primeiros vetores singulares a direita da matriz A, ou seja, $\mathcal{V}_k = \operatorname{span}\{\mathsf{v}_1, \ldots, \mathsf{v}_k\}$ ou, simplesmente, $\mathcal{V}_k = \mathcal{S}_k$ (veja equação (2.41));

- o subespaço com base ortonormal obtida da transformação discreta de cossenos [130]

$$\begin{aligned} v_1 &= \sqrt{1/n}(1, 1, \ldots, 1)^T, \\ v_i &= \sqrt{2/n}\left(\cos\left(\frac{i\pi}{2n}\right), \cos\left(\frac{3i\pi}{2n}\right), \ldots, \cos\left(\frac{(2n-1)i\pi}{2n}\right)\right)^T, \quad i = 2, \ldots, k; \end{aligned} \tag{3.2}$$

- os subespaços de Krylov. Dada uma matriz $A \in \mathbb{R}^{n\times n}$ e um vetor $g \in \mathbb{R}^n$, o subespaço de Krylov associado ao par (A, g) é $\mathcal{K}_k(A, g) = \operatorname{span}\{g, Ag, A^2g, \ldots, A^{k-1}g\}$;

- as wavelets de Daubechies [39].

Na Figura 3.1 temos os 5 primeiros vetores para cada uma destas bases (exceto a última) considerando o problema phillips com dimensão $n = 512$. Podemos perceber que, quando i aumenta, a frequência dos vetores v_i das bases SVD e DCT aumenta. Isto é comum em problemas mal-postos discretos. O mesmo não ocorre com os vetores geradores do conjunto $\mathcal{K}_5(A, g)$.

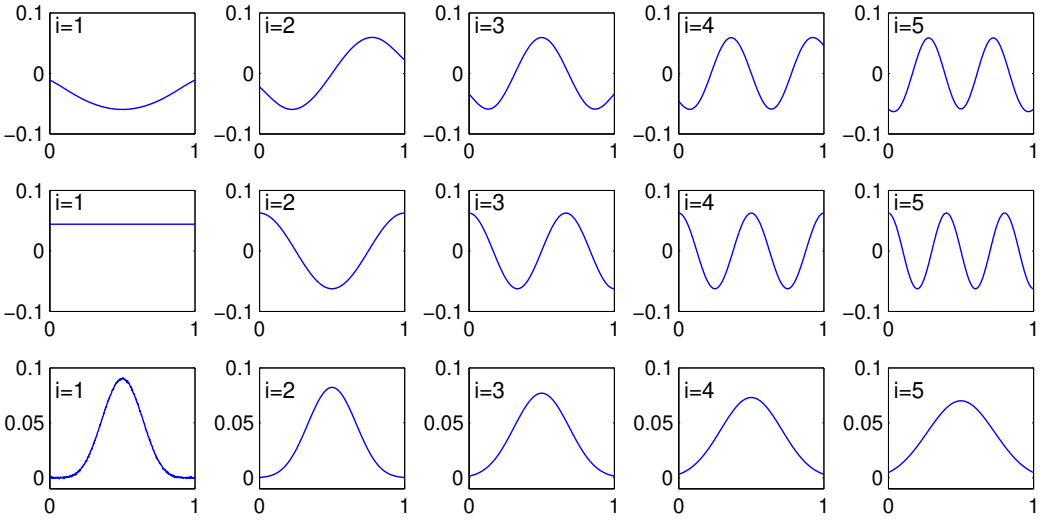


Figura 3.1: Os 5 primeiros vetores da base SVD (linha superior), da base DCT (linha intermediária) e do conjunto gerador normalizado para $\mathcal{K}_5(A, g)$ (linha inferior) para o problema phillips com $n = 512$ e NL = 0,01.

Por estarmos usando subespaços encaixantes, a norma do resíduo $r_k = g - Af_k$ associado às soluções iteradas forma uma sequência não-crescente. Por ser um resultado trivial, omitiremos a demonstração e, em seguida, veremos alguns métodos clássicos.

Lema 3.1. *Seja f_k, $k \geq 1$, dada por (3.1), então $\|r_{k+1}\|_2 \leq \|r_k\|_2$ para $k \geq 1$.*

3.1.1 O método da SVD truncada

Talvez o método baseado em subespaços mais simples seja o método da SVD Truncada (TSVD) que, essencialmente, não incorpora no somatório (1.6) as componentes associadas aos menores valores singulares da matriz A. Para distinguir a solução TSVD, usaremos a

notação f_k em vez de f_k. Assim, a k-ésima solução TSVD é dada por

$$\mathsf{f}_k = \sum_{j=1}^{k} \frac{\mathsf{u}_j^T g}{\sigma_j} \mathsf{v}_j, \quad k \leq n, \tag{3.3}$$

o que implica que $\|\mathsf{f}_{k+1}\|_2 \geq \|\mathsf{f}_k\|_2$. Do ponto de vista de subespaços, a solução f_k é dada por

$$\mathsf{f}_k = \underset{f \in \mathcal{S}_k}{\operatorname{argmin}} \|g - Af\|_2^2 = \mathsf{A}_k^{\dagger} g, \quad \mathsf{A}_k = \sum_{j=1}^{k} \sigma_j \mathsf{u}_j \mathsf{v}_j^T. \tag{3.4}$$

A escolha do parâmetro de regularização k, também chamado de parâmetro de truncamento, tem um papel importante na qualidade da solução f_k. Se k for pequeno, pouca informação do problema será incluída no somatório, em contrapartida, se k for grande, componentes do ruído associadas aos menores valores singulares serão incluídas na solução f_k, deteriorando-a. A Figura 3.2 é bastante ilustrativa. Inicialmente a solução f_4 não capturou informação suficiente do problema, posteriormente a solução f_9 tem praticamente as mesmas propriedades da solução exata e, por fim, com $k = 18$ a solução iterada já não tem mais utilidade por estar sob forte influência do ruído.

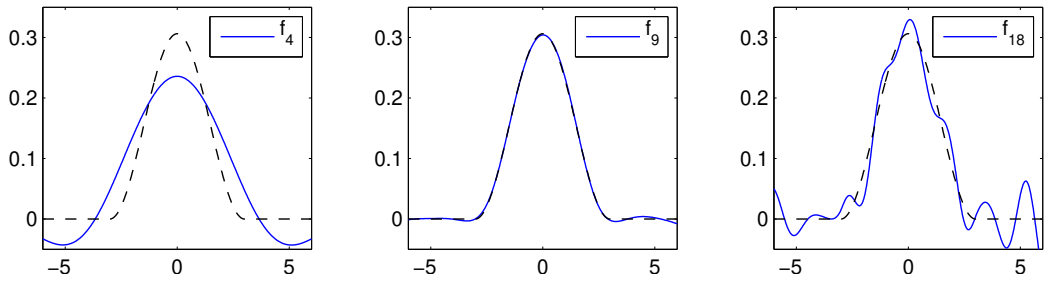


Figura 3.2: Solução exata (tracejado) e as soluções TSVD (traço contínuo) para $k = 4, 9, 18$ para o problema phillips com $n = 512$ e NL = 0,01.

Isto nos leva a conjecturar que o erro relativo $E_k = \|f_{\text{exato}} - \mathsf{f}_k\|_2 / \|f_{\text{exato}}\|_2$ deve ser grande nas primeiras iterações, atingir um possível mínimo nas iterações subsequentes e, em seguida, aumentar significativamente pois o ruído nos dados passa a ser dominante [65]. Na Figura 3.3, exibimos o comportamento do erro relativo E_k para as 25 primeiras soluções TSVD para o problema gravity (também disponível em [64]) com $n = 64$ e NL $= 10^{-5}$.

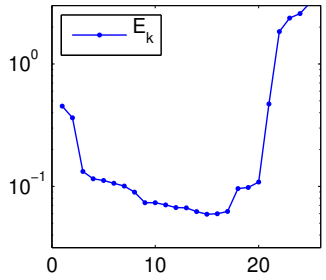


Figura 3.3: Erro relativo $E_k = \|\mathsf{f}_k - f_{\text{exato}}\|_2 / \|\|f_{\text{exato}}\|_2$ para as 25 primeiras soluções TSVD para o problema gravity com $n = 64$ e NL $= 10^{-5}$.

3.1.2 O método CGLS/LSQR

O método CGLS

O método dos gradientes conjugados (CG) é um dos métodos clássicos para solução de sistemas lineares quando a matriz de coeficientes é simétrica positiva definida. Sabemos que a matriz de coeficientes das equações normais associadas ao problema (1.4) é simétrica positiva definida. Assim, o método CGLS é o resultado de aplicar o método CG às equações normais. Para este método, as soluções f_k satisfazem

$$f_k = \underset{f \in \mathcal{K}_k(A^TA, A^Tg)}{\operatorname{argmin}} \|g - Af\|_2^2. \tag{3.5}$$

Hestenes e Stiefel [75] propuseram o que é considerada uma das implementações mais estáveis do método dos gradientes conjugados aplicado às equações normais (veja Algoritmo 3.1). Na Figura 3.4 apresentamos algumas soluções calculadas pelo algoritmo CGLS, bem como algumas soluções TSVD e a solução exata.

As primeiras soluções iteradas obtidas pelos métodos CGLS e TSVD tendem a aproximar cada vez melhor a solução exata, porém, com as iterações avançando, as mesmas vão em direção à solução f_{LS} que, como sabemos, está dominada pelo ruído. O maior desafio na prática é escolher o parâmetro k que produza uma solução que tenha capturado informação suficiente do problema e que ao mesmo tempo não esteja muito contaminada pelo ruído.

As propriedades regularizantes do algoritmo CG se devem ao fato de que, em certas aplicações, o subespaço de Krylov $\mathcal{K}_k(A^TA, A^Tg)$ pode ser considerado como uma aproximação para o subespaço $\mathcal{S}_k$ (veja Lema 2.4 e Figura 2.9). Logo, f_k pode ser considerada uma apro-

Entrada: A, g, f_0
Saída: Solução f_k

1. $r_0 = g - Af_0, \qquad d_0 = A^T r_0$

2. **Para i=1,2,...**

$\alpha_k = \|A^T r_{k-1}\|_2^2 / \|Ad_{k-1}\|_2^2$

$f_k = f_{k-1} + \alpha_k d_{k-1}$

$r_k = r_{k-1} - \alpha_k A d_{k-1}$

$\beta_k = \|A^T r_k\|_2^2 / \|A^T r_{k-1}\|_2^2$

$d_k = A^T r_k + \beta_k d_{k-1}$

Fim para

Algoritmo 3.1: Algoritmo CGLS.

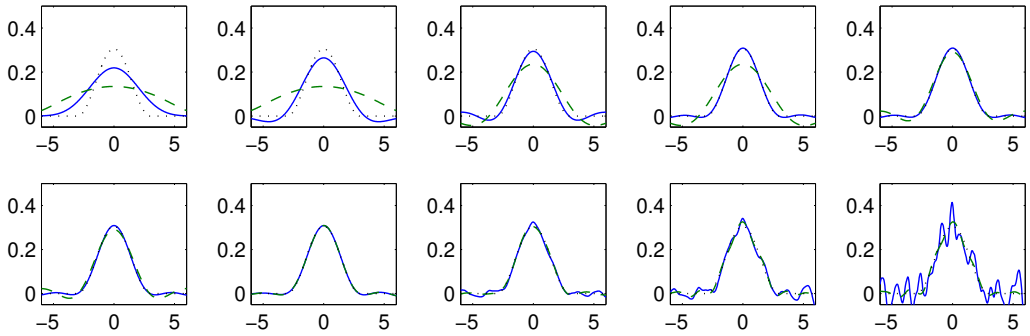


Figura 3.4: Soluções CGLS (—) e soluções TSVD (−−) para $k = 1, 2, 3, 4, 5$ (linha superior) e $k = 6, 8, 10, 12, 14$ (linha inferior) e solução exata f_{exato} $(\cdots)$ para o problema phillips com $n = 512$ e NL = 0,01.

ximação para a k-ésima solução TSVD. Para mais informações a respeito das propriedades regularizantes do método CG, sugerimos [65, 136, 137].

O método LSQR

Teoricamente, o método LSQR é idêntico ao CGLS pois as soluções iteradas satisfazem

$$f_k = \underset{f \in \mathcal{K}_k(A^TA, A^Tg)}{\operatorname{argmin}} \|g - Af\|_2^2. \tag{3.6}$$

A diferença está em como cada solução f_k é calculada na prática.

Usando as equações do processo GKB (2.16) a (2.18), segue que f_k satisfaz

$$f_k = V_k d_k, \quad d_k = \underset{d \in \mathbb{R}^k}{\operatorname{argmin}} \|\beta_1 e_1 - B_k d\|_2^2. \tag{3.7}$$

Paige e Saunders [112] mostraram que não precisamos armazenar os vetores $v_1, ..., v_k$ para encontrar f_k. Seja $B_k = Q_k^T R_k$ a decomposição QR reduzida da matriz B_k com

$$R_k = \begin{pmatrix} \rho_1 & \theta_1 & & \\ & \ddots & \ddots & \\ & & \rho_{k-1} & \theta_{k-1} \\ & & & \rho_k \end{pmatrix} \in \mathbb{R}^{k\times k}, \tag{3.8}$$

não-singular e Q_k^T com colunas ortonormais. Então,

$$d_k = R_k^{-1}\widetilde{d}_k, \quad \begin{pmatrix} \widetilde{d}_k \\ \bar{\varphi}_{k+1} \end{pmatrix} = \beta_1 Q_k e_1 \in \mathbb{R}^{k+1}, \quad \bar{\varphi}_{k+1} \in \mathbb{R}, \tag{3.9}$$

e o vetor $\widetilde{d}_k = [\varphi_1 \quad \cdots \quad \varphi_k]^T$. Assim, $f_k = V_k d_k = V_k R_k^{-1}\widetilde{d}_k \equiv Z_k \widetilde{d}_k$, e Z_k satisfaz o sistema triangular inferior $R_k^T Z_k^T = V_k^T$, logo as colunas de $Z_k = (z_1, z_2, ..., z_k)$ podem ser encontradas sucessivamente por substituição direta. Definindo $z_0 = 0$, usando a matriz R_k e identificando as últimas colunas em $Z_k R_k = V_k$, temos:

$$z_k = \frac{1}{\rho_k}(v_k - \theta_{k-1} z_{k-1}), \quad f_k = f_{k-1} + \varphi_k z_k. \tag{3.10}$$

Devido às propriedades monotônicas do método LSQR, vale $\|f_{k+1}\|_2 \geq \|f_k\|_2$ (veja [112]).

3.1.3 O método MINRES/MR-II

O método MINRES [110] foi desenvolvido para sistemas lineares com a matriz A simétrica e busca aproximar a solução $f_{\text{LS}} = A^\dagger g$ através de uma sequência de soluções f_k dadas por

$$f_k = \underset{f\in\mathcal{K}_k(A,g)}{\operatorname{argmin}} \|g - Af\|_2^2, \tag{3.11}$$

em que $\mathcal{K}_k(A, g)$ é gerado através do processo da tridiagonalização de Lanczos.

O processo da tridiagonalização de Lanczos [90, 109] (LT), é um método iterativo que aplicado à matriz A com vetor inicial g gera, após $k < n$ iterações, a matriz $V_k \in \mathbb{R}^{n\times k}$ com

colunas ortonormais e a matriz tridiagonal $T_k \in \mathbb{R}^{k\times k}$,

$$T_k = \begin{pmatrix} \alpha_1 & \beta_2 & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \beta_{k-1} & \alpha_{k-1} & \beta_k & \\ & & & \beta_k & \alpha_k & \end{pmatrix}, \tag{3.12}$$

tais que

$$\beta_1 V_k e_1 = g = \beta_1 v_1, \tag{3.13}$$

$$AV_k = V_k T_k + \beta_{k+1} v_{k+1} e_k^T, \tag{3.14}$$

em que e_k denota o k-ésimo vetor canônico de dimensão apropriada. No Algoritmo 3.2 temos o esquema do processo LT e podemos concluir que os vetores v_i gerados neste processo fornecem uma base ortonormal para o subespaço de Krylov $\mathcal{K}_k(A, g)$.

Entrada: A, g
Saída: Matrizes V_k e T_k.
1. $\beta_1 v_1 = g, \quad \alpha_1 = v_1^T A v_1, \quad v_0 = 0$
2. **Para i=1,2,...**
$\beta_{i+1} v_{i+1} = A v_i - \alpha_i v_i - \beta_i v_{i-1}$
$\alpha_{i+1} = v_{i+1}^T A v_{i+1}$
$T_{i,i} = \alpha_i, \quad T_{i+1,i} = T_{i,i+1} = \beta_{i+1}$
Fim para

Algoritmo 3.2: Processo da tridiagonalização de Lanczos.

Teoricamente, o processo da tridiagonalização de Lanczos gera vetores ortogonais, no entanto, numericamente pode haver perda desta ortogonalidade. Assim, implementações do processo LT podem ser feitas sem ou com reortogonalização.

Utilizando a matriz V_k como base para o subespaço de Krylov $\mathcal{K}_k(A, g)$, a resolução do sistema (3.11) torna-se extremamente simples, pois, se $f = V_k d$ para algum $d \in \mathbb{R}^k$, então

$\|g - Af\|_2^2 = \|\beta_1 V_k e_1 - (V_k T_k + \beta_{k+1} v_{k+1} e_k^T) d\|_2^2$. Assim, o problema (3.11) é equivalente à

$$f_k = V_k d_k, \quad d_k = \underset{d \in \mathbb{R}^k}{\operatorname{argmin}} \|\beta_1 V_k e_1 - (V_k T_k + \beta_{k+1} v_{k+1} e_k^T) d\|_2^2, \tag{3.15}$$

e o vetor d_k é obtido por $(T_k^2 + \beta_{k+1}^2 e_k e_k^T) d_k = \beta_1 T_k e_1$, que é um sistema linear cuja matriz é pentadiagonal, simétrica e, pelo menos, positiva semidefinida.

Através de rotações de Givens, podemos encontrar uma matriz ortogonal Q_k tal que $T_k Q_k^T = \overline{L}_k$ seja triangular inferior. Assim, $T_k^2 = T_k Q_k^T Q_k T_k = \overline{L}_k \overline{L}_k^T$ e disto temos que

$$T_k^2 + \beta_{k+1}^2 e_k e_k^T = \overline{L}_k \overline{L}_k^T + \beta_{k+1}^2 e_k e_k^T = L_k L_k^T, \tag{3.16}$$

em que a matriz L_k é triangular inferior e não-singular. Portanto, L_k é o fator de Cholesky da matriz $T_k^2 + \beta_{k+1}^2 e_k e_k^T$. Como consequência, a solução d_k pode ser encontrada através de $L_k L_k^T d_k = \beta_1 \overline{L}_k Q_k e_1$. Notemos que a matriz $\overline{L}_k$ pode ser escrita como $\overline{L}_k = L_k D_k$, em que D_k é uma matriz diagonal, logo

$$L_k L_k^T d_k = \beta_1 L_k D_k Q_k e_1 \Rightarrow L_k^T d_k = \beta_1 D_k Q_k e_1 \equiv t_k. \tag{3.17}$$

Finalmente, temos que a solução f_k é

$$f_k = V_k d_k = V_k L_k^{-T} L_k^T d_k = \check{M}_k t_k, \quad \check{M}_k = [\check{m}_1 \quad \cdots \quad \check{m}_k] = V_k L_k^{-T}. \tag{3.18}$$

A última coluna da matriz $\check{M}_k$ pode ser obtida pelas últimas três colunas da matriz V_k, pois $\check{M}_k L_k^T = V_k$, assim, apenas os últimos três vetores de $\check{M}_k$ precisam ser armazenados no processo iterativo. Logo, a solução f_k pode ser atualizada de uma iteração para outra.

Hanke [58] propôs uma modificação do método MINRES chamada de método MR-II, cuja diferença é que o subespaço de Krylov em (3.11) é $\mathcal{K}_k(A, Ag)$, ou seja, o vetor inicial no processo LT é o vetor Ag.

3.1.4 O método GMRES/RRGMRES

Com o propósito de generalizar as ideias do método MINRES para matrizes não-simétricas, foi desenvolvido o método GMRES [125] que utiliza o método de Arnoldi [2] para gerar uma base para o subespaço de Krylov $\mathcal{K}_k(A, g - Af_0)$, em que f_0 é alguma aproximação inicial.

O método de Arnoldi [2] é um método iterativo para matrizes quadradas que aplicado à matriz A com vetor inicial g gera, após $k < n$ iterações, a matriz $Q_k \in \mathbb{R}^{n\times k}$ com colunas ortogonais e a matriz Hessenberg superior $H_k \in \mathbb{R}^{k\times k}$,

$$H_k = \begin{pmatrix} h_{11} & h_{12} & \cdots & \cdots & h_{1k} \\ h_{21} & h_{22} & \cdots & \cdots & h_{2k} \\ & h_{32} & \cdots & \cdots & h_{3k} \\ & & \ddots & \cdots & \vdots \\ & & & h_{k,k-1} & h_{kk} \end{pmatrix}, \tag{3.19}$$

tais que

$$\|g\|_2 Q_k e_1 = g = \|g\|_2 q_1, \tag{3.20}$$

$$AQ_k = Q_k H_k + h_{k+1,k} q_{k+1} e_k^T, \tag{3.21}$$

em que e_k denota o k-ésimo vetor canônico de dimensão apropriada. No Algoritmo 3.3 temos o esquema da redução à forma Hessenberg de Arnoldi e podemos concluir que os vetores q_i gerados neste processo fornecem uma base ortonormal para o subespaço de Krylov $\mathcal{K}_k(A, g)$.

Teoricamente, o processo de Arnoldi gera vetores ortogonais, no entanto, numericamente pode haver perda desta ortogonalidade. Assim, implementações do processo de Arnoldi podem ser feitas sem ou com reortogonalização.

Notemos que a equação (3.21) pode ser escrita como $AQ_k = Q_{k+1}\widetilde{H}_k$ em que

$$\widetilde{H}_k = \begin{pmatrix} & H_k & \\ \mathbf{0} & & h_{k+1,k} \end{pmatrix}, \tag{3.22}$$

com $\mathbf{0} = [0 \quad \cdots \quad 0] \in \mathbb{R}^{1\times(k-1)}$. Assim, segue que se $f = Q_k d$ para $d \in \mathbb{R}^k$, temos

Entrada: A, g
Saída: Matrizes Q_k e H_k.

1. $q_1 = g/\|g\|_2$

1. **Para i=1,2,...**

Para j=1,...,i

$h_{ji} = q_j^T A q_i$

$H_{ji} = h_{ji}$

Fim para (j)

$$h_{i+1,i} q_{i+1} = A q_i - \sum_{j=1}^{i} h_{ji} q_j$$

$H_{i+1,i} = h_{i+1,i}$

Fim para (i)

Algoritmo 3.3: Processo de Arnoldi.

$\|g - Af\|_2 = \|\|g\|_2 e_1 - \widetilde{H}_k d\|_2$. Portanto, minimizar $\|g - Af\|_2^2$ restrito ao subespaço de Krylov $\mathcal{K}_k(A, g)$, é equivalente a resolver o problema irrestrito

$$f_k = Q_k d_k, \quad d_k = \underset{d \in \mathbb{R}^k}{\operatorname{argmin}} \left\| \|g\|_2 e_1 - \widetilde{H}_k d \right\|_2^2, \tag{3.23}$$

que pode ser resolvido através da decomposição QR da matriz $\widetilde{H}_k$ via rotações de Givens.

Uma modificação do método GMRES, chamada de RRGMRES [25], busca aproximar f_{LS} com soluções $f_k \in \mathcal{K}_k(A, A(g - Af_0))$, ou seja, o vetor inicial no processo de Arnoldi é $A(g - Af_0)$.

3.1.5 Parâmetro $\gamma_k = \|A - AP_k\|_2$ para os métodos MINRES e GMRES

O Lema 2.2 nos fornece uma expressão geral para o parâmetro $\gamma_k = \|A - AP_k\|_2$. Porém, para o método MINRES esta expressão se torna mais simples. Se a família de subespaços $\mathcal{V}_k$ são os subespaços de Krylov gerados pelo processo LT, então γ_k é dado pelo próximo teorema.

Teorema 3.2. *Assuma que o processo LT possa ser executado até o fim. Seja* $G_{T,k} \in$

$\mathbb{R}^{(n-k+1)\times(n-k)}$ *a matriz com largura de banda inferior 2*

$$G_{T,k} = \begin{pmatrix} \beta_{k+1} & & & & \\ \alpha_{k+1} & \beta_{k+2} & & & \\ \beta_{k+2} & \alpha_{k+2} & \beta_{k+3} & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{n-1} & \alpha_{n-1} & \beta_n \\ & & & \beta_n & \alpha_n \end{pmatrix}, \tag{3.24}$$

em que α_k *e* β_k *são gerados pelo processo de tridiagonalização de Lanczos aplicado à matriz* A*. Então para* $k = 1, \ldots, n-1$ *temos*

1. $\gamma_k = \|G_{T,k}\|_2$,

2. $\gamma_{k+1} \leq \gamma_k$.

Demonstração. O primeiro item segue como consequência do Lema 2.2 e o segundo pelo fato de $G_{T,k+1}$ ser uma submatriz de $G_{T,k}$.

□

Como não temos a matriz $G_{T,k}$ no passo k, podemos aproximar γ_k com a norma da primeira coluna de $G_{T,k}$, ou seja, $\gamma_k \gtrapprox \|G_{T,k}e_1\|_2 = \sqrt{\alpha_{k+1}^2 + \beta_{k+1}^2 + \beta_{k+2}^2}$. Porém, β_{k+2} também não está disponível no passo k, assim, podemos analisar apenas a quantidade $\sqrt{\alpha_{k+1}^2 + \beta_{k+1}^2}$. Sabemos que se posto$(AP_k) = k$, então $\gamma_k \geq \sigma_{k+1}$, portanto, na prática, ambos γ_k e $\sqrt{\alpha_{k+1}^2 + \beta_{k+1}^2}$ aproximam σ_{k+1} e esta aproximação melhora quando k aumenta. Podemos ver uma ilustração disto na Figura 3.5 e concluir, pelo menos neste exemplo, que o subespaço de Krylov gerado pelo processo GKB aproxima melhor o subespaço $\mathcal{S}_k$ do que o subespaço de Krylov gerado pelo processo LT.

Se a família de subespaços $\mathcal{V}_k$ são os subespaços de Krylov gerados pelo processo de Arnoldi, então a quantidade $\gamma_k = \|A - AP_k\|_2$ é dada pelo seguinte teorema.

Teorema 3.3. *Assuma que o processo de Arnoldi possa ser executado até o fim. Seja* $G_{H,k}$

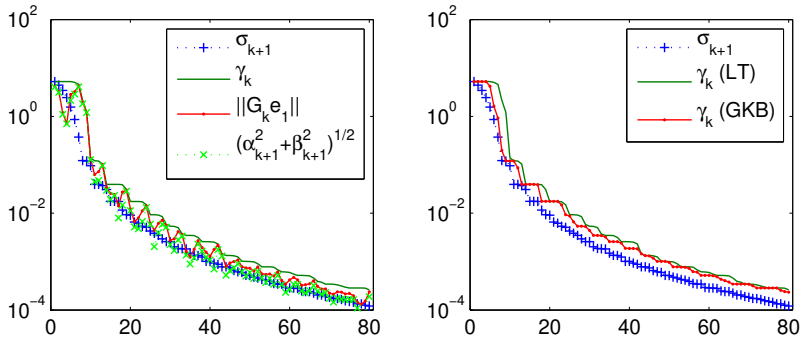


Figura 3.5: Estimativa $\|G_{T,k}e_1\|_2$ e $\sqrt{\alpha_{k+1}^2+\beta_{k+1}^2}$ (esquerda). Parâmetros γ_k obtidos pelos o processos GKB e LT para o problema phillips com $n = 512$ e NL = 0,01.

a matriz $n \times (n-k)$ *definida por*

$$G_{H,k} = \begin{pmatrix} h_{1,k+1} & h_{1,k+2} & \cdots & h_{1,n} \\ h_{2,k+1} & h_{2,k+2} & \cdots & h_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ h_{k+2,k+1} & h_{k+2,k+1} & \cdots & h_{k+2,n} \\ \ddots & \ddots & \ddots & \vdots \\ & h_{n-1,n-2} & h_{n-1,n-1} & h_{n-1,n} \\ & & h_{n,n-1} & h_{n,n} \end{pmatrix}, \tag{3.25}$$

em que h_{ij} *são gerados pelo processo de Arnoldi aplicado à matriz* A. *Então, para* $k = 1, \ldots, n-1$, *temos*

1. $\gamma_k = \|G_{H,k}\|_2$,

2. $\gamma_{k+1} \leq \gamma_k$.

Demonstração. Análoga ao caso anterior.

□

Notemos que, neste caso, para aproximarmos γ_k a partir de $\|G_{H,k}e_1\|_2$ é mais complicado pois no passo k não temos $h_{j,k+1}$ para $j = 1, \ldots, k+2$. Os coeficientes $h_{j,k+1}$ para $j = 1, \ldots, k+1$ até podem ser calculados pois $h_{j,k+1} = q_j^T A q_{k+1}$ e dispomos de q_{k+1}, porém, o custo pode se tornar grande caso o número de iterações também seja grande. Na Figura

3.6 podemos apreciar o parâmetro γ_k calculado pelo processo de Arnoldi e o calculado pelo processo GKB. O processo GKB novamente fornece uma boa aproximação para σ_{k+1}, porém o processo de Arnoldi não foi capaz de capturar as mesmas informações, pelo menos para o problema heat. Mesmo que o método de Arnoldi seja realizado com reortogonalização, não há alteração significativa no resultado.

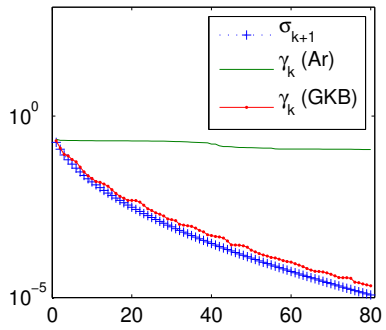


Figura 3.6: Parâmetro γ_k obtido usando o processo GKB e γ_k obtido usando as iterações de Arnoldi (Ar) para o problema heat com $n = 512$ e NL = 0,01.

A principal conclusão que podemos tirar deste simples exemplo é que as soluções iteradas obtidas pelo método GMRES podem não ter utilidade prática, isto é, o método GMRES pode não ser uma boa opção para tratar problemas mal-postos discretos (veja [79]).

3.2 Critério de parada automático para métodos iterativos

As propriedades de regularização dos métodos iterativos são bem compreendidas, porém, quando a tarefa é determinar o número apropriado de iterações sem um conhecimento a priori da norma do ruído, como exigido pelo princípio da discrepância, a situação é bem diferente. Nesta seção vamos desenvolver um critério de parada automático para os métodos iterativos que não requer nenhuma estimativa da norma do ruído e, para isto, vamos analisar o método da TSVD. Além disso, vamos assumir que a condição discreta de Picard é satisfeita e que o vetor de ruídos tem entradas aleatórias, com distribuição normal, média zero e variância

especificada. Uma estimativa para o erro na solução TSVD é

$$\|f_{\text{exato}} - \mathsf{f}_k\|_2 \leq \|f_{\text{exato}} - \mathsf{A}_k^{\dagger} g_{\text{exato}}\|_2 + \|\mathsf{A}_k^{\dagger}\epsilon\|_2 \equiv E_1(k) + E_2(k), \tag{3.26}$$

em que

$$E_1(k) = \sqrt{\sum_{j=k+1}^{n} \frac{|\mathsf{u}_j^T g_{\text{exato}}|^2}{\sigma_j^2}}, \quad E_2(k) = \sqrt{\sum_{j=1}^{k} \frac{|\mathsf{u}_j^T \epsilon|^2}{\sigma_j^2}}. \tag{3.27}$$

O primeiro termo denota o erro devido à regularização, este diminui com k e pode se tornar pequeno para k grande. O segundo termo, muitas vezes referido como erro de perturbação, mede a intensidade do erro e aumenta com k, este pode ser extremamente grande quando valores singulares pequenos entram no somatório. A escolha do parâmetro de regularização requer, então, que haja um balanço entre os dois termos de modo a minimizar a estimativa do erro na solução. Observamos agora que, por estarmos lidando com problemas mal-postos discretos e a CDP é satisfeita, existe um k^* tal que vale (1.7).

Analisando mais atentamente $E_1(k)$ e $E_2(k)$, podemos observar que, quando $k > k^*$, a norma do erro $E_2(k)$ começa a aumentar drasticamente enquanto que $E_1(k)$ permanece sob controle devido a CDP e que o erro $E_1(k) + E_2(k)$ não deve ser minimizado para $k > k^*$. Por outro lado, quando $k < k^*$, coeficientes $|\mathsf{u}_j^T g_{\text{exato}}|$ importantes são incluídos em $E_1(k)$, no qual começa a aumentar, enquanto que $E_2(k)$ permanece sob controle pois os valores singulares σ_j dominam os coeficientes $|\mathsf{u}_j^T \epsilon|$, e mais uma vez, o erro não é minimizado. Esta análise sugere que a estimativa do erro deve ser minimizada em $k = k^*$. Na prática não temos o vetor ϵ, então o que podemos fazer é minimizar alguma função que tenha comportamento semelhante.

Da SVD de A segue que

$$\|\mathsf{f}_k\|_2^2 = \sum_{j=1}^{k} \frac{|\mathsf{u}_j^T g|^2}{\sigma_j^2}, \quad \|\mathsf{r}_k\|_2^2 = \|g - A\mathsf{f}_k\|_2^2 = \sum_{j=k+1}^{n} |\mathsf{u}_j^T g|^2 + \delta_0^2, \tag{3.28}$$

em que δ_0 denota a norma da componente de g que não pertence ao espaço coluna de A.

Para $k \geq 1$, seja $\Psi_k = \|\mathsf{f}_k\|_2 \|\mathsf{r}_k\|_2$. Como $\|\mathsf{f}_k\|_2$ é crescente e $\|\mathsf{r}_k\|_2$ é decrescente, minimizar $\log(\Psi_k)$ é equivalente a minimizar a soma de dois termos, um crescente e outro decrescente.

Então, em princípio, deve existir um inteiro k no qual Ψ_k é minimizada. Da condição (1.7) e para $k > k^*$, a norma do resíduo decresce aproximadamente de modo linear, enquanto que a norma da solução começa a crescer drasticamente pois, neste caso, $0 \approx \sigma_k < |\mathsf{u}_k^T \epsilon|$. Então, Ψ_k não pode ser minimizada para k nesta faixa de valores. Por outro lado, quando $k < k^*$, a norma do resíduo fica extremamente grande pois os coeficientes $|\mathsf{u}_j^T g|$ são incluídos, enquanto que a norma da solução cresce lentamente com k. Logo, a função Ψ_k não pode ser minimizada para $k < k^*$ e, portanto, esta deve ser minimizada em $k = k^*$. Concluímos, então, que um bom parâmetro de regularização para a TSVD é o minimizador de Ψ_k (veja a Figura 3.7).

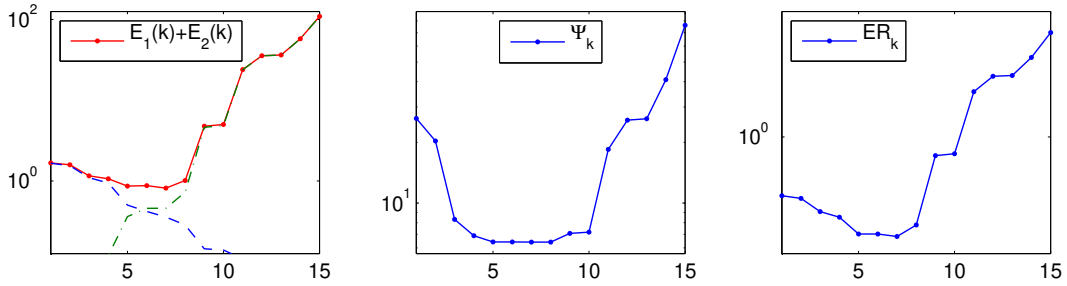


Figura 3.7: Cotas (3.27), função Ψ_k e erros relativos em f_k para o problema gravity com $n = 32$ e NL = 0,02. Erro relativo mínimo atingido em $k^* = 7 = \operatorname{argmin} \Psi_k$ e $\|\mathsf{f}_7 - f_{\text{exato}}\| = 0{,}0778\|f_{\text{exato}}\|_2$.

No entanto, a SVD pode ser inviável para problemas de grande porte e o método da TSVD pode não ter utilidade prática. Como alternativa à TSVD para problemas de grande porte, a ideia é usar o método iterativo LSQR em conjunto com o critério de parada definido similarmente ao critério de truncamento para a TSVD acima. Portanto, nosso critério de parada automático para o método LSQR seleciona o parâmetro $\widehat{k}$ tal que

$$\widehat{k} = \operatorname{argmin} \Psi_k, \quad \Psi_k = \|f_k\|_2 \|g - Af_k\|_2. \tag{3.29}$$

A motivação para usarmos este critério de parada vem da observação de que para k pequeno temos $\|f_k\|_2$ pequeno e $\|g - Af_k\|_2$ grande e, portanto, Ψ_k não é minimizada. Por outro lado, para k grande temos $\|f_k\|_2$ grande e $\|g - Af_k\|_2$ pequeno e, mais uma vez, Ψ_k não é minimizada. Isto sugere que o minimizador de Ψ_k corresponde a um bom balanço entre o tamanho de ambas as normas. Naturalmente que este critério não está restrito somente ao método LSQR, isto é, o mesmo pode ser usado em conexão com qualquer método iterativo.

Na Figura 3.8 apresentamos três sequências Ψ_k para o problema phillips com dimensão $n = 512$ e NL = 0,01. Cada sequência foi calculada, respectivamente, pelos métodos TSVD, LSQR e MR-II.

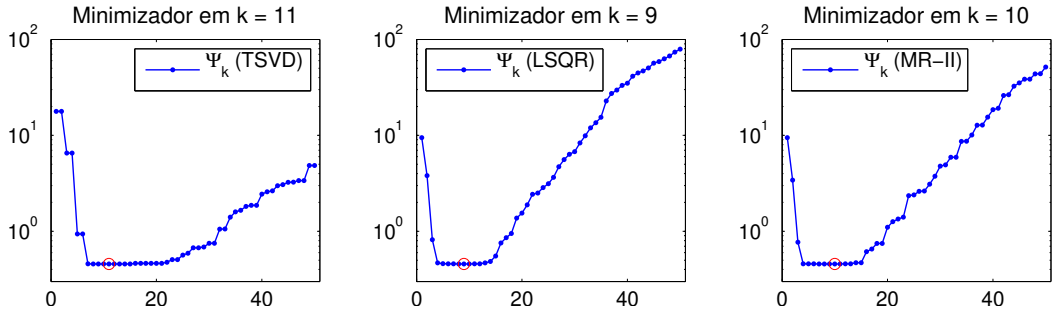


Figura 3.8: Sequências Ψ_k obtidas pelos métodos TSVD, LSQR and MR-II aplicados ao problema phillips com $n = 512$ e NL $= 0,01$, $k = 1, \ldots, 50$.

Do ponto de vista prático, a implementação deste critério pode ser feita monitorando a seguinte sequência de diferenças finitas avançadas

$$\nabla\Psi_k = \Psi_{k+1} - \Psi_k, \quad k \geq 1, \tag{3.30}$$

e então, selecionar o primeiro índice k tal que $\nabla\Psi_{k-1} \leq 0$ e $\nabla\Psi_k \geq 0$. Como Ψ_k é uma sequência de números não-negativos, o minimizador de Ψ_k^2 coincide com o de Ψ_k. Então,

$$\nabla\Psi_k^2 = \Psi_{k+1}^2 - \Psi_k^2 = \|f_{k+1}\|_2^2 \nabla\|f_k\|_2^2 \left(\phi_{k+1}^2 - \xi_k\right), \tag{3.31}$$

em que

$$\phi_k^2 = \frac{\|r_k\|_2^2}{\|f_k\|_2^2}, \quad \xi_k = -\frac{\|f_k\|_2^2}{\|f_{k+1}\|_2^2} \frac{\nabla\|r_k\|_2^2}{\nabla\|f_k\|_2^2}, \tag{3.32}$$

e $\nabla\|r_k\|_2^2$ e $\nabla\|f_k\|_2^2$ são, respectivamente, as diferenças finitas avançadas para $\|r_k\|_2^2$ e $\|f_k\|_2^2$. De modo a analisarmos o sinal de $\nabla\Psi_k^2$, três casos devem ser considerados com relação ao sinal de $\nabla\|f_k\|_2^2$:

- se $\nabla\|f_k\|_2^2 = 0$, o que é improvável que ocorra em problemas práticos, ξ_k não está definido e $\nabla\Psi_k^2 = \|f_k\|_2^2 \nabla\|r_k\|_2^2 \leq 0$. Portanto, $\Psi_{k+1} \leq \Psi_k$ e a sequência Ψ_k não é minimizada em k;

- se $\nabla\|f_k\|_2^2 < 0$, como pode ocorrer com, por exemplo, o método GMRES, segue que $\xi_k \leq 0$, pois $\nabla\|r_k\|_2^2 \leq 0$. Então, $\phi_{k+1}^2 - \xi_k \geq 0$ e se $\nabla\Psi_k^2 \neq 0$, segue que $\nabla\Psi_k^2 \leq 0$. Portanto, $\Psi_{k+1} \leq \Psi_k$ e a sequência Ψ_k não é minimizada em k;

- se $\nabla\|f_k\|_2^2 > 0$, segue que $\|f_{k+1}\|_2^2\nabla\|f_k\|_2^2 \geq 0$. Esta hipótese também implica que $\xi_k \geq 0$, e então, o sinal de $\nabla\Psi_k^2$ depende do sinal de $\phi_{k+1}^2 - \xi_k$. Portanto, $\nabla\Psi_k^2 \leq 0$ se $\xi_k/\phi_{k+1}^2 \geq 1$ e $\nabla\Psi_k \geq 0$ se $\xi_k/\phi_{k+1}^2 \leq 1$. Neste caso, k será o minimizador de Ψ_k se $\nabla\Psi_{k-1} \leq 0$ e $\nabla\Psi_k \geq 0$.

A Figura 3.9 mostra ambas as sequências ϕ_{k+1}^2 e ξ_k em que as quantidades $\|r_k\|_2^2$ and $\|f_k\|_2^2$ foram calculas pelos métodos TSVD, LSQR e MR-II usando os mesmos dados da figura anterior. Como esperado, a mudança de sinal ocorre nas iterações $k = 12$ para TSVD, $k = 10$ para LSQR e $k = 11$ para MR-II uma vez que os minimizadores de Ψ_k são, respectivamente, $k = 11$, $k = 9$ e $k = 10$.

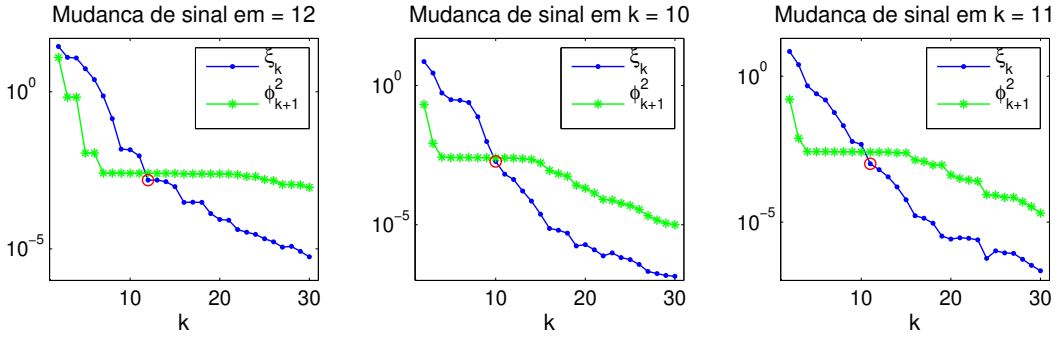


Figura 3.9: Sequências ϕ_{k+1}^2 e ξ_k calculadas pelos métodos TSVD (esquerda), LSQR (centro) e MR-II (direita) usando os mesmos dados da figura anterior. A troca de sinal em $\nabla\Psi_k$ ocorre uma iteração após o minimizador de Ψ_k ser atingido.

Vamos considerar que a sequência f_k é calculada através do método TSVD, ou seja, temos f_k e $\mathsf{r}_k = g - A\mathsf{f}_k$. Da equação (3.28) segue que

$$\frac{\nabla\|\mathsf{r}_k\|_2^2}{\nabla\|\mathsf{f}_k\|_2^2} = -\sigma_{k+1}^2. \tag{3.33}$$

E, portanto

$$\text{sign}\left(\nabla\Psi_k^2\right) = \begin{cases} 1, \text{ se } \dfrac{\|\mathsf{r}_{k+1}\|_2}{\|\mathsf{f}_k\|_2} > \sigma_{k+1}, \\ -1, \text{ if } \dfrac{\|\mathsf{r}_{k+1}\|_2}{\|\mathsf{f}_k\|_2} < \sigma_{k+1}. \end{cases} \tag{3.34}$$

Um ponto chave deste critério de parada automático é que não há necessidade de se executar um número predeterminado de passos como exigido, por exemplo, pelo critério da Curva-L, como veremos a seguir. Para deixarmos esta ideia mais clara, vamos supor que $\widehat{k}$ é selecionado por (3.29). Então, apenas $\widehat{k}+1$ passos são necessários. Algumas observações com relação à esta análise:

1. se a sequência $\|f_k\|_2$ é não-decrescente, como ocorre com os métodos TSVD e LSQR, sempre temos $\nabla\|f_k\|_2^2 \geq 0$ e a sequência ϕ_k é não-crescente;

2. a sequência ϕ_k ainda pode ser não-crescente mesmo se $\|f_k\|_2 > \|f_{k+1}\|_2$ para alguns valores de k, pois ϕ_k é um quociente entre $\|r_k\|_2$ e $\|f_k\|_2$.

Vamos finalizar esta seção com alguns métodos existentes para a escolha do parâmetro de regularização k.

3.2.1 Princípio da Discrepância

O uso do princípio da discrepância como critério de parada para métodos iterativos [54], sugere escolher o parâmetro $\widehat{k}$ como o primeiro k tal que

$$\|g - Af_k\|_2 \leq \delta. \tag{3.35}$$

Na Figura 3.10 temos a norma do resíduo associado às primeiras vinte aproximações f_k obtidas pelo método LSQR aplicado ao problema phillips com $n = 512$ e NL = 0,001, a reta constante $z = \delta$ (gráfico a esquerda) e o erro relativo para cada aproximação f_k (gráfico a direita). Podemos perceber que, para parâmetros $k \geq 9$, todas as aproximações satisfazem $\|g - Af_k\|_2 \leq \delta$, logo devemos escolher f_9 como solução regularizada.

3.2.2 Curva-L discreta

O critério de parada proposto anteriormente, equação (3.29), nada mais é do que uma versão discreta do método proposto por Regińska (2.10), que tem por objetivo localizar o "canto" da curva-L para o método de regularização de Tikhonov. Portanto, é razoável que o

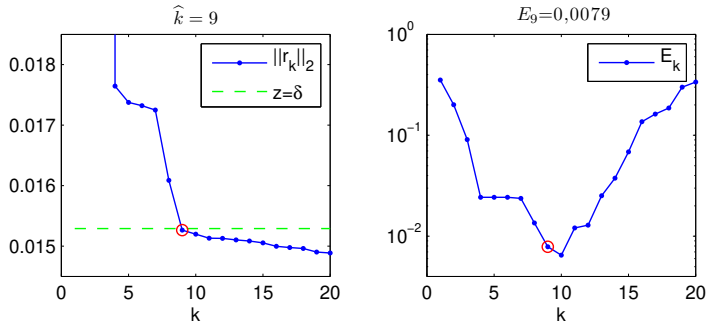


Figura 3.10: Norma do resíduo associado às primeiras vinte aproximações f_k (LSQR) e a reta $z = \delta$ (esquerda) e erro relativo nas aproximações (direita) para o problema phillips com $n = 512$ e NL = 0,001. O círculo denota os valores para $k = 9$.

minimizador de (3.29) esteja relacionado com um ponto na curva-L discreta

$$\mathcal{L}_{\mathsf{q}} = \left\{ (\log \|g - Af_k\|_2, \log \|f_k\|_2) \in \mathbb{R}^2 \quad / \quad k = 1, \ldots, \mathsf{q} \right\}. \tag{3.36}$$

Técnicas para encontrar o "canto" incluem o uso de *splines* por Hansen e O'Leary [73], o método do triângulo por Castellanos et al. [30], um método por Rodriguez e Theis [123] e um algoritmo adaptativo denominado por *pruning* por Hansen et al. [71].

Este *pruning* foi desenvolvido com o objetivo de contornar as dificuldades dos métodos existentes e uma avaliação de sua efetividade pode ser encontrada em [71]. No entanto, encontrar o "canto" com uma quantidade finita de pontos não é uma tarefa fácil e os algoritmos existentes têm suas peculiaridades (veja, por exemplo, Hansen [65, p. 190], Hansen et al. [71] para o caso de múltiplos "cantos" e [127] para uma aplicação da curva-L no contexto de microcirculação pulmonar). Veremos na seção 3.4 que este algoritmo *pruning*, apesar de funcionar em alguns problemas, é altamente dependente do número de pontos q.

3.3 Solução iterada f_k vs. solução TSVD $\mathbf{f}_k$

Nosso objetivo principal nesta seção é entender melhor o comportamento da solução TSVD f_k dada por (3.3) em relação à solução f_k obtida por (3.1) . Para tanto, começamos com uma cota superior para o ângulo entre os subespaços $\mathcal{V}_k$ e $\mathcal{S}_k$, ou seja, o subespaço gerado pelos k primeiros vetores singulares a direita da matriz A.

Teorema 3.4. *Seja $V_k = [v_1, \ldots, v_k] \in \mathbb{R}^{n\times k}$ uma matriz com colunas ortonormais tal que span$\{v_1, \ldots, v_k\} = \mathcal{V}_k$ e seja Ω_k o ângulo entre os subespaços $\mathcal{V}_k$ e $\mathcal{S}_k$ dado por (2.41). Então*

$$\text{sen}(\Omega_k) \leq \frac{\gamma_k}{\sigma_k}, \tag{3.37}$$

com γ_k dado por (2.28).

Demonstração. Seja a matriz $V = [V_k \;\; V_\perp] \in \mathbb{R}^{n\times n}$ ortogonal. Da demonstração do Lema 2.2 segue que $AV_\perp = UM_2$. Multiplicando, a esquerda, esta última equação por U_k^T dada por (2.41), e usando o fato de que $\mathsf{U}_k^T A = \Sigma_k \mathsf{V}_k^T$ em que a matriz Σ_k contém os k primeiros valores singulares na sua diagonal, temos $\mathsf{V}_k^T V_\perp = \Sigma_k^{-1}\mathsf{U}_k U M_2$. Tomando norma em ambos os lados e usando o Lema 2.2 temos que $\text{sen}(\Omega_k) \leq \gamma_k/\sigma_k$, em que usamos o fato de que $\text{sen}(\Omega_k) = \|\mathsf{V}_k^T V_\perp\|_2$ (veja [51]).

□

No caso particular em que o subespaço $\mathcal{V}_k$ é gerado pelo processo GKB temos uma outra limitante superior para $\text{sen}(\Omega_k)$.

Teorema 3.5. *Sejam Ω_k o ângulo entre os subespaços $\mathcal{V}_k$ e $\mathcal{S}_k$ gerados, respectivamente, pelo processo GKB e (2.41) e a matriz B_k dada por (2.15). Se $\sigma_{\min}(B_k) > \sigma_{k+1}$, então*

$$\text{sen}(\Omega_k) \leq \frac{\sigma_{\min}(B_k)\alpha_{k+1}}{\sigma_{\min}(B_k)^2 - \sigma_{k+1}^2}. \tag{3.38}$$

Demonstração. Vamos reescrever a SVD da matriz A como

$$A = [\mathsf{U}_k \;\; \mathsf{U}_0 \;\; \mathsf{U}_\perp] \begin{bmatrix} \Sigma_k & 0 \\ 0 & \Sigma_0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathsf{V}_k^T \\ \mathsf{V}_0^T \end{bmatrix} = \mathsf{U}_k\Sigma_k\mathsf{V}_k^T + \mathsf{U}_0\Sigma_0\mathsf{V}_0^T. \tag{3.39}$$

Como A tem posto completo, o processo GKB pode ser executado n passos sem interrupção,

logo podemos escrever A como

$$A = [U_{k+1} \;\; U_0 \;\; U_\perp] \begin{bmatrix} B_k & C_k \\ 0 & F_k \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_k^T \\ V_0^T \end{bmatrix}, \tag{3.40}$$

em que $B_k \in \mathbb{R}^{(k+1)\times k}$ é dada pela equação (2.15), $C_k = \alpha_{k+1} e_{k+1} e_1^T \in \mathbb{R}^{(k+1)\times(n-k)}$,

$$F_k = \begin{bmatrix} \beta_{k+2} & \alpha_{k+2} & & \\ & \ddots & \ddots & \\ & & \beta_n & \alpha_n \\ & & & \beta_{n+1} \end{bmatrix} \in \mathbb{R}^{(n-k)\times(n-k)}. \tag{3.41}$$

Seja $B_k = P_k \begin{pmatrix} D_k \\ 0 \end{pmatrix} Q_k^T$ a SVD de B_k, então podemos reescrever (3.40) como

$$A = [\widetilde{U}_k \;\; \widetilde{U}_0 \;\; \widetilde{U}_\perp] \begin{bmatrix} D_k & \widetilde{C}_k \\ 0 & \widetilde{F}_k \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \widetilde{V}_k^T \\ \widetilde{V}_0^T \end{bmatrix} = \widetilde{U}_k D_k \widetilde{V}_k^T + \widetilde{U}_k \widetilde{C}_k \widetilde{V}_0^T + \widetilde{U}_0 \widetilde{F}_k \widetilde{V}_0^T, \tag{3.42}$$

em que $\widetilde{U}_k \in \mathbb{R}^{m\times k}$ consiste das k primeiras colunas da matriz $U_{k+1}P_k$, $\widetilde{U}_0 = [\widetilde{p} \;\; U_0] \in \mathbb{R}^{m\times(n-k+1)}$ com $\widetilde{p}$ sendo a última coluna de $U_{k+1}P_k$, $\widetilde{U}_\perp = U_\perp P_k \in \mathbb{R}^{m\times(m-n-1)}$, $\widetilde{C}_k \in \mathbb{R}^{k\times(n-k)}$ consiste das k primeiras linhas da matriz $P_k^T C_k$, $\widetilde{F}_k = [\widetilde{f}^T \;\; F_k^T]^T \in \mathbb{R}^{(n-k+1)\times(n-k)}$ com $\widetilde{f}$ sendo a última linha de $P_k^T C_k$ e finalmente $\widetilde{V}_k = V_k Q_k$ e $\widetilde{V}_0 = V_0$. Como consequência, temos

$$\begin{aligned} \widetilde{V}_k^T \mathsf{V}_0 &= D_k^{-1} \left(\widetilde{U}_k^T A - \widetilde{C}_k \widetilde{V}_0^T \right) \mathsf{V}_0 = D_k^{-1} \left(\widetilde{U}_k^T A \mathsf{V}_0 - \widetilde{C}_k \widetilde{V}_0^T \mathsf{V}_0 \right) \\ &= D_k^{-1} \left(\widetilde{U}_k^T \mathsf{U}_0 \Sigma_0 - \widetilde{C}_k \widetilde{V}_0^T \mathsf{V}_0 \right). \end{aligned} \tag{3.43}$$

Além disso, temos

$$A\widetilde{V}_k = \widetilde{U}_k D_k \Rightarrow \widetilde{V}_k^T A^T = D_k^T \widetilde{U}_k^T \Rightarrow \widetilde{U}_k^T = D_k^{-1} \widetilde{V}_k^T A^T = D_k^{-1} \widetilde{V}_k^T A^T, \tag{3.44}$$

e, também, que $\widetilde{U}_k^T \mathsf{U}_0 = D_k^{-1} \widetilde{V}_k^T A^T \mathsf{U}_0 = D_k^{-1} \widetilde{V}_k^T \mathsf{V}_0 \Sigma_0$. Então,

$$\begin{aligned} \widetilde{V}_k^T \mathsf{V}_0 &= D_k^{-1} \left(D_k^{-1} \widetilde{V}_k^T \mathsf{V}_0 \Sigma_0^2 - \widetilde{C}_k \widetilde{V}_0^T \mathsf{V}_0 \right) \\ &= D_k^{-2} \widetilde{V}_k^T \mathsf{V}_0 \Sigma_0^2 - D_k^{-1} \widetilde{C}_k \widetilde{V}_0^T \mathsf{V}_0. \end{aligned} \tag{3.45}$$

Aplicando normas, temos:

$$\operatorname{sen}(\Omega_k) \leq \frac{\sigma_{k+1}^2}{\sigma_{\min}(B_k)^2} \operatorname{sen}(\Omega_k) + \frac{\|\widetilde{C}_k\|_2}{\sigma_{\min}(B_k)}. \tag{3.46}$$

Usando a hipótese de que $\sigma_{\min}(B_k) > \sigma_{k+1}$ e o fato de que $\|\widetilde{C}_k\|_2 \leq \alpha_{k+1}$, temos

$$\operatorname{sen}(\Omega_k) \leq \frac{\sigma_{\min}(B_k)\alpha_{k+1}}{\sigma_{\min}(B_k)^2 - \sigma_{k+1}^2}. \tag{3.47}$$

□

Uma consequência imediata deste teorema é que também temos uma cota inferior para $\operatorname{sen}(\Omega_k)$ que não necessita da hipótese $\sigma_{\min}(B_k) > \sigma_{k+1}$.

Corolário 3.6. *Seja $\eta_k = \sigma_{k+1}/\sigma_k$, então uma cota inferior para* $\operatorname{sen}(\Omega_k)$ *é dada por*

$$\frac{\|\widetilde{C}_k\|_2}{\sigma_k + \|\widetilde{F}_k\|_2 \eta_k} \leq \operatorname{sen}(\Omega_k). \tag{3.48}$$

Demonstração. Das equações (3.39) e (3.42) segue que

$$\widetilde{C}_k = \widetilde{U}_k^T \mathsf{U}_k \Sigma_k \mathsf{V}_k^T \widetilde{V}_0 + \widetilde{U}_k^T \mathsf{U}_0 \Sigma_0 \mathsf{V}_0^T \widetilde{V}_0. \tag{3.49}$$

Aplicando normas, temos:

$$\begin{aligned} \|\widetilde{C}_k\|_2 &\leq \|\widetilde{U}_k^T \mathsf{U}_k \Sigma_k \mathsf{V}_k^T \widetilde{V}_0\|_2 + \|\widetilde{U}_k^T \mathsf{U}_0 \Sigma_0 \mathsf{V}_0^T \widetilde{V}_0\|_2 \\ &\leq \|\widetilde{U}_k^T \mathsf{U}_k \Sigma_k \mathsf{V}_k^T \widetilde{V}_0\|_2 + \|\widetilde{U}_k^T \mathsf{U}_0\|_2 \|\Sigma_0\|_2 \\ &\leq \sigma_k \operatorname{sen}(\Omega_k) + \|\mathsf{U}_k^T \widetilde{U}_0\|_2 \sigma_{k+1}, \end{aligned} \tag{3.50}$$

em que usamos o fato de que $\|\widetilde{U}_k^T \mathsf{U}_0\|_2 = \|\mathsf{U}_k^T \widetilde{U}_0\|_2$. Das equações (3.39) e (3.42) segue que $\mathsf{U}_k^T \widetilde{U}_0 = \Sigma_k^{-1} \mathsf{V}_k^T \widetilde{V}_0 \widetilde{F}_k^T$. Logo, $\|\mathsf{U}_k^T \widetilde{U}_0\|_2 \leq \sigma_k^{-1} \|\widetilde{F}_k\|_2 \operatorname{sen}(\Omega_k)$ e, portanto,

$$\frac{\|\widetilde{C}_k\|_2}{\sigma_k + \|\widetilde{F}_k\|_2 \eta_k} \leq \operatorname{sen}(\Omega_k). \tag{3.51}$$

□

Tanto o Teorema 3.5 quanto o Corolário 3.6 são semelhantes a um resultado devido a Fierro e Bunch [44, Teo. 2.2]. Entretanto, Fierro e Bunch consideram um caso particular de subespaços, ou melhor, consideram decomposições do tipo URV/ULV e a decomposição (3.40) não pode ser considerada uma decomposição deste tipo, pois a matriz B_k não é quadrada. Exibimos na Figura 3.11 tanto as cotas superiores (3.37) e (3.38) denotadas, respectivamente, por "Cota A" e "Cota B", bem como o $\operatorname{sen}(\Omega_k)$ e a cota inferior (3.48) denotada por "Cota C" para as 20 primeiras iterações do processo GKB aplicados aos problemas i_laplace (também disponível em [64]) e shaw com $n = 512$ e NL = 0,01. Vale lembrar que, no caso da "Cota B", são exibidos apenas os índices k tais que $\sigma_{\min}(B_k) > \sigma_{k+1}$. Analisando ambos os gráficos podemos concluir que:

1. a "Cota A", apesar de existir para todo k, nem sempre é menor do que a "Cota B". Logo, uma boa cota superior para $\operatorname{sen}(\Omega_k)$ é considerar o mínimo entre (3.37) e (3.38);

2. a equação (3.48) fornece, de modo geral, uma boa estimativa inferior para $\operatorname{sen}(\Omega_k)$. Entretanto, devido às imprecisões numéricas por estarmos lidando com problemas mal-postos discretos, podem existir parâmetros k tais que a equação (3.48) não seja verificada. Podemos notar que para os parâmetros $k = 12$ e $k = 18$ no problema i_laplace, cujo número de condição da matriz é $\kappa(A_{\text{i\_laplace}}) > 10^{34}$, e para $k = 14$ no problema shaw, cujo número de condição da matriz é $\kappa(A_{\text{shaw}}) \approx 4,41 \times 10^{20}$, esta inconsistência ocorre.

O próximo teorema é o resultado principal desta seção e fornece uma cota superior para a distância entre a solução f_k e a solução f_k calculada pela TSVD.

Teorema 3.7. *Seja $V_k = [v_1, \ldots, v_k] \in \mathbb{R}^{n \times k}$ uma matriz com colunas ortonormais tal que $\operatorname{span}\{v_1, \ldots, v_k\} = \mathcal{V}_k$. Então, a distância relativa entre as soluções f_k dadas por (3.1) e as*

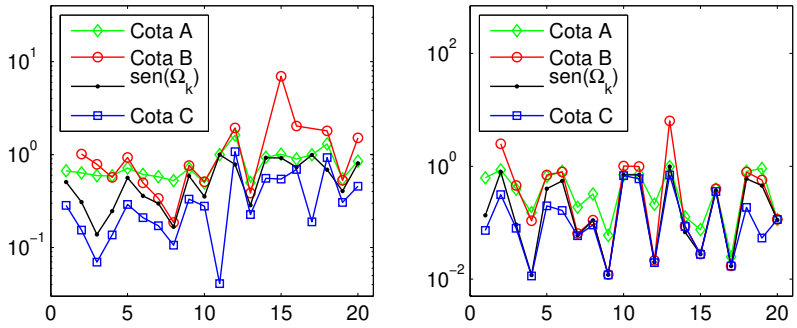


Figura 3.11: Cotas (3.37), (3.38) e (3.48) para os problemas i_laplace (esquerda) e shaw (direita) com $n = 512$ e NL = 0,01.

soluções TSVD f_k *pode ser limitada por*

$$\frac{\|\mathsf{f}_k - f_k\|_2}{\|f_k\|_2} \leq \frac{1}{\sigma_k}(\phi_k + \gamma_k), \tag{3.52}$$

em que $\phi_k = \|g - Af_k\|_2/\|f_k\|_2$ *e* γ_k *dado por (2.28). Além disso, sob as hipóteses do Teorema 3.5, temos*

$$\frac{\|\mathsf{f}_k - f_k\|_2}{\|f_k\|_2} \leq \frac{\phi_k}{\sigma_k} + \frac{\sigma_{\min}(B_k)\alpha_{k+1}}{\sigma_{\min}(B_k)^2 - \sigma_{k+1}^2}. \tag{3.53}$$

Demonstração. Sejam $r_k = g - Af_k$ e $f_k = V_k d_k$ em que $d_k = (AV_k)^\dagger g$. Então

$$\mathsf{f}_k - f_k = \mathsf{A}_k^\dagger g - V_k(AV_k)^\dagger g = \mathsf{A}_k^\dagger r_k + \mathsf{A}_k^\dagger A f_k - V_k(AV_k)^\dagger A f_k, \tag{3.54}$$

em que usamos o fato de que $V_k(AV_k)^\dagger r_k = 0$. Notemos que

$$V_k(AV_k)^\dagger A f_k = V_k(AV_k)^\dagger AV_k V_k^T f_k = V_k V_k^T f_k = f_k, \tag{3.55}$$

e, como $\mathsf{A}_k^\dagger A = \mathsf{A}_k^\dagger \mathsf{A}_k$, segue que

$$\mathsf{f}_k - f_k = \mathsf{A}_k^\dagger r_k + (\mathsf{A}_k^\dagger \mathsf{A}_k - I_n) f_k = \mathsf{A}_k^\dagger r_k - (I_n - \mathsf{P}_k) P_k f_k, \tag{3.56}$$

em que $P_k = V_k V_k^T$ e $\mathsf{P}_k = \mathsf{A}_k^\dagger \mathsf{A}_k$ são os projetores ortogonais sobre $\mathcal{V}_k$ e $\mathcal{S}_k$. Portanto,

$$\begin{aligned} \|\mathsf{f}_k - f_k\|_2 &\leq \|\mathsf{A}_k^\dagger\|_2 \|r_k\|_2 + \|(I_n - \mathsf{P}_k)P_k\|_2 \|f_k\|_2, \\ &\leq \|f_k\|_2 \phi_k/\sigma_k + \|f_k\|_2 \gamma_k/\sigma_k, \end{aligned} \tag{3.57}$$

em que usamos novamente o fato de que $\|(I_n - \mathsf{P}_k)P_k\|_2 = \operatorname{sen}(\Omega_k)$ e o Teorema 3.4. A segunda parte segue imediatamente da equação (3.57) e do Teorema 3.5.

□

Teorema 3.8. *Sejam $\mathcal{V}_k$ o subespaço de Krylov $\mathcal{K}_k(A^TA, A^Tg)$, $\mathcal{S}_k$ dado por (2.41) e $A_k = U_{k+1}B_kV_k^T$. Então*

$$\frac{\|\mathsf{f}_k - f_k\|_2}{\|f_k\|_2} \leq \operatorname{sen}(\Omega_k) \left(\frac{\sigma_1}{\sigma_k} \frac{\|r_k\|_2}{\|g\|_2} \frac{\sigma_{k+1}}{\tau_k \sigma_{min}(R_k)} + 1 \right), \quad \tau_k = \sqrt{1 - \frac{\|r_k\|_2^2}{\|g\|_2^2}}. \tag{3.58}$$

Demonstração. Da equação (3.42) segue que $A_k = \widetilde{U}_k D_k \widetilde{V}_k^T$, então

$$\begin{aligned} \mathsf{f}_k - f_k &= (\mathsf{A}_k^\dagger - A_k^\dagger)g \\ &= (\mathsf{A}_k^\dagger - A_k^\dagger)g - (\mathsf{A}_k^\dagger \mathsf{A}_k - A_k^\dagger A_k)f_k + (\mathsf{A}_k^\dagger \mathsf{A}_k - A_k^\dagger A_k)f_k \\ &= (\mathsf{A}_k^\dagger - A_k^\dagger)g - (\mathsf{A}_k^\dagger A - A_k^\dagger A + \widetilde{V}_k D_k^{-1} \widetilde{C}_k \widetilde{V}_0^T)f_k + (\mathsf{A}_k^\dagger \mathsf{A}_k - A_k^\dagger A_k)f_k \\ &= (\mathsf{A}_k^\dagger - A_k^\dagger)r_k + (\mathsf{A}_k^\dagger \mathsf{A}_k - A_k^\dagger A_k)f_k \\ &= \mathsf{A}_k^\dagger r_k + (\mathsf{A}_k^\dagger \mathsf{A}_k - A_k^\dagger A_k)f_k, \end{aligned} \tag{3.59}$$

em que usamos os fatos de que $\mathsf{A}_k^\dagger \mathsf{A}_k = \mathsf{A}_k^\dagger A$, $A = A_k + \widetilde{U}_k \widetilde{C}_k \widetilde{V}_0^T + \widetilde{U}_0 \widetilde{F}_k \widetilde{V}_0^T$, $r_k = g - Af_k$, $\widetilde{V}_0^T f_k = 0$ e $A_k^\dagger r_k = 0$. Como $\mathsf{A}_k^\dagger = \mathsf{A}_k^\dagger \mathsf{U}_k \mathsf{U}_k^T$ e $r_k = \widetilde{U}_k^\perp \widetilde{U}_k^{\perp T} r_k$, temos:

$$\mathsf{f}_k - f_k = \mathsf{A}_k^\dagger \mathsf{U}_k \mathsf{U}_k^T \widetilde{U}_k^\perp \widetilde{U}_k^{\perp T} r_k + (\mathsf{A}_k^\dagger \mathsf{A}_k - A_k^\dagger A_k)f_k. \tag{3.60}$$

Assim, segue que

$$\|\mathsf{f}_k - f_k\|_2 \leq \|\mathsf{A}_k^\dagger\|_2 \|\mathsf{U}_k^T \widetilde{U}_k^\perp\|_2 \|r_k\|_2 + \|(\mathsf{A}_k^\dagger \mathsf{A}_k - A_k^\dagger A_k)\|_2 \|f_k\|_2. \tag{3.61}$$

Usando a desigualdade[1]

$$\frac{1}{\|f_k\|_2}\sqrt{1-\frac{\|r_k\|_2^2}{\|g\|_2^2}} \leq \frac{\|A\|_2}{\|g\|_2}, \tag{3.62}$$

e considerando que Π_k é o ângulo entre os subespaços $\mathcal{R}(\mathsf{U}_k)$ e $\mathcal{R}(\widetilde{U}_k)$, temos

$$\frac{\|\mathsf{f}_k - f_k\|_2}{\|f_k\|_2} \leq \frac{\sigma_1}{\sigma_k}\frac{\|r_k\|_2}{\|g\|_2}\frac{\operatorname{sen}(\Pi_k)}{\tau_k} + \operatorname{sen}(\Omega_k), \quad \tau_k = \sqrt{1-\frac{\|r_k\|_2^2}{\|g\|_2^2}}. \tag{3.63}$$

Usando as equações (3.39) e (3.42), podemos provar que $\operatorname{sen}(\Pi_k) \leq \dfrac{\sigma_{k+1}}{\sigma_{\min}(B_k)}\operatorname{sen}(\Omega_k)$, então

$$\frac{\|\mathsf{f}_k - f_k\|_2}{\|f_k\|_2} \leq \operatorname{sen}(\Omega_k)\left(\frac{\sigma_1}{\sigma_k}\frac{\|r_k\|_2}{\|g\|_2}\frac{\sigma_{k+1}}{\tau_k\sigma_{\min}(R_k)} + 1\right), \quad \tau_k = \sqrt{1-\frac{\|r_k\|_2^2}{\|g\|_2^2}}. \tag{3.64}$$

□

A principal conclusão que podemos tirar do Teorema 3.8 é que as soluções iteradas estarão próximas das soluções TSVD desde que o seno entre os subespaços seja pequeno. Porém, na prática, é difícil monitorar esta quantidade e o melhor que podemos fazer é analisar alguma cota superior, conforme os Teoremas 3.4 e 3.5, em particular, a quantidade ϕ_k/σ_k.

Na Figura 3.12 podemos apreciar como o erro $E_k = \|f_k - \mathsf{f}_k\|_2/\|f_k\|_2$ se comporta quando as soluções f_k são calculadas pelo método LSQR (gráfico a esquerda). No gráfico central temos como a quantidade ϕ_k se comporta em função de σ_k e, finalmente, no gráfico a direita a função $\Psi_k = \|r_k\|_2\|f_k\|_2$ para as 10 primeiras iterações do algoritmo LSQR aplicado ao problema i_laplace com $n = 512$ e NL=0,01. Podemos perceber que o parâmetro encontrado pelo critério (3.29) coincide com o parâmetro k^* que produz o menor erro E_k. Além disso, como para $k \geq 8$ temos que $\phi_k/\sigma_k > 1$, o que significa que as cotas (3.52) e (3.53) passam a não ter muito mais serventia. Isto é um forte indicativo de que avançar com as iterações provavelmente deteriorará a qualidade das soluções iteradas f_k com relação às soluções TSVD.

[1]Isto vem do fato de que $g = r_k + Af_k$ e $r_k \perp Af_k$.

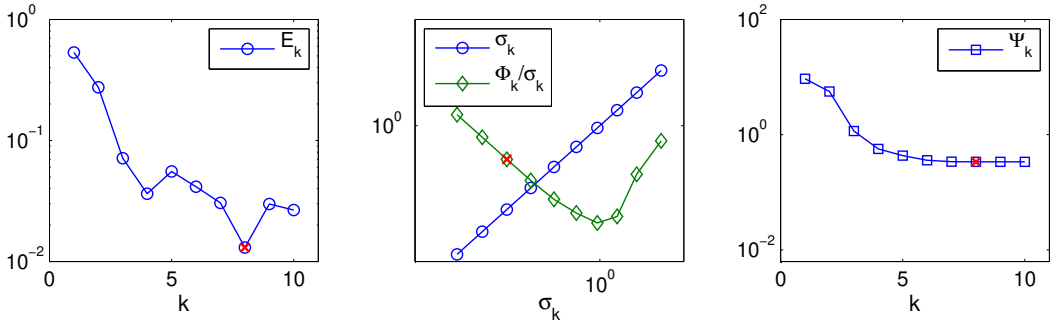


Figura 3.12: Erro $E_k = \|f_k - \mathsf{f}_k\|_2/\|f_k\|_2$ (esquerda), parâmetro ϕ_k em função de σ_k (centro) e gráfico da função Ψ_k para as 10 primeiras iterações do método LSQR para o problema i_laplace com $n = 512$ e NL = 0,01. O $\times$ denota os valores para $k = 8$, minimizador da função Ψ_k.

3.4 Exemplos numéricos - parte 1

Nesta seção temos dois propósitos distintos. O primeiro consiste em comparar o desempenho do critério de parada automático (3.29) em conexão com métodos como LSQR, MR-II, RRGMRES, todos com reortogonalização completa, e TSVD. O segundo tem por objetivo ilustrar o por que de abdicarmos o uso do algoritmo *pruning* para a escolha do parâmetro. Todos os experimentos foram realizados em um PC com processador Core 2 Duo 2,53GHz, 4 GB de memória RAM, sistema operacional Linux e Matlab [100] versão 7.

3.4.1 Métodos de subespaços equipados com o critério de parada automático

Vamos ilustrar o potencial do critério de parada (3.29) em conexão com os algoritmos LSQR, RRGMRES (se $A \neq A^T$), MR-II (se $A = A^T$) e TSVD aplicados aos seguintes problemas: foxgood, shaw, deriv2, phillips, heat e baart com dimensão $n = 800$, todos disponíveis no pacote [64], e um problema de restauração de imagem do tipo *deblurring* com uma imagem de tamanho 175×175 pixels que resulta em um problema com 30.625 variáveis [70].

Iniciamos com os seis problems obtidos no pacote RegularizationTools e com três níveis de ruídos diferentes, NL = 0,001, 0,01, 0,025. Nas Tabelas 3.1 a 3.6 apresentamos os parâmetros mínimos(máximos) determinados pelo critério (3.29), denotados por $\widehat{k}_m(\widehat{k}_M)$, os parâmetros mínimos(máximos) ótimos (entendemos por parâmetro ótimo como aquele que minimiza $\|f_k -$

$f_{\text{exato}}\|_2/\|f_{\text{exato}}\|_2)$, denotados por $k_m^*(k_M^*)$, e os respectivos erros médios $E_{\widehat{k}}$, E_{k^*}. Além disso, reportamos os valores médios do parâmetro de regularização λ determinado pelo método FP, o parâmetro ótimo e os respectivos erros médios E_λ. Como estamos mais interessados na qualidade (número de iterações, erro relativo, etc.) das soluções obtidas, não reportamos os tempos gastos.

	NL = 0,001			NL = 0,01			NL = 0,025		
	LSQR	MR-II	TSVD	LSQR	MR-II	TSVD	LSQR	MR-II	TSVD
$\widehat{k}_m(\widehat{k}_M)$	3(4)	3(4)	4(4)	2(2)	2(2)	2(2)	2(2)	2(2)	2(2)
$k_m^*(k_M^*)$	3(4)	3(4)	3(4)	2(3)	2(3)	2(3)	2(3)	2(3)	2(3)
$E_{\widehat{k}}$	0,0217	0,0211	0,0193	0,0311	0,0312	0,0312	0,0319	0,0319	0,0320
E_{k^*}	0,0080	0,0080	0,0078	0,0256	0,0256	0,0249	0,0308	0,0308	0,0296
SVD	FP		ótimo	FP		ótimo	FP		ótimo
λ	0,0008		0,0020	0,0077		0,0101	0,0195		0,0171
E_λ	0,0169		0,0073	0,0266		0,0215	0,0334		0,0279

Tabela 3.1: Parâmetros mínimos(máximos) encontrados por (3.29) e parâmetros mínimos(máximos) ótimos para LSQR, MR-II e TSVD e respectivos erros médios. Média dos parâmetro de regularização determinados por FP e ótimo e respectivos erros médios para o problema foxgood.

	NL = 0,001			NL = 0,01			NL = 0,025		
	LSQR	MR-II	TSVD	LSQR	MR-II	TSVD	LSQR	MR-II	TSVD
$\widehat{k}_m(\widehat{k}_M)$	7(8)	7(8)	7(8)	5(6)	5(7)	7(7)	4(4)	4(4)	4(5)
$k_m^*(k_M^*)$	7(9)	7(9)	7(10)	5(8)	5(8)	6(7)	5(7)	5(7)	5(7)
$E_{\widehat{k}}$	0,0498	0,0500	0,0500	0,0775	0,0898	0,0670	0,1683	0,1688	0,1679
E_{k^*}	0,0427	0,0430	0,0440	0,0637	0,0637	0,0665	0,0969	0,0952	0,1036
SVD	FP		ótimo	FP		ótimo	FP		ótimo
λ	0,0023		0,0032	0,0235		0,0118	0,0593		0,0220
E_λ	0,0463		0,0388	0,0816		0,0651	0,1346		0,0983

Tabela 3.2: Parâmetros mínimos(máximos) encontrados por (3.29) e parâmetros mínimos(máximos) ótimos para LSQR, MR-II e TSVD e respectivos erros médios. Média dos parâmetro de regularização determinados por FP e ótimo e respectivos erros médios para o problema shaw.

De modo geral, tanto o método LSQR como MR-II/RRGMRES comportaram-se de modo semelhante produzindo bons resultados em todos os experimentos, exceto RRGMRES no problema heat, Tabela 3.5, que não foi capaz de encontrar nenhuma solução f_k apropriada, pois o sucesso deste método é altamente dependente do problema em questão. Além disso,

	NL = 0,001			NL = 0,01			NL = 0,025		
	LSQR	MR-II	TSVD	LSQR	MR-II	TSVD	LSQR	MR-II	TSVD
$\widehat{k}_m(\widehat{k}_M)$	16(17)	13(13)	17(34)	7(7)	6(6)	9(10)	5(5)	4(5)	5(6)
$k^*_m(k^*_M)$	13(16)	11(12)	25(35)	7(9)	6(8)	10(17)	5(7)	5(6)	8(14)
$E_{\widehat{k}}$	0,1474	0,1523	0,1525	0,2145	0,2161	0,2323	0,2656	0,2689	0,2949
E_{k^*}	0,1392	0,1399	0,1451	0,2019	0,2027	0,2063	0,2344	0,2364	0,2406
SVD	FP		ótimo	FP		ótimo	FP		ótimo
λ	0,00001		0,0001	0,0008		0,0006	0,0023		0,0012
E_λ	0,1588		0,1402	0,2103		0,2028	0,2616		0,2355

Tabela 3.3: Parâmetros mínimos(máximos) encontrados por (3.29) e Parâmetros mínimos(máximos) ótimos para LSQR, MR-II e TSVD e respectivos erros médios. Média dos parâmetro de regularização determinados por FP e ótimo e respectivos erros médios para o problema deriv2.

	NL = 0,001			NL = 0,01			NL = 0,025		
	LSQR	MR-II	TSVD	LSQR	MR-II	TSVD	LSQR	MR-II	TSVD
$\widehat{k}_m(\widehat{k}_M)$	11(15)	7(17)	12(25)	9(10)	8(10)	11(11)	7(8)	7(8)	8(8)
$k^*_m(k^*_M)$	9(10)	9(10)	11(12)	5(9)	4(9)	7(11)	5(8)	4(7)	7(9)
$E_{\widehat{k}}$	0,0617	0,0343	0,0497	0,0374	0,0347	0,0280	0,0327	0,0335	0,0276
E_{k^*}	0,0072	0,0077	0,0065	0,0223	0,0196	0,0165	0,0255	0,0246	0,0224
SVD	FP		ótimo	FP		ótimo	FP		ótimo
λ	0,0050		0,0427	0,0505		0,1532	0,1268		0,2428
E_λ	0,0732		0,0081	0,0455		0,0209	0,0403		0,0281

Tabela 3.4: Parâmetros mínimos(máximos) encontrados por (3.29) e Parâmetros mínimos(máximos) ótimos para LSQR, MR-II e TSVD e respectivos erros médios. Média dos parâmetro de regularização determinados por FP e ótimo e respectivos erros médios para o problema phillips.

	NL = 0,001			NL = 0,01			NL = 0,025		
	LSQR	RRGM	TSVD	LSQR	RRGM	TSVD	LSQR	RRGM	TSVD
$\widehat{k}_m(\widehat{k}_M)$	28(29)	18(24)	17(33)	16(16)	17(25)	17(24)	11(11)	14(24)	16(18)
$k^*_m(k^*_M)$	20(22)	1(1)	29(34)	13(15)	1(1)	21(28)	11(13)	1(1)	16(23)
$E_{\widehat{k}}$	0,0812	$1{,}3\times10^6$	0,0660	0,0798	$8{,}8\times10^5$	0,0762	0,1091	$7{,}4\times10^5$	0,1129
E_{k^*}	0,0253	1,0756	0,0233	0,0711	1,0756	0,0681	0,1021	1,0755	0,1069
SVD	FP		ótimo	FP		ótimo	FP		ótimo
λ	0,0002		0,0008	0,0019		0,0026	0,0049		0,0041
E_λ	0,0869		0,0285	0,0851		0,0772	0,1185		0,1141

Tabela 3.5: Parâmetros mínimos(máximos) encontrados por (3.29) e parâmetros mínimos(máximos) ótimos para LSQR, RRGMRES (denotado por RRGM) e TSVD e respectivos erros médios. Média dos parâmetro de regularização determinados por FP e ótimo e respectivos erros médios para o problema heat.

	NL = 0,001			NL = 0,01			NL = 0,025		
	LSQR	RRGM	TSVD	LSQR	RRGM	TSVD	LSQR	RRGM	TSVD
$\widehat{k}_m(\widehat{k}_M)$	4(4)	3(5)	4(4)	3(3)	3(4)	3(3)	3(3)	3(4)	3(3)
$k^*_m(k^*_M)$	4(5)	3(4)	4(5)	3(4)	3(4)	3(4)	3(4)	3(4)	3(4)
$E_{\widehat{k}}$	0,1159	0,0402	0,1160	0,1662	0,0569	0,1668	0,1684	0,0655	0,1691
E_{k^*}	0,1027	0,0337	0,1028	0,1459	0,0385	0,1461	0,1614	0,0483	0,1629
SVD	FP	ótimo		FP	ótimo		FP	ótimo	
λ	0,0023	0,0009		0,0237	0,0047		0,0615	0,0117	
E_λ	0,1167	0,0848		0,1647	0,1171		0,2089	0,1365	

Tabela 3.6: Parâmetros mínimos(máximos) encontrados por (3.29) e parâmetros mínimos(máximos) ótimos para LSQR, RRGMRES (denotado por RRGM) e TSVD e respectivos erros médios. Média dos parâmetro de regularização determinados por FP e ótimo e respectivos erros médios para o problema baart.

os resultados obtidos por esta estratégia foram semelhantes aos resultados obtidos através do método de regularização de Tikhonov.

Para o problema de restauração de imagem, a matriz de coeficientes é do tipo $A = T \otimes T \in \mathbb{R}^{30.625 \times 30.625}$, com $T \in \mathbb{R}^{175 \times 175}$ sendo uma matriz Toeplitz e simétrica. A imagem exata e duas imagens com ruído, uma com NL = 0,001 e outra com NL = 0,05, são exibidas na Figura 3.13. Pelo fato deste problema ter um tamanho relativamente grande, na Tabela 3.7 usamos o algoritmo GKB-FP aplicado ao método de regularização de Tikhonov. Optamos por incluir os resultados com NL = 0,05, pois em [70] os experimentos numéricos são realizados com este nível de ruído. Vale ressaltar que em [70], nada é comentado sobre a iteração de parada, ou seja, houve apenas um estudo de como as iterações evoluem. Em contrapartida, aqui nós mostramos como parar as iterações de modo automático e que este critério produz bons resultados. Desconsiderando o número de iterações, tanto o LSQR quanto o MR-II produziram resultados com qualidade semelhante. Por este motivo, exibimos apenas as imagens reconstruídas pelo algoritmo LSQR para cada um dos níveis de ruído nos dados, ou seja, NL = 0,001, 0,01, 0,025 e 0,05 (veja Figura 3.14).

3.4.2 Dificuldades com o algoritmo *pruning*

Para ilustrarmos a performance do método *pruning* em conexão com o algoritmo LSQR com reortogonalização completa, vamos realizar um pequeno experimento numérico com oito

	NL = 0,001		NL = 0,01		NL = 0,025		NL = 0,05	
	LSQR	MR-II	LSQR	MR-II	LSQR	MR-II	LSQR	MR-II
$\widehat{k}_m(\widehat{k}_M)$	535(545)	69(70)	90(93)	22(22)	39(40)	13(14)	20(21)	9(9)
$k^*_m(k^*_M)$	247(272)	40(43)	43(47)	14(15)	22(23)	9(10)	13(15)	7(7)
$E_{\widehat{k}}$	0,1618	0,1616	0,1672	0,1673	0,1702	0,1698	0,1733	0,1722
E_{k^*}	0,1494	0,1492	0,1577	0,1575	0,1629	0,1629	0,1689	0,1688
	GKB-FP		GKB-FP		GKB-FP		GKB-FP	
$k_m(k_M)$	232(247)		176(181)		97(99)		58(59)	
λ	0,0009		0,0092		0,0231		0,0465	
E_λ	0,1494		0,1675		0,1710		0,1744	

Tabela 3.7: Parâmetros mínimos(máximos) encontrados por (3.29) e parâmetros mínimos(máximos) ótimos para LSQR e MR-II e respectivos erros médios. Média dos parâmetros de regularização determinados por GKB-FP e respectivos erros médios para o problema de restauração de imagem.

Figura 3.13: Imagem exata e imagens capturadas com NL = 0,001 e 0,05.

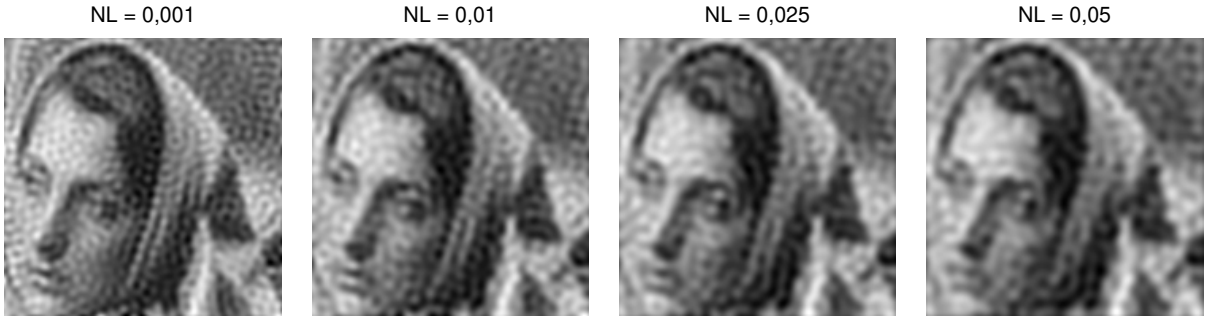


Figura 3.14: Solução obtida pelo algoritmo LSQR usando o critério de parada (3.29) e NL = 0,001, 0,01, 0,025, 0,05.

problemas, a saber: gravity, heat, foxgood e shaw disponíveis em [64], e os problemas moler (com $\alpha = 0,5$), lotkin, prolate e hilbert disponíveis na "matrix gallery" do Matlab. Em todos os casos, as matrizes têm tamanho 1.024×1.024 e são usados vinte vetores de dados da forma $g = g_{\text{exato}} + \epsilon$, em que ϵ é gerado pela função randn do Matlab e normalizados tal que

NL $= \|\epsilon\|_2/\|g_{\text{exato}}\|_2 = 10^{-4}, 10^{-3}, 10^{-2}$. Para garantirmos que o "canto" da curva-L esteja bem definido, vamos considerar $\mathsf{q} = 120$ pontos. O "canto" da curva-L, denotado por k_{LC}, é escolhido pela rotina corner (que implementa o algoritmo *pruning*) disponível em [64] e o erro relativo na solução correspondente, $f_{k_{\text{LC}}}$ é denotado por E_{LC}.

Para comparação, o parâmetro $\widehat{k}$ escolhido como o minimizador de Ψ_k, o erro relativo na solução $f_{\widehat{k}}$ denotado por E_Ψ, parâmetros ótimos e erros ótimos, denotados por k_{opt} e E_{opt}, respectivamente, também são calculados. Apresentamos na Tabela 3.8 a média dos valores obtidos após as vinte realizações, bem como os parâmetros mínimos/máximos determinados pelo algoritmo *pruning*, pelo critério (3.29) e os parâmetros ótimos.

Podemos notar que, para todos os problemas, as soluções produzidas pelo método *pruning* e pelo critério de parada (3.29) são semelhantes em qualidade. As únicas exceções foram os problemas moler com NL $= 10^{-4}$ e lotkin com NL $= 10^{-2}$. Para o problema moler, enquanto que o *pruning* produziu soluções de baixa qualidade, as soluções produzidas por (3.29) foram razoáveis.

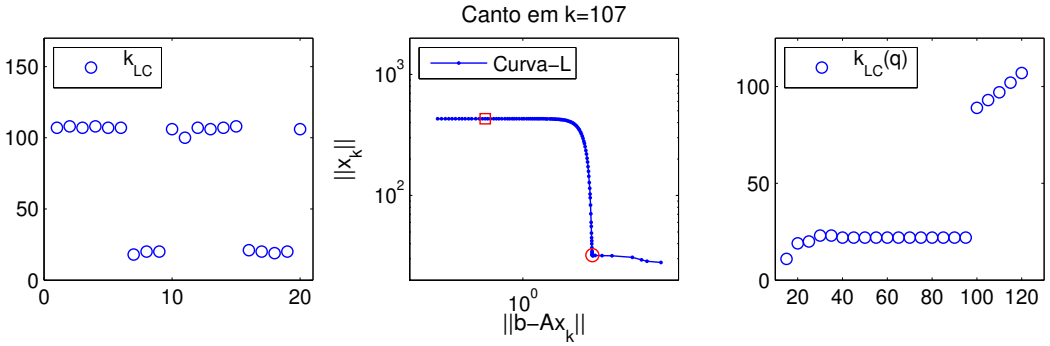


Figura 3.15: "Canto" da curva-L determinado pelo algoritmo *pruning* nas vinte execuções para o problema moler em conexão com LSQR (esquerda). Curva-L discreta considerando o primeiro vetor de dados (centro); neste caso, o "canto" determinado pelo algoritmo *pruning* está marcada com □ enquanto que o "canto" correto está marcado com ○. "Canto" determinado pelo algoritmo pruning usando $\mathsf{q} = 30, \ldots, 120$ pontos (direita).

O algoritmo *pruning* falhou em encontrar soluções aceitáveis pois o "canto" da curva-L não foi corretamente identificado diversas vezes, conforme Figura 3.15 (esquerda). Esta mesma figura mostra a curva-L discreta usando o primeiro vetor de dados g com o "canto" determinado pelo algoritmo *pruning* (calculado como $k_{\text{LC}} = 107$), marcado com um □ (em vermelho), e o "canto" correto localizado em $k = 19$, marcado com um ○ (em vermelho).

	NL= 10^{-4}					
	k_{LC}	$\widehat{k}$	k_{opt}	E_{LC}	E_{Ψ}	$\mathrm{E}_{\mathrm{opt}}$
gravity	13(19)	11(14)	11(13)	0,0522	0,0109	0,0043
heat	46(52)	28(42)	28(31)	0,0902	0,0175	0,0125
foxgood	4(5)	5(5)	4(4)	0,0096	0,0119	0,0028
shaw	9(9)	9(9)	9(9)	0,0325	0,0325	0,0325
moler	18(108)	19(21)	9(11)	8,7458	0,1283	0,0107
lotkin	7(7)	7(7)	7(9)	0,4384	0,4384	0,4317
prolate	20(22)	10(12)	9(13)	0,0223	0,0002	0,0002
hilbert	9(10)	9(9)	10(12)	0,4358	0,4382	0,4258
	NL= 10^{-3}					
	k_{LC}	$\widehat{k}$	k_{opt}	E_{LC}	E_{Ψ}	$\mathrm{E}_{\mathrm{opt}}$
gravity	11(15)	10(11)	9(11)	0,0368	0,0224	0,0118
heat	27(31)	28(29)	20(22)	0,0774	0,0691	0,0222
foxgood	3(4)	3(4)	3(3)	0,0201	0,0201	0,0074
shaw	7(8)	7(8)	7(9)	0,0514	0,0515	0,0439
moler	9(10)	9(10)	7(8)	0,0496	0,0654	0,0220
lotkin	3(5)	5(5)	5(7)	0,4478	0,4475	0,4445
prolate	17(17)	12(16)	6(11)	0,0260	0,0145	0,0008
hilbert	7(8)	7(8)	7(9)	0,4396	0,4396	0,4391
	NL= 10^{-2}					
	k_{LC}	$\widehat{k}$	k_{opt}	E_{LC}	E_{Ψ}	$\mathrm{E}_{\mathrm{opt}}$
gravity	7(11)	7(8)	6(8)	0,0454	0,0356	0,0266
heat	15(17)	16(16)	14(16)	0,0674	0,0674	0,0629
foxgood	2(2)	2(2)	2(3)	0,0311	0,0311	0,0217
shaw	5(5)	5(6)	6(8)	0,1094	0,0660	0,0534
moler	4(4)	4(4)	5(6)	0,1885	0,1885	0,0788
lotkin	3(18)	3(3)	3(5)	$1,9 \times 10^5$	0,4522	0,4505
prolate	10(13)	7(12)	1(6)	0,0173	0,0150	0,0071
hilbert	6(6)	6(6)	6(7)	0,4400	0,4400	0,4400

Tabela 3.8: Parâmetros de regularização e erros relativos nas soluções regularizadas determinadas pelo algoritmo *pruning*, pela regra (3.29) e os parâmetros ótimos. A solução exata para os problemas moler, lotkin, prolate e hilbert foi considerada como a mesma do problema shaw.

Uma observação similar vale para o problema lotkin.

Para melhor entendermos as propriedades do algoritmo *pruning*, vamos investigar o comportamento do parâmetro k_{LC} como função do número de pontos q, ou seja, $k_{\mathrm{LC}}(\mathsf{q})$. Os resultados para o problema moler com NL = 10^{-4} são exibidos na Figura 3.15 (direita). Analisando o resultado, podemos concluir que, para diversos valores de q, o "canto" k_{LC} estabiliza

em $k = 21$, o que coincide com o maior minimizador da função Ψ_k encontrado, e que, para valores grandes de q, este "canto" é localizado muito distante de $k = 21$. Os resultados exibidos na Figura 3.15 sugerem que o parâmetro k_{LC} depende do número de pontos q e que escolhas inadequadas podem conduzir à resultados indesejados. No Capítulo 5 voltaremos a esta questão em conexão com problemas de restauração de imagens para comprovar, mais uma vez, que esta abordagem é altamente dependente do número de pontos utilizados e, portanto, não é muito apropriada para problemas de grande porte uma vez que não sabemos a priori quantos passos devem ser executados.

3.5 Inclusão de informação a priori no processo iterativo

Vimos nos experimentos numéricos que, para o problema deriv2, as soluções iteradas não foram muito satisfatórias. Mas sabemos que, para este problema, a solução desejada tem certas propriedades, ou seja, a solução é diferenciável. Neste caso, a questão natural que surge é como incluir estas informações, contidas em uma matriz L, no processo iterativo. Através das equações (2.51) e (2.52), podemos obter uma matriz $\overline{A}$ que incorpora estas informações. Vamos aplicar os métodos descritos anteriormente, ou seja, LSQR, MINRES/MR-II e GMRES/RRGMRES, para construir uma sequência de soluções iteradas para

$$\overline{f}_{\text{LS}} = \underset{\overline{f} \in \mathbb{R}^p}{\operatorname{argmin}} \|\overline{g} - \overline{A}\overline{f}\|_2^2. \tag{3.65}$$

Uma vez determinada a iteração de parada, digamos k^*, transformamos esta solução para o cenário original, ou seja, $f_{k^*} = L_A^{\dagger}\overline{f}_{k^*} + f_N$.

3.5.1 O uso dos métodos MINRES/MR-II e GMRES/RRGMRES

Os métodos MINRES/MR-II e GMRES/RRGMRES só podem ser aplicados caso a matriz $\overline{A}$ seja quadrada. Para isto, vamos supor que A é quadrada e que L tenha mais colunas do que linhas e posto linha completo. Veremos duas abordagens distintas de como aplicar estes métodos. A primeira por Calvetti et al. [28] e a segunda por Hansen e Jensen [69].

Regularizador aumentado

Para a matriz $\overline{A}$ ser quadrada, a matriz L também deve ser quadrada. Caso $p < n$, Calvetti et al. [28] sugeriu usar uma nova matriz $\overline{L} \in \mathbb{R}^{n\times n}$ que seja, preferencialmente, não-singular para que $\overline{L}_A^{\dagger} = \overline{L}^{-1}$, e que L seja uma submatriz de $\overline{L}$. Por exemplo, se

$$L = \begin{pmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix} \in \mathbb{R}^{(n-1)\times n}, \tag{3.66}$$

a ideia é usar $\overline{L}$ como $\overline{L} = \begin{pmatrix} L \\ w^T \end{pmatrix}$, em que $w \in \mathbb{R}^n$ é um vetor escolhido de modo apropriado.

O interessante desta abordagem, é que a matriz $\overline{L}^{-1}$ atua como precondicionador para o método GMRES, mas não no sentido de acelerar a convergência, e sim para produzir soluções em um subespaço mais apropriado. Usando a notação polinomial, a k-ésima iterada do método GMRES pode ser escrita como $f_k = \mathcal{P}_k(\overline{L}^{-1}A)\overline{L}^{-1}g$, em que $\mathcal{P}_k$ é o polinônimo associado ao método GMRES aplicado ao sistema $A\overline{L}^{-1}f = g$. Assim, as soluções f_k pertencem ao subespaço de Krylov $\mathcal{K}_k(\overline{L}^{-1}A, \overline{L}^{-1}g)$. Esta abordagem apresenta dois pontos críticos:

1. mesmo que a matriz A seja simétrica, não necessariamente $A\overline{L}^{-1}$ será simétrica, isto exclui o uso do algoritmo MINRES/MR-II;

2. utilizar a decomposição QR da matriz $\overline{L} = \overline{Q}\,\overline{R}$ não é conveniente, pois os subespaços de Krylov não são os mesmos, ou seja,

$$\mathcal{K}_k(\overline{L}^{-1}A, \overline{L}^{-1}g) = \mathcal{K}_k(\overline{R}^{-1}\overline{Q}^T A, \overline{R}^{-1}\overline{Q}^T g) \neq \mathcal{K}_k(\overline{R}^{-1}A, \overline{R}^{-1}g). \tag{3.67}$$

A abordagem SN (*Smoothing norm*)

Hansen e Jensen [69] propuseram uma maneira mais elegante para aplicar os métodos MINRES/MR-II e GMRES/RRGMRES mesmo quando $p < n$. A solução iterada obtida após a transformação para o cenário original é $f = L_A^{\dagger}\overline{f} + f_N = L_A^{\dagger}\overline{f} + Wz$, então a equação

$Af = g$ pode ser escrita como

$$A(L_A^\dagger, W) \begin{pmatrix} \overline{f} \\ z \end{pmatrix} = g. \tag{3.68}$$

Multiplicando pela esquerda este sistema por $(L_A^\dagger \quad W)^T$ resulta em um sistema de blocos 2×2

$$\begin{pmatrix} L_A^{\dagger T} A L_A^\dagger & L_A^{\dagger T} A W \\ W^T A L_A^\dagger & W^T A W \end{pmatrix} \begin{pmatrix} \overline{f} \\ z \end{pmatrix} = \begin{pmatrix} L_A^{\dagger T} g \\ W^T g \end{pmatrix}. \tag{3.69}$$

Usando o complemento de Schur (veja, por exemplo, [51]), temos que $S\overline{f} = \overline{d}$, em que

$$\begin{aligned} S &= L_A^{\dagger T} A L_A^\dagger - L_A^{\dagger T} A W (W^T A W)^{-1} W^T A L_A^\dagger = L_A^{\dagger T} P A L_A^\dagger, \\ \overline{d} &= L_A^{\dagger T} g - L_A^{\dagger T} A W (W^T A W)^{-1} W^T g = L_A^{\dagger T} P g, \end{aligned} \tag{3.70}$$

com $P = I - AW(W^T AW)^{-1} W^T$.

A ideia é aplicar os métodos iterativos no sistema $S\overline{f} = \overline{d}$. Assim, quando o método GMRES é aplicado, existe um polinômio tal que a k-ésima solução iterada é $\overline{f}_k = \mathcal{P}_k(L_A^{\dagger T} P A L_A^\dagger) L_A^{\dagger T} P g$. Transformando $\overline{f}_k$ para o cenário original, temos a solução SN-GMRES $f_k = \mathcal{P}_k(L_A^\dagger L_A^{\dagger T} P A) L_A^\dagger L_A^{\dagger T} P g$, o que significa que esta pertence ao subespaço de Krylov $\mathcal{K}_k(L_A^\dagger L_A^{\dagger T} P A, L_A^\dagger L_A^{\dagger T} P g)$, e $L_A^\dagger L_A^{\dagger T} P$ aparece como um precondicionador. Um resultado bastante interessante é que se A é simétrica, então a matriz $L_A^{\dagger T} P A L_A^{\dagger T}$ também é simétrica, permitindo o uso de MINRES/MR-II, resultando nos métodos SN-MINRES e SN-MR-II.

Talvez o resultado mais interessante em [69] com relação a este sistema $S\overline{f} = \overline{d}$, é que se os subespaços $\mathcal{R}(L^T)$ e $\mathcal{R}(AW)$ são complementares, então o sistema $S\overline{f} = \overline{d}$ se reduz a

$$L^{\dagger T} P A L^\dagger \overline{f} = L^{\dagger T} P g, \tag{3.71}$$

ou seja, substituímos $L_A^\dagger$ por $L^\dagger$. No Algoritmo 3.4 apresentamos as ideias elementares do algoritmo SN-X, em que X é qualquer um dos métodos MINRES, MR-II, GMRES e RRGMRES.

Entrada: A, L, g
Saída: Solução regularizada f_k.

1. Aplicamos o algoritmo X ao problema $S\overline{f} = \overline{d}$ até algum criério de parada ser satisfeito obtendo a solução $\overline{f}_k$.
2. Transformamos a solução $\overline{f}_k$ para o cenário original, ou seja, $f_k = L_A^{\dagger}\overline{f}_k + f_N$ em que f_N e $L_A^{\dagger}$ são dados por (2.51)-(2.52).

Algoritmo 3.4: Algoritmo SN-X em que X = MINRES/MR-II ou X = GMRES/RRGMRES.

3.5.2 O uso do método LSQR

Caso a matriz A não seja quadrada, os métodos MINRES/MR-II e GMRES/RRGMRES não podem ser aplicados, restando apenas o método LSQR. Vamos chamar de P-LSQR como sendo o algoritmo LSQR aplicado ao sistema (3.65) e sua descrição está no Algoritmo 3.5.

Entrada: A, L, g
Saída: Solução regularizada f_k

1. Aplicamos o algoritmo LSQR ao problema $\operatorname{argmin} \|\overline{g} - \overline{A}f\|_2^2$ até o critério de parada automático ser satisfeito obtendo a solução $\overline{f}_k$.
2. Transformamos a solução $\overline{f}_k$ para o cenário original, ou seja, $f_k = L_A^{\dagger}\overline{f}_k + f_N$ em que f_N e $L_A^{\dagger}$ são dados por (2.51) - (2.52).

Algoritmo 3.5: Algoritmo P-LSQR.

Claramente que para P-LSQR ser computacionalmente viável, a dimensão do subespaço $\mathcal{N}(L)$ deve ser pequena e os produtos do tipo matriz-vetor envolvendo as matrizes $L^{\dagger}$ e $L^{\dagger T}$ devem ser realizados do modo mais eficiente possível, isto é, todas as possibilidades devem ser exploradas para reduzir o custo computacional, seja através de produtos matriz-vetor rápidos ou através do uso de precondicionadores.

Comparação do custo entre as técnicas SN-GMRES e P-LSQR

Vamos comparar o custo computacional entre os algoritmos com a abordagem SN e o P-LSQR para o caso em que a matriz A não é simétrica, ou seja, vamos usar SN-GMRES.

Na Tabela 3.9 usamos a notação $\Omega = (I_n - W(AW)^{\dagger}A)$ e $P = I_n - AW(W^TAW)^{-1}W^T$. O caso 1 se refere quando os subespaços $\mathcal{R}(L^T)$ e $\mathcal{R}(AW)$ não são complementares e o caso 2 se refere quando estes subespaços são complementares.

Produto	caso 1		caso 2	
Matriz-Vetor	SN-GMRES	P-LSQR	SN-GMRES	P-LSQR
$L^{\dagger}x$ ou ${L^{\dagger}}^{T}x$	2	2	2	2
Ax ou $A^{T}x$	1	2	1	2
Ωx ou $\Omega^{T}x$	2	2	-	2
Px	1	-	1	-

Tabela 3.9: Produtos matriz-vetor para as abordagens SN-GMRES e P-LSQR por iteração.

Podemos pensar em aplicar o algoritmo LSQR para resolver o sistema $S\overline{f} = \overline{d}$, porém, isto é algo muito mais caro pois são necessários, por iteração, 4 produtos matriz-vetor envolvendo a matriz $L^{\dagger}$. Logo, esta a abordagem não será considerada em nenhum momento.

A inicialização de ambos os algoritmos também tem um custo. Considerando apenas o caso 1, notemos que ambos os algoritmos realizam produtos envolvendo a matriz Ω e isto requer o produto de $(AW)^{\dagger}$. Se W tem poucas colunas isto é extremamente barato de ser calculado, assim, podemos calcular e armazenar a matriz $\Omega_1 = (AW)^{\dagger}A$. Se W tem apenas uma coluna, o que significa que $\dim(\mathcal{N}(L)) = 1$, então Ω_1, que é apenas um vetor linha, é obtido através de dois produtos matriz-vetor envolvendo a matriz A. O vetor $\overline{g} = g - AW(AW)^{\dagger}g$, pode ser encontrado, considerando novamente $\dim(\mathcal{N}(L)) = 1$, de modo muito eficiente com apenas mais um produto interno (pois o vetor AW pode ser previamente armazenado no cálculo de Ω_1). Para o algoritmo SN-GMRES, precisamos encontrar o vetor $\overline{d} = {L_A^{\dagger}}^{T}Pg$ e isto requer resolver um sistema linear envolvendo a matriz $L^{\dagger}$ e um produto matriz-vetor com as matrizes Ω e P. Sob as mesmas condições de $\dim(\mathcal{N}(L)) = 1$, produtos com a matriz $P = I_n - AW(W^{T}AW)^{-1}W^{T}$ tem um custo pequeno, pois $W^{T}AW$ é um escalar e AW pode ser previamente armazenado.

3.6 Exemplos numéricos - parte 2

Finalizamos este capítulo com alguns experimentos usando os algoritmos P-LSQR e SN-X no problema (3.65) com o critério de parada (3.29). Para isto, vamos considerar o problema deriv2 e a matriz L como uma versão discretizada do operador diferencial de primeira ordem. Na Tabela 3.10 usamos as abreviações SN-R para SN-RRGMRES, P-L para P-LSQR e TGS para TGSVD e apresentamos os resultados obtidos após 20 execuções dos algoritmos. As

informações são as mesmas das tabelas anteriores, ou seja, os parâmetros mínimos(máximos) determinados pelo critério (3.29), os parâmetros mínimos(máximos) ótimos e os respectivos erros médios. Além disso, reportamos os resultados obtidos usando a GSVD do par (A, L). Todos os experimentos foram realizados em um PC com processador Core 2 Duo 2,53GHz, 4 GB de memória RAM, sistema operacional Linux e Matlab versão 7. Como estamos mais interessados na qualidade (número de iterações, erro relativo, etc.) das soluções obtidas, não reportamos os tempos gastos.

	NL = 0,001			NL = 0,01			NL = 0,025		
	SN-R	P-L	TGS	SN-R	P-L	TGS	SN-R	P-L	TGS
$\widehat{k}_m(\widehat{k}_M)$	9(11)	5(5)	5(6)	5(6)	2(2)	2(2)	4(5)	1(1)	1(1)
$k^*_m(k^*_M)$	7(10)	7(9)	9(15)	3(5)	3(5)	3(7)	3(5)	3(4)	3(5)
$E_{\widehat{k}}$	0,0122	0,0162	0,0176	0,0291	0,0543	0,0609	0,0405	0,0689	0,0701
E_{k^*}	0,0087	0,0091	0,0092	0,0210	0,0219	0,0221	0,0299	0,0306	0,0307
GSVD	FP	ótimo		FP	ótimo		FP	ótimo	
λ	0,0782	0,0122		0,9431	0,0794		2,8533	0,1681	
E_λ	0,0175	0,0094		0,0570	0,0233		0,0968	0,0314	

Tabela 3.10: Parâmetros mínimos(máximos) encontrados por (3.29) e parâmetros mínimos(máximos) ótimos para SN-RRGMRES, P-LSQR e TGSVD e respectivos erros médios. Média dos parâmetro de regularização determinados por FP e ótimo e respectivos erros médios para o problema deriv2 com uma versão discretizada do operador diferencial de primeira ordem.

Analisando os dados, concluimos que usando um regularizador apropriado é possível obter soluções próximas da solução f_{exato} (compare com os resultados da Tabela 3.3) com um número pequeno de iterações. Uma observação importante a respeito deste experimento é com relação à NL = 0,025. Para $\mu = 1$ não existe ponto fixo convexo não nulo. Assim, de modo a obtermos um ponto fixo convexo, ajustamos $\mu = 0,8$.

Para finalizarmos, vamos considerar novamente o problema de restauração de imagem anterior mas com a matriz de regularização sendo uma versão discretizada do operador diferencial de primeira ordem bidimensional. Na Tabela 3.11 apresentamos os resultados obtidos para os algoritmos P-LSQR e SN-RRGMRES (abreviado para SN-R). Além disso, apresentamos os resultados obtidos aplicando o algoritmo GKB-FP como discutido no Capítulo 2. Neste exemplo em particular, o algoritmo SN-RRGMRES apresentou resultados um pouco melhores do que os obtidos por P-LSQR. No que diz respeito ao número de iterações, o

critério de parada proposto foi bastante eficaz, exceto para o caso com NL = 0,05 em que P-LSQR parou muito antes do número ótimo de iterações, e como consequência, a solução f_k não capturou informação relevante do problema conforme Figura 3.16.

	NL = 0,001		NL = 0,01		NL = 0,025		NL = 0,05	
	P-LSQR	SN-R	P-LSQR	SN-R	P-LSQR	SN-R	P-LSQR	SN-R
$\widehat{k}$	285(290)	182(183)	67(68)	85(86)	33(33)	59(60)	17(17)	45(46)
$k_{\text{ótimo}}$	479(507)	208(216)	160(173)	96(100)	98(106)	69(73)	67(73)	55(58)
$E_{\widehat{k}}$	0,1518	0,1498	0,1633	0,1576	0,1731	0,1627	0,1875	0,1684
$E_{k_{\text{ótimo}}}$	0,1495	0,1493	0,1572	0,1571	0,1618	0,1617	0,1669	0,1667
	GKB-FP		GKB-FP		GKB-FP		GKB-FP	
k	345(353)		124(128)		73(74)		47(48)	
λ	0,0133		0,1724		0,5299		1,5347	
E_λ	0,1516		0,1628		0,1729		0,1901	

Tabela 3.11: Parâmetros mínimos(máximos) encontrados por (3.29) e parâmetros mínimos(máximos) ótimos para P-LSQR e SN-RRGMRES e respectivos erros médios. Média dos parâmetros de regularização determinados pela versão estendida de GKB-FP e respectivos erros médios para o problema de restauração de imagem com regularizador como uma versão discretizada do operador diferencial de primeira ordem bidimensional.

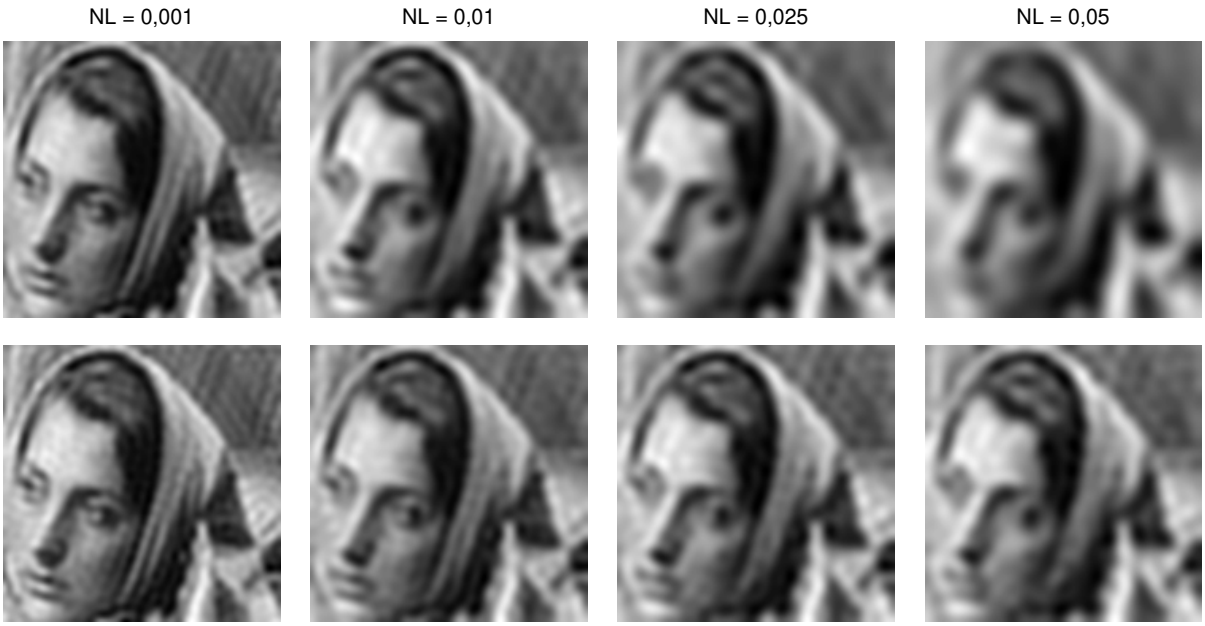


Figura 3.16: Solução obtida pelo algoritmo P-LSQR (linha superior) e SN-RRGMRES (linha inferior) usando o critério de parada (3.29) e NL = 0,001, 0,01, 0,025, 0,05.

Capítulo 4

Iterações de ponto fixo para o método de regularização de Tikhonov com múltiplos parâmetros

O método de regularização de Tikhonov tem sido utilizado com sucesso em diversas áreas das ciências e engenharias desde seu surgimento com os trabalhos de Riley [122], Phillips [115], Twomey [133], Tikhonov [131], Golub [48] e com a teoria do método de regularização de Tikhonov para equações integrais de Fredholm de primeira espécie por Groetsch [52]. Contudo, tal método quase sempre está restrito a um único parâmetro de regularização. O uso de vários parâmetros de regularização tem ganhado interesse nos últimos anos nas mais distintas áreas. Para o caso com um parâmetro existe quantidade significativa de métodos para a sua seleção, porém, para o caso com múltiplos parâmetros, há uma carência de métodos para selecioná-los. Neste capítulo revisamos o método original de Regińska e apresentamos resultados teóricos que suportam a generalização para o caso com múltiplos parâmetros.

4.1 Método de regularização de Tikhonov com múltiplos parâmetros

Nos últimos anos tem crescido o interesse no método de regularização de Tikhonov com múltiplos parâmetros e aplicações desta técnica tem aparecido nas mais diversas áreas como na determinação de geopotenciais a partir de órbitas precisas de satélites [135], reconstrução de imagens de alta resolução com erros de deslocamento [98], super-resolução de imagens [142] e estimação de parâmetros em processos de difusão de salto [41]. Neste método, a solução (1.4) é substituída pela solução regularizada f_λ dada por

$$f_\lambda = \underset{f\in\mathbb{R}^n}{\operatorname{argmin}} \left\{ \|g - Af\|_2^2 + \sum_{i=1}^{q} \lambda_i^2 \|L_i f\|_2^2 \right\}, \tag{4.1}$$

em que $L_i \in \mathbb{R}^{p_i \times n}$, $i = 1, \ldots, q$, são as matrizes de regularização e $\lambda = [\lambda_1, \ldots, \lambda_q]^T$, $\lambda_i > 0$, é o vetor de parâmetros de regularização. Resolver o problema (4.1) é equivalente a encontrar a solução das equações normais regularizadas

$$\left(A^T A + \sum_{i=1}^{q} \lambda_i^2 L_i^T L_i \right) f = A^T g, \tag{4.2}$$

cuja unicidade de solução é garantida com a condição

$$\mathcal{N}(A) \cap \mathcal{N}(L_1) \cap \cdots \cap \mathcal{N}(L_q) = 0. \tag{4.3}$$

Esta condição é satisfeita quando, por exemplo, $L_i = I_n$ para algum i ou quando A tem posto coluna completo. Para calcularmos de modo eficiente a solução do problema de minimização (4.1), devemos considerá-lo como um problema de mínimos quadrados

$$f_\lambda = \operatorname{argmin} \|\overline{g}_0 - \overline{A}_\lambda\|_2^2, \tag{4.4}$$

em que

$$\overline{A}_\lambda = \begin{bmatrix} A \\ \lambda_1 L_1 \\ \vdots \\ \lambda_q L_q \end{bmatrix}, \quad \overline{g}_0 = \begin{bmatrix} g \\ \mathbf{0} \end{bmatrix}, \tag{4.5}$$

e $\mathbf{0}$ é o vetor nulo de dimensão apropriada. Como não conhecemos uma decomposição do tipo GSVD para uma $(q+1)$-upla de matrizes $(A, L_1, \ldots, L_q)$, ou seja, até onde sabemos, não existe uma decomposição do tipo

$$A = \mathsf{U}\Sigma_A \mathsf{X}^{-1}, \quad L_i = \mathsf{V}_i \Sigma_{L_i} \mathsf{X}^{-1}, \tag{4.6}$$

com as matrizes U e V_i ortogonais, X não-singular e Σ_A e Σ_{L_i} diagonais, este problema deve ser resolvido através da decomposição QR ou da SVD da matriz $\overline{A}_\lambda$. Caso estas decomposições não sejam viáveis, métodos iterativos como CGLS e LSQR podem ser utilizados.

Com apenas um parâmetro de regularização, o erro $E(\lambda) = \|f_\lambda - f_{\text{exato}}\|_2 / \|f_{\text{exato}}\|_2$ é uma curva em $\mathbb{R}^2$. Para o caso com dois parâmetros, o erro é uma superfície em $\mathbb{R}^3$. Na Figura 4.1 podemos apreciar a superfície do erro relativo na solução regularizada para o caso com dois parâmetros (esquerda), e temos as curvas do erro com apenas um parâmetro considerando $L_1 = I_n$ e L_2 uma versão discretizada do operador diferencial de primeira ordem (direita).

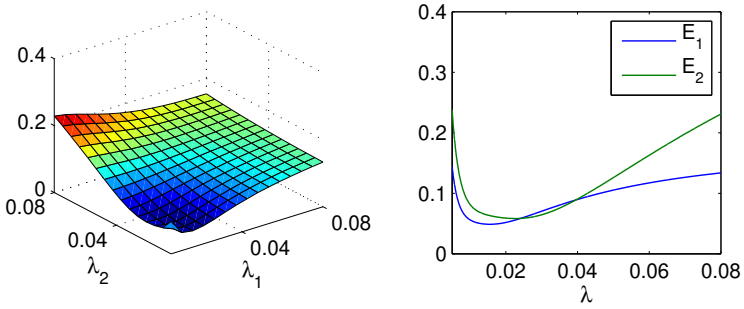


Figura 4.1: Superfície $E(\lambda_1, \lambda_2) = \|f_{\text{exato}} - f_\lambda\|_2 / \|f_{\text{exato}}\|_2$ (esquerda) e curvas $E_1(\lambda_1, 0) = \|f_{\text{exato}} - f_\lambda\|_2 / \|f_{\text{exato}}\|_2$ e $E_2(0, \lambda_2) = \|f_{\text{exato}} - f_\lambda\|_2 / \|f_{\text{exato}}\|_2$ (direita) para o problema shaw com $n = 32$, NL = 0,005 e regularizadores identidade e uma versão discretizada do operador diferencial de primeira ordem.

A conclusão interessante ao compararmos o gráfico da esquerda com o gráfico da direita é que, enquanto que no caso com um parâmetro (seja com L_1 ou com L_2) a intersecção das

curvas $E_1(\lambda_1, 0)$ e $E_2(0, \lambda_2)$ com a reta $z = \nu$ para, por exemplo, $\nu = 0, 1$, tem dois pontos, para o caso com múltiplos parâmetros esta intersecção, da superfície $E(\lambda_1, \lambda_2)$ com o plano $z = (\lambda_1, \lambda_2, \nu)$, é uma curva de parâmetros que fornecem soluções com a mesma qualidade. Sendo assim, a seleção de um par de parâmetros (λ_1, λ_2) torna-se uma tarefa muito mais difícil quando comparado com o caso com um parâmetro.

4.2 Métodos para a escolha dos parâmetros de regularização

Diferentemente do caso com um parâmetro, não temos disponível muitas ferramentas para escolher os parâmetros $\lambda = [\lambda_1, \ldots, \lambda_q]$ e recentemente tem aparecido algumas propostas interessantes. Usaremos ao longo deste capítulo as funções

$$x(\lambda) = \|g - Af_\lambda\|_2^2, \quad y_i(\lambda) = \|L_i f_\lambda\|_2^2, \quad i = 1, \ldots, q. \tag{4.7}$$

4.2.1 Princípio da Discrepância

De modo análogo ao caso com um parâmetro, a ideia é encontrar um conjunto de parâmetros $\lambda = [\lambda_1, \ldots, \lambda_q]$ tal que a solução f_λ satisfaça o princípio da discrepância (DP)

$$\|g - Af_\lambda\|_2 = c\|\epsilon\|_2 \equiv \delta, \quad c \geq 1. \tag{4.8}$$

A interpretação geométrica da equação (4.8) para o caso com múltiplos parâmetros tem o mesmo princípio do caso com um parâmetro, ou seja, devemos encontrar a intersecção da superfície $x(\lambda) \subset \mathbb{R}^{q+1}$ com o plano $z = (\lambda_1^2, \ldots, \lambda_q^2, \delta^2) \subset \mathbb{R}^{q+1}$. Na Figura 4.2 temos estas superfícies para o caso com dois parâmetros. A esquerda temos a superfície $x(\lambda)$ e o plano z, e a direita a curva da intersecção entre estas duas superfícies projetada no plano $z_0 = (\lambda_1, \lambda_2, 0)$.

A principal dificuldade com este método é que podem existir, como mostrado na Figura 4.2, infinitos pares de parâmetros (λ_1, λ_2) que satisfazem (4.8) e, escolher um ponto sobre esta curva que produza bons resultados, é tão difícil quanto encontrar um ponto sobre ela.

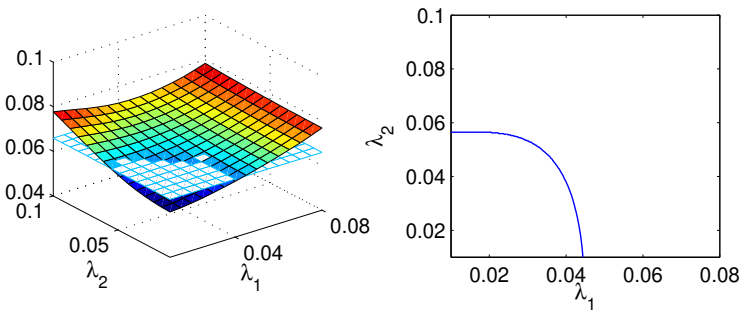


Figura 4.2: Superfície $x(\lambda) = \|g - Af_\lambda\|_2^2$ e plano $z = \delta^2$ (esquerda) e projeção da curva da intersecção entre a superfície e o plano $z = \delta^2$ sobre o plano $z = 0$ (direita) para o problema shaw com $n = 32$, NL = 0,005 e regularizadores identidade e uma versão discretizada do operador diferencial de primeira ordem.

Lu e Pereverzev [96] propuseram aproximar a função discrepância $\|g - Af_\lambda\|_2^2$ por uma função mais simples (chamada de função modelo) e mostraram como estimar os parâmetros de regularização a partir da equação aproximada obtida. A função modelo para o caso com dois parâmetros e com $L_1 = I_n$ e $L_2 \neq I_n$, é

$$m(\lambda_1, \lambda_2) = \|g\|_2^2 + \frac{C}{\lambda_2^2} + \frac{D}{T + \lambda_1^2}, \tag{4.9}$$

em que C, D e T são constantes relativamente simples de serem calculadas e dependem dos parâmetros de regularização. Ressaltamos que os quadrados dos parâmetros λ_1^2, λ_2^2 fazem o papel de β e α, respectivamente, em [96]. Um algoritmo que implementa o princípio da discrepância de acordo com o proposto por Lu e Pereverzev em [96] é:

1. dados $\delta, g, A, (\lambda_1^{(0)})^2, (\lambda_2^{(0)})^2, 0 < \gamma < 1$, definimos $k = 0$;

2. encontramos f_λ e calculamos os coeficientes C, D e T como descrito na subseção 3.1 em [96]. Atualizamos $\lambda_2^{(k+1)}$ e definimos $(\lambda_1^{(k+1)})^2 = \gamma(\lambda_1^{(k)})^2$;

3. paramos se $\|g - Af_\lambda\| < \delta$ ou se $(\lambda_2^{(k+1)})^2$ for muito pequeno, por exemplo $(\lambda_2^{(k+1)})^2 < 2,2 \times 10^{-16}$. Caso contrário definimos $k = k + 1$ e voltamos para 2.

Como já comentamos, a dificuldade com o uso do princípio da discrepância está no fato de que em muitos problemas o nível de ruído não está disponível. Neste caso, a solução calculada pode não ter utilidade prática caso uma estimativa ruim seja fornecida. Além

disso, como $(\lambda_1^{(k)})^2$ sempre diminui com k, pode existir um inteiro k tal que $(\lambda_1^{(k)})^2 \approx 0$, e isto pode resultar em soluções regularizadas altamente influenciadas de um modo negativo pelo parâmetro estimado por $(\lambda_2^{(k)})^2$. Voltaremos com este ponto no Capítulo 5.

Mais recentemente, Wang [140] propôs novas funções modelos para aproximar a equação da discrepância. Como as ideias não diferem muito do que Lu e Pereverzev descrevem, não vamos explorar tais abordagens, especialmente pelo fato de que neste trabalho consideramos que estimativas para a norma do ruído não estão disponíveis.

4.2.2 Combinação Linear Restrita

Brezinski et al. [21] propuseram aproximar a solução f_λ pela combinação linear das soluções dos problemas com apenas um parâmetro, ou seja, aproximar f_λ por $f_\lambda(\eta)$ dada por

$$f_\lambda(\eta) = \sum_{i=1}^{q} \eta_i f_{\lambda_i}, \tag{4.10}$$

em que

$$f_{\lambda_i} = \underset{f\in\mathbb{R}^n}{\operatorname{argmin}} \left\{ \|g - Af\|_2^2 + q\lambda_i^2 \|Lf\|_2^2 \right\}, \quad i = 1, \dots q, \tag{4.11}$$

e λ_i é determinado por qualquer método disponível na literatura, em particular FP. A utilização de $q\lambda_i^2$ em vez λ_i^2 na equação (4.11) é apenas por questões numéricas [21].

Podemos perceber que na equação (4.10) temos total liberdade na escolha dos parâmetros η_i. Contudo, Brezinski et al. [21] sugerem escolher o vetor $\eta = (\eta_1, \dots, \eta_q)^T$ como solução de

$$\eta = \left\{ \underset{\eta\in\mathbb{R}^q}{\operatorname{argmin}} \|\rho_\lambda(\eta)\|_2, \quad \text{sujeito a } \sum_{i=1}^{q} \eta_i = 1 \right\}, \tag{4.12}$$

em que

$$\rho_\lambda(\eta) = \sum_{i=1}^{q} \eta_i \left[q\lambda_i^2 L_i^T L_i - \sum_{j=1}^{q} \lambda_j^2 L_i^T L_i \right] f_{\lambda_i}. \tag{4.13}$$

A ideia desta abordagem é que se o vetor de parâmetros λ está disponível e η é escolhido de acordo com (4.12), então a solução (4.10) é uma boa aproximação para a solução f_λ. No entanto, isto não necessariamente garante que $f_\lambda(\eta)$ seja uma boa aproximação para f_{exato}.

Vamos considerar dois parâmetros de regularização e analisar o comportamento do erro

na solução $f_\lambda(\eta)$ com relação à escolha dos parâmetros η_1 e $\eta_2 = 1 - \eta_1$ e com os parâmetros λ_1 e λ_2 escolhidos por diferentes métodos. O comportamento das curvas da Figura 4.3 nos leva a concluir que, uma escolha ruim do par (η_1, η_2) pode produzir resultados insatisfatórios mesmo que os parâmetros λ_1 e λ_2 sejam escolhidos pelo princípio da discrepância.

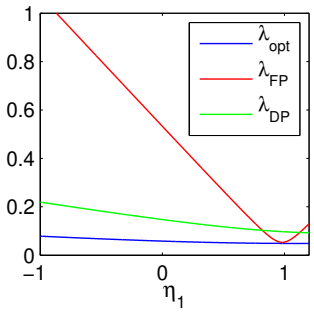


Figura 4.3: Erro $E_\lambda = \|f_{\text{exato}} - f_\lambda(\eta)\|_2 / \|f_{\text{exato}}\|_2$ com $\lambda_{\text{ótimo}}$, λ_{FP} e λ_{DP} para o problema shaw com $n = 32$, NL = 0,005 e regularizadores identidade e uma versão discretizada do operador diferencial de primeira ordem.

Se considerarmos, por exemplo, $\lambda_1 = 0,0110$, $\lambda_2 = 0,0158$, $\eta_1 = 1,0789$ e $\eta_2 = -0,0789$, a solução $f_\lambda(\eta)$, conforme equações (4.11) - (4.13), apresenta erro relativo igual a 0,0487.

4.2.3 Generalização da Curva-L e GCV

O método da curva-L foi generalizado em [14] para a hipersurperfície-L e o "canto" foi definido como o ponto que maximiza a curvatura Gaussiana da superfície

$$\mathcal{L}(\lambda) = \{(a, b_1, \ldots, b_q) \ / \ a = \log(x(\lambda)), b_i = \log(y_i(\lambda)), i = 1, \ldots, q\}\,. \tag{4.14}$$

Contudo, determinar tal ponto é computacionalmente caro. Belge et al. [15] propuseram substituir a curvatura Gaussiana por uma função, chamada de função distância, que é mais simples de ser otimizada. Infelizmente, a eficiência do algoritmo gerado depende fortemente de uma origem que é especificada pelo usuário como mostram os resultados em [9].

O método da função GCV para múltiplos parâmetros consiste em escolher o vetor λ como

o minimizador da função

$$V(\lambda) = \frac{\|g - Af_\lambda\|_2^2}{(\text{traço}(I_n - B(\lambda)))^2}, \quad \text{com} \quad B(\lambda) = A\left(A^TA + \sum_{i=1}^{q} \lambda_i^2 L_i^T L_i\right)^{-1} A^T. \tag{4.15}$$

No entanto, minimizar esta função é uma tarefa computacional extraordinária pois, para determinarmos a matriz $B(\lambda)$, precisamos inverter uma matriz. Brezinski et al. [21] propuseram uma modificação da função GCV que é mais simples de ser otimizada e concluiu que sua abordagem corresponde a escolher os parâmetros λ_i aplicando o método GCV separadamente a cada subproblema (4.11) e calculando soluções regularizadas de acordo com a aproximação (4.10). Na Figura 4.4 temos a superfície GCV (4.15) para o problema shaw, notemos que existe uma região quase plana que dificulta a localização do minimizador.

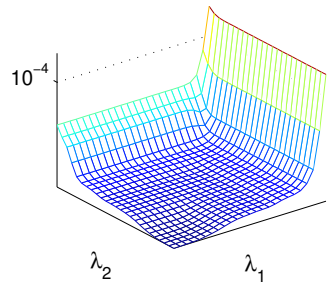


Figura 4.4: Superfície $V(\lambda)$ em escala log-log-log para o problema shaw com $n = 32$, NL = 0,005 e regularizadores identidade e uma versão discretizada do operador diferencial de primeira ordem.

4.3 Generalização do método de Regińska e o método de ponto fixo

Conforme vimos no Capítulo 2, Regińska [118] propôs escolher como parâmetro de regularização de Tikhonov um minimizador da função

$$\Psi(\lambda) = x(\lambda)y(\lambda)^\mu = \|g - Af_\lambda\|_2^2 \|Lf_\lambda\|_2^{2\mu}, \quad \mu > 0. \tag{4.16}$$

Bazán [9] mostrou que os minimizadores de $\Psi(\lambda)$ são pontos fixos da função

$$\phi(\lambda;\mu) = \sqrt{\mu}\frac{\|g - Af_\lambda\|_2}{\|Lf_\lambda\|}, \quad \mu > 0, \tag{4.17}$$

dando origem ao algoritmo FP [9, 13]. Na Figura 4.5 temos um típico exemplo das funções $\Psi(\lambda)$ e $\phi(\lambda;1)$.

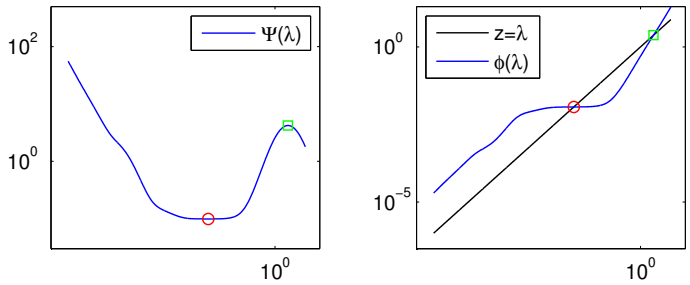


Figura 4.5: Funções $\Psi(\lambda)$ e $\phi(\lambda;1)$ para o problema i_laplace com $n = 256$ e NL = 0,005. O círculo (quadrado) denota o minimizador (maximizador) local e o ponto fixo associado.

Sugerimos escolher como parâmetro de regularização $\lambda = [\lambda_1, \ldots, \lambda_q]^T$ do método de regularização de Tikhonov com múltiplos parâmetros um minimizador local de

$$\Psi(\lambda) = x(\lambda)y_1(\lambda)^{\mu_1} \cdots y_q(\lambda)^{\mu_q}, \quad \mu_i > 0, \tag{4.18}$$

em que f_λ é solução do problema (4.1) e as funções $x(\lambda)$ e $y_i(\lambda)$ são dadas por (4.7).

Vamos agora discutir condições para que um ponto $\lambda = [\lambda_1, \ldots, \lambda_q]^T$ seja um minimizador local da função $\Psi(\lambda)$. Das condições de otimalidade [99] temos o seguinte resultado.

Lema 4.1. *O gradiente de* $\Psi(\lambda)$ *pode ser escrito como*

$$\nabla\Psi(\lambda) = y_1(\lambda)^{\mu_1} \cdots y_q(\lambda)^{\mu_q}(J\Omega + \nabla x(\lambda)), \tag{4.19}$$

em que

$$J = J(\lambda) = \begin{bmatrix} \dfrac{\partial y_1}{\partial \lambda_1}(\lambda) & \cdots & \dfrac{\partial y_q}{\partial \lambda_1}(\lambda) \\ \vdots & \ddots & \vdots \\ \dfrac{\partial y_1}{\partial \lambda_q}(\lambda) & \cdots & \dfrac{\partial y_q}{\partial \lambda_q}(\lambda) \end{bmatrix} \quad e \quad \Omega = \Omega(\lambda) = \begin{pmatrix} \mu_1 \dfrac{x(\lambda)}{y_1(\lambda)} \\ \vdots \\ \mu_q \dfrac{x(\lambda)}{y_q(\lambda)} \end{pmatrix}. \tag{4.20}$$

Demonstração. Do cálculo, segue de imediato que

$$\begin{aligned} \nabla\Psi &= y_1^{\mu_1} \cdots y_q^{\mu_q} \left(\begin{bmatrix} \dfrac{\partial y_1}{\partial \lambda_1} & \cdots & \dfrac{\partial y_q}{\partial \lambda_1} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial y_1}{\partial \lambda_q} & \cdots & \dfrac{\partial y_q}{\partial \lambda_q} \end{bmatrix} \begin{pmatrix} \mu_1 \dfrac{x}{y_1} \\ \vdots \\ \mu_q \dfrac{x}{y_q} \end{pmatrix} + \begin{pmatrix} \dfrac{\partial x}{\partial \lambda_1} \\ \vdots \\ \dfrac{\partial x}{\partial \lambda_q} \end{pmatrix} \right) \\ &= y_1^{\mu_1} \cdots y_q^{\mu_q} \left(J\Omega + \nabla x \right). \end{aligned} \tag{4.21}$$

□

Outro resultado imediato concerne às derivadas das funções $x(\lambda)$ e $y_i(\lambda)$.

Lema 4.2. *As derivadas parciais de $x(\lambda)$ e $y_i(\lambda)$ com relação a λ_j são*

$$\frac{\partial x}{\partial \lambda_j} = 2 \left(\frac{\partial f_\lambda}{\partial \lambda_j} \right)^T A^T (A f_\lambda - g), \quad \frac{\partial y_i}{\partial \lambda_j} = 2 \left(\frac{\partial f_\lambda}{\partial \lambda_j} \right)^T L_i^T L_i f_\lambda. \tag{4.22}$$

Demonstração. Seja $x(\lambda) = \|g - A f_\lambda\|_2^2 = g^T g - 2 g^T A f_\lambda + f_\lambda^T A^T A f_\lambda$. Então, derivando com relação a λ_j, temos

$$\frac{\partial x}{\partial \lambda_j} = 2 \left(\frac{\partial f_\lambda}{\partial \lambda_j} \right)^T A^T (A f_\lambda - g). \tag{4.23}$$

Agora, para $y_i(\lambda)$, $i = 1, \ldots, q$, segue que $y_i(\lambda) = \|L_i f_\lambda\|_2^2 = f_\lambda^T L_i^T L_i f_\lambda$ Então, derivando com relação a λ_j, temos

$$\frac{\partial y_i}{\partial \lambda_j} = 2 \left(\frac{\partial f_\lambda}{\partial \lambda_j} \right)^T L_i^T L_i f_\lambda. \tag{4.24}$$

□

O lema a seguir relaciona as derivadas parciais das funções $x(\lambda)$ e $y_i(\lambda)$ e nos fornece

condições para garantirmos que os pontos extremos de $\Psi(\lambda)$ são pontos fixos de uma função vetorial que veremos adiante.

Lema 4.3. *Sob a condição (4.3), as seguintes propriedades valem*

i)

$$\frac{\partial x}{\partial \lambda_j} = -\lambda_1^2 \frac{\partial y_1}{\partial \lambda_j} - \cdots - \lambda_q^2 \frac{\partial y_q}{\partial \lambda_j}. \tag{4.25}$$

ii) os vetores $\dfrac{\partial f_\lambda}{\partial \lambda_j}$, $j = 1, \ldots, q$ *são linearmente independentes se, e somente se, os vetores* $L_j^T L_j f_\lambda$, $j = 1, \ldots, q$ *são linearmente independentes.*

iii) J *é não-singular desde que* $\dfrac{\partial f_\lambda}{\partial \lambda_j}$, $j = 1, \ldots, q$ *sejam linearmente independentes.*

Demonstração. A hipótese (4.3) implica que o problema (4.1) tem única solução f_λ tal que

$$(A^T A + \lambda_1^2 L_1^T L_1 + \cdots + \lambda_q^2 L_q^T L_q) f_\lambda = A^T g. \tag{4.26}$$

Isto implica que $A^T(Af_\lambda - g) = -\lambda_1^2 L_1^T L_1 f_\lambda - \cdots - \lambda_q^2 L_q^T L_q f_\lambda$. Agora, basta multiplicarmos esta equação por $\left(\dfrac{\partial f_\lambda}{\partial \lambda_j}\right)^T$, $j = 1, \ldots, q$, e usarmos o Lema 4.2 para provar o primeiro item. Por outro lado, diferenciando a equação (4.26) com respeito a λ_j nos leva a

$$2\lambda_j L_j^T L_j f_\lambda + (A^T A + \lambda_1^2 L_1^T L_1 + \cdots + \lambda_q^2 L_q^T L_q) \left(\frac{\partial f_\lambda}{\partial \lambda_j}\right) = 0. \tag{4.27}$$

Seja $B = A^T A + \lambda_1^2 L_1^T L_1 + \cdots + \lambda_q^2 L_q^T L_q$. Então, podemos reescrever a equação (4.27) como

$$B \left[\frac{\partial f_\lambda}{\partial \lambda_1} \;\; \frac{\partial f_\lambda}{\partial \lambda_2} \cdots \frac{\partial f_\lambda}{\partial \lambda_q}\right] = -\left[2\lambda_1 L_1^T L_1 f_\lambda \;\; 2\lambda_2 L_2^T L_2 f_\lambda \cdots 2\lambda_q L_q^T L_q f_\lambda\right], \tag{4.28}$$

e isto prova o segundo item pois pela condição (4.3) B é positiva definida. Por outro lado, uma consequência imediata de (4.27) é

$$\left(\frac{\partial f_\lambda}{\partial \lambda_i}\right)^T (A^T A + \lambda_1^2 L_1^T L_1 + \cdots + \lambda_q^2 L_q^T L_q) \left(\frac{\partial f_\lambda}{\partial \lambda_j}\right) = -2\lambda_j \left(\frac{\partial f_\lambda}{\partial \lambda_i}\right)^T L_j^T L_j f_\lambda. \tag{4.29}$$

Usando o Lema 4.2 temos

$$\left(\frac{\partial f_\lambda}{\partial \lambda_i}\right)^T (A^T A + \lambda_1^2 L_1^T L_1 + \cdots + \lambda_q^2 L_q^T L_q) \left(\frac{\partial f_\lambda}{\partial \lambda_j}\right) = -\lambda_j \frac{\partial y_j}{\partial \lambda_i}, \tag{4.30}$$

ou seja, $F^T(A^T A + \lambda_1^2 L_1^T L_1 + \cdots + \lambda_q^2 L_q^T L_q)F = -J\,\mathrm{diag}(\lambda_1, \ldots, \lambda_q)$, com a matriz $F = \left[\frac{\partial f_\lambda}{\partial \lambda_1} \;\; \frac{\partial f_\lambda}{\partial \lambda_2} \cdots \frac{\partial f_\lambda}{\partial \lambda_q}\right]$ e o terceiro item está demonstrado pois F tem posto coluna completo.

□

No caso em que usamos apenas um parâmetro de regularização, a seminorma $\|Lf_\lambda\|_2$ é uma função decrescente em λ. Para o caso com múltiplos parâmetros podemos usar (4.30) para provar o seguinte resultado similar.

Corolário 4.4. *A derivada parcial de $y_i(\lambda)$ com respeito a λ_i é sempre negativa.*

Demonstração. Da equação (4.30) temos que se $\lambda_j > 0$ então $\dfrac{\partial y_j}{\partial \lambda_j} < 0$.

□

Voltando para a expressão do gradiente de $\Psi(\lambda)$, podemos então usar a equação (4.25) em (4.19) e concluir que o gradiente de $\Psi(\lambda)$ pode ser escrito como

$$\nabla\Psi(\lambda) = y_1^{\mu_1} \cdots y_q^{\mu_q} J \left(\Omega - \begin{pmatrix} \lambda_1^2 \\ \vdots \\ \lambda_q^2 \end{pmatrix} \right). \tag{4.31}$$

A expressão acima mostra que a condição necessária para $\Psi(\lambda)$ ter um extremo local em $\lambda^* = [\lambda_1^*, \ldots, \lambda_q^*]^T \neq 0$, é que $J(\lambda^*)$ seja não-singular e

$$\Omega(\lambda^*) = \begin{pmatrix} \lambda_1^{*2} \\ \vdots \\ \lambda_q^{*2} \end{pmatrix} \Leftrightarrow \lambda_i^{*2} = \mu_i \frac{x(\lambda^*)}{y_i(\lambda^*)}, \qquad i = 1, \ldots, q. \tag{4.32}$$

Portanto, se $\Psi(\lambda)$ atingir um máximo/mínimo local em λ^*, este λ^* deve ser um ponto

fixo da função vetorial $\Phi(\lambda) : \mathbb{R}^q \to \mathbb{R}^q$ definida por

$$\Phi(\lambda) = \begin{pmatrix} \phi_1(\lambda, \mu_1) \\ \vdots \\ \phi_q(\lambda, \mu_q) \end{pmatrix}, \quad \phi_i(\lambda; \mu_i) = \sqrt{\mu_i} \frac{\|g - Af_\lambda\|_2}{\|L_i f_\lambda\|_2}, \qquad i = 1, \ldots, q. \tag{4.33}$$

A questão sobre existência de pontos fixos será comentada no final desta seção. Agora, temos o seguinte teorema que fornece condições para minimizar $\Psi(\lambda)$.

Teorema 4.5. *Uma condição suficiente para que o ponto fixo λ^* de $\Phi(\lambda)$ seja um minimizador local de $\Psi(\lambda)$ é que a matriz*

$$2J(\lambda^*)diag(\lambda^*) + J(\lambda^*)H(\lambda^*)J(\lambda^*)^T, \tag{4.34}$$

seja negativa definida, em que

$$H(\lambda) = \begin{bmatrix} \frac{\lambda_1^2}{y_1}(1+\mu_1) & \lambda_2^2\frac{\mu_1}{y_1} & \cdots & \lambda_q^2\frac{\mu_1}{y_1} \\ \lambda_1^2\frac{\mu_2}{y_2} & \frac{\lambda_2^2}{y_2}(1+\mu_2) & \cdots & \lambda_q^2\frac{\mu_2}{y_2} \\ \vdots & \ddots & \cdots & \vdots \\ \lambda_1^2\frac{\mu_q}{y_q} & \lambda_2^2\frac{\mu_q}{y_q} & \cdots & \frac{\lambda_q^2}{y_q}(1+\mu_q) \end{bmatrix}. \tag{4.35}$$

Demonstração. Se λ^* é um ponto fixo de $\Phi(\lambda)$, então o gradiente e a Hessiana de $\Psi(\lambda)$ no ponto λ^* satisfazem

$$\nabla\Psi(\lambda^*) = y_1^{\mu_1}(\lambda^*)\cdots y_q^{\mu_q}(\lambda^*)J(\lambda^*) \begin{pmatrix} -\lambda_1^{*^2} + \phi_1(\lambda^*)^2 \\ \vdots \\ -\lambda_q^{*^2} + \phi_q(\lambda^*)^2 \end{pmatrix} = 0, \tag{4.36}$$

$$\nabla^2\Psi(\lambda^*) = -2y_1^{\mu_1}(\lambda^*)\cdots y_q^{\mu_q}(\lambda^*)J(\lambda^*)\mathrm{diag}(\lambda_1^*, \ldots, \lambda_q^*)(I_q - J_\phi(\lambda^*)), \tag{4.37}$$

em que $J_\phi(\lambda)$ denota a matriz Jacobiana da função $\Phi(\lambda)$.

Usando a equação (4.25) em λ^* e o fato de que $y_i(\lambda^*)\lambda_i^{*2} = \mu_i x(\lambda^*)$, podemos escrever a

derivada parcial de $\phi_i(\lambda)$ com respeito a λ_j como

$$\frac{\partial \phi_i}{\partial \lambda_j} = \frac{1}{2y_i}\left(-\lambda_1^2\frac{\mu_i}{\lambda_i}\frac{\partial y_1}{\partial \lambda_j} - \cdots - \lambda_i(1+\mu_i)\frac{\partial y_i}{\partial \lambda_j} - \cdots - \lambda_q^2\frac{\mu_i}{\lambda_i}\frac{\partial y_q}{\partial \lambda_j}\right). \tag{4.38}$$

Portanto, a matriz $J_\phi(\lambda^*)$ é

$$J_\phi(\lambda^*) = -\frac{1}{2}\begin{bmatrix} \frac{\lambda_1}{y_1}(1+\mu_1) & \lambda_2^2\frac{\mu_1}{y_1\lambda_1} & \cdots & \lambda_q^2\frac{\mu_1}{y_1\lambda_1} \\ \lambda_1^2\frac{\mu_2}{y_2\lambda_2} & \frac{\lambda_2}{y_2}(1+\mu_2) & \cdots & \lambda_q^2\frac{\mu_2}{y_2\lambda_2} \\ \vdots & \ddots & \cdots & \vdots \\ \lambda_1^2\frac{\mu_q}{y_q\lambda_q} & \lambda_2^2\frac{\mu_q}{y_q\lambda_q} & \cdots & \frac{\lambda_q}{y_q}(1+\mu_q) \end{bmatrix}\begin{bmatrix} \frac{\partial y_1}{\partial \lambda_1} & \cdots & \frac{\partial y_1}{\partial \lambda_q} \\ \frac{\partial y_2}{\partial \lambda_1} & \cdots & \frac{\partial y_2}{\partial \lambda_q} \\ \vdots & \cdots & \vdots \\ \frac{\partial y_q}{\partial \lambda_1} & \cdots & \frac{\partial y_q}{\partial \lambda_q} \end{bmatrix}, \tag{4.39}$$

e a expressão para a Hessiana de $\Psi(\lambda)$ em λ^* é

$$\nabla^2\Psi(\lambda^*) = -y_1^{\mu_1}(\lambda^*)\cdots y_q^{\mu_q}(\lambda^*)\left[2J(\lambda^*)\mathrm{diag}(\lambda_1^*,\ldots,\lambda_q^*) + J(\lambda^*)H(\lambda^*)J(\lambda^*)^T\right]. \tag{4.40}$$

Por hipótese, a matriz $\left[2J(\lambda^*)\mathrm{diag}(\lambda^*) + J(\lambda^*)H(\lambda^*)J(\lambda^*)^T\right]$ é negativa definida, logo a Hessiana $\nabla^2\Psi(\lambda^*)$ é positiva definida. Das condições de otimalidade de segunda ordem [99, 107] segue que λ^* é um minimizador local de $\Psi(\lambda)$.

□

Sabemos que para o caso com um parâmetro e para λ maior do que o maior valor singular generalizado do par matricial (A, L) vale $\phi(\lambda; 1) > \lambda$ (veja [9, Lema 1]). O seguinte resultado similar vale para o caso com múltiplos parâmetros.

Teorema 4.6. *Assuma que uma matriz de regularização, digamos L_1, tem posto coluna completo. Seja $\overline{\gamma}_1$ o maior valor singular generalizado do par (A, L_1). Assuma também que $f_\lambda \notin \mathcal{N}(L_i)$, $i = 2,\ldots,q$. Então para todo $\lambda = (\lambda_1,\ldots,\lambda_q)$ com $\lambda_1 > \overline{\gamma}_1$ e $\lambda_i > 0$ vale*

$$\phi_i(\lambda; 1) > \lambda_i, \quad i = 1,\ldots,q. \tag{4.41}$$

Demonstração. Vamos considerar o método de regularização de Tikhonov com um parâmetro

$$\overline{f}_{\xi} = \underset{\overline{f}\in\mathbb{R}^n}{\operatorname{argmin}} \left\{ \|g - A\overline{f}\|_2^2 + \xi^2 \|L_\lambda \overline{f}\|_2^2 \right\}, \quad \xi > 0, \tag{4.42}$$

em que

$$L_\lambda = \begin{bmatrix} \lambda_1 L_1 \\ \vdots \\ \lambda_q L_q \end{bmatrix}, \quad \lambda_i > 0, i = 1, \ldots, q. \tag{4.43}$$

Disto segue que $\overline{f}_{\xi}$ é única e $\overline{f}_1 = f_\lambda$. Seja $\varphi(\xi) = \|g - A\overline{f}_{\xi}\|_2 / \|L_\lambda \overline{f}_{\xi}\|_2$, $\xi > 0$. Então para todo $\lambda_i > 0$ temos

$$\frac{\phi_i(\lambda;1)^2}{\lambda_i^2} = \frac{\|g - Af_\lambda\|_2^2}{\lambda_i^2 \|L_i f_\lambda\|_2^2} \geq \frac{\|g - Af_\lambda\|_2^2}{\lambda_1^2 \|L_1 f_\lambda\|_2^2 + \cdots + \lambda_q^2 \|L_q f_\lambda\|_2^2} = \varphi^2(1), \tag{4.44}$$

então basta provarmos que $\varphi(1) > 1$. Seja $\overline{\gamma}_\lambda$ e $\overline{x}$ o maior autovalor generalizado e o autovetor correspondente do par matricial (A, L_λ), respectivamente. Sabemos que $\varphi(\xi) > \xi$ sempre que $\xi > \overline{\gamma}_\lambda$ (veja, [9, Lema 1]), logo precisamos provar que $\overline{\gamma}_\lambda < 1$. Além disso, o maior valor singular generalizado do par (A, L_1) pode ser caracterizado por

$$\overline{\gamma}_1^2 = \max_{x\neq 0} \frac{x^T A^T A x}{x^T L_1^T L_1 x}. \tag{4.45}$$

Então, para todo $x \neq 0$

$$\begin{aligned} \frac{\overline{\gamma}_1^2}{\lambda_1^2} = \frac{\overline{x}^T A^T A \overline{x}}{\overline{x}^T (\lambda_1 L_1)^T (\lambda_1 L_1) \overline{x}} &\geq \frac{x^T A^T A x}{x^T (\lambda_1 L_1)^T (\lambda_1 L_1) x} \\ &\geq \frac{x^T A^T A x}{x^T (\lambda_1 L_1)^T (\lambda_1 L_1) x + \cdots + x^T (\lambda_q L_q)^T (\lambda_q L_1) x}. \end{aligned} \tag{4.46}$$

Mas isso implica que

$$\frac{\overline{\gamma}_1^2}{\lambda_1^2} \geq \max_{x\neq 0} \frac{x^T A^T A x}{x^T (\lambda_1 L_1)^T (\lambda_1 L_1) x + \cdots + x^T (\lambda_q L_q)^T (\lambda_q L_1) x} = {\overline{\gamma}_\lambda}^2. \tag{4.47}$$

Portanto, dado $\lambda_1 > \overline{\gamma}_1$, o maior valor singular generalizado do par matricial (A, L_λ) satisfaz $\overline{\gamma}_\lambda < 1$, e o teorma está provado.

□

Na prática, muitos problemas utilizam matrizes de regularização de posto incompleto, mas o uso de regularizadores de posto completo aparece em diversas áreas (veja, por exemplo, [121]). Uma consequência prática do Teorema 4.6 é que se todas as matrizes de regularização forem de posto completo, então existe uma caixa

$$B = [0, \overline{\gamma}_1] \times [0, \overline{\gamma}_2] \times \cdots [0, \overline{\gamma}_q], \tag{4.48}$$

em que $\overline{\gamma}_i$ denota o maior valor singular generalizado do par matricial (A, L_i), tal que

$$\phi_i(\lambda, 1) > \lambda_i \quad \text{com} \quad \lambda_i > \overline{\gamma}_i, \lambda_j > 0, j \neq i, i = 1, \ldots, q, \tag{4.49}$$

e este resultado generaliza o Lema 1 em [9].

Enquanto o Teorema 4.6 nos informa que os pontos fixos de $\Phi(\lambda)$ não podem estar fora da região em $\mathbb{R}^q$ que incluam pontos λ com $\lambda_1 > \overline{\gamma}_1$, a equação (4.49) nos permite concluir que os pontos fixos de $\Phi(\lambda)$ não podem estar fora da caixa B. Estas conclusões são importantes pois nos ajudam a detectar divergência do algoritmo baseado em iterações de ponto fixo que apresentaremos adiante.

Para o caso com um parâmetro, a existência de ponto fixo foi discutida em [9] e melhor compreendida em [10]. Vamos agora fornecer uma breve discussão a respeito da existência de pontos fixos para $\Phi(\lambda)$. Existência de pontos fixos de $\Phi(\lambda)$ para o caso com dois parâmetros tem uma simples e elegante interpretação geométrica. Para isto, vamos considerar os planos Π_i e as superfícies $\mathcal{T}_i$ em $\mathbb{R}^3$ definidas, respectivamente, por:

$$\Pi_i = \{(\lambda_1, \lambda_2, z)/z = \lambda_i\}, \quad \mathcal{T}_i = \{(\lambda_1, \lambda_2, z)/z = \phi_i(\lambda, 1)\}, \quad i = 1, 2. \tag{4.50}$$

Seja $C_i = \Pi_i \cap \mathcal{T}_i$. Então, baseado em (4.32), $\Phi(\lambda)$ terá pontos fixos desde que $C_1 \cap C_2 \neq \varnothing$, neste caso, C_i são curvas em $\mathbb{R}^3$ que se intersectam para fornecer os pontos fixos de $\Phi(\lambda)$. Na Figura 4.6 temos uma superfície $z = \Psi(\lambda)$ e as respectivas projeções C_i sobre o plano $z = (\lambda_1, \lambda_2, 0)$.

Na Figura 4.7 temos um exemplo das superfícies $\mathcal{T}_i$ e os planos Π_i. As curvas em azul são

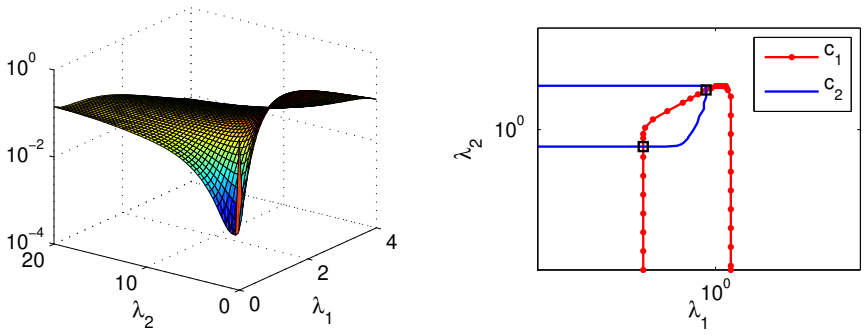


Figura 4.6: Superfície $z = \Psi(\lambda)$ (esquerda) e as projeções das curvas C_i sobre o plano $z = (\lambda_1, \lambda_2, 0)$ (direita) para o problema i_laplace com $n = 256$, NL = 0,01 e regularizadores identidade e uma versão discretizada do operador diferencial de primeira ordem. Notemos que a intersecção de $C_1 \cap C_2$ fornece dois pontos fixos de $\Phi(\lambda)$.

precisamente a intersecção entre as superfícies $\mathcal{T}_i$ com os planos Π_i.

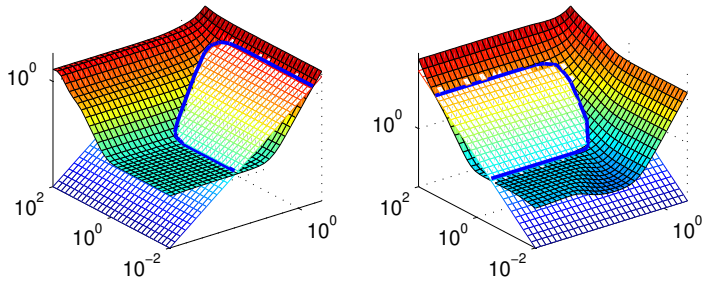


Figura 4.7: Superfícies $\mathcal{T}_i$ e planos Π_i para $i = 1$ (esquerda) e $i = 2$ (direita).

Nosso algoritmo para a seleção dos múltiplos parâmetros, denotado por MFP (*Multi-parameter Fixed-Point*), segue exatamente os mesmos passos do caso com um parâmetro. Podemos descrevê-lo da seguinte maneira:

- fornecemos uma aproximação inicial $\lambda^{(0)} = [\lambda_1^{(0)}, \ldots, \lambda_q^{(0)}]^T$ e definimos $\mu_i = 1$. Até algum critério de parada ser satisfeito, calculamos a sequência

$$\lambda_i^{(k+1)} = \phi_i(\lambda^{(k)}; 1), \quad i = 1, \ldots, q \ \text{ e } \ k \geq 0; \tag{4.51}$$

- se $\lambda_i^{(k)}$ divergir para algum i, o parâmetro μ_i é ajustado e as iterações reiniciam. Ajustes nos parâmetros μ_i são realizados de modo similar ao caso com um parâmetro (veja [9] para os detalhes).

A escolha da aproximação inicial é um ponto crucial para métodos iterativos e, provavelmente, existem diversas alternativas para a aproximação inicial de MFP. Descrevemos duas delas:

a) podemos escolher como aproximação inicial um conjunto pequeno de parâmetros, digamos $\lambda_i = 10^{-4}$, e então prosseguir conforme descrito acima;

b) podemos aplicar o algoritmo de ponto fixo para cada um dos q subproblemas considerando apenas um parâmetro, um para cada L_i, e usar os parâmetros encontrados como aproximação inicial para MFP.

A principal vantagem da opção b) é que o algoritmo FP para o caso com um parâmetro nos fornece parâmetros $\mu_i \neq 1$ quando ajustes são necessários. Nossos experimentos numéricos no capítulo seguinte são realizados utilizando esta opção.

Como critério de parada optamos por interromper as iterações quando a distância relativa entre duas iteradas consecutivas é pequena, isto é, quando

$$\|\lambda^{(k+1)} - \lambda^{(k)}\|_2 < \varepsilon_1 \|\lambda^{(k)}\|_2, \tag{4.52}$$

em que $\varepsilon_1 > 0$ é um pequeno parâmetro de tolerância ou, para prevenir baixa taxa de convergência, quando

$$\|\lambda^{(k+1)} - \lambda^{(k)}\|_2 < \varepsilon_2 \|\lambda^{(1)}\|_2, \tag{4.53}$$

em que $\varepsilon_2 > 0$ é outro pequeno parâmetro de tolerância.

Terminamos esta seção com uma breve discussão sobre as propriedades de convergência das iteradas (4.51), nos concentrando, em particular, no caso com dois parâmetros. O caso geral pode ser tratado de modo similar. Pelo fato da sequência (4.51) ser gerada por iterações de ponto fixo, o melhor que conseguimos garantir é convergência local das iteradas. A ideia central da análise de convergência está na dinâmica das iteradas para o caso com múltiplos parâmetros apresentar, essencialmente, a mesma dinâmica do caso com um parâmetro. Vamos considerar apenas o caso com um parâmetro e que a norma do resíduo não se anule para $\lambda = 0$. Além disso, vamos considerar que a função $\phi(\lambda; 1)$ tenha um único ponto fixo que minimiza $\Psi(\lambda)$, denotado por $\overline{\lambda}$, e um único ponto fixo que maximiza $\Psi(\lambda)$, denotado por $\widetilde{\lambda}$. Tal situação é ilustrada na Figura 4.8 para o problema i_laplace.

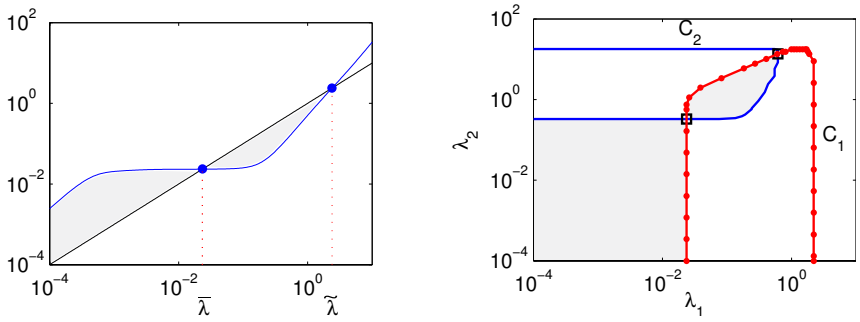


Figura 4.8: Função $\phi(\lambda; 1)$ e região de convergência das iteradas (2.13) (área sombreada) para o problema i_laplace com $n = 256$, NL = de 0,01 e $L = I_n$ (esquerda). Região de convergência das iteradas (4.51) (área sombreada) para o caso com dois parâmetros (direita), o segundo regularizador é uma versão discretizada do operador diferencial de primeira ordem. As curvas C_1 e C_2 são as mesmas da Figura 4.6.

Assim, baseado na propriedade da função $\phi(\lambda; 1)$ ser crescente, a análise em [9] nos permite concluir que:

- a região de convergência para as iteradas $\lambda^{(k)}$ é o intervalo aberto $]0, \widetilde{\lambda}[$;
- $\lambda^{(k)}$ converge para $\overline{\lambda}$ desde que $\lambda^{(0)}$ pertença à região de convergência. Neste caso, $\lambda^{(k)}$ será uma sequência decrescente se $\lambda^{(0)} \in]\overline{\lambda}, \widetilde{\lambda}[$ (neste caso $\phi(\lambda^{(0)}; 1) < \lambda^{(0)}$) ou será uma sequência crescente se $\lambda^{(0)} \in]0, \overline{\lambda}[$ (neste caso $\phi(\lambda^{(0)}; 1) > \lambda^{(0)}$).

No caso com dois parâmetros, seja $\mathcal{R}_1$ a região entre as curvas C_1 e C_2 e seja $\mathcal{R}_2$ a região limitada pelas curvas C_1 e C_2 e pelas retas $\lambda_1 = 0$ e $\lambda_2 = 0$ (veja a região sombreada na Figura 4.8). Então, a sequência $\lambda_i^{(k)}$, $i = 1, 2$, convergirá para $\overline{\lambda}_i$ desde que a aproximação inicial $\lambda_i^{(0)}$, $i = 1, 2$ esteja na região $\mathcal{R}_1 \cup \mathcal{R}_2$, o que significa que $\phi_i(\lambda_1^{(0)}, \lambda_2^{(0)}) < \lambda_i^{(0)}$, $i = 1, 2$ ou que $\phi_i(\lambda_1^{(0)}, \lambda_2^{(0)}) > \lambda_i^{(0)}$, $i = 1, 2$. Em qualquer um dos casos, similar ao caso com um parâmetro, a convergência é assegurada pois $\phi_i(\lambda_1, \lambda_2)$ é uma função crescente de λ_i se o outro parâmetro é mantido fixo como consequência do Lema 4.3 e do Corolário 4.4. Uma análise mais rigorosa para a convergência do método MFP será realizada em um trabalho futuro.

4.4 Extensão de GKB-FP para o método de regularização de Tikhonov com múltiplos parâmetros para problemas de grande porte

Vimos que o algoritmo MFP requer que o problema de minimização (4.5) seja resolvido repetidamente para diversos vetores de parâmetros $\lambda = [\lambda_1, \ldots, \lambda_q]$, para que sejam calculadas as seminormas $\|L_i f_\lambda\|_2$ e a norma do resíduo associado $\|g - Af_\lambda\|_2$, e que tal problema pode ser resolvido eficientemente via decomposição QR/SVD da matriz $\overline{A}_\lambda$. No entanto, se $\overline{A}_\lambda$ for muito grande, a decomposição QR/SVD desta matriz é computacionalmente inviável. Nosso propósito agora é estender o algoritmo GKB-FP para o método de regularização de Tikhonov com múltiplos parâmetros para problemas de grande porte. A ideia principal é produzir uma sequência $f_\lambda^{(k)}$ de soluções aproximadas obtidas resolvendo o problema (4.1) restrito ao subespaço $\mathcal{K}_k(A^TA, A^Tg)$. Portanto, a solução aproximada $f_\lambda^{(k)}$ é determinada como

$$f_\lambda^{(k)} = \underset{f \in \mathcal{K}_k(A^TA, A^Tg)}{\operatorname{argmin}} \left\{ \|g - Af\|_2^2 + \sum_{i=1}^{q} \lambda_i^2 \|L_i f\|_2^2 \right\}, \tag{4.54}$$

que através do processo GKB aplicado à matriz A pode ser calculada por

$$f_\lambda^{(k)} = V_k d_\lambda^{(k)}, \quad d_\lambda^{(k)} = \underset{d \in \mathbb{R}^k}{\operatorname{argmin}} \left\{ \|\beta_1 e_1 - B_k d\|_2^2 + \sum_{i=1}^{q} \lambda_i^2 \|L_i V_k d\|_2^2 \right\}. \tag{4.55}$$

Quando $L_1 = I_n$ e $L_i = 0, i \neq 1$, o método apresentado se reduz ao algoritmo GKB-FP.

Seguindo as mesmas ideias do algoritmo PROJ-L descrito no Capítulo 2, podemos calcular a decomposição QR de $L_i V_k$, e assim, $\|L_i V_k d\|_2$ em (4.55) pode ser substituido por $\|R_i^{(k)} d\|_2$. Isto torna o problema de minimização mais simples, neste caso, (4.55) reduz a

$$f_\lambda^{(k)} = V_k d_\lambda^{(k)}, \quad d_\lambda^{(k)} = \underset{d \in \mathbb{R}^k}{\operatorname{argmin}} \|\overline{g}_0 - B_\lambda^{(k)} d\|_2, \tag{4.56}$$

em que

$$B_\lambda^{(k)} = \begin{bmatrix} B_k \\ \lambda_1 R_1^{(k)} \\ \vdots \\ \lambda_q R_q^{(k)} \end{bmatrix} \quad \text{e} \quad \overline{g}_0 = \begin{bmatrix} \beta_1 e_1 \\ 0 \end{bmatrix}, \tag{4.57}$$

então, $d_\lambda^{(k)}$ pode ser eficientemente calculado de diversas maneiras, por exemplo, por métodos diretos ou por primeiro transformar a matriz $B_\lambda^{(k)}$ em uma matriz triangular superior, como feito na implementação de GKB-FP [10]. Além disso, a solução aproximada $f_\lambda^{(k)}$ e o resíduo correspondente $r_\lambda^{(k)} = g - Af_\lambda^{(k)}$, satisfazem

$$\|L_i f_\lambda^{(k)}\|_2 = \|R_k d_\lambda^{(k)}\|_2, \quad \|r_\lambda^{(k)}\|_2 = \|\beta_1 e_1 - B_k d_\lambda^{(k)}\|. \tag{4.58}$$

Para cada $k \geq 1$, vamos considerar a função

$$\Phi^{(k)}(\lambda) = \left[\phi_1^{(k)}(\lambda, \mu_1), \ldots, \phi_q^{(k)}(\lambda, \mu_q)\right]^T, \quad \phi_i^{(k)}(\lambda, \mu_i) = \sqrt{\mu_i}\frac{\|\beta_1 e_1 - B_k d_\lambda^{(k)}\|_2}{\|R_i^{(k)} d_\lambda^{(k)}\|_2}. \tag{4.59}$$

Nossa proposta para o método de regularização de Tikhonov com múltiplos parâmetros para problemas de grande porte segue os mesmos passos de GKB-FP. Isto é, dado $p_0 > 1$ e $k \geq p_0$, nosso algoritmo de projeção calcula pontos fixos $\lambda^{(k)^*}$ de $\Phi^{(k)}(\lambda)$ que minimizam

$$\Psi^{(k)}(\lambda) = \|r_\lambda^{(k)}\|_2^2 \|L_1 f_\lambda^{(k)}\|_2^{2\mu_1} \cdots \|L_q f_\lambda^{(k)}\|_2^{2\mu_q}, \tag{4.60}$$

e o processo é repetido até algum critério de parada ser satisfeito. Para tornar nossa proposta computacionalmente viável, os seguintes aspectos devem ser considerados:

1. A aproximação inicial no passo $k+1$ é escolhida como o ponto fixo do passo k.

2. A decomposição QR é feita uma única vez, isto é, no passo $k = p_0$ temos $L_i V_{p_0} = Q_i^{(p_0)} R_i^{(p_0)}$ e nos passos subsequêntes esta é atualizada, ou seja, a decomposição QR de $L_i V_{k+1}$ é atualizada a partir da decomposição QR de $L_i V_k$.

O custo para este algoritmo é um pouco maior do que o necessário pelo PROJ-L, isto é, além das multiplicações matriz-vetor Au e $A^T v$ e de uma atualização QR, temos a atualização

de mais $q-1$ decomposições QR. e, para cada vetor $\lambda = [\lambda_1, \ldots, \lambda_q]$, o problema (4.56) pode ser resolvido pela decomposição QR da matriz $B_\lambda^{(k)}$ via rotações de Givens, e isto, é da ordem de $\mathcal{O}(k)$ operações.

Capítulo 5

Experimentos numéricos

Para ilustrar os algoritmos propostos neste trabalho, vamos aplicá-los em alguns problemas disponíveis na literatura. A primeira parte contém os resultados referentes aos Capítulos 2 e 3 que tratam, respectivamente, do método de regularização de Tikhonov na forma geral e o uso de métodos iterativos. A segunda parte contém os resultados referentes ao Capítulo 4, no qual estudamos o método de regularização de Tikhonov com múltiplos parâmetros e introduzimos o algoritmo MFP. O vetor de dados dos problemas são da forma $g = g_{\text{exato}} + \epsilon$, em que ϵ é gerado pela função `randn` do Matlab. Todos os experimentos foram realizados em um PC com processador Core 2 Duo 2,53GHz, 4 GB de memória RAM, sistema operacional Linux e Matlab versão 7.

5.1 Método de regularização de Tikhonov na forma geral e regularização iterativa

Nesta primeira parte ilustramos os métodos propostos nos Capítulos 2 e 3 e, para isso, dois exemplos de *image deblurring* e um de super-resolução de imagem são usados.

Para deixarmos nossa notação simples, a extensão do algoritmo GKB-FP baseada na fatoração LU (página 33) da matriz de regularização L será denotada por FP-LU e chamaremos de FP-L_D a extensão baseada na abordagem que utiliza a matriz L_D (2.68). Valores médios para o parâmetro de regularização determinado, erro relativo na solução $f_\lambda^{(k)}$ e tempo gasto

durante o processo, ou seja, para a determinação do parâmetro de regularização e cálculo da solução regularizada $f_\lambda^{(k)}$, após vinte diferentes vetores de dados g, são denotados por $\overline{\lambda}$, $\overline{E}$ e $\overline{t}$, respectivamente, enquanto que o número mínimo/máximo de iterações necessárias para atingir o critério de parada são denotados por k_m/k_M, respectivamente.

5.1.1 Problema 1: *deblurring* - imagem rice

O objetivo aqui é recuperar uma imagem armazenada na forma vetorial $f_{\text{exato}} \in \mathbb{R}^{MN}$, a partir de uma imagem embaçada e contaminada com ruído armazenada na forma vetorial $g = g_{\text{exato}} + \epsilon \in \mathbb{R}^{MN}$, tal que $Af_{\text{exato}} = g_{\text{exato}}$, em que a matriz A é responsável pelo processo de *blurring* (embaçamento). A matriz A às vezes é conhecida como a matriz PSF (*Point Spread Function* - Função de Espalhamento de Ponto). O detalhamento deste processo foge ao escopo desta tese, no entanto, sugerimos [68, 72] para mais informações. Por simplicidade, vamos considerar imagens $N \times N$ e usar uma matriz PSF, $N^2 \times N^2$, definida por $A = (2\pi\sigma^2)^{-1} T \otimes T$. Nesta definição, σ controla a largura da função de espalhamento de ponto Gaussiana e a matriz T é uma matriz $N \times N$, simétrica, Toeplitz e com largura de banda igual a metade do parâmetro *band* (veja [84]). Usando os mesmos valores de [84], temos $\sigma = 2$ e $band = 16$ e, como matriz de regularização, usamos

$$L_{1,2D} = \begin{bmatrix} \mathcal{L}_{1,N} \otimes I_N \\ I_N \otimes \mathcal{L}_{1,N} \end{bmatrix}, \quad \mathcal{L}_{1,N} = \begin{bmatrix} 1 & -1 & & & \\ & 1 & -1 & & \\ & & \ddots & \ddots & \\ & & & 1 & -1 \end{bmatrix} \in \mathbb{R}^{(N-1)\times N}, \tag{5.1}$$

em que $\mathcal{L}_{1,N}$ é uma versão discretizada do operador diferencial de primeira ordem.

Rice 64

Temos dois propósitos principais nesta subseção. O primeiro é ilustrar numericamente que os métodos propostos neste trabalho são mais eficientes do que o algoritmo da bidiagonalização simultânea (JBD) proposto por Kilmer et al., descrito em [84] e descrito de modo sucinto no apêndice D. O segundo é entender melhor as capacidades do algoritmo *pruning*

para a identificação do "canto" da curva-L discreta.

No algoritmo baseado na bidiagonalização simultânea optamos por usar o método FP para a seleção do parâmetro de regularização de Tikhonov e denotamos o algoritmo resultante por JBD-FP. O algoritmo JBD-FP procede de modo análogo ao PROJ-L no sentido de que, para $p_0 > 1$ e $k \geq p_0$, determinamos o maior ponto fixo convexo de $\phi^{(k)}(\lambda;\mu)$ e o processo é repetido até algum critério de parada ser satisfeito. Um ingrediente crucial do algoritmo JBD é o cálculo de dois produtos do tipo matriz-vetor da forma $\widehat{v} = QQ^T v$ por iteração. Então, a eficiência de JBD-FP está diretamente ligada com o modo como estes dois produtos são realizados. Notemos que o cálculo de $\widehat{v}$ pode ser interpretado como encontrar a projeção ortogonal de v sobre o espaço coluna da matriz $\widehat{A} = [A^T \ L^T]^T$ e isto é equivalente a

$$\widehat{v} = \widehat{A}u_{\mathrm{LS}}, \quad \text{com} \quad u_{\mathrm{LS}} = \underset{u\in\mathbb{R}^n}{\operatorname{argmin}} \|v - \widehat{A}u\|_2, \tag{5.2}$$

o que justifica a razão pela qual JBD-FP deve ser computacionalmente caro pois o problema (5.2) deve ser resolvido duas vezes por iteração. Pela equação (5.2), $\widehat{v}$ pode ser calculado de diversas maneiras e duas destas são usadas aqui. A primeira calcula projeções através do algoritmo LSQR (veja [128]), e a outra é baseada no algoritmo LSQR com subespaço precondicionador, como em [22, 78]. Optamos em exibir os resultados obtidos pela primeira opção, pois esta mostrou-se mais eficiente. Duas implementações distintas para JBD-FP são consideradas: a primeira, denotada por JBD-FP$_L$, lida com o problema (2.1) usando a matriz L e a segunda, denotada por JBD-FP$_D$, lida com o problema (2.71).

Vamos considerar a subimagem de tamanho 64×64 pixels usada em [84] (a imagem inteira é de tamanho 256×256 pixels) para ilustrar os algoritmos baseados no processo JBD. Então, com $N = 64$, a matriz $A \in \mathbb{R}^{4.096\times4.096}$ e a matriz de regularização $L \in \mathbb{R}^{8.064\times4.096}$. Assim, o número de condição de A é $\kappa(A) \approx 2,14 \times 10^{16}$ e o posto numérico é de 3.946. Dada uma tolerância $\hat{\delta} > 0$, o posto numérico de uma matriz A é $\hat{k}$ se $\sigma_{\hat{k}} > \hat{\delta} \geq \sigma_{\hat{k}+1}$. Neste trabalho, o posto numérico é determinado pelo comando `rank` do Matlab, usando seu valor padrão de tolerância.

Além disso, usamos dados perturbados com NL = 0,01. Em todos os casos, o processo GKB e o processo JBD são implementados com reortogonalização completa. O cálculo de

pontos fixos inicia com $p_0 = 10$ e as iterações finalizam usando $\varepsilon_1 = \varepsilon_2 = 10^{-6}$ em (4.52) e (4.53). Por motivos de comparação, o parâmetro de regularização de Tikhonov determinado pelos métodos LC e FP, o parâmetro ótimo e os erros relativos são apresentados na Tabela 5.1. Os parâmetros encontrados pela curva-L e pelo ponto fixo foram determinados usando a decomposição GSVD do par matricial (A, R), em que $R \in \mathbb{R}^{4.095 \times 4.096}$ foi obtida através da decomposição QR da matriz de regularização L. Na mesma tabela também exibimos o parâmetro k encontrado pelo método P-LSQR, o parâmetro ótimo e o erros relativos.

Para o algoritmo P-LSQR, usamos $\overline{A}$ e $\overline{g}$ explicitamente obtidos pela transformação para a forma padrão através da rotina std_form, disponível em [64], e o parâmetro ótimo para P-LSQR é definido de modo análogo ao parâmetro de regularização ótimo para o método de regularização de Tikhonov. Para este exemplo, a curva-L tem um "canto" bem definido e a função $\phi(\lambda; 1)$ tem um único ponto fixo que minimiza $\Psi(\lambda)$ (veja Figura 5.2). Na Figura 5.1 temos a imagem exata, a imagem embaçada e as imagens reconstruídas com a GSVD do par (A, R) e os parâmetros encontrados pela curva-L (LC) e pelo método FP.

	LC	FP	ótimo	P-LSQR	ótimo
λ	0,0783	0,0956	0,0409	$k^* = 48$	$k = 83$
E	0,0803	0,0819	0,0775	0,0831	0,0782

Tabela 5.1: Parâmetros de regularização e erros relativos na solução para LC, FP e P-LSQR.

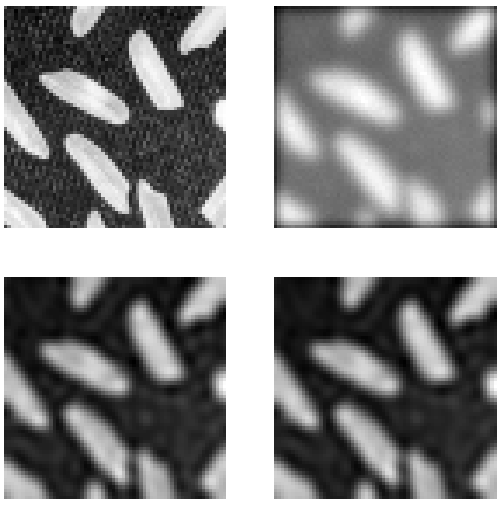


Figura 5.1: Linha superior: imagem exata (esquerda) e embaçada (direita). Linha inferior: imagem reconstruída pela curva-L (esquerda) e pelo ponto fixo (direita).

Na Tabela 5.2 temos o tempo médio gasto em 20 execuções (cada uma com um novo

vetor de ruído ϵ) dos algoritmos propostos e das duas versões baseadas no processo JBD. A conclusão é que o processo JBD é consideravelmente mais lento do que os demais algoritmos. Contudo, isto já era esperado devido ao problema (5.2) ser resolvido duas vezes por iteração.

Com relação à qualidade dos resultados obtidos, todas as soluções calculadas tiveram erro aproximado de 0,082 com relação a solução exata e isto está de acordo com a qualidade obtida utilizando a GSVD do par matricial (A, R) (veja Tabela 5.1).

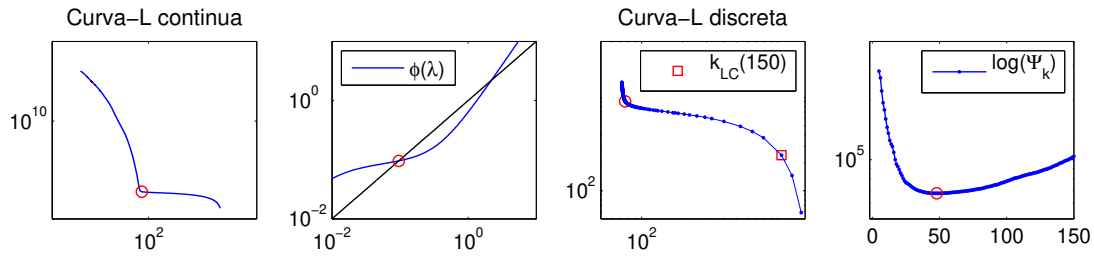


Figura 5.2: Curva-L, função $\phi(\lambda; 1)$ em escala log-log e Ψ_k para o problema rice64. Os círculos mostram o canto da curva-L, o minimizador de $\Psi(\lambda)$ (ponto fixo de $\phi(\lambda)$) e o minimizador de Ψ_k, respectivamente.

	JBD-FP$_L$	JBD-FP$_D$	FP-LU	FP-L$_D$	PROJ-L	P-LSQR
$\bar{t}$	5,2206	4,2688	2,0084	0,8863	0,2576	0,2809

Tabela 5.2: Tempo médio (em segundos) por execução.

Com relação às capacidades do algoritmo *pruning* para localizar o "canto" da curva-L discreta, chegamos na mesma conclusão do Capítulo 3. O índice k_{LC} pode variar com o número de pontos q da curva-L discreta. Na Tabela 5.3 exibimos o parâmetro retornado pela rotina corner e o erro relativo na solução iterada correspondente para diversos valores de q. O valor inicial para q foi escolhido relativamente próximo do minimizador de Ψ_k, $k = 48$, para avaliarmos como o "canto" calculado pelo algoritmo *pruning* se comporta. Um "canto" falso, correspondendo a q = 150, é exibido na Figura 5.2. Por estas conclusões o *pruning* não será usado nos demais exemplos.

Rice 256

Agora vamos considerar a imagem inteira rice de 256×256 pixels. Assim, a matriz PSF $A \in \mathbb{R}^{65.536 \times 65.536}$ com número de condição $\kappa(A) \approx 3,40 \times 10^{16}$ e a matriz de regularização

q	60	80	100	120	140	160	180	200
$k_{\mathrm{LC}}(\mathsf{q})$	3	3	43	43	3	3	46	43
Erro	0,2065	0,2065	0,0846	0,0846	0,2065	0,2065	0,0838	0,0846

Tabela 5.3: "Canto", $k_{\mathrm{LC}}(\mathsf{q})$, localizado pelo algoritmo *pruning* como função do número de pontos da curva-L discreta $(\log \|\overline{g} - \overline{Af}_k\|_2, \log \|\overline{f}_k\|_2)$ em que $\overline{f}_k$ é obtido por LSQR.

$L \in \mathbb{R}^{130.560 \times 65.536}$. Além disso, usamos $p_0 = 15$. Para este exemplo, apenas FP-L$_D$, P-LSQR e PROJ-L são usados e as médias obtidas após 20 execuções com NL = 0,01 são exibidos na Tabela 5.4. Foram realizados experimentos sem e com reortogonalização completa.

	FP-L$_D$	P-LSQR	PROJ-L	FP-L$_D$	P-LSQR	PROJ-L
$\overline{\lambda}$	0,0706	-	0,0874	0,0746	-	0,0874
$\overline{E}$	0,0820	0,0851	0,0831	0,0812	0,0845	0,0831
$\overline{t}$	7,9603	4,3365	1,0652	54,6983	15,9995	1,8808
$k_m(k_M)$	433 (512)	242 (268)	28 (30)	280 (290)	140 (141)	28 (30)
$\overline{\mu}$	0,6519	-	1	0,6635	-	1
		reortog = 0			reortog = 1	

Tabela 5.4: Resultados para a imagem inteira rice para NL = 0,01, $p_0 = 15$ e $\varepsilon_1 = \varepsilon_2 = 10^{-6}$.

Como podemos observar, o algoritmo FP-L$_D$ ajustou o parâmetro μ. O algoritmo PROJ-L, especialmente sem reortogonalização, foi o que precisou de menos tempo para atingir o critério de parada. Todos os algoritmos apresentaram soluções com qualidade semelhante e, comparando o desempenho entre sem e com reortogonalização, houve, para FP-L$_D$ e P-LSQR, redução entre 35,3% e 47,3% no número de iterações, mas isso tem um custo, ou seja, o tempo computacional aumentou, aproximadamente 6,8 vezes para FP-L$_D$ e 3,6 vezes para P-LSQR.

Teoricamente, o processo GKB gera uma base ortonormal para o subespaço de Krylov $\mathcal{K}_k(A^TA, A^Tg)$. Porém, devido às imprecisões numéricas e às dificuldades inerentes aos problemas mal-postos discretos, há uma perda de ortogonalidade dos vetores v_i e, também, dos vetores u_i, à medida que as iterações avançam. Logo, a reortogonalização destes vetores é necessária para que a informação calculada seja condizente com o que a teoria prevê. Contudo, conforme vimos na Tabela 5.4, em problemas práticos, há um aumento no tempo computacional gasto (apesar do número de iterações reduzir).

5.1.2 Problema 2: *deblurring* - imagem pirate

Neste exemplo, vamos considerar uma imagem grande de 512×512 pixels chamada de pirate (veja Figura 5.3). Assim, $N = 512$ e, portanto, a matriz $A \in \mathbb{R}^{262.144 \times 262.144}$ com número de condição $\kappa(A) \approx 1,57 \times 10^{19}$ e a matriz de regularização $L \in \mathbb{R}^{523.264 \times 262.144}$. Neste experimento usamos $p_0 = 20$. De modo análogo ao problema anterior, apresentamos valores médios obtidos em 20 execuções para os algoritmos FP-L_D, P-LSQR e PROJ-L com e sem reortogonalização completa. Os resultados numéricos para NL = 0,01 são exibidos na Tabela 5.5. Neste caso, o erro relativo nas soluções calculadas é de aproximadamente 0,14 e, mais uma vez, o algoritmo mais rápido foi PROJ-L. A imagem original, a imagem embaçada e a reconstrução que obtivemos pelo algoritmo PROJ-L são exibidas na Figura 5.3. Para este problema, a escolha $\mu = 1$ funcionou satisfatoriamente em todas as execuções.

	FP-L_D	P-LSQR	PROJ-L	FP-L_D	P-LSQR	PROJ-L
$\overline{\lambda}$	0,1563	-	0,1491	0,1577	-	0,1492
$\overline{E}$	0,1479	0,1483	0,1463	0,1472	0,1477	0,1462
$\overline{t}$	64,5863	36,4999	12,5072	426,0727	156,7809	17,6866
$k_m(k_M)$	516 (572)	309 (335)	39 (39)	282 (282)	163 (163)	39 (39)
		reortog=0			reortog=1	

Tabela 5.5: Resultado para o problema pirate com NL = 0,01, $p_0 = 20$ e $\varepsilon_1 = \varepsilon_2 = 10^{-6}$.

Observamos que, do mesmo modo como no exemplo anterior, o número de iterações reduziu significativamente quando usamos reortogonalização, mas o tempo gasto foi consideravelmente mais elevado, tanto para o algoritmo FP-L_D como para P-LSQR.

Figura 5.3: Imagem exata, imagem embaçada e melhor restauração obtida.

5.1.3 Problema 3: Super-Resolução de Imagens

O último exemplo para ilustrar os algoritmos propostos para o método de regularização de Tikhonov na forma geral consiste num problema de super-resolução de imagens.

Imagens de alta resolução (HR - *High-Resolution*) são importantes em inúmeras áreas tais como tomografia computadorizada e vigilância eletrônica. No entanto, devido às limitações de equipamentos e sistemas de aquisição de imagens (câmeras, por exemplo), geralmente temos disponível apenas imagens de baixa resolução (LR - *Low-Resolution*).

Vamos considerar o problema de estimar uma imagem de alta resolução a partir de várias imagens de baixa resolução e, para isto, vamos denotar por $f \in \mathbb{R}^M$ a imagem de alta resolução de tamanho $M = M_1 \times M_2$ e por $g_k \in \mathbb{R}^N$, $k = 1, \ldots, k_0$, a k-ésima imagem de baixa resolução de tamanho $N = N_1 \times N_2$, com $M_1 = N_1 \times D_1$ e $M_2 = N_2 \times D_2$, em que D_1 e D_2 representam fatores *down-sampling* nas direções verticais e horizontais, respectivamente. Assumindo que o processo de aquisição de imagens em baixa resolução envolve embaçamento, movimento, *subsampling* e ruído aditivo, um modelo que relaciona f com g_k pode ser escrito como [114]

$$g_k = A_k f + \epsilon_k, \tag{5.3}$$

em que a matriz $A_k \in \mathbb{R}^{N \times M}$ e ϵ_k representa o ruído. Nosso objetivo é estimar a imagem em alta resolução f a partir de todas as imagens em baixa resolução g_k. Neste caso, o método de regularização de Tikhonov é dado por

$$f_\lambda = \underset{f \in \mathbb{R}^M}{\operatorname{argmin}} \left\{ \|g - Af\|_2^2 + \lambda^2 \|Lf\|_2^2 \right\}, \tag{5.4}$$

em que $g = [g_1^T \cdots g_{k_0}^T]^T$, $A = [A_1^T \cdots A_{k_0}^T]^T$ e a matriz de regularização L dada como nos exemplos anteriores mas com dimensões apropriadas.

Neste exemplo, vamos estimar uma imagem de 96×96 pixels, chamada tree, a partir de uma sequência de 5 imagens de baixa resolução com $D_1 = D_2 = 2$, ou seja, as imagens de baixa resolução têm 48×48 pixels. Portanto, a matriz $A \in \mathbb{R}^{11.520 \times 9.216}$, a matriz de regularização $L \in \mathbb{R}^{18.240 \times 9.216}$ e optamos por usar $p_0 = 15$. O número de condição de A é $\kappa(A) \approx 7,1 \times 10^{16}$ e o posto numérico é de 9.021. Como ambas as matrizes não são

muito grandes, os algoritmos baseados no processo JBD são utilizados, novamente, com reortogonalização completa. Valores médios para 20 execuções são mostrados na Tabela 5.6.

	FP-LU	FP-L_D	P-LSQR	PROJ-L	JBD-FP	JBD-FP
$\overline{\lambda}$	0,0309	0,0309	-	0,0300	0,0309	0,0309
$\overline{E}$	0,0496	0,0496	0,0502	0,0535	0,0497	0,0497
$\overline{t}$	16,4869	9,1734	3.2403	0,8753	21,6653	25,1221
$k_m(k_M)$	281 (284)	281 (284)	157 (160)	47 (57)	112 (114)	112 (114)

Tabela 5.6: Resultados para o problema de super resolução com NL = 0,01, $\varepsilon_1 = \varepsilon_2 = 10^{-6}$ e $p_0 = 15$.

Analisando os dados desta tabela podemos concluir que PROJ-L é superior aos demais, pelo menos no sentido de ser o mais rápido. Para este exemplo, a escolha $\mu = 1$ funcionou satisfatoriamente em todos os casos. A imagem original em alta resolução e duas imagens em baixa resolução são exibidas na primeira linha da Figura 5.4. Uma imagem obtida pelo algoritmo PROJ-L é exibida na segunda linha da mesma figura.

Para ilustrar a performance dos algoritmos usando distintas matrizes de regularização, vamos considerar $L = I_M$ (no qual aplicamos o algoritmo GKB-FP original) e $L = L_{2,2D}$

$$L_{2,2D} = \begin{bmatrix} \mathcal{L}_{2,N} \otimes I_N \\ I_N \otimes \mathcal{L}_{2,N} \end{bmatrix}, \quad \mathcal{L}_{2,N} = \begin{bmatrix} -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \end{bmatrix} \in \mathbb{R}^{(N-2)\times N}, \tag{5.5}$$

em que $\mathcal{L}_{2,N}$ é uma versão discretizada do operador diferencial de segunda ordem. Neste caso, obtivemos soluções com erro relativo de aproximadamente 0,28 e 0,04, respectivamente. Isto é, enquanto a qualidade das soluções deteriorou significativamente para o caso $L = I_M$, as soluções utilizando $L = L_{2,2D}$ permaneceu com a mesma qualidade usando $L = L_{1,2D}$ (veja na Figura 5.4 as imagens obtidas).

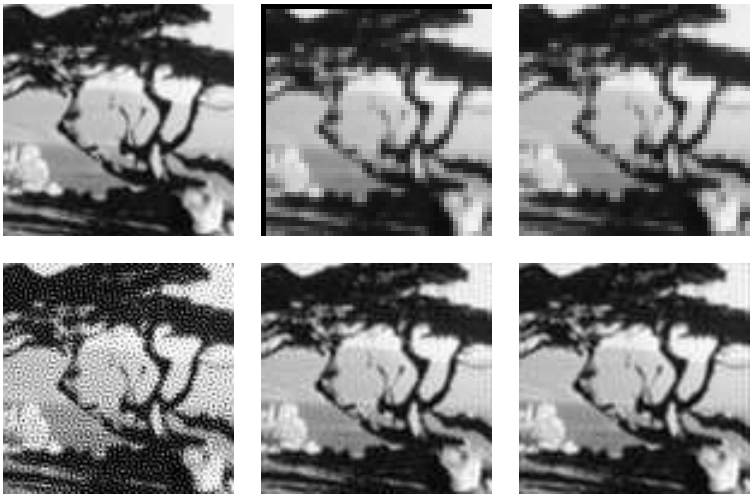


Figura 5.4: Primeira linha: Imagem em alta resolução (esquerda) e duas imagens em baixa resolução. Segunda linha: imagem obtida com $L = I_M$ determinada por GKB-FP (esquerda), com L de (5.1) determinada por PROJ-L (centro) e com $L_{2,2D}$ determinada por PROJ-L (direita).

5.2 Método de Regularização de Tikhonov com múltiplos parâmetros

Nesta seção, vamos usar quatro problemas mal-postos discretos para ilustrar a performance do algoritmo proposto no Capítulo 4, considerando regularização com dois e três parâmetros. Dois problemas estão disponíveis no pacote RegularizationTools [64], um é sobre espalhamento de onda [141] e um de super-resolução de imagens, o mesmo usado anteriormente. Para cada problema foram gerados 20 vetores de dados na forma $g = g_{\text{exato}} + \epsilon$ satisfazendo $\text{NL} = \|\epsilon\|_2/\|g_{\text{exato}}\|_2 = 10^{-3}, 10^{-2}, 2,5 \times 10^{-2}$.

Para comparação, vamos apresentar os resultados obtidos pelo método proposto por Brezinski et al. (CLC - *Constrained Linear Combination*) com o método FP para selecionar o parâmetro de regularização no problema (4.11), e o princípio da discrepância (DP) como descrito no Capítulo 4. Os valores iniciais para MFP foram considerados como os parâmetros usados por CLC e as iterações finalizadas quando $\|\lambda^{(k+1)} - \lambda^{(k)}\|_2 \leq \|\lambda^{(k)}\|_2 10^{-6}$. A aproximação inicial e demais parâmetros para DP são os mesmos utilizados em [96], isto é, consideramos $\lambda_1^{(0)} = \sqrt{0,2}$, $\lambda_2^{(0)} = \sqrt{0,1}$, $c = 1$, $\gamma = 0,5$ e $\delta = \|\epsilon\|_2$. Para descrevermos os resultados, assim como na seção anterior, usamos a seguinte notação:

- $\overline{E}_f$, $\overline{\lambda}_1$, $\overline{\lambda}_2$, $\overline{\lambda}_3$: valores médios do erro relativo em f_λ e valores médios dos parâmetros calculados;

- k_M: número máximo de iterações requerido por MFP para convergir ou a dimensão máxima do subespaço de Krylov $\mathcal{K}_k(A^TA, A^Tg)$ após a versão estendida de GKB-FP convergir.

5.2.1 Problema 4: Equações integrais de Fredholm de primeira espécie

Vamos considerar dois problemas mal-postos discretos originados da discretização de duas equações de Fredholm de primeira espécie da forma

$$\int_a^b K(s,t)f(t)dt = g(s), \quad c \leq s \leq d, \tag{5.6}$$

gerados pelas funções i_laplace e phillips de [64]. Cada uma destas funções nos fornece triplas $\{A, f_{\text{exato}}, g_{\text{exato}}\}$ tal que $Af_{\text{exato}} = g_{\text{exato}}$. Para as matrizes de regularização vamos usar as matrizes I_n, $\mathcal{L}_{1,n}$ e $\mathcal{L}_{2,n}$ (veja equações (5.1) e (5.5)).

Inversão da transformada de Laplace

Neste exemplo, o núcleo $K(s,t)$ em (5.6) é

$$K(s,t) = e^{-st}, \quad s,t \in [0,\infty), \tag{5.7}$$

e as funções $g(s)$ e $f(t)$ são dadas por

$$g(s) = 1/(s+1/2), \quad f(t) = e^{-t/2}. \tag{5.8}$$

Para o tamanho escolhido, $n = 256$, o posto numérico da matriz A é 36 e o número de condição $\kappa(A) \approx 1,4 \times 10^{33}$. Para as matrizes de regularização consideramos os casos: (i) $L_1 = I_n$ e $L_2 = \mathcal{L}_{1,n}$, e (ii) $L_1 = I_n$ e $L_2 = \mathcal{L}_{2,n}$.

Os resultados numéricos para o caso (i) são exibidos na Tabela 5.7. Notemos que todos os

algoritmos produziram resultados com a mesma ordem de acurácia e o número de iterações para MFP atingir o critério de parada foi relativamente pequeno (menos de 5). Com relação ao DP, este também é rápido (no sentido de exigir poucas iterações), mas notamos que a constante $c = 1$ pode exigir muitas iterações, como ocorreu com $NL = 0,001$, em que o número máximo de iterações (definido em 100) é atingido em algumas execuções. A razão é que o critério de parada para DP com $c = 1$, $\|g - Af\|_2 < \delta$, pode precisar de muitas iterações para ser satisfeito, assim, considerar $c \gtrapprox 1$ pode ser mais interessante. Repetimos o experimento com c =1,1 e o número máximo de iterações foi 9.

Os resultados para o caso (ii), exibidos na Tabela 5.8, mostram que MFP produziu resultados melhores do que DP e CLC com a observação de que CLC produziu soluções sem utilidade prática para NL = 0,01 e NL = 0,025.

	NL = 0,001			NL = 0,01			NL = 0,025		
	MFP	CLC	DP	MFP	CLC	DP	MFP	CLC	DP
$\overline{E}_f$	0,0170	0,0632	0,0104	0,0200	0,0597	0,0292	0,0277	0,0754	0,0579
$\overline{\lambda}_1$	0,0023	0,0023	0,0002	0,0229	0,0231	0,0482	0,0580	0,0584	0,0859
$\overline{\lambda}_2$	0,0322	0,0322	0,1980	0,3262	0,3257	0,2106	0,8315	0,8265	0,2318
k_M	3	–	100	4	–	8	5	–	6

Tabela 5.7: Resultados numéricos para o problema i_laplace com $L_1 = I_n$ e $L_2 = \mathcal{L}_{1,n}$.

	NL = 0,001			NL = 0,01			NL = 0,025		
	MFP	CLC	DP	MFP	CLC	DP	MFP	CLC	DP
$\overline{E}_f$	0,0176	0,5148	0,0615	0,0792	1,4898	0,1469	0,0809	1,9383	0,1627
$\overline{\lambda}_1$	0,0023	0,0023	0,0075	0,0255	0,0231	0,0429	0,0615	0,0584	0,0774
$\overline{\lambda}_2$	0,3864	0,3868	0,0115	6,7183	6,8705	0,0323	18,0429	17,6902	0,0658
k_M	6	–	13	10	–	8	10	–	7

Tabela 5.8: Resultados numéricos para o problema i_laplace com $L_1 = I_n$ e $L_2 = \mathcal{L}_{2,n}$.

No Capítulo 4 comentamos que DP pode produzir soluções ruins. Para ilustramos esta situação vamos considerar como aproximação inicial os mesmos valores utilizados por CLC e manter os demais parâmetros inalterados. Os resultados são mostrados na Tabela 5.9. Notemos que para NL = 0,01 e NL = 0,025 as soluções calculadas são dominadas pelo ruído (com erro relativo na solução superando 100%). A razão é que pelo fato de $\overline{\lambda}_1^2$ praticamente se anular (veja Tabela 5.9), as soluções determinadas por DP se comportam similarmente às encontradas pelo método de regularização de Tikhonov com $L = \mathcal{L}_{2,n}$ e o parâmetro escolhido

por DP, conforme visto na Tabela 5.10 em que os erros correspondentes às três escolhas de L são apresentados. Notemos que os erros nas soluções determinadas por DP (em negrito) estão próximos dos erros apresentados na Tabela 5.9 (em negrito também).

	NL = 0,001		NL = 0,01		NL = 0,025	
	MFP	DP	MFP	DP	MFP	DP
$\overline{E}_f$	0,0176	0,4883	0,0792	**1,0634**	0,0809	**1,2596**
$\overline{\lambda}_1$	0,0023	0,0003	0,0255	0,00000001	0,0615	0,00000002
$\overline{\lambda}_2$	0,3864	0,3376	6,7183	1,8030	18,0429	5,1006
k_M	6	35	10	42	10	44

Tabela 5.9: Resultados numéricos para i_laplace com $L_1 = I_n$ e $L_2 = \mathcal{L}_{2,n}$ e a mesma aproximação inicial para CLC.

	NL = 0,001		NL = 0,01		NL = 0,025	
	FP	DP	FP	DP	FP	DP
$L = I_n$:	0,1552	0,1612	0,1793	0,1866	0,1930	0,1995
$L = \mathcal{L}_{1,n}$:	0,0636	0,0127	0,0600	0,0475	0,0758	0,0702
$L = \mathcal{L}_{2,n}$:	0,5146	0,6402	1,4897	**1,1710**	1,9383	**1,2348**

Tabela 5.10: Erros médios para regularização com um parâmetro para o problema i_laplace.

Problema Phillips

Este problema foi estudado primeiramente por Phillips [115] e posteriormente analisado em diversos lugares. O núcleo é da forma

$$K(s,t) = \varphi(s-t), \quad s,t \in [-6,6], \tag{5.9}$$

com

$$\varphi(t) = \begin{cases} 1 + \cos(\pi t/3), & |t| < 3 \\ 0, & |t| \geq 3 \end{cases}, \tag{5.10}$$

enquanto que as funções $f(t)$ e $g(s)$ são

$$\begin{aligned} f(t) &= \varphi(t), \\ g(s) &= (6 - |s|)\left[1 + \frac{1}{2}\cos(\pi s/3)\right] + \frac{9}{2\pi}\,\mathrm{sen}\left(\frac{\pi|s|}{3}\right). \end{aligned} \tag{5.11}$$

Como antes, consideramos $n = 256$ e matrizes de regularização como em (i) e (ii) do exemplo anterior. O número de condição da matriz é $\kappa(A) \approx 1,13 \times 10^8$ e o posto numérico é 256. O conjunto de dados é gerado pela função `phillips` e os resultados estão nas Tabelas 5.11 e 5.12. Notemos que, independentemente do nível de ruído, todos os algoritmos obtiveram bons resultados com um número pequeno de iterações.

	NL = 0,001			NL = 0,01			NL = 0,025		
	MFP	CLC	DP	MFP	CLC	DP	MFP	CLC	DP
$\overline{E}_f$	0,0138	0,0143	0,0100	0,0218	0,0248	0,0238	0,0309	0,0328	0,0339
$\overline{\lambda}_1$	0,0049	0,0048	0,0534	0,0502	0,0494	0,1614	0,1274	0,1247	0,2468
$\overline{\lambda}_2$	0,1742	0,1741	0,0836	1,7829	1,7806	0,2155	4,5766	4,5476	0,3807
k_M	4	–	7	4	–	3	5	–	2

Tabela 5.11: Resultados numéricos para o problema phillips com $L_1 = I_n$ e $L_2 = \mathcal{L}_{1,n}$.

	NL = 0,001			NL = 0,01			NL = 0,025		
	MFP	CLC	DP	MFP	CLC	DP	MFP	CLC	DP
$\overline{E}_f$	0,0097	0,0097	0,0174	0,0262	0,0262	0,0240	0,0477	0,0461	0,0344
$\overline{\lambda}_1$	0,0050	0,0048	0,1581	0,0507	0,0494	0,1614	0,1326	0,1247	0,2468
$\overline{\lambda}_2$	3,7597	3,7591	0,00003	39,4760	39,3964	0,0009	110,507	109,297	0,1383
k_M	4	–	3	5	–	3	9	–	2

Tabela 5.12: Resultados numéricos para o problema phillips com $L_1 = I_n$ e $L_2 = \mathcal{L}_{2,n}$.

5.2.2 Problema 5: Problema de espalhamento

Apesar de considerarmos $A \in \mathbb{R}^{m\times n}$ e $g \in \mathbb{R}^m$, as discussões e os resultados que apresentamos neste trabalho podem ser facilmente estendidos para $A \in \mathbb{C}^{m\times n}$ e $g \in \mathbb{C}^m$. Assim, veremos agora um exemplo em que os dados estão no corpo dos complexos.

Problemas de espalhamento, que surgem naturalmente em diversas áreas como acústica e eletromagnetismo [36], estão relacionados com os efeitos que uma onda, chamada onda incidente e denotada por $u^i(x)$, sofre ao colidir com um objeto $D \subset \mathbb{R}^2$ gerando, assim, uma onda espalhada $u^s(x)$. Na Figura 5.5 exibimos uma interpretação física do problema de espalhamento.

Para uma onda incidente $u^i(x) = e^{ikd^T x} \in \mathbb{R}^2$, em que $k > 0$ é o número de onda e $d \in \mathbb{R}^2$ a direção da onda, a onda espalhada por um objeto impenetrável $D \subset \mathbb{R}^2$ com fronteira do

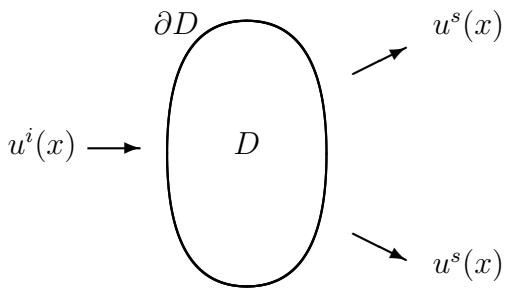


Figura 5.5: Ilustração física do problema de espalhamento.

tipo *sound-soft* é descrita pelo sistema

$$\begin{cases} \Delta u + k^2 u = 0, & \text{em } \mathbb{R}^2 \backslash \overline{D}, \\ u = 0, & \text{em } \partial D, \\ \dfrac{\partial u^s}{\partial r} - iku^s = \mathcal{O}\left(\dfrac{1}{r}\right), & r = \|x\|_2 \to \infty, \end{cases} \tag{5.12}$$

em que $u(x) = u^s(x) + u^i(x)$ é a onda total e $u^s(x)$ é a onda espalhada pelo exterior de D que admite a seguinte assintótica

$$u^s(x) = \frac{e^{ik\|x\|_2}}{\sqrt{\|x\|_2}} \left\{ u_\infty(\hat{x}) + \mathcal{O}\left(\frac{1}{\|x\|_2}\right) \right\}, \quad \hat{x} = \frac{x}{\|x\|_2}, \quad \|x\|_2 \to \infty. \tag{5.13}$$

A função $u_\infty(\hat{x})$ é chamada de amplitude de espalhamento de $u^s(x)$, também conhecida como *far-field pattern* da onda espalhada e é fisicamente mensurável. Sugerimos [23, 33, 36, 94, 95, 116, 117] para mais informações sobre problemas de espalhamento direto e inverso.

O problema inverso que estamos interessados consiste na determinação da onda espalhada $u^s(x)$ a partir das informações obtidas da amplitude de espalhamento $u_\infty(\hat{x})$ que pode ser medida, ou seja, não dispomos de $u_\infty(\hat{x})$ exata, e sim apenas de uma aproximação de $u_\infty(\hat{x})$. Usando o potencial *single-layer* [36], podemos expressar a onda espalhada por

$$u^s(x) = \int_{\partial D} \varphi(y)\Phi(x,y)ds(y), \quad \Phi(x,y) = \frac{i}{4} H_0^{(1)}(k\|x-y\|_2), \tag{5.14}$$

em que $H_0^{(1)}(\cdot)$ é a função de Hankel de primeira espécie de ordem zero e a amplitude de

espalhamento $u_\infty(\hat{x})$ por

$$u_\infty(\hat{x}) = \frac{e^{i\pi/4}}{\sqrt{8k\pi}} \int_{\partial D} \varphi(y) e^{-ik\hat{x}^T y} ds(y), \quad \hat{x} \in \mathbb{S}^1 = \{x \in \mathbb{R}^2 \ / \ \|x\|_2 = 1\}. \tag{5.15}$$

Assim, dada $u_\infty(\hat{x})$, ou uma aproximação, encontramos a função densidade $\varphi(y)$ resolvendo o problema inverso (5.15) e, em seguida, calculamos $u^s(x)$ por (5.14), que é um problema direto.

Para ilustrarmos numericamente o uso do método de regularização de Tikhonov com múltiplos parâmetros, usamos os mesmos dados usados em [141, Exemplo 3]. Assim, a fronteira do conjunto $D \subset \mathbb{R}^2$ é $\partial D = \{(3\cos t, 4\sin t) \in \mathbb{R}^2 \ / \ t \in [0, 2\pi]\}$ e a onda incidente é dada por $u^i(x) = \dfrac{i}{4} H_0^{(1)}(k\|x - y_0\|_2)$ com $k = 2$ e $y_0 = [0\ 1]^T$. Neste caso particular, a onda espalhada e a amplitude de espalhamento são, respectivamente,

$$u^s(x) = -\frac{i}{4} H_0^{(1)}(k\|x - y_0\|_2), \quad u_\infty(\hat{x}) = -\frac{i}{4}\sqrt{\frac{2}{k\pi}} e^{-ik\left(\hat{x}^T y_0 + \pi/(4k)\right)}. \tag{5.16}$$

Discretizamos a equação integral (5.15) usando quadratura de ponto médio com $n = 256$ pontos e obtemos um sistema linear $Af = g$ em que $A \in \mathbb{C}^{n\times n}$. O número de condição da matriz gerada para este problema é $\kappa(A) \approx 2,72 \times 10^{18}$ e o posto numérico é 55.

Inicialmente aplicamos o método de regularização de Tikhonov com apenas um parâmetro ao sistema $Af = g$ e, uma vez encontrada a solução regularizada f_λ, ou seja, uma aproximação para a função densidade $\varphi(\lambda)$, usamos esta informação para calcular a onda espalhada $u^s(x)$. Para a seleção do parâmetro λ optamos por usar o método de ponto fixo e o princípio da discrepância. Os resultados encontrados estão nas Tabelas 5.13 a 5.15.

	$L = I_n$		$L = \mathcal{L}_1$		$L = \mathcal{L}_2$	
	FP	DP	FP	DP	FP	DP
$\overline{E}_f$	0,0023	0,0007	0,0012	0,0019	0,0007	0,0011
$\overline{\lambda}$	0,0001	0,0017	0,0019	0,0084	0,0220	0,0950

Tabela 5.13: Resultados numéricos para o problema de espalhamento com para diferentes regularizadores e NL = 0,001.

Analisando a qualidade das soluções encontradas, notamos que o uso de apenas um parâmetro é capaz de encontrar soluções de boa qualidade.

	$L = I_n$		$L = \mathcal{L}_1$		$L = \mathcal{L}_2$	
	FP	DP	FP	DP	FP	DP
$\overline{E}_f$	0,0074	0,0058	0,0054	0,0079	0,0043	0,0063
$\overline{\lambda}$	0,0011	0,0053	0,0196	0,0619	0,2253	0,6375

Tabela 5.14: Resultados numéricos para o problema de espalhamento com para diferentes regularizadores e NL = 0,01.

	$L = I_n$		$L = \mathcal{L}_1$		$L = \mathcal{L}_2$	
	FP	DP	FP	DP	FP	DP
$\overline{E}_f$	0,0135	0,0134	0,0112	0,0153	0,0098	0,0135
$\overline{\lambda}$	0,0027	0,0083	0,0498	0,1203	0,5723	1,2482

Tabela 5.15: Resultados numéricos para o problema de espalhamento com para diferentes regularizadores e NL = 0,025.

Utilizando o método de regularização de Tikhonov com múltiplos parâmetros, os resultados encontrados estão nas Tabelas 5.16 e 5.17. Notemos que, independentemente do nível de ruído, tanto MFP quanto CLC produziram soluções com ótima qualidade enquanto que DP falhou em todos os casos. Repetimos todos os experimentos considerando como aproximação inicial para DP os mesmos valores utilizados por CLC e os demais parâmetros inalterados e, assim, os resultados que obtivemos foram da mesma qualidade que MFP e CLC.

	NL = 0,001			NL = 0,01			NL = 0,025		
	MFP	CLC	DP	MFP	CLC	DP	MFP	CLC	DP
$\overline{E}_f$	0,0012	0,0012	0,5489	0,0053	0,0054	0,5490	0,0111	0,0111	0,5491
$\overline{\lambda}_1$	0,0001	0,0001	0,1581	0,0011	0,0008	0,1581	0,0027	0,0019	0,1581
$\overline{\lambda}_2$	0,0019	0,0014	0,0002	0,0196	0,0139	0,0002	0,0499	0,0352	0,0002
k_M	5	–	3	5	–	3	6	–	3

Tabela 5.16: Resultados numéricos para o problema de espalhamento com $L_1 = I_n$ e $L_2 = \mathcal{L}_{1,n}$.

	NL = 0,001			NL = 0,01			NL = 0,025		
	MFP	CLC	DP	MFP	CLC	DP	MFP	CLC	DP
$\overline{E}_f$	0,0007	0,0007	0,5489	0,0043	0,0043	0,5490	0,0099	0,0098	0,5491
$\overline{\lambda}_1$	0,0001	0,0001	0,1581	0,011	0,0008	0,1581	0,0028	0,0019	0,1581
$\overline{\lambda}_2$	0,0220	0,0156	0,0000001	0,2254	0,1593	0,0000001	0,5739	0,4047	0,0000001
k_M	5	–	3	5	–	3	6	–	3

Tabela 5.17: Resultados numéricos para o problema de espalhamento com $L_1 = I_n$ e $L_2 = \mathcal{L}_{2,n}$.

Por fim, exibimos na Figura 5.6 a solução exata $u^s(x)$ e a solução $u^s_\lambda(x)$ obtida pelo método de regularização de Tikhonov com múltiplos parâmetros e os parâmetros λ_1 e λ_2 escolhidos por MFP. As matrizes de regularização neste caso foram $L_1 = I_n$ e $L_2 = \mathcal{L}_{2,n}$.

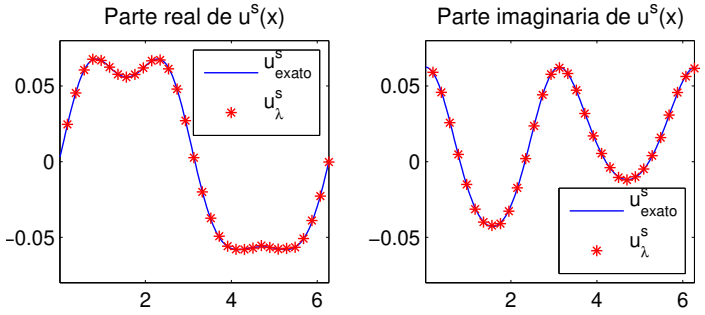


Figura 5.6: Parte real (esquerda) e parte imaginária da onda espalhada $u^s(x)$ e $u^s_\lambda(x)$ para o problema de espalhamento com $n = 256$, NL = 0,025 e regularizadores identidade e uma versão discretizada do operador diferencial de segunda ordem.

5.2.3 Problema 6: Super-Resolução de Imagens (revisitado)

Para finalizarmos, vamos usar novamente o problema de super-resolução descrito na seção 5.1.3, mas agora consideramos dois e três parâmetros envolvendo a matriz identidade e operadores diferenciais $L_{1,2D}$ e $L_{2,2D}$ (veja equações (5.1) e (5.5)). Como $L_{1,2D} \in \mathbb{R}^{18.240\times 9.216}$ e $L_{2,2D} \in \mathbb{R}^{18.048\times 9.216}$, temos que a matriz $\overline{A}_\lambda$ em (4.5) se torna grande demais e o problema (4.1) não pode ser resolvido eficientemente via decomposição QR como feito nos exemplos anteriores. Assim, usaremos nosso algoritmo de projeção.

Usaremos a seguinte notação para designar o conjuntos de regularizadores: A $= \{L_1 = I_{M^2}, L_2 = L_{1,2D}\}$; B $= \{L_1 = I_{M^2}, L_2 = L_{2,2D}\}$; C $= \{L_1 = I_{M^2}, L_2 = L_{1,2D}, L_3 = L_{2,2D}\}$. A média dos erros relativos e parâmetros de regularização, e a dimensão máxima do subespaço de Krylov necessário estão na Tabela 5.18. Podemos concluir que a qualidade da solução calculada para cada nível de ruído fica dentro de valores aceitáveis independente do conjunto de regularizadores. Na Figura 5.4 exibimos uma imagem em baixa resolução, embaçada e com ruído, a imagem em alta resolução exata, e três imagems em alta resolução correspondentes aos conjuntos A, B e C. Como as imagens obtidas são praticamente indistinguíveis independente do nível de ruído, concluímos que para este problema não há necessidade de aplicarmos regularização com três parâmetros.

	NL = 0,001			NL = 0,01			NL = 0,025		
	A	B	C	A	B	C	A	B	C
$\overline{E}_f$	0,0361	0,0274	0,0310	0,0533	0,0493	0,0516	0,0709	0,0720	0,0703
$\overline{\lambda}_1$	0,0007	0,0007	0,0007	0,0082	0,0086	0,0089	0,0222	0,0238	0,0257
$\overline{\lambda}_2$	0,0025	0,0026	0,0027	0,0301	0,0335	0,0340	0,0827	0,0970	0,1019
$\overline{\lambda}_3$	–	–	0,0027	–	–	0,0351	–	–	0,1104
k_M	313	207	177	51	47	46	38	41	54

Tabela 5.18: Resultados numérios para o problema de super-resolução tree.

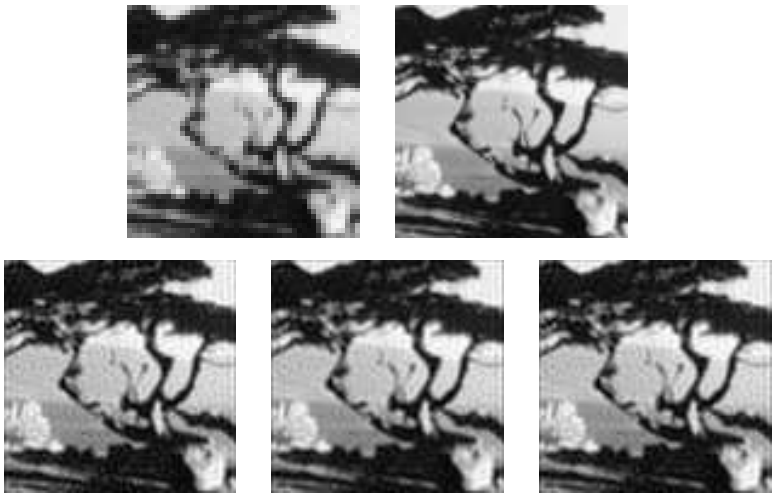


Figura 5.7: Linha superior: imagem embaçada+ruído (esquerda) e imagem em alta resolução (direita). Linha inferior: imagens estimadas para A (esquerda), B (centro) e C (direita), respectivamente. NL = 0,025.

Capítulo 6

Conclusão

Desenvolvemos o algoritmo GKB-FP [10] com detalhes a respeito das suas propriedades de convergência. Além disso, mostramos como podemos estendê-lo para ser aplicado ao método de regularização de Tikhonov na forma geral, em especial para problemas de grande porte. Como resultado, duas distintas abordagens que não exigem conhecimentos a respeito do ruído foram propostas e ilustradas numericamente em problemas de grande porte do tipo *deblurring* e super-resolução de imagens. Resultados numéricos dispostos no Capítulo 5 mostraram que em termos de acurácia e eficiência, as versões estendidas de GKB-FP funcionaram satisfatoriamente do mesmo modo que a versão original do algoritmo e são, portanto, competitivos e atrativos para problemas de grande porte para o método de regularização de Tikhonov na forma geral. Para contornar as dificuldades associadas com a determinação do parâmetro de regularização, propomos no Capítulo 3 um critério de parada automático para métodos iterativos e mostramos, através do algoritmo P-LSQR (no qual informações a priori da solução do problema são incorporadas nas soluções iteradas), que esta abordagem é uma boa alternativa ao método de regularização de Tikhonov.

No Capítulo 4 apresentamos um algoritmo de ponto fixo para a escolha dos parâmetros de regularização para o método de regularização de Tikhonov com múltiplos parâmetros que não requer nenhum conhecimento a respeito do nível de ruído nos dados. A análise e o algoritmo apresentados podem ser vistos como a generalização natural dos resultados e do algoritmo publicados em [9]. Também, seguindo as mesmas ideias de GKB-FP (que pode ser visto como o método de ponto fixo para o método de regularização de Tikhonov na forma padrão

de grande porte), propomos um algoritmo do tipo GKB-FP para o método de regularização de Tikhonov com múltiplos parâmetros para problemas de grande porte. Os resultados numéricos no Capítulo 5, indicaram que nosso algoritmo pode produzir soluções regularizadas com acurácia comparável a produzida pelo princípio da discrepância, porém, sem suas dificuldades como a perda de unicidade dos parâmetros e a necessidade de conhecimentos a priori da norma do ruído.

Experiências com os algoritmos propostos neste trabalho com problemas das mais diversas áreas são necessárias para que todo o potencial destes seja explorado, bem como o uso de outros regularizadores que incorporem mais características nas soluções regularizadas.

Como sugestões para trabalhos futuros propomos:

1. análise de erro para as soluções regularizadas f_λ e f_k em que:

   a) f_λ é calculada pelo método de regularização de Tikhonov com o parâmetro λ calculado pelo método de ponto fixo;

   b) f_k é calculada por métodos iterativos e o parâmetro k selecionado pelo critério automático proposto;

2. resolver de modo mais eficiente o problema de minimização associado ao método de regularização de Tikhonov com múltiplos parâmetros;

3. análise mais rigorosa para existência de ponto fixo no caso de múltiplos parâmetros, análise de erro e de convergência do algoritmo MFP;

4. utilização de normas p em vez da norma 2:

   - $p \in (1, 2)$, em especial $p \approx 1$;
   - análise da solução em função de p;
   - múltiplos parâmetros e para cada λ_i uma norma p_i;

5. aplicar as técnicas aqui descritas em problemas de mínimos quadrados não-lineares [31] em conjunto com métodos como Gauss-Newton e Levenberg-Marquardt.

Apêndice A

Produto de Kronecker

Definição A.1. *Dadas duas matrizes $A \in \mathbb{R}^{m\times n}$ e $B \in \mathbb{R}^{p\times q}$, o produto de Kronecker entre A e B é a matriz $C = A \otimes B \in \mathbb{R}^{mp\times nq}$ definida por [92]*

$$C = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix}.$$

A seguir fornecemos uma série de resultados, alguns dos quais utilizados neste trabalho, a respeito do produto de Kronecker.

Proposição A.2. *Sejam vetores $x \in \mathbb{R}^m$ e $y \in \mathbb{R}^n$ então*

$$x \otimes y^T = xy^T \in \mathbb{R}^{m\times n}.$$

Demonstração.

$$x \otimes y^T = \begin{bmatrix} x_1y_1 & \cdots & x_1y_n \\ \vdots & \ddots & \vdots \\ x_my_1 & \cdots & x_my_n \end{bmatrix} = xy^T \in \mathbb{R}^{m\times n}.$$

□

Proposição A.3. *Sejam matrizes A, B e C de dimensões compatíveis quando necessário e α escalar, então*

1. $A \otimes (B + C) = A \otimes B + A \otimes C$,

2. $(A + B) \otimes C = A \otimes C + B \otimes C$,

3. $(A \otimes B) \otimes C = A \otimes (B \otimes C)$,

4. $(\alpha A) \otimes B = A \otimes (\alpha B) = \alpha(A \otimes B)$,

5. $(A \otimes B)^T = A^T \otimes B^T$.

Demonstração. Seguem imediatamente da definição do produto de Kronecker.

□

Proposição A.4. *Sejam A e B matrizes simétricas, então a matriz $A \otimes B$ também é simétrica.*

Demonstração. Pelo item 5 da proposição A.3 temos

$$(A \otimes B)^T = A^T \otimes B^T = A \otimes B.$$

□

O próximo resultado relaciona o produto entre matrizes de Kronecker com um único produto de Kronecker. Este resultado foi utilizado no Capítulo 2 (veja equações (2.71)-(2.72)).

Proposição A.5. *Sejam A, B, C e D matrizes com dimensões tal que o produto AC e BD seja possível, então*

$$(A \otimes B)(C \otimes D) = AC \otimes BD.$$

Demonstração. A demonstração segue imediatamente da definição:

$$
\begin{aligned}
(A \otimes B)(C \otimes D) &= \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix} \begin{bmatrix} c_{11}D & \cdots & c_{1p}D \\ \vdots & \ddots & \vdots \\ c_{n1}D & \cdots & c_{np}D \end{bmatrix} \\
&= \begin{bmatrix} \sum_{k=1}^{n} a_{1k}c_{k1}BD & \cdots & \sum_{k=1}^{n} a_{1k}c_{kp}BD \\ \vdots & \ddots & \vdots \\ \sum_{k=1}^{n} a_{mk}c_{k1}BD & \cdots & \sum_{k=1}^{n} a_{mk}c_{kp}BD \end{bmatrix} \\
&= (AC) \otimes (BD).
\end{aligned}
$$

□

Proposição A.6. *Sejam $A \in \mathbb{R}^{m\times m}$ e $B \in \mathbb{R}^{n\times n}$ matrizes não-singulares, então a matriz $(A \otimes B)$ é não-singular e sua inversa é dada por*

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}.$$

Demonstração. Pela proposicao A.5 temos

$$(A \otimes B)(A^{-1} \otimes B^{-1}) = AA^{-1} \otimes BB^{-1} = I \otimes I = I.$$

□

Os próximos dois resultados mostram como o traço e a norma de Frobenius de uma matriz $C = A \otimes B$ estão relacionados com o traço e as normas de Frobenius das matrizes A e B.

Proposição A.7. *Sejam A e B matrizes quadradas, então*

$$tr(A \otimes B) = tr(A)tr(B) = tr(B \otimes A).$$

Demonstração. Seja $A \in \mathbb{R}^{n\times n}$ e $B \in \mathbb{R}^{m\times m}$, então

$$\mathrm{tr}(A \otimes B) = \sum_{i=1}^{n} \mathrm{tr}(a_{ii}B) = \sum_{i=1}^{n} a_{ii}\mathrm{tr}(B) = \mathrm{tr}(B)\sum_{i=1}^{n} a_{ii} = \mathrm{tr}(B)\mathrm{tr}(A).$$

□

Proposição A.8. *Sejam as matrizes $A \in \mathbb{R}^{m\times n}$ e $B \in \mathbb{R}^{p\times q}$, então*

$$\|A \otimes B\|_F = \|A\|_F \|B\|_F.$$

Demonstração. Pelo fato de $\mathrm{tr}(A^TA) = \|A\|_F^2$ e pela proposição anterior

$$\|A \otimes B\|_F^2 = \mathrm{tr}\left((A \otimes B)^T(A \otimes B)\right) = \mathrm{tr}(A^TA)\mathrm{tr}(B^TB) = \|A\|_F^2\|B\|_F^2.$$

□

Na proposição anterior, vale para outras normas como $\|\cdot\|_1$, $\|\cdot\|_2$ e $\|\cdot\|_\infty$. Veremos um resultado sobre matrizes normais e posteriormente sobre a SVD.

Proposição A.9. *Sejam A e B matrizes normais, então a matriz $A \otimes B$ também é normal. E se A e B são matrizes ortogonais, então a matriz $A \otimes B$ também é ortogonal.*

Demonstração. Como as matrizes A e B são matrizes normais temos que $AA^T = A^TA$ e $BB^T = B^TB$. Assim, pelo item 5 da proposição A.3 e pela proposição A.5 segue que

$$(A \otimes B)(A \otimes B)^T = AA^T \otimes BB^T = A^TA \otimes B^TB = (A \otimes B)^T(A \otimes B).$$

Se as matrizes A e B são ortogonais, então a matriz $A \otimes B$ é normal (pois $A^TA = AA^T = I$ e $B^TB = BB^T = I$). Falta provarmos que $(A \otimes B)(A \otimes B)^T = I$. Mas isto é imediato, pois $(A \otimes B)(A \otimes B)^T = AA^T \otimes BB^T = I \otimes I = I$.

□

Para finalizarmos, vamos mostrar um resultado com relação a decomposição em valores singulares da matriz $A \otimes B$ levando em consideração a decomposição em valores singulares

das matrizes A e B e como realizar eficientemente o produto matriz-vetor $(A \otimes B)x$ sem formar a matriz $A \otimes B$.

Proposição A.10. *Sejam as matrizes $A \in \mathbb{R}^{m\times n}$ e $B \in \mathbb{R}^{p\times q}$ e suas SVD's $A = U_A\Sigma_A V_A^T$ e $B = U_B\Sigma_B V_B^T$. Então*

$$(U_A \otimes U_B)(\Sigma_A \otimes \Sigma_B)(V_A^T \otimes V_B^T),$$

fornece uma SVD da matriz $A \otimes B$ (após a reordenação apropriada dos elementos da matriz diagonal $\Sigma_A \otimes \Sigma_B$ e dos vetores singulares correspondentes a direita e a esquerda).

Demonstração. A demonstração segue através da proposição A.5

$$\begin{aligned} A \otimes B &= (U_A\Sigma_A V_A^T) \otimes (U_B\Sigma_B V_B^T) \\ &= (U_A\Sigma_A \otimes U_B\Sigma_B)(V_A^T \otimes V_B^T) \\ &= (U_A \otimes U_B)(\Sigma_A \otimes \Sigma_B)(V_A^T \otimes V_B^T). \end{aligned}$$

□

Por fim, dado um vetor x, o produto matriz-vetor $(A \otimes B)x$ pode ser realizado sem a necessidade da matriz $A \otimes B$ ser formada. Para isto vamos usar a função $\mathrm{vec}(X)$, em que $X = [x_1 \cdots x_n] \in \mathbb{R}^{m\times n}$, dada por

$$\mathrm{vec}(X) = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}. \tag{A.1}$$

Em outras palavras, $\mathrm{vec}(X)$ transforma a matriz em um vetor empilhando suas colunas.

Teorema A.11. *Sejam as matrizes $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{p\times q}$ e o vetor $x \in \mathbb{R}^{nq}$ escrito como*

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \tag{A.2}$$

em que $x_i \in \mathbb{R}^q$. *Seja a matriz* $X = [x_1 \;\; x_2 \;\; \cdots \;\; x_n] \in \mathbb{R}^{q\times n}$ *tal que* $vec(X) = x$, *então*

$$(A \otimes B)x = vec(BXA^T). \tag{A.3}$$

Demonstração. A demonstração segue por verificação direta. Primeiro temos

$$(A \otimes B)x = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, = \begin{bmatrix} a_{11}Bx_1 + \cdots + a_{1n}Bx_n \\ \vdots \\ a_{m1}Bx_1 + \cdots + a_{mn}Bx_n \end{bmatrix}. \tag{A.4}$$

Vamos analisar a matriz BXA^T,

$$\begin{aligned} B[x_1 \;\; \cdots \;\; x_n] \begin{bmatrix} a_{11} & \cdots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \cdots & a_{mn} \end{bmatrix} &= [Bx_1 \;\; \cdots \;\; Bx_n] \begin{bmatrix} a_{11} & \cdots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \cdots & a_{mn} \end{bmatrix} \\ &= [\tilde{x}_1 \;\; \cdots \;\; \tilde{x}_m] = \tilde{X}, \end{aligned} \tag{A.5}$$

em que

$$\tilde{x}_i = a_{i1}Bx_1 + a_{i2}Bx_2 + \cdots a_{in}Bx_n, \tag{A.6}$$

para $i = 1, \ldots, m$. O resultado segue fazendo vec$(\tilde{X})$.

□

Apêndice B

Ângulo e distância entre subespaços

O algoritmo GKB-FP calcula uma aproximação para a solução f_λ no subespaço de Krylov $\mathcal{K}_k(A^TA, A^Tg)$ de dimensão k. Podemos pensar em substituir este subespaço por qualquer outro subespaço e uma questão natural neste contexto é como comparar estes subespaços ou, de um modo mais geral, dados dois subespaços não-nulos $\mathcal{U}, \mathcal{V} \subset \mathbb{R}^n$, que tipo de comparação pode ser feita entre eles?

No caso mais simples, ou seja, quando queremos comparar dois subespaços unidimensionais $\mathcal{U} = \text{span}\{u\}$ e $\mathcal{V} = \text{span}\{v\}$ com $\|u\|_2 = \|v\|_2 = 1$, imediatamente pensamos no ângulo θ entre os vetores u e v dado por $\cos(\theta) = u^Tv$. Pensar em definir este θ como sendo o ângulo entre $\mathcal{U}$ e $\mathcal{V}$ pode não ser interessante caso $\cos(\theta) < 0$, assim, podemos escolher $-u$ ou $-v$ de tal forma que $\cos(\theta) \geq 0$. Logo, parece razoável que o ângulo entre $\mathcal{U}$ e $\mathcal{V}$ seja o número θ tal que

$$\cos(\theta) = \max_{\substack{u\in\mathcal{U}, v\in\mathcal{V} \\ \|u\|_2=\|v\|_2=1}} u^Tv, \tag{B.1}$$

e como consequência $0 \leq \theta \leq \pi/2$.

Com esta ideia em mente, temos a seguinte definição [102].

Definição B.1. *Sejam $\mathcal{U}, \mathcal{V} \subset \mathbb{R}^n$ subespaços não-nulos. O ângulo mínimo entre eles é o número $\theta_{\min}$ definido por*

$$\cos(\theta_{\min}) = \max_{\substack{u\in\mathcal{U}, v\in\mathcal{V} \\ \|u\|_2=\|v\|_2=1}} u^Tv. \tag{B.2}$$

Claramente que do ponto de vista prático, calcular este ângulo através da definição é

inviável. O lema a seguir fornece uma simples expressão para o cálculo deste ângulo mínimo através de projetores ortogonais.

Lema B.2. *Sejam $\mathcal{U}, \mathcal{V} \subset \mathbb{R}^n$ subespaços não-nulos e $P_\mathcal{U}, P_\mathcal{V}$ projetores ortogonais sobre $\mathcal{U}$ e $\mathcal{V}$ respectivamente. Então,*

$$\cos(\theta_{\min}) = \|P_\mathcal{U} P_\mathcal{V}\|_2. \tag{B.3}$$

Demonstração. Para todo $x, y \in \mathbb{R}^n$ tal que $\|x\|_2 = \|y\|_2 = 1$ temos que $P_\mathcal{U} x \in \mathcal{U}$ e $P_\mathcal{V} y \in \mathcal{V}$. Logo,

$$\cos(\theta_{\min}) = \max_{\substack{u \in \mathcal{U}, v \in \mathcal{V} \\ \|u\|_2 = \|v\|_2 = 1}} u^T v = \max_{\|x\|_2 \leq 1, \|y\|_2 \leq 1} y^T P_\mathcal{U} P_\mathcal{V} x = \|P_\mathcal{U} P_\mathcal{V}\|_2, \tag{B.4}$$

em que, na igualdade do meio, usamos o fato de que se $h : \mathcal{X} \to \mathbb{R}$ é uma função definida num subespaço $\mathcal{X}$ tal que $h(\alpha x) = \alpha h(x)$ para todo escalar $\alpha \geq 0$, então $\max_{\|x\|_2 = 1} h(x) = \max_{\|x\|_2 \leq 1} h(x)$ e na última igualdade usamos o fato de que $\|A\|_2 = \max_{\|y\|_2 = \|x\|_2 = 1} |y^T A x|$.

□

Vamos agora nos concentrar na distância entre dois subespaços não-nulos e para isso precisamos de alguns conceitos preliminares.

Definição B.3. *Seja $\mathcal{U} \subset \mathbb{R}^n$ um subespaço não-nulo e $v \in \mathbb{R}^n$ um vetor qualquer. A distância ortogonal entre v e $\mathcal{U}$ é definida por*

$$\min_{u \in \mathcal{U}} \|v - u\|_2 = \mathit{dist}(v, \mathcal{U}). \tag{B.5}$$

Lema B.4. *Sob as condições da definição acima, existe um único vetor $\bar{u} \in \mathcal{U}$ tal que*

$$\min_{u \in \mathcal{U}} \|v - u\|_2 = \|v - \bar{u}\|_2, \tag{B.6}$$

e este é dado por $\bar{u} = P_\mathcal{U} v$, em que $P_\mathcal{U}$ é o projetor ortogonal sobre $\mathcal{U}$.

Demonstração. Se $\bar{u} = P_\mathcal{U} v$, então $\bar{u} - u \in \mathcal{U}$ para todo $u \in \mathcal{U}$ e consequentemente

$$v - \bar{u} = (I_n - P_\mathcal{U}) v \in \mathcal{U}^\perp. \tag{B.7}$$

logo

$$\|v-u\|_2^2 = \|v-\bar{u}\|_2^2 + \|\bar{u}-u\|_2^2 \geq \|v-\bar{u}\|_2^2. \tag{B.8}$$

Portanto, $\min_{u\in\mathcal{U}} \|v-u\|_2 = \|v-\bar{u}\|_2$. Com relação a unicidade, seja $\tilde{u} \in \mathcal{U}$ tal que $\|v-\tilde{u}\|_2 = \|v-\bar{u}\|_2$, então $\|v-\tilde{u}\|_2^2 = \|v-\bar{u}\|_2^2 + \|\bar{u}-\tilde{u}\|_2^2 \Rightarrow \|\bar{u}-\tilde{u}\|_2 = 0$. Portanto $\tilde{u} = \bar{u}$.

□

Isto nos conduz à definição de distância direta entre dois subespaços.

Definição B.5. *Sejam $\mathcal{U}, \mathcal{V} \subset \mathbb{R}^n$ subespaços não-nulos. A distância direta entre eles é dada por*

$$\delta(\mathcal{U},\mathcal{V}) = \max_{\substack{u\in\mathcal{U} \\ \|u\|_2=1}} dist(u,\mathcal{V}) = \max_{\substack{u\in\mathcal{U} \\ \|u\|_2=1}} \|(I_n - P_{\mathcal{V}})u\|_2. \tag{B.9}$$

De modo geral, não vale $\delta(\mathcal{U},\mathcal{V}) = \delta(\mathcal{V},\mathcal{U})$. Por exemplo, seja $\mathcal{V} = \text{span}\{(0,1,1)\}$ e $\mathcal{U}$ o plano $\Pi = \{(x,y,0) \ / \ x,y \in \mathbb{R}\}$, então $\delta(\mathcal{U},\mathcal{V}) = 1$ e $\delta(\mathcal{V},\mathcal{U}) = 1/\sqrt{2}$.

Assim, definimos a distância entre dois subespaços como sendo o máximo entre as distâncias diretas.

Definição B.6. *Sejam $\mathcal{U}, \mathcal{V} \subset \mathbb{R}^n$ subespaços não-nulos. A distãncia, ou gap, entre eles é dada por*

$$gap(\mathcal{U},\mathcal{V}) = \max\{\delta(\mathcal{U},\mathcal{V}), \delta(\mathcal{V},\mathcal{U})\}. \tag{B.10}$$

Do mesmo como usamos projetores ortogonais para calcular o ângulo mínimo entre subespaços, vamos usá-los para calcular a distância entre subespaços. E isto é dado pelo lema a seguir.

Lema B.7. *Sejam $\mathcal{U}, \mathcal{V} \subset \mathbb{R}^n$ subespaçoes não-nulos e $P_{\mathcal{U}}, P_{\mathcal{V}}$ projetores ortogonais sobre $\mathcal{U}$ e $\mathcal{V}$ respectivamente. Então*

1. *$gap(\mathcal{U},\mathcal{V}) = \max\{\|(I_n - P_{\mathcal{V}})P_{\mathcal{U}}\|_2, \|(I_n - P_{\mathcal{U}})P_{\mathcal{V}}\|_2\}$.*
2. *$gap(\mathcal{U},\mathcal{V}) = \|P_{\mathcal{U}} - P_{\mathcal{V}}\|_2$.*
3. *$gap(\mathcal{U},\mathcal{V}) = 1$ sempre que $dim(\mathcal{U}) \neq dim(\mathcal{V})$.*

Demonstração. Da definição de distância direta temos

$$\delta(\mathcal{U},\mathcal{V}) = \max_{\substack{u\in\mathcal{U}\\ \|u\|_2=1}} \|(I_n - P_\mathcal{V})u\|_2 = \max_{\substack{u\in\mathcal{U}\\ \|u\|_2\leq 1}} \|(I_n - P_\mathcal{V})u\|_2. \tag{B.11}$$

Como para $x \in \mathbb{R}^n$ com $\|x\|_2 = 1$ vale $P_\mathcal{U} x = u \in \mathcal{U}$ com $\|u\|_2 \leq 1$, segue que

$$\max_{\substack{u\in\mathcal{U}\\ \|u\|_2\leq 1}} \|(I_n - P_\mathcal{V})u\|_2 = \max_{\|x\|_2=1} \|(I_n - P_\mathcal{V})P_\mathcal{U} x\|_2 = \|(I_n - P_\mathcal{V})P_\mathcal{U}\|_2. \tag{B.12}$$

Com um argumento similar provamos que $\delta(\mathcal{V},\mathcal{U}) = \|(I_n - P_\mathcal{U})P_\mathcal{V}\|_2$ e com isto o primeiro item está demonstrado.

Sejam $U = [U_1\ U_2]$ e $V = [V_1\ V_2]$ matrizes ortogonais tais que as colunas de U_1 e U_2 formam, respectivamente, bases ortonormais para $\mathcal{U}$ e $\mathcal{U}^\perp$ e as colunas de V_1 e V_2 formam, respectivamente, bases ortonormais para $\mathcal{V}^\perp$ e $\mathcal{V}$. Sob estas condições segue que

$$\begin{aligned} P_\mathcal{U} &= U_1 U_1^T, \qquad I_n - P_\mathcal{U} = U_2 U_2^T, \\ P_\mathcal{V} &= V_2 V_2^T, \qquad I_n - P_\mathcal{V} = V_1 V_1^T. \end{aligned} \tag{B.13}$$

e, portanto,

$$\delta(\mathcal{U},\mathcal{V}) = \|U_1^T V_1\|_2, \quad \delta(\mathcal{V},\mathcal{U}) = \|U_2^T V_2\|_2. \tag{B.14}$$

Notemos agora que

$$\|P_\mathcal{U} - P_\mathcal{V}\|_2 = \|U^T(P_\mathcal{U} - P_\mathcal{V})V\|_2 = \left\| \begin{pmatrix} U_1^T V_1 & 0 \\ 0 & -U_2^T V_2 \end{pmatrix} \right\|. \tag{B.15}$$

Logo,

$$\|P_\mathcal{U} - P_\mathcal{V}\|_2 = \max\left\{\|U_1^T V_1\|_2, \|U_2^T V_2\|_2\right\} = \max\left\{\delta(\mathcal{U},\mathcal{V}), \delta(\mathcal{V},\mathcal{U})\right\} = \text{gap}(\mathcal{U},\mathcal{V}), \tag{B.16}$$

e o segundo item está demonstrado.

Para o terceito item, vamos supor que $\dim(\mathcal{U}) = r$ e $\dim(\mathcal{V}) = k$, em que $r < k$. Isto implica que $\mathcal{U}^\perp \cap \mathcal{V} \neq 0$, seja então $x \in \mathcal{U}^\perp \cap \mathcal{V} \neq 0$ tal que $\|x\|_2 = 1$. Consequentemente, $(I_n - P_\mathcal{U})x = x = P_\mathcal{V} x$, então $\|(I_n - P_\mathcal{U})P_\mathcal{V} x\|_2 = 1$, logo $\delta(\mathcal{V},\mathcal{U}) = 1$.

□

A consequência mais interessante deste lema é com relação à distância entre subespaços ser facilmente calculada através de projetores ortogonais (veja item 2). Para subespaços de dimensões diferentes vimos que o *gap* tem sempre o mesmo valor. O lema a seguir fornece alguns resultados para o caso de subespaços de mesma dimensão.

Lema B.8. *Sejam $\mathcal{U}, \mathcal{V} \subset \mathbb{R}^n$ subespaços não-nulos de mesma dimensão, então*

1. $\delta(\mathcal{U}, \mathcal{V}) = \delta(\mathcal{V}, \mathcal{U})$.
2. *se* $\mathcal{U}^\perp \cap \mathcal{V} \neq 0$ *(ou* $\mathcal{U} \cap \mathcal{V}^\perp \neq 0$*) então* $gap(\mathcal{U}, \mathcal{V}) = 1$.
3. *se* $\mathcal{U}^\perp \cap \mathcal{V} = 0$ *(ou* $\mathcal{U} \cap \mathcal{V}^\perp = 0$*) então* $gap(\mathcal{U}, \mathcal{V}) < 1$.

Demonstração. Vamos supor que $\dim(\mathcal{U}) = \dim(\mathcal{V}) = r$. Então, do teorema da dimensão da soma de subespaços (veja [102, pp. 205]), segue que

$$\begin{aligned} \dim(\mathcal{U}^\perp \cap \mathcal{V}) &= \dim(\mathcal{U}^\perp) + \dim(\mathcal{V}) - \dim(\mathcal{U}^\perp + \mathcal{V}) \\ &= (n - r) + r - \dim\left((\mathcal{U} \cap \mathcal{V}^\perp)^\perp\right) \\ &= \dim(\mathcal{U} \cap \mathcal{V}^\perp), \end{aligned} \tag{B.17}$$

em que usamos o fato de que $(\mathcal{U} \cap \mathcal{V}^\perp)^\perp = \mathcal{U}^\perp + \mathcal{V}$. Precisamos, então, analisar a dimensão do subespaço $\mathcal{U} \cap \mathcal{V}^\perp$.

Se $\dim(\mathcal{U} \cap \mathcal{V}^\perp) > 0$, então existem vetores $x \in \mathcal{U}^\perp \cap \mathcal{V}$ e $y \in \mathcal{U} \cap \mathcal{V}^\perp$ tais que $\|x\|_2 = \|y\|_2 = 1$. Logo, $\|(I_n - P_\mathcal{U})P_\mathcal{V} x\|_2 = \|x\|_2 = 1$ e $\|(I_n - P_\mathcal{V})P_\mathcal{U} y\|_2 = \|y\|_2 = 1$. Portanto,

$$\delta(\mathcal{V}, \mathcal{U}) = \|(I_n - P_\mathcal{U})P_\mathcal{V}\|_2 = 1 = \|(I_n - P_\mathcal{V})P_\mathcal{U}\|_2 = \delta(\mathcal{U}, \mathcal{V}). \tag{B.18}$$

Se $\dim(\mathcal{U} \cap \mathcal{V}^\perp) = 0$, então $U_2^T V_1$ é não singular pois é uma matriz $r \times r$ e tem posto r (basta lembrarmos que $\text{posto}(U_2^T V_1) = \text{posto}(V_1) - \dim(\mathcal{N}(U_2^T) \cap \mathcal{R}(V_1))$, assim

$$\begin{aligned} (\delta(\mathcal{U}, \mathcal{V}))^2 &= \|U_1^T V_1\|_2^2 = \|(I_n - U_2 U_2^T)V_1\|_2^2, \\ &= \max_{\|x\|_2 = 1} x^T V_1^T (I_n - U_2 U_2^T) V_1 x = \max_{\|x\|_2 = 1} \left(1 - \|U_2^T V_1 x\|_2^2\right) \\ &= 1 - \min_{\|x\|_2 = 1} \|U_2^T U_1 x\|_2^2 = 1 - \frac{1}{\|(U_2^T V_1)^{-1}\|_2^2} < 1. \end{aligned} \tag{B.19}$$

Logo $\delta(\mathcal{V},\mathcal{U}) = \delta(\mathcal{U},\mathcal{V}) < 1$.

□

Corolário B.9. *Sejam $\mathcal{U},\mathcal{V} \subset \mathbb{R}^n$ subespaços de mesma dimensão. Então*

$$gap(\mathcal{U},\mathcal{V}) = \|U_1^T V_1\|_2 = \|V_2^T U_2\|_2. \tag{B.20}$$

Demonstração. Segue imediatamente da equação (B.16). □

Inicialmente definimos o conceito de ângulo mínimo entre dois subespaços, agora veremos o ângulo máximo.

Definição B.10. *Sejam $\mathcal{U},\mathcal{V} \subset \mathbb{R}^n$ subespaços não-nulos. O ângulo maximo entre eles é o número $0 \leq \theta_{\max} \leq \pi/2$ tal que*

$$\operatorname{sen}(\theta_{\max}) = gap(\mathcal{U},\mathcal{V}) = \|P_{\mathcal{U}} - P_{\mathcal{V}}\|_2 \tag{B.21}$$

O ângulo mínimo, $\theta_{\min}$, e o ângulo máximo, $\theta_{\max}$, são dois dos chamados ângulos principais (ou ângulos canônicos).

Definição B.11. *Sejam $\mathcal{U},\mathcal{V} \subset \mathbb{R}^n$ subespaços não-nulos e $k = \min\{dim(\mathcal{U}), dim(\mathcal{V})\}$. Os ângulos principais são definidos por*

$$\cos(\theta_i) = \max_{u\in\mathcal{U}} \max_{v\in\mathcal{V}} u^T v = u_i^T v_i, \tag{B.22}$$

sujeitos a

$$\|u\|_2 = \|v\|_2 = 1, \tag{B.23}$$

$$u^T u_j = 0, \quad j = 1,\ldots,i-1, \tag{B.24}$$

$$v^T v_j = 0, \quad j = 1,\ldots,i-1. \tag{B.25}$$

Os vetores $\{u_1,\ldots,u_k\}$ e $\{v_1,\ldots,v_k\}$ são os vetores principais.

Apêndice C

Implementação de FP-LU para $\dim(\mathcal{N}(L)) = 1$

A eficácia do algoritmo FP-LU depende de como as multiplicações matriz-vetor $L^{\dagger}u$ e $L^{\dagger^T}v$, que estão associadas com o problema de minimização (2.54), são realizadas. Vamos discutir como o problema (2.54), mais especificamente como os subproblemas (2.64) podem ser resolvidos eficientemente através da decomposição LU e da fórmula de Sherman-Morrison-Woodbury

$$(\tilde{A} + \tilde{U}\tilde{V}^T)^{-1} = \tilde{A}^{-1} - \tilde{A}^{-1}\tilde{U}(I + \tilde{V}^T\tilde{A}^{-1}\tilde{U})^{-1}\tilde{V}^T\tilde{A}^{-1}, \tag{C.1}$$

em que $\tilde{A}, (I + \tilde{V}^T\tilde{A}^{-1}\tilde{U}) \in \mathbb{R}^{n\times n}$ são matrizes não-singulares e $\tilde{U}, \tilde{V} \in \mathbb{R}^{n\times k}$.

Neste caso temos posto$(L) = n - 1$ e, portanto, podemos encontrar matrizes $P_L \in \mathbb{R}^{p\times p}$ e $Q_L \in \mathbb{R}^{n\times n}$ tais que

$$P_L L Q_L = \hat{L}\hat{U}, \tag{C.2}$$

em que $\hat{L} \in \mathbb{R}^{p\times(n-1)}$ é "triangular inferior com diagonal unitária" e $\hat{U} \in \mathbb{R}^{(n-1)\times n}$ é "triangular superior", ambas tendo posto $n - 1$.

Veremos agora como os produtos matriz-vetor $\dot{u} = L^{\dagger}u$ e $\dot{v} = L^{\dagger^T}u$ podem ser eficientemente resolvidos.

Produto matriz-vetor $\dot{u} = L^{\dagger}u$

Dado um vetor $u \in \mathbb{R}^p$, vamos usar a decomposição LU (C.2) para realizar o produto matriz-vetor $\dot{u} = L^{\dagger}u$. O primeiro subproblema associado com $\dot{u} = L^{\dagger}u$ é

$$y_{\text{LS}} = \underset{y \in \mathbb{R}^{n-1}}{\operatorname{argmin}} \|\hat{L}y - P_L u\|_2, \tag{C.3}$$

e, como este tem posto coluna completo, existe única solução dada por

$$y_{\text{LS}} = (\hat{L}^T\hat{L})^{-1}(\hat{L}^T P_L u). \tag{C.4}$$

Para o segundo problema,

$$t_{\text{LS}} = \underset{t \in \mathbb{R}^n}{\operatorname{argmin}} \|\hat{U}t - y_{\text{LS}}\|_2, \tag{C.5}$$

notemos que precisamos apenas de uma solução qualquer, pois, pelo lema 2.6, temos que $\dot{u} = Q_L t_{\text{LS}} - WW^T(Q_L t_{\text{LS}})$.

Vamos particionar a matriz $\hat{U}$ em $\hat{U} = [\hat{U}_1 \ \hat{u}_1]$ com a matriz $\hat{U}_1 \in \mathbb{R}^{n-1\times n-1}$ triangular superior e não-singular, então consideramos como solução de (C.5) o vetor

$$t_{\text{LS}} = \begin{bmatrix} \hat{U}_1^{-1} y_{\text{LS}} \\ 0 \end{bmatrix}, \tag{C.6}$$

e finalmente $\dot{u} = Q_L t_{\text{LS}} - WW^T(Q_L t_{\text{LS}})$.

O produto matriz-vetor $\hat{U}_1^{-1} y_{\text{LS}}$ é um sistema triangular e este é resolvido rapidamente. Teceremos alguns comentários a respeito do sistema linear com a matriz $\hat{L}^T\hat{L}$ após a discussão sobre o produto matriz-vetor $\dot{v} = L^{\dagger T}v$.

Produto matriz-vetor $\dot{v} = L^{\dagger T}v$

Dado um vetor $v \in \mathbb{R}^n$, o produto matriz-vetor $\dot{v} = L^{\dagger T}v$ é equivalente a resolver

$$\dot{v} = \underset{t \in \mathbb{S}}{\operatorname{argmin}} \|t\|_2, \quad \mathbb{S} = \left\{ t \in \mathbb{R}^p \,\middle/\, t = \underset{t \in \mathbb{R}^p}{\operatorname{argmin}} \|v - L^T t\|_2 \right\}. \tag{C.7}$$

Com a decomposição LU da matriz de regularização L, $P_L L Q_L = \hat{L}\hat{U}$, temos que

$$\|v - L^T t\|_2^2 = \|Q_L^T v - Q_L^T L^T P_L^T P_L t\|_2^2 = \|Q_L^T v - \hat{U}^T \hat{L}^T P_L^T t\|_2^2, \tag{C.8}$$

e, assim, o primeiro subproblema associado com o produto matriz-vetor $\dot{v} = L^{\dagger^T} v$ é

$$y_{\text{LS}} = \underset{t \in \mathbb{R}^p}{\operatorname{argmin}} \|Q_L^T v - \hat{U}^T \hat{L}^T P_L^T t\|_2^2. \tag{C.9}$$

Fazendo a mudança de variável $\hat{v} = Q_L^T v$ e $\hat{t} = \hat{L}^T P_L^T t$, temos

$$y_{\text{LS}} = \underset{\hat{t} \in \mathbb{R}^p}{\operatorname{argmin}} \|\hat{v} - \hat{U}^T \hat{t}\|_2^2, \tag{C.10}$$

que tem única solução pois a matriz $\hat{U}^T$ tem posto coluna completo.

Vamos mostrar como usar a fórmula de fórmula de Sherman-Morrison-Woodbury (C.1) para resolver eficientemente este problema.

A solução do problema de minimização (C.10) é dada pelas equações normais

$$\hat{U}\hat{U}^T y_{\text{LS}} = \hat{U}\hat{v}. \tag{C.11}$$

Seja, novamente, a matriz $\hat{U}$ particionada em $\hat{U} = [\hat{U}_1 \ \hat{u}_1]$ com a matriz $\hat{U}_1$ triangular superior, então

$$[\hat{U}_1 \ \hat{u}] \begin{bmatrix} \hat{U}_1^T \\ \hat{u}^T \end{bmatrix} y_{\text{LS}} = [\hat{U}_1 \ \hat{u}]\hat{v} \equiv [\hat{U}_1 \ \hat{u}] \begin{bmatrix} \hat{v}_1 \\ \hat{v}_2 \end{bmatrix}. \tag{C.12}$$

Temos, portanto,

$$y_{\text{LS}} = (\hat{U}_1\hat{U}_1^T + \hat{u}\hat{u}^T)^{-1}(\hat{U}_1\hat{v}_1 + \hat{u}\hat{v}_2). \tag{C.13}$$

Pela fórmula de Sherman-Morrison-Woodbury (C.1) temos que

$$\begin{aligned} (\hat{U}_1\hat{U}_1^T + \hat{u}\hat{u}^T)^{-1} &= (\hat{U}_1\hat{U}_1^T)^{-1} - (\hat{U}_1\hat{U}_1^T)^{-1}\hat{u}(1 + \hat{u}^T(\hat{U}_1\hat{U}_1^T)^{-1}\hat{u})^{-1}\hat{u}^T\hat{U}_1\hat{U}_1^T)^{-1} \\ &= \hat{U}_1^{-T}\left[\hat{U}_1^{-1} - \beta\overline{vv}^T\hat{U}_1^{-1}\right], \end{aligned} \tag{C.14}$$

em que as variáveis $\overline{v}$, α e β são dadas por

$$\hat{U}_1^{-1}\hat{u} = \overline{v}, \quad \alpha = \overline{v}^T\overline{v}, \quad \beta = \frac{1}{1+\alpha}. \tag{C.15}$$

Notemos que o vetor $\overline{v}$ é extremamente simples de ser calculado pois a matriz $\hat{U}_1$ é triangular superior. Assim, segue que

$$\begin{aligned} y_{\text{LS}} &= \hat{U}_1^{-T}\left[\hat{U}_1^{-1} - \beta\overline{v}\overline{v}^T\hat{U}_1^{-1}\right]\left[\hat{U}_1\hat{v}_1 + \hat{u}\hat{v}_2\right] \\ &= \hat{U}_1^{-T}\left[\hat{v}_1 + \left(\hat{v}_2 - \beta\left(\overline{v}^T\hat{v}_1 + \hat{v}_2\alpha\right)\right)\overline{v}\right], \end{aligned} \tag{C.16}$$

e temos mais um sistema linear triangular para ser resolvido.

Resumindo, o subproblema (C.10) pode ser resolvido através de dois sistemas lineares triangulares - o primeiro para determinar $\overline{v}$ e o segundo para y_{LS}.

Agora, temos o sistema

$$\hat{L}^T P_L^T t = y_{\text{LS}}, \tag{C.17}$$

que tem mais colunas do que linhas. Este sistema pode ser resolvido pela abordagem descrita pelas equações (2.60)-(2.61). Assim, com $P_L^T t = \hat{t}_1$ e $\hat{t}_1 = \hat{L}\hat{t}_2$ para algum $\hat{t}_2$ temos

$$\hat{L}^T\hat{L}\hat{t}_2 = y_{\text{LS}}, \tag{C.18}$$

que tem única solução pois $\hat{L}^T\hat{L}$ é uma matriz quadrada e tem posto coluna completo. Então

$$\hat{t}_2 = (\hat{L}^T\hat{L})^{-1}y_{\text{LS}}, \tag{C.19}$$

$$\hat{t}_1 = \hat{L}\hat{t}_2, \tag{C.20}$$

$$\dot{v} = P_L^T\hat{t}_1. \tag{C.21}$$

Analisando as equações (C.4) e (C.19) temos que são resolvidos dois sistemas lineares com a matriz $\hat{L}^T\hat{L}$ em cada iteração, um no cálculo de $\dot{u}$ e um no cálculo de $\dot{v}$.

Apesar desta matriz ser quadrada e de posto completo, possivelmente esparsa e de banda, nossas experiências numéricas mostraram que realizar a decomposição LU da matriz $\hat{L}^T\hat{L}$ logo no inicio do algoritmo e resolver dois sistemas triangulares, um superior e um inferior,

se mostrou mais eficiente do que usar, por exemplo, a contra-barra "\" do Matlab em cada iteração.

Portanto, mostramos como a abordagem baseada na decomposição LU da matriz de regularização L pode ser atrativa. As mesmas ideias aqui desenvolvidas podem ser aplicadas para o caso em que $\dim(\mathcal{N}(L)) = \dot{k} > 1$, porém na equação (C.15) o cálculo de $\overline{v}$ envolverá a resolução de $\dot{k}$ sistemas lineares (o que significa que $\overline{v}$ é uma matriz), logo, α que antes era escalar será uma matriz $\dot{k} \times \dot{k}$ e $\beta = (I_{\dot{k}} + \alpha)^{-1}$ será uma matriz.

Apêndice D

Bidiagonalização simultânea do par matricial (A, L)

A bidiagonalização simultânea [84] de duas matrizes, digamos A e L, consiste num procedimento iterativo que, no passo k, existem matrizes $U_{k+1} \in \mathbb{R}^{m\times(k+1)}$ e $\widehat{U}_k \in \mathbb{R}^{p\times k}$ com colunas ortogonais, uma matriz $Z_k \in \mathbb{R}^{n\times k}$ e matrizes bidiagonais inferior e superior, $B_k \in \mathbb{R}^{(k+1)\times k}$ e $\bar{B}_k \in \mathbb{R}^{k\times k}$ tais que

$$AZ_k = U_{k+1}B_k, \tag{D.1}$$

$$LZ_k = \widehat{U}_k\bar{B}_k, \tag{D.2}$$

$$\beta_1 U_{k+1}e_1 = g = \beta_1 u_1. \tag{D.3}$$

O Algoritmo D.1 apresenta o algoritmo para a bidiagonalização simultânea de duas matrizes. Devemos notar que no passo 1 a fatoração QR da matriz $[A^T \;\; L^T]^T$ é realizada e, caso isto não seja feito, então o cálculo de

$$QQ^T \begin{bmatrix} u_{i+1} \\ 0 \end{bmatrix}, \tag{D.4}$$

é equivalente a resolução de um sistema linear envolvendo a matriz $[A^T \;\; L^T]^T$.

Independentemente de qual abordagem seja feita, o custo computacional é extremamente

grande, seja para calcular a QR de $[A^T \;\; L^T]^T$ ou, então, para resolver um sistema linear, em cada iteração, envolvendo a matriz $[A^T \;\; L^T]^T$.

Entrada: A, L, g
Saída: Matrizes U_{k+1}, $\hat{U}_k$, $\hat{V}_k$ e B_k, $\bar{B}_k$.

1. $\begin{bmatrix} A \\ L \end{bmatrix} = QR$

2. $\beta_1 u_1 = g, \qquad \alpha_1 \hat{v}_1 = QQ^T \begin{bmatrix} u_1 \\ 0 \end{bmatrix}, \qquad \hat{\alpha}_1 \hat{u}_1 = \hat{v}_1(m+1:m+p)$

3. **Para i=1,2,...**

$\beta_{i+1} u_{i+1} = \hat{v}_i(1:m) - \alpha_i u_i$

$\alpha_{i+1} \hat{v}_{i+1} = QQ^T \begin{bmatrix} u_{i+1} \\ 0 \end{bmatrix} - \beta_{i+1} \hat{v}_i$

$\hat{\beta}_i = (\alpha_{i+1}\beta_{i+1})/\hat{\alpha}_i$

$\hat{\alpha}_{i+1} \hat{u}_{i+1} = (-1)^i \hat{v}_{i+1}(m+1:m+p) - \hat{\beta}_i \hat{u}_i$

$B_{i,i} = \alpha_i$

$B_{i+1,i} = \beta_{i+1}$

$\hat{B}_{i,i} = \hat{\alpha}_i$

$\hat{B}_{i,i+1} = \hat{\beta}_i$

Fim para

4. $\bar{B}_k = \hat{B}\hat{P}, \quad P = \mathrm{diag}(1, -1, 1, -1, \ldots) \in \mathbb{R}^{k \times k}$

Algoritmo D.1: Processo da bidiagonalização simultânea.

Também em [84] um algoritmo que usa a bidiagonalização simultânea de duas matrizes foi proposto e ilustrado numericamente. Este algoritmo, que chamamos aqui de JBD (*Joint-Bidiagonalization*), determina soluções aproximadas definidas por

$$f_\lambda^{(k)} = \underset{f \in \mathcal{Z}_k}{\operatorname{argmin}} \left\{ \|g - Af\|_2^2 + \lambda^2 \|Lf\|_2^2 \right\}, \tag{D.5}$$

em que $\mathcal{Z}_k$ é um subespaço de Krylov gerado durante o processo JBD aplicado às matrizes $Q_A \in \mathbb{R}^{m \times n}$ e $Q_L \in \mathbb{R}^{p \times n}$, com $Q = [Q_A^T \;\; Q_L^T]^T$ gerada pela decomposição QR de $\widehat{A} = [A^T \; L^T]^T$. Assim, de modo análogo ao algoritmo LSQR, resolver o sistema restrito (D.5) é equivalente a resolver o problema irrestrito

$$f_\lambda^{(k)} = Z_k d_\lambda^{(k)}, \quad d_\lambda^{(k)} = \underset{d \in \mathbb{R}^k}{\operatorname{argmin}} \left\{ \|\beta_1 e_1 - B_k d\|_2^2 + \lambda^2 \|\bar{B}_k d\|_2^2 \right\}, \tag{D.6}$$

em que as colunas da matriz Z_k formam uma base para o subespaço $\mathcal{Z}_k$.

Apêndice E

Códigos fontes

Na tabela E.1 apresentamos um resumo dos códigos fontes usados neste trabalho. Todos os códigos estão em linguagem Matlab.

Regularização de Tikhonov na forma geral e regularização iterativa	
`ggkbfp_std_LU.m`	Extensão de GKB-FP baseada na fatoração LU.
`ggkbfp_std_kronLU.m`	Extensão de GKB-FP baseada na fatoração LU e $A = (A_1 \otimes A_2)$.
`ggkbfp_std_D.m`	Extensão de GKB-FP baseada na matriz L_D.
`ggkbfp_std_kronD.m`	Extensão de GKB-FP baseada na matriz L_D e $A = (A_1 \otimes A_2)$.
`plsqr_D.m`	Algoritmo P-LSQR baseado na matriz L_D.
`plsqr_kronD.m`	Algoritmo P-LSQR baseado na matriz L_D e $A = (A_1 \otimes A_2)$.
`projl.m`	Algoritmo PROJ-L.
`projl_kron.m`	Algoritmo PROJ-L e $A = (A_1 \otimes A_2)$.
`fp.m`	Algoritmo FP para seleção do parâmetro λ.
`fp_n.m`	Modificação do codigo `fp.m` para `projl.m` e `projl_kron.m`.
`fun_phi.m`	Função auxiliar para avaliar $\phi(\lambda)$.
`kronmult.m`	Função auxiliar para realizar o produto de Kronecker $(A \otimes B)x$.
`lambda_k.m`	Função auxiliar para atualizar $\lambda^{(k)*}$.
`pinv_lu.m`	Solução de mínimos quadrados via LU.
Regularização de Tikhonov com múltiplos parâmetros	
`sevfp.m`	Algoritmo MFP.
`projfp_sev2.m`	Algoritmo MFP para problemas de grande porte.
Rotinas usadas do pacote RegularizationTools [64]	
`cgsvd.m`	GSVD do par matricial (A, L).
`get_l.m`	Discretização de operadores diferenciais.
`tikhonov.m`	Método de regularização de Tikhonov.
`tsvd.m`	Método da TSVD.

Tabela E.1: Programas e rotinas.

```
function [x,lambda_new,k,mu,numfp,ll]=ggkbfp_std_LU(A,L,W,b,p,tol,reorth)
%
% Transf. to standard form using LU
%
% Usage
%
%  [x,reg_fp,k,mu]=ggkbfp_std_LU(A,L,W,b,p,tol,reorth)
%
% Input arguments:
%
%   A,L,b: the matrix A, the regularizator and the right hand side
%   W: basis for null space of the regularizator
%   p: initial dimension of the projected subspace
%   tol: tolerance
%   reorth: 0, GKB without reorthogonalization
%           1, GKB with reorthogonalization
%
% Output arguments:
%
%   x: the regularized solution
%   reg_fp: the regularization parameter
%   k: the dimension of the Krylov subspace (number of iterations)
%   mu: the parameter mu
%

[m,n]=size(L);

if m<n
    error('L must have more rows than columns')
end

[L_hat,U_hat,P,Q] = lu(L);
Laa = L_hat(:,1:n-1)'*L_hat(:,1:n-1);
[Lh,Uh]=lu(Laa);

%comp. of (AW)^+A , x_n and b_bar=b-Ax_n
awa = pinv(A*W);
x_n = W*(awa*b);
b = b-A*x_n;
awa = awa*A;%awa*A;

% Inicialization with p steps of GKB
beta(1) = norm(b);
u = b/beta(1);
%v = A'*u;
```

```
u_til = A'*u;
u_til = (u_til - awa'*(W'*u_til));
%%%v = L'\u_til;
v = pinv_lu(L_hat,U_hat,P,Q,W,n-1,u_til,{Lh,Uh},1);

alpha(1) = norm(v);
v = v/alpha(1);
k_max = 600;
if p>=k_max
    error('p must be smaller than k_max=1000')
end
if reorth==1
    UU=zeros(length(u),k_max+1);
    VV=zeros(length(v),k_max+1);
    UU(:,1)=u;
    VV(:,1)=v;
else
    VV=zeros(length(v),k_max+1);
    VV(:,1)=v;
end
for k=1:p
    %u = A*v - alpha(k)*u;
    %v_til = L\v;
    %v_til = v_til - W*(W'*v_til);
    v_til = pinv_lu(L_hat,U_hat,P,Q,W,n-1,v,{Lh,Uh},0);
    u = A*(v_til - W*(awa*v_til)) - alpha(k)*u;
    if reorth==1
        for j=1:k
            u=u - (UU(:,j)'*u)*UU(:,j);
        end
    end
    beta(k+1) = norm(u);
    u = u/beta(k+1);
    %v = A'*u - beta(k+1)*v;
    u_til = A'*u;
    u_til = (u_til - awa'*(W'*u_til));
    %%%v = L'\u_til - beta(k+1)*v;
    v = pinv_lu(L_hat,U_hat,P,Q,W,n-1,u_til,{Lh,Uh},1) - beta(k+1)*v;
    if reorth==1
        for j=1:k
            v = v - (VV(:,j)'*v)*VV(:,j);
        end
    end
    alpha(k+1) = norm(v);
    v = v/alpha(k+1);
```

```
    B(k,k)=alpha(k);
    B(k+1,k)=beta(k+1);
    if reorth==1
        UU(:,k+1) = u;  VV(:,k+1) = v;
    else
        VV(:,k+1) = v;
    end
end

% Look for the first fixed-point
[U2,s2,V2]=svd(B,0);s2=diag(s2);
[lambda_old,k_g,mu] = fp(U2,s2,[norm(b);zeros(p,1)]);
lambda_1=lambda_old; % the first fixed-point
for k=p+1:n-1
    % Perform one more step of GKB
    %u = A*v - alpha(k)*u;
    %v_til = L\v;
    %v_til = v_til - W*(W'*v_til);
    v_til = pinv_lu(L_hat,U_hat,P,Q,W,n-1,v,{Lh,Uh},0);
    u = A*(v_til - W*(awa*v_til)) - alpha(k)*u;
    if reorth==1
        for j=1:k
            u=u - (UU(:,j)'*u)*UU(:,j);
        end
    end
    beta(k+1) = norm(u);
    u = u/beta(k+1);
    %v = A'*u - beta(k+1)*v;
    u_til = A'*u;
    u_til = (u_til - awa'*(W'*u_til));
    %%%v = L'\u_til - beta(k+1)*v;
    v = pinv_lu(L_hat,U_hat,P,Q,W,n-1,u_til,{Lh,Uh},1) - beta(k+1)*v;
    if reorth==1
        for j=1:k
            v = v - (VV(:,j)'*v)*VV(:,j);
        end
    end
    alpha(k+1) = norm(v);
    v = v/alpha(k+1);
    if reorth==1
        UU(:,k+1) = u;  VV(:,k+1) = v;
    else
        VV(:,k+1) = v;
    end
    % Look for the next fixed-point
```

```
    [lambda_new,numfp(k-p+1),y] = lambda_k(alpha,beta,lambda_old,sqrt(tol),mu);
    ll(k) = lambda_new;
    % Verify stopping criteria
    if (abs(lambda_new-lambda_old)/lambda_old < tol) || ...
            (abs(lambda_new-lambda_old)/lambda_1< tol)
        break;
    end
    lambda_old=lambda_new;
end
%if reorth == 0
%    x=stdlsqr(A,L,W,b,k,lambda_new,0,tol);
%else
    x=VV(:,1:k)*y';
%end

% back transf.
%x = L\x;
%x = x - W*(W'*x);
x = pinv_lu(L_hat,U_hat,P,Q,W,n-1,x,{Lh,Uh},0);
x = (x - W*(awa*x)) + x_n;
```

```
function [x,lambda_new,k,mu,numfp,ll]=ggkbfp_std_kronLU(T,L,W,b,p,tol,reorth)
%
% Transf. to standard form using LU and A = T kron T
%
% Usage
%
%  [x,reg_fp,k,mu]=ggkbfp_std_kronLU(T,L,W,b,p,tol,reorth)
%
% Input arguments:
%
%   T,L,b: the matrix T, the regularizator and the right hand side
%   W: basis for null space of the regularizator
%   p: initial dimension of the projected subspace
%   tol: tolerance
%   reorth: 0, GKB without reorthogonalization
%           1, GKB with reorthogonalization
%
% Output arguments:
%
%   x: the regularized solution
%   reg_fp: the regularization parameter
%   k: the dimension of the Krylov subspace (number of iterations)
```

```
%   mu: the parameter mu
%

[m,n]=size(L);

if m<n
    error('L must have more rows than columns')
end

[L_hat,U_hat,P,Q] = lu(L);
Laa = L_hat(:,1:n-1)'*L_hat(:,1:n-1);
[Lh,Uh]=lu(Laa);

%comp. of (AW)^+A , x_n and b_bar=b-Ax_n
awa = pinv(kronmult({T,T},W));
x_n = W*(awa*b);
b = b-kronmult({T,T},x_n);
awa = kronmult({T,T},awa')';%awa*A;

% Inicialization with p steps of GKB
beta(1) = norm(b);
u = b/beta(1);
%v = A'*u;
u_til = kronmult({T,T},u);
u_til = (u_til - awa'*(W'*u_til));
%%%v = L'\u_til;
v = pinv_lu(L_hat,U_hat,P,Q,W,n-1,u_til,{Lh,Uh},1);

alpha(1) = norm(v);
v = v/alpha(1);
k_max = 600;
if p>=k_max
    error('p must be smaller than k_max=1000')
end
if reorth==1
    UU=zeros(length(u),k_max+1);
    VV=zeros(length(v),k_max+1);
    UU(:,1)=u;
    VV(:,1)=v;
else
    VV=zeros(length(v),k_max+1);
    VV(:,1)=v;
end
for k=1:p
    %u = A*v - alpha(k)*u;
```

```
    %v_til = L\v;
    %v_til = v_til - W*(W'*v_til);
    v_til = pinv_lu(L_hat,U_hat,P,Q,W,n-1,v,{Lh,Uh},0);
    u = kronmult({T,T},v_til - W*(awa*v_til)) - alpha(k)*u;
    if reorth==1
        for j=1:k
            u=u - (UU(:,j)'*u)*UU(:,j);
        end
    end
    beta(k+1) = norm(u);
    u = u/beta(k+1);
    %v = A'*u - beta(k+1)*v;
    u_til = kronmult({T,T},u);
    u_til = (u_til - awa'*(W'*u_til));
    %%%v = L'\u_til - beta(k+1)*v;
    v = pinv_lu(L_hat,U_hat,P,Q,W,n-1,u_til,{Lh,Uh},1) - beta(k+1)*v;
    if reorth==1
        for j=1:k
            v = v - (VV(:,j)'*v)*VV(:,j);
        end
    end
    alpha(k+1) = norm(v);
    v = v/alpha(k+1);
    B(k,k)=alpha(k);
    B(k+1,k)=beta(k+1);
    if reorth==1
        UU(:,k+1) = u;  VV(:,k+1) = v;
    else
        VV(:,k+1) = v;
    end
end

% Look for the first fixed-point
[U2,s2,V2]=svd(B,0);s2=diag(s2);
[lambda_old,k_g,mu] = fp(U2,s2,[norm(b);zeros(p,1)]);
lambda_1=lambda_old; % the first fixed-point
for k=p+1:n-1
    % Perform one more step of GKB
    %u = A*v - alpha(k)*u;
    %v_til = L\v;
    %v_til = v_til - W*(W'*v_til);
    v_til = pinv_lu(L_hat,U_hat,P,Q,W,n-1,v,{Lh,Uh},0);
    u = kronmult({T,T},v_til - W*(awa*v_til)) - alpha(k)*u;
    if reorth==1
        for j=1:k
```

```
            u=u - (UU(:,j)'*u)*UU(:,j);
        end
    end
    beta(k+1) = norm(u);
    u = u/beta(k+1);
    %v = A'*u - beta(k+1)*v;
    u_til = kronmult({T,T},u);
    u_til = (u_til - awa'*(W'*u_til));
    %%%v = L'\u_til - beta(k+1)*v;
    v = pinv_lu(L_hat,U_hat,P,Q,W,n-1,u_til,{Lh,Uh},1) - beta(k+1)*v;
    if reorth==1
        for j=1:k
            v = v - (VV(:,j)'*v)*VV(:,j);
        end
    end
    alpha(k+1) = norm(v);
    v = v/alpha(k+1);
    if reorth==1
        UU(:,k+1) = u;  VV(:,k+1) = v;
    else
        VV(:,k+1) = v;
    end
    % Look for the next fixed-point
    [lambda_new,numfp(k-p+1),y] = lambda_k(alpha,beta,lambda_old,sqrt(tol),mu);
    ll(k) = lambda_new;
    % Verify stopping criteria
    if (abs(lambda_new-lambda_old)/lambda_old < tol) || ...
            (abs(lambda_new-lambda_old)/lambda_1< tol)
        break;
    end
    lambda_old=lambda_new;
end
%if reorth == 0
%    x=stdlsqr(A,L,W,b,k,lambda_new,0,tol);
%else
    x=VV(:,1:k)*y';
%end

% back transf.
%x = L\x;
%x = x - W*(W'*x);
x = pinv_lu(L_hat,U_hat,P,Q,W,n-1,x,{Lh,Uh},0);
x = (x - W*(awa*x)) + x_n;
```

```
function [x,lambda_new,k,mu]=ggkbfp_std_D(A,L,W,b,p,tol,reorth)
%
% Transf. to standard form using the L_D approach
%
% Usage
%
%  [x,reg_fp,k,mu]=ggkbfp_std_D(A,L,W,b,p,tol,reorth)
%
% Input arguments:
%
%   A,L,b: the matrix A, the small regularizator and the right hand side
%   W: basis for null space of the regularizator
%   p: initial dimension of the projected subspace
%   tol: tolerance
%   reorth: 0, GKB without reorthogonalization
%           1, GKB with reorthogonalization
%
% Output arguments:
%
%   x: the regularized solution
%   reg_fp: the regularization parameter
%   k: the dimension of the Krylov subspace (number of iterations)
%   mu: the parameter mu
%

[m,n]=size(L);m=n;
[Uss,Sss,Vss]=svd(full(L));Sss=sparse(Sss);
D = zeros(n*n,1);
D = D+diag(sqrt(kron(Sss'*Sss,speye(n)) + kron(speye(n),Sss'*Sss)));
Dp = zeros(n*n,1);
%W = Dp;%%%
for i=1:n*n;
    if D(i) >0
        Dp(i)=1/D(i);
        %W(i) = 0;
    %else
        %W(i) = 1;
    end
end

%comp. of (AW)^+A , x_n and b_bar=b-Ax_n
awa = pinv(A*W);
x_n = W*(awa*b);
b = b-A*x_n;
awa = awa*A;%awa*A;
```

```
% Inicialization with p steps of GKB
beta(1) = norm(b);
u = b/beta(1);
%v = A'*u;
u_til = A'*u;
u_til = (u_til - awa'*(W'*u_til));
%%%v = L'\u_til;

v = Dp.*kronmult({Vss',Vss'},u_til);

alpha(1) = norm(v);
v = v/alpha(1);

k_max = 600;
if p>=k_max
    error('p must be smaller than k_max=1000')
end
if reorth==1
    UU=zeros(length(u),k_max+1);
    VV=zeros(length(v),k_max+1);
    UU(:,1)=u;
    VV(:,1)=v;
else
    VV=zeros(length(v),k_max+1);
    VV(:,1)=v;
end
for k=1:p
    %u = A*v - alpha(k)*u;
    v_til = kronmult({Vss,Vss},Dp.*v);
    u = A*(v_til - W*(awa*v_til)) - alpha(k)*u;
    if reorth==1
        for j=1:k
            u=u - (UU(:,j)'*u)*UU(:,j);
        end
    end
    beta(k+1) = norm(u);
    u = u/beta(k+1);
    %v = A'*u - beta(k+1)*v;
    u_til = A'*u;
    u_til = (u_til - awa'*(W'*u_til));
    %%%v = L'\u_til - beta(k+1)*v;
    v = Dp.*kronmult({Vss',Vss'},u_til) - beta(k+1)*v;
    if reorth==1
        for j=1:k
```

```
            v = v - (VV(:,j)'*v)*VV(:,j);
        end
    end
    alpha(k+1) = norm(v);
    v = v/alpha(k+1);
    B(k,k)=alpha(k);
    B(k+1,k)=beta(k+1);
    if reorth==1
        UU(:,k+1) = u;  VV(:,k+1) = v;
    else
        VV(:,k+1) = v;
    end
end

% Look for the first fixed-point
[U2,s2,V2]=svd(B,0);s2=diag(s2);
[lambda_old,k_g,mu] = fp(U2,s2,[norm(b);zeros(p,1)]);
lambda_1=lambda_old; % the first fixed-point
for k=p+1:k_max
    % Perform one more step of GKB
    %u = A*v - alpha(k)*u;
    v_til = kronmult({Vss,Vss},Dp.*v);
    u = A*(v_til - W*(awa*v_til)) - alpha(k)*u;
    if reorth==1
        for j=1:k
            u=u - (UU(:,j)'*u)*UU(:,j);
        end
    end
    beta(k+1) = norm(u);
    u = u/beta(k+1);
    %v = A'*u - beta(k+1)*v;
    u_til = A'*u;
    u_til = (u_til - awa'*(W'*u_til));
    %%%v = L'\u_til - beta(k+1)*v;
    v = Dp.*kronmult({Vss',Vss'},u_til) - beta(k+1)*v;
    if reorth==1
        for j=1:k
            v = v - (VV(:,j)'*v)*VV(:,j);
        end
    end
    alpha(k+1) = norm(v);
    v = v/alpha(k+1);
    if reorth==1
        UU(:,k+1) = u;  VV(:,k+1) = v;
    else
```

```
        VV(:,k+1) = v;
    end
    % Look for the next fixed-point
    [lambda_new,numfp(k-p+1),y] = lambda_k(alpha,beta,lambda_old,sqrt(tol),mu);
    ll(k) = lambda_new;
    % Verify stopping criteria
    if (abs(lambda_new-lambda_old)/lambda_old < tol) || ...
            (abs(lambda_new-lambda_old)/lambda_1< tol)
        break;
    end
    lambda_old=lambda_new;
end
%if reorth == 0
%    x=stdlsqr_kron(T,L,W,b,k,lambda_new,0,tol);
%else
    x=VV(:,1:k)*y';
%end

% back transf.
%x = L\x;
%x = x - W*(W'*x);
x = kronmult({Vss,Vss},Dp.*x);
x = (x - W*(awa*x)) + x_n;
```

```
function [x,lambda_new,k,mu]=ggkbfp_std_kronD(T,L,W,b,p,tol,reorth)
%
% Transf. to standard form using the L_D approach and A = T kron T
%
% Usage
%
%  [x,reg_fp,k,mu]=ggkbfp_std_kronD(T,L,W,b,p,tol,reorth)
%
% Input arguments:
%
%   T,L,b: the matrix T, the small regularizator and the right hand side
%   W: basis for null space of the regularizator
%   p: initial dimension of the projected subspace
%   tol: tolerance
%   reorth: 0, GKB without reorthogonalization
%           1, GKB with reorthogonalization
%
% Output arguments:
%
```

```
%   x: the regularized solution
%   reg_fp: the regularization parameter
%   k: the dimension of the Krylov subspace (number of iterations)
%   mu: the parameter mu
%

[m,n]=size(L);m=n;
[Uss,Sss,Vss]=svd(full(L));Sss=sparse(Sss);
D = zeros(n*n,1);
D = D+diag(sqrt(kron(Sss'*Sss,speye(n)) + kron(speye(n),Sss'*Sss)));
Dp = zeros(n*n,1);
W = Dp;%%%
for i=1:n*n;
    if D(i) >0
        Dp(i)=1/D(i);
        W(i) = 0;
    else
        W(i) = 1;
    end
end

TV = T*Vss;

%comp. of (AW)^+A , x_n and b_bar=b-Ax_n
awa = pinv(kronmult({TV,TV},W));
x_n = W*(awa*b);
b = b-kronmult({TV,TV},x_n);
awa = kronmult({TV',TV'},awa')';%awa*A;

% Inicialization with p steps of GKB
beta(1) = norm(b);
u = b/beta(1);
%v = A'*u;
u_til = kronmult({TV',TV'},u);
u_til = (u_til - awa'*(W'*u_til));
%%%v = L'\u_til;

v = Dp.*u_til;

alpha(1) = norm(v);
v = v/alpha(1);

k_max = 1000;
if p>=k_max
    error('p must be smaller than k_max=1000')
```

```
end
if reorth==1
    UU=zeros(n*n,k_max+1);
    VV=UU;
    UU(:,1)=u;
    VV(:,1)=v;
else
    VV=zeros(n*n,k_max+1);
    VV(:,1)=v;
end
for k=1:p
    %u = A*v - alpha(k)*u;
    v_til = Dp.*v;
    u = kronmult({TV,TV},v_til - W*(awa*v_til)) - alpha(k)*u;
    if reorth==1
        for j=1:k
            u=u - (UU(:,j)'*u)*UU(:,j);
        end
    end
    beta(k+1) = norm(u);
    u = u/beta(k+1);
    %v = A'*u - beta(k+1)*v;
    u_til = kronmult({TV',TV'},u);
    u_til = (u_til - awa'*(W'*u_til));
    %%%v = L'\u_til - beta(k+1)*v;
    v = Dp.*u_til - beta(k+1)*v;
    if reorth==1
        for j=1:k
            v = v - (VV(:,j)'*v)*VV(:,j);
        end
    end
    alpha(k+1) = norm(v);
    v = v/alpha(k+1);
    B(k,k)=alpha(k);
    B(k+1,k)=beta(k+1);
    if reorth==1
        UU(:,k+1) = u;  VV(:,k+1) = v;
    else
        VV(:,k+1) = v;
    end
end

% Look for the first fixed-point
[U2,s2,V2]=svd(B,0);s2=diag(s2);
[lambda_old,k_g,mu] = fp(U2,s2,[norm(b);zeros(p,1)]);
```

```
lambda_1=lambda_old; % the first fixed-point
for k=p+1:k_max
    % Perform one more step of GKB
    %u = A*v - alpha(k)*u;
    v_til = Dp.*v;
    u = kronmult({TV,TV},v_til - W*(awa*v_til)) - alpha(k)*u;
    if reorth==1
        for j=1:k
            u=u - (UU(:,j)'*u)*UU(:,j);
        end
    end
    beta(k+1) = norm(u);
    u = u/beta(k+1);
    %v = A'*u - beta(k+1)*v;
    u_til = kronmult({TV',TV'},u);
    u_til = (u_til - awa'*(W'*u_til));
    %%%v = L'\u_til - beta(k+1)*v;
    v = Dp.*u_til - beta(k+1)*v;
    if reorth==1
        for j=1:k
            v = v - (VV(:,j)'*v)*VV(:,j);
        end
    end
    alpha(k+1) = norm(v);
    v = v/alpha(k+1);
    if reorth==1
        UU(:,k+1) = u;  VV(:,k+1) = v;
    else
        VV(:,k+1) = v;
    end
    % Look for the next fixed-point
    [lambda_new,numfp(k-p+1),y] = lambda_k(alpha,beta,lambda_old,sqrt(tol),mu);
    ll(k) = lambda_new;
    % Verify stopping criteria
    if (abs(lambda_new-lambda_old)/lambda_old < tol) || ...
            (abs(lambda_new-lambda_old)/lambda_1< tol)
        break;
    end
    lambda_old=lambda_new;
end
%if reorth == 0
%    x=stdlsqr_kron(T,L,W,b,k,lambda_new,0,tol);
%else
    x=VV(:,1:k)*y';
%end
```

```
% back transf.
%x = L\x;
%x = x - W*(W'*x);
x = Dp.*x;
x = (x - W*(awa*x)) + x_n;

% change of variable
x = kronmult({Vss,Vss},x);
```

```
function [x,k,k_1,psi]=plsqr_D(A,L,W,b,p,tol,reorth)
%
% P-LSQR algorithm using the L_D approach
%
% Usage
%
%  [x,k]=plsqr_D(A,L,W,b,p,tol,reorth)
%
% Input arguments:
%
%   A,L,b: the matrix A, the small regularizator and the right hand side
%   W: basis for null space of the regularizator
%   p: initial dimension of the projected subspace
%   tol: tolerance
%   reorth: 0, GKB without reorthogonalization
%           1, GKB with reorthogonalization
%
% Output arguments:
%
%   x: the regularized solution
%   k: the dimension of the Krylov subspace
%

[m,n]=size(L);m=n;
[Uss,Sss,Vss]=svd(full(L));Sss=sparse(Sss);
D = zeros(n*n,1);
D = D+diag(sqrt(kron(Sss'*Sss,speye(n)) + kron(speye(n),Sss'*Sss)));
Dp = zeros(n*n,1);
for i=1:n*n;
    if D(i) >0
        Dp(i)=1/D(i);
        %W(i) = 0;
    %else
```

```
        %W(i) = 1;
    end
end

%comp. of (AW)^+A , x_n and b_bar=b-Ax_n
awa = pinv(A*W);
x_n = W*(awa*b);
b = b-A*x_n;
awa = awa*A;%awa*A;

% Inicialization with p steps of GKB
beta(1) = norm(b);
u = b/beta(1);
%v = A'*u;
u_til = A'*u;
u_til = (u_til - awa'*(W'*u_til));
%%%v = L'\u_til;

v = Dp.*kronmult({Vss',Vss'},u_til);

alpha(1) = norm(v);

k_max = 1000; k_1=0;

v = v/alpha(1);

if p>=k_max
    error('p must be smaller than k_max=1000')
end
if reorth==1
    UU=zeros(length(u),k_max+1);
    VV=zeros(length(v),k_max+1);
    UU(:,1)=u;
    VV(:,1)=v;
else
    VV=zeros(length(v),k_max+1);
    VV(:,1)=v;
end

w=v;
x=0;
phi_bar = beta;
rho_bar = alpha;
cc=0;
```

```
for k=1:k_max
    %u = A*v - alpha(k)*u;
    v_til = kronmult({Vss,Vss},Dp.*v);
    u = A*(v_til - W*(awa*v_til)) - alpha(k)*u;
    if reorth==1
        for j=1:k
            u=u - (UU(:,j)'*u)*UU(:,j);
        end
    end
    beta(k+1) = norm(u);
    u = u/beta(k+1);
    %v = A'*u - beta(k+1)*v;
    u_til = A'*u;
    u_til = (u_til - awa'*(W'*u_til));
    %%%v = L'\u_til - beta(k+1)*v;
    v = Dp.*kronmult({Vss',Vss'},u_til) - beta(k+1)*v;
    if reorth==1
        for j=1:k
            v = v - (VV(:,j)'*v)*VV(:,j);
        end
    end
    alpha(k+1) = norm(v);
    v = v/alpha(k+1);
    B(k,k)=alpha(k);
    B(k+1,k)=beta(k+1);
    if reorth==1
        UU(:,k+1) = u;  VV(:,k+1) = v;
    else
        VV(:,k+1) = v;
    end
    %
    % Givens, null Beta
    rho = sqrt(rho_bar*rho_bar + beta(k+1)*beta(k+1));
    c = rho_bar/rho;
    s = beta(k+1)/rho;
    %
    % update a few values
    theta = s*alpha(k+1);
    rho_bar = c*alpha(k+1);
    phi = c*phi_bar;
    phi_bar = -s*phi_bar;
    %
    % update solution
    x = x + (phi/rho)*w;
    psi(k) = norm(x) * abs(phi_bar);
```

```
    if k>=p && cc==0
        if psi(k-1)<=psi(k)
        else
            k_1 = k;
            psi_1 = abs(psi(k));
            cc=1;
        end
    elseif cc==1
        if psi(k-1)<=psi(k) || abs(psi(k)-psi(k-1))<=tol*psi_1
            break
        end
    end
    %
    %X(:,i)=x;
    w = VV(:,k+1) - (theta/rho)*w;
end

% back transf.
%x = L\x;
%x = x - W*(W'*x);
x = kronmult({Vss,Vss},Dp.*x);
x = (x - W*(awa*x)) + x_n;
```

```
function [x,k,k_1,psi]=plsqr_kronD(T,L,W,b,p,tol,reorth)
%
% P-LSQR algorithm using the L_D approach and A = T kron T
% also, a change of variables is done
%
% Usage
%
%  [x,k]=plsqr_kronD(T,L,W,b,p,tol,reorth)
%
% Input arguments:
%
%   T,L,b: the matrix T, the small regularizator and the right hand side
%   W: basis for null space of the regularizator
%   p: initial dimension of the projected subspace
%   tol: tolerance
%   reorth: 0, GKB without reorthogonalization
%           1, GKB with reorthogonalization
%
% Output arguments:
```

```
%
%   x: the regularized solution
%   k: the dimension of the Krylov subspace
%

[m,n]=size(L);m=n;
[Uss,Sss,Vss]=svd(full(L));Sss=sparse(Sss);
D = zeros(n*n,1);
D = D+diag(sqrt(kron(Sss'*Sss,speye(n)) + kron(speye(n),Sss'*Sss)));
Dp = zeros(n*n,1);
W = Dp;%%%
for i=1:n*n;
    if D(i) >0
        Dp(i)=1/D(i);
        W(i) = 0;
    else
        W(i) = 1;
    end
end

TV = T*Vss;

%comp. of (AW)^+A , x_n and b_bar=b-Ax_n
awa = pinv(kronmult({TV,TV},W));
x_n = W*(awa*b);
b = b-kronmult({TV,TV},x_n);
awa = kronmult({TV',TV'},awa')';%awa*A;

% Inicialization with p steps of GKB
beta(1) = norm(b);
u = b/beta(1);
%v = A'*u;
u_til = kronmult({TV',TV'},u);
u_til = (u_til - awa'*(W'*u_til));
%%%v = L'\u_til;

v = Dp.*u_til;

alpha(1) = norm(v);

k_max = 1000; k_1=0;

v = v/alpha(1);

if p>=k_max
```

```
    error('p must be smaller than k_max=1000')
end
if reorth==1
    UU=zeros(n*n,k_max+1);
    VV=UU;
    UU(:,1)=u;
    VV(:,1)=v;
else
    VV=zeros(n*n,k_max+1);
    VV(:,1)=v;
end

w=v;
x=0;
phi_bar = beta;
rho_bar = alpha;
cc=0;

for k=1:k_max
    %u = A*v - alpha(k)*u;
    v_til = Dp.*v;
    u = kronmult({TV,TV},(v_til - W*(awa*v_til))) - alpha(k)*u;
    if reorth==1
        for j=1:k
            u=u - (UU(:,j)'*u)*UU(:,j);
        end
    end
    beta(k+1) = norm(u);
    u = u/beta(k+1);
    %v = A'*u - beta(k+1)*v;
    u_til = kronmult({TV',TV'},u);
    u_til = (u_til - awa'*(W'*u_til));
    %%%v = L'\u_til - beta(k+1)*v;
    v = Dp.*u_til - beta(k+1)*v;
    if reorth==1
        for j=1:k
            v = v - (VV(:,j)'*v)*VV(:,j);
        end
    end
    alpha(k+1) = norm(v);
    v = v/alpha(k+1);
    B(k,k)=alpha(k);
    B(k+1,k)=beta(k+1);
    if reorth==1
        UU(:,k+1) = u;  VV(:,k+1) = v;
```

```
    else
        VV(:,k+1) = v;
    end
    %
    % Givens, null Beta
    rho = sqrt(rho_bar*rho_bar + beta(k+1)*beta(k+1));
    c = rho_bar/rho;
    s = beta(k+1)/rho;
    %
    % update a few values
    theta = s*alpha(k+1);
    rho_bar = c*alpha(k+1);
    phi = c*phi_bar;
    phi_bar = -s*phi_bar;
    %
    % update solution
    x = x + (phi/rho)*w;
    psi(k) = norm(x) * abs(phi_bar);

    if k>=p && cc==0
        if psi(k-1)<=psi(k)
        else
            k_1 = k;
            psi_1 = abs(psi(k));
            cc=1;
        end
    elseif cc==1
        if psi(k-1)<=psi(k) || abs(psi(k)-psi(k-1))<=tol*psi_1
            break
        end
    end
    %
    %X(:,i)=x;
    w = VV(:,k+1) - (theta/rho)*w;
end

% back transf.
%x = L\x;
%x = x - W*(W'*x);
x = Dp.*x;
x = (x - W*(awa*x)) + x_n;

% change variables
x = kronmult({Vss,Vss},x);
```

```
function [x,lambda_new,k,mu]=projl(A,L,b,p,tol,reorth)
%
% Proj-L algorithm
%
% Usage
%
%   [x,reg_fp,k,mu]=projl(A,L,b,p,tol,reorth)
%
% Input arguments:
%
%   A,L,b: the matrix, regularizator and the right hand side
%   p: initial dimension of the projected subspace
%   tol: tolerance
%   reorth: 0, GKB without reorthogonalization
%           1, GKB with reorthogonalization
%
% Output arguments:
%
%   x: the regularized solution
%   reg_fp: the regularization parameter
%   k: the dimension of the Krylov subspace
%   mu: the mu parameter.
%

[m,n]=size(A);

% Inicialization with p steps of GKB
beta(1) = norm(b);
u = b/beta(1);
v = A'*u;
alpha(1) = norm(v);
v = v/alpha(1);
k_max=600;
if p>=k_max
    error('p must be smaller than k_max=1000')
end
if reorth==1
    UU=zeros(length(u),k_max+1);
    VV=zeros(length(v),k_max+1);
    UU(:,1)=u;
    VV(:,1)=v;
else
    VV=zeros(length(v),k_max+1);
```

```
    VV(:,1)=v;
end
for k=1:p
    u = A*v - alpha(k)*u;
    if reorth==1
        for j=1:k
            u=u - (UU(:,j)'*u)*UU(:,j);
        end
    end
    beta(k+1) = norm(u);
    u = u/beta(k+1);
    v = A'*u - beta(k+1)*v;
    if reorth==1
        for j=1:k
            v = v - (VV(:,j)'*v)*VV(:,j);
        end
    end
    alpha(k+1) = norm(v);
    v = v/alpha(k+1);
    B(k,k)=alpha(k);
    B(k+1,k)=beta(k+1);
    if reorth==1
        UU(:,k+1) = u;  VV(:,k+1) = v;
    else
        VV(:,k+1) = v;
    end
end

%
[Qk,Rk]=qr(L*VV(:,1:p),0);

% Look for the first fixed-point
[U2,s2]=cgsvd(B,Rk);
[lambda_old,k_fp,mu] = fp(U2,s2,[norm(b);zeros(p,1)]);
lambda_1=lambda_old; % the first fixed-point
for k=p+1:k_max
    % Perform one more step of GKB
    u = A*v - alpha(k)*u;
    if reorth==1
        for j=1:k
            u=u - (UU(:,j)'*u)*UU(:,j);
        end
    end
    beta(k+1) = norm(u);
    u = u/beta(k+1);
```

```
    v = A'*u - beta(k+1)*v;
    if reorth==1
        for j=1:k
            v = v - (VV(:,j)'*v)*VV(:,j);
        end
    end
    alpha(k+1) = norm(v);
    v = v/alpha(k+1);
    if reorth==1
        UU(:,k+1) = u;  VV(:,k+1) = v;
    else
        VV(:,k+1)=v;
    end
    % Look for the next fixed-point
    B(k,k)=alpha(k);
    B(k+1,k)=beta(k+1);
    % Update QR
    %%%Qk(:,k) = L*VV(:,k);
    %%%for i=1:k-1
    %%%    Rk(i,k) = Qk(:,i)'*Qk(:,k);
    %%%    Qk(:,k) = Qk(:,k) - Rk(i,k)*Qk(:,i);
    %%%end
    %%%Rk(k,k) = norm(Qk(:,k));
    %%%Qk(:,k) = Qk(:,k)/Rk(k,k);
    nv = L*VV(:,k);
    Rk(1:k-1,k) = Qk'*nv;
    nq = nv-Qk*Rk(1:k-1,k);
    Rk(k,k)=norm(nq);
    Qk(:,k) = nq/Rk(k,k);
    [U2,s2,X2]=cgsvd(B,Rk);
    [lambda_new,k_g] = fp_n(U2,s2,[norm(b);zeros(k,1)],lambda_old,mu);
    % Verify stopping criteria
    if (abs(lambda_new-lambda_old)/lambda_old < tol) || ...
            (abs(lambda_new-lambda_old)/lambda_1< tol)|| ...
            (lambda_new - lambda_old > 0)
        break;
    end
    lambda_old=lambda_new;
end
if reorth == 0
    %x=lsqr_auto([A;lambda_new*L],[b;zeros(p1,1)],n-1,0,0,tol);
    y = tikhonov(U2,s2,X2,[norm(b);zeros(k,1)],lambda_new);
    x=VV(:,1:k)*y;
else
    y = tikhonov(U2,s2,X2,[norm(b);zeros(k,1)],lambda_new);
```

```
    x=VV(:,1:k)*y;
end
```

```
function [x,lambda_new,k,mu]=projl_kron(T,L,b,p,tol,reorth)
%
% Proj-L algorithm with  A = (T kron T)
%
% Usage
%
%  [x,lambda,iter,mu]=projfp_kron(T,L,b,p,tol,reorth);
%
% Input arguments:
%
%   T,L,b: the matrix T, regularizator and the right hand side
%   p: initial dimension of the projected subspace
%   tol: tolerance
%   reorth: 0, GKB without reorthogonalization
%           1, GKB with reorthogonalization
%
% Output arguments:
%
%   x: the regularized solution
%   reg_fp: the regularization parameter
%   k: the dimension of the Krylov subspace
%   mu: the mu parameter.
%

[m,n]=size(T); n=n*n;

% Inicialization with p steps of GKB
beta(1) = norm(b);
u = b/beta(1);
v = kronmult({T,T},u);
alpha(1) = norm(v);
v = v/alpha(1);
k_max=1000;
if p>=k_max
    error('p must be smaller than k_max=1000')
end
if reorth==1
    UU=zeros(n,k_max+1);
    VV=UU;
    UU(:,1)=u;
```

```
    VV(:,1)=v;
else
    VV=zeros(n,k_max+1);
    VV(:,1)=v;
end
for k=1:p
    u = kronmult({T,T},v) - alpha(k)*u;
    if reorth==1
        for j=1:k
            u=u - (UU(:,j)'*u)*UU(:,j);
        end
    end
    beta(k+1) = norm(u);
    u = u/beta(k+1);
    v = kronmult({T,T},u) - beta(k+1)*v;
    if reorth==1
        for j=1:k
            v = v - (VV(:,j)'*v)*VV(:,j);
        end
    end
    alpha(k+1) = norm(v);
    v = v/alpha(k+1);
    B(k,k)=alpha(k);
    B(k+1,k)=beta(k+1);
    if reorth==1
        UU(:,k+1) = u;  VV(:,k+1) = v;
    else
        VV(:,k+1) = v;
    end
end

%
[Qk,Rk]=qr(L*VV(:,1:p),0);

% Look for the first fixed-point
[U2,s2]=cgsvd(B,Rk);
[lambda_old,k_fp,mu] = fp(U2,s2,[norm(b);zeros(p,1)]);
lambda_1=lambda_old; % the first fixed-point
for k=p+1:k_max
    % Perform one more step of GKB
    u = kronmult({T,T},v) - alpha(k)*u;
    if reorth==1
        for j=1:k
            u=u - (UU(:,j)'*u)*UU(:,j);
        end
```

```
    end
    beta(k+1) = norm(u);
    u = u/beta(k+1);
    v = kronmult({T,T},u) - beta(k+1)*v;
    if reorth==1
        for j=1:k
            v = v - (VV(:,j)'*v)*VV(:,j);
        end
    end
    alpha(k+1) = norm(v);
    v = v/alpha(k+1);
    if reorth==1
        UU(:,k+1) = u;  VV(:,k+1) = v;
    else
        VV(:,k+1)=v;
    end
    % Look for the next fixed-point
    B(k,k)=alpha(k);
    B(k+1,k)=beta(k+1);
    % Update QR
    %%%Qk(:,k) = L*VV(:,k);
    %%%for i=1:k-1
    %%%    Rk(i,k) = Qk(:,i)'*Qk(:,k);
    %%%    Qk(:,k) = Qk(:,k) - Rk(i,k)*Qk(:,i);
    %%%end
    %%%Rk(k,k) = norm(Qk(:,k));
    %%%Qk(:,k) = Qk(:,k)/Rk(k,k);
    nv = L*VV(:,k);
    Rk(1:k-1,k) = Qk'*nv;
    nq = nv-Qk*Rk(1:k-1,k);
    Rk(k,k)=norm(nq);
    Qk(:,k) = nq/Rk(k,k);
    [U2,s2,X2]=cgsvd(B,Rk);
    [lambda_new,k_g] = fp_n(U2,s2,[norm(b);zeros(k,1)],lambda_old,mu);
    % Verify stopping criteria
    if (abs(lambda_new-lambda_old)/lambda_old < tol) || ...
            (abs(lambda_new-lambda_old)/lambda_1< tol)|| ...
            (lambda_new - lambda_old > 0)
        break;
    end
    lambda_old=lambda_new;
end
if reorth == 0
    %x=lsqr_auto([A;lambda_new*L],[b;zeros(p1,1)],n-1,0,0,tol);
    y = tikhonov(U2,s2,X2,[norm(b);zeros(k,1)],lambda_new);
```

```
    x=VV(:,1:k)*y;
else
    y = tikhonov(U2,s2,X2,[norm(b);zeros(k,1)],lambda_new);
    x=VV(:,1:k)*y;
end
```

```
function [reg_fp,k,mu] = fp(U,s,b,reg_old,mu,tol)
%FP Locate the fixed-point of the function phi.
%
% [reg_fp,k,mu] = fp(U,s,b,lambda_0,mu,tol)
% [reg_fp,k,mu] = fp(U,sm,b,lambda_0,mu,tol)
%
% Locates one convex fixed-point of the function phi, possible
% the largest one.
%
% The initial guess lambda_0 and the parameter mu (default mu=1) are optional.
% The iterations stop when the relative difference between two consecutives
% iterates are smaller than tol. Default is tol = 10^-6.
%

% Reference: F.S.V. Bazan, Fixed-point iterations in determining Tikhonov
% regularization parameter. Inverse Problems, vol. 24, 2008.

% L. S. Borges, April 14, 2012.

[Um,Un] = size(U);
[p,pn] = size(s);
if pn == 2
    s = s(:,1)./s(:,2);
end

if nargin==3
    reg_old = max(s);
    mu = 1;
    tol = 10^(-6);
elseif nargin==4
    mu = 1;
    tol = 10^(-6);
elseif nargin==5
    tol = 10^(-6);
elseif nargin < 3 || nargin > 6
    error('Invalid number of arguments.')
end
```

```
% Square of (gen) singular values
s2 = s.^2;

% Fourier coefficients
fc = U(:,1:p)'*b;

% Square of the size of the incompatible componenet of b that lies outside
% the column space of A
delta = norm(b-U*[fc;U(:,p+1:Un)'*b])^2;

% Square of Fourier coefficients
fc = abs(fc).^2;
k = 0;
if reg_old > max(s)/sqrt(mu) || nargin==3
    ratio = 1/(((max(s)/sqrt(mu))/(16*eps))^(1/(200-1)));
    reg0 = reg_old;
    ml = phi(fc,s2,delta,reg0,mu)/reg0; k=k+1;
    for i=1:200
        reg0 = ratio*reg0;
        mln = phi(fc,s2,delta,reg0,mu)/reg0; k=k+1;
        if mln < 1
            reg_old = reg0;
            break
        elseif mln < ml
            reg_old = reg0;
            ml = mln;
        end
    end
    if i==200
        mu = 0.99*(reg_old/phi(fc,s2,delta,reg_old,mu))^2; k=k+1;
    end
end

for j=1:200
    reg_fp = phi(fc,s2,delta,reg_old,mu); k=k+1;

    if reg_fp > max(s)/sqrt(mu)
        ratio = 1/(((max(s)/sqrt(mu))/(16*eps))^(1/(200-1)));
        reg0 = reg_fp;
        ml = phi(fc,s2,delta,reg0,mu)/reg0; k=k+1;
        for i=1:200
            reg0 = ratio*reg0;
            mln = phi(fc,s2,delta,reg0,mu)/reg0; k=k+1;
            if mln < 1
```

```
                reg_fp = reg0;
                break
            elseif mln < ml
                reg_fp = reg0;
                ml = mln;
            end
        end
        if i==200
            mu = 0.99*(reg_fp/phi(fc,s2,delta,reg_fp,mu))^2; k=k+1;
        end
    end

    if abs(reg_fp - reg_old) < reg_old*tol
        return
    else
        reg_old = reg_fp;
    end
end
return

function val = phi(fc,s2,delta,lambda,mu)
v = fc./((s2+lambda^2).^2);
x = (lambda^4)*sum(v) + delta;
y = sum(s2.*v);
val = sqrt(mu*x/y);
return
```

```
function [reg_fp,k] = fp_n(u,sigma,b,reg_old,mu)
%
% Modification of fp.m used by projl.m and projl_kron.m


% size of u
[ur,uc] = size(u);

% if the GSVD is used, then sigma is changed to a column vector
% containing the generalized singular values
[tm1,tm2] = size(sigma);
if tm2==2
    sigma = sigma(:,1)./sigma(:,2);
end

% Tolerance, minimum and maximum value of lambda
```

```
tol = 0.001;
l_min = 16*eps;
l_max = max(sigma);

% Constant mu
if nargin<5
    mu=1;
end

% squared (generalized) singular values
sigma2 = sigma.*sigma;

% squared Fourier coeficients
utb = u(:,1:tm1)'*b;
alpha = utb.*utb;

% size of the incompatible component of b that lies outside the
% column space of U and squared size.
del0 = norm(b - u*[utb;u(:,tm1+1:uc)'*b]);
del02 = del0*del0;

% max number of iteration
k_max = 100;

% max number of global-iteration
k = 0;

for k=1:k_max
    reg_fp = fun_phi(alpha,sigma2,del02,reg_old,mu);
    if abs(reg_fp - reg_old) <= tol*reg_old
        break
    end
    reg_old = reg_fp;
end
```

```
function phi = fun_phi(alpha,sigma2,del02,lambda,mu)
%
% This auxiliar function calculates \phi(lambda)
%
% Usage:
%
%   phi = fun_phi(alpha,sigma2,del02,lambda,mu)
%
```

```
% Input arguments:
%
%   alpha: the squared Fourier coeficients
%   sigma2: the squared (generalized) singular values
%   del02: the squared size of incompatible part of b
%   lambda: the variable to evaluate phi
%   mu: the constant of the function phi.
%
% Output arguments:
%
%   phi: the function \phi evaluated at lambda
%


% squared lambda
lambda2 = lambda*lambda;

% common quantities for residual and (semi)norm
s2l2 = sigma2 + lambda2; % s^2 + lambda^2
common = (s2l2).*(s2l2); % (s^2 + lambda^2)^2
common = alpha./common; % alpha/(s^2 + lambda^2)^2

% residual and (semi)norm
rho = (lambda2*lambda2)*sum(common) + del02;
eta = sum(sigma2.*common);

% division of rho/eta
phi = sqrt(mu*rho/eta);
```

```
function X = kronmult(Q, X)
n = sqrt(length(X));

X1 = Q{2}*(reshape(X,n,n)*Q{1}');

X = X1(:);
```

```
function [lambda_star,numfp,y] = lambda_k(alpha,beta,lambda,tol,mu)
%
% This auxiliar function evaluates de function
% \phi^{(k)}(\lambda) of the projected problem
%
% Usage:
```

```
%
%  [lambda_star,numfp,y] = lambda_k(alpha,beta,lambda,tol,mu)
%
% Input arguments:
%
%   alpha,beta: the diagonal and subdiagonal of the matrix B_k.
%   lambda: the initial guess
%   tol: tolerance.
%   mu: the constant for the function \phi_mu^{(k)}
%
% Output arguments:
%
%   lambda_star: the regularization parameter
%   numfp: the number of evaluations of the function \phi
%   y: the solution to the projected problem.

k=length(alpha)-1;
numfp=0;
exec=1;
ct = 0; % a counter
while exec==1 && ct <= 100
    lambda2=lambda*lambda;
    phi_bar = beta(1);
    rho_bar = alpha(1);
    ng2=0;
    for i=1:k-1
%
% Givens rotation
        rho(i) = sqrt(rho_bar*rho_bar + lambda2);
        c = rho_bar/rho(i);
        s = lambda/rho(i);
%
% Update some quantities
        rho_bar = rho(i);
        ng2 = ng2+(s*phi_bar)*(s*phi_bar);
        phi_bar = c*phi_bar;
%
% Givens rotation
        rho(i) = sqrt(rho_bar*rho_bar + beta(i+1)*beta(i+1));
        c = rho_bar/rho(i);
        s = beta(i+1)/rho(i);
%
% Update some quantities
        theta(i) = s*alpha(i+1);
        rho_bar = c*alpha(i+1);
```

```
        phi(i) = c*phi_bar;
        phi_bar = -s*phi_bar;
    end
% When i=k:
% Givens rotation
    rho(k) = sqrt(rho_bar*rho_bar + lambda2);
    c = rho_bar/rho(k);
    s = lambda/rho(k);
%
% Update some quantities
    rho_bar = rho(k);
    ng2 = ng2+(s*phi_bar)*(s*phi_bar);
    phi_bar = c*phi_bar;
%
% Givens rotation
    rho(k) = sqrt(rho_bar*rho_bar + beta(k+1)*beta(k+1));
    c = rho_bar/rho(k);
    s = beta(k+1)/rho(k);
%
% Update some quantities
    phi(k) = c*phi_bar;
    phi_bar = -s*phi_bar;

% Solve for y:
    y(k)=phi(k)/rho(k);
    for i=k-1:-1:1
        y(i)=(phi(i)-theta(i)*y(i+1))/rho(i);
    end
%
% solution norm
    soln = norm(y);
%
% residual norm
    resn = sqrt(phi_bar*phi_bar + ng2 - lambda2*soln*soln);
    lambda_old=lambda;
    lambda=sqrt(mu)*resn/soln;
    numfp=numfp+1;
    if abs((lambda_old-lambda)/lambda_old) < tol
        exec=0;
        lambda_star=lambda;
    end
    ct = ct + 1;
end
```

```
function x = pinv_lu(L,U,P,Q,W,k,b,Laa,t)
if t==0
    tre = Laa{2}\(Laa{1}\(L(:,1:k)'*(P*b)));
    tty = Q(:,1:k)*(U(1:k,1:k)\tre);
    x = tty-W*(W'*tty);
else
    %aux = (U(1:k,:)')\(Q'*b);
    ld = Q'*b;
    v = U(1:end-1,1:end-1)\U(1:end-1,end);
    alpha = v'*v;
    beta = 1/(1+alpha);
    aux=U(1:end-1,1:end-1)'\(ld(1:end-1) + (ld(end)-(beta*(v'*ld(1:end-1)+...
        ld(end)*alpha)))*v);
    x = P'*(L(:,1:k)*(Laa{2}\(Laa{1}\aux)));
end
```

```
function [x,lambda,k]=sevfp(A,L,b,tol,lambda)
%
% MFP algorithm
%
% Usage
%
%   [x,lambda,k]=sevfp(A,L,b,p,tol,reorth)
%
% Input arguments:
%
%   A,L,b: the matrix, regularizator and the right hand side
%   p: initial dimension of the projected subspace
%   tol: tolerance
%   reorth: 0, GKB without reorthogonalization
%           1, GKB with reorthogonalization
%
% OBS: L must be a cell array, e.g., L = {L1;L2}
%
% Output arguments:
%
%   x: the regularized solution
%   lambda: the regularization parameter
%   k: the dimension of the Krylov subspace
%

q = length(L); p = zeros(q,1); mu = ones(q,1);
for i=1:q
```

```
    p(i) = size(L{i},1); % numero de linhas de cada regul.
    sm = cgsvd(A,L{i});
    gam(i,1) = max(sm(:,1)./sm(:,2));
end
[m,n] = size(A);
G = max(gam);

if nargin == 4
    lambda(:,1) = 10^(-3)*ones(q,1);
end

Lstack = L;
for i=1:q
    Lstack{i} = lambda(i)*L{i};
end
[U,s,V]=svd([A;cell2mat(Lstack)],0);s=diag(s);
x=tsvd(U,s,V,[b;zeros(sum(p),1)],n);
rho = norm(b-A*x);

for i=1:q
    ph(i,1) = rho/norm(L{i}*x);
    ss(i) = ph(i,1)/lambda(i,1);
end
sk(1) = min(ss);

quyt=0;
for k=1:200

    for i=1:q
        lambda(i,k+1) = sqrt(mu(i))*ph(i,k);
        if lambda(i,k+1) > G
            quyt = 1;
            break
        end
    end
    if quyt == 1;
        break
    end

    Lstack = L;
    for i=1:q
        Lstack{i} = lambda(i,k+1)*L{i};
    end
    [U,s,V]=svd([A;cell2mat(Lstack)],0);s=diag(s);
    x=tsvd(U,s,V,[b;zeros(sum(p),1)],n);
```

```
    rho = norm(b-A*x);

    for i=1:q
        ph(i,k+1) = rho/norm(L{i}*x);
        ss(i) = ph(i,k+1)/lambda(i,k+1);
    end
    sk(k+1) = min(ss);

    if norm(lambda(:,k+1) - lambda(:,k)) < norm(lambda(:,k))*tol || ...
        norm(lambda(:,k+1) - lambda(:,k)) < norm(lambda(:,1))*tol
        break
    end
end
return

% ajuste no mu
[val,kstar] = min(sk(1:k));
for j=1:q
    if lambda(j,k) > G
        mu(j) = 0.9*(lambda(j,kstar)/ph(j,kstar))^2;
    end
end

lambda(:,1) = lambda(:,kstar);

Lstack = L;
for i=1:q
    Lstack{i} = lambda(i)*L{i};
end
[U,s,V]=svd([A;cell2mat(Lstack)],0);s=diag(s);
x=tsvd(U,s,V,[b;zeros(sum(p),1)],n);
rho = norm(b-A*x);

for i=1:q
    ph(i,1) = rho/norm(L{i}*x);
    ss(i) = ph(i,1)/lambda(i,1);
end
sk(1) = min(ss);

for k=1:200
    for i=1:q
        lambda(i,k+1) = sqrt(mu(i))*ph(i,k);
    end

    Lstack = L;
```

```
    for i=1:q
        Lstack{i} = lambda(i,k+1)*L{i};
    end
    [U,s,V]=svd([A;cell2mat(Lstack)],0);s=diag(s);
    x=tsvd(U,s,V,[b;zeros(sum(p),1)],n);
    rho = norm(b-A*x);

    for i=1:q
        ph(i,k+1) = rho/norm(L{i}*x);
        ss(i) = ph(i,k+1)/lambda(i,k+1);
    end
    sk(k+1) = min(ss);

    if norm(lambda(:,k+1) - lambda(:,k)) < norm(lambda(:,k))*tol ||...
        norm(lambda(:,k+1) - lambda(:,k)) < norm(lambda(:,1))*tol
        break
    end
end
```

```
function [x,lambda_new,k]=projfp_sev2(A,L,b,p0,tol,reorth)
%
% MFP algorithm for large-scale problems
%
% Usage
%
%   [x,reg_fp,k]=projfp_sev2(A,L,b,p,tol,reorth)
%
% Input arguments:
%
%   A,L,b: the matrix, regularizator and the right hand side
%   p: initial dimension of the projected subspace
%   tol: tolerance
%   reorth: 0, GKB without reorthogonalization
%           1, GKB with reorthogonalization
%
% OBS: L must be a cell array, e.g., L = {L1;L2}
%
% Output arguments:
%
%   x: the regularized solution
%   reg_fp: the regularization parameter
%   k: the dimension of the Krylov subspace
%
```

```
q = length(L); p = zeros(q,1); lambda = p;
for i=1:q
    p(i) = size(L{i},1); % numero de linhas de cada regul.
end
[m,n] = size(A);

% Inicialization with p steps of GKB
beta(1) = norm(b);
u = b/beta(1);
v = A'*u;
alpha(1) = norm(v);
v = v/alpha(1);
k_max=1000;
if p0>=k_max
    error('p must be smaller than k_max=1000')
end
if reorth==1
    UU=zeros(length(u),k_max+1);
    VV=zeros(length(v),k_max+1);
    UU(:,1)=u;
    VV(:,1)=v;
else
    VV=zeros(length(v),k_max+1);
    VV(:,1)=v;
end
for k=1:p0
    u = A*v - alpha(k)*u;
    if reorth==1
        for j=1:k
            u=u - (UU(:,j)'*u)*UU(:,j);
        end
    end
    beta(k+1) = norm(u);
    u = u/beta(k+1);
    v = A'*u - beta(k+1)*v;
    if reorth==1
        for j=1:k
            v = v - (VV(:,j)'*v)*VV(:,j);
        end
    end
    alpha(k+1) = norm(v);
    v = v/alpha(k+1);
    B(k,k)=alpha(k);
    B(k+1,k)=beta(k+1);
```

```
    if reorth==1
        UU(:,k+1) = u;  VV(:,k+1) = v;
    else
        VV(:,k+1) = v;
    end
end

% calculate QR of each L{i}*V
for i=1:q
    [Q{i,1},R{i,1}]=qr(L{i}*VV(:,1:k),0);
end

% Look for the first fixed-point
[x,lambda,iter,mu]=sevfp(B,R,[norm(b);zeros(p0,1)],tol/100,.0001*ones(q,1));
lambda_old = lambda(:,iter);
%%%[x,lambda]=sevconvex(B,R,[norm(b);zeros(p0,1)]);
%%%lambda_old = sqrt(lambda);
lambda_1=lambda_old; % the first fixed-point
for k=p0+1:k_max
    % Perform one more step of GKB
    u = A*v - alpha(k)*u;
    if reorth==1
        for j=1:k
            u=u - (UU(:,j)'*u)*UU(:,j);
        end
    end
    beta(k+1) = norm(u);
    u = u/beta(k+1);
    v = A'*u - beta(k+1)*v;
    if reorth==1
        for j=1:k
            v = v - (VV(:,j)'*v)*VV(:,j);
        end
    end
    alpha(k+1) = norm(v);
    v = v/alpha(k+1);
    if reorth==1
        UU(:,k+1) = u;  VV(:,k+1) = v;
    else
        VV(:,k+1)=v;
    end
    % Look for the next fixed-point
    B(k,k)=alpha(k);
    B(k+1,k)=beta(k+1);
    % Update QR
```

```
    %%%Qk(:,k) = L*VV(:,k);
    %%%for i=1:k-1
    %%%    Rk(i,k) = Qk(:,i)'*Qk(:,k);
    %%%    Qk(:,k) = Qk(:,k) - Rk(i,k)*Qk(:,i);
    %%%end
    %%%Rk(k,k) = norm(Qk(:,k));
    %%%Qk(:,k) = Qk(:,k)/Rk(k,k);
    for i=1:q
        nv = L{i}*VV(:,k);
        R{i}(1:k-1,k) = Q{i}'*nv;
        nq = nv-Q{i}*R{i}(1:k-1,k);
        R{i}(k,k)=norm(nq);
        Q{i}(:,k) = nq/R{i}(k,k);
    end
    [x,lambda,iter]=sevfp(B,R,[norm(b);zeros(k,1)],tol/100,lambda_old);
    lambda_new = lambda(:,iter);
    % Verify stopping criteria
    if (norm(lambda_new-lambda_old)/norm(lambda_old) < tol) || ...
            (norm(lambda_new-lambda_old)/norm(lambda_1)< tol)%|| ...
            %(lambda_new - lambda_old > 0)
        break;
    end
    lambda_old=lambda_new;
end
if reorth == 0
%    %x=lsqr_auto([A;lambda_new*L],[b;zeros(p1,1)],n-1,0,0,tol);
%    y = tikhonov(U2,s2,X2,[norm(b);zeros(k,1)],lambda_new);
%    x=VV(:,1:k)*y;
else
    MR = 0;
    for i=1:q
        MR = MR + lambda_new(i)^2*(R{i}'*R{i});
    end
    y = (B'*B + MR)\(B'*[norm(b);zeros(k,1)]);
    x=VV(:,1:k)*y;
end
```

```
function [U,sm,X,V,W] = cgsvd(A,L)
%CGSVD Compact generalized SVD of a matrix pair in regularization problems.
%
% sm = cgsvd(A,L)
% [U,sm,X,V] = cgsvd(A,L) ,  sm = [sigma,mu]
% [U,sm,X,V,W] = cgsvd(A,L) ,  sm = [sigma,mu]
```

```
%
% Computes the generalized SVD of the matrix pair (A,L). The dimensions of
% A and L must be such that [A;L] does not have fewer rows than columns.
%
% If m >= n >= p then the GSVD has the form:
%    [ A ] = [ U  0 ]*[ diag(sigma)      0    ]*inv(X)
%    [ L ]   [ 0  V ] [      0       eye(n-p) ]
%                     [  diag(mu)        0    ]
% where
%    U  is  m-by-n ,    sigma  is  p-by-1
%    V  is  p-by-p ,    mu     is  p-by-1
%    X  is  n-by-n .
%
% Otherwise the GSVD has a more complicated form (see manual for details).
%
% A possible fifth output argument returns W = inv(X).

% Reference: C. F. Van Loan, "Computing the CS and the generalized
% singular value decomposition", Numer. Math. 46 (1985), 479-491.

% Per Christian Hansen, IMM, March 17, 2008.

% Initialization.
[m,n] = size(A); [p,n1] = size(L);
if (n1 ~= n)
  error('No. columns in A and L must be the same')
end
if (m+p < n)
  error('Dimensions must satisfy m+p >= n')
end

% Call Matlab's GSVD routine.
[U,V,W,C,S] = gsvd(full(A),full(L),0);

if (m >= n)
  % The overdetermined or square case.
  sm = [diag(C(1:p,1:p)),diag(S(1:p,1:p))];
  if (nargout < 2)
    U = sm;
  else
    % Full decomposition.
    X = inv(W');
  end
else
  % The underdetermined case.
```

```
  sm = [diag(C(1:m+p-n,n-m+1:p)),diag(S(n-m+1:p,n-m+1:p))];
  if (nargout < 2)
    U = sm;
  else
    % Full decomposition.
    X = inv(W');
    X = X(:,n-m+1:n);
  end
end

if (nargout==5), W = W'; end
```

```
function [L,W] = get_l(n,d)
%GET_L Compute discrete derivative operators.
%
% [L,W] = get_l(n,d)
%
% Computes the discrete approximation L to the derivative operator
% of order d on a regular grid with n points, i.e. L is (n-d)-by-n.
%
% L is stored as a sparse matrix.
%
% Also computes W, an orthonormal basis for the null space of L.

% Per Christian Hansen, IMM, 02/05/98.

% Initialization.
if (d<0), error ('Order d must be nonnegative'), end

% Zero'th derivative.
if (d==0), L = speye(n); W = zeros(n,0); return, end

% Compute L.
c = [-1,1,zeros(1,d-1)];
nd = n-d;
for i=2:d, c = [0,c(1:d)] - [c(1:d),0]; end
L = sparse(nd,n);
for i=1:d+1
  L = L + sparse(1:nd,(1:nd)+i-1,c(i)*ones(1,nd),nd,n);
end

% If required, compute the null vectors W via modified Gram-Schmidt.
if (nargout==2)
```

```
  W = zeros(n,d);
  W(:,1) = ones(n,1);
  for i=2:d, W(:,i) = W(:,i-1).*(1:n)'; end
  for k=1:d
     W(:,k) = W(:,k)/norm(W(:,k));
     W(:,k+1:d) = W(:,k+1:d) - W(:,k)*(W(:,k)'*W(:,k+1:d));
  end
end
```

```
function [x_lambda,rho,eta] = tikhonov(U,s,V,b,lambda,x_0)
%TIKHONOV Tikhonov regularization.
%
% [x_lambda,rho,eta] = tikhonov(U,s,V,b,lambda,x_0)
% [x_lambda,rho,eta] = tikhonov(U,sm,X,b,lambda,x_0) ,  sm = [sigma,mu]
%
% Computes the Tikhonov regularized solution x_lambda.  If the SVD
% is used, i.e. if U, s, and V are specified, then standard-form
% regularization is applied:
%    min { || A x - b ||^2 + lambda^2 || x - x_0 ||^2 } .
% If, on the other hand, the GSVD is used, i.e. if U, sm, and X are
% specified, then general-form regularization is applied:
%    min { || A x - b ||^2 + lambda^2 || L (x - x_0) ||^2 } .
%
% If x_0 is not specified, then x_0 = 0 is used
%
% Note that x_0 cannot be used if A is underdetermined and L ~= I.
%
% If lambda is a vector, then x_lambda is a matrix such that
%    x_lambda = [ x_lambda(1), x_lambda(2), ... ] .
%
% The solution norm (standard-form case) or seminorm (general-form
% case) and the residual norm are returned in eta and rho.

% Per Christian Hansen, IMM, April 14, 2003.

% Reference: A. N. Tikhonov & V. Y. Arsenin, "Solutions of
% Ill-Posed Problems", Wiley, 1977.

% Initialization.
if (min(lambda)<0)
  error('Illegal regularization parameter lambda')
end
m = size(U,1);
```

```
n = size(V,1);
[p,ps] = size(s);
beta = U(:,1:p)'*b;
zeta = s(:,1).*beta;
ll = length(lambda); x_lambda = zeros(n,ll);
rho = zeros(ll,1); eta = zeros(ll,1);

% Treat each lambda separately.
if (ps==1)

  % The standard-form case.
  if (nargin==6), omega = V'*x_0; end
  for i=1:ll
    if (nargin==5)
      x_lambda(:,i) = V(:,1:p)*(zeta./(s.^2 + lambda(i)^2));
      rho(i) = lambda(i)^2*norm(beta./(s.^2 + lambda(i)^2));
    else
      x_lambda(:,i) = V(:,1:p)*...
        ((zeta + lambda(i)^2*omega)./(s.^2 + lambda(i)^2));
      rho(i) = lambda(i)^2*norm((beta - s.*omega)./(s.^2 + lambda(i)^2));
    end
    eta(i) = norm(x_lambda(:,i));
  end
  if (nargout > 1 & size(U,1) > p)
    rho = sqrt(rho.^2 + norm(b - U(:,1:n)*[beta;U(:,p+1:n)'*b])^2);
  end

elseif (m>=n)

  % The overdetermined or square general-form case.
  gamma2 = (s(:,1)./s(:,2)).^2;
  if (nargin==6), omega = V\x_0; omega = omega(1:p); end
  if (p==n)
    x0 = zeros(n,1);
  else
    x0 = V(:,p+1:n)*U(:,p+1:n)'*b;
  end
  for i=1:ll
    if (nargin==5)
      xi = zeta./(s(:,1).^2 + lambda(i)^2*s(:,2).^2);
      x_lambda(:,i) = V(:,1:p)*xi + x0;
      rho(i) = lambda(i)^2*norm(beta./(gamma2 + lambda(i)^2));
    else
      xi = (zeta + lambda(i)^2*(s(:,2).^2).*omega)./...
           (s(:,1).^2 + lambda(i)^2*s(:,2).^2);
```

```
      x_lambda(:,i) = V(:,1:p)*xi + x0;
      rho(i) = lambda(i)^2*norm((beta - s(:,1).*omega)./...
               (gamma2 + lambda(i)^2));
    end
    eta(i) = norm(s(:,2).*xi);
  end
  if (nargout > 1 & size(U,1) > p)
    rho = sqrt(rho.^2 + norm(b - U(:,1:n)*[beta;U(:,p+1:n)'*b])^2);
  end

else

  % The underdetermined general-form case.
  gamma2 = (s(:,1)./s(:,2)).^2;
  if (nargin==6), error('x_0 not allowed'), end
  if (p==m)
    x0 = zeros(n,1);
  else
    x0 = V(:,p+1:m)*U(:,p+1:m)'*b;
  end
  for i=1:ll
    xi = zeta./(s(:,1).^2 + lambda(i)^2*s(:,2).^2);
    x_lambda(:,i) = V(:,1:p)*xi + x0;
    rho(i) = lambda(i)^2*norm(beta./(gamma2 + lambda(i)^2));
    eta(i) = norm(s(:,2).*xi);
  end

end
```

```
function [x_k,rho,eta] = tsvd(U,s,V,b,k)
%TSVD Truncated SVD regularization.
%
% [x_k,rho,eta] = tsvd(U,s,V,b,k)
%
% Computes the truncated SVD solution
%    x_k = V(:,1:k)*inv(diag(s(1:k)))*U(:,1:k)'*b .
% If k is a vector, then x_k is a matrix such that
%    x_k = [ x_k(1), x_k(2), ... ] .
%
% The solution and residual norms are returned in eta and rho.

% Per Christian Hansen, IMM, 12/21/97.
```

```
% Initialization.
[n,p] = size(V); lk = length(k);
if (min(k)<0 | max(k)>p)
  error('Illegal truncation parameter k')
end
x_k = zeros(n,lk);
eta = zeros(lk,1); rho = zeros(lk,1);
beta = U(:,1:p)'*b;
xi = beta./s;

% Treat each k separately.
for j=1:lk
  i = k(j);
  if (i>0)
    x_k(:,j) = V(:,1:i)*xi(1:i);
    eta(j) = norm(xi(1:i));
    rho(j) = norm(beta(i+1:p));
  end
end

if (nargout > 1 & size(U,1) > p)
  rho = sqrt(rho.^2 + norm(b - U(:,1:p)*beta)^2);
end
```

Referências Bibliográficas

[1] O. M. Alifanov e S. V. Rumjancev, *On the stability of iterative methods for the solution of linear ill-posed problems*, Soviet Mathematics Doklady, volume 20, pp. 1133-1136, 1979.

[2] W. E. Arnoldi, *The principle of minimized iteration in the solution of the matrix eigenvalue problem*, Quarterly of Applied Mathematics, volume 9, pp. 17-29, 1951.

[3] R. C. Aster, B. Borchers e C. H. Thurber, *Parameter estimation and inverse problem*, Elsevier Academic Press, 2005.

[4] A. B. Bakushinski, *Remarks on choosing a regularization parameter using quasi-optimality and ratio criterion*, USSR Computational Mathematics and Mathematical Physics, volume 24, número 4, pp. 181-182, 1984.

[5] R. Barakat e G. Newsam, *Remote sensing of the refractive index structure parameter via inversion of Tatarski's integral equation for both spherical and plane wave situations*, Radio Science, volume 19, número 4, pp. 1041-1056, 1984.

[6] F. Bauer e O. Ivanyshyn, *Optimal regularization with two interdependent regularization parameters*, Inverse Problems, volume 23, número 1, pp. 331-342, 2007.

[7] F. Bauer e M. A. Lukas, *Comparing parameter choice methods for regularization of ill-posed problems*, Mathematics and Computers in Simulation, volume 81, número 9, pp 1795-1841, 2011.

[8] F. Bauer e S. V. Pereverzev, *An utilization of a rough approximation of a noise covariance within the framework of multi-parameter regularization*, International Journal of Tomography and Statistics, volume 4, pp. 1-12, 2006.

[9] F. S. V. Bazán, *Fixed-point iterations in determining the Tikhonov regularization parameter*, Inverse Problems, volume. 24, número 3, doi:10.1088/0266-5611/24/3/035001, 2008.

[10] F. S. V. Bazán e L. S. Borges, *GKB-FP: an algorithm for large-scale discrete ill-posed problems*, BIT, volume 50, número 3, pp. 481-507, 2010.

[11] F. S. V. Bazán, L. S. Borges e J. B. Francisco, *On a generalization of Regińska's parameter choice rule and its numerical realization in large-scale multi-parameter Tikhonov regularization*, Applied Mathematics and Computation, volume 219, número 4, pp. 2100-2113, 2012.

[12] F. S. V. Bazán, M. C. C. Cunha e L. S. Borges, *Extension of GKB-FP algorithm to large-scale general-form Tikhonov regularization*, Numerical Linear Algebra with Applications, aceito.

[13] F. S. V. Bazán e J. B. Francisco, *Improved fixed-point algorithm for determining the Tikhonov regularization parameter*, Inverse Problems, volume 25, número 4, doi:10.1088/0266-5611/25/4/045007, 2009.

[14] M. Belge, M. E. Kilmer e E. L. Miller, *Simultaneous multiple regularization parameter selection by means of the L-hypersurface with applications to linear inverse problems posed in the wavelet domain*, Proc. SPIE - Bayesian Inference for Inverse Problems, volume 3459, 1998.

[15] M. Belge, M. E. Kilmer e E. L. Miller, *Efficient determination of multiple regularization parameters in a generalized L-curve framework*, Inverse Problems, volume 18, número 4, pp. 1161-1185, 2002.

[16] M. Benzi, *Gianfranco Cimmino's contributions to numerical mathematics*, Atti del Seminario di Analisi Matematica, Dipartimento di Matematica dell'Universita' di Bologna. Volume Speciale: Ciclo di Conferenze in Memoria di Gianfranco Cimmino, Marzo-Aprile 2004, Tecnoprint, Bologna, pp. 87-109, 2005.

[17] M. Bertero, C. De Mol e E. R. Pike, *Applied inverse problems in optics*, em *inverse and ill-posed problems*, Eds.: H. W. Engl e C. W. Groetsch, Academic press, Inc. London, pp. 291-313, 1987.

[18] Å. Björck, *A bidiagonalization algorithm for solving large and sparse ill-posed systems of linear equations*, BIT, volume 28, número 3, pp. 659-670, 1988.

[19] Å. Björck, *Numerical methods for least squares problems*, SIAM, Philadelphia, 1996.

[20] L. S. Borges, *Lanc-FP: um algoritmo para problemas discretos mal-postos de grande porte*, Florianópolis: UFSC, 2009, 112 p. Dissertação (Mestrado) - Programa de Pós-Graduação em Matemática e Computação Científica, Universidade Federal de Santa Catarina, Florianópolis, 2009.

[21] C. Brezinski, M. Redivo-Zaglia, G. Rodriguez e S. Seatzu, *Multi-parameter regularization techniques for ill-conditioned linear systems*, Numerische Mathematik, volume 94, número 2, pp. 203-228, 2003.

[22] A. Bunse-Gerstner, V. Guerra-Ones e H. Madrid de la Vega, *An improved preconditioned LSQR for discrete ill-posed problems*, Mathematics and Computers in Simulation, volume 73, número 1, pp. 65-75, 2006.

[23] I. Bushuyev, *Stability of recovering the near-field wave from the scattering amplitude*, Inverse Problems, volume 12, número 6, pp. 859-868, 1996.

[24] D. Calvetti, G. H. Golub e L. Reichel, *Estimation of the L-curve via Lanczos bidiagonalization*, BIT, volume 39, número 4, pp. 603-619, 1999.

[25] D. Calvetti, B. Lewis e L. Reichel, *GMRES-type methods for inconsistent systems*, Linear Algebra and its Applications, volume 316, número 1, pp. 157-169, 2000.

[26] D. Calvetti e L. Reichel, *Tikhonov regularization of large linear problems*, BIT, volume 43, número 2, pp. 263-283, 2003.

[27] D. Calvetti, L. Reichel e A. Shuibi, *L-curve and curvature bounds for Tikhonov regularization*, Numerical Algorithms, volume 35, número 2-4, pp. 301-314, 2004.

[28] D. Calvetti, L. Reichel e A. Shuibi, *Invertible smoothing preconditioners for linear discrete ill-posed problems*, Applied Numerical Mathematics, volume 54, número 2, pp. 135-149, 2005.

[29] A. S. Carasso, *Determining surface temperatures from interior observations*, SIAM Journal on Applied Mathematics, volume 42, número 3, pp. 558-574, 1982.

[30] J. J. Castellanos, S. Gómez e V. Guerra, *The triangle method for finding the corner of the L-curve*, Applied Numerical Mathematics, volume 43, número 4, pp. 359-373, 2002.

[31] G. Chavent, *Nonlinear least squares for inverse problems: theoretical foundations and step-by-step guide for applications*, Springer, 2nd edition, 2009.

[32] Z. Chen, Y. Lu, Y. Xu e H. Yang, *Multi-parameter Tikhonov regularization for linear ill-posed operator equations*, Journal of Computational Mathematics, volume 26, número 1, pp. 37-55, 2008.

[33] J. Cheng, J. J. Liu e G. Nakamura, *The numerical realization of the probe method for the inverse scattering problems from the near field data*, Inverse Problems, volume 21, número 3, pp. 839-855, 2005.

[34] J. Chung, J. G. Nagy e D. P. O'Leary, *A weighted-GCV method for Lanczos-hybrid regularization*, Electronic Transaction on Numerical Analysis, volume 28, pp 149-167, 2008.

[35] G. Cimmino, *Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari*, La Ricerca Scientifica, volume 16, número 9, pp. 326-333, 1938.

[36] D. L. Colton e R. Kress, *Inverse acoustic and electromagnetic scattering theory*, Springer-Verlag, Berlin, 1991.

[37] I. J. D. Craig e J. C. Brown, *Inverse problems in astronomy*, Adam Hilger, Bristol, UK, 1986.

[38] J. J. M. Cuppen, *A numerical solution of the inverse problem of electrocardiography*, Ph.D. Thesis, Dept. of Mathematics, University of Amsterdam, 1983.

[39] I. Daubechies, *Ten lectures on wavelets*, SIAM, Philadelphia, 1992.

[40] A. Dold e B. Eckmann, *Lecture notes in mathematics*, Inverse Problems, 1986.

[41] D. Düvelmeyer e B. Hofmann, *A multi-parameter regularization approach for estimating parameters in jump diffusion processes*, Journal of Inverse and Ill-Posed Problems, volume 14, número 9, pp. 861-880, 2006.

[42] L. Eldén, *A weighted pseudoinverse, generalized singular values, and constrained least square problems*, BIT, volume 22, número 4, pp. 487-502, 1982.

[43] H. W. Engl, M. Hanke e A. Neubauer, *Regularization of inverse problems*, vol. 375 of Mathematics and its Applications. Kluwer Academic Publishers Group, Dordrecht, 1996.

[44] R. D. Fierro e J. R. Bunch, *Bounding the subspaces from rank revealing two-sided orthogonal decomposition*, SIAM Journal on Matrix Analysis and Applications, volume 16, número 3, pp. 743-759, 1995.

[45] D. C. Fong e M. Saunders, *LSMR: an iterative algorithm for sparse least-squares problems*, SIAM Journal on Scientific Computing, volume 33, número 5, pp. 2950-2971, 2011.

[46] S. Gazzola e P. Novati, *Automatic parameter setting for Arnoldi-Tikhonov methods*, submetido, 2012.

[47] S. Gazzola e P. Novati, *Multi-parameter Arnoldi-Tikhonov methods*, submetido, 2012.

[48] G. H. Golub, *Numerical methods for solving linear least squares problems*, Numerische Mathematik, volume 7, número 3, pp. 206-216, 1965.

[49] G. H. Golub, M. Heath e G. Wahba, *Generalized cross-validation as a method for choosing a good ridge parameter*, Technometrics, volume 21, número 2, pp. 215-223, 1979.

[50] G. H. Golub e W. Kahan, *Calculating the singular values and pseudo-inverse of a matrix*, SIAM Journal on Numerical Analysis, volume 2, número 2, pp. 205-224, 1965.

[51] G. H. Golub e C. F. Van Loan, *Matrix computations*, 3ª ed., Johns Hopkins University Press, London, 1996.

[52] C. W. Groetsch, *The theory of Tikhonov regularization for Fredholm equation of the first kind*, Research Notes in Mathematics, 105, Pitman, Boston, 1984.

[53] J. Hadamard, *Sur les problèmes aux dérivées partielles et leur signification physique*, Princeton University Bulletin, pp. 49-52, 1902.

[54] U. Hämarik e R. Palm, *Comparison of stopping rules in conjugate gradient type methods for solving ill-posed problems*, em: MMA2005 Proceedings: 10-th International Conference Mathematical Modelling and Analysis & 2nd International Conference Computational Methods in Applied Mathematics; Trakai, Lithuania: Technika, pp. 285 - 291, 2005.

[55] U. Hämarik, R. Palm e T. Raus, *Use of extrapolation in regularization methods*, Journal of Inverse and III-posed Problems, volume 15, número 3, pp. 277-294, 2007.

[56] U. Hämarik e T. Raus, *On the a posteriori parameter choice in regularization methods*, Proceedings of the Estonian Academy of Sciences Physics Mathematics, volume 48, número 2, pp. 133-145, 1999.

[57] U. Hämarik e T. Raus, *On the choice of the regularization parameter in ill-posed problems with approximately given noise level of data*, Journal of Inverse and III-posed Problems, volume 14, número 3, pp. 251-266, 2006.

[58] M. Hanke, *Conjugate gradient type methods for ill-posed problems*, Longman, Harlow, UK, 1995.

[59] M. Hanke, *Limitations of the L-curve method in ill-posed problemas*, BIT, volume 36, número 2, pp. 287-301, 1996.

[60] M. Hanke e P. C. Hansen, *Regularization methods for large-scale problems*, Surveys on Mathematics for Industry, volume 3, número 4, pp. 253-315, 1993.

[61] P. C. Hansen, *The truncated SVD as a method for regularization*, BIT, volume 27, número 4, pp. 534-553, 1987.

[62] P. C. Hansen, *Regularization, GSVD and truncated GSVD*, BIT, volume 29, número 3, pp. 491-504, 1989.

[63] P. C. Hansen, *The discrete Picard condition for discrete ill-posed problems*, BIT, volume 30, número 4, pp. 658-672, 1990.

[64] P. C. Hansen, *Regularization tools: a MATLAB package for analysis and solution of discrete ill-posed problems*, Numerical Algorithms, volume 6, número 1, pp. 1-35, 1994.

[65] P. C. Hansen, *Rank-deficient and discrete ill-posed problems*, SIAM, Philadelphia, 1998.

[66] P. C. Hansen, *The L-curve and its use in the numerical treatment of inverse problems*, em Computational Inverse Problems in Electrocardiology, ed. P. Johnston, Advances in Computational Bioengineering, WIT Press, pp. 119-142, 2000.

[67] P. C. Hansen, *Deconvolution and regularization with Toeplitz matrices*, Numerical Algorithms, volume 29, número 4, pp. 323-378, 2002.

[68] P. C. Hansen, *Discrete inverse problems: insight and algorithms*, SIAM, Philadelphia, 2010.

[69] P. C. Hansen e T. K. Jensen, *Smoothing-norm preconditioning for regularizing minimum-residual methods*, SIAM Journal on Matrix Analysis and Applications, volume 29, número 1, pp. 1-14, 2007.

[70] P. C. Hanen e T. K. Jensen, *Noise propagation in regularizing iterations for image deblurring*, Electronic Transactions on Numerical Analysis volume 31, pp. 204-220, 2008.

[71] P. C. Hansen, T. K. Jensen e G. Rodriguez, *An adaptive pruning algorithm for the discrete L-curve criterion*, Journal of Computational and Applied Mathematics, volume 198, número 2, pp. 483-492, 2007.

[72] P. C. Hansen, J. G. Nagy e D. P. O'Leary, *Deblurring images: matrices, spectra, and filtering*, SIAM, Philadelphia, 2006.

[73] P. C. Hansen e D. P. O'Leary, *The use of the L-curve in the regularization of discrete ill-posed problems*, SIAM Journal on Scientific Computing, volume 14, número 6, pp. 1487-1503, 1993.

[74] R. F. Harrington, *Field computations by moment methods*, Macmillan, New York, 1968.

[75] M. R. Hestenes e E. Stiefel, *Methods of conjugate gradients for solving linear systems*, Journal of Research of the National Bureau of Standards, volume 49, número 6, pp. 409-436, 1952.

[76] M. E. Hochstenback e L. Reichel, *An iterative method for Tikhonov regularization with a general linear regularization operator*, Journal of Integral Equations and Applications, volume 22, número 3, pp. 465-482, 2010.

[77] R. A. Horn e C. R. Johnson, *Topics in matrix analysis*, Cambridge University Press, 1994.

[78] M. Jacobsen, P. C. Hansen e M. A. Saunders, *Subspace preconditioned LSQR for discrete ill-posed problems*, BIT, volume 43, número 5, pp. 975-989, 2003.

[79] T. K. Jensen e P. C. Hansen, *Iterative regularization with minimum-residual methods*, BIT, volume 47, número 1, pp. 103-120, 2007.

[80] M. Jiang, L. Xia e G. Shou, *Combining regularization frameworks for solving the electrocardiography inverse problem*, Communications in Computer and Information Science, volume 2, pp. 1210-1219, 2007.

[81] M. Jiang, L. Xia, G. Shou e M. Tang, *Combination of the LSQR method and a genetic algorithm for solving the electrocardiography inverse problem*, Physics in Medicine and Biology, volume 52, número 5, pp. 1277-1294, 2007.

[82] M. Jiang, L. Xia, G. Shou, F. Liu e S. Crozier, *Two hybrid regularization frameworks for solving the electrocardiography inverse problem*, Physics in Medicine and Biology, volume 53, número 18, pp. 5151-5164, 2008.

[83] P. R. Johnston e R. M. Gulrajani, *An analysis of the zero-crossing method for choosing regularization parameter*, SIAM Journal on Scientific Computing, volume 24, número 2, pp. 428-442, 2002.

[84] M. E. Kilmer, P. C. Hansen e M. I. Español, *A projection-based approach to general-form Tikhonov regularization*, SIAM Journal on Scientific Computing, volume 29, número 1, pp. 315-330, 2007.

[85] M. E. Kilmer e D. P. O'Leary, *Choosing regularization parameters in iterative methods for ill-posed problems*, SIAM Journal on Matrix Analysis and Application, volume 22, número 4, pp. 1204-1221, 2001.

[86] S. Kindermann, *Convergence analysis of minimization-based noise level-free parameter choice rules for linear ill-posed problems*, Electronic Transactions on Numerical Analysis, volume 38, pp. 233-257, 2011.

[87] A. Kirsch, *An introduction to the mathematical theory of inverse problems*, Springer, New York, 1996.

[88] D. Krawczyk-Stańdo e M. Rudnicki, *Regularization parameter selection in discrete ill-posed problems - the use of the U-curve*, International Journal of Applied Mathematics and Computer Science, volume 17, número 2, pp. 157-164, 2007.

[89] J. Lampe, L. Reichel e H. Voss, *Large-scale Tikhonov regularization via reduction by orthogonal projection*, Linear Algebra and its Applications, volume 436, número 8, pp. 2845-2865, 2012.

[90] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, Journal of research of the National Bureau of Standards volume 45, número 4, pp. 255-282, 1950.

[91] L. Landweber, *An iteration formula for Fredholm integral equations of the first kind*, American Journal of Mathematics, volume 73, número 3, pp. 615-624, 1951.

[92] A. J. Laub, *Matrix analysis for scientists and engineers*, SIAM, Philadelphia, 2004.

[93] O. Lepskij, *On a problem of adaptive estimation in Gaussian white noise*, Theory of Probability and Its Applications, volume 35, número 3, pp.454-466, 1990.

[94] J. J. Liu, J. Cheng e G. Nakamura, *Reconstruction of scattered field from far-field by regularization*, Journal of Computational Mathematics, volume 22, número 3, pp. 389-402, 2004.

[95] J. Liu e M. Ni, *A model function method for determining the regularizing parameter in potential approach for the recovery of scattered wave*, Applied Numerical Mathematics, volume 58, número 8, pp. 1113-1128, 2008.

[96] S. Lu e S. V. Pereverzev, *Multi-parameter regularization and its numerical realization*, Numerische Mathematik, volume 118, número 1, pp. 1-31, 2011.

[97] S. Lu, S. V. Pereverzev, Y. Shao e U. Tautenhahn, *Discrepancy curves for multi-parameter regularization*, Journal of Inverse and III-posed Problems, volume 18, número 6, pp. 655-676, 2010.

[98] Y. Lu, L. Shen e Y. Xu, *Multi-parameter regularization methods for high-resolution image reconstruction with displacement errors*, IEEE Transactions on Circuits and Systems I volume 54, número 8, pp. 1788-1799, 2007.

[99] D. G. Luenberger e Y. Ye, *Linear and nonlinear programming*, Third Edition, Springer, 2010.

[100] MATLAB 7, The MathWorks, Inc., Natick, Massachusetts, United States.

[101] W. Menke, *Geophysical data analysis: discrete inverse theory*, Academic Press, San Diego, 1989.

[102] C. D. Meyer, *Matrix analysis and applied linear algebra*, SIAM, Cambridge University Press, 2000.

[103] S. Morigi, L. Reichel, F. Sgallari e F. Zama, *Iterative methods for ill-posed problems and semiconvergent sequences*, Journal of Computational and Applied Mathematics, volume 193, número 1, pp. 157-167, 2006.

[104] V. A. Morozov, *On the solution of functional equations by the method of regularization*, Soviet Mathematics Doklady volume 7, pp. 414-417, 1966.

[105] V. A. Morozov, *Regularization methods for solving incorrectly posed problems*, Springer-Verlag, New York, 1984.

[106] J. G. Nagy, K. Palmer e L. Perrone, *Iterative methods for image deblurring: a Matlab object oriented approach*, Numerical Algorithms, volume 36, número 1, pp 73-93, 2004.

[107] J. Nocedal e S. J. Wright, *Numerical optimization*, Springer-Verlag, New York, 1999.

[108] F. Natterer, *The mathematics of computerized tomography*, Wiley, New York, 1986.

[109] C. C. Paige, *Computational variants of the Lanczos method for the eigenproblem*, Journal of the Institute of Mathematics and its Applications, volume 10, pp. 373-381, 1972.

[110] C. C. Paige e M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM Journal on Numerical Analysis, volume 12, número 4, pp. 617-629, 1975.

[111] C. C. Paige e M. A. Saunders, *Towards a generalized singular value decomposition*, SIAM Journal on Numerical Analysis, volume 18, número 3, pp. 398-405, 1981.

[112] C. C. Paige e M. A. Saunders, *LSQR: An algorithm for sparse linear equations and sparse least squares*, ACM Transactions on Mathematical Software, volume 8, número 1, pp. 43-71, 1982.

[113] C. C. Paige e M. A. Saunders, *Algorithm 583. LSQR: Sparse linear equations and least squares problems*, ACM Transactions on Mathematical Software, volume 8, número 2, pp. 195-209, 1982.

[114] S. C. Park, M. K. Park e M. G. Kang, *Super-resolution image reconstruction: a technical overview*, IEEE Signal Processing Magazine, volume 20, número 3, pp. 21-36, 2003.

[115] D. L. Phillips, *A technique for the numerical solution of certain integral equations of the first kind*, Journal of the ACM, volume 9, número 1, pp. 84-97, 1962.

[116] R. Potthast, *Point sources and multipoles in inverse scattering theory*, Chapman & Hall/CRC, London, 2001.

[117] R. Potthast, *Stability estimates and the reconstructions in inverse acoustic scattering using singular sources*, Journal of Computational and Applied Mathematics, volume 114, número 2, pp. 247-274, 2000.

[118] T. Regińska, *A regularization parameter in discrete ill-posed problems*, SIAM Journal on Scientific Computing, volume 17, número 3, pp. 740-749, 1996.

[119] L. Reichel e G. Rodriguez, *Old and new parameter choice rules for discrete ill-posed problems*, Numerical Algorithms, doi 10.1007/s11075-012-9612-8.

[120] L. Reichel, F. Sgallari e Q. Ye, *Tikhonov regularization based on generalized Krylov subspace methods*, Applied Numerical Mathematics, volume 62, número 9, pp. 1215-1228, 2012.

[121] L. Reichel e Q. Ye, *Simple square smoothing regularization operators*, Electronic Transactions on Numerical Analysis, volume 33, pp. 63-83, 2009.

[122] J. Riley, *Solving systems of linear equations with a positive definite, symmetric, but possibly ill-conditioned matrix*, Mathematical Tables and Other Aids to Computation, volume 9, número 51, pp. 96-101, 1955.

[123] G. Rodriguez e D. Theis, *An algorithm for estimating the optimal regularization parameter by the L-curve*, Rendiconti di Matematica, número 25(1), pp. 69-84, 2005.

[124] B. W. Rust e D. P. O'Leary, *Residual periodograms for choosing regularization parameters for ill-posed problems*, Inverse Problems, volume 24, número 3, doi:10.1088/0266-5611/24/3/034005, 2008.

[125] Y. Saad e M. H. Schultz, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM Journal on Scientific and Statistical Computing, volume 7, número 3, pp. 856-869, 1986.

[126] F. Santosa, Y.-H. Pao, W. W. Symes e C. Holland, *Inverse problems of acoustic and elastic waves*, SIAM, Philadelphia, 1984.

[127] M. Salehi Ravesh, G. Brix, F. B. Laun, T. A. Kuder, M. Puderbach, J. Ley-Zaporozhan, S. Ley, A. Fieselmann, M. F. Herrmann, W. Schranz, W. Semmler e F. Risse, *Quantification of pulmonary microcirculation by dynamic contrast-enhanced magnetic resonance imaging: comparison of four regularization methods*, Magnetic Resonance in Medicine, doi: 10.1002/mrm.24220, 2012.

[128] M. A. Saunders, *Computing projections with LSQR*, BIT, volume 37, número 1, pp. 96-104, 1997.

[129] C. B. Shaw Jr., *Improvements of the resolution of an instrument by numerical solution of an integral equation*, Journal of Mathematical Analysis and Applications, volume 37, número 1, pp. 83-112, 1972.

[130] G. Strang, *The discrete cosine transform*, SIAM Review, volume 41, número 1, pp. 135-147, 1999.

[131] A. N. Tikhonov, *Solution of incorrectly formulated problems and the regularization method*, Soviet Mathematics Doklady, volume 4, pp. 1035-1038, 1963.

[132] D. M Titterington, *Common structure of smoothing techniques in statistics*, International Statistical Review, volume 53, número 2, pp. 141-170, 1985.

[133] S. Twomey, *On the numerical solution of Fredholm integral equations of the first kind by inversion of the linear system produced by quadrature*, Journal of the ACM, volume 10, número 1, pp. 97-101, 1963.

[134] C. R. Vogel, *Non-convergence of the L-curve regularization parameter selection method*, Inverse Problems, volume 12, número 4, pp. 535-547, 1996.

[135] P. Xu, Y. Fukuda e Y. Liu, *Multiple parameter regularization: numerical solutions and applications to the determination of geopotential from precise satellite orbits*, Journal of Geodesy, volume 80, número 1, pp. 17-27, 2006.

[136] A. van der Sluis e H. A. van der Vorst, *The rate of convergence of conjugate gradients*, Numerische Mathematik, volume 48, número 5, pp. 543-560, 1986.

[137] A. van der Sluis e H. A. van der Vorst, *SIRT- and CG-type methods for the iterative solution of sparse linear least-squares problems*, Linear Algebra and its Applications, volume 130, pp. 257-303, 1990.

[138] U. Tautenhahn e U. Hämarik, *The use of monotonicity for choosing the regularization parameter in ill-posed problems*, Inverse Problems, volume 15, número 6, pp. 1487-1506, 1999.

[139] A. J. Thorpe e L. L. Scharf, *Data adaptive rank-shaping methods for solving least squares problems*, IEEE Transactions on Signal Processing, volume 43, número 7, pp. 1591-1601, 1995.

[140] Z. Wang, *Multi-parameter Tikhonov regularization and model function approach to the damped Morozov principle for choosing regularization parameters*, Journal of Computational and Applied Mathematics, volume 236, número 7, pp. 1815-1832, 2012.

[141] Z. Wang e J. Liu, *New model function methods for determining regularization parameters in linear inverse problems*, Applied Numerical Mathematics, volume 59, número 10, pp. 2489-2506, 2009.

[142] M. V. W. Zibetti, F. S. V. Bazán e J. Mayer, *Determining the regularization parameters for super-resolution problems*, Signal Processing, volume 88, número 12, pp. 2890-2901, 2008.