



UNIVERSIDADE ESTADUAL DE
CAMPINAS

Instituto de Matemática, Estatística e
Computação Científica

DOUGLAS MENDES

**Um método de espaço nulo livre de bases para
a resolução de problemas de ponto-de-sela
simétricos e generalizados**

Campinas

2016

Douglas Mendes

**Um método de espaço nulo livre de bases para a
resolução de problemas de ponto-de-sela simétricos e
generalizados**

Tese apresentada ao Instituto de Matemática,
Estatística e Computação Científica da Uni-
versidade Estadual de Campinas como parte
dos requisitos exigidos para a obtenção do
título de Doutor em Matemática Aplicada.

Orientadora: Maria Aparecida Diniz Ehrhardt

Coorientador: Lucas Garcia Pedroso

Este exemplar corresponde à versão fi-
nal da Tese defendida pelo aluno Dou-
glas Mendes e orientada pela Profa.
Dra. Maria Aparecida Diniz Ehrhardt.

Campinas

2016

Agência(s) de fomento e nº(s) de processo(s): CAPES

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Maria Fabiana Bezerra Muller - CRB 8/6162

Mendes, Douglas, 1985-
M522m Um método de espaço nulo livre de bases para a resolução de problemas de ponto-de-sela simétricos e generalizados / Douglas Mendes. – Campinas, SP : [s.n.], 2016.

Orientador: Maria Aparecida Diniz Ehrhardt.
Coorientador: Lucas Garcia Pedroso.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1. Sistemas ponto de sela. 2. Soluções exatas. 3. Métodos numéricos. 4. Pré-condicionadores. I. Diniz Ehrhardt, Maria Aparecida, 1956-. II. Pedroso, Lucas Garcia. III. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: A basis-free null space method for solving symmetric and generalized saddle point problems

Palavras-chave em inglês:

Saddle point systems

Exact solutions

Numerical methods

Preconditioners

Área de concentração: Matemática Aplicada

Titulação: Doutor em Matemática Aplicada

Banca examinadora:

Maria Aparecida Diniz Ehrhardt [Orientador]

Márcia Aparecida Gomes Ruggiero

Maicon Ribeiro Correa

Luiz Carlos Matioli

Luís Felipe Cesar da Rocha Bueno

Data de defesa: 16-12-2016

Programa de Pós-Graduação: Matemática Aplicada

Tese de Doutorado defendida em 16 de dezembro de 2016 e aprovada

Pela Banca Examinadora composta pelos Profs. Drs.

Prof(a). Dr(a). MARIA APARECIDA DINIZ EHRHARDT

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

Prof(a). Dr(a). MAICON RIBEIRO CORREA

Prof(a). Dr(a). LUIZ CARLOS MATIOLI

Prof(a). Dr(a). LUÍS FELIPE CESAR DA ROCHA BUENO

A Ata da defesa com as respectivas assinaturas dos membros encontra-se no processo de vida acadêmica do aluno.

Agradecimentos

Agradeço a minha família por todo o suporte que me deram durante o meu doutorado, especialmente nestes dois últimos anos. Agradeço aos meus orientadores pela confiança que depositaram em mim ao terem permitido que eu desenvolvesse minha pesquisa com total liberdade, por acreditarem em minha capacidade e, não menos importante, pela paciência que também demonstraram comigo. Em especial, agradeço a prof^a Cheti por ter permitido que eu aproveitasse todas as oportunidades profissionais que surgiram em minha vida durante esse período. Elas certamente foram muito importantes para minha formação profissional. Por fim, agradeço a CAPES pelo auxílio financeiro dado.

Resumo

Nesta tese empregamos uma abordagem matricial de lagrangiano aumentado para determinar as soluções exatas de uma ampla classe de sistemas lineares, composta por problemas de ponto-de-sela, tanto simétricos quanto não simétricos. Modestas hipóteses são levantadas para esse fim e um pré-condicionamento é especialmente desenvolvido para diminuir a sensibilidade dos sistemas lineares antes do cálculo de suas soluções. Dessa forma as expressões encontradas podem ser propriamente empregadas para fins computacionais em um grande número de aplicações. Como um exemplo de um tal procedimento, desenvolvemos um método direto para a resolução de problemas de ponto-de-sela simétricos. Então mostramos que esse método está intrinsicamente relacionado a métodos de espaço nulo, mas não depende da determinação de uma base para o espaço nulo desses outros. Por essa razão, dizemos que o método proposto é um método de espaço nulo livre de bases. Ainda estendemos o método anterior a um método direto de resolução de problemas de ponto-de-sela generalizados, para o qual podemos livremente fazer as mesmas afirmações. Até onde sabemos, não existem métodos diretos especializados na resolução destes últimos problemas. Tão pouco a solução exata deles é conhecida na literatura. Assim ela é do maior interesse, já que pode levar ao desenvolvimento de algoritmos eficientes para a resolução dos mesmos, além dos nossos próprios. Por fim, explicitamos a expressão da inversa da matriz de coeficientes dos sistemas não-simétricos, também não conhecida na literatura, e identificamos as condições mais fracas possíveis necessárias para se garantir a solubilidade de todos os problemas de ponto-de-sela considerados, uma vez mais obtendo resultados inéditos.

Palavras-chave: sistemas ponto-de-sela, soluções exatas, métodos numéricos, pré-condicionadores.

Abstract

In this thesis we use an augmented Lagrangian matrix approach to determine the exact solutions of a broad class of linear systems composed by symmetric and nonsymmetric saddle point problems. Some mild assumptions are made to this end and a preconditioning is specially designed to decrease the sensitivity of the linear systems before the calculation of their solutions. In this way, the expressions found can be properly employed for computational purposes in a large number of applications. As an example of such a procedure, we develop a direct method for solving symmetric saddle point problems. Then we show that this method is intrinsically related with the null space method, but it doesn't depend on finding a basis to the null space of the latter. For this reason, we say the proposed method is a basis-free null space method. This method is also extended to a direct method for the solving of generalized saddle point problems, for which we can also freely make the same statements. To the best of our knowledge, there isn't any specific direct method for the solving of the last problems. So little their exact solution is known in the literature. Thus it is of major interest, since it can lead to the development of efficient algorithms, besides our own. Lastly, we explicit the expression of the inverse of the coefficient matrix of the nonsymmetric systems, also not known in the literature, and we identify the weakest possible assumptions that are needed to guarantee the solvability of all the saddle point problems considered here, once again obtaining new results.

Keywords: saddle point systems, exact solutions, numerical methods, preconditioners.

Lista de abreviaturas e siglas

$\text{card}(X)$	Cardinalidade de um conjunto X
$\text{cond}_2(C)$	Número de condição, na norma 2, de uma matriz C
$\ker(C)$	Espaço nulo de uma matriz C
$\text{ran}(C)$	Espaço imagem de uma matriz C
$\text{rank}(C)$	Posto de uma matriz C
$\text{span}\{v_1, \dots, v_n\}$	Espaço vetorial gerado pelos vetores v_1, \dots, v_n

Lista de códigos

Código C.1	–	Configuração de parâmetros comuns de várias funções	141
Código C.2	–	Geração de conjunto $\mathcal{P}_1(\mathbb{R}^d)$ -unisolvante de pontos	142
Código C.3	–	Geração de problema de ponto-de-sela simétrico	145
Código C.4	–	Conversão de problema de ponto-de-sela simétrico em generalizado .	150
Código C.5	–	Atualização do bloco $(1, 1)$ de problemas de ponto-de-sela	153
Código C.6	–	Geração de dados para análise dos algoritmos	156
Código C.7	–	Geração de gráficos dos dados dos algoritmos	166
Código C.8	–	Implementação do Algoritmo 3	172
Código C.9	–	Implementação do Algoritmo 4	175
Código C.10	–	Implementação do Algoritmo 5	178
Código C.11	–	Implementação do Algoritmo 6	183
Código C.12	–	<i>Solver</i> final para resolução de problemas de ponto-de-sela	188

Sumário

	INTRODUÇÃO	12
1	OTIMIZAÇÃO NUMÉRICA	17
1.1	Fundamentos de otimização	17
1.2	Otimização com restrições de igualdade	25
1.3	Métodos clássicos de resolução de sistemas KKT	33
1.3.1	O método do complemento de Schur	34
1.3.2	Os métodos de espaço nulo	35
1.3.3	O método dos gradientes conjugados projetados	36
1.4	Métodos de penalidade e a técnica de lagrangiano aumentado	41
1.5	Uma generalização da técnica de lagrangiano aumentado	45
2	RESOLUÇÃO DE PROBLEMAS DE PONTO-DE-SELA SIMÉTRICOS	47
2.1	Um método iterativo de resolução de problemas de ponto-de-sela simétricos	49
2.2	Alguns resultados técnicos	55
2.3	A solução exata de problemas de ponto-de-sela simétricos	59
2.4	Um método direto de resolução de problemas de ponto-de-sela simétricos	63
2.5	Uma dedução alternativa do método de espaço nulo	67
3	RESOLUÇÃO DE PROBLEMAS DE PONTO-DE-SELA GENERALIZADOS	71
3.1	A solução exata de problemas de ponto-de-sela generalizados	73
3.2	Um método direto de resolução de problemas de ponto-de-sela generalizados	78
3.3	Identificando as hipóteses mais fracas possíveis para a resolução de problemas de ponto-de-sela	81
4	EXPERIMENTOS NUMÉRICOS	89
5	CONSIDERAÇÕES FINAIS	122
	REFERÊNCIAS	127

	APÊNDICE A – MÉTODOS ESTACIONÁRIOS PARA SISTEMAS LINEARES	131
	APÊNDICE B – ALTERNATIVAS PARA INTERPOLAÇÃO DE DADOS	133
B.1	O problema de interpolação de dados	133
B.2	Interpolação com funções definidas positivas	134
B.3	Interpolação com funções condicionalmente definidas positivas . . .	137
	APÊNDICE C – CÓDIGOS-FONTES	141

Introdução

Assim que iniciei o doutorado, fui apresentado à teoria de otimização sem derivadas. Cursando disciplinas e também frequentando seminários nesta área, logo percebi que os algoritmos deste tipo, com melhor desempenho para a minimização dos valores de funções, eram aqueles que faziam uso de modelos interpoladores. Havia, entretanto, um fator comum a todos eles, justamente em relação a esses modelos, que até aquele momento se mostraram sempre polinomiais. A razão para isso claramente se devia à simplicidade desses objetos, nem um pouco abstratos e facilmente manipuláveis, tanto analiticamente, quanto algebricamente, ou ainda, computacionalmente. De um ponto de vista mais teórico, no entanto, nada de realmente concreto existia que justificasse a falta de utilização de modelos interpoladores pertencentes a outras classes de funções. A noção de que poderiam existir modelos diferentes e mais apropriados à minimização de uma mesma função parecia-me suficientemente natural para que obrigatoriamente tivesse de ser considerada em algum instante. Interessado na questão, acabei conhecendo a teoria de interpolação com funções condicionalmente definidas positivas, bem mais abrangente do que aquela polinomial. E foquei meus estudos nela por um certo período.

A determinação de interpoladores com a teoria então descoberta equivale à resolução de sistemas lineares muito específicos, mas bastante comuns em otimização: os chamados sistemas KKT. O formato deles é o seguinte:

$$\begin{bmatrix} A & B^T \\ B & O \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix},$$

onde $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{m \times n}$, $a \in \mathbb{R}^n$ e $b \in \mathbb{R}^m$, com $m \leq n$. Inicialmente era apenas curioso para mim enxergar uma tal relação entre áreas tão distintas, mas conforme fui investigando a questão, entendi que esses sistemas, na realidade, surgem naturalmente em uma série de aplicações reais de matemática, onde são também chamados de problemas de ponto-de-sela. Algumas dessas áreas são: dinâmica de fluidos computacional, otimização com restrições, economia, redes e circuitos eletrônicos, eletromagnetismo, finanças, reconstrução de imagens, interpolação de dados dispersos, elasticidade linear, geração de malhas para gráficos, aproximações de equações diferenciais parciais elípticas por elementos finitos mistos, redução de ordem de modelos para sistemas dinâmicos, controle ótimo, identificação de parâmetros em problemas diversos ((BENZI; GOLUB; LIESEN, 2005, p. 5-6)). Portanto a própria resolução dos problemas de ponto-de-sela é muito mais relevante do que qualquer aplicação que leve a eles. Em especial, porque essa resolução é complicada, devido às matrizes de coeficientes dos sistemas serem indefinidas e usualmente bastante mal-condicionadas. Devido a isso, eu e meus orientadores procuramos desenvolver nesta tese um método de resolução de problemas de ponto-de-sela, a princípio apenas

simétricos, capaz de aproveitar a estrutura especial desses problemas, de lidar com o mau-condicionamento deles e, ainda, de resolvê-los sob condições minimamente suficientes. Pois assim este único método poderia servir a todas as áreas mencionadas e possivelmente ainda outras. Em particular, à otimização de um modo geral. Os objetivos indicados foram todos simultaneamente alcançados e demonstramos isso analiticamente. Experimentos numéricos realizados com problemas gerados a partir da teoria de interpolação com funções condicionalmente definidas positivas ainda reforçam que o método proposto para a resolução de problemas de ponto-de-sela é robusto, eficiente e preciso. Sob condições numéricas apropriadas, majoritariamente expressas pela condição de que B é uma matriz bem-condicionada, conseguimos concluir que tal método é competitivo para problemas de ordem até mais ou menos 15000. Entendemos que essa ordem é suficiente para atender a um bom número de problemas reais e condiz com a natureza do método, que é direto.

Neste trabalho identificamos ainda as condições mais fracas possíveis para a solubilidade de um problema de ponto-de-sela. Mas as hipóteses originalmente consideradas para isso foram simplesmente a de que B é uma matriz de posto completo e a de que A é uma matriz definida positiva no $\ker(B)$. Elas podem ser vistas como condições naturais a um problema de otimização com restrições, em vista, por exemplo, das condições suficientes de segunda ordem para a otimalidade de um ponto estacionário da função lagrangiana associada a este problema. Foram estas próprias hipóteses que nos levaram a desenvolver um método de resolução de problemas de ponto-de-sela do tipo espaço nulo, cuja aplicação em otimização também é muito comum. Porém, este método conta com algumas diferenças importantes em relação aos métodos de espaço nulo tradicionais. A saber: a de que ele não necessita de uma base do $\ker(B)$ e a de que sua aplicação é mais vantajosa quando $m \ll n$. A primeira diferença, nada mais, nada menos, elimina do método proposto a maior dificuldade para aplicação de um método de espaço nulo tradicional. É em razão dela que dizemos que o método desenvolvido é livre de bases. A segunda diferença, por sua vez, representa justamente aquela situação pouco favorável para os métodos de espaço nulo tradicionais, mas corriqueira para os problemas de ponto-de-sela. Garantimos essas boas propriedades para o nosso método, empregando projeções no $\ker(B)$, a exemplo do que [Gould, Hribar e Nocedal \(2001\)](#) fazem em um algoritmo de gradientes conjugados projetados, de autoria deles.

A construção do método proposto é bastante elaborada e parte da expressão explícita da solução do problema de ponto-de-sela. Para conseguir determiná-la, utilizamos momentaneamente um método iterativo estacionário, cuja convergência aceleramos com um pré-condicionamento especial — que nunca explicitamos, por falta de necessidade — para sempre terminar em três iterações, independentemente de n , m e do problema de ponto-de-sela. Entre outras técnicas, utilizamos no pré-condicionamento uma generalização da técnica de lagrangiano aumentado de otimização, com a qual é possível aproximadamente minimizar, em um determinado conjunto de matrizes, o número de condição do bloco

$(1, 1)$ dos sistemas lineares que compõem o método iterativo em questão. Sob uma módica condição sobre este bloco, conseguimos ainda provar que o número de condição dele é menor do que $\text{cond}_2(A)$. Quem assegura a referida propriedade de minimização, primeiramente empiricamente e depois analiticamente, são [Golub e Greif \(2003\)](#) e [Golub, Greif e Varah \(2006\)](#), respectivamente. Já nós garantimos analiticamente que o complemento de Schur associado a este bloco é perfeitamente bem-condicionado. Além disso, aproveitamos o resultado desses mesmos autores para também concluir analiticamente que o número de condição da matriz de coeficientes do sistema linear que resolvemos internamente no método proposto é aproximadamente minimizado. Com o auxílio de uma manipulação numérica, obtemos um método de resolução de problemas de ponto-de-sela bastante estável. Admitimos, porém, que ainda não entendemos completamente o efeito da manipulação mencionada nos nossos cálculos.

Todos os resultados anteriores foram estendidos a problemas de ponto-de-sela com matriz A não-simétrica, os chamados problemas de ponto-de-sela generalizados. Em particular, isso resultou em um método direto próprio para a resolução destes novos problemas. Estamos quase certos de que não há na literatura um método direto especializado na resolução dos mesmos e dizemos isso nos apoiando em uma afirmação feita pelo próprio Golub (([BENZI; GOLUB; LIESEN, 2005](#), p. 40)). Por essa razão, acreditamos que o nosso método para a resolução de problemas de ponto-de-sela generalizados seja o primeiro nesta categoria. Por se tratar de uma extensão do método anterior, algumas vezes preferimos nos referir a eles como as versões simétrica e não-simétrica de um único método, comum à resolução de ambos os problemas de ponto-de-sela.

Nesta tese fornecemos ainda um *framework* para a resolução de problemas gerais de otimização com restrições, como uma ilustração de uso da versão simétrica do método que desenvolvemos para a resolução de problemas de ponto-de-sela. A partir desse *framework*, vários outros trabalhos em otimização são possíveis, bastando, para tanto, variar as estratégias de conjuntos ativos em uso ou os métodos de atualização das aproximações das hessianas da função objetivo e das restrições. Apesar da inclusão desse *framework*, o foco do trabalho ainda permanece sendo o método de resolução de problemas de ponto-de-sela desenvolvido.

Resumimos a seguir todas as nossas contribuições com este trabalho:

- Determinamos as soluções exatas dos problemas de ponto-de-sela simétrico e generalizado, tomando o cuidado de calcular expressões apropriadas para a avaliação computacional delas. Logo não se tratam de quaisquer expressões para as soluções de determinados problemas. Não conseguimos encontrar até esta data na literatura nenhuma expressão equivalente à solução do problema de ponto-de-sela generalizado;
- Encontramos as expressões explícitas das matrizes de coeficientes dos sistemas que

compõem os problemas de ponto-de-sela simétrico e generalizado, mesmo que elas popularmente tenham apenas um apelo teórico. A primeira delas é diferente daquela conhecida hoje na literatura, embora equivalente a ela por uma questão de unicidade da inversa. Esta expressão reflete o cuidado que tivemos ao procurar por fórmulas para avaliações computacionais mais precisas. Até onde sabemos, nenhuma expressão foi antes exibida para a inversa da matriz de coeficientes do problema generalizado, sendo a nossa, então, inédita;

- Identificamos as condições mais fracas possíveis que são suficientes para se garantir a existência e unicidade de solução para os problemas de ponto-de-sela simétricos e generalizados. Em particular, condições mais fracas do que aquelas propostas por autores consagrados na área, como o [Benzi e Golub \(2004\)](#). As referidas condições são dadas simplesmente em termos das matrizes A e B , ou de partes delas. Segundo nossa pesquisa bibliográfica, estamos convencidos de que estas condições ainda não foram notadas por outros pesquisadores. A especificidade das construções que levam a elas sugerem que este é mesmo o caso. Aliás, tais construções podem possivelmente ser vistas como ótimas, em um sentido que deixaremos claro mais adiante;
- Propomos métodos diretos para a resolução de problemas de ponto-de-sela simétricos e generalizados, que são adequados para serem utilizados quando $m \ll n$, ou seja, na situação frequente em que aplicações práticas de matemática resultam em problemas de ponto-de-sela. Tais métodos podem ser vistos como métodos de espaço nulo livre de bases. Quando a matriz B é bem-condicionada, todos eles se mostram bastante robustos, eficientes e precisos, embora para isso ainda seja necessária a utilização de um artifício computacional, cujo efeito entendemos apenas parcialmente. No melhor do nosso conhecimento, o método direto que propomos, específico para a resolução de problemas de ponto-de-sela generalizados, é o primeiro desta natureza;
- Propomos ainda, um método iterativo estacionário para a resolução de problemas de ponto-de-sela simétricos. Não realizamos experimentos numéricos com ele, devido ao momento tardio em que o concebemos, mas argumentamos que se trata de um método com boas propriedades. Ele também é muito barato de ser executado computacionalmente, pois majoritariamente apenas emprega produtos de matrizes por vetores em seus cálculos. A exceção fica por conta da resolução de um número muito pequeno de sistemas lineares de ordem m , a partir do fator de Cholesky de uma matriz comum a todos eles. O impacto disso na performance do método não é significativo, porque eventualmente vamos supor que $m \ll n$;
- Por fim, também fornecemos um *framework* para a resolução de problemas gerais de otimização com restrições, no qual é possível aplicar os métodos propostos para problemas de ponto-de-sela simétricos. Ainda estamos investigando a inediticidade dele.

Os capítulos desta tese são como seguem. No Capítulo 1, tratamos de fundamentos da otimização e de problemas de minimização com restrições de igualdade, além da generalização da técnica de lagrangiano aumentado mencionada anteriormente. Todos tópicos necessários para a melhor compreensão deste texto. Nos Capítulos 2 e 3, apresentamos a teoria que desenvolvemos para a resolução de problemas de ponto-de-sela simétricos e generalizados, dedicando cada capítulo especificamente à análise de somente um destes problemas. No Capítulo 4, descrevemos os experimentos numéricos realizados e seus resultados, os quais foram gerados com a teoria de interpolação com funções condicionalmente definidas positivas, brevemente introduzida no Apêndice B. Discorremos no Apêndice A sobre métodos estacionários para sistemas lineares e listamos no Apêndice C os códigos-fontes das implementações dos métodos propostos, entre outros códigos pertinentes a esta tese. Na parte de Considerações Finais, resumimos nossas descobertas, apontando diversas propostas de possíveis trabalhos futuros.

1 Otimização numérica

Neste capítulo apresentamos os conceitos e técnicas de otimização que motivarão mais tarde o desenvolvimento de um método para resolução de sistemas lineares muito específicos, os chamados sistemas KKT. Tratam-se, em sua totalidade, de conhecimentos bastante difundidos, mas que algumas vezes exibiremos em um contexto menos tradicional, para atender melhor nossas necessidades futuras.

Os tópicos deste capítulo são como seguem. Na Seção 1.1 introduzimos brevemente alguns conceitos básicos de otimização e os métodos de Newton e das secantes. Na Seção 1.2 focamos nossa atenção na resolução de problemas de otimização com restrições de igualdade, fornecendo, por outro lado, um *framework* de nossa autoria para a resolução de problemas gerais de otimização com restrições. Na Seção 1.3 resumimos alguns métodos clássicos de resolução de um determinado tipo de sistema linear, que é o objeto de estudo desta tese. São as ideias que permeiam esses métodos que motivarão o desenvolvimento do restante deste trabalho. Na Seção 1.4 introduzimos duas técnicas importantes de otimização, a de penalidade e de lagrangiano aumentado. E na Seção 1.5 tratamos de analisar uma generalização da técnica de lagrangiano aumentado, própria para a resolução dos sistemas lineares que acabamos de mencionar.

1.1 Fundamentos de otimização

Em otimização irrestrita, nós procuramos minimizar uma função $f : \mathbb{R}^n \rightarrow \mathbb{R}$ suave, chamada **função objetivo**, sem impor qualquer tipo de restrição aos valores de suas variáveis. Representamos este problema por

$$\min_{x \in \mathbb{R}^n} f(x). \quad (1.1)$$

Para resolvê-lo, gostaríamos de sempre poder encontrar um ponto de \mathbb{R}^n onde a função atinge seu menor valor. Mas pode ser difícil localizar um tal ponto, uma vez que nosso conhecimento da função f usualmente é apenas local. Assim muitas vezes nos contentamos apenas em determinar um ponto no qual temos certeza de que o valor de f é mínimo em pelo menos uma vizinhança ao redor dele. Definimos formalmente estes conceitos abaixo.

Definição 1.1. Um ponto $x^* \in \mathbb{R}^n$ é um **minimizador global** de f se $f(x^*) \leq f(x)$ para todo $x \in \mathbb{R}^n$. Um ponto $x^* \in \mathbb{R}^n$ é um **minimizador local** de f se existe uma vizinhança $U \subset \mathbb{R}^n$ de x^* tal que $f(x^*) \leq f(x)$ para todo $x \in U$. Se $f(x^*) < f(x)$ para todo $x \in U$, com $x \neq x^*$, então x^* também é dito um **minimizador local estrito** de f .

Examinar os valores da função f em todos os pontos de alguma vizinhança de um dado ponto $x^* \in \mathbb{R}^n$ para verificar que em nenhum deles f atinge um valor menor do que em x^* obviamente não é prático. Ou até mesmo é impossível, como no caso de otimização contínua. Por isso caracterizações melhores de um minimizador local se fazem necessárias. Quando f é duas vezes continuamente diferenciável, por exemplo, é possível dizer que x^* é um minimizador local de f apenas examinando os valores do gradiente e da hessiana da função f no ponto x^* . Este resultado provê as fundações do desenvolvimento de algoritmos para otimização irrestrita. Antes de enunciá-lo, porém, mencionamos alguns resultados relacionados.

Proposição 1.1 (Condição necessária de primeira ordem). *Se x^* é um minimizador local de f e f é continuamente diferenciável em uma vizinhança aberta de x^* , então $\nabla f(x^*) = 0$.*

Demonstração. Pode ser encontrada em (NOCEDAL; WRIGHT, 2006, p. 14). \square

Nós dizemos que $x \in \mathbb{R}^n$ é um **ponto estacionário** de f se $\nabla f(x) = 0$. De acordo com a proposição acima, todo minimizador local de uma função continuamente diferenciável deve ser também um ponto estacionário dela.

Para os próximos resultados, recordamos que uma matriz simétrica $C \in \mathbb{R}^{n \times n}$ é **definida positiva** se $x^T C x > 0$ para todo $x \in \mathbb{R}^n \setminus \{0\}$ e é **semi-definida positiva** se $x^T C x \geq 0$ para todo $x \in \mathbb{R}^n$. Um conceito relacionado é o de **matriz indefinida**, que é quando existem vetores não-nulos $x, y \in \mathbb{R}^n$ tais que $x^T C x > 0$ e $y^T C y < 0$. Pode-se mostrar que: C é uma matriz definida positiva se todos os seus autovalores são positivos; C é uma matriz semi-definida positiva se todos os seus autovalores são não-negativos; e C é uma matriz indefinida se ela possui autovalores positivos e negativos ((GOLUB; VAN LOAN, 2013, p. 160, 167) e (HORN; JOHNSON, 2013, p. 234)).

Proposição 1.2 (Condições necessárias de segunda ordem). *Se x^* é um minimizador local de f e $\nabla^2 f$ existe e é contínua em uma vizinhança aberta de x^* , então $\nabla f(x^*) = 0$ e $\nabla^2 f(x^*)$ é semi-definida positiva.*

Demonstração. Pode ser encontrada em (NOCEDAL; WRIGHT, 2006, p. 15). \square

Teorema 1.3 (Condições suficientes de segunda ordem). *Suponha que $\nabla^2 f$ é contínua em uma vizinhança aberta de x^* , que $\nabla f(x^*) = 0$ e que $\nabla^2 f(x^*)$ é definida positiva. Então x^* é um minimizador local estrito de f .*

Demonstração. Pode ser encontrada em (NOCEDAL; WRIGHT, 2006, p. 16). \square

Muitas vezes também precisamos minimizar funções cujas variáveis não podem assumir quaisquer valores. Ou seja, para as quais há restrições bem definidas. Representa-

mos este outro problema por

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{sujeita a} \quad c_i(x) = 0, \quad i \in \mathcal{E}, \quad \text{e} \quad c_i(x) \leq 0, \quad i \in \mathcal{I}, \quad (1.2)$$

onde f e c_i , $i \in \mathcal{E} \cup \mathcal{I}$, são funções reais e suaves, definidas em \mathbb{R}^n , e \mathcal{E} e \mathcal{I} são dois conjuntos de cardinalidades finitas de números naturais que indexam as **restrições de igualdade** e **desigualdade**, respectivamente. Mais simplificada, denotamos o conjunto de todos os pontos que satisfazem simultaneamente a todas essas restrições por

$$\Omega = \{x \in \mathbb{R}^n \mid c_i(x) = 0, \quad i \in \mathcal{E}; \quad c_i(x) \leq 0, \quad i \in \mathcal{I}\}$$

e dizemos que Ω é o **conjunto viável** do problema. Chamamos qualquer um de seus pontos de **ponto viável**.

Definições de soluções locais para o problema acima são extensões daquelas do caso irrestrito, mas que levam agora em consideração a viabilidade da solução.

Definição 1.2. Um ponto $x^* \in \mathbb{R}^n$ é um **minimizador local** do problema (1.2) se $x^* \in \Omega$ e se existe uma vizinhança $U \subset \mathbb{R}^n$ de x^* tal que $f(x^*) \leq f(x)$ para todo $x \in U \cap \Omega$. Se $x^* \in \Omega$ e se $f(x^*) < f(x)$ para todo $x \in U \cap \Omega$, com $x \neq x^*$, então x^* também é dito um **minimizador local estrito** de f .

Suponha que o conjunto viável do problema de minimizar f sujeita a Ω seja determinado apenas a partir de restrições de desigualdade. Deve estar claro que este problema é essencialmente um problema de minimização irrestrita se o minimizador de (1.2) está no interior de Ω . Evitamos essa situação menos interessante do ponto de vista de otimização com restrições, definindo para o problema geral o **conjunto das restrições ativas** em um ponto viável x :

$$\mathcal{A}(x) = \mathcal{E} \cup \{i \in \mathcal{I} \mid c_i(x) = 0\} \subset \mathbb{N}.$$

Com ele distinguimos entre **restrições ativas** e **inativas** nesse mesmo ponto, ou seja, identificamos aquelas restrições para as quais $c_i(x) = 0$ e $c_i(x) < 0$ em x , respectivamente. Assim, daqui em diante somente nos importam os problemas de otimização com restrições que tenham $\mathcal{A}(x^*) \neq \emptyset$ em um minimizador local $x^* \in \Omega$. Diante dessa consideração, a fronteira do conjunto viável tem um papel fundamental na determinação dos minimizadores locais de (1.2). Levantando hipóteses sobre as restrições que a define, conseguimos estabelecer condições necessárias a serem satisfeitas por esses minimizadores. Uma tal hipótese é chamada **condição de qualificação**. Abaixo enunciamos a condição de qualificação mais frequentemente utilizada no desenvolvimento de algoritmos de otimização com restrições.

Definição 1.3. Dizemos que a **condição de qualificação de independência linear (LICQ)** se verifica em um ponto $x \in \Omega$ se os gradientes das restrições ativas em x ,

$$\nabla c_i(x), \quad i \in \mathcal{A}(x),$$

constituem um conjunto linearmente independente.

Suponhamos que m é a cardinalidade do conjunto $\mathcal{E} \cup \mathcal{I}$, isto é, $m = \text{card}(\mathcal{E} \cup \mathcal{I})$. As condições necessárias de primeira ordem para o problema (1.2) são descritas a partir dos pontos estacionários de uma função auxiliar da função objetivo f .

Definição 1.4. A *função lagrangiana* ou *lagrangiano* para (1.2) é a função

$$\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}, \quad \mathcal{L}(x, y) = f(x) + \sum_{i \in \mathcal{E} \cup \mathcal{I}} y_i c_i(x).$$

As quantidades escalares y_i , $i \in \mathcal{E} \cup \mathcal{I}$, são chamadas **multiplicadores de Lagrange**.

Proposição 1.4 (Condições necessárias de primeira ordem). *Se x^* é um minimizador local do problema (1.2), as funções f e c_i em (1.2) são continuamente diferenciáveis e LICQ se verifica em x^* , então existe um vetor y^* de multiplicadores de Lagrange, com componentes y_i^* , $i \in \mathcal{E} \cup \mathcal{I}$, tal que as seguintes condições são satisfeitas em (x^*, y^*) :*

$$\begin{aligned} \nabla_x \mathcal{L}(x^*, y^*) &= 0, \\ c_i(x^*) &= 0, \quad \text{para todo } i \in \mathcal{E}, \\ c_i(x^*) &\leq 0, \quad \text{para todo } i \in \mathcal{I}, \\ y_i^* &\geq 0, \quad \text{para todo } i \in \mathcal{I}, \\ y_i^* c_i(x^*) &= 0, \quad \text{para todo } i \in \mathcal{E} \cup \mathcal{I}. \end{aligned}$$

Demonstração. Pode ser encontrada em (NOCEDAL; WRIGHT, 2006, p. 329). \square

As condições apresentadas nesta proposição são conhecidas como **condições de KKT**, por terem sido formuladas por Karush, Khun e Tucker. Dado um índice $i \in \mathcal{E} \cup \mathcal{I}$, a última delas nos diz que, ou a restrição correspondente a i está ativa, ou $y_i^* = 0$, ou ambos. Isso em particular implica que todos os multiplicadores de Lagrange associados a uma restrição de desigualdade inativa são nulos. Assim é possível omitir os índices $i \notin \mathcal{A}(x^*)$ da primeira equação e reescrevê-la com menos termos, simplesmente como

$$\nabla f(x^*) + \sum_{i \in \mathcal{A}(x^*)} y_i^* \nabla c_i(x^*) = 0.$$

Para o propósito dos próximos resultados, supomos que f e c_i , $i \in \mathcal{E} \cup \mathcal{I}$, são todas funções duas vezes continuamente diferenciáveis. Também denotamos por

$$B(x^*) = \left[\nabla c_i(x^*)^T \right]_{i \in \mathcal{A}(x^*)} \quad (1.3)$$

a matriz jacobiana das restrições ativas de (1.2), avaliada em x^* . Notamos que $B(x^*)$ tem pelo menos uma linha, porque assumimos anteriormente que o conjunto $\mathcal{A}(x^*)$ é não-vazio. Consequentemente, o núcleo de $B(x^*)$ é não-trivial sempre que $\text{card}(\mathcal{A}(x^*)) < n$.

Proposição 1.5 (Condição necessária de segunda ordem). *Se x^* é um minimizador local de (1.2), no qual LICQ se verifica, e y^* é o vetor de multiplicadores de Lagrange para o qual as condições de KKT são satisfeitas, então*

$$\tilde{x}^T \nabla_{xx}^2 \mathcal{L}(x^*, y^*) \tilde{x} \geq 0$$

para todo $\tilde{x} \in \ker(B(x^*))$.

Demonstração. Pode ser encontrada em (NOCEDAL; WRIGHT, 2006, p. 332). \square

Teorema 1.6 (Condições suficientes de segunda ordem). *Suponha que para algum ponto viável x^* exista um vetor de multiplicadores de Lagrange y^* , com componente $y_i^* > 0$, para todo $i \in \mathcal{I} \cap \mathcal{A}(x^*)$, tal que as condições de KKT são satisfeitas, e que*

$$\tilde{x}^T \nabla_{xx}^2 \mathcal{L}(x^*, y^*) \tilde{x} > 0 \quad \text{para todo } \tilde{x} \in \ker(B(x^*)) \setminus \{0\}.$$

Então x^ é um minimizador local estrito de (1.2).*

Demonstração. Pode ser encontrada em (NOCEDAL; WRIGHT, 2006, p. 333). \square

Seja $m = \text{card}(\mathcal{A}(x^*))$ e suponha daqui em diante que $m < n$. Se $K \in \mathbb{R}^{n \times (n-m)}$ é uma matriz de colunas que geram o $\ker(B(x^*))$, então os resultados das proposições anteriores evidentemente podem ser reenunciados em termos da (semi-)positividade da matriz $K^T \nabla_{xx}^2 \mathcal{L}(x^*, y^*) K$. Para o último, preferimos dizer ainda que $\nabla_{xx}^2 \mathcal{L}(x^*, y^*)$ é uma matriz definida positiva no $\ker(B(x^*))$.

Claramente, para podermos utilizar o Teorema 1.3 ou 1.6, precisamos antes conhecer um candidato a solução do problema de otimização que se deseja resolver, seja ele irrestrito (1.1) ou com restrições (1.2). Segundo as Proposições 1.1 e 1.4, a procura por esse candidato se dá entre os pontos estacionários da própria função f ou então da função lagrangiana \mathcal{L} , conforme o problema abordado. Em qualquer caso, dependemos da determinação de um zero de função. Vejamos, por isso, um método para obtê-lo.

Sejam $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$ uma função continuamente diferenciável e $J_F(x^-) \in \mathbb{R}^{n \times n}$ a matriz jacobiana de F , avaliada em um ponto $x^- \in \mathbb{R}^n$. Vamos considerar o problema de se encontrar $x^* \in \mathbb{R}^n$ tal que $F(x^*) = 0$, assumindo que a função F admita um tal ponto. Em vista da expansão de Taylor de F em torno de x^- , que existe porque F tem derivadas contínuas, podemos empregar a aproximação

$$F(x) \approx F(x^-) + J_F(x^-)(x - x^-)$$

para a função F em uma vizinhança $U \subset \mathbb{R}^n$ suficientemente pequena do ponto x^- . Se determinamos $x^+ \in \mathbb{R}^n$ que verifica a equação $F(x^-) + J_F(x^-)(x^+ - x^-) = 0$, acabamos

então com um ponto cuja expectativa é a de que ele seja uma estimativa melhor para o zero de F do que x^- . Conseguimos isso resolvendo o sistema linear

$$J_F(x^-)d = -F(x^-) \quad (1.4)$$

para $d \in \mathbb{R}^n$ e, em seguida, tomando $x^+ = x^- + d$. Podemos esperar estimativas ainda um pouco melhores, se repetirmos este procedimento algumas vezes, sempre partindo do ponto recém calculado. Isso motiva a definição do seguinte esquema iterativo, dado com base em um ponto $x^{(0)} \in \mathbb{R}^n$ arbitrário:

$$J_F(x^{(k)})d^{(k)} = -F(x^{(k)}), \quad x^{(k+1)} = x^{(k)} + d^{(k)}, \quad k \in \mathbb{N}_0, \quad (1.5)$$

onde $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. O procedimento descrito é chamado método de Newton para equações não-lineares ((DENNIS; SCHNABEL, 1996, p. 87)) e pode ser aplicado aos problemas (1.1) e (1.2) fazendo-se neles $F = \nabla f$ e $F = \nabla \mathcal{L}$, respectivamente. Sob condições apropriadas, a sequência $(x^{(k)})$ converge em poucos termos para x^* , qualquer que seja o ponto $x^{(0)}$ tomado suficientemente perto de x^* .

Teorema 1.7. *Suponha que $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$ é uma função continuamente diferenciável em um conjunto aberto e convexo $U \subset \mathbb{R}^n$. Seja $x^* \in U$ um zero isolado de F tal que $J_F(x^*)$ é não-singular. Então a sequência $(x^{(k)})$, definida por (1.5) e por algum ponto $x^{(0)} \in U$ suficientemente próximo de x^* , está bem-definida, converge para x^* e satisfaz*

$$\|x^{(k+1)} - x^*\|_2 \leq \Lambda^{(k)} \|x^{(k)} - x^*\|_2 \quad (1.6)$$

para todo $k \in \mathbb{N}_0$ e alguma sequência $(\Lambda^{(k)})$ de número reais positivos tal que $\Lambda^{(k)} \rightarrow 0$, quando $k \rightarrow \infty$. Melhor ainda: quando J_F também é uma função Lipschitz contínua em uma vizinhança suficientemente pequena de x^* , a sequência $(x^{(k)})$ satisfaz

$$\|x^{(k+1)} - x^*\|_2 \leq \Lambda \|x^{(k)} - x^*\|_2^2 \quad (1.7)$$

para todo $k \in \mathbb{N}_0$ e alguma constante $\Lambda > 0$.

Demonstração. Pode ser encontrada em (NOCEDAL; WRIGHT, 2006, p. 276). \square

As desigualdades (1.6) e (1.7) no teorema anterior indicam as taxas de convergência do método de Newton e traduzem sua eficiência. Quando a primeira se verifica, dizemos que o método tem **convergência superlinear**. Quando é a segunda que ocorre, **convergência quadrática** ((DENNIS; SCHNABEL, 1996, p. 20, 42)). Na prática, ambas as convergências são rápidas, mas a segunda é preferível, por ser muito mais rápida do que a primeira. Por isso que o método de Newton, quando converge, é bastante eficiente. Estas definições também se aplicam a qualquer outro método iterativo, não somente ao método de Newton.

Uma das desvantagens do método de Newton (exatamente como apresentado aqui) é depender de condições restritivas de convergência em alguns problemas, como a escolha de um ponto inicial $x^{(0)}$. Outras desvantagens do método, que também merecem atenção, dizem respeito ao cálculo de $J_F(x^{(k)})$ e à resolução de um sistema linear com matriz de coeficientes $J_F(x^{(k)})$, ambos a cada iteração. Isso porque pode ser computacionalmente caro obter $J_F(x^{(k)})$ e ainda há a possibilidade desta matriz ser singular ou mal-condicionada. Estratégias que contornam estas dificuldades podem ser vistas em (NOCEDAL; WRIGHT, 2006, p. 277-301) e, melhor ainda, em (DENNIS; SCHNABEL, 1996, p. 94-238). Abaixo resumimos muito brevemente uma delas, chamada **método das secantes**, que faremos uso já na seção seguinte.

Vamos considerar por um momento uma função $f : \mathbb{R} \rightarrow \mathbb{R}$ de uma única variável. Se f é diferenciável em um ponto $x^+ \in \mathbb{R}$, podemos aproximá-la em um intervalo $U \subset \mathbb{R}$ suficientemente pequeno e que contém este ponto, por um modelo afim

$$m^+(x) = f(x^+) + a^+(x - x^+),$$

onde $a^+ = f'(x^+)$. Se f não é diferenciável em x^+ e x^- é qualquer ponto em $U \setminus \{x^+\}$, é razoável ainda aproximá-la pelo mesmo modelo, mas para tanto devemos tomar

$$a^+ = \frac{f(x^+) - f(x^-)}{x^+ - x^-}, \quad (1.8)$$

em razão da própria definição de derivada de uma função real. Estendemos esta ideia a uma função $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$, aproximando-a em uma vizinhança $U \subset \mathbb{R}^n$ suficientemente pequena de um ponto $x^+ \in \mathbb{R}^n$, pelo modelo afim

$$M^+(x) = F(x^+) + A^+(x - x^+),$$

onde escolhemos $A^+ \in \mathbb{R}^{n \times n}$ para substituir $J_F(x^+)$ e satisfazer a chamada **equação secante**,

$$A^+(x^+ - x^-) = F(x^+) - F(x^-), \quad (1.9)$$

evidentemente análoga a (1.8). Ao contrário do que ocorre no caso unidimensional, contudo, somente esta equação não define a matriz A^+ unicamente, porque ainda há $n(n-1)$ graus de liberdade para sua determinação. Gostaríamos de eliminá-los com o mínimo de esforço computacional possível. Assim procuramos encontrar uma matriz A^+ que satisfaça a equação secante, mas que também minimize as mudanças necessárias a um modelo

$$M^-(x) = F(x^-) + A^-(x - x^-)$$

previamente disponível para F . A diferença entre esses modelos, em qualquer $x \in \mathbb{R}^n$, é

$$\begin{aligned} M^+(x) - M^-(x) &= F(x^+) + A^+(x - x^+) - F(x^-) - A^-(x - x^-) \\ &= F(x^+) - F(x^-) - A^+(x^+ - x^-) + (A^+ - A^-)(x - x^-) \\ &= (A^+ - A^-)(x - x^-), \end{aligned}$$

onde a última igualdade é devido a (1.9). Se definimos $s^- = x^+ - x^-$ e $z^- = F(x^+) - F(x^-)$ temos $A^+ s^- = z^-$. Se para qualquer $x \in \mathbb{R}^n$ ainda expressamos $x - x^- = \alpha s^- + t$, onde $t^T s^- = 0$, então o termo que desejamos minimizar se torna

$$M^+(x) - M^-(x) = \alpha(A^+ - A^-)s^- + (A^+ - A^-)t.$$

Podemos fazer o segundo termo se anular para todo $x \in \mathbb{R}^n$, tomando uma matriz A^+ com a propriedade de que $(A^+ - A^-)t = 0$ para todo vetor t ortogonal a s^- . Isto faz com que $A^+ - A^- = u(s^-)^T$ para algum vetor $u \in \mathbb{R}^n$, isto é, que a diferença $A^+ - A^-$ seja uma matriz de posto um. Logo $(A^+ - A^-)s^- = z^- - A^-s^-$ e, conseqüentemente, $u = (z^- - A^-s^-)/[(s^-)^T s^-]$. Assim

$$A^+ = A^- + \frac{(z^- - A^-s^-)(s^-)^T}{(s^-)^T s^-} \quad (1.10)$$

é uma fórmula para obtenção de A^+ a partir de A^- , consistente com a equação secante e que implica na menor quantidade possível de mudanças para o modelo M^- (Lema 8.1.1 em (DENNIS; SCHNABEL, 1996, p. 171)). Esta fórmula é conhecida como **atualização de Broyden**. A dificuldade com (1.10) é que a matriz A^+ obtida não necessariamente é uma matriz simétrica, mesmo que a matriz A^- utilizada em seu cálculo seja simétrica. Isto não é conveniente, porque muitas vezes a usamos para atualizar aproximações da hessiana de uma função $f : \mathbb{R}^n \rightarrow \mathbb{R}$ duas vezes diferenciável, que são matrizes simétricas. Fórmulas alternativas para (1.10) que resultam em matrizes simétricas são as seguintes:

- **Atualização simétrica de Powell-Broyden (PSB):**

$$A^+ = A^- + \frac{(z^- - A^-s^-)(s^-)^T + s^-(z^- - A^-s^-)^T}{(s^-)^T s^-} - \frac{[(z^- - A^-s^-)^T s^-]s^-(s^-)^T}{[(s^-)^T s^-]^2};$$

- **Atualização de Broyden-Fletcher-Goldfarb-Shanno (BFGS):**

$$A^+ = A^- + \frac{z^-(z^-)^T}{(z^-)^T s^-} - \frac{A^-s^-(s^-)^T A^-}{(s^-)^T A^-s^-};$$

- **Atualização de Davidon-Fletcher-Powell (DFP):**

$$A^+ = A^- + \frac{(z^- - H^-s^-)(z^-)^T + z^-(z^- - H^-s^-)^T}{(z^-)^T s^-} - \frac{[(z^- - A^-s^-)^T z^-]z^-(z^-)^T}{[(z^-)^T s^-]^2}.$$

Sob certas condições, podemos mostrar que algoritmos de determinação de zeros de funções ou de minimização irrestrita de funções reais baseados nas fórmulas de atualização (1.10), PSB, BFGS e DFP têm convergência local superlinear (Teoremas 8.2.2, 9.1.2 e 9.3.1 em (DENNIS; SCHNABEL, 1996, p. 177, 197 e 206)).

1.2 Otimização com restrições de igualdade

Um problema de otimização com uma função objetivo quadrática e restrições lineares é chamado **problema de programação quadrática**. Problemas deste tipo são bastante comuns, muitas vezes por corresponderem a subproblemas em métodos mais gerais de otimização com restrições. Logo o estudo específico deles é relevante.

Considere uma matriz simétrica $A \in \mathbb{R}^{n \times n}$, dois conjuntos finitos de índices \mathcal{E} e \mathcal{I} , vetores a , x e g_i , $i \in \mathcal{E} \cup \mathcal{I}$, em \mathbb{R}^n e escalares reais b_i , $i \in \mathcal{E} \cup \mathcal{I}$. Então o problema de programação quadrática, em sua forma mais geral, pode ser assim apresentado:

$$\min_{x \in \mathbb{R}^n} q(x) = \frac{1}{2}x^T A x - x^T a \quad \text{sujeita a} \quad g_i^T x = b_i, \quad i \in \mathcal{E}, \quad \text{e} \quad g_i^T x \leq b_i, \quad i \in \mathcal{I}. \quad (1.11)$$

Trata-se de um problema cujo esforço computacional necessário à sua resolução depende fortemente das características da função objetivo e do número de restrições presentes. Por exemplo: (1.11) tem resolução relativamente simples quando A é uma matriz definida positiva e $\mathcal{E} = \emptyset$, porque se existe um minimizador local viável para a função objetivo neste caso, ele é único e é também um minimizador global de q (Proposição 3.1.1 em (BERTSEKAS, 2009, p. 117)). Resolver o problema acima quando A é uma matriz indefinida pode, por outro lado, ser bem mais desafiador, já que daí existe a possibilidade de (1.11) admitir muitos pontos estacionários e minimizadores locais ((NOCEDAL; WRIGHT, 2006, p. 449)). Obviamente a dificuldade pode ser ainda maior, se A é uma matriz singular. Portanto, algum cuidado sempre tem de ser tomado com relação às hipóteses incluídas em um tal problema. Assumindo que ele admite uma solução, supomos que LICQ se verifica em todos os seus minimizadores locais.

O problema (1.11) pode ser resolvido iterativamente com estratégias que exploram o conjunto das restrições ativas $\mathcal{A}(x)$. Curiosamente, cada uma dessas iterações também requer a resolução de um problema de programação quadrática. No entanto, desta vez apenas com restrições de igualdade. Simplificamos esta breve introdução, analisando somente problemas como estes últimos, que pelo que acabamos de explicar podem ser vistos como elementos estruturais para a resolução dos primeiros. Suponha, por isso, que somente restrições de igualdade são impostas a (1.11) e que $\mathcal{E} = \{1, \dots, m\}$, com $m \leq n$. Assim o problema de programação quadrática se reduz a

$$\min_{x \in \mathbb{R}^n} q(x) = \frac{1}{2}x^T A x - x^T a \quad \text{sujeita a} \quad Bx = b, \quad (1.12)$$

onde $B \in \mathbb{R}^{m \times n}$ é a matriz de linhas g_1^T, \dots, g_m^T e $b \in \mathbb{R}^m$, o vetor de componentes b_1, \dots, b_m .

Definindo $c_i(x) = g_i^T x - b_i$, $i \in \mathcal{E}$, encontramos que $B = B(x^*)$ para qualquer minimizador local x^* de (1.12). Consequentemente, descobrimos que B tem posto-linha completo, já que LICQ também se verifica em x^* . Além disso, vemos com esta definição que a função lagrangiana para o problema é

$$(x, y) \in \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathcal{L}(x, y) = \frac{1}{2}x^T A x - x^T a + y^T (Bx - b).$$

Portanto as condições de KKT para a solução x^* de (1.12) se resumem ao seguinte sistema de equações lineares, conhecido no contexto de otimização como **sistema KKT**:

$$M \begin{bmatrix} x^* \\ y^* \end{bmatrix} := \begin{bmatrix} A & B^T \\ B & O \end{bmatrix} \begin{bmatrix} x^* \\ y^* \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix}. \quad (1.13)$$

Aqui $y^* \in \mathbb{R}^m$ representa o vetor de multiplicadores de Lagrange com o qual as condições de KKT são satisfeitas em (x^*, y^*) . Neste sentido, dizemos que y^* é o vetor de multiplicadores de Lagrange associado à x^* . Quando é conveniente, também identificamos um par ordenado da forma $(x, y) \in \mathbb{R}^n \times \mathbb{R}^m$ com um único vetor de \mathbb{R}^{n+m} . O símbolo $:=$ em (1.13) significa que $M \in \mathbb{R}^{(n+m) \times (n+m)}$ é igual a matriz quadrada à sua direita. Esta matriz é simétrica, porém frequentemente mal-condicionada ((BENZI; GOLUB; LIESEN, 2005, p. 27)) e sempre indefinida.

Definição 1.5. A **assinatura** de uma matriz quadrada C , denotada por $\text{sign}(C)$, é a terna ordenada (n_+, n_-, n_0) que indica os números de autovalores positivos, negativos e nulos de C , respectivamente, n_+ , n_- e n_0 .

Proposição 1.8. Se $\text{rank}(B) = m$ e $K \in \mathbb{R}^{n \times (n-m)}$ é uma matriz cujas colunas geram o $\ker(B)$, então $\text{sign}(M) = \text{sign}(K^T A K) + (m, m, 0)$.

Demonstração. Pode ser encontrada em (GOULD, 1985, p. 96). \square

Concluimos daí que a não-singularidade da matriz M é equivalente à não-singularidade da matriz $K^T A K$. Em particular, inferimos que existe uma única solução (x^*, y^*) para o sistema linear (1.13) que satisfaz as condições necessárias de primeira ordem para o problema (1.12), se A é uma matriz definida positiva no $\ker(B)$. Sob esta hipótese, concluimos também que as condições suficientes de segunda ordem são satisfeitas por (x^*, y^*) . Isto implica que x^* é o único minimizador global do problema de programação quadrática com restrições de igualdade.

Teorema 1.9. Se B tem posto completo e A é uma matriz simétrica definida positiva no $\ker(B)$, então as n primeiras componentes do vetor (x^*, y^*) satisfazendo (1.13) compõem a única solução global de (1.12).

Demonstração. Pode ser encontrada em (NOCEDAL; WRIGHT, 2006, p. 453). \square

Este teorema sozinho já evidencia a importância dos sistemas KKT em otimização. Em especial, em abordagens que reduzem um problema de otimização à resolução de sucessivos problemas de programação quadrática, que são muitas e que representam uma fração significativa de todos os métodos de otimização existentes. Uma outra abordagem, bastante geral, que não depende explicitamente de problemas da forma (1.12) e que permite rápidas conclusões, ainda é indicada abaixo. Apontamos mais essa abordagem unicamente

para justificar que uma análise e desenvolvimento criteriosos de métodos de resolução de sistemas KKT é de suma importância para otimização, entre diversas outras áreas.

Pois bem, vamos considerar novamente o problema geral de otimização (1.2). Se aplicamos alguma estratégia de conjuntos ativos a ele, reduzimos a resolução do problema à resolução de um esquema iterativo, que é uma sequência de subproblemas de restrições de igualdade,

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{sujeita a} \quad c_i(x) = 0, \quad i \in \mathcal{E}^{(\ell)} \subset \mathcal{E} \cup \mathcal{I}; \quad \ell \in \mathbb{N}, \quad (1.14)$$

onde a solução de cada subproblema é ponderada para se decidir o conjunto de restrições ativas da iteração seguinte. Denotamos a solução do ℓ -ésimo subproblema por $x^{((\ell))}$. Fica a cargo da própria estratégia de conjuntos ativos aceitar uma solução $x^{((\ell))} \in \mathbb{R}^n$ e prosseguir com a resolução de outro subproblema, ou rejeitá-la e modificar o subproblema corrente até que se encontre uma solução dele que cause um descenso apropriado no valor da função f . A estratégia faz isso de modo a forçar a sequência $(x^{((\ell))})$ das soluções dos subproblemas a convergir para uma solução x^* do problema original.

Desejamos resolver cada subproblema de (1.14) com o auxílio de sistemas KKT, porém sem transformá-los diretamente em uma sequência de problemas de programação quadrática. Sem perda de generalidade, assumimos para isso que $\mathcal{E}^{(\ell)} = \{1, \dots, m\}$ na ℓ -ésima iteração do esquema indicado e fazemos $c(x) = (c_1(x), \dots, c_m(x))$. A função lagrangiana para o subproblema correspondente a esta iteração é, portanto,

$$(x, y) \in \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathcal{L}(x, y) = f(x) + y^T c(x).$$

Segue daí e da condição necessária de primeira ordem que, para todo minimizador local $x^{((\ell))} \in \mathbb{R}^n$ do subproblema no qual LICQ se verifica, existe um vetor $y^{((\ell))} \in \mathbb{R}^m$ tal que

$$F(x^{((\ell))}, y^{((\ell))}) = \begin{bmatrix} \nabla f(x^{((\ell))}) + B(x^{((\ell))})^T y^{((\ell))} \\ c(x^{((\ell))}) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

onde $B(x^{((\ell))})$ é como em (1.3). Isto quer dizer que $(x^{((\ell))}, y^{((\ell))})$ é um zero de uma função $F : \mathbb{R}^{n+m} \rightarrow \mathbb{R}^{n+m}$. Logo podemos empregar o método de Newton para tentar encontrá-lo. Se fazemos isso a partir de um ponto arbitrário $(x^{(0)}, y^{(0)}) \in \mathbb{R}^{n+m}$, geramos uma sequência $(x^{(k)}, y^{(k)})$ de pontos em \mathbb{R}^{n+m} , definida a partir da relação

$$\begin{bmatrix} x^{(k+1)} \\ y^{(k+1)} \end{bmatrix} = \begin{bmatrix} x^{(k)} \\ y^{(k)} \end{bmatrix} + \begin{bmatrix} d_x^{(k)} \\ d_y^{(k)} \end{bmatrix}, \quad k \in \mathbb{N}_0,$$

onde $(d_x^{(k)}, d_y^{(k)}) \in \mathbb{R}^{n+m}$ é solução do sistema KKT

$$\begin{bmatrix} \nabla_{xx}^2 \mathcal{L}(x^{(k)}, y^{(k)}) & B(x^{(k)})^T \\ B(x^{(k)}) & O \end{bmatrix} \begin{bmatrix} d_x^{(k)} \\ d_y^{(k)} \end{bmatrix} = - \begin{bmatrix} \nabla f(x^{(k)}) + B(x^{(k)})^T y^{(k)} \\ c(x^{(k)}) \end{bmatrix}. \quad (1.15)$$

Vamos supor que consigamos garantir que a matriz $B(x^{(k)})$ tem posto completo para todo $k \in \mathbb{N}_0$. Então:

- se o sistema acima admite uma única solução, nós somos capazes de encontrá-la — aproveitando a estrutura especial do sistema linear — com o método de resolução de sistemas KKT que desenvolveremos no capítulo seguinte, até mesmo quando $\nabla_{xx}^2 \mathcal{L}(x^{(k)}, y^{(k)})$ é uma matriz singular;
- se o sistema não admite uma única solução, podemos torná-lo possível e determinado substituindo o bloco $(1, 1)$ de sua matriz de coeficientes por

$$\nabla_{xx}^2 \mathcal{L}(x^{(k)}, y^{(k)}) + \mu_k I,$$

para praticamente qualquer valor real $\mu_k \neq 0$, em vista da Proposição 1.8, e daí resolvê-lo com o método citado.

Em qualquer caso, obtemos um vetor $(d_x^{(k)}, d_y^{(k)})$ e, por conseguinte, $(x^{(k+1)}, y^{(k+1)})$. Portanto uma possível singularidade da matriz jacobiana de F não é nenhuma complicação séria. Um possível mau-condicionamento da matriz de coeficientes dos sistemas KKT, devido majoritariamente ao mau-condicionamento de seu bloco $(1, 1)$, também não representa uma grande dificuldade, porque o método a ser desenvolvido tratará de minimizar esse mau-condicionamento. Por causa disso, inclusive, a solução de qualquer um desses sistemas, perturbado com uma constante μ_k não-nula, tal que $|\mu_k| \ll 1$, deve aproximar bem a solução esperada para o respectivo sistema não perturbado. Além disso, a expressão encontrada para os sistemas é vantajosa para a aplicação do método de resolução de sistemas KKT que desenvolveremos, porque a sequência $(c(x^{(k)}))$ converge para zero quando a sequência $(x^{(k)}, y^{(k)})$ converge para um zero de F . Deste modo, as últimas m componentes do vetor do lado direito dos sistemas vão se tornando cada vez menores, em valor absoluto, e quanto mais próximas de zero elas estiverem, menos sensível também será o método proposto a um possível mau-condicionamento da matriz $B(x^{(k)})$, conforme veremos futuramente.

Suponha agora que $(x^{(k)}, y^{(k)}) \rightarrow (x^{((\ell))}, y^{((\ell))})$ quando $k \rightarrow \infty$. Então

$$c(x^{((\ell))}) = c\left(\lim_{k \rightarrow \infty} x^{(k)}\right) = \lim_{k \rightarrow \infty} c(x^{(k)}) = 0,$$

devido à continuidade da função c . Portanto $x^{((\ell))}$ é um ponto estacionário viável para o subproblema da ℓ -ésima iteração de (1.14). Assim a estratégia de conjuntos ativos sendo empregada pode fazer uso de $x^{((\ell))}$ (e também do vetor $y^{((\ell))}$ de multiplicadores de Lagrange associado a ele) para ocasionar um descenso adequado no valor da função f e/ou escolher um novo subproblema com o qual seja possível continuar a execução do esquema (1.14). Se o conjunto viável do problema de otimização (1.2) é fechado, a sequência $(x^{((\ell))})$ obtida dessa maneira acaba convergindo para um ponto viável deste problema. Se ele não é fechado, o valor de f em um tal ponto limite pode ser bem aproximado pelo valor de f em qualquer ponto viável da sequência $(x^{((\ell))})$, com $\ell \in \mathbb{N}_0$ suficientemente grande. Seja qual for o caso, uma aproximação viável para solução do problema original sempre é encontrada.

Na realidade, a determinação de um ponto $x^{((\ell))}$ ocorre quando simplesmente a direção $(d_x^{(k)}, d_y^{(k)})$ fica muito pequena em norma. Neste momento fazemos

$$x^{((\ell))} = x^{(k+1)} = x^{(k)} + d_x^{(k)}.$$

Apesar disso, ainda conseguimos garantir a viabilidade do ponto encontrado, sob uma módica condição para o problema (1.14). Vejamos que este de fato é o caso, considerando uma precisão $\epsilon > 0$ qualquer e assumindo para este problema que a matriz jacobiana das restrições ativas é limitada em uma vizinhança de $x^{((\ell))}$, isto é, que $\|B(x)\|_2 \leq \Lambda$ para todo $x \in \mathbb{R}^n$ tal que $\|x - x^{((\ell))}\|_2 < \delta$, onde Λ e δ são constantes positivas. Como também estamos supondo que a sequência $(x^{(k)}, y^{(k)})$ é convergente para $(x^{((\ell))}, y^{((\ell))})$, temos que

$$\|x^{(k+1)} - x^{((\ell))}\|_2 < \delta \quad \text{e} \quad \|d_x^{(k+1)}\|_2 < \epsilon/\Lambda$$

para todo k suficientemente grande. De (1.15), temos ainda que

$$c(x^{(k+1)}) = -B(x^{(k+1)})d_x^{(k+1)}.$$

Assim $x^{((\ell))}$ é um ponto viável para a precisão considerada, como afirmamos antes, pois

$$\|c(x^{((\ell))})\|_2 \leq \|B(x^{(k+1)})\|_2 \|d_x^{(k+1)}\|_2 < \Lambda \cdot (\epsilon/\Lambda) = \epsilon.$$

Se permitirmos que a estratégia de conjuntos ativos faça uso de pontos possivelmente não-viáveis na determinação da solução x^* de (1.2), então esta hipótese sobre a matriz jacobiana das restrições ativas pode ainda ser relaxada para: $B(x)$ limitada apenas em uma vizinhança de x^* . Realmente, pois a estratégia em questão cuida para que os iterandos $x^{((1))}, x^{((2))}, \dots, x^{((\ell))}, \dots$ sejam obtidos cada vez mais próximos de x^* e, em algum momento, eles passam a pertencer à região em que $B(x)$ é limitada. Daí o raciocínio anterior pode ser aplicado e isso nos deixa concluir que todo iterando $x^{((\ell))}$, com ℓ suficientemente grande, é viável. Claramente é somente isso que precisamos que aconteça para poder encontrar uma aproximação viável da solução do problema geral de otimização com restrições. Notamos que nem mesmo esta segunda hipótese é necessária, quando já assumimos (com o propósito de fazer o método de Newton convergir quadraticamente) que o jacobiano J_F da função F é Lipschitz contínuo em uma vizinhança de (x^*, y^*) . Digamos, em uma bola centrada neste ponto e de raio $\delta > 0$. De fato, se $L > 0$ é a constante de Lipschitz de J_F e

$$P_1 = \begin{bmatrix} I & O^T \\ O & O \end{bmatrix} \quad \text{e} \quad P_2 = \begin{bmatrix} O & O^T \\ O & I \end{bmatrix}$$

são duas matrizes idempotentes de projeção (isto é, $P_1^2 = P_1$ e $P_2^2 = P_2$), então

$$\begin{aligned} \|B(x) - B(x^*)\|_2 &= \|P_2[J_F(x, y^*) - J_F(x^*, y^*)]P_1\|_2 \\ &\leq \|J_F(x, y^*) - J_F(x^*, y^*)\|_2 \\ &\leq L\|(x, y^*) - (x^*, y^*)\|_2 \\ &< \delta L \end{aligned}$$

para todo $x \in \mathbb{R}^n$ tal que $\|x - x^*\|_2 < \delta$. Temos assim que

$$\|B(x)\|_2 < \delta L + \|B(x^*)\|_2 = \Lambda$$

e que Λ é finito, se $\|B(x^*)\|_2$ o é. Consequentemente, a matriz jacobiana das restrições ativas é limitada ao redor da solução x^* , como gostaríamos mesmo que fosse.

Se a estratégia de conjuntos ativos puder aproveitar o ponto $x^{((\ell))}$ calculado na ℓ -ésima iteração de (1.14) quase imediatamente (isto é, sem ter de substituí-lo muitas vezes por um outro ponto $x^{((\ell))}$ que, este sim, ocasione um decréscimo adequado no valor de f), então a eficiência do algoritmo no qual ela se insere deve ser proporcional à eficiência média de obtenção de um tal ponto. Como obtemos este ponto com o método de Newton, isso se traduz em um algoritmo com convergência superlinear, ou até mesmo quadrática. O maior entrave à eficiência de um algoritmo dessa natureza claramente é, então, a construção e resolução dos sistemas KKT nos subproblemas. Adiantamos que experimentos numéricos realizados com o nosso método de resolução de sistemas KKT sugerem que, sob condições adequadas, nosso método é competitivo para resolução de sistemas lineares de ordem até mais ou menos 15000, inclusive com relação ao tempo gasto para se determinar a solução dos sistemas. Portanto o maior empecilho à eficiência do algoritmo em questão é realmente somente a construção dos sistemas KKT. Por isso sugerimos abaixo uma abordagem para contornar esta dificuldade.

Vamos assumir que as avaliações das funções f , ∇f , c_i e ∇c_i , $i \in \mathcal{E}^{(\ell)}$, são todas computacionalmente baratas de serem realizadas. Então o custo para se obter um sistema como (1.15) é em sua maior parte devido ao cálculo da hessiana da função lagrangiana:

$$\nabla_{xx}^2 \mathcal{L}(x^{(k)}, y^{(k)}) = \nabla^2 f(x^{(k)}) + \sum_{i=1}^m y_i^{(k)} \nabla^2 c_i(x^{(k)}).$$

Uma ideia natural para reduzir esse custo é aproximar $\nabla_{xx}^2 \mathcal{L}(x^{(0)}, y^{(0)})$ de alguma maneira simples e daí encontrar aproximações para qualquer outra matriz hessiana $\nabla_{xx}^2 \mathcal{L}(x^{(k)}, y^{(k)})$, $k \in \mathbb{N}$, com base em sucessivas atualizações baratas desta matriz inicial. É importante que as aproximações levem em conta o somatório indicado na expressão das hessianas, já que é aquela parcela que incorpora nos sistemas KKT informação sobre a curvatura das restrições e assim acelera a resolução dos subproblemas em (1.14). Pois bem, sejam dadas as seguintes aproximações em $\mathbb{R}^{n \times n}$,

$$\begin{aligned} \mathcal{A}^{--} &\approx \nabla_{xx}^2 \mathcal{L}(x^-, y^-), & \mathcal{A}^{++} &\approx \nabla_{xx}^2 \mathcal{L}(x^+, y^+), \\ H_{\Sigma}^{--} &\approx \sum_{i=1}^m y_i^- \nabla^2 c_i(x^-), & H_{\Sigma}^{+-} &\approx \sum_{i=1}^m y_i^+ \nabla^2 c_i(x^-), & H_{\Sigma}^{++} &\approx \sum_{i=1}^m y_i^+ \nabla^2 c_i(x^+), \end{aligned}$$

e, também,

$$A^- \approx \nabla^2 f(x^-), \quad A^+ \approx \nabla^2 f(x^+), \quad H_i^- \approx \nabla^2 c_i(x^-) \quad \text{e} \quad H_i^+ \approx \nabla^2 c_i(x^+), \quad i \in \mathcal{E}^{(\ell)},$$

onde as matrizes A^+ e H_1^+, \dots, H_m^+ satisfazem as equações secantes para as funções f e c_1, \dots, c_m , respectivamente,

$$A^+(x^+ - x^-) = \nabla f(x^+) - \nabla f(x^-) \quad \text{e} \quad H_i^+(x^+ - x^-) = \nabla c_i(x^+) - \nabla c_i(x^-), \quad i \in \mathcal{E}^{(\ell)}.$$

Se as aproximações H_i^- e H_i^+ , $i \in \mathcal{E}^{(\ell)}$, são boas, ou seja, se a distância delas às funções que elas aproximam é menor do que uma dada precisão, então é razoável definir

$$H_{\Sigma}^{--} = \sum_{i=1}^m y_i^- H_i^-, \quad H_{\Sigma}^{+-} = \sum_{i=1}^m y_i^+ H_i^- \quad \text{e} \quad H_{\Sigma}^{++} = \sum_{i=1}^m y_i^+ H_i^+.$$

Porém não desejamos obter H_{Σ}^{++} a partir das atualizações H_i^+ de cada H_i^- , $i \in \mathcal{E}^{(\ell)}$, porque esta ainda é uma tarefa que pode ser computacionalmente muito cara de ser realizada. Uma alternativa melhor seria determinar a equação secante necessária para atualizar H_{Σ}^{--} diretamente. Mas isso é impossível, pois exigiria uma matriz H_{Σ}^{++} de ordem $n + m$. Ao invés disso, observamos então que

$$\begin{aligned} H_{\Sigma}^{++}(x^+ - x^-) &= \sum_{i=1}^m y_i^+ [H_i^+(x^+ - x^-)] \\ &= \sum_{i=1}^m y_i^+ [\nabla c_i(x^+) - \nabla c_i(x^-)] \\ &= [B(x^+) - B(x^-)]^T y^+ \end{aligned}$$

é a equação secante para atualizar H_{Σ}^{+-} a partir de H_{Σ}^{--} . Seja $(d_y^-)_i$ a i -ésima componente do vetor d_y^- . Como

$$H_{\Sigma}^{+-} = \sum_{i=1}^m (y_i^- + (d_y^-)_i) H_i^- = H_{\Sigma}^{--} + \sum_{i=1}^m (d_y^-)_i H_i^-$$

e $d_y^{(k)} \rightarrow 0$, quando $k \rightarrow \infty$, porque estamos assumindo que a sequência $(x^{(k)}, y^{(k)})$ converge para um zero de F , devemos ter

$$H_{\Sigma}^{+-} \approx H_{\Sigma}^{--}$$

para todo $k \in \mathbb{N}_0$ suficientemente grande. Assim é viável substituir H_{Σ}^{+-} por H_{Σ}^{--} nas expressões para obtenção da matriz H_{Σ}^{++} (dadas em função de H_{Σ}^{+-}), desde a primeira atualização, já que o erro que cometemos ao fazer isso tende muito rapidamente a zero. Se as fórmulas de atualização PSB, BFGS ou DFP são empregadas, por exemplo, conseguimos obter H_{Σ}^{++} de H_{Σ}^{--} com duas correções de posto um, o que tem um custo computacional perfeitamente aceitável.

Agora, se as aproximações A^- e A^+ também são boas, novamente, no sentido de que a distância delas às funções que elas aproximam é menor do que uma dada precisão, é razoável definir

$$\mathcal{A}^{--} = A^- + H_{\Sigma}^{--}$$

e querer atualizar esta matriz através das atualizações de suas parcelas, ou seja, fazer

$$\mathcal{A}^{++} = A^+ + H_{\Sigma}^{++}.$$

Isso é satisfatório porque, se A^- também é atualizada com fórmulas como PSB, BFGS ou DFP, então a matriz \mathcal{A}^{++} pode ser obtida com quatro correções de posto um, o que ainda tem um custo computacional aceitável. Além disso, \mathcal{A}^{++} eventualmente acaba satisfazendo a equação secante esperada para a função lagrangiana. De fato,

$$\begin{aligned} \mathcal{A}^{++}(x^+ - x^-) &= A^+(x^+ - x^-) + H_{\Sigma}^{++}(x^+ - x^-) \\ &= [\nabla f(x^+) - \nabla f(x^-)] + [B(x^+) - B(x^-)]^T y^+ \\ &= [\nabla f(x^+) + B(x^+)^T y^+] - [\nabla f(x^-) + B(x^-)^T (y^- + d_y^-)] \\ &= [\nabla_x \mathcal{L}(x^+, y^+) - \nabla_x \mathcal{L}(x^-, y^-)] - B(x^-)^T d_y^- \end{aligned}$$

e como $d_y^{(k)} \rightarrow 0$, quando $k \rightarrow \infty$, devemos ter

$$\mathcal{A}^{++}(x^+ - x^-) \approx \nabla_x \mathcal{L}(x^+, y^+) - \nabla_x \mathcal{L}(x^-, y^-)$$

para todo $k \in \mathbb{N}_0$ suficientemente grande. Desta forma é como se realmente passássemos a atualizar \mathcal{A}^{++} a partir de \mathcal{A}^{--} , porque o vetor do lado direito na equação acima corresponde às n primeiras componentes da diferença $F(x^+, y^+) - F(x^-, y^-)$, que é justamente o que poderíamos esperar para a equação secante da atualização considerada, que está sendo dada com base apenas no vetor x de incógnitas.

Como ponto de partida para todo este esquema, sugerimos inicialmente obter \hat{A}^- por diferenças finitas de $\nabla^2 f(x^{(0)})$ e daí fazer

$$A^- = \frac{1}{2} [\hat{A}^- + (\hat{A}^-)^T].$$

Sugerimos também tomar

$$H_{\Sigma}^{--} = B(x^{(0)})^T D B(x^{(0)}) = \sum_{i=1}^m d_i \nabla c_i(x^{(0)}) \nabla c_i(x^{(0)})^T,$$

onde $D = \text{diag}(d_1, \dots, d_m) \in \mathbb{R}^{m \times m}$ é uma matriz diagonal com algumas poucas entradas unitárias e não-nulas. Com isso fica completamente descrito um *framework* (Algoritmo 1) para aplicação do método de resolução de sistemas KKT que proporemos no Capítulo 2, em um contexto puramente de otimização. Observamos que trata-se realmente de um *framework*, porque não apontamos especificamente nenhuma estratégia de conjuntos ativos para ser empregada, tão pouco fixamos fórmulas de atualização para as matrizes A^+ e H_{Σ}^{++} , as quais, aliás, podem até mesmo ser distintas. Notamos também que o método de resolução de sistemas KKT em questão possui, na realidade, aplicações em diversas outras áreas além de otimização.

Algoritmo 1: Um esquema para a resolução do problema geral de otimização com restrições (1.2), através de um método de resolução de sistemas KKT.

Entrada: $(x^{(0)}, y^{(0)}) \in \mathbb{R}^{n+m}$, $\epsilon_1 > 0$ e $\epsilon_2 > 0$;

Fixe alguma estratégia de conjuntos ativos;

Faça $k \leftarrow -1$, $\ell \leftarrow 1$ e $flag \leftarrow 1$;

repita

se $flag = 1$ **então**

 Escolha $\mathcal{E}^{(\ell)}$ com a estratégia de conjuntos ativos fixada;

fim

repita

 Faça $k \leftarrow k + 1$;

 Avalie $f(x^{(k)})$, $\nabla f(x^{(k)})$, $c(x^{(k)})$ e $B(x^{(k)})$ (esta última, como em (1.3));

se $k = 0$ **então**

 Obtenha $A^{(0)}$ por diferenças finitas de $\nabla^2 f(x^{(0)})$;

 Escolha $d \in \mathbb{R}^m$ com poucas componentes unitárias e as demais nulas;

 Faça $H_\Sigma^{(0)} \leftarrow B(x^{(0)})^T D B(x^{(0)})$, onde $D = \text{diag}(d)$;

senão

 Obtenha $A^{(k)}$ e $H_\Sigma^{(k)}$ atualizando $A^{(k-1)}$ e $H_\Sigma^{(k-1)}$, respectivamente (por exemplo, com as fórmulas PSB, BFGS ou DFP);

fim

 Faça $\mathcal{A}^{(k)} \leftarrow A^{(k)} + H_\Sigma^{(k)}$;

 Resolva o sistema (1.15), com bloco $\nabla_{xx}^2 \mathcal{L}(x^{(k)}, y^{(k)})$ substituído por $\mathcal{A}^{(k)}$;

 Faça $x^{(k+1)} \leftarrow x^{(k)} + d_x^{(k)}$ e $y^{(k+1)} \leftarrow y^{(k)} + d_y^{(k)}$;

até $\|(d_x^{(k)}, d_y^{(k)})\|_2 < \epsilon_1$;

 Faça $x^{((\ell))} \leftarrow x^{(k+1)}$ e $y^{((\ell))} \leftarrow y^{(k+1)}$;

se $\|c(x^{((\ell))})\|_2 < \epsilon_2$ e $x^{((\ell))}$ ocasionar um descenso adequado no valor de f **então**

 Faça $(x^{(0)}, y^{(0)}) \leftarrow (x^{((\ell))}, y^{((\ell))})$, $k \leftarrow -1$, $\ell \leftarrow \ell + 1$ e $flag \leftarrow 1$;

senão

 Deixe a estratégia de conjuntos ativos modificar o subproblema corrente;

 Faça $flag \leftarrow 0$;

fim

até a sequência $(x^{((\ell))})$ convergir;

Saída: Uma aproximação da solução $x^* \in \mathbb{R}^n$ do problema (1.2).

Não encontramos na literatura referências para este *framework*. No entanto, ainda não podemos garantir que este não tenha sido anteriormente explorado de alguma maneira. Assim preferimos focar a tese na apresentação e discussão de um método de resolução dos elementos estruturais do *framework*, que são os próprios sistemas KKT.

1.3 Métodos clássicos de resolução de sistemas KKT

Nas próximas subseções apresentamos alguns métodos clássicos de resolução de sistemas KKT, que mais adiante se mostrarão úteis para a presente tese. A notação

que empregamos aqui para os blocos e vetores desses sistemas é a mesma da equação (1.13). Maiores detalhes dos métodos podem ser vistos em (NOCEDAL; WRIGHT, 2006, p. 455-463).

1.3.1 O método do complemento de Schur

Vamos assumir que A , além de ser uma matriz simétrica, é também definida positiva. Então, se multiplicamos a primeira equação de (1.13) por BA^{-1} e subtraímos do resultado a segunda equação, obtemos um sistema linear apenas no vetor de variáveis y^* :

$$(BA^{-1}B^T)y^* = BA^{-1}a - b. \quad (1.16)$$

Determinamos a solução y^* deste sistema, que tem matriz de coeficientes simétrica definida positiva, já que B tem posto completo, e com ela encontramos x^* a partir da primeira equação do sistema KKT, resolvendo

$$Ax^* = a - B^T y^*. \quad (1.17)$$

Fica assim conhecida a solução de (1.13).

Por definição, a matriz de coeficientes do sistema (1.16) é o negativo do **complemento de Schur** $S = -BA^{-1}B^T$ de A em M . O nome do método deriva daí. O procedimento acima faz uso da ação de A^{-1} e requer o cálculo da fatoração de Cholesky de $-S$, que é uma matriz de ordem m . Portanto, ele é recomendado quando A é uma matriz bem-condicionada e facilmente inversível ou quando o número m de restrições de igualdade para (1.12) é pequeno e $-S$ pode ser formada com poucas operações de ponto flutuante.

Podemos aplicar eliminação gaussiana à matriz de coeficientes de (1.13), utilizando A como bloco pivô, para encontrar a expressão explícita da inversa de M :

$$\begin{bmatrix} A & B^T \\ B & O \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} + A^{-1}B^T S^{-1}BA^{-1} & -A^{-1}B^T S^{-1} \\ -S^{-1}BA^{-1} & S^{-1} \end{bmatrix}.$$

A solução do sistema linear (1.13) pode então também ser obtida multiplicando-se o vetor do seu lado direito por M^{-1} . Se aproveitamos as expressões comuns a esta multiplicação e agrupamos os termos apropriadamente, conseguimos recobrar o procedimento descrito pelas equações (1.16) e (1.17). Ou seja: o próprio método do complemento de Schur. De fato, pois a multiplicação do vetor (a, b) do sistema (1.13) pela matriz acima fornece

$$\begin{bmatrix} x^* \\ y^* \end{bmatrix} = \begin{bmatrix} A^{-1}[a + B^T S^{-1}(BA^{-1}a - b)] \\ -S^{-1}(BA^{-1}a - b) \end{bmatrix}$$

e as ações mais externas das inversas de S e A nestas expressões correspondem, numericamente, às resoluções dos sistemas lineares (1.16) e (1.17), nesta ordem.

1.3.2 Os métodos de espaço nulo

Os métodos de espaço nulo têm maior aplicabilidade do que o método do complemento de Schur na resolução de sistemas KKT, porque apenas requerem que a matriz A seja definida positiva em um subespaço de \mathbb{R}^n , a saber: o $\ker(B)$. Trata-se de uma exigência de algum modo natural à otimização, em vista dos Teoremas 1.6 e 1.9.

Para serem aplicados, os métodos de espaço nulo dependem tanto do conhecimento prévio de uma matriz $K \in \mathbb{R}^{n \times (n-m)}$ de colunas que geram o $\ker(B)$, quanto de uma solução particular $\dot{x} \in \mathbb{R}^n$ do sistema linear $Bx = b$, $x \in \mathbb{R}^n$. Essencialmente, todos eles variam na maneira como obtêm estes objetos.

Seja $Q \in \mathbb{R}^{n \times n}$ o fator ortogonal da fatoração QR de B^T . Escolhas simples de serem descritas para \dot{x} e K são, por exemplo, $\dot{x} = B^T(BB^T)^{-1}b$ e K igual a submatriz de Q correspondente às suas $n - m$ últimas colunas.

Utilizamos a solução particular $\dot{x} \in \mathbb{R}^n$ do sistema linear $Bx = b$ fornecida para colocar qualquer outra solução do sistema na forma $x = K\tilde{x} + \dot{x}$, onde $\tilde{x} \in \mathbb{R}^{n-m}$. Isto é possível porque o produto $BK = O$, já que o $\ker(B)$ é um espaço ortogonal à $\text{ran}(B^T)$. Assim conseguimos escrever $x^* = K\tilde{x}^* + \dot{x}$ para algum $\tilde{x}^* \in \mathbb{R}^{n-m}$. Procuramos resolver o sistema KKT, inicialmente substituindo x^* na equação $Ax^* + B^Ty^* = a$, que fornece $A(K\tilde{x}^* + \dot{x}) = a - B^Ty^*$. Então a pré-multiplicamos por K^T para encontrar o sistema linear reduzido, de ordem $n - m$:

$$(K^TAK)\tilde{x}^* = K^T(a - A\dot{x}). \quad (1.18)$$

Este sistema é possível e determinado, porque A é uma matriz definida positiva no $\ker(B)$ por hipótese. Determinamos sua solução e a substituímos na expressão de x^* , que passa então a ser conhecido. Em seguida voltamos a substituir x^* em $Ax^* + B^Ty^* = a$, mas desta vez pré-multiplicamos o resultado por B . Encontramos com isso um outro sistema linear reduzido, agora de ordem m :

$$(BB^T)y^* = B(a - Ax^*). \quad (1.19)$$

Assim como o anterior, este sistema também admite uma única solução, por ter uma matriz de coeficientes definida positiva. Obtemos sua solução e, com ela, a solução (x^*, y^*) de (1.13). Esta estratégia descreve a família de métodos de espaço nulo, em função das escolhas de \dot{x} e K .

A maior dificuldade para aplicação dos métodos de espaço nulo é a exigência de uma base para o $\ker(B)$, por meio da matriz K . São inúmeras as razões para isso: há várias escolhas possíveis para K ; algumas escolhas resultam em uma matriz K^TAK muito mal-condicionada; outras consistem de uma matriz K de posto numericamente deficiente; para alguns problemas grandes, a determinação de uma tal K é uma tarefa bastante onerosa; em particular, pode ser impraticável se calcular uma base ortonormal

do $\ker(B)$ quando B é grande e esparsa; nem sempre existe uma base esparsa do $\ker(B)$, mesmo quando B é esparsa; e se existe, encontrar uma matriz K com um número mínimo de entradas não-nulas é um problema NP-difícil; entre outras ((BENZI; GOLUB; LIESEN, 2005, p. 32-40) e (NOCEDAL; WRIGHT, 2006, p. 458-459)).

Apesar de todas as complicações citadas, métodos de espaço nulo são muito utilizados em otimização, sendo especialmente interessantes quando a diferença $n - m$ é pequena. Neste caso é comum empregá-los em aplicações que requeiram a solução de uma sequência de sistemas KKT,

$$\begin{bmatrix} A^{(k)} & B^T \\ B & O \end{bmatrix} \begin{bmatrix} (x^*)^{(k)} \\ (y^*)^{(k)} \end{bmatrix} = \begin{bmatrix} a^{(k)} \\ b^{(k)} \end{bmatrix}, \quad k \in \mathbb{N},$$

onde a matriz $A^{(k)} \in \mathbb{R}^{n \times n}$ e o vetor $(a^{(k)}, b^{(k)}) \in \mathbb{R}^{n+m}$ variam com k , mas B se mantém fixa. Porém não podemos perder de vista que frequentemente $m \ll n$ em situações práticas ((BENZI; GOLUB; LIESEN, 2005, p. 5)). Por fim, observamos que as performances dos métodos do complemento de Schur e de espaço nulo variam muito, mesmo para problemas de mesma dimensão ((NOCEDAL; WRIGHT, 2006, p. 459)).

1.3.3 O método dos gradientes conjugados projetados

Um sistema KKT obviamente também pode ser resolvido com métodos iterativos. Algumas possibilidades nesta categoria são métodos iterativos derivados de métodos de espaço nulo pela aplicação do método dos gradientes conjugados ao sistema reduzido (1.18). Introduzimos adiante um desses métodos, fixando *a priori* as escolhas de alguns parâmetros dele, a saber: de uma base da $\text{ran}(B^T)$, de uma solução particular do sistema $Bx = b$ e de uma matriz de projeção no $\ker(B)$. Fazemos isso com o propósito de facilitar a visualização de paralelos entre este método e a teoria que desenvolveremos nos Capítulos 2 e 3. Antes disso, porém, recordamos brevemente o método dos gradientes conjugados e suas principais propriedades. No que segue, usamos por um momento as letras A , b , q e x sem querer estabelecer qualquer relação com o que elas representam para (1.12) e (1.13).

O método dos gradientes conjugados é um método iterativo utilizado para se resolver um sistema linear $Ax = b$ com matriz de coeficientes $A \in \mathbb{R}^{n \times n}$ simétrica definida positiva. Devido às condições suficientes de segunda ordem, determinar a solução deste sistema é equivalente à resolução do problema de minimização irrestrita

$$\min_{x \in \mathbb{R}^n} q(x) = \frac{1}{2} x^T A x - b^T x.$$

Por esta razão, o método dos gradientes conjugados também é visto como uma técnica de minimização de funções quadráticas convexas.

Partindo de um ponto qualquer $x^{(0)}$ e de uma direção específica e não-nula $d^{(0)}$, ambos em \mathbb{R}^n , a ideia básica por trás do método dos gradientes conjugados é sucessivamente

minimizar a função objetivo q em espaços afins

$$V^{(k)} = x^{(0)} + \text{span}\{d^{(0)}, \dots, d^{(k-1)}\}, \quad k \in \mathbb{N},$$

de dimensão k crescente, determinando a cada iteração uma nova direção $d^{(k)} \in \mathbb{R}^n \setminus \{0\}$, tal que $(d^{(i)})^T A d^{(k)} = 0$ para todo $i \neq k$. Pois então $V^{(n)} = \mathbb{R}^n$ e a solução de $Ax = b$ pode ser encontrada em no máximo n iterações.

Um conjunto de direções de \mathbb{R}^n que satisfazem a propriedade citada acima, com relação a uma dada matriz simétrica $A \in \mathbb{R}^{n \times n}$ definida positiva, é chamado **A -conjugado**. Facilmente se vê que é a A -conjugação de $\{d^{(0)}, \dots, d^{(k-1)}\}$ que garante que $\dim(V^{(k)}) = k$, uma vez que as direções são linearmente independentes neste caso.

Seja $(x^{(0)}, \dots, x^{(\ell)})$ a sequência de pontos de \mathbb{R}^n gerada pelas iterações do método. Definimos $r^{(k)} = Ax^{(k)} - b$ para todo $k \in \{0, \dots, \ell\}$ e tomamos

$$d^{(0)} = -r^{(0)}. \quad (1.20)$$

Como podemos resolver o sistema linear $Ax = b$ através de um problema de minimização, procuramos relacionar os iterandos pelas equações

$$x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}, \quad k \in \{0, \dots, \ell - 1\}, \quad (1.21)$$

onde $\alpha^{(0)}, \dots, \alpha^{(\ell-1)}$ são constantes reais a serem determinadas para atender a propriedade de minimização de q em $V^{(1)}, \dots, V^{(\ell)}$, respectivamente. Por sua vez, procuramos escolher cada direção $d^{(k+1)}$ como uma combinação linear de $-r^{(k+1)} = -\nabla q(x^{(k)})$ e $d^{(k)}$, a saber

$$d^{(k+1)} = -r^{(k+1)} + \beta_{k+1} d^{(k)}, \quad \beta_{k+1} \in \mathbb{R}, \quad (1.22)$$

com o intuito de provocar um rápido decrescimento da função objetivo e ainda permitir que o conjunto $\{d^{(0)}, \dots, d^{(\ell-1)}\}$ seja A -conjugado. Observando que

$$r^{(k+1)} = r^{(k)} + \alpha_k A d^{(k)}, \quad (1.23)$$

podemos mostrar que

$$\alpha_k = \frac{(r^{(k)})^T r^{(k)}}{(d^{(k)})^T A d^{(k)}} \quad \text{e} \quad \beta_{k+1} = \frac{(r^{(k+1)})^T r^{(k+1)}}{(r^{(k)})^T r^{(k)}} \quad (1.24)$$

para todo índice k apropriado (([NOCEDAL; WRIGHT, 2006](#), p. 101-112)). As relações (1.20)-(1.24) determinam completamente o método dos gradientes conjugados, que está bem-definido porque o denominador das constantes $\alpha_0, \dots, \alpha_{\ell-1}$ nunca se anula com a hipótese de que a matriz A é definida positiva.

A aplicação deste método à resolução de um sistema linear $Ax = b$ é recomendada quando a ordem da matriz A é grande, pois o método não depende de fatorações de A e ainda requer pouca memória para o cálculo das direções $d^{(0)}, \dots, d^{(\ell-1)}$. Além disso, ele pode convergir com bem poucas iterações, uma vez presente alguma distribuição adequada dos autovalores de A na reta real.

Teorema 1.10. *Se A tem autovalores $\lambda_1 \leq \dots \leq \lambda_n$, então*

$$\|x^{(k+1)} - x^*\|_A^2 \leq \left(\frac{\lambda_{n-k} - \lambda_1}{\lambda_{n-k} + \lambda_1} \right)^2 \|x^{(0)} - x^*\|_A^2,$$

onde $\|\cdot\|_A$ denota a norma em \mathbb{R}^n induzida pela matriz A .

Demonstração. Pode ser encontrada em (NOCEDAL; WRIGHT, 2006, p. 115). \square

Compreendemos facilmente este teorema supondo que $n - \ell$ dos autovalores de A estão agrupados em um intervalo $(1 - \epsilon, 1 + \epsilon) \subset \mathbb{R}$ de raio $\epsilon > 0$ e que os demais autovalores de A são relativamente grandes em comparação a estes. Pois daí

$$\|x^{(\ell+1)} - x^*\|_A \approx \epsilon \|x^{(0)} - x^*\|_A,$$

se $\epsilon \ll 1$, já que $\lambda_{n-\ell} - \lambda_1 < 2\epsilon$ e $\lambda_{n-\ell} + \lambda_1 \approx 2$. Logo o método dos gradientes conjugados aproximará bem a solução de $Ax = b$ após ℓ iterações, dado ϵ significativamente pequeno.

Outro resultado de convergência para o método dos gradientes conjugados é baseado no número de condição da matriz A .

Teorema 1.11. *Nas condições do teorema anterior temos*

$$\|x^{(k)} - x^*\|_A \leq 2 \left(\frac{\sqrt{\text{cond}_2(A)} - 1}{\sqrt{\text{cond}_2(A)} + 1} \right)^{(k)} \|x^{(0)} - x^*\|_A,$$

onde $\text{cond}_2(A) = \lambda_n/\lambda_1$.

Demonstração. Pode ser encontrado em (TREFETHEN; BAU III, 1997, p. 299). \square

Segue imediatamente daí que, quanto mais bem-condicionada é A , menor é o número de iterações do método dos gradientes conjugados necessário para se aproximar bem x^* .

Os dois resultados anteriores deixam claro que a convergência do método dos gradientes conjugados pode ser acelerada melhorando-se a distribuição dos autovalores da matriz A . Conseguimos isso por meio de uma mudança de variáveis adequada ao problema, que fazemos escrevendo $\bar{x} = P_{LR}x$ para alguma matriz $P_{LR} \in \mathbb{R}^{n \times n}$ não-singular. De fato, pois daí podemos colocar o sistema linear $Ax = b$ na forma

$$(P_{LR}^{-T} A P_{LR}^{-1}) \bar{x} = P_{LR}^{-T} b \quad (1.25)$$

e resolvê-lo com o método dos gradientes conjugados. Assim o número de iterações do método passa a depender dos autovalores da matriz $P_{LR}^{-T} A P_{LR}^{-1}$ e não mais daqueles de A . A solução do sistema original é então obtida resolvendo-se $P_{LR}x = \bar{x}$. Na prática, nenhuma mudança de variáveis é realizada explicitamente. O que se faz, na realidade, é aplicar o método dos gradientes conjugados à resolução de (1.25), em termos de \bar{x} , e então reverter

as transformações para re-expressar todas as relações do método em termos de x . Elas podem ser consultadas em (NOCEDAL; WRIGHT, 2006, p. 119).

Sejam $P_L, P_R \in \mathbb{R}^{n \times n}$ matrizes não-singulares. O procedimento de se substituir a resolução de um sistema linear $Ax = b$ por outro

$$(P_L^{-T} A P_R^{-1}) \bar{x} = P_L^{-T} b, \quad P_R x = \bar{x},$$

com a finalidade de agrupar os autovalores de A e de melhorar seu número de condição é denominado **pré-condicionamento**. Dizemos que P_L é um **pré-condicionador pela esquerda** de A e que P_R é um **pré-condicionador pela direita** de A . Quando o sentido da multiplicação está claro, os chamamos simplesmente de **pré-condicionadores**.

Agora já estamos em condições de retornar à resolução do sistema KKT (1.13), para o qual assumimos que a matriz A é simétrica definida positiva no $\ker(B)$. Mais uma vez representamos por $K \in \mathbb{R}^{n \times (n-m)}$ uma matriz de colunas que geram este espaço. Como as colunas de B^T geram o $\text{ran}(B^T)$ e \mathbb{R}^n é uma soma direta destes dois subespaços,

$$\mathbb{R}^n = \ker(B) \oplus \text{ran}(B^T),$$

temos $x^* = K\tilde{x}^* + B^T\hat{x}^*$ para alguns vetores $\tilde{x}^* \in \mathbb{R}^{n-m}$ e $\hat{x}^* \in \mathbb{R}^m$. Determinamos \hat{x}^* recordando que $Bx^* = b$, pois daí $\hat{x}^* = (BB^T)^{-1}b$. Então

$$x^* = K\tilde{x}^* + B^\dagger b,$$

onde $B^\dagger = B^T(BB^T)^{-1}$. Observamos que $B^\dagger b$ é exatamente a solução particular $\dot{x} \in \mathbb{R}^n$ do sistema linear $Bx = b$, indicada na apresentação dos métodos de espaço nulo. Assim sabemos que \tilde{x}^* é solução de

$$(K^T A K) \tilde{x}^* = K^T (a - AB^\dagger b), \quad (1.26)$$

em vista do que foi discutido na Seção 1.3.2. Resolvemos este sistema com o método dos gradientes conjugados, devido à positividade de sua matriz de coeficientes, e em seguida encontramos x^* , substituindo \tilde{x}^* em sua expressão. Uma vez obtido x^* , determinamos y^* por meio do sistema reduzido (1.19). Podemos resolvê-lo, por exemplo, através de uma fatoração QR reduzida da matriz B^T .

Já que estamos aplicando o método dos gradientes conjugados à resolução de (1.26), é interessante procurar por um pré-condicionador $P_{LR} \in \mathbb{R}^{(n-m) \times (n-m)}$ simétrico definido positivo para calcular \tilde{x}^* em um número menor de iterações. Idealmente

$$P_{LR} = (K^T A K)^{\frac{1}{2}},$$

porque daí $P_{LR}^{-T} (K^T A K) P_{LR}^{-1} = I$. Ou melhor: $P_{LR} = K^T A K$, pois as relações do método podem ser ajustadas para se evitar o cálculo de raízes quadradas de matrizes. Mas esta não

Algoritmo 2: Método dos gradientes conjugados projetados.**Entrada:** $\epsilon > 0$;Faça $x \leftarrow B^\dagger b$, $r \leftarrow Ax - a$, $g \leftarrow (I - P)r$ e $d \leftarrow -g$;**repita**

$$\alpha \leftarrow r^T g / d^T A d;$$

$$x \leftarrow x + \alpha d;$$

$$r^+ \leftarrow r + \alpha A d;$$

$$g^+ \leftarrow (I - P)r^+;$$

$$\beta \leftarrow (r^+)^T g^+ / r^T g;$$

$$d \leftarrow -g^+ + \beta d;$$

$$g \leftarrow g^+; \quad r \leftarrow r^+;$$

até $r^T g < \epsilon$;

Resolva o sistema (1.19);

Saída: Uma aproximação da solução (x^*, y^*) de (1.13).

é uma escolha acertada de pré-condicionador, pois com ela invariavelmente teríamos de resolver um sistema linear de matriz de coeficientes P_{LR} (Equação (16.25d) em (NOCEDAL; WRIGHT, 2006, p. 460)) e isso tem um custo computacional equivalente à resolução direta de (1.26). A expressão anterior para P_{LR} sugere, no entanto, pré-condicionadores da forma $K^T \tilde{A} K$, onde $\tilde{A} \in \mathbb{R}^{n \times n}$ é alguma matriz simétrica tal que $K^T \tilde{A} K$ é definida positiva.

A escolha que fazemos para o pré-condicionador do sistema reduzido é

$$P_{LR} = K^T K,$$

pelo seguinte motivo. O método dos gradientes conjugados não depende explicitamente da matriz $K^T A K$, devido às relações (1.20)-(1.24) somente requererem produtos desta matriz por vetores. Mas ele ainda necessita de uma base do $\ker(B)$, por meio da matriz K . É possível, entretanto, modificar o método pré-condicionado com P_{LR} para deixar de exigir K formalmente. A alteração consiste simplesmente em se re-expressar as relações do método diretamente a partir dos iterandos

$$(x^*)^{(k)} = K((\tilde{x}^*)^{(k)}) + B^\dagger b, \quad k \in \mathbb{N}_0,$$

ao invés de $(\tilde{x}^*)^{(k)}$. O método passa então a fazer uso da matriz de projeção ortogonal no $\ker(B)$, que é

$$I - P = K(K^T K)^{-1} K^T = K P_{LR}^{-1} K^T.$$

Outras escolhas para \tilde{A} resultariam em projeções escaladas neste espaço, o que não vemos razões para fazer, ao menos de imediato.

Descrevemos no Algoritmo 2 o método dos gradientes conjugados projetados, construído a partir de todas as escolhas que realizamos. Ele faz uso da matriz K apenas

através do cálculo de g^+ . Podemos escrever $g^+ = r^+ - B^T \hat{g}^+$ e usar a forma alternativa de $I - P$, dada a partir da matriz B ,

$$I - P = I - B^T(BB^T)^{-1}B,$$

para calcular \hat{g}^+ como solução de

$$(BB^T)\hat{g}^+ = Br^+.$$

Assim obtemos g^+ sem depender de K explicitamente. Deixando de utilizar uma base do $\ker(B)$, a exigência de que a diferença $n - m$ tem de ser pequena, própria dos métodos de espaço nulo, também é relaxada. Este procedimento requer alguma atenção, contudo, porque o cálculo indicado pode incorrer em erros significativos de arredondamento ((NO-CEDAL; WRIGHT, 2006, p. 463)). Deste modo algum refinamento do método ainda é recomendado para melhorar sua precisão.

Notamos que as direções $d^{(k)} \in \mathbb{R}^n$ geradas pelo algoritmo estão no $\ker(B)$, por construção. Em razão disso e da escolha do iterando inicial $x^{(0)} \in \mathbb{R}^n$, temos que $Bx^{(k)} = b$ para todo $k \in \mathbb{N}_0$, ou seja, que todos estes iterandos são pontos viáveis do problema de programação quadrática com restrições de igualdade (1.12).

1.4 Métodos de penalidade e a técnica de lagrangiano aumentado

Algumas estratégias importantes para otimização substituem a resolução de um problema com restrições pela resolução de uma sequência de subproblemas irrestritos, nos quais as restrições do problema original são representadas por termos adicionados à função objetivo. O método de penalidade quadrática e a técnica de lagrangiano aumentado são alguns exemplos dessas estratégias, que analisaremos a seguir devido à simplicidade e apelo intuitivo da primeira e a capacidade de prevenção de mau-condicionamento da segunda. Como veremos na próxima seção, a técnica de lagrangiano aumentado particularmente nos interessa, pois podemos conectá-la à resolução de sistemas KKT. Aplicações da técnica de lagrangiano aumentado a problemas reais podem ser vistas em (FORTIN; GLOWINSKI, 1983), (GLOWINSKI; TALLEC, 1989) e (BIRGIN; MARTÍNEZ, 2014), entre vários outros trabalhos.

Por brevidade, mais uma vez focamos nossa atenção apenas em problemas com restrições de igualdade, como o seguinte:

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{sujeita a} \quad c_i(x) = 0, \quad i \in \mathcal{E}, \quad (1.27)$$

onde $m = \text{card}(\mathcal{E})$. Dada uma constante $\gamma > 0$ qualquer, consideramos substituir este problema por um outro, que é irrestrito e da forma

$$\min_{x \in \mathbb{R}^n} \mathcal{Q}(x; \gamma) = f(x) + \frac{\gamma}{2} \sum_{i \in \mathcal{E}} c_i^2(x).$$

Fazemos isso porque a soma em \mathcal{Q} , além de ser diferenciável, serve como uma medida da violação das restrições do problema original. De fato, pois devido aos termos quadráticos, ela é contínua e não-negativa para todo $x \in \mathbb{R}^n$ e somente se anula em pontos viáveis de (1.27). Esta medida é escalada com a constante γ , cuja finalidade é então penalizar a inviabilidade de um ponto $x \in \mathbb{R}^n$, candidato a solução do problema (1.27), com maior ou menor severidade. Por essas razões, chamamos γ de **parâmetro de penalidade** e a soma em questão, de **função de penalidade quadrática**.

Diante dos propósitos de γ e da função de penalidade, parece natural a ideia de tentar se construir uma sequência $(x^{(k)})$ em \mathbb{R}^n , convergente para uma solução x^* , com base nos minimizadores de uma sequência de subproblemas

$$\min_{x \in \mathbb{R}^n} \mathcal{Q}(x; \gamma^{(k)}), \quad k \in \mathbb{N}_0, \quad (1.28)$$

definidos através de constantes positivas $\gamma^{(1)}, \gamma^{(2)}, \dots$ tais que $\gamma^{(k)} \uparrow \infty$ quando $k \rightarrow \infty$. Dizemos isso porque é razoável esperar que os sucessivos incrementos no parâmetro de penalidade de \mathcal{Q} tragam o minimizador $x^{(k)} \in \mathbb{R}^n$ de cada subproblema cada vez mais para perto do conjunto viável de (1.27). E também é razoável esperar que estes mesmos incrementos aumentem a importância do valor de f na minimização de $\mathcal{Q}(x; \gamma^{(k)})$. Assim, ainda é razoável esperar que as minimizações dos subproblemas cada vez mais digam respeito à minimização da própria função objetivo f , em conjuntos progressivamente mais viáveis. Sob hipóteses apropriadas, podemos mostrar que todo ponto limite de uma subsequência convergente de $(x^{(k)})$ também é um candidato à solução de (1.27).

Proposição 1.12. *Sejam $(\gamma^{(k)})$ e $(\tau^{(k)})$ sequências de números reais positivos tais que*

$$\gamma^{(k)} \uparrow \infty \quad \text{e} \quad \tau^{(k)} \rightarrow 0 \quad \text{quando} \quad k \rightarrow \infty.$$

Considere uma sequência $(x^{(k)})$ em \mathbb{R}^n determinada através da resolução aproximada dos subproblemas da relação (1.28), com critério de parada

$$\|\nabla_x \mathcal{Q}(x; \gamma^{(k)})\|_2 \leq \tau^{(k)}.$$

Suponha que $x^ = \lim_{k \in \mathcal{K}} x^{(k)}$ para algum conjunto $\mathcal{K} \subset \mathbb{N}_0$ de cardinalidade infinita. Então:*

- x^* é um ponto estacionário da função de penalidade quadrática, se ele não é viável;
- x^* satisfaz as condições de KKT para o problema (1.27), se ele é viável e os vetores $\nabla c_i(x^*)$, $i \in \mathcal{E}$, são linearmente independentes. Neste caso, ainda é verdade que

$$\lim_{k \in \mathcal{K}} \gamma^{(k)} c_i(x^{(k)}) = y_i^* \quad i \in \mathcal{E}, \quad (1.29)$$

onde y_i^ são as componentes do vetor de multiplicadores de Lagrange associado à x^* .*

Demonstração. Pode ser encontrada em (NOCEDAL; WRIGHT, 2006, p. 503). \square

O esquema iterativo implícito na Proposição 1.12 é o que chamamos método de penalidade quadrática. Trata-se, a princípio, de um método interessante, pois podemos utilizar técnicas de otimização irrestrita para encontrar o minimizador aproximado de cada subproblema dele. De fato, afinal a função de penalidade é suave. Na realidade, porém, há algumas dificuldades com este esquema. A função \mathcal{Q} , por exemplo, pode não ser limitada inferiormente para um determinado $\gamma^{(k)}$, fazendo com que o cálculo aproximado de $x^{(k)}$ muitas vezes divirja. Não somente isso: fica cada vez mais difícil minimizar \mathcal{Q} para valores cada vez maiores do parâmetro de penalidade, porque a hessiana $\nabla_{xx}^2 \mathcal{Q}(x; \gamma^{(k)})$ se torna arbitrariamente mal-condicionada em uma vizinhança suficientemente pequena de $x^{(k)}$. Para termos uma ideia do porquê isso ocorre, observamos que

$$\nabla_{xx}^2 \mathcal{Q}(x; \gamma^{(k)}) = \nabla^2 f(x) + \sum_{i \in \mathcal{E}} \gamma^{(k)} c_i(x) \nabla^2 c_i(x) + \gamma^{(k)} B(x)^T B(x),$$

onde $B(x)$ é como em (1.3). E também notamos que $\gamma^{(k)} c_i(x) \approx y_i^*$, se $k \in \mathbb{N}_0$ é suficientemente grande, x está perto de um minimizador de $\mathcal{Q}(\cdot; \gamma^{(k)})$ e as condições da Proposição 1.12 são satisfeitas, em razão da equação (1.29). Nesta situação, é válida a aproximação

$$\nabla_{xx}^2 \mathcal{Q}(x; \gamma^{(k)}) \approx \nabla_{xx}^2 \mathcal{L}(x, y^*) + \gamma^{(k)} B(x)^T B(x).$$

Um argumento bastante elaborado, baseado nesta expressão, garante que $\nabla_{xx}^2 \mathcal{Q}(x; \gamma^{(k)})$ tem exatamente $m = \text{card}(\mathcal{E})$ autovalores de ordem $\gamma^{(k)}$ ((LUENBERGER, 1984, p. 373-376)). O mau-condicionamento desta matriz fica então aparente, uma vez que $\gamma^{(k)} \uparrow \infty$. Logo, é importante procurar alguma forma de reduzir o mau-condicionamento de $\nabla_{xx}^2 \mathcal{Q}(x; \gamma^{(k)})$. Contudo,

$$c_i(x^{(k)}) \approx \lambda_i^* / \gamma^{(k)}, \quad i \in \mathcal{E},$$

para todo $k \in \mathbb{N}_0$ suficientemente grande, devido ao limite (1.29). Assim $x^{(k)}$ se aproxima de um ponto viável somente se $\gamma^{(k)} \uparrow \infty$, que é justamente o comportamento que precisamos evitar. A técnica de lagrangiano aumentado resolve esse aparente impasse, modificando a função objetivo dos problemas de (1.28) de maneira a permitir que $c(x^{(k)}) \rightarrow 0$, quando $k \rightarrow \infty$, mas sem deixar o valor do parâmetro de penalidade crescer excessivamente. A modificação a que nos referimos também pode ser pensada como uma penalização da função lagrangiana.

Definição 1.6. Dado $\gamma > 0$, a **função lagrangiano aumentado** para (1.27) é a função

$$\mathcal{L}_{\mathcal{Q}} : \mathbb{R}^n \times \mathbb{R}^m \times (0, \infty) \rightarrow \mathbb{R}, \quad \mathcal{L}_{\mathcal{Q}}(x, y; \gamma) = f(x) + \sum_{i \in \mathcal{E}} y_i c_i(x) + \frac{\gamma}{2} \sum_{i \in \mathcal{E}} c_i^2(x).$$

Naturalmente, as quantidades escalares y_i , $i \in \mathcal{E}$, e a constante positiva γ ainda são denominadas **multiplicadores de Lagrange** e **parâmetro de penalidade**, respectivamente.

Então, ao invés de tentarmos construir uma sequência $(x^{(k)})$ em \mathbb{R}^n , convergente para x^* , a partir de (1.28), fazemos isso com a resolução aproximada dos subproblemas

$$\min_{x \in \mathbb{R}^n} \mathcal{L}_{\mathcal{Q}}(x, y^{(k)}; \gamma^{(k)}), \quad k \in \mathbb{N}_0, \quad (1.30)$$

para os quais previamente fixamos estimativas $y^{(0)}, y^{(1)}, \dots \in \mathbb{R}^m$ dos vetores de multiplicadores de Lagrange associados às soluções de cada um deles e, também, valores positivos $\gamma^{(0)}, \gamma^{(1)}, \dots$ para o parâmetro de penalidade dos mesmos. Segue da condição necessária de primeira ordem para $x^{(k)}$ que

$$0 \approx \nabla_x \mathcal{L}_{\mathcal{Q}}(x^{(k)}, y^{(k)}; \gamma^{(k)}) = \nabla f(x^{(k)}) + \sum_{i \in \mathcal{E}} [y_i^{(k)} + \gamma^{(k)} c_i(x^{(k)})] \nabla c_i(x^{(k)}).$$

Por outro lado, nós temos das condições de KKT para x^* que

$$0 = \nabla_x \mathcal{L}(x^*, y^*) = \nabla f(x^*) + \sum_{i \in \mathcal{E}} y_i^* \nabla c_i(x^*),$$

onde $y^* \in \mathbb{R}^m$ é o vetor de multiplicadores de Lagrange associado à x^* . Comparando as equações, podemos deduzir que

$$y_i^* \approx y_i^{(k)} + \gamma^{(k)} c_i(x^{(k)}), \quad i \in \mathcal{E}.$$

Rearranjando esta expressão, encontramos que

$$c_i(x^{(k)}) \approx \frac{1}{\gamma^{(k)}} (y_i^* - y_i^{(k)}), \quad i \in \mathcal{E}.$$

Vemos daí que agora também é possível fazer $c(x^{(k)}) \rightarrow 0$, quando $k \rightarrow \infty$, trazendo $y^{(k)}$ para perto de y^* e sem requerer que o valor de γ aumente indefinidamente. A relação acima sugere, inclusive, uma fórmula para melhorar dinamicamente o valor de $y^{(k)}$, que é

$$y_i^{(k+1)} = y_i^{(k)} + \gamma^{(k)} c_i(x^{(k)}), \quad i \in \mathcal{E}, \quad k \in \mathbb{N}_0. \quad (1.31)$$

Resumimos, portanto, a técnica de lagrangiano aumentado observando que ela consiste em se resolver aproximadamente os subproblemas de (1.30), atualizando com a relação (1.31) as estimativas dos multiplicadores de Lagrange aí utilizados, ocasionalmente aumentando o valor do parâmetro de penalidade em uso, evitando deixá-lo crescer demais.

Podemos justificar o uso da técnica de lagrangiano aumentado na resolução de um problema de otimização com restrições de igualdade com o seguinte resultado.

Teorema 1.13. *Seja x^* um minimizador local de (1.27) com vetor de multiplicadores de Lagrange associado y^* . Suponha que LICQ se verifique em x^* e que as condições suficientes de segunda ordem também estejam satisfeitas neste ponto. Então existe um valor $\bar{\gamma} > 0$ tal que x^* é um minimizador local estrito de $\mathcal{L}_{\mathcal{Q}}(x, y^*; \gamma)$ para todo $\gamma \geq \bar{\gamma}$.*

Demonstração. Pode ser encontrada em (NOCEDAL; WRIGHT, 2006, p. 517). \square

Porém, na realidade, não conhecemos *a priori* o vetor de multiplicadores de Lagrange para fazer uso deste teorema exatamente do modo como ele está enunciado. O que ocorre, na prática, é que a técnica funciona muito bem. Uma justificativa formal e longa para esse fato é dada por Bertsekas (1996, p. 108-111).

1.5 Uma generalização da técnica de lagrangiano aumentado

A técnica de lagrangiano aumentado pode ainda ser generalizada no caso de problemas de programação com restrições lineares de igualdade, $Bx = b$, penalizando na função lagrangiana correspondente a (1.27) não exatamente os quadrados das violações das restrições, mas sim os produtos de cada uma das violações por outra, de maneira ponderada. Para tanto, uma matriz de pesos $W \in \mathbb{R}^{m \times m}$, simétrica semi-definida positiva, é utilizada e a função lagrangiana aumentada para o problema em questão é definida como

$$\mathcal{L}_Q(x, y; W) = f(x) + y^T(Bx - b) + \frac{1}{2}\|Bx - b\|_W^2,$$

onde $\|\cdot\|_W$ é a semi-norma em \mathbb{R}^m induzida por W (ou seja, ela é tal que $\|y\|_W = \sqrt{y^T W y}$ para todo $y \in \mathbb{R}^m$). Sem perda de generalidade, vamos supor que f é a função objetivo quadrática de (1.12). Esta generalização é então usualmente vista na literatura como uma técnica matricial, que consiste da escolha de uma matriz simétrica semi-definida positiva $W \in \mathbb{R}^{m \times m}$ e da substituição do sistema KKT (1.13) por um outro equivalente, que depende de W e tem a forma

$$M_W \begin{bmatrix} x \\ y \end{bmatrix} := \begin{bmatrix} A + B^T W B & B^T \\ B & O \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a + B^T W b \\ b \end{bmatrix}.$$

Evidentemente o sistema acima é o resultado da imposição das condições necessárias de primeira ordem à função lagrangiana aumentada em destaque. No restante deste trabalho optamos pela segunda formulação, por ela ser mais conveniente para os nossos objetivos.

Escolhas apropriadas para a matriz W são aquelas através das quais alguma propriedade algébrica ou numérica é adicionada ao novo sistema. Evidentemente não é fácil identificá-las e por isso as escolhas costumam ser as triviais. Porém, mesmo escolhas simples podem fornecer resultados interessantes. Hestenes (1980, p. 76-77), por exemplo, fez uso dessa técnica, embora originalmente não em linguagem matricial, para obter um sistema linear como o acima, com matriz de coeficientes de bloco $(1, 1)$ definido positivo em todo o \mathbb{R}^n . Para isso ele supôs que a matriz A é definida positiva no $\ker(B)$, com B de ordem $1 \times n$, e tomou

$$W = \gamma I,$$

com $\gamma > 0$ suficientemente grande. Por sua vez, Golub e Greif (2003) observaram que, para esta mesma escolha de W e, principalmente, para uma matriz A definida positiva em todo o \mathbb{R}^n , o valor

$$\gamma = \|A\|_2 / \|B\|_2^2 \tag{1.32}$$

aproximadamente minimizava, entre as escolhas de $\gamma > 0$, os números de condição, na norma 2, de duas — senão todas as três — matrizes

$$M_W, \quad A + B^T W B \quad \text{e} \quad -B(A + B^T W B)^{-1} B^T.$$

Posteriormente, [Golub, Greif e Varah \(2006\)](#) concluíram analiticamente que (1.32) de fato aproximadamente minimiza $\text{cond}_2(A + BWB^T)$. Quando está definida, a terceira matriz acima é simplesmente o complemento de Schur de $A + B^T WB$ em M_W . Esta é assim uma abordagem que pode atenuar o efeito de um possível mau-condicionamento de A em vários métodos numéricos de resolução de problemas de ponto-de-sela. Em particular, naqueles que proporemos logo mais. Para referências adicionais, sugerimos uma consulta aos trabalhos de [Estrin e Greif \(2016\)](#), [Benzi e Olshanskii \(2006\)](#) e [Hu, Shi e Yu \(2004\)](#).

Nos capítulos seguintes, além de empregar a técnica de lagrangiano aumentado, nós ainda faremos operações elementares com as linhas de um sistema KKT, de um modo que envolve a escolha de uma matriz real \tilde{W} , desta vez de ordem $n \times m$, e a reescrita do sistema na forma equivalente

$$M_{\tilde{W}} \begin{bmatrix} x \\ y \end{bmatrix} := \begin{bmatrix} A + \tilde{W}B & B^T \\ B & O \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a + \tilde{W}b \\ b \end{bmatrix}.$$

A primeira técnica é obviamente um caso particular da segunda, com a qual privilegia-se a manutenção da simetria de M e do sinal de seu bloco $(1, 1)$.

2 Resolução de problemas de ponto-de-sela simétricos

Muita atenção tem sido dada ao longo das últimas décadas para o problema de resolver sistemas lineares em forma de ponto-de-sela, uma vez que eles surgem naturalmente em diversas áreas, como a de dinâmica de fluidos computacional, de otimização com restrições e de interpolação de dados dispersos ((GLOWINSKI, 1984), (WRIGHT, 1992) e (WENDLAND, 2005), por exemplo). Por essa razão, atualmente é possível encontrar na literatura muitos métodos para a resolução dos mesmos, cada um com suas próprias características, requerimentos e limitações. Neste e no próximo capítulo nós desenvolvemos métodos que fazem uso extensivo da técnica de lagrangiano aumentado para computar a solução de uma classe importante de tais sistemas, apresentados abaixo.

Sejam $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{m \times n}$, $a \in \mathbb{R}^n$ e $b \in \mathbb{R}^m$, com $m < n$. O **problema de ponto-de-sela** que nós vamos resolver é o problema de determinar a solução de um sistema linear da seguinte forma

$$M \begin{bmatrix} x \\ y \end{bmatrix} := \begin{bmatrix} A & B^T \\ B & O \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix}. \quad (2.1)$$

O símbolo $:=$ nesta equação significa que $M \in \mathbb{R}^{(n+m) \times (n+m)}$ é igual a matriz quadrada à sua direita. Quando $A^T \neq A$, este problema também é chamado **problema de ponto-de-sela generalizado**. Como já vimos anteriormente, ele é conhecido em otimização como **sistema KKT** e tem um papel fundamental lá.

Para poder resolver (2.1), nós consideramos que B tem posto completo, porque essa é uma condição necessária para que ele admita uma única solução (Teorema 3.3 em (BENZI; GOLUB; LIESEN, 2005, p. 17)). Inicialmente nós também vamos assumir que a matriz A é simétrica e definida positiva no $\ker(B)$, já que M é não-singular com seus blocos tendo as propriedades mencionadas (Teorema 3.2 em (BENZI; GOLUB; LIESEN, 2005, p. 16) e a observação que o segue). Mas eventualmente também vamos considerar a situação em que A é uma matriz não simétrica. Para garantir a não-singularidade de M , neste caso, assumiremos que a parte simétrica de A que é definida positiva no $\ker(B)$. Uma adaptação imediata do Teorema 3.2 em (BENZI; GOLUB; LIESEN, 2005, p. 16) assegura que isso é suficiente. Se necessário, ainda é possível enfraquecer mais as exigências sobre a matriz A para ambos os problemas. De fato, pois como Gould (1985, p. 96) demonstra com seu Lema 3.4, a não-singularidade de uma matriz M simétrica é equivalente a não-singularidade de uma matriz $K^T A K$, onde $K \in \mathbb{R}^{n \times (n-m)}$ é qualquer matriz de colunas que geram o $\ker(B)$. E como a teoria que apresentaremos no Capítulo 3 também permite inferir, conclusão semelhante pode ser feita a respeito de problemas generalizados. Deste modo acabaremos

desenvolvendo métodos para a resolução de problemas de ponto de sela da forma (2.1), que funcionam sob as condições mais fracas possíveis. Na verdade, condições até mesmo não conhecidas na literatura. Salientamos, ainda, que possivelmente não existe nenhum método direto especializado na resolução de problemas de ponto-de-sela generalizados além deste nosso. E estamos praticamente certos disso, pelo menos até o ano de 2005 ((BENZI; GOLUB; LIESEN, 2005, p. 40)).

Os cálculos apresentados aqui fornecem a solução exata de (2.1). Apesar de pouco convencional, nos interessamos em determiná-la, principalmente quando $A^T \neq A$, porque acreditamos que o conhecimento dela é essencial para iluminar o desenvolvimento de métodos eficientes de resolução de problemas de ponto-de-sela. Quando A é uma matriz simétrica, os cálculos citados podem ser facilmente traduzidos em um método direto para a resolução de (2.1), o qual então nos auxilia a estabelecer no Capítulo 3 um método direto para a resolução do problema de ponto-de-sela generalizado. Ambos os métodos envolvem a construção de um sistema linear equivalente a (2.1), de sensibilidade melhorada, em um sentido que deixaremos claro mais adiante. Esta não é obviamente nenhuma ideia nova, mas não é por isso que não podemos uma vez mais abordá-la. Ainda mais quando ela trata de problemas tão comuns quanto problemas de ponto-de-sela. De fato, pois Stewart (1989) estabeleceu uma cota superior para um tipo particular de problema de ponto-de-sela, que possui bloco A muito mal-condicionado e vetor b nulo. E Vavasis (1994) investigou as implicações deste resultado, concluindo que muitos métodos de resolução de problemas de ponto-de-sela, comuns até hoje, são instáveis. Por causa disso, ele propôs um método que pudesse encontrar soluções precisas para esses problemas, mesmo quando A fosse uma matriz mal-condicionada. Este é justamente o espírito deste trabalho, porém bem mais geral, por não contar com hipóteses facilitadoras à resolução de problemas de ponto-de-sela, tais como: uma matriz A diagonal ou simétrica definida positiva, um vetor b nulo, etc. Em se tratando especificamente de otimização com restrições, nos referimos a (FORSGREN; GILL; SHINNERL, 1996) para uma investigação abrangente da estabilidade de sistemas simétricos mal-condicionados provenientes de métodos de pontos interiores.

Ainda quando A é uma matriz simétrica, nós somos capazes de mostrar que o método direto proposto para essa situação e os métodos de espaço nulo, dados em sua forma mais geral possível ((BENZI; GOLUB; LIESEN, 2005, p. 32)), estão intrinsecamente relacionados. Com isso notamos que ambos os métodos têm diversas características em comum, razão pela qual passamos a ver o primeiro como uma variante dos demais. Em particular, inferimos que esse novo método pode servir como uma alternativa aos métodos de espaço nulo que não requer o cálculo prévio de uma base do $\ker(B)$. Todas essas conclusões se estendem livremente ao método proposto no Capítulo 3 para resolução de problemas de ponto-de-sela generalizados, porque esse outro método é uma extensão daquele proposto para problemas simétricos.

Nós observamos que construções mais elaboradas são necessárias à resolução do problema (2.1) na ausência de hipóteses fortes para ele. Ainda assim, tais construções têm um custo computacional da ordem de mn^2 operações de ponto flutuante, o que é perfeitamente aceitável do ponto de vista prático, já que $m \ll n$ frequentemente ((GOLUB; GREIF, 2003, p. 1), (GANSTERER; SCHNEID; UEBERHUBER, 2003, p. 1) e (BENZI; GOLUB; LIESEN, 2005, p. 5)). Dessa forma os métodos desenvolvidos aqui são também preferíveis na situação menos atraente aos métodos de espaço nulo tradicionais, onde a diferença $n - m$ é significativamente grande. Parte desse custo pode ainda ser atenuada aproveitando-se estruturas especiais possivelmente presentes nas matrizes A e B , como esparsidade da primeira e estrutura de banda da segunda — a exemplo do que se faz com o próprio método de espaço nulo. Isso exige, entretanto, a especialização de alguns cálculos para a resolução de problemas específicos, o que não discutimos nesta tese, por divergir muito de seus objetivos, já bastante gerais.

Os tópicos deste capítulo são como seguem. Na Seção 2.1 nós utilizamos a técnica de lagrangiano aumentado, juntamente com outras técnicas de manipulação de matrizes, para converter a resolução do sistema linear (2.1) em um problema equivalente, com melhores propriedades numéricas. Obtemos assim um esquema que fornece a solução de (2.1) iterativamente. Na Seção 2.2 listamos alguns resultados técnicos de uso recorrente nas seções seguintes. Na Seção 2.3 explicitamos o limite do processo iterativo encontrado anteriormente. Na Seção 2.4 convertemos esse limite em um método direto para a resolução de (2.1) e fornecemos algoritmos para ele. Mostramos na Seção 2.5 como redescobrir os métodos de espaço nulo a partir do método desenvolvido, discorrendo sobre as propriedades comuns a ambos. Então seguimos no Capítulo 3 desenvolvendo a teoria para a resolução de problemas de ponto-de-sela generalizados.

2.1 Um método iterativo de resolução de problemas de ponto-de-sela simétricos

O primeiro passo que damos na direção de resolver (2.1) é para modificar o espectro do bloco $(1, 1)$ de M , de maneira a torná-lo definido positivo em todo o \mathbb{R}^n . A fim de alcançar este objetivo definimos as matrizes

$$B^\dagger = B^T(BB^T)^{-1}, \quad P = B^\dagger B, \quad \tilde{W} = (P - 2I)AB^\dagger \quad \text{e} \quad \tilde{A} = A + \tilde{W}B,$$

todas de entradas reais, onde a primeira e a terceira delas são de ordem $n \times m$ e as demais, de ordem $n \times n$. Dada uma constante positiva ν qualquer, consideramos ainda as matrizes

$$W = \nu I \quad \text{e} \quad A_\nu = \tilde{A} + B^T W B,$$

ambas também reais, de ordem $m \times m$ e $n \times n$, respectivamente, e o vetor

$$a_\nu = a + (\tilde{W} + B^T W)b$$

de \mathbb{R}^n . Segue então da própria expressão da matriz P que $I - P$ e P são, nesta ordem, as representações matriciais relativas à base canônica de \mathbb{R}^n dos operadores de projeção ortogonal no $\ker(B)$ e na $\text{ran}(B^T)$.

Afirmamos que as funções quadráticas reais definidas a partir de \tilde{A} e A_ν , respectivamente

$$\tilde{q}(x) = x^T \tilde{A}x \quad \text{e} \quad q_\nu(x) = x^T A_\nu x, \quad x \in \mathbb{R}^n,$$

têm as seguintes propriedades:

- \tilde{q} é identicamente nula na $\text{ran}(B^T)$ e é positiva no $\ker(B) \setminus \{0\}$;
- q_ν é positiva em $\mathbb{R}^n \setminus \{0\}$.

Para justificar estes fatos decompomos \mathbb{R}^n em uma soma direta do $\ker(B)$ e de seu complemento ortogonal,

$$\mathbb{R}^n = \ker(B) \oplus \text{ran}(B^T),$$

e relembramos que somente a parte simétrica de uma matriz é relevante para a função quadrática que ela induz ((HORN; JOHNSON, 2013, p. 231)). Assim podemos tomar $x = \tilde{x} + B^T y \in \mathbb{R}^n$, com $\tilde{x} \in \ker(B)$ e $y \in \mathbb{R}^m$, e calcular $\tilde{A} + \tilde{A}^T$ com o auxílio da identidade

$$\tilde{W} = -2(I - P)AB^\dagger - PAB^\dagger \quad (2.2)$$

para ver que

$$\begin{aligned} \tilde{q}(x) &= x^T \left[\frac{1}{2}(\tilde{A} + \tilde{A}^T) \right] x \\ &= x^T \left[\frac{1}{2}(A + A^T) + \frac{1}{2}(\tilde{W}B + B^T \tilde{W}^T) \right] x \\ &= x^T \left[A - PAP - (I - P)AP - PA(I - P) \right] x \\ &= x^T \left[(I - P)A(I - P) \right] x \\ &= \tilde{x}^T A \tilde{x}. \end{aligned}$$

Isto estabelece as propriedades mencionadas para \tilde{q} , já que $\tilde{q}(x) = \tilde{q}(\tilde{x})$ e A é definida positiva no $\ker(B)$ por hipótese. Por sua vez,

$$q_\nu(x) = x^T \tilde{A}x + \nu(Bx)^T(Bx) = \tilde{q}(\tilde{x}) + \nu\|BB^T y\|_2^2$$

e ao menos uma das parcelas desta expressão é positiva, se $x = \tilde{x} + B^T y$ é um vetor não-nulo. De fato, pois neste caso $\tilde{x} \neq 0$ ou $y \neq 0$, já que B tem posto completo. Então $\tilde{q}(\tilde{x}) > 0$, se $\tilde{x} \neq 0$, porque nós acabamos de concluir que \tilde{q} é positiva no $\ker(B) \setminus \{0\}$, ou $\nu\|BB^T y\|_2^2 > 0$, se $y \neq 0$, pois $\nu > 0$ e BB^T é uma matriz não-singular. Como nenhuma das parcelas pode assumir valores negativos, isto implica que a função quadrática q_ν é positiva em $\mathbb{R}^n \setminus \{0\}$, conforme afirmado.

Em face destas observações e do conteúdo apresentado na Seção 1.5, que é aplicado aqui utilizando-se W e \tilde{W} como lá, sabemos que o sistema (2.1) é equivalente a

$$M_\nu \begin{bmatrix} x \\ y \end{bmatrix} := \begin{bmatrix} A_\nu & B^T \\ B & O \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_\nu \\ b \end{bmatrix}, \quad (2.3)$$

cujas matriz de coeficientes tem bloco $(1, 1)$ “definido positivo” em todo o \mathbb{R}^n . Por “definido positivo” nós queremos dizer que $x^T A_\nu x > 0$ para todo $x \in \mathbb{R}^n \setminus \{0\}$, mas sem A_ν necessariamente ser uma matriz simétrica. É possível que A_ν não seja uma matriz simétrica porque sua parte anti-simétrica é dada por

$$A_{\nu,ss} = \frac{1}{2}(A_\nu - A_\nu^T) = \frac{1}{2}\{[(P - 2I)AP] - [(P - 2I)AP]^T\} = PA - AP$$

e não se pode garantir que A e P comutam sob as hipóteses levantadas.

Por razões óbvias, preferimos não perder a simetria do problema de ponto-de-sela original. Logo damos um segundo passo na direção de resolver (2.1) — através de (2.3) — procurando recuperá-la. Para tanto, observamos que a parte simétrica de A_ν ,

$$A_{\nu,s} = \frac{1}{2}(A_\nu + A_\nu^T) = \frac{1}{2}(\tilde{A} + \tilde{A}^T) + \nu B^T B = (I - P)A(I - P) + \nu B^T B,$$

é definida positiva em todo o \mathbb{R}^n no sentido usual, por ser uma matriz simétrica que também induz a função quadrática q_ν . Podemos assim cogitar o esquema iterativo

$$M_{\nu,s} \begin{bmatrix} x^{(k+1)} \\ y^{(k+1)} \end{bmatrix} := \begin{bmatrix} A_{\nu,s} & B^T \\ B & O \end{bmatrix} \begin{bmatrix} x^{(k+1)} \\ y^{(k+1)} \end{bmatrix} = \begin{bmatrix} a_\nu - A_{\nu,ss}x^{(k)} \\ b \end{bmatrix}, \quad k \in \mathbb{N}_0, \quad (2.4)$$

definido a partir das partes de A_ν e de um ponto $(x^{(0)}, y^{(0)}) \in \mathbb{R}^{n+m}$ qualquer, como uma alternativa simétrica para a obtenção da solução de (2.1). De fato, porque a positividade de $A_{\nu,s}$ assegura que cada uma das iterações do esquema tem uma única solução ($M_{\nu,s}$ é não-singular neste caso, pelo Teorema 3.1 em (BENZI; GOLUB; LIESEN, 2005, p. 15-16)) e também porque a sequência de soluções geradas por (2.4) converge para a solução do sistema linear (2.1) se o raio espectral da matriz

$$\begin{bmatrix} A_{\nu,s} & B^T \\ B & O \end{bmatrix}^{-1} \begin{bmatrix} -A_{\nu,ss} & O^T \\ O & O \end{bmatrix}$$

é menor do que a unidade (Teorema 7.19 em (BURDEN; FAIRES, 2005, p. 442-443)). No entanto, este critério e suas variantes, como aquela encontrada em (GOLUB; WATHEN, 1998, p. 532), nada esclarecem a respeito da solução em si. Desta forma, dificilmente podemos aplicá-los na determinação da solução exata de (2.1). Por isso analisaremos na Seção 2.3 a sequência $((x^{(k)}, y^{(k)}))$ termo-a-termo, sem fazer uso algum deles. Isto será possível devido à construção realizada anteriormente, que na verdade terá um papel de pré-condicionamento, acelerando a convergência da sequência indicada para a solução de

(2.1). Podemos então resolver o problema de ponto-de-sela original através do esquema iterativo considerado. Apontamos o trabalho de [Golub e Wathen \(1998\)](#) para maiores detalhes sobre o método de decomposição que o configura.

Antes da análise prometida acima, porém, gostaríamos de dar ainda um terceiro e último passo na direção de resolver (2.1), com o objetivo de melhorar a sensibilidade dos sistemas lineares que compõem o processo iterativo (2.4). Para isso, notamos que tais sistemas foram propositalmente construídos para satisfazer as hipóteses levantadas em ([GOLUB; GREIF, 2003](#)) e ([GOLUB; GREIF; VARAH, 2006](#)), sobre as quais já discorremos na Seção 1.5. Assim podemos mais uma vez fazer uso da técnica de lagrangiano aumentado, desta vez com $W = \gamma(\nu)I$, para substituir aqueles sistemas por estes outros,

$$\begin{bmatrix} A_{\nu,s} + \gamma(\nu)B^T B & B^T \\ B & O \end{bmatrix} \begin{bmatrix} x^{(k+1)} \\ y^{(k+1)} \end{bmatrix} = \begin{bmatrix} a_\nu + \gamma(\nu)B^T b - A_{\nu,ss}x^{(k)} \\ b \end{bmatrix}, \quad k \in \mathbb{N}_0,$$

onde $\gamma(\nu) = \|A_{\nu,s}\|_2 / \|B\|_2^2$. Como comentamos naquela seção, para este valor de $\gamma(\nu)$, os números de condição, na norma 2, de ao menos duas das matrizes

$$(M_{\nu,s})_{\gamma(\nu)I}, \quad A_{\nu,s} + \gamma(\nu)B^T B \quad \text{e} \quad -B(A_{\nu,s} + \gamma(\nu)B^T B)^{-1}B^T$$

são aproximadamente minimizados dentre todas as escolhas possíveis de $\gamma(\nu) > 0$. Nesta situação, melhor estabilidade numérica é esperada na resolução dos últimos sistemas.

O processo iterativo determinado pela última transformação depende de um parâmetro positivo ν , mas não há um valor preferencial para ele. Assim podemos eliminar esta dependência simplesmente fixando um valor positivo qualquer para este parâmetro. Ou melhor ainda: como ν pode ser feito tão pequeno quanto se queira e todas as expressões que dependem dele são também contínuas em ν , podemos eliminar a referida dependência substituindo as expressões pelos limites

$$\begin{aligned} \gamma_* &= \lim_{\nu \rightarrow 0^+} \gamma(\nu) &&= \|(I - P)A(I - P)\|_2 / \|B\|_2^2, \\ A_* &= \lim_{\nu \rightarrow 0^+} (A_{\nu,s} + \gamma(\nu)B^T B) &&= (I - P)A(I - P) + \gamma_* B^T B, \\ A_{**} &= \lim_{\nu \rightarrow 0^+} A_{\nu,ss} &&= PA - AP, \\ a_*^{(k)} &= \lim_{\nu \rightarrow 0^+} (a_\nu + \gamma(\nu)B^T b - A_{\nu,ss}x^{(k)}) &&= a + (\tilde{W} + \gamma_* B^T)b - A_{**}x^{(k)}, \quad k \in \mathbb{N}_0. \end{aligned}$$

Com este procedimento, obtemos uma pequena simplificação das expressões, que deixam de contar com parcelas adicionais. O esquema resultante é

$$M_* \begin{bmatrix} x^{(k+1)} \\ y^{(k+1)} \end{bmatrix} := \begin{bmatrix} A_* & B^T \\ B & O \end{bmatrix} \begin{bmatrix} x^{(k+1)} \\ y^{(k+1)} \end{bmatrix} = \begin{bmatrix} a_*^{(k)} \\ b \end{bmatrix}, \quad k \in \mathbb{N}_0. \quad (2.5)$$

Observamos que γ_* é uma constante positiva, caso contrário $(I - P)A(I - P)$ seria a matriz nula, por ser diagonalizável, ser semi-definida positiva e ter então somente autovalores

nulos. Mas isto não pode acontecer, porque contradiz o fato de que esta mesma matriz é definida positiva no núcleo não-trivial de B . Segue diretamente daí que $A_* = A_{\gamma_*,s}$ ($A_{\nu,s}$ com $\nu = \gamma_*$). Assim permanecemos com um esquema iterativo de sistemas lineares cujas matrizes de coeficientes têm bloco $(1, 1)$ simétrico definido positivo em todo o \mathbb{R}^n . A título de notação, também podemos dizer que $A_{**} = A_{\gamma_*,ss}$ ($A_{\nu,ss}$ com $\nu = \gamma_*$), uma vez que $A_* = A_{\gamma_*,s}$.

Embora constitua um método iterativo de resolução de problemas de ponto-de-sela simétricos, nós não estamos muito interessados no esquema acima em si, porque qualquer passo adicional com ele provavelmente exigiria a avaliação da ação de A_*^{-1} a cada uma de suas iterações, o que teria então um custo computacional alto. Assim, na verdade, vamos utilizá-lo para encontrar a expressão da solução exata de (2.1). Como veremos, (2.5) termina logo em suas primeiras iterações — independentemente de n e m —, razão pela qual conseguimos calcular esta solução explicitamente. Com base nesta solução, um método direto será desenvolvido para a resolução de (2.1). Esse método requerirá apenas uma única resolução de um sistema linear com matriz de coeficientes A_* . Aproximações para esta matriz poderiam ser consideradas, mas não faremos isso aqui, uma vez que o custo para se obter a matriz A_* é apenas da ordem de mn^2 operações de ponto flutuante. De fato, já que o produto $(I - P)A(I - P)$ pode ser calculado primeiro computando-se

$$(I - P)A = A - B^T[(B^\dagger)^T A] \quad (2.6)$$

e, então,

$$[(I - P)A](I - P) = \{[(I - P)A](I - P)\}^T = (I - P)[(I - P)A]^T \quad (2.7)$$

de um modo análogo. Assim, para uma boa performance do método que vamos desenvolver, é importante que $m \ll n$. Como já comentamos, esta não é de modo algum uma restrição séria. Sugerimos consultas a (FRANK; HEALY; MASTRO, 1991), (CARR et al., 2001) e (BOCHEV; LEHOUCQ, 2005) para ver alguns problemas reais em que esta condição ocorre, inclusive na área de otimização. A propósito, enquanto estamos discorrendo sobre custos computacionais, achamos conveniente também apontar que o cálculo de γ_* envolve um esforço computacional mínimo, pois somente estimativas de autovalores extremos são necessárias a ele ((GOLUB; GREIF; VARAH, 2006, p. 791)).

Evidentemente todo o último passo na obtenção de (2.5) poderia ser evitado se tomássemos $\nu = \gamma_*$ desde o início da construção de A_* . Não fazemos isso, entretanto, porque uma motivação para esta escolha ficaria faltando. De fato, afinal contamos com o trabalho desenvolvido por Golub e Greif (2003) para isso e a análise realizada lá está predominantemente baseada em um sistema linear como (2.1), mas de bloco $(1, 1)$ definido positivo. Assim a constante γ_* faz pouco sentido por conta própria, ainda que ela seja positiva, porque seu numerador não foi obtido a partir de uma matriz definida positiva. Nosso procedimento, por outro lado, reforça uma tal escolha ao determiná-la a partir de

outras mais razoáveis. Também permite que expliquemos facilmente porque temos absoluta certeza de que dois dos números de condição, na norma 2, das matrizes

$$M_*, \quad A_* \quad \text{e} \quad -BA_*^{-1}B^T \quad (2.8)$$

foram aproximadamente minimizados com o uso de γ_* . Porque dada uma constante positiva ϵ tal que $0 < \epsilon < \gamma_*$ e $\epsilon \ll 1$, nós sempre podemos enxergar A_* como a matriz

$$[(I - P)A(I - P) + \epsilon B^T B] + (\gamma_* - \epsilon)B^T B = A_{\epsilon,s} + (\gamma_* - \epsilon)B^T B,$$

onde $\gamma_* - \epsilon$ aproxima o valor de $\gamma(\epsilon) = \|A_{\epsilon,s}\|_2 / \|B\|_2^2$. Deste modo as próprias conclusões de Golub e Greif (2003) e Golub, Greif e Varah (2006) suportam nossa afirmação com relação à matriz A_* . Além disso, com a construção indicada, nós podemos teoricamente mostrar que o número de condição do complemento de Schur de A_* em M_* é simplesmente um. A prova desta última afirmação fica para a próxima seção, contudo.

Como um comentário final antes de encerrar esta seção, nós gostaríamos de comparar a teoria desenvolvida até aqui com o trabalho de Estrin e Greif (2015), porque eles têm algumas semelhanças estruturais. Para tanto, recordamos que M_W é a matriz que se obtém de (2.1) aplicando-se a técnica de lagrangiano aumentado a partir de uma matriz $W \in \mathbb{R}^{m \times m}$. Pois bem: no artigo citado os autores mostram que a inversa da matriz de coeficientes M do referido sistema tem bloco $(2, 2)$ nulo quando A é uma matriz simétrica semi-definida positiva e de nulidade m . Com base nesta descoberta, eles demonstram que o negativo do complemento de Schur, em M_W , do bloco líder $A + B^T W B$ de M_W assume uma forma muito especial quando W é uma matriz não-singular: a da própria inversa de W . Ou seja:

$$B(A + B^T W B)^{-1} B^T = W^{-1}.$$

Com isso eles simplificam a expressão explícita da matriz M^{-1} , que pode ser escrita a partir daquela de M_W^{-1} , que por sua vez é bem conhecida na literatura porque tem bloco $(1, 1)$ não-singular nas condições indicadas. A matriz M^{-1} passa então a ser dependente de m^2 parâmetros livres, correspondentes às entradas de W , que podem ser utilizados para se aumentar a estabilidade da ação de M^{-1} (ou de aproximações desta matriz) sobre um dado vetor. A partir dessa nova expressão, Estrin e Greif deduzem pré-condicionadores $\tilde{M}^{-1} \approx M^{-1}$ eficientes para o problema de ponto-de-sela especificamente considerado por eles. Informam, ainda, que várias de suas construções podem ser aplicadas a problemas de ponto-de-sela generalizados, muito embora não forneçam absolutamente nenhuma informação de como se fazer isso. Já nós, no presente trabalho, levantamos hipóteses menos restritivas do que as dos autores — em especial, deixamos de impor sobre o problema de ponto-de-sela a condição de que a nulidade de A tem de ser exatamente m — e aplicamos diversas técnicas matriciais — entre as quais, a técnica de lagrangiano aumentado — para substituir a resolução de (2.1) pela resolução de (2.5). Com este procedimento e

as escolhas (direta) de $W = \nu I$ e (indireta) de $\nu = \gamma_*$, logo mais seremos capazes de mostrar que no nosso caso o negativo do complemento de Schur de A_* em M_* também é a matriz W^{-1} . Além disso, com base no valor fixado para o único parâmetro livre que W possui, conseguiremos ainda melhorar a estabilidade da ação da matriz M_*^{-1} sobre os sistemas lineares que compõem (2.5). Assim determinaremos a solução exata de (2.1), que então carregará essa informação de maior estabilidade em sua expressão. Partindo dessa expressão, deduziremos um método direto para a resolução de (2.1), que consistirá em se construir e em se resolver um sistema linear equivalente a este sistema, mas com matriz de coeficientes bloco-diagonal e de número de condição igual a $\text{cond}_2(A_*)$. Ou seja: ao invés de (2.1), acabaremos resolvendo um sistema linear de matriz de coeficientes facilmente inversível e cuja ação da inversa é numericamente mais estável. Ainda estenderemos essas construções para a resolução de problemas de ponto-de-sela generalizados, fornecendo absolutamente todos os detalhes necessários para isso. Optamos por esta abordagem, ao invés de procurar explicitar algum pré-condicionador para M^{-1} , como o fazem Estrin e Greif (2015), porque a construção de um tal pré-condicionador, no nosso caso, provavelmente seria computacionalmente muito cara e certamente é desnecessária. De fato, já que com uma manipulação simples conseguiremos colocar (2.1) na forma bloco-diagonal mencionada com apenas $\mathcal{O}(mn^2)$ operações de ponto flutuante, isto é, com um custo similar à ação de um pré-condicionador \tilde{M}^{-1} eficiente e prontamente disponível. As abordagens empregadas e as hipóteses levantadas em ambos os trabalhos são suficientemente diferentes para que esteja claro que as ideias dos autores e as nossas foram tidas independentemente. Mas para que não reste dúvidas, observamos que o trabalho de Estrin e Greif (2015) foi publicado em 2 de abril de 2015, enquanto um resumo da teoria que apresentaremos até a Seção 2.3 já constava em um trabalho nosso ((MENDES; DINIZ-EHRHARDT; PEDROSO, 2015)), aceito para publicação desde 10 de março de 2015. Além disso, aparentemente não há na literatura outros paralelos com o restante da teoria desenvolvida neste capítulo e no seguinte. A quase coincidência das datas dos trabalhos citados corrobora com esta afirmação que fazemos.

2.2 Alguns resultados técnicos

A maior parte daqueles resultados que se mostrarão de uso recorrente neste e no capítulo seguinte estão listados aqui. No que segue, $T_C : \mathbb{R}^q \rightarrow \mathbb{R}^p$ denota a transformação linear naturalmente associada a uma matriz real C de ordem $p \times q$.

Proposição 2.1. *Os subespaços de \mathbb{R}^n , $\ker(B)$ e $\text{ran}(B^T)$, são invariantes pelos operadores lineares T_{A_*} e $T_{A_*^{-1}}$.*

Demonstração. Sejam $\tilde{x} \in \ker(B)$ e $\hat{x} \in \text{ran}(B^T)$. Devido à expressão de A_* e à ortogonali-

dade entre os espaços indicados, nós temos que

$$T_{A_*}(\tilde{x}) = (I - P)(A\tilde{x}) \in \ker(B) \quad \text{e} \quad T_{A_*}(\hat{x}) = B^T(\gamma_* B\hat{x}) \in \text{ran}(B^T).$$

Portanto $\ker(B)$ e $\text{ran}(B^T)$ são invariantes por T_{A_*} .

Seja $K \in \mathbb{R}^{n \times (n-m)}$ uma matriz cujas colunas geram o $\ker(B)$. Então existem $\tilde{y}, \hat{y} \in \mathbb{R}^{n-m}$ e $\tilde{z}, \hat{z} \in \mathbb{R}^m$ tais que

$$A_*^{-1}\tilde{x} = K\tilde{y} + B^T\tilde{z} \quad \text{e} \quad A_*^{-1}\hat{x} = K\hat{y} + B^T\hat{z},$$

ou ainda,

$$\tilde{x} = A_*(K\tilde{y}) + A_*(B^T\tilde{z}) \quad \text{e} \quad \hat{x} = A_*(K\hat{y}) + A_*(B^T\hat{z}).$$

Logo $B^T\tilde{z} = 0$ e $K\hat{y} = 0$, porque a interseção do $\ker(B)$ e da $\text{ran}(B^T)$ é trivial e acabamos de ver que estes espaços são invariantes por T_{A_*} . Segue daí que $\tilde{z} = 0$ e que $\hat{y} = 0$, já que ambas as matrizes B^T e K têm posto-coluna completo. Portanto,

$$T_{A_*^{-1}}(\tilde{x}) = K\tilde{y} \in \ker(B) \quad \text{e} \quad T_{A_*^{-1}}(\hat{x}) = B^T\hat{z} \in \text{ran}(B^T),$$

de onde concluímos que $\ker(B)$ e $\text{ran}(B^T)$ são igualmente invariantes por $T_{A_*^{-1}}$. \square

A partir de agora K sempre denotará uma matriz cujas colunas geram o $\ker(B)$.

Corolário 2.2. *Ambas as restrições de T_{A_*} e $T_{A_*^{-1}}$ ao $\ker(B)$ e à $\text{ran}(B^T)$ são operadores auto-adjuntos.*

Demonstração. As transformações lineares T_{A_*} e $T_{A_*^{-1}}$ são operadores auto-adjuntos, porque A_* e A_*^{-1} são matrizes simétricas. E a restrição de qualquer operador auto-adjunto a um subespaço de dimensão finita e invariante de seu domínio também é auto-adjunto. \square

Corolário 2.3. *As restrições de T_{BA_*} e $T_{BA_*^{-1}}$ ao $\ker(B)$ são identicamente nulas, assim como o são as restrições de $T_{(I-P)A_*}$ e $T_{(I-P)A_*^{-1}}$ à $\text{ran}(B^T)$.*

Demonstração. Basta notar que $T_{C_1 C_2} = T_{C_1} T_{C_2}$ para quaisquer matrizes C_1 e C_2 de dimensões apropriadas e aplicar a Proposição 2.1. \square

Muitas identidades matriciais seguem imediatamente do corolário acima, todas as quais serão utilizadas diversas vezes ao longo deste texto:

$$BA_*(I - P) = O, \quad BA_*^{-1}(I - P) = O, \quad (I - P)A_*B^T = O, \quad (I - P)A_*^{-1}B^T = O,$$

e outras similares a elas, com a matriz $I - P$ substituída por K ou K^T (conforme a posição de $I - P$ em relação à A_* e A_*^{-1}), ou a matriz P substituindo B ou B^T . Informamos ao leitor deste texto que é muito importante manter estas identidades em mente, porque não podemos nos referir a elas todas as vezes que forem necessárias, ou do contrário ocorreria uma repetição excessiva de argumentos.

Lema 2.4. *As matrizes A_* , A_*^{-1} , P e $I - P$ comutam entre si.*

Demonstração. Apenas os produtos não-triviais precisam ser justificados. Nós temos que

$$A_*P = (I - P)A[(I - P)P] + \gamma_*B^T[BP] = \gamma_*B^TB$$

e, analogamente, $PA_* = \gamma_*B^TB$. Assim $A_*P = PA_*$. Sabendo disso, nós vemos que

$$A_*(I - P) = A_* - A_*P = A_* - PA_* = (I - P)A_*.$$

Uma vez que $A_*A_*^{-1} = I$, segue que

$$P = PA_*A_*^{-1} = A_*PA_*^{-1}.$$

Portanto $A_*^{-1}P = PA_*^{-1}$. Assim como no caso de A_* e $I - P$, só este fato é suficiente para assegurar que as matrizes A_*^{-1} e $I - P$ também comutam. \square

O próximo resultado responde a várias afirmações deixadas em aberto ao final da seção anterior, todas elas relativas ao complemento de Schur de A_* em M_* .

Proposição 2.5. *O complemento de Schur de A_* em M_* é simplesmente um múltiplo da matriz identidade, a saber: $-BA_*^{-1}B^T = -(1/\gamma_*)I$.*

Demonstração. Para ver que a assertiva é verdadeira, podemos pós-multiplicar a identidade $I = A_*^{-1}A_*$ por B^\dagger e utilizar a expressão de A_* para encontrar que

$$B^\dagger = A_*^{-1}(A_*B^\dagger) = A_*^{-1}\{(I - P)A[(I - P)B^\dagger] + \gamma_*B^T[BB^\dagger]\} = \gamma_*A_*^{-1}B^T. \quad (2.9)$$

Daí, segue imediatamente que $-BA_*^{-1}B^T = -(1/\gamma_*)I$. \square

Esta proposição é bastante interessante porque ela mostra que o cálculo do complemento de Schur de A_* em M_* absolutamente não representa uma dificuldade para o método em construção. Tal fato contrasta com alguns outros métodos para os quais aproximações para o complemento de Schur precisam ser consideradas. Por exemplo: (PEARSON; WATHEN, 2012) e (GOLUB; GREIF; VARAH, 2006).

Lema 2.6. $BA_*^{-1}a_*^{(k)} = (1/\gamma_*)(B^\dagger)^T\{a - A[B^\dagger b + (I - P)x^{(k)}]\} + b$, $k \in \mathbb{N}_0$.

Demonstração. De (2.2) e da expressão de A_{**} nós temos que

$$\begin{aligned} Pa_*^{(k)} &= Pa + (-PAB^\dagger + \gamma_*B^T)b - PA(I - P)x^{(k)} \\ &= P\{a - A[B^\dagger b + (I - P)x^{(k)}]\} + \gamma_*B^Tb. \end{aligned} \quad (2.10)$$

Por outro lado,

$$BA_*^{-1}a_*^{(k)} = BA_*^{-1}[Pa_*^{(k)} + (I - P)a_*^{(k)}] = BA_*^{-1}(Pa_*^{(k)})$$

por uma das identidades que seguem o Corolário 2.3. Portanto,

$$BA_*^{-1}a_*^{(k)} = (BA_*^{-1}P)\{a - A[B^\dagger b + (I - P)x^{(k)}]\} + \gamma_*(BA_*^{-1}B^T)b.$$

Como $P = B^T(B^\dagger)^T$, nós podemos usar a Proposição 2.5 para concluir que

$$BA_*^{-1}a_*^{(k)} = (1/\gamma_*)(B^\dagger)^T\{a - A[B^\dagger b + (I - P)x^{(k)}]\} + b,$$

que é o que gostaríamos de demonstrar. \square

Daqui em diante vamos denotar por $\lambda_{\max}(C)$ e $\lambda_{\min}(C)$, nesta ordem, o maior e menor autovalor, em valores absolutos, de uma matriz C .

Proposição 2.7. *O espectro de A_* consiste de todos os autovalores positivos das matrizes $(I - P)A(I - P)$ e γ_*BB^T , sendo que m deles estão associados a autovetores na $\text{ran}(B^T)$ e os outros $n - m$, a autovetores no $\ker(B)$. Além disso, as matrizes A_* , $(I - P)A(I - P)$ e γ_*BB^T têm todas o mesmo maior autovalor.*

Demonstração. Sejam \tilde{T}_{A_*} e \hat{T}_{A_*} , nesta ordem, as restrições da transformação linear T_{A_*} ao $\ker(B)$ e à $\text{ran}(B^T)$. Devido ao Corolário 2.2 e ao teorema espectral, sabemos que existem bases ortonormais do $\ker(B)$ e da $\text{ran}(B^T)$ que consistem, respectivamente, de autovetores de \tilde{T}_{A_*} e \hat{T}_{A_*} , todos os quais estão associados a autovalores reais. A união destas bases fornece uma base ortonormal de \mathbb{R}^n , porque este espaço pode ser decomposto em uma soma direta do $\ker(B)$ e da $\text{ran}(B^T)$. É então possível analisar todo o espectro de T_{A_*} a partir dos espectros individuais de \tilde{T}_{A_*} e \hat{T}_{A_*} , já que, para qualquer vetor $v \in \mathbb{R}^n$ nesta base, nós temos que

$$T_{A_*}(v) = \tilde{T}_{A_*}(v) = \lambda v \quad \text{ou} \quad T_{A_*}(v) = \hat{T}_{A_*}(v) = \lambda v,$$

onde $\lambda \in \mathbb{R}$ é o autovalor associado a v por uma das restrições em destaque. Uma vez que as transformações \tilde{T}_{A_*} e \hat{T}_{A_*} coincidem com as restrições de $T_{(I-P)A(I-P)}$ e $T_{\gamma_*B^TB}$ ao $\ker(B)$ e à $\text{ran}(B^T)$, nesta ordem, nós podemos concluir que todos os autovalores positivos das matrizes $(I - P)A(I - P)$ e γ_*B^TB são também autovalores de A_* e que eles são todos os autovalores que esta matriz possui. Realmente, pois ao todo temos n autovalores positivos, $n - m$ dos quais são devidos à positividade da matriz A no $\ker(B)$, que é um subespaço de \mathbb{R}^n de dimensão $n - m$, e os m demais autovalores, em razão de B^TB ser uma matriz semi-definida positiva e ter posto m . A primeira assertiva fica então verificada, porque as matrizes γ_*BB^T e γ_*B^TB compartilham autovalores positivos, devido às suas decomposições em valores singulares. Por outro lado, $\|C\|_2 = \sqrt{\lambda_{\max}(CC^T)}$ para qualquer matriz C . Logo,

$$\lambda_{\max}(\gamma_*BB^T) = \gamma_*\|B\|_2^2 = \|(I - P)A(I - P)\|_2 = \lambda_{\max}((I - P)A(I - P)),$$

onde a segunda igualdade é ainda em razão da definição de γ_* . Assim,

$$\lambda_{\max}(A_*) = \max\{\lambda_{\max}((I - P)A(I - P)), \lambda_{\max}(\gamma_* BB^T)\} = \lambda_{\max}((I - P)A(I - P))$$

e, conseqüentemente, as matrizes

$$A_*, \quad (I - P)A(I - P) \quad \text{e} \quad \gamma_* BB^T$$

têm todas o mesmo maior autovalor. □

2.3 A solução exata de problemas de ponto-de-sela simétricos

Finalmente estamos em condições de determinar a solução exata do sistema linear (2.1). Como já mencionamos antes, faremos isso por meio da resolução do processo iterativo (2.5). Pois bem, usando eliminação gaussiana, a equação que define este esquema pode ser reformulada como no método do complemento de Schur ((BENZI; GOLUB; LIESEN, 2005, p. 30)):

$$\begin{cases} A_* x^{(k+1)} = a_*^{(k)} - B^T y^{(k+1)} \\ (BA_*^{-1}B^T)y^{(k+1)} = BA_*^{-1}a_*^{(k)} - b \end{cases}.$$

Nós vemos da segunda equação, da Proposição 2.5 e do Lema 2.6 que

$$y^{(k+1)} = (B^\dagger)^T \{a - A[B^\dagger b + (I - P)x^{(k)}]\}.$$

Podemos então substituir $y^{(k+1)}$ na primeira equação para encontrar que

$$A_* x^{(k+1)} = a_*^{(k)} - P\{a - A[B^\dagger b + (I - P)x^{(k)}]\}$$

e ainda pré-multiplicar esta última expressão por BA_*^{-1} para descobrir que

$$Bx^{(k+1)} = BA_*^{-1}a_*^{(k)} - (BA_*^{-1}B^T)(B^\dagger)^T \{a - A[B^\dagger b + (I - P)x^{(k)}]\}.$$

Então, segue claramente da Proposição 2.5 e do Lema 2.6, que $Bx^{(k+1)} = b$. Assim,

$$Px^{(k+1)} = B^\dagger b, \quad k \in \mathbb{N}_0.$$

Isto e a definição de $a_*^{(k)}$ implicam que

$$(I - P)a_*^{(k)} = (I - P)a - 2(I - P)AB^\dagger b + (I - P)A(Px^{(k)}) = (I - P)(a - AB^\dagger b)$$

para todo $k \in \mathbb{N}$ (a partir daqui $k \neq 0$). Por outro lado, nós vemos de (2.10) que

$$P\{a - A[B^\dagger b + (I - P)x^{(k)}]\} = Pa_*^{(k)} - \gamma_* B^T b.$$

A substituição desta expressão na equação encontrada para $A_* x^{(k+1)}$ fornece

$$A_* x^{(k+1)} = (I - P)a_*^{(k)} + \gamma_* B^T b.$$

Segue então de ambas as observações que

$$A_* x^{(k+1)} = (I - P)(a - AB^\dagger b) + \gamma_* B^T b, \quad k \in \mathbb{N}.$$

Deste modo a sequência $(x^{(k)})$ é constante a partir do seu terceiro termo e convergente à

$$x^* = (I - P)A_*^{-1}(a - AB^\dagger b) + B^\dagger b.$$

Notamos que a última igualdade é devida ao Lema 2.4 e à equação (2.9). Substituímos o ponto limite assim encontrado na expressão de $y^{(k+1)}$ para concluir que a sequência $(y^{(k)})$ é constante a partir do seu quarto termo e convergente à

$$\begin{aligned} y^* &= (B^\dagger)^T \{a - A[B^\dagger b + (I - P)A_*^{-1}(a - AB^\dagger b)]\} \\ &= (B^\dagger)^T [I - AA_*^{-1}(I - P)](a - AB^\dagger b). \end{aligned}$$

A última igualdade é novamente devida ao Lema 2.4. Se agora observarmos que

$$(I - P)A[(I - P)A_*^{-1}] = (I - P)A[(I - P)A_*^{-1}(I - P)]$$

pelo Corolário 2.3, teremos que

$$[(I - P)A(I - P)]A_*^{-1} = (A_* - \gamma_* B^T B)A_*^{-1}(I - P) = I - P. \quad (2.11)$$

Deste modo,

$$\begin{aligned} Ax^* + B^T y^* &= A(I - P)A_*^{-1}(a - AB^\dagger b) + AB^\dagger b + P[I - A(I - P)A_*^{-1}](a - AB^\dagger b) \\ &= [(I - P)A(I - P)A_*^{-1}](a - AB^\dagger b) + P(a - AB^\dagger b) + AB^\dagger b \\ &= (I - P)(a - AB^\dagger b) + P(a - AB^\dagger b) + AB^\dagger b \\ &= a. \end{aligned}$$

Além disso, $Bx^* = b$ trivialmente.

Teorema 2.8. *Se B tem posto completo e A é uma matriz simétrica definida positiva no $\ker(B)$, então o sistema (2.1) admite uma única solução, cuja expressão é*

$$\begin{bmatrix} x^* \\ y^* \end{bmatrix} = \begin{bmatrix} (I - P)A_*^{-1}(a - AB^\dagger b) + B^\dagger b \\ (B^\dagger)^T [I - AA_*^{-1}(I - P)](a - AB^\dagger b) \end{bmatrix}.$$

O número de condição, na norma 2, da matriz A_ nesta expressão é ainda aproximadamente o menor dentre aqueles das matrizes de forma*

$$(I - P)A(I - P) + \gamma B^T B, \quad \gamma > 0. \quad (2.12)$$

Se $\lambda_{\min}(A_) > \lambda_{\min}(A)$, então também $\text{cond}_2(A_*) < \text{cond}_2(A)$.*

Demonstração. Segue do Teorema 3.2 em (BENZI; GOLUB; LIESEN, 2005, p. 16) que o sistema linear (2.1) é possível e determinado. A expressão da solução do sistema é como indicada, devido às contas desta seção. Que a matriz A_* tem número de condição, na norma 2, aproximadamente mínimo entre as matrizes (2.12) é a conclusão a que se chega no parágrafo que contém (2.8). Assim só temos de verificar a última assertiva. Pois bem,

$$\text{cond}_2(A_*) = \frac{\|(I - P)A(I - P)\|_2}{\lambda_{\min}(A_*)} \leq \frac{\|A\|_2}{\lambda_{\min}(A_*)} < \frac{\|A\|_2}{\lambda_{\min}(A)} = \text{cond}_2(A),$$

em razão da Proposição 2.7, de $I - P$ ser uma matriz de projeção ortogonal e de estarmos supondo que $\lambda_{\min}(A_*) > \lambda_{\min}(A)$. Portanto, $\text{cond}_2(A_*) < \text{cond}_2(A)$, como gostaríamos de demonstrar. \square

Suponha que o mau-condicionamento da matriz de coeficientes do sistema linear (2.1) seja, em sua maior parte, devido à matriz A (até mesmo porque, se não fosse assim, não seria lógico tentar melhorar o condicionamento do bloco $(1, 1)$ de M somando a ele um múltiplo da matriz $B^T B$). Então, quando $A = (A_{ij})$ é uma matriz muito mal-condicionada, usualmente é devido ao seu menor autovalor, em valor absoluto, ser muito pequeno. De fato, já que o valor de $\lambda_{\max}(A)$ frequentemente está controlado, uma vez que pelo teorema de Gershgorin ((KREYSZIG, 1978, p. 313)),

$$|\lambda_{\max}(A) - A_{ii}| \leq \sum_{j \neq i} |A_{ij}|$$

para algum $i \in \{1, \dots, n\}$, ou ainda,

$$\lambda_{\max}(A) \leq n \cdot \max_{i,j} \{|A_{ij}|\},$$

e $\max_{i,j} \{|A_{ij}|\}$ não é um número muito grande. Após tantas transformações com a matriz A , é bastante improvável que o menor autovalor de A_* , em valor absoluto, não seja superior a uma quantidade tão diminuta quanto $\lambda_{\min}(A)$ neste caso. Ainda mais se $\text{cond}_2(A_*)$ satisfaz uma propriedade de minimização. Assim, a condição $\lambda_{\min}(A_*) > \lambda_{\min}(A)$ no teorema anterior está mais para uma condição técnica do que uma condição cuja ocorrência, na prática, realmente precisa ser assegurada.

Corolário 2.9. *Nas condições do Teorema 2.8, a inversa da matriz M é*

$$\begin{bmatrix} F & (I - FA)B^\dagger \\ (B^\dagger)^T(I - AF) & -(B^\dagger)^T A(I - FA)B^\dagger \end{bmatrix},$$

onde $F = (I - P)A_*^{-1}$ é uma matriz simétrica.

Demonstração. Para encontrar a expressão de M^{-1} , nós substituímos as letras a e b das equações do Teorema 2.8 pelas matrizes $I \in \mathbb{R}^{n \times n}$ e $O \in \mathbb{R}^{m \times n}$ ou $O^T \in \mathbb{R}^{n \times m}$ e $I \in \mathbb{R}^{m \times m}$, nesta ordem. Para ver que F é uma matriz simétrica, é suficiente recordarmos que A_*^{-1} e $I - P$ comutam e que ambas são matrizes simétricas. \square

A expressão exibida acima para a inversa da matriz M é similar àquela dada por [Gansterer, Schneid e Ueberhuber \(2003, p. 11\)](#), diferenciando-se dela majoritariamente pelo bloco $(1, 1)$ de M^{-1} . Para os autores, ele é

$$K(K^T AK)^{-1}K^T,$$

onde $K \in \mathbb{R}^{n \times (n-m)}$ é uma matriz de colunas ortonormais que geram o $\ker(B)$. Claramente, as duas formas do bloco $(1, 1)$ são equivalentes, já que a inversa de uma matriz não-singular é única e isto implica que os blocos de M^{-1} também o são. Nossa expressão, contudo, não depende de uma base do $\ker(B)$ para ficar bem determinada. Além disso, o cálculo de uma aproximação para a inversa de M a partir da expressão citada, se necessária, ainda pode ser mais estável ([\(ESTRIN; GREIF, 2015, p. 373\)](#)).

Não surpreendentemente, segue do Teorema [2.8](#) que a expressão da solução exata de [\(2.1\)](#) corresponde à multiplicação da inversa da matriz M pelo vetor do lado direito do sistema. Salientamos, no entanto, que o cálculo da solução de um problema de ponto-de-sela simétrico com a referida expressão não é a mesma coisa que inverter M explicitamente. De fato, pois a solução assim determinada apenas envolve produtos de matriz por vetor, onde somente as matrizes A_* e $(B^\dagger)^T$ não estão prontamente disponíveis (mas podem ser obtidas com $\mathcal{O}(mn^2)$ operações de ponto flutuante, independentemente da maneira que se obtém o fator de Cholesky de BB^T) e somente o vetor

$$A_*^{-1}[(I - P)(a - AB^\dagger b)] \tag{2.13}$$

é proveniente da resolução de um sistema linear auxiliar (de ordem menor do que o sistema linear original). Na área de otimização, inclusive, [Nocedal e Wright \(2006, p. 456\)](#) apontam que um tal procedimento é uma maneira de se recobrar o método do complemento de Schur, quando o bloco $(1, 1)$ de um sistema KKT é não-singular. No nosso caso, porém, acabaremos recuperando o método de espaço nulo. Os detalhes disso serão exibidos na Seção [2.5](#).

A vantagem de se utilizar a expressão encontrada para a solução de [\(2.1\)](#) é a certeza de que ela carrega consigo a informação de maior estabilidade presente nos blocos da matriz M^{-1} e não depende do conhecimento de uma base do $\ker(B)$. A desvantagem é que provavelmente o sistema linear auxiliar implícito em [\(2.13\)](#) é necessário para avaliá-la é denso. Temos, contudo, de entender que isso é aceitável no contexto deste trabalho, porque há um intercâmbio entre boas propriedades de condicionamento e esparsidade ([\(BENZI; GOLUB; LIESEN, 2005, p. 39\)](#)) e estamos buscando exatamente essas propriedades. Sabemos que isso limita a aplicabilidade dos métodos a problemas de menores dimensões, se nenhuma estrutura especial das matrizes A e B for aproveitada adicionalmente. Por outro lado, também sabemos que métodos diretos não são apropriados à resolução de problemas imensamente grandes, por causa da maior necessidade de recursos computacionais e da maior propagação de erros de arredondamento nesses casos. Ao invés disso, métodos diretos

são mais empregados naquelas situações em que confiabilidade e previsibilidade de recursos computacionais são requeridos ((BENZI, 2002, p. 418-419)). Portanto, não poder aplicar nossos métodos a problemas muito grandes não se trata realmente de uma perda. Com isso em mente, observamos que experimentos numéricos realizados sugerem que os métodos propostos aqui são competitivos para problemas de ordem até mais ou menos 15000, o que é suficientemente razoável para abranger um bom número de problemas reais.

2.4 Um método direto de resolução de problemas de ponto-de-sela simétricos

De acordo com o Teorema 2.8 e também a equação (2.9), nós podemos naturalmente pensar em um método direto para a resolução de (2.1) que consiste na construção e resolução do seguinte sistema linear equivalente a ele:

$$\begin{bmatrix} A_* & O^T \\ O & \gamma_* BB^T \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} (I - P)(a - AB^\dagger b) + \gamma_* B^T b \\ \gamma_* B[I - AA_*^{-1}(I - P)](a - AB^\dagger b) \end{bmatrix}. \quad (2.14)$$

Claramente, é a construção de A_* que predomina no custo computacional de obtenção da matriz de coeficientes do sistema. Já o cálculo do vetor do seu lado direito depende da obtenção do vetor (2.13), que é simplesmente um subproduto da resolução das primeiras n equações de (2.14), novamente devido a (2.9). Portanto ele não implica em nenhum custo adicional. Desta forma, este sistema, que nunca é montado explicitamente, mas sim resolvido como dois subsistemas lineares desacoplados, tem um custo geral de construção de $\mathcal{O}(mn^2)$ operações de ponto flutuante. Se denso, resolvemos o primeiro subsistema por métodos iterativos como, por exemplo, métodos do tipo gradientes conjugados generalizados. Caso contrário, é também possível empregar a fatoração de Cholesky de A_* .

O sistema acima é interessante porque muito certamente $\text{cond}_2(A_*) < \text{cond}_2(A)$, porque $\text{cond}_2(A_*)$ foi aproximadamente minimizado dentre as escolhas de matrizes da forma (2.12) e também porque o número de condição, na norma 2, da matriz de coeficientes do sistema é justamente $\text{cond}_2(A_*)$. De fato, uma vez que a matriz de coeficientes é bloco diagonal — significando que seu espectro é determinado pelos espectros isolados dos blocos em sua diagonal não-nula — e todos os autovalores de $\gamma_* BB^T$ são também autovalores de A_* pela Proposição 2.7. Em adição a esta observação, o sistema (2.14) também é interessante porque o valor de $\text{cond}_2(A_*)$ é conhecido em termos das matrizes A e B e de uma base do $\ker(B)$.

Proposição 2.10. *Seja $K \in \mathbb{R}^{n \times (n-m)}$ uma matriz de colunas ortonormais que geram o $\ker(B)$. Então $\text{cond}_2(A_*) = \max\{\text{cond}_2(K^T A K), \text{cond}_2(BB^T)\}$.*

Demonstração. Da demonstração da Proposição 2.7, nós sabemos que \mathbb{R}^n possui uma base de autovetores de A_* , onde $n - m$ deles são também autovetores de $(I - P)A(I - P)$,

que pertencem ao $\ker(B)$. Afirmamos que, se (λ, \tilde{x}) é um autopar de A_* , com \tilde{x} desta forma, então $(\lambda, K^T \tilde{x})$ é um autopar de $K^T AK$. Realmente, pois $I - P = KK^T$, devido à ortogonalidade das colunas de K , e daí

$$(K^T AK)(K^T \tilde{x}) = K^T \{[(I - P)A(I - P)]\tilde{x}\} = K^T(\lambda \tilde{x}) = \lambda(K^T \tilde{x}).$$

Assim, todo autovalor positivo de $(I - P)A(I - P)$ é também um autovalor de $K^T AK$ e estes são todos os autovalores que esta matriz possui. Segue daí e da Proposição 2.7 que

$$\lambda_{\min}(A_*) = \min\{\lambda_{\min}(K^T AK), \lambda_{\min}(\gamma_* BB^T)\}.$$

Segue novamente da Proposição 2.7 e da ortonormalidade das colunas da matriz K , que

$$\lambda_{\max}(\gamma_* BB^T) = \lambda_{\max}(A_*) = \|(I - P)A(I - P)\|_2 = \|K^T AK\|_2 = \lambda_{\max}(K^T AK).$$

Logo, $\text{cond}_2(A_*) = \max\{\text{cond}_2(K^T AK), \text{cond}_2(BB^T)\}$, como queríamos demonstrar. \square

Esta proposição mostra uma primeira relação entre o método em construção e os métodos de espaço nulo, uma vez que os últimos dependem da resolução de sistemas lineares com matrizes de coeficientes $K^T AK$ e BB^T . Não se tratam de quaisquer métodos de espaço nulo, mas sim daqueles cujo sistema reduzido de ordem $n - m$ é determinado a partir de uma base ortonormal do $\ker(B)$. Esta também é uma observação interessante, porque tais sistemas são bem-condicionados quando a matriz A o é ((BENZİ; GOLUB; LIESEN, 2005, p. 39)).

De (2.14), nós vemos que $(I - P)(a - AB^\dagger b) = A_* x - \gamma_* B^T b$ e que

$$\begin{aligned} \gamma_* BB^T y &= \gamma_* B(a - AB^\dagger b) - \gamma_* BA[A_*^{-1}(I - P)(a - AB^\dagger b)] \\ &= \gamma_* B(a - AB^\dagger b) - \gamma_* BA[x - (\gamma_* A_*^{-1} B^T)b]. \end{aligned}$$

Recordando (2.9), temos que

$$\gamma_* BAx + \gamma_* BB^T y = \gamma_* Ba.$$

Assim o sistema linear (2.14) é equivalente a

$$\begin{bmatrix} A_* & O^T \\ \gamma_* BA & \gamma_* BB^T \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} (I - P)(a - AB^\dagger b) + \gamma_* B^T b \\ \gamma_* Ba \end{bmatrix}. \quad (2.15)$$

A aparência deste sistema reforça ainda mais a relação entre o método sendo desenvolvido e os métodos de espaço nulo, por fornecer as últimas m componentes da solução de (2.1) da mesma maneira que os métodos de espaço nulo o fazem.

O Algoritmo 3 exibido a seguir fornece a solução de (2.1) por meio da resolução de (2.14). Embora na maior parte das vezes não seja mandatório, a utilização de um valor positivo para o parâmetro δ em seu Passo 3 é recomendada, porque isso torna A_* uma

Algoritmo 3: Calcula a solução de (2.1) através da resolução de (2.14).

Dados de entrada:

m, n inteiros positivos tais que $m \ll n$,
 $B \in \mathbb{R}^{m \times n}$ de posto completo, $A \in \mathbb{R}^{n \times n}$ simétrica e definida positiva no $\ker(B)$,
 $a \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $\delta \geq 0$.

Variáveis de trabalho:

$\gamma_*, \sigma_* \in \mathbb{R}$; $x^*, w_n, v_n \in \mathbb{R}^n$; $y^* \in \mathbb{R}^m$; $A_*, W_{nn} \in \mathbb{R}^{n \times n}$; $R \in \mathbb{R}^{m \times m}$; $W_{mn}, V_{mn} \in \mathbb{R}^{m \times n}$;
 $W_{nm}, V_{nm} \in \mathbb{R}^{n \times m}$.

Passo 1: Faça $W_{nm} = B^T$ e calcule o fator de Cholesky R do produto BB^T por meio da fatoração QR reduzida de W_{nm} . Usando R , calcule a solução do sistema matricial $(BB^T)W_{mn} = B$ e então estime o valor de $\sigma_* = \|B\|_2^2 = \|R^T R\|_2$.

Passo 2: Calcule $(I - P)A(I - P)$ conforme indicado nas equações (2.6) e (2.7). Para tanto, calcule $V_{mn} = W_{mn}A$ e $W_{nn} = W_{nm}V_{mn}$. Então substitua W_{nn} por $A - W_{nn}$. Daí calcule $V_{nm} = W_{nn}W_{nm}$ e $A_* = V_{nm}W_{mn}$. Por fim, substitua A_* por $W_{nn} - A_*$.

Passo 3: Estime o maior autovalor γ_* de A_* e daí o substitua pela razão γ_*/σ_* . Também substitua a matriz V_{nm} por γ_*W_{nm} e, em seguida, W_{nn} por $V_{nm}B$. Faça $A_* \leftarrow A_* + W_{nn}$. Se $\delta > 0$, atualize a diagonal principal de A_* adicionando $\delta\gamma_*$ a elas.

Passo 4: Substitua V_{nm} por W_{mn}^T e então calcule $w_n = V_{nm}b$ e $v_n = Aw_n$.

Passo 5: Substitua v_n por $a - v_n$ e calcule $y^* = W_{mn}v_n$. Substitua a inicialmente por $W_{nm}y^*$ e então por $v_n - a$. Daí substitua a pela solução do sistema linear $A_*x = a$, resolvido iterativamente ou através do fator de Cholesky de A_* . Faça $x^* = a + w_n$.

Passo 6: Substitua a por Aa e v_n por $v_n - a$. Faça $y^* = W_{mn}v_n$.

Dados de retorno:

A solução de (2.1): (x^*, y^*) .

matriz mais diagonalmente dominante do que predita pela teoria, o que por sua vez auxilia a resolução numérica do sistema linear de matriz de coeficientes A_* .

Como deve estar claro da Proposição 2.10, uma matriz B muito próxima de uma matriz de posto numericamente deficiente pode prejudicar significativamente o desempenho do método proposto, de modo que seu uso não é recomendado neste caso. Mas se o mau-condicionamento de B ainda é moderado, podemos procurar atenuar um pouco do seu efeito, fazendo uma mudança tradicional de variáveis, que não tem um custo proibitivo na situação usual em que $m \ll n$. Sejam $Q \in \mathbb{R}^{n \times m}$ e $R \in \mathbb{R}^{m \times m}$ os fatores da fatoração QR reduzida de B^T , onde Q tem colunas ortonormais e R é triangular superior.

Se $b_Q = R^{-T}b$, então o sistema linear (2.1) pode ser colocado na seguinte forma

$$\begin{bmatrix} A & Q \\ Q^T & O \end{bmatrix} \begin{bmatrix} x \\ y_Q \end{bmatrix} = \begin{bmatrix} a \\ b_Q \end{bmatrix}, \quad Ry = y_Q \in \mathbb{R}^m, \quad (2.16)$$

notando-se, para tanto, simplesmente que

$$\begin{bmatrix} A & Q \\ Q^T & O \end{bmatrix} = \begin{bmatrix} I & O^T \\ O & R^{-T} \end{bmatrix} \begin{bmatrix} A & B^T \\ B & O \end{bmatrix} \begin{bmatrix} I & O^T \\ O & R^{-1} \end{bmatrix}.$$

Assim um sistema análogo a (2.14) pode ser trivialmente construído a partir de (2.16),

$$\begin{bmatrix} A_* & O^T \\ O & \gamma_* I \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} (I - P)(a - AQR^{-T}b) + \gamma_* QR^{-T}b \\ \gamma_* R^{-1}Q^T[I - AA_*^{-1}(I - P)](a - AQR^{-T}b) \end{bmatrix}, \quad (2.17)$$

para o qual ainda temos que

$$P = QQ^T, \quad \gamma_* = \|(I - P)A(I - P)\|_2 \quad \text{e} \quad A_* = (I - P)A(I - P) + \gamma_* P. \quad (2.18)$$

Vamos supor que a matriz R^{-1} está disponível. Então o número de condição da matriz de coeficientes do sistema acima não depende diretamente de B . De fato, pois todos os autovalores de $\gamma_* I$ são também autovalores de A_* , já que $A_*Q = \gamma_*Q$, e daí a Proposição 2.10 assegura que

$$\text{cond}_2(A_*) = \text{cond}_2(K^T AK),$$

para toda matriz $K \in \mathbb{R}^{n \times (n-m)}$ de colunas ortonormais que geram o $\ker(B)$. Na prática, porém, é difícil que R^{-1} esteja disponível. Mas ainda assim a transformação indicada melhora a precisão do cálculo da solução de (2.1), por causa das muitas expressões que passam a depender exclusivamente da matriz Q de colunas ortonormais.

De (2.18) nós imediatamente notamos uma série de simplificações nas expressões de elementos do método desenvolvido, que compensam o custo adicional necessário para se computar e armazenar o fator Q de B^T . O Algoritmo 4, a seguir, considera estas mudanças. Em seu primeiro passo, nós explicitamente indicamos a ortogonalização das colunas da matriz B^T pelo método de Householder. Isto porque o fator ortogonal \check{Q} desta fatoração, computado em aritmética de precisão finita, é, para fins práticos, uma matriz de colunas ortonormais. De fato, já que

$$\check{Q}^T \check{Q} = I + E_H \quad \text{e} \quad \|E_H\|_2 \approx u,$$

onde u é a unidade de arredondamento da máquina ((GOLUB; VAN LOAN, 2013, p. 255)). Se desejamos resolver o problema (2.1) com o Algoritmo 4 e o número de condição da matriz B não é uma preocupação, então um método de fatoração QR mais indicado é o de Gram-Schmidt modificado. Para ele nós temos que

$$\check{Q}^T \check{Q} = I + E_{MGS} \quad \text{e} \quad \|E_{MGS}\|_2 \approx u \cdot \text{cond}_2(B),$$

mas por outro lado \check{Q} pode ser obtido com aproximadamente metade das operações de ponto flutuante necessárias à ortogonalização de Householder ((GOLUB; VAN LOAN, 2013, p. 255)).

Algoritmo 4: Calcula a solução de (2.1) através da resolução de (2.17), com últimas m linhas multiplicadas por R .

Dados de entrada:

m, n inteiros positivos tais que $m \ll n$,
 $B \in \mathbb{R}^{m \times n}$ de posto completo, $A \in \mathbb{R}^{n \times n}$ simétrica e definida positiva no $\ker(B)$,
 $a \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $\delta \geq 0$.

Variáveis de trabalho:

$\gamma_* \in \mathbb{R}$; $x^*, w_n, v_n \in \mathbb{R}^n$; $y^* \in \mathbb{R}^m$; $A_*, W_{nn} \in \mathbb{R}^{n \times n}$; $Q, W_{nm} \in \mathbb{R}^{n \times m}$; $R \in \mathbb{R}^{m \times m}$.

Passo 1: Faça $W_{nm} = B^T$ e calcule os fatores Q e R da fatoração QR reduzida de W_{nm} pelo método de Householder. Substitua B por Q^T .

Passo 2: Calcule $(I - P)A(I - P)$ conforme indicado nas equações (2.6) e (2.7). Para tanto, calcule $W_{mn} = BA$ e $W_{nn} = QW_{mn}$. Então substitua W_{nn} por $A - W_{nn}$. Daí calcule $W_{nm} = W_{nn}Q$ e $A_* = W_{nm}B$. Por fim, substitua A_* por $W_{nn} - A_*$.

Passo 3: Estime o maior autovalor γ_* de A_* e substitua a matriz W_{nm} por γ_*Q . Também substitua a matriz W_{nn} por $W_{nm}B$ e, em seguida, A_* por $A_* + W_{nn}$. Se $\delta > 0$, atualize a diagonal principal de A_* adicionando $\delta\gamma_*$ a elas.

Passo 4: Calcule a solução y^* do sistema linear $R^T y = b$ e também os vetores $w_n = Qy^*$ e $v_n = Aw_n$.

Passo 5: Substitua v_n por $a - v_n$ e y^* por Bv_n . Substitua a inicialmente por Qy^* e então por $v_n - a$. Daí substitua a pela solução do sistema linear $A_*x = a$, resolvido iterativamente ou através do fator de Cholesky de A_* . Faça $x^* = a + w_n$.

Passo 6: Substitua a por Aa e v_n por $v_n - a$. Faça $y^* = Bv_n$ e então substitua y^* pela solução do sistema linear triangular $Ry = y^*$.

Dados de retorno:

A solução de (2.1): (x^*, y^*) .

2.5 Uma dedução alternativa do método de espaço nulo

Nesta seção, a representação matricial de uma transformação linear $T : \mathbb{R}^p \rightarrow \mathbb{R}^q$ relativa a bases *out* de \mathbb{R}^p e *in* de \mathbb{R}^q é denotada por $[T]_{in}^{out}$, enquanto as coordenadas de um vetor $u \in \mathbb{R}^p$ e $v \in \mathbb{R}^q$, relativas a estas mesmas bases, são denotadas por $[u]_{out}$ e $[v]_{in}$.

Seja $K \in \mathbb{R}^{n \times (n-m)}$ uma matriz de colunas K_1, \dots, K_{n-m} que geram o $\ker(B)$ e sejam $B_1, \dots, B_m \in \mathbb{R}^n$ as linhas da matriz B . Então,

$$\{K_1, \dots, K_{n-m}, B_1^T, \dots, B_m^T\}$$

é uma base ordenada para \mathbb{R}^n , devido à decomposição em soma direta exibida para este espaço na Seção 2.1. Denotamos esta base por *alt*, de alternativa, e a base canônica de

\mathbb{R}^n , por *can.* Seja, também, $K^\dagger = (K^T K)^{-1} K^T$. Como o $\ker(B)$ e a $\text{ran}(B^T)$ são espaços T_{A_*} -invariantes, o operador linear T_{A_*} tem uma representação matricial bloco-diagonal com respeito à base *alt*, a saber:

$$[T_{A_*}]_{alt}^{alt} = \begin{bmatrix} K^\dagger & \\ & (B^\dagger)^T \end{bmatrix} A_* \begin{bmatrix} K & B^T \end{bmatrix} = \begin{bmatrix} K^\dagger A K & O^T \\ O & \gamma_* B B^T \end{bmatrix}.$$

A segunda igualdade é devido às expressões das matrizes A_* e K^\dagger e às identidades

$$K^T(I - P) = K^T, \quad K^T B^T = O, \quad (B^\dagger)^T(I - P) = O \quad \text{e} \quad (B^\dagger)^T B^T = I.$$

Estas também são as razões pelas quais as coordenadas do vetor

$$\bar{a} = (I - P)(a - AB^\dagger b) + \gamma_* B^T b,$$

com relação à base *alt*, são

$$[\bar{a}]_{alt} = \begin{bmatrix} K^\dagger & \\ & (B^\dagger)^T \end{bmatrix} \bar{a} = \begin{bmatrix} K^\dagger(a - AB^\dagger b) \\ \gamma_* b \end{bmatrix}.$$

Assim $[x^*]_{alt}$ é solução do sistema linear

$$\begin{bmatrix} K^T A K & O^T \\ O & B B^T \end{bmatrix} [x]_{alt} = \begin{bmatrix} K^T(a - AB^\dagger b) \\ b \end{bmatrix},$$

porque $[T_{A_*}]_{alt}^{alt} [x]_{alt} = [\bar{a}]_{alt}$, em virtude de $A_* x = \bar{a}$, devido a (2.14). Se

$$[x^*]_{alt}^T = \begin{bmatrix} \tilde{x}^T & \hat{x}^T \end{bmatrix},$$

com $\tilde{x} \in \mathbb{R}^{n-m}$ e $\hat{x} \in \mathbb{R}^m$, então \tilde{x} é solução do sistema linear

$$(K^T A K) \tilde{x} = K^T(a - AB^\dagger b). \quad (2.19)$$

Além disso, x^* pode ser escrito a partir de \tilde{x} e de uma solução particular de $Bx = b$:

$$x^* = [x^*]_{can} = \begin{bmatrix} K & B^T \end{bmatrix} [x^*]_{alt} = K \tilde{x} + B^T \hat{x} = K \tilde{x} + B^\dagger b. \quad (2.20)$$

Esta observação e o fato conhecido do sistema linear (2.15) de que y^* é solução de

$$(B B^T) y = B(a - A x^*), \quad (2.21)$$

estabelecem uma estratégia que descreve os métodos de espaço nulo de maneira geral, a menos da escolha de uma solução particular do sistema $Bx = b$. No nosso caso, esta solução acabou sendo naturalmente fixada na escolha mais simples possível: $B^T \hat{x} = B^\dagger b$.

Em vista de (2.19)-(2.21) e de expressões semelhantes encontradas no valor de $\text{cond}_2(A_*)$, determinado na Proposição 2.10, no formato do sistema linear (2.15), que é equivalente a (2.14), e na própria expressão da solução de (2.1), dada no Teorema 2.8,

nós entendemos que é correto enxergar o método desenvolvido na seção anterior como um método de espaço nulo, que não depende de nenhuma base do $\ker(B)$. É neste sentido que dizemos que o método é livre de bases. Algumas referências para métodos iterativos do tipo gradientes conjugados pré-condicionados, que utilizam implicitamente métodos de espaço nulo livre de bases, através de projeções no $\ker(B)$, são (BARLOW; NICHOLS; PLEMMONS, 1988), (JAMES; PLEMMONS, 1990), (GOULD; HRIBAR; NOCEDAL, 2001) e (GOULD; ORBAN; REES, 2014). Observamos que os autores do terceiro artigo destacam que as projeções no $\ker(B)$, realizadas pelo método deles, acarretam significativos erros de arredondamento, que eles procuram reduzir com técnicas iterativas de refinamento e também com a reformulação dos problemas tratados. Como deve estar claro de nossa apresentação, cuidado similar também foi tomado aqui.

Afirmamos que o método proposto é útil e complementa os métodos de espaço nulo tradicionais pela seguinte razão. Quanto menor a diferença $n - m$, mais apropriada é a aplicação dos métodos de espaço nulo tradicionais à resolução de problemas de ponto-de-sela, por causa dos menores custos computacionais envolvidos na determinação de uma base do $\ker(B)$, no cálculo do produto $K^T AK$ e também na resolução do sistema linear cuja matriz de coeficientes é esta própria matriz. Conforme a diferença $n - m$ aumenta, entretanto, os métodos de espaço nulo tradicionais perdem performance em todos estes passos. Nosso procedimento, por outro lado, tem um custo de inicialização negligenciável na situação corriqueira em que $m \ll n$. De fato, pois ao contrário do que pode parecer, a obtenção da matriz A_* requer $4mn^2$ operações de ponto flutuante, a grosso modo, enquanto somente o cálculo de $K^T AK$ utiliza $[(n - m)n^2 + (n - m)^2n]$ destas operações. Para que a primeira quantidade seja menor do que a segunda, é então necessário que n e m satisfaçam a desigualdade $2n^2 - 7mn + m^2 > 0$, ou ainda, que

$$m < \left(\frac{4}{7 + \sqrt{41}} \right) n \approx \frac{3}{10}n,$$

o que trivialmente ocorre na situação indicada. Além dessa vantagem, o método desenvolvido também não necessita de uma base do $\ker(B)$, que é justamente a maior dificuldade para a aplicação dos métodos de espaço nulo tradicionais ((BENZI; GOLUB; LIESEN, 2005, p. 33-34)). Em contrapartida, a resolução do sistema linear com matriz de coeficientes A_* é computacionalmente mais cara do que a resolução do outro sistema, com matriz de coeficientes $K^T AK$. Mas quando $m \ll n$, como frequentemente ocorre, ambas as resoluções têm um custo muito similar. Assim, a única desvantagem que conseguimos enxergar do método proposto em relação aos métodos de espaço nulo tradicionais é não poder reaproveitar uma mesma matriz K de colunas que geram o $\ker(B)$ para resolver uma sequência de problemas de ponto-de-sela,

$$\begin{bmatrix} A^{(k)} & B^T \\ B & O \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a^{(k)} \\ b^{(k)} \end{bmatrix}, \quad k \in \mathbb{N}_0,$$

onde a matriz $A^{(k)} \in \mathbb{R}^{n \times n}$ e o vetor $(a^{(k)}, b^{(k)}) \in \mathbb{R}^{n+m}$ podem ser alterados com k , mas a matriz B permanece fixada. Que se trata de uma aplicação usual de métodos de espaço nulo tradicionais. Porém, mesmo essa desvantagem é, na realidade, em parte apenas aparente, pois como o custo de construção de A_* é bem menor do que o de $K^T A K$, quando $m \ll n$, até uma sequência curta de problemas de ponto-de-sela,

$$\begin{bmatrix} A^{(k)} & (B^{(k)})^T \\ B^{(k)} & O \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a^{(k)} \\ b^{(k)} \end{bmatrix}, \quad k \in \{0, 1, \dots, \ell\},$$

onde agora o bloco $B^{(k)} \in \mathbb{R}^{m \times n}$ também tem a chance de variar com k , pode ser resolvida com o nosso método, sem ultrapassar o custo adicional do cálculo de uma tal matriz K . De fato, porque a quantidade de operações de ponto flutuante que deixam de ser efetuadas com cada uma de nossas construções e com a não determinação de uma base do $\ker(B)$, compensa o custo adicional de se resolver diversos sistemas lineares de ordem n , ao invés de sistemas de ordem $n - m$. Essa é uma propriedade interessante, por exemplo, quando a sequência em questão se origina de um problema de otimização com restrições de igualdade, onde o método de Newton ou métodos *quasi*-Newton são aplicados às condições de KKT do problema, pois uma tal sequência converge rapidamente, se ela for convergente. Logo podemos ter o método desenvolvido neste trabalho em mente, como uma alternativa aos métodos de espaço nulo tradicionais, quando $m \ll n$.

Para as aplicações acima é evidentemente interessante explorar as estruturas especiais possivelmente presentes nas matrizes $A^{(k)}$ e $B^{(k)}$, para que os produtos

$$(I - P^{(k)})A^{(k)}(I - P^{(k)}),$$

onde $P^{(k)} = (B^{(k)})^\dagger B^{(k)}$, e as constantes

$$\gamma_*^{(k)} = \|(I - P^{(k)})A^{(k)}(I - P^{(k)})\|_2 / \|B^{(k)}\|_2^2,$$

ambos para $k \in \{0, 1, \dots, \ell\}$, possam ser calculadas com um número menor de operações de ponto flutuante. Infelizmente não podemos garantir que as matrizes

$$A_*^{(k)} = (I - P^{(k)})A^{(k)}(I - P^{(k)}) + \gamma_*^{(k)}(B^{(k)})^T B^{(k)}, \quad k \in \{0, 1, \dots, \ell\},$$

têm — além de um número de condição melhor — alguma estrutura especial que facilite a resolução do subsistema linear de (2.14) com matriz de coeficientes $A_*^{(k)}$. Deste modo, o conselho geral é utilizar métodos do tipo gradientes conjugados para resolvê-lo. É válido lembrar que esse também é o tratamento dado à resolução do sistema reduzido com matriz de coeficientes $K^T A^{(k)} K$ nos métodos de espaço nulo tradicionais ((BENZI; GOLUB; LIESEN, 2005, p. 33)).

3 Resolução de problemas de ponto-de-sela generalizados

As hipóteses levantadas no Capítulo 2 sobre o sistema linear (2.1), então já bastante modestas, são agora um pouco mais relaxadas para podermos analisar a resolução de problemas de ponto-de-sela generalizados. Ou seja, de sistemas da forma

$$M \begin{bmatrix} x \\ y \end{bmatrix} := \begin{bmatrix} A & B^T \\ B & O \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix}, \quad (3.1)$$

onde A é uma matriz não-simétrica. Além de interessante e possível, uma extensão da teoria desenvolvida anteriormente para este novo problema é desejada, porque existem aplicações importantes de matemática que resultam nele. Um exemplo é o problema de resolver as equações linearizadas de Navier-Stokes, cujas linearizações foram obtidas pelo processo iterativo de Picard ou por alguma variante do método de Newton, através de discretizações estáveis das equações de Navier-Stokes (no sentido explicado em (GUNZBURGER, 1989, p. 12)), realizadas com o método dos elementos finitos ((GUNZBURGER, 1989, p. 81-98)).

Seja A_s a parte simétrica da matriz A , ou seja, $A_s = 1/2(A + A^T)$. Conforme apontamos na introdução do Capítulo 2, com uma hipótese bastante natural sobre A_s e uma simples adaptação na demonstração do Teorema 3.2 encontrado em (BENZI; GOLUB; LIESEN, 2005, p. 16), é possível se garantir que o sistema (3.1) admite uma única solução.

Teorema 3.1. *Se B tem posto completo e A_s é uma matriz definida positiva no $\ker(B)$, então a matriz de coeficientes do sistema (3.1) é não-singular.*

Demonstração. Suponha que M seja singular. Então existe um vetor não-nulo (x, y) em \mathbb{R}^{n+m} tal que

$$Ax + B^T y = 0 \quad \text{e} \quad Bx = 0.$$

Da segunda expressão, nós vemos que $x \in \ker(B)$. Devido a isso, à primeira expressão e também ao fato de que somente a parte simétrica de uma matriz importa para a função quadrática que ela induz, nós ainda temos que

$$x^T A_s x = x^T A x = -x^T B^T y = -(Bx)^T y = 0.$$

Portanto $x = 0$, já que por hipótese A_s é uma matriz definida positiva no $\ker(B)$. Logo,

$$B^T y = Ax + B^T y = 0$$

e isto implica que $y = 0$, porque a matriz B tem posto-completo. Mas então $(x, y) = (0, 0)$, o que é uma contradição. Portanto, a matriz de coeficientes M é não-singular. \square

Em vista do teorema anterior, assumimos daqui em diante que B tem posto completo e que A_s é uma matriz definida positiva no $\ker(B)$. Além disso, mantemos aqui as notações B^\dagger e P do Capítulo 2. Seja

$$(A_s)_* = (I - P)A_s(I - P) + \gamma_{s,*}B^TB,$$

onde $\gamma_{s,*} = \|(I - P)A_s(I - P)\|_2 / \|B\|_2^2$. Então $\gamma_{s,*}$ é uma constante positiva e $(A_s)_*$ é uma matriz simétrica definida positiva em todo o \mathbb{R}^n , conforme já mostramos ser o caso, no capítulo anterior, para objetos análogos a estes. Vamos também supor que a matriz

$$E = (I - P)A_{ss}(A_s)_*^{-1}(I - P), \quad (3.2)$$

dada a partir de $(A_s)_*$ e da parte anti-simétrica A_{ss} da matriz A , não possui autovalor -1 . Precisamos desta condição para trivialmente assegurar que a matriz $I + E$ é não-singular, propriedade esta que será necessária logo mais. Afirmamos que a hipótese em questão é bastante modesta e não implica em nenhuma perda séria de generalidade para a teoria que apresentaremos. De fato, porque se E possuíse um tal autovalor, ainda poderíamos prosseguir com o nosso desenvolvimento, empregando nele uma matriz

$$M^{\epsilon,ss} = \begin{bmatrix} A_s + (1 - \epsilon)A_{ss} & B^T \\ B & O \end{bmatrix},$$

ligeiramente perturbada com uma constante $\epsilon \in (0, 1)$, ao invés da própria matriz M . Pois assim encontraríamos que a matriz $E_\epsilon = (1 - \epsilon)E$ não possui autovalor -1 para qualquer valor de ϵ (possivelmente a menos de um número finito deles). Este é o caso, realmente, já que $(-1/(1 - \epsilon), v)$ é um autopar de E , se $(-1, v)$ é um autopar de E_ϵ , com $v \in \mathbb{R}^n$. E existem valores de ϵ em $(0, 1)$ para os quais $-1/(1 - \epsilon)$ não pode ser um autovalor de E , porque o espectro da matriz E é discreto e o valor de ϵ é tomado em um contínuo. Uma vez que os autovalores da matriz $M^{\epsilon,ss}$ dependem continuamente de ϵ , a conclusão a que desejamos chegar para o problema original poderia, então, ser obtida, considerando-se o limite $\epsilon \rightarrow 0^+$, se ele existir. Um argumento semelhante a este foi originalmente utilizado por Notay (2014, p. 148) para problemas de ponto-de-sela simétricos e de bloco $(2, 2)$ não necessariamente nulo, também com a finalidade de se garantir a não-singularidade de certas matrizes importantes para o andamento do trabalho do autor.

Os tópicos deste capítulo são como seguem. Na Seção 3.1 nós empregamos as condições acima indicadas e a expressão exata da solução de (2.1) para também resolver o sistema linear (3.1) exatamente. Então fazemos uso do método direto desenvolvido para problemas de ponto-de-sela simétricos, juntamente com a solução recém descoberta, para estabelecer na Seção 3.2 um método direto para a resolução de (3.1). Fornecemos ainda algoritmos para ele. Na Seção 3.3 retomamos a questão das hipóteses suficientes para a resolução de problemas de ponto-de-sela, introduzida no início do Capítulo 2, procurando enfraquecer estas condições ainda mais.

3.1 A solução exata de problemas de ponto-de-sela generalizados

Para simplificar a análise que se seguirá, vamos denotar por

$$F_s = (I - P)(A_s)_*^{-1} \quad \text{e} \quad G_s = (B^\dagger)^T(I - A_s F_s)$$

os blocos $(1, 1)$ e $(2, 1)$ da inversa da matriz de coeficientes de (2.1), tomada com $A = A_s$. Temos para o primeiro deles que

$$F_s = (I - P)(A_s)_*^{-1}(I - P) = (A_s)_*^{-1}(I - P), \quad (3.3)$$

em decorrência das identidades que seguem o Corolário 2.3, todas as quais, inclusive, devem neste capítulo ser pensadas com A_s no papel da matriz A . Vamos momentaneamente também considerar o sistema linear auxiliar

$$\bar{M} \begin{bmatrix} x \\ y \end{bmatrix} := \begin{bmatrix} \bar{A} & B^T \\ B & O \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \bar{a} \\ b \end{bmatrix}, \quad (3.4)$$

onde $\bar{A} = A_s + PA_{ss}$ e $\bar{a} \in \mathbb{R}^n$ é um vetor dado, a princípio, sem qualquer relação com o vetor a de (3.1). Então as partes simétrica e anti-simétrica da matriz \bar{A} são, respectivamente,

$$\bar{A}_s = A_s + \frac{1}{2}(PA_{ss} - A_{ss}P) \quad \text{e} \quad \bar{A}_{ss} = \frac{1}{2}(PA_{ss} + A_{ss}P).$$

Afirmamos que \bar{A}_s é definida positiva no $\ker(B)$. De fato, porque

$$\tilde{x}^T \bar{A}_s \tilde{x} = \tilde{x}^T A_s \tilde{x} + \frac{1}{2}[(\tilde{x}^T P)A_{ss}\tilde{x} - \tilde{x}^T A_{ss}(P\tilde{x})] = \tilde{x}^T A_s \tilde{x} > 0$$

para todo vetor \tilde{x} não-nulo no $\ker(B)$. Nestas condições, (3.4) admite uma única solução e nós podemos — seguindo Golub e Wathen (1998) — fazer corresponder à resolução deste sistema o esquema iterativo

$$\bar{M}_s \begin{bmatrix} x^{(k+1)} \\ y^{(k+1)} \end{bmatrix} := \begin{bmatrix} \bar{A}_s & B^T \\ B & O \end{bmatrix} \begin{bmatrix} x^{(k+1)} \\ y^{(k+1)} \end{bmatrix} = \begin{bmatrix} \bar{a} - \bar{A}_{ss}x^{(k)} \\ b \end{bmatrix}, \quad k \in \mathbb{N}_0, \quad (3.5)$$

definido a partir de \bar{A}_s , \bar{A}_{ss} e de um ponto $(x^{(0)}, y^{(0)})$ arbitrário de \mathbb{R}^{n+m} . As soluções de cada uma dessas iterações são conhecidas do Teorema 2.8. Antes de enunciá-las, porém, é interessante notar que

$$(I - P)\bar{A}_s(I - P) = (I - P)A_s(I - P) \quad (3.6)$$

pela definição da matriz \bar{A}_s e, consequentemente, que

$$(\bar{A}_s)_* = (I - P)\bar{A}_s(I - P) + \left(\frac{\|(I - P)\bar{A}_s(I - P)\|_2}{\|B\|_2^2} \right) B^T B = (A_s)_*.$$

Pois assim as expressões do referido teorema podem ser escritas a partir da matriz F_s , ao invés de $(I - P)(\bar{A}_s)_*^{-1}$, e então fornecem

$$\begin{bmatrix} x^{(k+1)} \\ y^{(k+1)} \end{bmatrix} = \begin{bmatrix} F_s(\bar{a} - \bar{A}_s B^\dagger b - \bar{A}_{ss}x^{(k)}) + B^\dagger b \\ (B^\dagger)^T(I - \bar{A}_s F_s)(\bar{a} - \bar{A}_s B^\dagger b - \bar{A}_{ss}x^{(k)}) \end{bmatrix}, \quad k \in \mathbb{N}_0.$$

Segue daí que $Px^{(k+1)} = B^\dagger b$ e também que

$$x^{(k+1)} = (A_s)_*^{-1} \{ (I - P)\bar{a} - [(I - P)\bar{A}_s]B^\dagger b - [(I - P)\bar{A}_{ss}]x^{(k)} \} + B^\dagger b,$$

onde a última identidade ainda é devida a (3.3). Empregando as definições das matrizes \bar{A}_s e \bar{A}_{ss} e novamente (3.3), encontramos que

$$x^{(k+1)} = F_s[\bar{a} - (A_s - 1/2 A_{ss}P) B^\dagger b - 1/2 A_{ss}P x^{(k)}] + B^\dagger b,$$

ou ainda, que

$$x^{(k+1)} = F_s(\bar{a} - A_s B^\dagger b) + B^\dagger b, \quad k \in \mathbb{N},$$

já que $Px^{(k)} = B^\dagger b$ para $k \neq 0$. Portanto a sequência $(x^{(k)})$ é constante do seu terceiro termo em diante e convergente para

$$x^{**} = F_s(\bar{a} - A_s B^\dagger b) + B^\dagger b.$$

Substituímos este ponto limite na expressão de $y^{(k+1)}$ para concluir que a sequência $(y^{(k)})$ também é constante, desta vez do seu quarto termo em diante, e convergente para

$$y^{**} = (B^\dagger)^T (I - \bar{A}_s F_s) (\bar{a} - \bar{A}_s B^\dagger b - \bar{A}_{ss} x^{**}).$$

Afirmamos que a expressão acima pode ser escrita independentemente de \bar{A}_s e \bar{A}_{ss} . Para tanto, devemos perceber que

$$\begin{aligned} (B^\dagger)^T \bar{A}_s F_s &= (B^\dagger)^T [(A_s + 1/2 P A_{ss}) (I - P) (A_s)_*^{-1}] \\ &= (B^\dagger)^T [(I - P) A_s + P A_s + 1/2 P A_{ss}] F_s \\ &= (B^\dagger)^T (A_s + 1/2 A_{ss}) F_s, \end{aligned} \tag{3.7}$$

novamente pela definição da matriz \bar{A}_s e agora também a de F_s . Além disso,

$$\begin{aligned} \bar{a} - \bar{A}_s B^\dagger b - \bar{A}_{ss} x^{**} &= \bar{a} - A_s B^\dagger b - 1/2 (P A_{ss} - A_{ss}) B^\dagger b - 1/2 (P A_{ss} x^{**} + A_{ss} B^\dagger b) \\ &= \bar{a} - A_s B^\dagger b - 1/2 P A_{ss} (x^{**} + B^\dagger b), \end{aligned} \tag{3.8}$$

pelas definições de \bar{A}_s , \bar{A}_{ss} e x^{**} e o fato de que $Px^{**} = B^\dagger b$. Não somente:

$$(B^\dagger)^T A_{ss} (x^{**} + B^\dagger b) = (B^\dagger)^T A_{ss} F_s (\bar{a} - A_s B^\dagger b) + 2(B^\dagger)^T A_{ss} B^\dagger b, \tag{3.9}$$

pela definição de x^{**} . Logo,

$$\begin{aligned} y^{**} &= \{ (B^\dagger)^T - [(B^\dagger)^T \bar{A}_s F_s] \} (\bar{a} - \bar{A}_s B^\dagger b - \bar{A}_{ss} x^{**}) \\ &= (B^\dagger)^T [I - (A_s + 1/2 A_{ss}) F_s] [\bar{a} - A_s B^\dagger b - 1/2 P A_{ss} (x^{**} + B^\dagger b)] \\ &= (B^\dagger)^T [I - (A_s + 1/2 A_{ss}) F_s] (\bar{a} - A_s B^\dagger b) - 1/2 (B^\dagger)^T A_{ss} (x^{**} + B^\dagger b) \\ &= (B^\dagger)^T [I - (A_s + 1/2 A_{ss}) F_s - 1/2 A_{ss} F_s] (\bar{a} - A_s B^\dagger b) - (B^\dagger)^T A_{ss} B^\dagger b \\ &= (B^\dagger)^T (I - A_s F_s) (\bar{a} - A_s B^\dagger b) - (B^\dagger)^T A_{ss} [F_s (\bar{a} - A_s B^\dagger b) + B^\dagger b] \\ &= (B^\dagger)^T [(I - A F_s) (\bar{a} - A_s B^\dagger b) - A_{ss} B^\dagger b]. \end{aligned} \tag{3.10}$$

Na segunda igualdade nós usamos (3.7) e (3.8). Na terceira igualdade, distribuímos a multiplicação da matriz $(B^\dagger)^T[I - (A_s + 1/2 A_{ss})F_s]$ pelo vetor

$$\bar{a} - A_s B^\dagger b - 1/2 P A_{ss} (x^{**} + B^\dagger b),$$

empregando também as identidades $(B^\dagger)^T P = (B^\dagger)^T$ e $F_s P = O$. Na quarta igualdade, utilizamos (3.9) e coletamos termos comuns às expressões. A fórmula descoberta para y^{**} mostra que este vetor não depende de \bar{A}_s e \bar{A}_{ss} , como afirmado anteriormente.

Podemos agora concluir que $(x^{**}, y^{**}) \in \mathbb{R}^{n+m}$ é a solução de (3.4), porque

$$\begin{aligned} \bar{A}x^{**} + B^T y^{**} &= A_s x^{**} + P A_{ss} x^{**} + B^T y^{**} \\ &= A_s F_s (\bar{a} - A_s B^\dagger b) + A_s B^\dagger b \\ &\quad + P A_{ss} F_s (\bar{a} - A_s B^\dagger b) + P A_{ss} B^\dagger b \\ &\quad + P (\bar{a} - A_s B^\dagger b) - P A_s F_s (\bar{a} - A_s B^\dagger b) \\ &\quad - P A_{ss} F_s (\bar{a} - A_s B^\dagger b) - P A_{ss} B^\dagger b \\ &= [(I - P) A_s F_s] (\bar{a} - A_s B^\dagger b) + P \bar{a} + (I - P) A_s B^\dagger b \\ &= (I - P) (\bar{a} - A_s B^\dagger b) + P \bar{a} + (I - P) A_s B^\dagger b \\ &= \bar{a} \end{aligned}$$

e, trivialmente, $Bx^{**} = b$. Utilizamos a definição da matriz \bar{A} na primeira igualdade das contas logo acima e as definições dos vetores x^{**} e y^{**} na segunda igualdade, ainda decompondo, nesta última, o produto PAF_s nas partes $PA_s F_s$ e $PA_{ss} F_s$. Por fim, também empregamos (2.11) na quarta igualdade.

Retornando agora à solução de (3.1), nós vemos que

$$Ax^{**} + B^T y^{**} = \bar{A}x^{**} + (I - P)A_{ss}x^{**} + B^T y^{**} = \bar{a} + (I - P)A_{ss}x^{**}.$$

Porém gostaríamos que a última equação resultasse em a . Este será o caso se \bar{a} for solução do sistema linear

$$(I + E)\bar{a} = a - (I - P)A_{ss}G_s^T b, \quad (3.11)$$

uma vez que

$$\begin{aligned} \bar{a} + (I - P)A_{ss}x^{**} &= \bar{a} + (I - P)A_{ss}F_s(\bar{a} - A_s B^\dagger b) + (I - P)A_{ss}B^\dagger b \\ &= [I + (I - P)A_{ss}F_s]\bar{a} + (I - P)A_{ss}(I - F_s A_s)B^\dagger b \\ &= (I + E)\bar{a} + (I - P)A_{ss}G_s^T b \\ &= a. \end{aligned}$$

O sistema (3.11) é possível e, ainda, determinado, porque os autovalores de sua matriz de coeficientes são da forma $1 + \lambda$, onde λ é um autovalor da matriz E e nós assumimos anteriormente para ela que $\lambda \neq -1$.

Teorema 3.2. *Sejam $\bar{a} \in \mathbb{R}^n$ e $(x^*, y^*) \in \mathbb{R}^{n+m}$, respectivamente, as soluções dos sistemas (3.11) e (2.1), o último dos quais é considerado com $A = A_s$ e $a = \bar{a}$. Suponha que B tem posto completo, que a matriz A_s é definida positiva no $\ker(B)$ e que a matriz E não possui autovalor -1 . Então o sistema linear (3.1) admite uma única solução, cuja expressão é*

$$\begin{bmatrix} x^{**} \\ y^{**} \end{bmatrix} = \begin{bmatrix} F_s(\bar{a} - A_s B^\dagger b) + B^\dagger b \\ (B^\dagger)^T [(I - AF_s)(\bar{a} - A_s B^\dagger b) - A_{ss} B^\dagger b] \end{bmatrix} = \begin{bmatrix} x^* \\ y^* - (B^\dagger)^T A_{ss} x^* \end{bmatrix}.$$

O número de condição, na norma 2, da matriz $(A_s)_*$ nesta expressão é ainda aproximadamente o menor dentre aqueles das matrizes de forma

$$(I - P)A_s(I - P) + \gamma B^T B, \quad \gamma > 0.$$

Se $\lambda_{\min}((A_s)_*) > \lambda_{\min}(A_s)$, então também $\text{cond}_2((A_s)_*) < \text{cond}_2(A_s)$.

Demonstração. Segue do Teorema 3.1 que o sistema linear (3.1) é possível e determinado. O restante da prova é idêntico a demonstração fornecida para o Teorema 2.8, sendo a expressão para y^{**} em função de x^* e y^* diretamente decorrente de (3.10). \square

Reiteramos que a condição $\lambda_{\min}((A_s)_*) > \lambda_{\min}(A_s)$, necessária para se assegurar que $\text{cond}_2((A_s)_*) < \text{cond}_2(A_s)$, é puramente técnica e que ela frequentemente não precisa ser verificada na prática. Notamos, também, que com o auxílio do sistema linear (3.11) houve uma separação das hipóteses levantadas para o problema de ponto-de-sela generalizado, ficando a influência da condição sobre E (que está relacionada exclusivamente com a parte anti-simétrica da matriz A) totalmente restrita à resolução de (3.11). Achamos isso conveniente, porque uma vez determinada a solução \bar{a} deste sistema, a solução do problema de ponto-de-sela generalizado original tem uma expressão muito similar aquela de um problema simétrico, o que sugere que talvez seja possível elaborar, com simplicidade, métodos de resolução dos primeiros a partir de métodos de resolução dos últimos.

Corolário 3.3. *Nas condições do teorema anterior,*

$$\begin{bmatrix} F_s(I + E)^{-1} & [I - F_s(I + E)^{-1}A]B^\dagger \\ (B^\dagger)^T(I - AF_s)(I + E)^{-1} & -(B^\dagger)^T(I - AF_s)(I + E)^{-1}AB^\dagger \end{bmatrix}$$

é a inversa da matriz de coeficientes do sistema (3.1), que tem bloco (1,1) não-simétrico.

Demonstração. Para simplificar a determinação da expressão da inversa, vamos antes destacar algumas identidades. De $(I + E)^{-1}(I + E) = I$, temos que

$$(I + E)^{-1} = I - (I + E)^{-1}E. \quad (3.12)$$

Pós-multiplicando (3.12) por P , nós também vemos que

$$(I + E)^{-1}P = P - (I + E)^{-1}(EP) = P, \quad (3.13)$$

pois $EP = O$ pela definição da matriz E . Além disso, encontramos que

$$\begin{aligned}
 F_s(I + E)^{-1}[(I - P)A_{ss} + A_s] &= F_s(I + E)^{-1}[(I - P)A + PA_s] \\
 &= F_s(I + E)^{-1}(I - P)A + F_s[(I + E)^{-1}P]A_s \\
 &= F_s(I + E)^{-1}A - F_s[(I + E)^{-1}P]A \\
 &= F_s(I + E)^{-1}A,
 \end{aligned} \tag{3.14}$$

usando (3.13) na segunda e terceira igualdades e notando que $F_sP = O$ por (3.3). Se agora tratamos a e b como matrizes, ao invés de vetores, é possível substituí-las por matrizes nas expressões do Teorema 3.2. Fazendo $a = I \in \mathbb{R}^{n \times n}$ e $b = O \in \mathbb{R}^{m \times n}$, é então imediato ver que os blocos $(1, 1)$ e $(2, 1)$ de M^{-1} são, respectivamente,

$$F_s(I + E)^{-1} \quad \text{e} \quad (B^\dagger)^T(I - AF_s)(I + E)^{-1},$$

porque neste caso \bar{a} passa a ser simplesmente $(I + E)^{-1}$. Por outro lado, fazendo $a = O^T \in \mathbb{R}^{n \times m}$ e $b = I \in \mathbb{R}^{m \times m}$, descobrimos que

$$\begin{aligned}
 \bar{a} - A_s B^\dagger b &= -[(I + E)^{-1}(I - P)A_{ss}(I - F_s A_s) + A_s]B^\dagger \\
 &= -\{(I + E)^{-1}(I - P)A_{ss} + [I - (I + E)^{-1}E]A_s\}B^\dagger \\
 &= -(I + E)^{-1}[(I - P)A_{ss} + A_s]B^\dagger
 \end{aligned} \tag{3.15}$$

$$\begin{aligned}
 &= -\{(I + E)^{-1}A_{ss} - [(I + E)^{-1}P]A_{ss} + (I + E)^{-1}A_s\}B^\dagger \\
 &= -[(I + E)^{-1}A - PA_{ss}]B^\dagger.
 \end{aligned} \tag{3.16}$$

Notamos que na terceira igualdade fizemos uso de (3.12) e na quinta, de (3.13). Segue assim de (3.14) e (3.15) que

$$F_s(\bar{a} - A_s B^\dagger b) = -F_s(I + E)^{-1}AB^\dagger \tag{3.17}$$

Portanto, o bloco $(1, 2)$ de M^{-1} é

$$[I - F_s(I + E)^{-1}A]B^\dagger.$$

Resta-nos agora apenas encontrar a expressão do bloco $(2, 2)$. Da fórmula de y^{**} dada no Teorema 3.2, nós sabemos que ele é

$$(B^\dagger)^T\{(\bar{a} - A_s B^\dagger b) - A[F_s(\bar{a} - A_s B^\dagger b)] - A_{ss}B^\dagger\},$$

ou ainda,

$$-(B^\dagger)^T\{[(I + E)^{-1}A - PA_{ss}] - AF_s(I + E)^{-1}A + A_{ss}\}B^\dagger,$$

devido a (3.16) e (3.17). Mas como também $(B^\dagger)^T PA_{ss} = (B^\dagger)^T A_{ss}$, encontramos que

$$-(B^\dagger)^T(I - AF_s)(I + E)^{-1}AB^\dagger$$

é, na verdade, a expressão do bloco $(2, 2)$. Fica assim conhecida explicitamente a inversa da matriz de coeficientes do sistema linear (3.1), que tem bloco $(1, 1)$ não-simétrico. \square

Até onde sabemos, a expressão da inversa da matriz M de (3.1) exibida acima não é conhecida na literatura. Usualmente uma tal expressão explícita tem uso prático limitado e interesse majoritariamente teórico. Porém está provado que o conhecimento de expressões como esta tornam possível um entendimento mais abrangente dos problemas em análise e abrem caminho para o desenvolvimento de métodos mais eficientes de resolução dos mesmos. Para fundamentar o que dizemos, citamos novamente o artigo de [Estrin e Greif \(2015\)](#), publicado muito recentemente em um periódico internacional de grande impacto, no qual os autores empregam a expressão explícita da inversa da matriz simétrica de coeficientes de (2.1), curiosamente também recentemente fornecida por [Gansterer, Schneid e Ueberhuber \(2003, p. 11\)](#), para determinar pré-condicionadores apropriados à resolução das equações de Maxwell harmônicas no tempo. Além desse caso, encontramos no trabalho de [Powell \(2004\)](#) uma situação que depende explicitamente do conhecimento de M^{-1} , quando M é uma matriz simétrica. E este trabalho simplesmente resultou em um dos melhores e mais conhecidos algoritmos de otimização sem derivadas existente até hoje, o NEWUOA (([POWELL, 2006](#))).

3.2 Um método direto de resolução de problemas de ponto-de-sela generalizados

Como no caso simétrico, deve estar claro que um método direto de resolução de problemas de ponto-de-sela generalizados, desenvolvido a partir das expressões do Teorema 3.2, somente envolve produtos de matrizes por vetores e a resolução de uns poucos sistemas lineares auxiliares. Mas ao contrário de lá, o sistema (3.11), que possui matriz de coeficientes $I + E$, tem uma construção inviável computacionalmente. Podemos, contudo, contornar esta dificuldade, se assumirmos que o raio espectral de E é menor do que a unidade. De fato, pois neste caso a matriz E não possui autovalor -1 e então podemos aplicar o Teorema 3.2 e ainda explicitar $(I + E)^{-1}$ no formato de uma série, qualquer que seja a matriz A , de parte simétrica definida positiva no $\ker(B)$, que esteja presente no sistema (3.1). Para tanto, devemos empregar a série de Neumann de $-E$ (Lema 7.18 em ([BURDEN; FAIRES, 2005, p. 442](#))) e o fato de que $I - P$ é uma matriz idempotente, com o que encontramos que

$$(I + E)^{-1} = [I + (I - P)A_{ss}F_s]^{-1} = I + (I - P) \left[\sum_{i=1}^{\infty} (-1)^i (A_{ss}F_s)^i \right]. \quad (3.18)$$

Desta forma, conseguimos obter um vetor arbitrariamente preciso para a solução \bar{a} de (3.11), truncando a série acima e usando apenas produtos de matrizes por vetores. Esta situação — que leva à solução de (3.1) — é, por conta própria, muito melhor do que a solução iterada de um número indefinido de sistemas lineares de ordem $n + m$, como originalmente foi sugerido por [Golub e Wathen \(1998\)](#) para a resolução do sistema linear (3.1). É claro

que a hipótese que levantamos sobre a matriz E é restritiva, mas também o é a condição indicada no trabalho desses autores para finalidade análoga (a convergência do método de decomposição deles). No capítulo de experimentos numéricos desta tese identificamos um truque numérico com o qual podemos aproximar E pela matriz nula, tornando assim desnecessária a utilização da série (3.18) para a obtenção de \bar{a} . Mas mantemos esta dedução aqui porque ainda não entendemos completamente o efeito positivo que este truque tem sobre nossos algoritmos. Porém compreendemos totalmente a razão pela qual a aproximação mencionada é válida. Deixamos esta explicação para o Capítulo 4, contudo.

De acordo com o Teorema 3.2, somos naturalmente inclinados a pensar em um método direto para a resolução de (3.1), que consiste do cálculo de \bar{a} (como indicado no parágrafo anterior), da resolução do sistema linear (2.14) (tomado com $A = A_s$ e $a = \bar{a}$), e da multiplicação da solução assim obtida, pela matriz

$$\begin{bmatrix} I & O^T \\ -(B^\dagger)^T A_{ss} & I \end{bmatrix}. \quad (3.19)$$

Estes dois últimos passos são, contudo, equivalentes à resolução do mesmo sistema, mas agora pré-condicionado pela direita com a inversa da matriz (3.19). Ou seja, a execução de ambos na verdade corresponde à resolução de um sistema linear com matriz de coeficientes

$$\begin{bmatrix} (A_s)_* & O^T \\ \gamma_{s,*} B A_{ss} & \gamma_{s,*} B B^T \end{bmatrix} = \begin{bmatrix} (A_s)_* & O^T \\ O & \gamma_{s,*} B B^T \end{bmatrix} \begin{bmatrix} I & O^T \\ (B^\dagger)^T A_{ss} & I \end{bmatrix}$$

e lado direito dado pelo lado direito de (2.14), porém fazendo-se nele $A = A_s$ e $a = \bar{a}$. Digamos que ele é $(\dot{a}, \dot{b}) \in \mathbb{R}^{n+m}$. Então o sistema do qual tratamos é ainda equivalente a

$$\begin{bmatrix} (A_s)_* & O^T \\ O & \gamma_{s,*} B B^T \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \dot{a} \\ \dot{b} - \gamma_{s,*} B A_{ss} (A_s)_*^{-1} \dot{a} \end{bmatrix} \quad (3.20)$$

e para ele temos que

$$\begin{aligned} \dot{b} - \gamma_{s,*} B A_{ss} (A_s)_*^{-1} \dot{a} &= \gamma_{s,*} B (I - A_s F_s - A_{ss} F_s) (\bar{a} - A_s B^\dagger b) - \gamma_{s,*} B A_{ss} B^\dagger b \\ &= \gamma_{s,*} B [(I - A F_s) (\bar{a} - A_s B^\dagger b) - A_{ss} B^\dagger b]. \end{aligned} \quad (3.21)$$

Sabemos que a matriz de coeficientes de (3.20) tem número de condição igual a

$$\text{cond}_2((A_s)_*) = \max\{\text{cond}_2(K^T A_s K), \text{cond}_2(B B^T)\},$$

devido à análise apresentada na Seção 2.4. Assim, dada uma matriz B bem-condicionada, nós estamos certos de que a resolução do sistema linear (3.20) é mais estável do que àquela do sistema linear original (3.1), devido aos resultados fornecidos por Golub e Greif (2003), Golub, Greif e Varah (2006) e, também, em razão do nosso Teorema 2.8 e a discussão que o segue. O Algoritmo 5 resolve (3.20), reutilizando nele uma grande parte dos cálculos efetuados para a obtenção de \bar{a} , em (3.11). Deste modo os dois sistemas lineares são

Algoritmo 5: Calcula a solução de (3.1) baseado em (3.20) e (3.21).

Dados de entrada:

m, n inteiros positivos tais que $m \ll n$, um inteiro não-negativo i_{max} ,
 $B \in \mathbb{R}^{m \times n}$ de posto completo, $A \in \mathbb{R}^{n \times n}$ com parte simétrica A_s definida positiva no
 $\ker(B)$ e tal que o raio espectral de $(I - P)A_{ss}F_s$ é menor do que 1,
 $a \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $\delta \geq 0$.

Variáveis de trabalho:

$\gamma_{s,*}, \sigma_* \in \mathbb{R}$; $x^*, w_n, v_n, u_n, s_n \in \mathbb{R}^n$; $y^* \in \mathbb{R}^m$; $A_{ss}, A_*, W_{nn} \in \mathbb{R}^{n \times n}$; $R \in \mathbb{R}^{m \times m}$;
 $W_{mn}, V_{mn} \in \mathbb{R}^{m \times n}$; $W_{nm}, V_{nm} \in \mathbb{R}^{n \times m}$.

Passo 1 ao Passo 3: Execute os Passos 1 a 3 do Algoritmo 3, tomando $A = A_s$ lá.

Passo 4a: Substitua V_{nm} por W_{mn}^T e então calcule $w_n = V_{nm}b$ e $v_n = A_s w_n$.

Passo 4b: Calcule $y^* = W_{mn}v_n$ e $u_n = W_{nm}y^*$. Então inicialmente substitua u_n
pela diferença $v_n - u_n$ e depois pela solução do sistema linear $(A_s)_*x = u_n$, resolvido
iterativamente ou através do fator de Cholesky de $(A_s)_*$. Faça $u_n \leftarrow w_n - u_n$ e,
depois, $u_n \leftarrow A_{ss}u_n$. Substitua y^* por $W_{mn}u_n$ e defina $x^* = W_{nm}y^*$. Faça ainda
 $u_n \leftarrow u_n - x^*$ e $a \leftarrow a - u_n$.

Passo 4c: Faça $u_n \leftarrow a$ e $s_n \leftarrow 0$. Então, para i de 1 a i_{max} , repita: faça $y^* \leftarrow W_{mn}u_n$,
 $x^* \leftarrow W_{nm}y^*$ e $u_n \leftarrow u_n - x^*$; Substitua u_n pela solução do sistema $(A_s)_*x = u_n$;
Faça $u_n \leftarrow A_{ss}u_n$, $x^* \leftarrow (-1)^i u_n$ e $s_n \leftarrow s_n + x^*$. Assim que a (i_{max}) -ésima iteração
tiver sido executada, faça $y \leftarrow W_{mn}s_n$, $x^* \leftarrow W_{nm}y^*$ e $a \leftarrow a + s_n - x^*$.

Passo 5: Execute o Passo 5 do Algoritmo 3, tomando $A = A_s$ lá.

Passo 6: Faça $u_n \leftarrow A_{ss}a$ e $a \leftarrow A_s a$. Substitua a por $a + u_n$ e v_n por $v_n - a$. Faça
 $u_n \leftarrow A_{ss}w_n$ e $v_n \leftarrow v_n - u_n$. Faça, ainda, $y^* \leftarrow W_{mn}v_n$.

Dados de retorno:

A solução de (3.1): (x^*, y^*) .

resolvidos sem custos computacionais adicionais significativos. O parâmetro não-negativo i_{max} do referido algoritmo é responsável pelo truncamento da série exibida em (3.18). Com exceção deste truncamento, a solução do sistema linear (3.1) permanece sendo computada por meio de expressões exatas.

Se B é uma matriz moderadamente mal-condicionada, podemos novamente procurar atenuar um pouco do efeito do seu mau-condicionamento, utilizando a fatoração QR reduzida de B^T para resolver, no lugar de (3.20), um outro sistema linear, baseado em (2.16), (3.20) e (3.21). A saber:

$$\begin{bmatrix} (A_s)_* & O^T \\ O & \gamma_{s,*}R \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} (I - P)(\bar{a} - A_s Q b_Q) + \gamma_{s,*} Q b_Q \\ \gamma_{s,*} Q^T [(I - A F_s)(\bar{a} - A_s Q b_Q) - A_{ss} Q b_Q] \end{bmatrix}, \quad (3.22)$$

onde $b_Q = R^{-T}b$. O Algoritmo 6 fornecido a seguir trata deste caso.

Algoritmo 6: Calcula a solução de (3.1) resolvendo (3.22).

Dados de entrada:

m, n inteiros positivos tais que $m \ll n$, um inteiro não-negativo i_{max} ,
 $B \in \mathbb{R}^{m \times n}$ de posto completo, $A \in \mathbb{R}^{n \times n}$ com parte simétrica A_s definida positiva no
 $\ker(B)$ e tal que o raio espectral de $(I - P)A_{ss}F_s$ é menor do que 1,
 $a \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $\delta \geq 0$.

Variáveis de trabalho:

$\gamma_{s,*} \in \mathbb{R}$; $x^*, w_n, v_n, u_n, s_n \in \mathbb{R}^n$; $y^* \in \mathbb{R}^m$; $A_{ss}, A_*, W_{nn} \in \mathbb{R}^{n \times n}$; $W_{mn} \in \mathbb{R}^{m \times n}$;
 $Q, W_{nm} \in \mathbb{R}^{n \times m}$; $R \in \mathbb{R}^{m \times m}$.

Passo 1 ao Passo 3: Execute os Passos 1 a 3 do Algoritmo 4, tomando $A = A_s$ lá.

Passo 4a: Tome y^* como sendo a solução do sistema linear $R^T y = b$. Calcule $w_n = Qy$ e $v_n = A_s w_n$.

Passo 4b: Calcule $y^* = Bv_n$ e $u_n = Qy^*$. Então inicialmente substitua u_n pela
diferença $v_n - u_n$ e depois pela solução do sistema linear $(A_s)_* x = u_n$, resolvido
iterativamente ou através do fator de Cholesky de $(A_s)_*$. Faça $u_n \leftarrow w_n - u_n$ e, depois,
 $u_n \leftarrow A_{ss} u_n$. Substitua y^* por Bu_n e defina $x^* = Qy^*$. Faça ainda $u_n \leftarrow u_n - x^*$ e
 $a \leftarrow a - u_n$.

Passo 4c: Faça $u_n \leftarrow a$ e $s_n \leftarrow 0$. Então, para i de 1 a i_{max} , repita: faça $y^* \leftarrow Bu_n$,
 $x^* \leftarrow Qy$ e $u_n \leftarrow u_n - x^*$; Substitua u_n pela solução do sistema $(A_s)_* x = u_n$; Faça
 $u_n \leftarrow A_{ss} u_n$, $x^* \leftarrow (-1)^i u_n$ e $s_n \leftarrow s_n + x^*$. Assim que a (i_{max}) -ésima iteração tiver
sido executada, faça $y \leftarrow Bs_n$, $x^* \leftarrow Qy^*$ e $a \leftarrow a + s_n - x^*$.

Passo 5: Execute o Passo 5 do Algoritmo 4, exatamente como lá.

Passo 6: Faça $u_n \leftarrow A_{ss} a$ e $a \leftarrow A_s a$. Substitua a por $a + u_n$ e v_n por $v_n - a$. Faça
 $u_n \leftarrow A_{ss} w_n$ e $v_n \leftarrow v_n - u_n$. Faça, ainda, $y^* \leftarrow Bv_n$ e então substitua y^* pela
solução do sistema linear $Ry = y^*$.

Dados de retorno:

A solução de (3.1): (x^*, y^*) .

No Capítulo 4, consideraremos ainda variações dos Algoritmos 5 e 6, motivados
unicamente pelos resultados de experimentos numéricos que obteremos.

3.3 Identificando as hipóteses mais fracas possíveis para a resolução de problemas de ponto-de-sela

Nesta seção desejamos chamar atenção para a necessidade e suficiência das
hipóteses que devem ser empregadas na resolução de problemas de ponto-de-sela, tanto
simétricos, quanto generalizados, com o intuito de identificar as hipóteses mais fracas pos-
síveis para a resolução dos mesmos. Precisaremos desse conhecimento para a realização dos

experimentos numéricos que proporemos, nos quais eventualmente acabaremos perdendo a propriedade inicialmente pensada para a parte simétrica A_s da matriz A , que é a dela ser definida positiva no $\ker(B)$. Esta hipótese já é mais fraca do que aquelas usualmente vistas em trabalhos similares a este, como, por exemplo, em (GOLUB; WATHEN, 1998) e (BENZI; GOLUB, 2004), onde os autores simplesmente supõem que a matriz A_s é definida positiva em todo o \mathbb{R}^n . Mas mesmo assim ela não se mostrará suficiente para a realização dos experimentos citados. Curiosamente, os algoritmos que desenvolvemos anteriormente continuam funcionando sob essas hipóteses mais fracas e que não foram originalmente imaginadas para eles. Até onde sabemos, todos os resultados relevantes descobertos aqui são inéditos.

Lema 3.4. *A matriz $K^T A_s K$ é não-singular para toda matriz K de colunas que geram o $\ker(B)$ se, e somente se, $K^T A_s K$ é não-singular para alguma matriz K de colunas que geram o $\ker(B)$.*

Demonstração. Apenas a recíproca não é imediata. Vamos, portanto, prová-la. Seja $K_0 \in \mathbb{R}^{n \times (n-m)}$ uma matriz de colunas que geram o $\ker(B)$ e suponha que $K_0^T A_s K_0$ é não-singular. Então, toda matriz $K \in \mathbb{R}^{n \times (n-m)}$ de colunas que geram o $\ker(B)$ podem ser escritas a partir de K_0 , na forma $K = K_0 C$, onde $C \in \mathbb{R}^{(n-m) \times (n-m)}$ é não-singular. Assim $K^T A_s K = C^T (K_0^T A_s K_0) C$ e, por conseguinte,

$$\det(K^T A_s K) = (\det(C))^2 \det(K_0^T A_s K_0) \neq 0.$$

Logo $K^T A_s K$ é não-singular, qualquer que seja a matriz K de colunas que geram o $\ker(B)$ considerada. \square

Proposição 3.5. *Seja $K \in \mathbb{R}^{n \times (n-m)}$ uma matriz de colunas que geram o $\ker(B)$. Então, a matriz $K^T A_s K$ é não-singular se, e somente se, $\ker((I - P)A_s(I - P)) \cap \ker(B) = \{0\}$.*

Demonstração. Em primeiro lugar, vamos verificar a afirmação direta. Seja v um vetor na intersecção $\ker((I - P)A_s(I - P)) \cap \ker(B)$. Então

$$v \in \ker((I - P)A_s(I - P)) \quad \text{e} \quad v \in \ker(B).$$

Segue da segunda pertinência que $v = K\tilde{v}$. Da primeira delas e da identidade $I - P = K(K^T K)^{-1}K^T$, nós também temos que

$$0 = [(I - P)A_s(I - P)]v = [K(K^T K)^{-1}(K^T A_s K)(K^T K)^{-1}K^T]v.$$

Pré-multiplicando a expressão acima por K^T e recordando que $v = K\tilde{v}$, encontramos que

$$(K^T A_s K)\tilde{v} = 0.$$

Ou ainda, que $\tilde{v} = 0$, já que a matriz $K^T A_s K$ é não-singular por hipótese. Isto implica que $v = 0$. Uma vez que $0 \in \ker((I - P)A_s(I - P)) \cap \ker(B)$ trivialmente, nós concluímos que

$$\ker((I - P)A_s(I - P)) \cap \ker(B) = \{0\}, \quad (3.23)$$

como gostaríamos.

Vejam agora a afirmação recíproca. Para tanto, vamos supor que a matriz $K^T A_s K$ é singular. Então existe um vetor \tilde{v} não-nulo em \mathbb{R}^{n-m} tal que $(K^T A_s K)\tilde{v} = 0$. Seja $v = K\tilde{v}$. Sabemos que $v \neq 0$, pois a matriz K tem posto completo, e também que $v \in \ker(B)$, pela sua própria definição. Ocorre ainda que $v \in \ker((I - P)A_s(I - P))$, já que

$$\begin{aligned} [(I - P)A_s(I - P)]v &= K(K^T K)^{-1}(K^T A_s K)[(K^T K)^{-1}(K^T K)]\tilde{v} \\ &= K(K^T K)^{-1}[(K^T A_s K)\tilde{v}] \\ &= 0. \end{aligned}$$

Assim v encontra-se na intersecção $\ker((I - P)A_s(I - P)) \cap \ker(B)$, que é trivial. Portanto $v = 0$, o que é uma contradição. Logo, a matriz $K^T A_s K$ é não-singular. \square

Teorema 3.6. *Seja M a matriz simétrica de coeficientes do sistema linear (2.1), tomada com $A = A_s$, e suponha que B tem posto completo. Então M é uma matriz não-singular se, e somente se, $\ker((I - P)A_s(I - P)) \cap \ker(B) = \{0\}$.*

Demonstração. Nas condições apontadas, o Lema 3.4 de Gould (1985, p. 96) assegura que a não-singularidade da matriz M é equivalente a não-singularidade de uma matriz $K^T A_s K$, onde $K \in \mathbb{R}^{n \times (n-m)}$ é qualquer matriz de colunas que geram o $\ker(B)$. Por sua vez, a não-singularidade de $K^T A_s K$, para alguma matriz $K \in \mathbb{R}^{n \times (n-m)}$ de colunas que geram o $\ker(B)$, é equivalente a (3.23), em razão do Lema 3.4 e da Proposição 3.5. Consequentemente, a não-singularidade de M é equivalente a termos $\ker((I - P)A_s(I - P)) \cap \ker(B) = \{0\}$. \square

Nós consideramos o teorema acima mais interessante do que o Lema 3.4 de Gould (1985, p. 96), porque ele não depende de uma base do $\ker(B)$ para ser expresso, mas somente de objetos conhecidos, que são os próprios blocos da matriz M . De fato, uma vez que $P = B^T(BB^T)^{-1}B$. Além disso, a condição suficiente para a não-singularidade de M , indicada no Teorema 3.6, é mais fraca do que aquela recentemente identificada por Benzi e Golub (2004, p. 21) para o mesmo fim, a saber: que a matriz M é não-singular, se B tem posto completo, A_s é semi-definida positiva em todo o \mathbb{R}^n e $\ker(A_s) \cap \ker(B) = \{0\}$. Realmente, pois a nossa condição é implicada pela deles, mas ela não implica esta outra. Para ver que a primeira afirmação é verdadeira, tomamos $v \in \ker((I - P)A_s(I - P)) \cap \ker(B)$. Pois então $(I - P)v = v$ e daí

$$v^T A_s v = [(I - P)v]^T A_s [(I - P)v] = v^T \{[(I - P)A_s(I - P)]v\} = 0.$$

Isto implica que $v \in \ker(A)$, devido a semi-positividade da matriz A_s (Observação 7.1.6 em (HORN; JOHNSON, 2013, p. 431)). Desta forma $v = 0$, porque acabamos de mostrar que $v \in \ker(A) \cap \ker(B) = \{0\}$. Assim a intersecção $\ker((I - P)A_s(I - P)) \cap \ker(B)$ é trivial. Para poder concluir a validade da segunda assertiva, por outro lado, notamos que (3.23) implica em $\ker(A_s) \cap \ker(B) = \{0\}$, mas não necessariamente em A_s ser uma matriz semi-definida positiva. De fato, pois se $v \in \ker(A_s) \cap \ker(B)$, então

$$[(I - P)A_s(I - P)]v = (I - P)(A_s v) = 0$$

e daí $v \in \ker((I - P)A_s(I - P)) \cap \ker(B) = \{0\}$, ou ainda, $v = 0$. Porém, a matriz

$$M = \begin{bmatrix} A_s & B^T \\ B & O \end{bmatrix} = \left[\begin{array}{ccc|cc} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{array} \right] \quad (3.24)$$

é tal que $\ker((I - P)A_s(I - P)) \cap \ker(B) = \{0\}$, já que

$$I - P = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

e, por conseguinte, $(I - P)A_s(I - P) = A_s$,

$$\ker((I - P)A_s(I - P)) = \text{span}\{(1, 0, 0), (0, 1, 0)\} \quad \text{e} \quad \ker(B) = \text{span}\{(0, 0, 1)\}.$$

Mas A_s tem autovalores negativos, mostrando que, realmente, a condição de Benzi e Golub não foi satisfeita. A título de curiosidade, também notamos que a matriz

$$(A_s)_* = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

é perfeitamente bem-condicionada, muito embora ela tenha sido derivada de A_s , que é uma matriz singular.

Diante do que foi exposto, o Teorema 3.6 configura-se em um resultado mais forte do que o resultado mencionado acima de Benzi e Golub (2004, p. 21), pois chegamos à mesma conclusão que esses autores, mas com exigências menos restritivas. Na verdade, não é possível que existam outras condições para a não-singularidade da matriz M que sejam mais fracas do que esta que indicamos aqui, pois todas elas acabariam trivialmente implicando na nossa, devido a equivalência presente na última. Assim, o Teorema 3.6 é, na realidade, o resultado mais forte com o qual podemos contar neste sentido. Além disso, ele está apresentado em um formato ideal para este trabalho, porque podemos

concluir dele que os Algoritmos 3 e 4 ainda podem ser aplicados sob sua condição, que é a mais fraca possível. Colocando de outra forma, isto significa que, se um problema de ponto-de-sela simétrico como (2.1) admite uma única solução, então os Algoritmos 3 e 4 sempre conseguem determiná-la em aritmética exata. Se eles não forem capazes disso, então é porque o problema em si não possui solução única.

Vejamos que, de fato, os Algoritmos 3 e 4 ainda podem ser aplicados quando (3.23) se verifica. A condição acima implica que $(I - P)A_s(I - P)$ não é a matriz nula, pois se ela fosse, teríamos $[(I - P)A_s(I - P)]v = 0$ para todo $v \in \ker(B)$ e daí a intersecção em questão não seria trivial. Como $(I - P)A_s(I - P) \neq O$, devemos então ter que $\gamma_{s,*} > 0$, pela própria definição desta constante. Assim $\gamma_{s,*}BB^T$ é uma matriz não-singular e $(A_s)_*$ tem, em razão da demonstração da Proposição 2.7, m autovalores positivos, os quais estão associados a autovetores na $\text{ran}(B^T)$. Afirmamos que os demais $n - m$ autovalores da matriz $(A_s)_*$, que estão associados a autovetores no $\ker(B)$, novamente pela demonstração da Proposição 2.7, são todos não-nulos. Realmente, pois se (λ, v) é um autopar de $(A_s)_*$, com $v \in \ker(B)$, então (λ, v) também é um autopar de $(I - P)A_s(I - P)$, devido a demonstração mencionada. Assim,

$$\lambda v = [(I - P)A_s(I - P)]v \neq 0,$$

por causa também de (3.23). Portanto, λ tem de ser um autovalor não-nulo. Mostramos com isso que a matriz $(A_s)_*$ não possui autovalor nulo, de onde então concluímos que ela é não-singular. Esta propriedade é suficiente para se resolver o esquema iterativo (2.5), ou ainda, os sistemas lineares (2.14) e (2.17), que são as bases dos Algoritmos 3 e 4. Logo estes algoritmos ainda podem ser empregados na resolução de problemas de ponto-de-sela simétricos da forma (2.1), mesmo que eles não tenham sido originalmente deduzidos com a condição do Teorema 3.6 em mente. Para nós isso somente reforça que as construções exibidas em todo este trabalho são, provavelmente, ótimas, em um sentido que quer dizer que nenhuma outra abordagem levaria tão naturalmente às mesmas conclusões, quanto esta nossa. Ainda mais quando nelas devem também ser pesadas a exatidão do tratamento de fenômenos numéricos adversos e os custos computacionais das construções, na situação corriqueira em que $m \ll n$.

Não consideramos a condição mais fraca possível para a não-singularidade da matriz de coeficientes de (2.1), desde o início da tese, porque com ela temos de abrir mão da certeza analítica de que a escolha do valor do parâmetro γ , realizada na Seção 2.1, aproximadamente minimiza $\text{cond}_2((A_s)_*)$, entre as matrizes da forma (2.12) (onde A é tomada como A_s). Com esta outra condição, precisamos contar com os dados empíricos de Golub e Greif (2003), Golub, Greif e Varah (2006) e os nossos próprios para esse mesmo fim, todos os quais indicam que uma redução significativa no número de condição da matriz A_s de fato ocorre.

Em vista do que já foi discutido, o seguinte resultado é trivialmente suficiente

para se garantir o funcionamento dos Algoritmos 5 e 6 em condições mais fracas do que as inicialmente previstas para eles.

Teorema 3.7. *Seja M a matriz de coeficientes do sistema linear (3.1), de bloco $(1, 1)$ não-simétrico, e suponha que B tem posto completo. Então M é uma matriz não-singular se $\ker((I - P)A_s(I - P)) \cap \ker(B) = \{0\}$ e a matriz E não possui autovalor -1 .*

Demonstração. Segue da primeira hipótese, da equação (3.6) e do Teorema 3.6 que os sistemas lineares em (3.5) são possíveis e determinados. E a segunda hipótese é suficiente para se resolver o sistema (3.11). Portanto, é a própria determinação da solução exata de (3.1), realizada na Seção 3.1, que garante a validade desta assertiva. \square

Ainda não conseguimos provar a recíproca do teorema acima, mas muito nos espantaria se ela não fosse verdadeira, uma vez que a primeira hipótese deste resultado é, pelo Teorema 3.6, a condição mais fraca possível para a solubilidade de (2.1). E a solubilidade deste sistema está intrinsicamente relacionada com a solubilidade de (3.1), conforme podemos constatar da expressão exata da solução do último deles, fornecida no Teorema 3.2. Além disso, já comentamos anteriormente que a hipótese levantada sobre a matriz E é bastante modesta. Temos certeza, porém, de que a nossa condição para a não-singularidade da matriz de coeficientes de (3.1) é mais fraca do que aquela dada por Benzi e Golub (2004, p. 21) e já mencionada. Vejamos que isso é verdade, mostrando que a condição desses autores implica na nossa, mas a nossa não implica nesta outra. Para justificar a primeira afirmação, precisamos apenas mostrar que a matriz E não possui autovalor -1 , porque o fato de que a intersecção $\ker((I - P)A_s(I - P)) \cap \ker(B)$ é trivial sob a condição de Benzi e Golub (2004, p. 21) já foi demonstrada. Para o que falta provar, precisamos validar a seguinte identidade:

$$EA_s(I - P) = (I - P)A_{ss}(I - P).$$

Pois bem, recordando (3.3) e as definições das matrizes E e $(A_s)_*$, nós vemos que

$$\begin{aligned} EA_s(I - P) &= (I - P)A_{ss}(I - P)(A_s)_*^{-1}[(I - P)A_s(I - P)] \\ &= (I - P)A_{ss}(I - P)(A_s)_*^{-1}[(A_s)_* - \gamma_{s,*}B^TB] \\ &= (I - P)A_{ss}(I - P) - \gamma_{s,*}(I - P)A_{ss}[(I - P)(A_s)_*^{-1}B^T]B. \end{aligned}$$

Mas $(I - P)(A_s)_*^{-1}B^T = O$ por uma das expressões que seguem o Corolário 2.3. Assim, a identidade em questão é verdadeira. Ela implica que

$$\tilde{x}^T[EA_s(I - P)]\tilde{x} = \tilde{x}^TA_{ss}\tilde{x} = 0, \quad (3.25)$$

para todo $\tilde{x} \in \ker(B)$, já que A_{ss} é uma matriz anti-simétrica. Suponha agora que a matriz E possui um autovalor -1 , ou seja, que existe um vetor não-nulo $v \in \mathbb{R}^n$ tal que

$Ev = -v$. Como então $Pv = -PEv = 0$, também pela definição de E , nós encontramos que $v \in \ker(B)$. E mais ainda: que o vetor $(A_s)_*^{-1}v$ também encontra-se no $\ker(B)$, já que este é um espaço $T_{(A_s)_*}^{-1}$ -invariante pela Proposição 2.1. Por causa disso,

$$\begin{aligned} E^T[(A_s)_*^{-1}v] &= (I - P)(A_s)_*^{-1}A_{ss}^T(I - P)(A_s)_*^{-1}v \\ &= -(I - P)(A_s)_*^{-1}[(I - P)A_{ss}(A_s)_*^{-1}(I - P)]v \\ &= -(A_s)_*^{-1}(I - P)Ev \\ &= (A_s)_*^{-1}v. \end{aligned}$$

Na segunda igualdade nós usamos (3.3), que A_{ss} é uma matriz anti-simétrica e que $(A_s)_*^{-1}$ e $(I - P)$ comutam. Na terceira igualdade nós utilizamos novamente esta propriedade de comutação e, na última, a suposição de que $Ev = -v$ e o fato de que $(I - P)v = v$. Mas então

$$[(A_s)_*^{-1}v]^T[EA_s(I - P)][(A_s)_*^{-1}v] = 0,$$

por (3.25), e, em consequência dessas observações,

$$\begin{aligned} [(A_s)_*^{-1}v]^T A_s [(A_s)_*^{-1}v] &= \{E^T[(A_s)_*^{-1}v]\}^T A_s [(A_s)_*^{-1}v] \\ &= [(A_s)_*^{-1}v]^T [EA_s(I - P)][(A_s)_*^{-1}v] \\ &= 0. \end{aligned}$$

Em razão da Observação 7.1.6 de Horn e Johnson (2013, p. 431), nós assim temos que $(A_s)_*^{-1}v \in \ker(A)$, porque A_s é uma matriz simétrica semi-definida, por hipótese. Portanto, $(A_s)_*^{-1}v = 0$, ou ainda, $v = 0$, porque $(A_s)_*^{-1}v \in \ker(A_s) \cap \ker(B)$ e esta intersecção é trivial, também por hipótese. Isto é uma contradição. Logo somente podemos concluir que E não possui autovalor -1 .

Agora vejamos que a condição apontada no Teorema 3.7 não implica a condição de Benzi e Golub (2004, p. 21). Para tanto, vamos considerar a matriz

$$M = \begin{bmatrix} A & B^T \\ B & O \end{bmatrix} = \left[\begin{array}{ccc|cc} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & -1 & -1 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{array} \right],$$

que é quase idêntica à matriz (3.24), porém com um bloco $(1, 1)$ de parte anti-simétrica não trivial. Exatamente por isso, nós temos que $\ker((I - P)A_s(I - P)) \cap \ker(B) = \{0\}$. Além disso, a matriz E não possui autovalor -1 , uma vez que

$$(A_s)_*^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

e, então, $E = O$. Ainda assim, A_s tem autovalores negativos, mostrando que, de fato, a condição de Benzi e Golub não foi satisfeita.

Com os novos resultados desta seção, acreditamos que a confiabilidade dos métodos propostos, em aritmética de precisão exata, está assegurada, como esperaríamos mesmo que estivesse, por se tratarem de métodos diretos.

4 Experimentos numéricos

Neste capítulo analisamos os desempenhos dos Algoritmos 3, 4, 5 e 6 a partir de diversos experimentos numéricos realizados em problemas de ponto-de-sela artificialmente gerados. Procuramos, assim, focar nossa atenção na análise das propriedades algébricas dos métodos que correspondem a eles, sem, entretanto, associá-los à resolução de algum problema específico, a exemplo do que é feito em (GOLUB; GREIF; VARAH, 2006). Os experimentos realizados reforçam que as expressões encontradas para as soluções exatas dos problemas de ponto-de-sela simétrico e generalizado estão corretas, ao mesmo tempo em que confirmam que o “pré-condicionamento” que antecede a determinação dessas soluções é de fato bastante efetivo para o que ele se propõe: aumentar a precisão das soluções calculadas, resolvendo os problemas de ponto-de-sela sob condições minimamente suficientes. Implementamos os algoritmos e executamos os experimentos em Matlab, versão R2014a, para o qual a precisão da máquina é da ordem de 10^{-16} . Foram testadas duas versões de todos os algoritmos, cada uma delas correspondendo a um *solver* diferente de sistemas lineares utilizado internamente pelos algoritmos. Para facilitar comparações futuras de resultados, selecionamos o BiCGSTAB e o GMRES para as versões 1 e 2 dos algoritmos, respectivamente, que são dois métodos do tipo gradientes conjugados generalizados, largamente empregados por engenheiros e cientistas ((VAN DER VORST, 1992), (SAAD; SCHULTZ, 1986) e (KELLEY, 1995)). As implementações de ambos os *solvers* são aquelas do próprio Matlab. O computador empregado para os testes foi um desktop equipado com um processador Intel Core 2 Quad Q9550 2.83GHz @ 3.40GHz, 8GB de RAM DDR2 800MHz e sistema operacional Windows 7 SP1 instalado.

No que segue, descrevemos de que maneira os problemas de ponto-de-sela são gerados e como ocorre a escolha dos valores dos parâmetros dos algoritmos. Em síntese, utilizamos a teoria de interpolação de funções condicionalmente definidas positivas para a geração dos problemas com as propriedades que necessitamos para eles. Então analisamos individualmente os parâmetros dos algoritmos, para poder configurá-los para ter o melhor desempenho possível. Somente daí é que os testamos, resolvendo os problemas gerados.

Seja $Z = \{z_1, \dots, z_n\}$ um conjunto de n pontos de \mathbb{R}^s , ou quando for mais conveniente, a matriz de ordem $s \times n$ cujas colunas correspondem aos pontos z_1, \dots, z_n , nesta sequência. Sejam também $\Phi : \mathbb{R}^s \rightarrow \mathbb{R}$ uma função (condicionalmente) definida positiva e $e \in \mathbb{R}^n$ um vetor de componentes unitárias. Geramos os problemas de ponto-de-sela simétricos, com os quais testamos os Algoritmos 3 e 4, a partir de interpoladores de uma função $f : \mathbb{R}^s \rightarrow \mathbb{R}$ baseados em funções (condicionalmente) definidas positivas. Esse procedimento é viável porque vimos no Apêndice B que, para um conjunto Z que é

$\mathcal{P}_1(\mathbb{R}^s)$ -unisolvante, a determinação de um interpolador da forma

$$\tilde{f}_Z(x_1, \dots, x_n) = \sum_{j=1}^n \xi_j \Phi((x_1, \dots, x_n) - z_j) + (v_1 + v_2 x_1 + \dots + v_{s+1} x_s)$$

é equivalente à resolução de um sistema linear possível e determinado,

$$\begin{bmatrix} A & B^T \\ B & O \end{bmatrix} \begin{bmatrix} \xi \\ v \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix},$$

de ordem $n + m = n + (s + 1)$, onde $\xi = (\xi_1, \dots, \xi_n)$, $v = (v_1, \dots, v_m)$,

$$A = (\Phi(z_i - z_j)), \quad B = \begin{bmatrix} e & Z^T \end{bmatrix}^T, \quad a = (f(z_1), \dots, f(z_n)) \quad \text{e} \quad b = 0.$$

Não desejamos, entretanto, fixar nenhuma função f em especial para gerar os problemas. Por isso substituímos o vetor do lado direito dos sistemas por um outro, para o qual a solução dos sistemas lineares correspondentes é forçadamente um vetor de uns. Em vez de (ξ, v) , ainda utilizamos a notação usual (x, y) para o vetor de incógnitas dos sistemas.

De modo a diversificar os problemas gerados, várias funções Φ foram testadas. Fundamentalmente elas pertencem a apenas duas classes de funções, que até mesmo estão relacionadas, mas na verdade compõem uma grande variedade de problemas, porque possuem características bastante diversas. Por exemplo: enquanto a função gaussiana é simétrica, as funções de Wendland têm suporte compacto. Além disso, cada função Φ selecionada é apenas um representante de toda uma família de funções parametrizadas por um ou dois parâmetros, para a qual não temos razões para esperar um desempenho dos algoritmos muito diferente daquele observado com um de seus representantes. De fato, pois, por exemplo, $\Phi_1(x) = e^{-\|x\|_2^2}$ representa a família $\Phi_\tau(x) = e^{-\tau\|x\|_2^2}$, $\tau > 0$, de funções gaussianas e descreve suficientemente bem o comportamento de todas as outras funções da família a qual pertence. Portanto o desempenho dos algoritmos aplicados a problemas de ponto-de-sela gerados com Φ_1 e Φ_τ devem ser próximos. Logo o leque considerado de funções é realmente grande e a análise feita, bastante abrangente.

Quando Φ é uma função estritamente condicionalmente definida positiva, isto é, quando Φ é condicionalmente definida positiva, mas não é definida positiva, a matriz A determinada com base nela é definida positiva no $\ker(B)$, mas não em todo o \mathbb{R}^n . Assim, ao empregar uma função Φ estritamente condicionalmente definida positiva para gerar os problemas de ponto-de-sela simétricos — e como já veremos, os generalizados também —, conseguimos analisar os Algoritmos 3, 4, 5 e 6 na condição inicialmente ponderada no Capítulo 2 para o bloco (1, 1) de todos esses problemas. A saber: que (a parte simétrica de) A é definida positiva no $\ker(B)$. Segundo nossa pesquisa bibliográfica, essa condição não é frequentemente considerada em trabalhos da área, possivelmente por causar dificuldades analíticas adicionais ao tratamento dos problemas. Qualquer que seja a razão concreta para isso, gostaríamos de ressaltar que foi principalmente devido ao levantamento dessa

hipótese que o desenvolvimento de experimentos numéricos apropriados para a avaliação dos nossos algoritmos acabou se tornando uma tarefa bastante desafiadora.

A unisolvência do conjunto Z , necessária ao esquema de geração de problemas de ponto-de-sela simétricos, é equivalente à condição $\text{rank}(B) = m$. Para construirmos uma matriz B de posto completo nós geramos os pontos z_1, \dots, z_n do conjunto Z aleatoriamente, mas controlamos o ângulo entre quaisquer duas linhas $\begin{bmatrix} 1 & z_i^T \end{bmatrix}$ e $\begin{bmatrix} 1 & z_j^T \end{bmatrix}$ de B^T , de maneira que ele nunca seja inferior a um determinado valor θ_0 previamente fixado. Este controle é realizado a cada novo ponto gerado, comparando o ângulo que ele faz com todos os pontos anteriormente gerados e já incluídos em Z . Caso algum desses ângulos seja menor do que θ_0 , o ponto é rejeitado e outro ponto é gerado em seu lugar até que a referida condição seja inteiramente satisfeita. Somente daí o novo ponto é incluído em Z e o procedimento pode ser repetido até se obter n pontos.

Fixado θ_0 , evidentemente existe uma relação entre a quantidade de pontos que podem ser gerados com esse procedimento e o número de componentes de cada ponto gerado. Pois, por exemplo, é impossível encontrar 5 pontos em \mathbb{R}^2 cujos ângulos relativos um ao outro sejam todos superiores a 90° . Tomando-se os devidos cuidados relativos a θ_0 e a proporção de pontos e componentes dos pontos, não é difícil, no entanto, se obter um conjunto Z adequado para a elaboração dos experimentos numéricos.

Ao controlar o posicionamento relativo das linhas de B^T , controlamos também o condicionamento desta matriz. Isso é importante nas implementações tradicionais de um método de espaço nulo porque elas são sensíveis ao valor de $\text{cond}_2(B)$ (FLETCHER; JOHNSON, 1995, p. 1). Com nossos algoritmos a situação não é diferente, já que uma matriz B mal-condicionada afeta o condicionamento da matriz BB^T e sabemos que esta última influencia diretamente a estabilidade dos algoritmos propostos, segundo a Proposição 2.10 do Capítulo 2. Por essa razão fixamos $\theta_0 = 18^\circ$, que é um valor que se mostrou suficiente para que os algoritmos não sofressem inadvertidamente com este problema durante a realização dos experimentos numéricos.

Os problemas de ponto-de-sela generalizados são gerados da mesma maneira que os problemas de ponto-de-sela simétricos, mas adicionamos ao bloco $(1, 1)$, digamos A_s , uma matriz anti-simétrica A_{ss} não-nula, gerada aleatoriamente. Esta transformação preserva a positividade inicial da matriz A_s no $\ker(B)$, porque

$$x^T(A_s + A_{ss})x = x^T A_s x$$

para todo $x \in \mathbb{R}^n$. Assim também são preservadas as condições mais interessantes para aplicação dos algoritmos propostos — já discutidas. Um escalamento é feito em A_{ss} antes desta adição, para que a média dos valores absolutos das entradas de A_{ss} seja a mesma que a média dos valores absolutos das entradas da matriz A_s . Com isso nenhuma das partes simétrica e anti-simétrica do bloco $(1, 1)$ dos problemas de ponto-de-sela generalizados

predomina sobre a outra. O vetor do lado direito dos sistemas ainda é recalculado após esta adição, de modo que a solução do problema continua sendo um vetor de uns.

Todas as operações que envolvem a geração de números aleatórios podem ser fielmente reproduzidas para análise posterior, se desejado, pois antes de cada uma delas reinicializamos a semente do gerador de números aleatórios do Matlab e especificamos a maneira como ele faz isso com a opção `generator='twister'`. As sementes são fornecidas nos próprios códigos dos experimentos, presentes no Apêndice C, e todas são calculadas a partir de um único valor base: `seed=1`.

Temos então detalhada até aqui a construção básica dos problemas de ponto-de-sela nos quais aplicaremos nossos algoritmos. Resta-nos, contudo, ainda explicar como alterar os problemas gerados dessa forma para poder analisar o desempenho dos algoritmos com relação a um mau-condicionamento crescente das matrizes A e B , isto é, dos fatores problemáticos para a resolução precisa de (2.1). Assumimos daqui em diante que, sempre que o bloco A ou B de um sistema de ponto-de-sela for modificado, o vetor do lado direito do sistema também será modificado para que a solução do problema permaneça sendo um vetor de uns.

Seja $\lambda_{\min}(C)$ o menor autovalor, em valor absoluto, de uma matriz quadrada C . Para fazer A arbitrariamente mal-condicionada, consideramos trazer $\lambda_{\min}(A)$ gradativamente para próximo de zero. Assim substituímos a matriz A em nossos problemas por $A - \mu I$, onde

$$\mu = \left(1 - \frac{1}{10^\iota}\right) \lambda_{\min}(A) \in \mathbb{C},$$

e fazemos μ tender rapidamente a $\lambda_{\min}(A)$. Para tanto, inicializamos a variável ι com um valor $\iota_0 \in \mathbb{R}$ escolhido de acordo com a ordem da matriz A e da própria função Φ empregada em sua construção e incrementamos ι de uma em uma unidade até que μ fique suficientemente próximo de $\lambda_{\min}(A)$. Utilizamos diferentes valores de ι_0 para diferentes funções Φ , porque com isso conseguimos variar a ordem de magnitude do número de condição de $A - \mu I$ com apenas um incremento de ι , praticamente sempre por um fator de aproximadamente 10. Isto foi particularmente útil aos experimentos numéricos, porque o comando `cond` do Matlab é muito lento e sem este ajuste diversos incrementos de ι eram necessários até que se confirmasse que o valor de $\text{cond}_2(A - \mu I)$ havia sido suficientemente aumentado para poder se realizar uma nova coleta de dados. Este procedimento apresenta uma limitação numérica, no entanto, pois a partir de um determinado valor de ι , que varia com a ordem de A e a função Φ que a define, a ordem de magnitude de $\text{cond}_2(A - \mu I)$, calculada com o comando `cond`, estagna. Assim, para algumas funções Φ nós conseguimos elevar $\text{cond}_2(A - \mu I)$ até a ordem 10^{16} , por exemplo, mas para outras, somente até a ordem 10^{12} , ou um pouco mais. Seja como for, esses valores já bastam para nossos testes.

Seja $K \in \mathbb{R}^{n \times (n-m)}$ uma matriz de colunas ortonormais que geram o núcleo de

B . A positividade da parte simétrica A_s da matriz A no $\ker(B)$ pode eventualmente ser perdida com a transformação do parágrafo anterior, uma vez que ela também promove um deslocamento dos autovalores de $K^T AK$ e conseqüentemente altera a forma quadrática que esta matriz induz. De fato, pois se $(K^T AK - \mu I)_s$ denota a parte simétrica da matriz $K^T AK - \mu I$ então

$$\tilde{x}^T [K^T (A - \mu I) K] \tilde{x} = \tilde{x}^T (K^T AK - \mu I)_s \tilde{x} = \tilde{x}^T (K^T A_s K - \mu I) \tilde{x}$$

para todo $\tilde{x} \in \mathbb{R}^{n-m}$. Assim, se $\lambda_{\min}(A) \in (0, \infty)$, a propriedade mencionada é retirada do problema para qualquer valor de $\mu \geq \lambda_{\min}(K^T A_s K)$. Por outro lado, se $\lambda_{\min}(A)$ tem parte imaginária não-nula, falar em positividade da parte simétrica da matriz $A - \mu I$ no $\ker(B)$ nem sequer faz sentido.

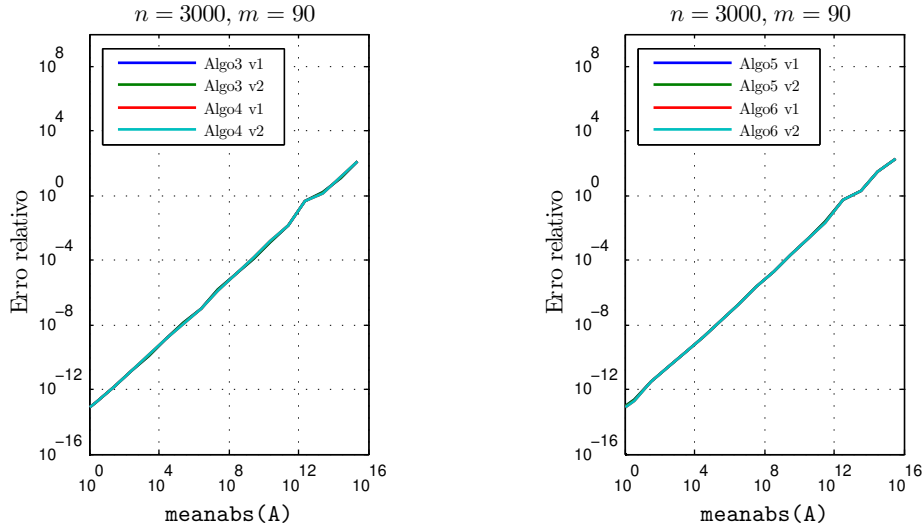
Ao contrário do que pode parecer à primeira vista, a possível perda da positividade de A_s no $\ker(B)$ não constitui nenhuma dificuldade séria para a realização dos experimentos numéricos. Muito pelo contrário: ela nos deixa analisar os Algoritmos 3, 4, 5 e 6 nas condições mais restritivas possíveis, porque estes algoritmos também podem ser utilizados quando a intersecção

$$\ker((I - P)(A_s - \mu I)(I - P)) \cap \ker(B)$$

é trivial, devido as discussões que seguem os Teoremas 3.6 e 3.7. Segundo a Proposição 3.5 e o Lema 3.4, podemos assegurar esta condição aos experimentos fazendo com que todos os autovalores da matriz $K^T (A_s - \mu I) K$ sejam não-nulos. Para tanto, basta garantirmos que $\mu \neq \lambda_{\min}(K^T A_s K)$, o que é muito fácil de se obter. Notamos, também, que a segunda hipótese solicitada pelo Teorema 3.7 não precisa ser ponderada nestes experimentos e mais adiante justificamos o porquê disso.

A transformação que substitui a matriz A pela matriz $A - \mu I$ tem ainda uma outra característica interessante, que é a de não modificar significativamente o tamanho médio das entradas de A para um valor moderado de $|\mu|$. Isto é importante em nossa análise porque não queremos que erros de arredondamento provenientes de um mau escalamento dos blocos dos problemas de ponto-de-sela interfiram em nossos experimentos, se o mal escalamento for causado exclusivamente por uma tal transformação. E temos boas razões para tomarmos esse cuidado, porque a precisão das soluções calculadas com nossos algoritmos, em aritmética de precisão finita, depende fortemente da escala do bloco A . Para ver que isso é verdade, podemos gerar um problema de ponto-de-sela, simétrico ou generalizado, a partir de uma função (condicionalmente) definida positiva e, por um momento, derivar dele uma sequência de problemas de ponto-de-sela obtidos através da substituição de A por $10A$, e de $10A$ por $100A$, e assim sucessivamente. A aplicação dos Algoritmos 3, 4, 5 e 6 à resolução destes problemas retorna erros relativos que crescem linearmente em magnitude, acompanhando o crescimento das próprias entradas da matriz A , conforme ilustra a Figura 1. Esse comportamento não é completamente inesperado,

Figura 1 – Efeito da escala do bloco A sobre os algoritmos. Função base: $\phi(r) = (1 + r^2)^{-\frac{1}{2}}$. Parâmetros: $\delta = 10^{16}$ e $\text{gmode} = \text{'fast'}$ para os Algoritmos 5 e 6.



porém, pois Rozložník e Simoncini (2002, p. 384-387) mostraram que a precisão do erro associado à solução computada pelo método dos gradientes conjugados pré-condicionado de um sistema (2.1) de bloco (1, 1) simétrico definido positivo, pré-condicionado pela direita com a matriz

$$\begin{bmatrix} I & B^T \\ B & O \end{bmatrix},$$

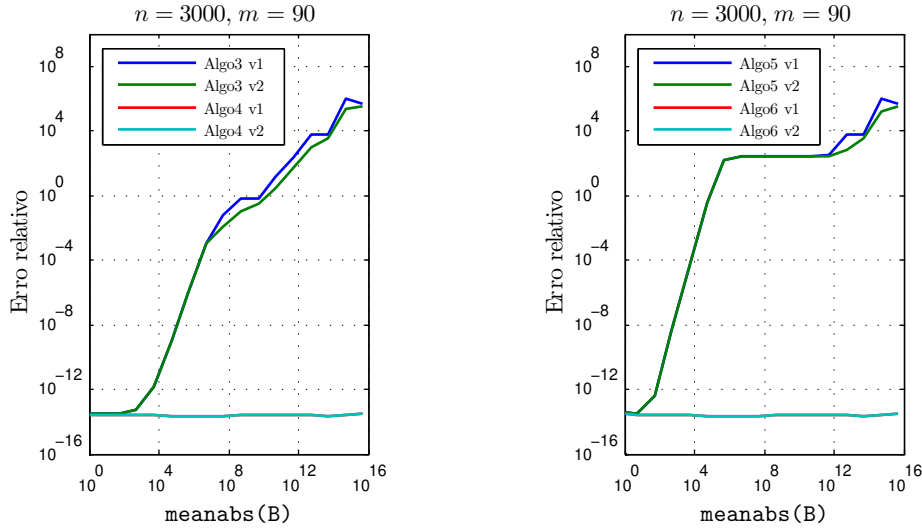
depende da escala de A . E podemos apontar algumas semelhanças entre os nossos métodos e o método desses autores, devidas em sua maior parte ao fato do pré-condicionador utilizado por eles projetar o problema original no núcleo de B (em um sentido explicado em (PERUGIA; SIMONCINI; ARIOLI, 1999, p. 1097-1098)).

Vejamos agora o efeito da escala da matriz B sobre nossos métodos. De maneira análoga àquela empregada para o bloco A , podemos por um momento considerar uma sequência de problemas de ponto-de-sela obtidos através da substituição de B por $10B$, e de $10B$ por $100B$, e assim sucessivamente. Conforme ilustra a Figura 2, os Algoritmos 3 e 5 também sofrem acentuadamente com o crescimento da magnitude das entradas de B . Já os Algoritmos 4 e 6 são indiferentes a ele.

Os resultados discutidos acima se repetem variando-se o problema que dá origem às sequências $(A, 10A, 100A, \dots)$ e $(B, 10B, 100B, \dots)$. Motivados por essas observações puramente empíricas, implementamos na versão final dos nossos métodos a opção $\text{smode} = \text{'scale'}$, com a qual calculamos a solução do sistema linear escalado

$$\begin{bmatrix} A & (\eta B)^T \\ \eta B & O \end{bmatrix} \begin{bmatrix} \bar{x} \\ \bar{y} \end{bmatrix} = \begin{bmatrix} a \\ \eta b \end{bmatrix}, \quad \eta = \frac{m}{n} \cdot \frac{\sum_i \sum_j |(A_s)_{ij}|}{\sum_i \sum_j |B_{ij}|} > 0, \quad (4.1)$$

Figura 2 – Efeito da escala do bloco B sobre os algoritmos. Função base: $\phi(r) = (1 + r^2)^{-\frac{1}{2}}$. Parâmetros: $\delta = 10^{16}$ e `gmode='fast'` para os Algoritmos 5 e 6.

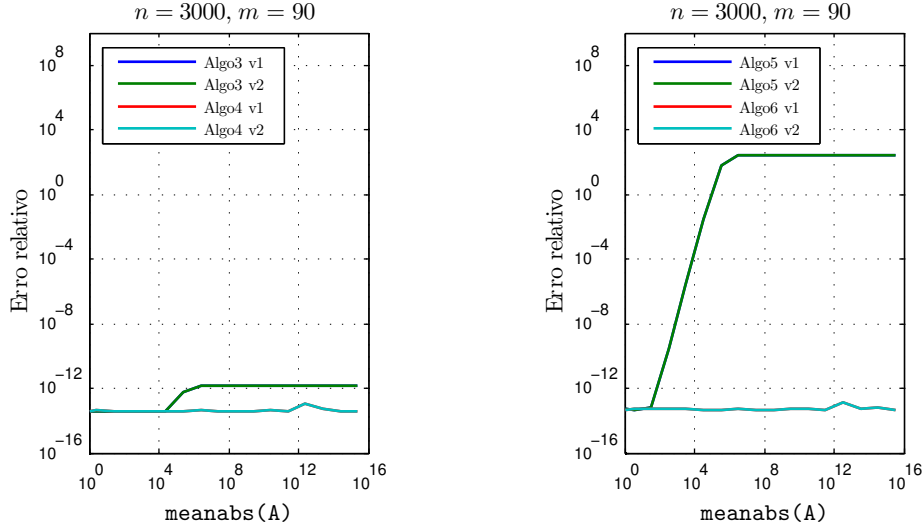


ao invés da própria solução de (2.1). Claramente as duas soluções estão relacionadas pelas equações $x = \bar{x}$ e $y = \eta \bar{y}$. A Figura 3 mostra os erros relativos associados à resolução da mesma sequência de problemas da Figura 1, mas previamente escalados com η . Apesar de não ter havido nenhuma melhora significativa no desempenho do Algoritmo 5, vemos na figura citada que o impacto do crescimento da escala da matriz A sobre o Algoritmo 3 agora é muito menor do que antes, sendo ainda praticamente nulo sobre os Algoritmos 4 e 6. Trata-se, portanto, de um escalamento recomendado para o caso em que a magnitude das entradas de A é grande. Em particular, ele faz dos Algoritmos 4 e 6 opções completamente insensíveis às escalas dos blocos de um problema de ponto-de-sela.

A opção `smode='scale'` é particularmente útil para problemas de ponto-de-sela originados de problemas de otimização com restrições, já que o escalamento feito com η somente modifica o vetor de multiplicadores de Lagrange do problema e nem precisa ser desfeito se apenas o sinal dos multiplicadores for relevante. Desligamos a opção `smode='scale'` nos experimentos numéricos que realizamos, entretanto, porque a substituição da matriz A pela matriz $A - \mu I$ pouco altera a razão entre as escalas dos blocos A e B e queremos que os resultados dependam de um parâmetro a menos.

O procedimento para fazer a matriz B cada vez mais mal-condicionada foi emprestado de Fletcher e Johnson (1995) e consiste em substituir gradativamente as últimas linhas de B pelas primeiras linhas da matriz de Hilbert H de ordem n . As substituições ocorrem na seguinte ordem: a última linha de B pela primeira linha de H , as duas últimas linhas de B pelas duas primeiras linhas de H , e assim por diante. Este procedimento promove um rápido crescimento da ordem de magnitude de $\text{cond}_2(B)$, que nos experimentos elaborados se mostrou exponencial em função do número de linhas

Figura 3 – Efeito da escala do bloco A sobre os algoritmos, para problemas pré-escalados com a constante η , segundo (4.1). Função base: $\phi(r) = (1 + r^2)^{-\frac{1}{2}}$. Parâmetros: $\delta = 10^{16}$ e `smode='scale'` para todos os algoritmos e também `gmode='fast'` para os Algoritmos 5 e 6.



substituídas.

Agora é oportuno discutir sobre os parâmetros dos algoritmos implementados. Todos eles dependem de uma constante positiva δ , a partir da qual soma-se $\delta\gamma_*$ a cada entrada diagonal da matriz A_* (no caso simétrico) ou soma-se $\delta\gamma_{s,*}$ a cada entrada diagonal da matriz $(A_s)_*$ (no caso generalizado). A função de δ é, portanto, fazer A_* ou $(A_s)_*$ mais diagonalmente dominante e auxiliar, assim, a execução numérica de operações com estes blocos, como uma fatoração de Cholesky ou uma eliminação gaussiana. Este parâmetro apresenta ainda uma outra função, completamente inesperada, que pode ser observada nas Figuras 4 e 5. Vemos nelas que as curvas dos erros relativos dos Algoritmos 3, 4, 5 e 6, inicialmente bastante distintas, gradualmente se justapõem (visualmente e relativamente às escalas dos gráficos) para valores crescentes de δ , como se existissem assíntotas para os erros relativos desses algoritmos correspondendo a um valor grande de δ . Mais interessante ainda: assim que as curvas das variações de um mesmo algoritmo convergem (visualmente e relativamente às escalas dos gráficos) para uma “assíntota” comum a ambas as variações, a “assíntota” correspondente passa a sofrer uma espécie de translação com o contínuo crescimento do valor de δ , atingindo um patamar mínimo em que todos os algoritmos têm praticamente o mesmo desempenho e do qual é impossível haver mais qualquer melhora. Fenômeno idêntico também ocorre com os resíduos relativos dos algoritmos e se repete variando-se a função Φ que determina as matrizes A e A_s . Admitimos que não sabemos explicar a razão para esse fenômeno, pois esperávamos que perturbações cada vez maiores no bloco (1,1) dos problemas de ponto-de-sela prejudicassem cada vez mais a precisão das soluções calculadas. Entendemos, porém, como incrementos maiores no valor de δ

aumentam a estabilidade dos sistemas lineares cuja matriz de coeficientes é

$$A_{*,\delta} = A_* + \delta\gamma_* I \quad \text{ou} \quad (A_s)_{*,\delta} = (A_s)_* + \delta\gamma_{s,*} I.$$

E isso explica a “convergência” para uma única curva das curvas correspondentes a variações de um mesmo algoritmo, no caso dos resíduos relativos. O que ocorre é que o número de condição de $A_{*,\delta}$ e $(A_s)_{*,\delta}$, na norma 2, tende para 1 quando δ tende a infinito. De fato, seja $\lambda_{\max}(C)$ o maior autovalor, em valor absoluto, de uma matriz quadrada C . Então

$$\text{cond}_2(A_{*,\delta}) = \frac{\lambda_{\max}(A_* + \delta\gamma_* I)}{\lambda_{\min}(A_* + \delta\gamma_* I)} = \frac{\lambda_{\max}(A_*) + \delta\gamma_*}{\lambda_{\min}(A_*) + \delta\gamma_*} \rightarrow 1, \quad \delta \rightarrow \infty,$$

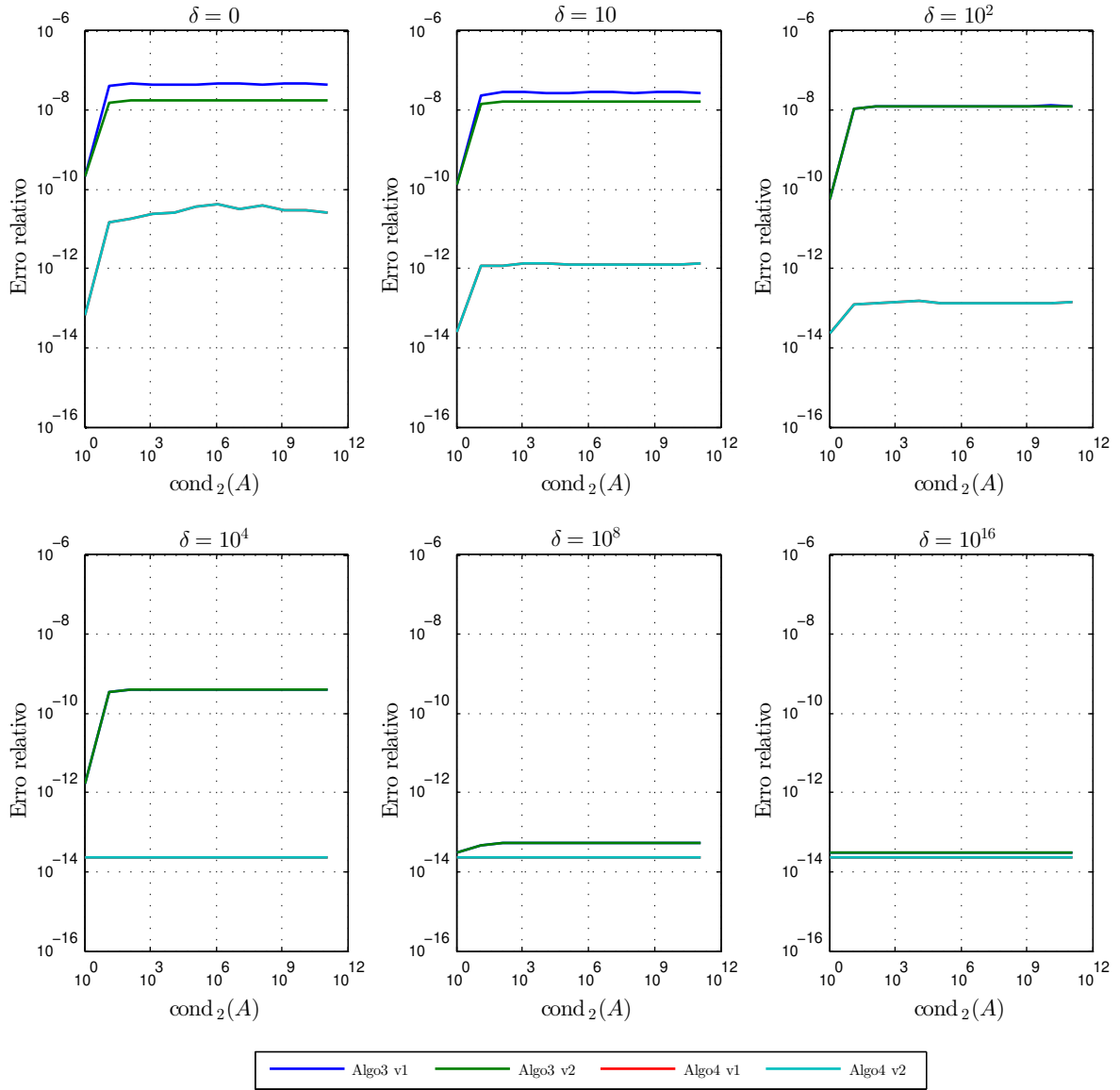
e igualmente para $(A_s)_{*,\delta}$. Assim os resíduos relativos associados aos iterandos $x_{BiCGSTAB}^{(k)}$ e $x_{GMRES}^{(k)}$, calculados pelas chamadas internas do BiCGSTAB e GMRES nos Algoritmos 3, 4, 5 e 6 durante a resolução dos sistemas lineares citados, têm de se aproximar — assumindo-se obviamente a convergência das sequências de iterandos. Realmente, porque para um número suficientemente grande de iterações, $x_{BiCGSTAB}^{(k)}$ e $x_{GMRES}^{(k)}$ passam a pertencer a uma mesma bola de raio muito pequeno e centro na solução x^* . Consequentemente,

$$A_{*,\delta} \cdot x_{BiCGSTAB}^{(k)} \text{ e } A_{*,\delta} \cdot x_{GMRES}^{(k)} \quad \text{e} \quad (A_s)_{*,\delta} \cdot x_{BiCGSTAB}^{(k)} \text{ e } (A_s)_{*,\delta} \cdot x_{GMRES}^{(k)}$$

não podem ser vetores muito diferentes, já que as matrizes $A_{*,\delta}$ e $(A_s)_{*,\delta}$ são praticamente perfeitamente bem-condicionadas para um valor muito grande de δ e os iterandos $x_{BiCGSTAB}^{(k)}$ e $x_{GMRES}^{(k)}$ são, para todos os fins, pequenas perturbações um do outro.

O método dos gradientes conjugados pré-condicionado de Rozložník e Simoncini (2002, p. 369) também é controverso, no que diz respeito aos comportamentos teórico predito e prático observado. Esses autores concluíram que um pré-escalamento diagonal de um bloco simétrico A pode não somente garantir a convergência do método que desenvolveram, mas também pode preservar a estabilidade numérica do esquema deles em aritmética de precisão finita. Resguardadas as devidas diferenças entre os métodos, a conclusão desses autores sugere que o estranho comportamento observado com os Algoritmos 3, 4, 5 e 6 é devido a efeitos ocasionados por cálculos realizados em aritmética de precisão finita, já que a adição da matriz $\delta\gamma_* I$ à A_* ou $\delta\gamma_{s,*} I$ à $(A_s)_*$ também pode ser vista como um escalamento diagonal de A_* ou $(A_s)_*$. Ficamos assim com a impressão de que a diminuição provavelmente extrema de fenômenos numéricos adversos, em decorrência da utilização de um parâmetro δ de valor grande, supera a imprecisão que um termo extra $\delta\gamma_* I$ ou $\delta\gamma_{s,*} I$ traz para o problema. Deste modo, mesmo a solução computada do problema perturbado — que é de fato o que calculamos quando usamos qualquer δ positivo — é mais precisa do que a solução computada do problema não perturbado, porque, no todo, ela carrega menos erros de cálculos. Queremos dizer: parece que as expressões deduzidas para as soluções dos problemas de ponto-de-sela simétrico e generalizado, quando avaliadas em aritmética de precisão finita, trazem consigo uma quantidade de erro p , enquanto as expressões

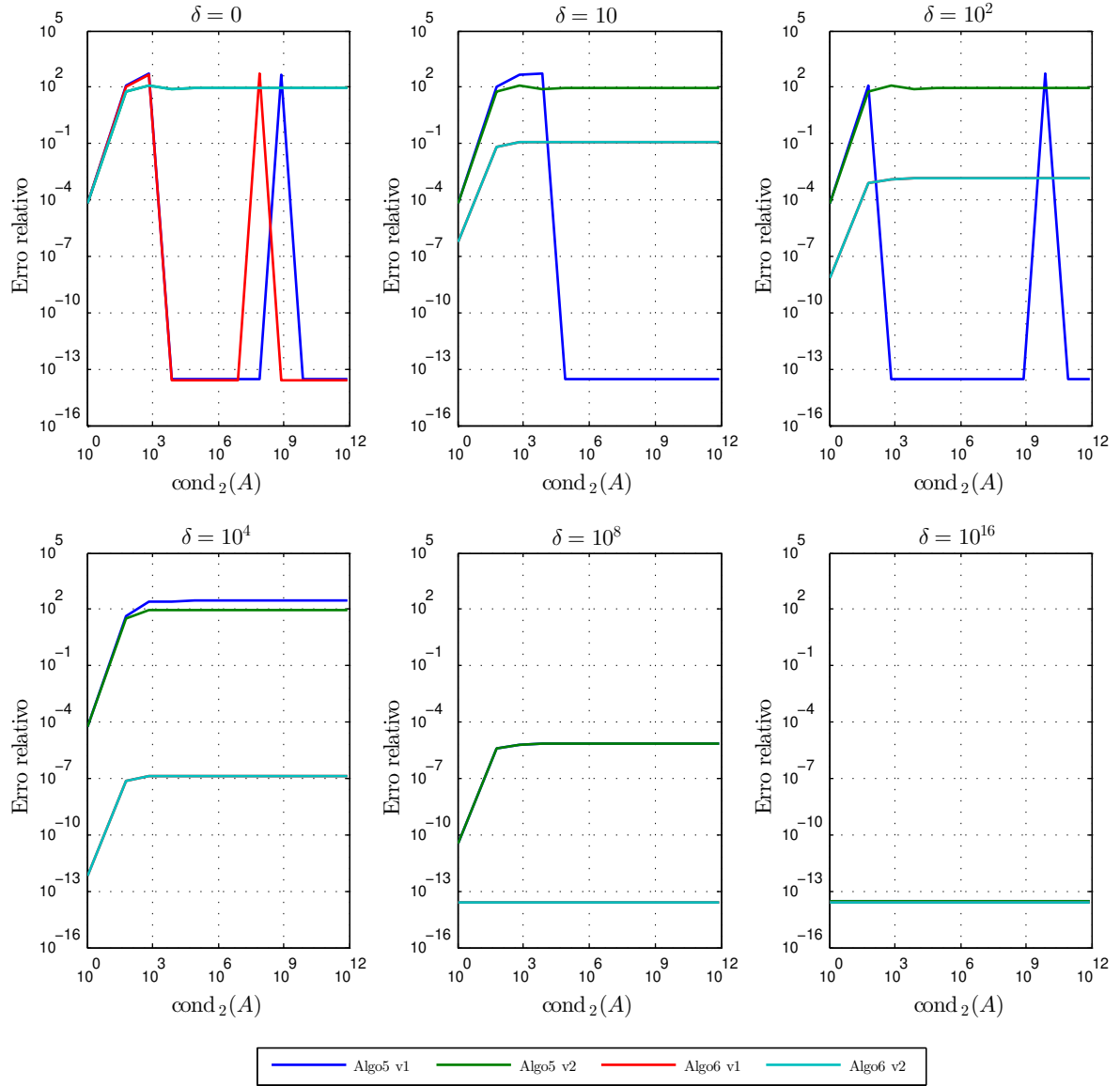
Figura 4 – Efeito do parâmetro δ sobre os Algoritmos 3 e 4, fixados $n = 3000$ e $m = 90$. Função base: $\phi(r) = e^{-r^2}$.



perturbadas correspondentes a elas trazem, pelo mesmo motivo, uma quantidade de erro q , além de uma quantidade de erro r devida à introdução inicial do termo extra $\delta\gamma_* I$ ou $\delta\gamma_{s,*} I$, satisfazendo a relação $q + r \ll p$. Seja como for, trata-se de uma questão que ainda requer uma análise mais aprofundada e que pretendemos desenvolver futuramente. O que é certo é que este é um fenômeno bastante bem-vindo, já que simplifica a análise dos algoritmos propostos ao reduzir o número de curvas nos gráficos. E é ainda um fenômeno crucial para os Algoritmos 5 e 6, uma vez que estes algoritmos não exibiram bons desempenhos numéricos para valores pequenos de δ , como ilustra a própria Figura 5. Por isso fazemos $\delta = 10^{16}$ como valor padrão para este parâmetro em todos os algoritmos.

Os Algoritmos 5 e 6 dependem ainda do parâmetro i_{max} . Este parâmetro está

Figura 5 – Efeito do parâmetro δ sobre os Algoritmos 5 e 6, fixados $n = 3000$ e $m = 90$. Função base: $\phi(r) = e^{-r^2}$. Outros parâmetros: `gmode='fast'`.



relacionado com a resolução do sistema linear (3.11), ou explicitamente

$$[I + (I - P)A_{ss}(A_s)_*^{-1}(I - P)]\bar{a} = a - (I - P)A_{ss}[I - (I - P)(A_s)_*^{-1}A_s]B^\dagger b,$$

para o qual supomos que a matriz

$$E = (I - P)A_{ss}(A_s)_*^{-1}(I - P) \quad (4.2)$$

não tem autovalor -1 . Claramente não é uma boa estratégia construir a matriz $I + E$ para se resolver o sistema acima, porque o custo computacional para se obter E explicitamente é da ordem de n^3 operações de ponto flutuante. Ainda assim, implementamos em nossos algoritmos a opção `gmode='slow'` para fazer este cálculo, com a intenção de verificar

numericamente se a expressão encontrada para a solução do problema de ponto-de-sela generalizado está correta, excluídas obviamente possíveis adversidades numéricas.

No Capítulo 3, procuramos resolver (3.11) mais eficientemente, escrevendo a inversa de $I + E$ como uma série de Neumann e truncando o produto

$$(I + E)^{-1} \{a - (I - P)A_{ss}[I - (I - P)(A_s)_*^{-1}A_s]B^\dagger b\} \quad (4.3)$$

em seus primeiros termos. Em contrapartida, tivemos de supor que o raio espectral $\rho(E)$ da matriz E é menor do que a unidade, que é uma hipótese mais restritiva para se ter no problema. Implementamos a opção `gmode='other'` nos algoritmos para executar esta resolução. Não a testamos exaustivamente, entretanto, por ser bastante lento gerar problemas de ponto-de-sela para os quais E satisfaz a condição mencionada e por não termos nenhuma garantia de que esta condição é preservada após sucessivas aplicações da transformação que substitui o bloco A por $A - \mu I$, necessárias a um dos experimentos que realizamos.

Modificamos o bloco (1, 1) de um problema de ponto-de-sela generalizado — ainda não alterado por nenhuma das transformações anteriormente discutidas — para fazer a matriz (4.2) não ter autovalor -1 da seguinte maneira. A partir da parte simétrica de A , construímos a matriz $(A_s)_*$ seguindo os passos de ambos os Algoritmos 5 e 6. Obtemos com isso duas versões desta mesma matriz: ${}^5(A_s)_*$ e ${}^6(A_s)_*$, respectivamente. Identificamos os menores expoentes inteiros não-negativos ω_5 e ω_6 para os quais estimativas da norma 2 das matrizes

$$0.9572^{\omega_5} A_{ss} [{}^5(A_s)_*]^{-1} \quad \text{e} \quad 0.9572^{\omega_6} A_{ss} [{}^6(A_s)_*]^{-1}$$

— calculadas com o comando `normest` do Matlab — são menores do que 1. Fazemos $\omega = \max\{\omega_5, \omega_6\}$ e substituímos A pela soma $A_s + 0.9572^\omega A_{ss}$. Como só modificamos a parte anti-simétrica da matriz A inicial, asseguramos que a matriz A assim atualizada tem parte simétrica definida positiva no $\ker(B)$. Além disso, garantimos que a matriz E construída a partir da nova matriz A não tem autovalor -1 , porque $\|I - P\|_2 = 1$ e então

$$\rho(E) \leq \|E\|_2 \leq \|0.9572^\omega A_{ss}(A_s)_*^{-1}\|_2 < 1.$$

Não há nenhum motivo especial para a base da exponencial 0.9572^ω , tendo sido esta escolha arbitrária.

As entradas da parte anti-simétrica $0.9572^\omega A_{ss}$ do bloco (1, 1) do novo problema de ponto-de-sela generalizado possivelmente não são muito menores do que as entradas da parte anti-simétrica da matriz A original, porque 0.9572^ω decresce exponencialmente em função de ω , mas não tão rapidamente quanto poderia. Isso usualmente evita que a parte simétrica de A domine muito fortemente sua parte anti-simétrica. Resulta também em uma matriz E de raio espectral próximo de 1 e força uma convergência mais lenta da série de Neumann de $I + E$. Como uma consequência desta última observação, mais termos

Tabela 1 – Variação dos erros relativos associados às soluções calculadas pelo Algoritmo 5, em função de δ e i_{max} e fixados $n = 3000$ e $m = 90$. Função base: $\phi(r) = r^2 \log r$.

$\delta \setminus i_{max}$	0	2	4
0	1.69e+04	2.34e+08	4.87e+12
10^4	5.17e+03	1.05e+07	3.24e+10
10^8	7.14e-04	5.87e-08	6.74e-12
10^{16}	1.19e-12	1.19e-12	1.19e-12

Tabela 2 – Variação dos erros relativos associados às soluções calculadas pelo Algoritmo 6, em função de δ e i_{max} e fixados $n = 3000$ e $m = 90$. Função base: $\phi(r) = r^2 \log r$.

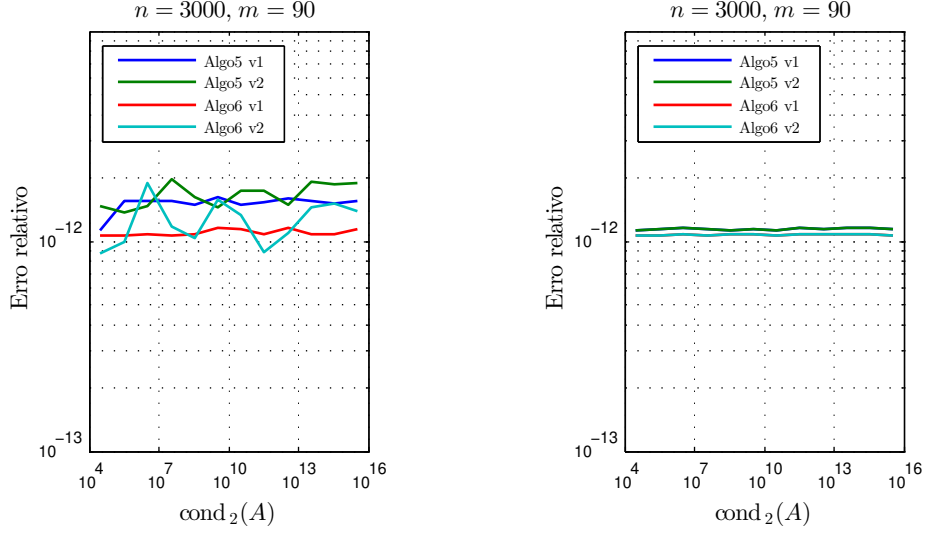
$\delta \setminus i_{max}$	0	2	4
0	1.69e+04	2.34e+08	4.87e+12
10^4	1.46e-05	2.46e-11	1.13e-12
10^8	1.14e-12	1.12e-12	1.12e-12
10^{16}	1.14e-12	1.13e-12	1.13e-12

são necessários ao truncamento de (4.3) para se obter uma boa aproximação da solução de (3.11). As Tabelas 1 e 2 mostram, todavia, que mesmo nesta situação desfavorável, poucos termos precisam ser utilizados para se alcançar a máxima precisão atingível do esquema (dado um valor adequado de δ , obviamente). Podemos então inferir que o método indicado provavelmente é eficiente, mas nos mantemos incomodados com a inclusão de uma hipótese mais restritiva sobre a matriz E .

Mesmo realizando poucos testes com a opção `gmode='other'`, notamos que a quantidade de termos no truncamento de (4.3), exigida para uma aproximação satisfatória de \bar{a} , diminui com o crescimento do valor de δ , podendo até mesmo se anular para um valor suficientemente grande deste último parâmetro (Tabelas 1 e 2). Isto sugere a aproximação $I + E \approx I$, segundo a qual a solução aproximada de (3.11) é tomada como sendo o próprio vetor do lado direito do sistema. A opção `gmode='fast'` foi implementada nos algoritmos para executar este procedimento e convenientemente elimina a necessidade de se exigir qualquer coisa sobre a matriz E . Em particular, que $\rho(E) < 1$. A Figura 6 mostra uma comparação entre as opções `gmode='slow'` e `gmode='fast'`. Vemos nela que o primeiro método retorna erros relativos com maiores variações do que o segundo, possivelmente devido a uma maior incidência de erros de arredondamento no cálculo explícito de E . Por isso tornamos `gmode='fast'` a opção padrão para resolução de problemas de ponto-de-sela generalizados dos Algoritmos 5 e 6 e ignoramos completamente o parâmetro i_{max} deles.

Após a realização de todos os experimentos numéricos desta tese, entendemos porque a aproximação indicada para a matriz $I + E$ é correta, quando fazemos $\delta \rightarrow \infty$. Para justificá-la, vamos considerar uma base $\{v_1, \dots, v_n\}$ de \mathbb{R}^n , composta somente de autovetores da matriz $(A_s)_*$, onde os seus $n - m$ primeiros vetores também constituem uma base do $\ker(B)$, e os m últimos, uma base da $\text{ran}(B^T)$. Uma tal base existe em razão da demonstração dada para a Proposição 2.7. Se e_1, \dots, e_n denotam os vetores canônicos de \mathbb{R}^n , se eles são tais que $e_i = \sum_{k=1}^n \alpha_{ik} v_k$, $i \in \{1, \dots, n\}$, e se $\lambda_1, \dots, \lambda_n$ são os autovalores de

Figura 6 – Variação dos erros relativos associados às soluções calculadas pelos Algoritmos 5 e 6, em função de $\text{cond}_2(A)$. Função base: $\phi(r) = r^2 \log r$. Parâmetros, da esquerda para a direita: $\text{gmode} = \text{'slow'}$ e $\text{gmode} = \text{'fast'}$.



$(A_s)_*$ associados, respectivamente, à v_1, \dots, v_n , então a entrada E_{ij} da matriz $E = (E_{ij})$ utilizada nos algoritmos é

$$\begin{aligned}
 E_{ij} &= \sum_{k=1}^n \sum_{\ell=1}^n (\alpha_{ik} \alpha_{j\ell}) v_k^T E v_j \\
 &= \sum_{k=1}^{n-m} \sum_{\ell=1}^{n-m} (\alpha_{ik} \alpha_{j\ell}) v_k^T A_{ss} [(A_s)_{*,\delta}^{-1} v_\ell] \\
 &= \sum_{k=1}^{n-m} \sum_{\ell=1}^{n-m} \left(\frac{\alpha_{ik} \alpha_{j\ell}}{\lambda_\ell + \delta \gamma_{s,*}} \right) v_k^T A_{ss} v_\ell.
 \end{aligned}$$

Na segunda igualdade, nós usamos que $v_i^T E = 0^T$ e $E v_j = 0$, para todo índice i e j maior do que $n - m$, devido a própria definição de E . Logo podemos concluir que toda entrada da matriz E se anula, quando $\delta \rightarrow \infty$. Em particular, fica certo que E não possui autovalor -1 , que é a razão pela qual dissemos anteriormente que não nos preocupávamos com esta hipótese, nestes experimentos que realizamos. Observamos que esse procedimento, de modo algum, elimina a influência da parte anti-simétrica da matriz A sobre a nossa resolução do problema de ponto-de-sela generalizado, uma vez que claramente conseguimos ver da expressão da inversa da matriz de coeficientes de (3.1), fornecida no Corolário 3.3, que A_{ss} permanece tendo um papel nos cálculos realizados pelos Algoritmos 5 e 6. Por outro lado, constatamos que a matriz E é uma grande fonte de instabilidade para os algoritmos e que a parte deles que envolve o cálculo da série de Neumann (3.18), definitivamente não funciona para o Algoritmo 5, a menos que façamos E se anular para ele, com o auxílio do parâmetro δ . Mas neste caso a série de Neumann nem mesmo precisaria ser considerada. A referida série ainda poderia ser ponderada para o Algoritmo 6, porque da Tabela 2

nós vemos que há uma melhora na precisão das soluções dos problemas de ponto-de-sela calculadas com ela, conforme mais termos são permitidos no truncamento da série (caso $\delta = 10^4$), sem, no entanto, termos de necessariamente forçar a matriz E a se anular. Porém é muito mais simples fazer $\delta \rightarrow \infty$ e deixar de calcular esta série de uma vez por todas.

Finalmente estamos em condições de exibir os resultados dos experimentos numéricos que propomos, apresentando a seguir comparações entre os desempenhos dos Algoritmos 3, 4, 5 e 6 e os desempenhos do BiCGSTAB e do GMRES. Como tolerância para o BiCGSTAB, utilizamos a ordem do menor resíduo relativo associado às soluções computadas com a versão 1 dos Algoritmos 3 e 4 ou dos Algoritmos 5 e 6. Utilizamos este mesmo valor como tolerância para o GMRES, mas então o calculamos com base na versão 2 dos algoritmos, conforme também indicado para o BiCGSTAB. Deixamos que os *solvers* executem até $\lfloor n/20 \rfloor$ iterações e mantemos a configuração padrão do GMRES de nunca reinicializar. A finalidade para um número máximo de iterações grande é aumentar a chance dos *solvers* de atingirem a tolerância desejada, ao custo do tempo de execução. Achemos esse um comprometimento necessário, porque não estamos interessados em qualquer solução que possa ser encontrada pelo BiCGSTAB e o GMRES, mas sim em soluções precisas, cujos resíduos relativos associados a elas tenham, pelo menos, a mesma ordem de magnitude dos resíduos relativos associados às soluções computadas com nossos algoritmos. Isso em razão dos Algoritmos 3, 4, 5 e 6 priorizarem a precisão das soluções calculadas e não somente a eficiência do cômputo das mesmas. Assim, uma leitura alta para o tempo de execução desses *solvers* simplesmente indica a dificuldade que eles sozinhos têm de alcançar o mesmo nível de precisão dos algoritmos propostos — algo que esperamos que ocorra na realidade, pois do contrário o “pré-condicionamento” elaborado no Capítulo 2 não se justificaria. Notamos, ainda, que os problemas de ponto-de-sela gerados não serão pré-condicionados antes de sua resolução com o BiCGSTAB ou o GMRES, porque pré-condicionadores costumam estar atrelados a problemas específicos e aqui estes problemas simulam problemas de ponto-de-sela totalmente gerais. Isso claramente dificulta a determinação de um pré-condicionador apropriado para essa finalidade. Não descartamos, porém, a possibilidade de que, no tratamento de problemas específicos, os sistemas (2.1) e (3.1), adequadamente pré-condicionados, possam ser melhores resolvidos com o BiCGSTAB e o GMRES do que com os algoritmos propostos. Por outro lado, salientamos que métodos iterativos como o GMRES são muito pouco afetados pela presença de um único autovalor nulo, ao menos quando o iterando inicial para eles é um vetor de zeros ((BENZI; GOLUB; LIESEN, 2005, p. 17)), como justamente ocorre nas implementações desses *solvers* no Matlab. Desta forma, compensamos em parte a falta de pré-condicionamento dos sistemas, favorecendo o BiCGSTAB e o GMRES deste outro modo.

Começamos examinando o comportamento dos Algoritmos 3 e 4 sob o efeito de um mau condicionamento crescente do bloco A , para problemas de ponto-de-sela simétricos determinados a partir de uma função condicionalmente definida positiva $\Phi(x) = \phi(\|x\|_2)$,

dada por $\phi(r) = r^3$ (potência radial). A Figura 7 mostra que os erros relativos associados às soluções computadas com esses algoritmos são até 11 ordens de magnitude menores do que os erros relativos associados às soluções obtidas com o BiCGSTAB e o GMRES. Todos os erros relativos aumentam ligeiramente com o crescimento dos valores de n e m , mas os erros relativos dos algoritmos se mantêm em níveis bem menores do que os níveis de erros relativos apresentados pelos *solvers*, mesmo para problemas de ponto-de-sela já de ordem aproximada 5000 e razão $m/n = 5\%$. Comportamento similar pode ser identificado na Figura 8 com relação aos resíduos relativos dos Algoritmos 3 e 4, embora as performances do BiCGSTAB e do GMRES melhorem com incrementos de n neste caso. Isto sugere que os *solvers* retornariam resíduos relativos de mesma ordem que os nossos para problemas de ordem superior a 15000, mais ou menos. Para problemas pequenos, de ordem de cerca de 1000, os *solvers* são centésimos de segundo mais rápidos do que os Algoritmos 3 e 4, conforme ilustra a Figura 9. No entanto, não têm a mesma precisão destes últimos. Para problemas maiores, eles esgotam o orçamento computacional que possuem ao tentar alcançar a mesma precisão dos Algoritmos 3 e 4 e não conseguem, gastando, por isso, um tempo bem maior de execução do que os referidos algoritmos. Na Figura 10, observamos que o “pré-condicionamento” elaborado no Capítulo 2 é efetivo, porque o número de condição da matriz A_* , na norma 2, é várias ordens de magnitude menor do que $\text{cond}_2(A)$. Algumas vezes este valor coincide com $\text{cond}_2(BB^T)$, como predito teoricamente na Proposição 2.10.

O comportamento dos Algoritmos 3 e 4 sob a ação de um mau condicionamento crescente da matriz B já não é tão excepcional. De fato, como ilustra a Figura 11, os erros relativos associados às soluções calculadas com esses algoritmos crescem aproximadamente linearmente, acompanhando o próprio crescimento de $\text{cond}_2(B)$. Inicialmente há um grande *gap* entre eles e os erros relativos associados às soluções do BiCGSTAB e do GMRES, mas os primeiros já passam a ser equivalentes aos dois últimos *solvers* — praticamente em todos os casos observados — quando $\text{cond}_2(B) \approx 10^8$, ou seja, quando B ainda é uma matriz apenas moderadamente mau-condicionada. A partir desse valor, os erros relativos dos nossos algoritmos já são bastante ruins. Estas observações não nos surpreendem, porém, tendo em vista novamente a Proposição 2.10 e o fato bem conhecido de que uma matriz B mal-condicionada impacta o desempenho de métodos de espaço nulo. A Figura 12 mostra que a situação não é muito diferente para os resíduos relativos associados às soluções dos Algoritmos 3 e 4, com o agravante das ordens de magnitude deles rapidamente serem aproximadas pelas ordens de magnitude dos resíduos relativos do BiCGSTAB e do GMRES com incrementos em n . Como vemos na Figura 13, os tempos de execução dos algoritmos são praticamente sempre melhores do que os tempos de execução dos *solvers*, mas isso é irrelevante diante dos outros resultados. Concordando com a teoria desenvolvida, a Figura 14 ressalta a importância de se utilizar a fatoração QR reduzida de B^T quando a matriz B é mal-condicionada, porque somente com ela é que a construção de A_* resultou em uma matriz de número de condição, na norma 2, menor do que $\text{cond}_2(A)$.

A dificuldade causada por uma matriz B mal-condicionada ao Algoritmo 4 é muito menor do que a discutida acima se $b = 0$, porque neste caso as n primeiras componentes da solução do problema de ponto-de-sela simétrico dependem de B simplesmente por meio do fator ortogonal e perfeitamente bem-condicionado Q da fatoração QR reduzida de B^T . De fato, já que daí $x^* = (I - P)A_*^{-1}a$, sendo

$$P = QQ^T, \quad \gamma_* = \|(I - P)A(I - P)\|_2 \quad \text{e} \quad A_* = (I - P)A(I - P) + \gamma_*P.$$

Além disso, $n \gg m$ e ainda devemos ter que $\text{cond}_2(A_*) < \text{cond}_2(A)$, como até mesmo já vimos ocorrer na Figura 14. Isso torna o Algoritmo 4 especialmente interessante para problemas de otimização com restrições e também problemas de interpolação de dados dispersos (os quais não devem ter o vetor do lado direito modificado, como fizemos aqui), entre outros.

A análise dos Algoritmos 5 e 6, em função de n , m , $\text{cond}_2(A)$ e $\text{cond}_2(B)$, é análoga à análise dos Algoritmos 3 e 4 apresentada acima e permite as mesmas conclusões, inclusive quanto à menor dificuldade que o Algoritmo 6 mostra na resolução de problemas de ponto-de-sela generalizados em que B é uma matriz mal-condicionada e b é o vetor nulo. As Figuras 15 a 22, baseadas em uma função condicionalmente definida positiva $\Phi(x) = \phi(\|x\|_2)$, dada por $\phi(r) = (1 - r)_+^{[(m-1)/2]+1}$ (potência truncada), fomentam esse comentário.

Figura 7 – Variação dos erros relativos associados às soluções calculadas pelos algoritmos, em função de n , m e $\text{cond}_2(A)$. Função base: $\phi(r) = r^3$.

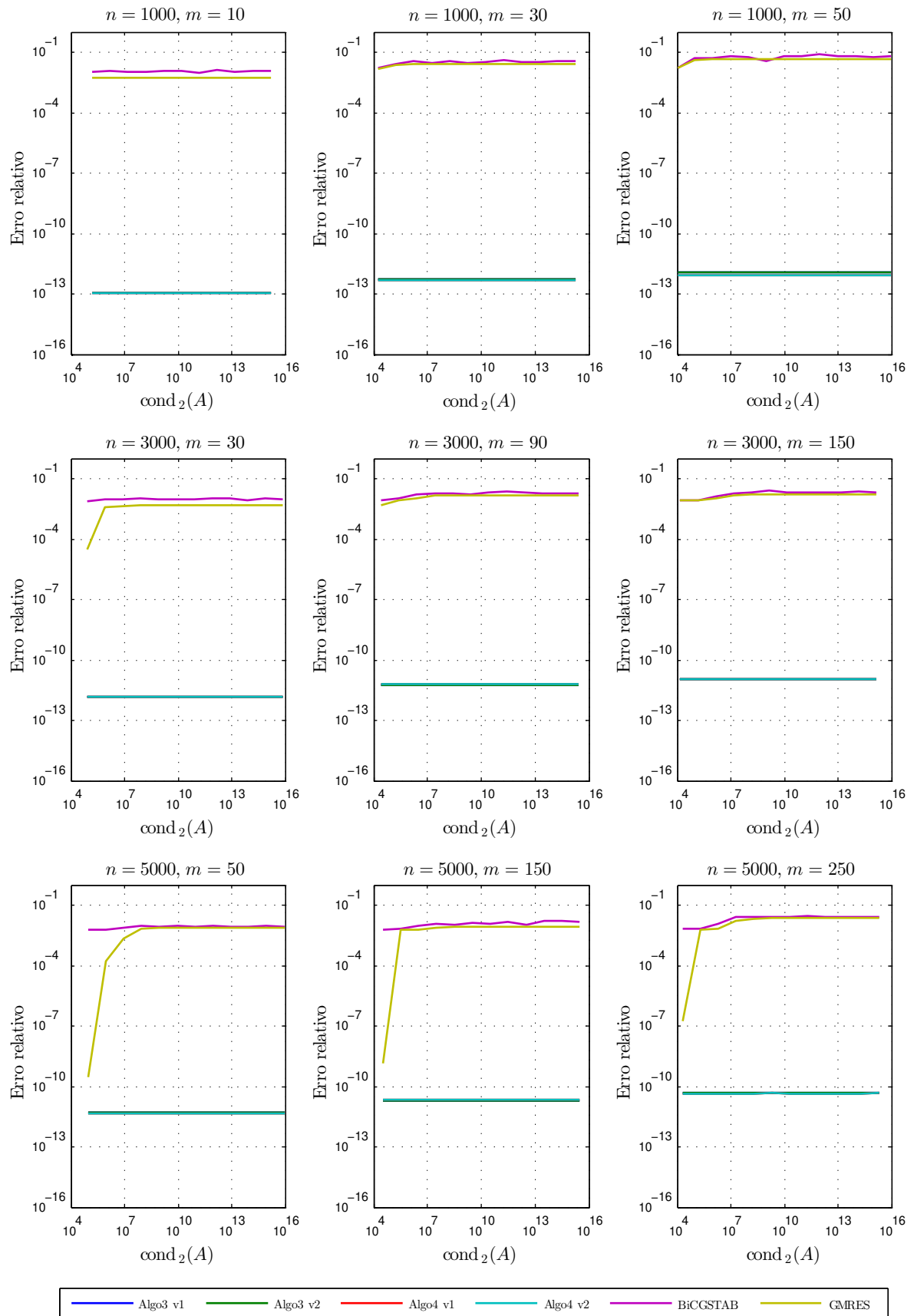


Figura 8 – Variação dos resíduos relativos associados às soluções calculadas pelos algoritmos, em função de n , m e $\text{cond}_2(A)$. Função base: $\phi(r) = r^3$.

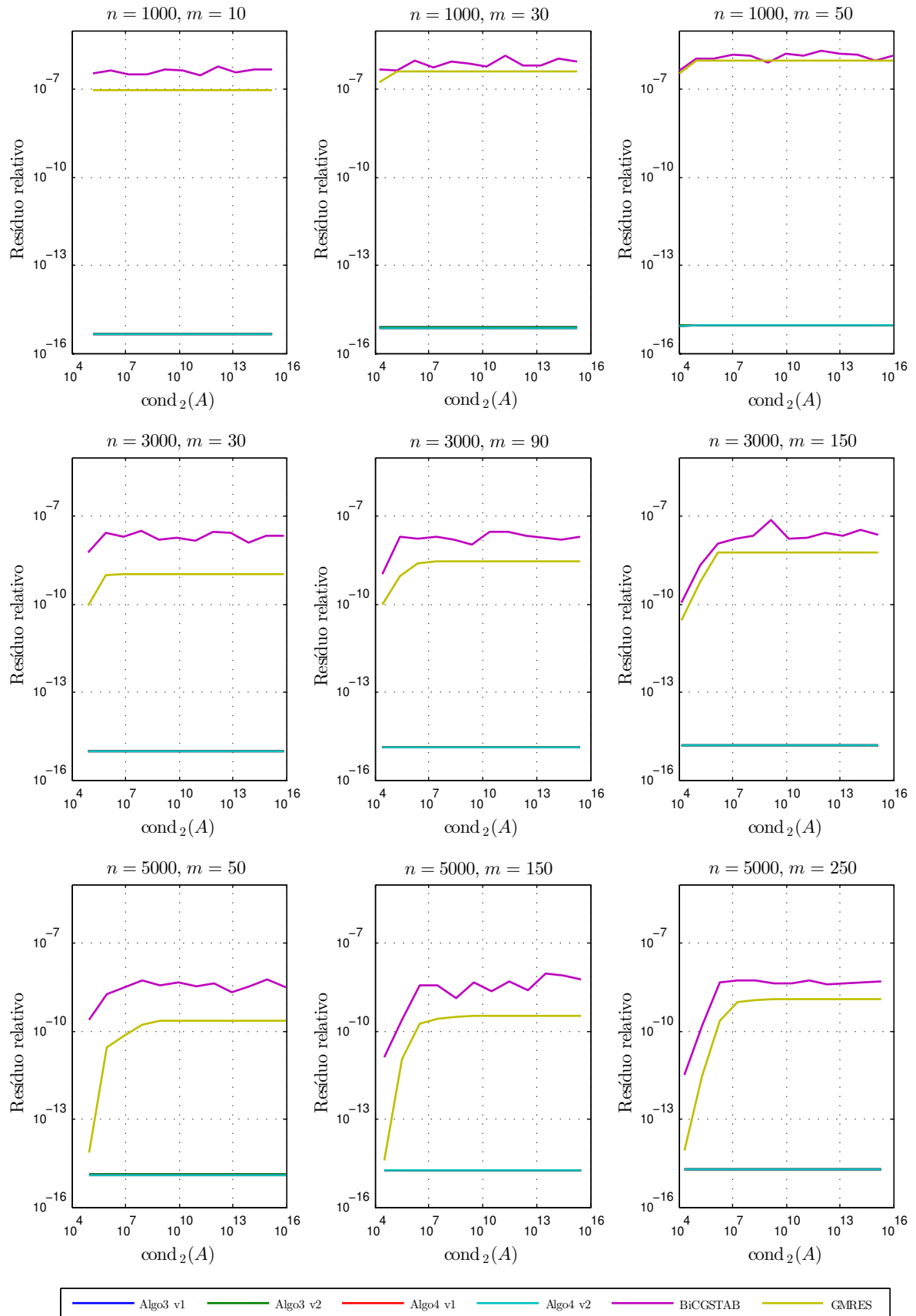


Figura 9 – Variação dos tempos gastos pelos algoritmos para resolução dos problemas gerados, em função de n , m e $\text{cond}_2(A)$. Função base: $\phi(r) = r^3$.

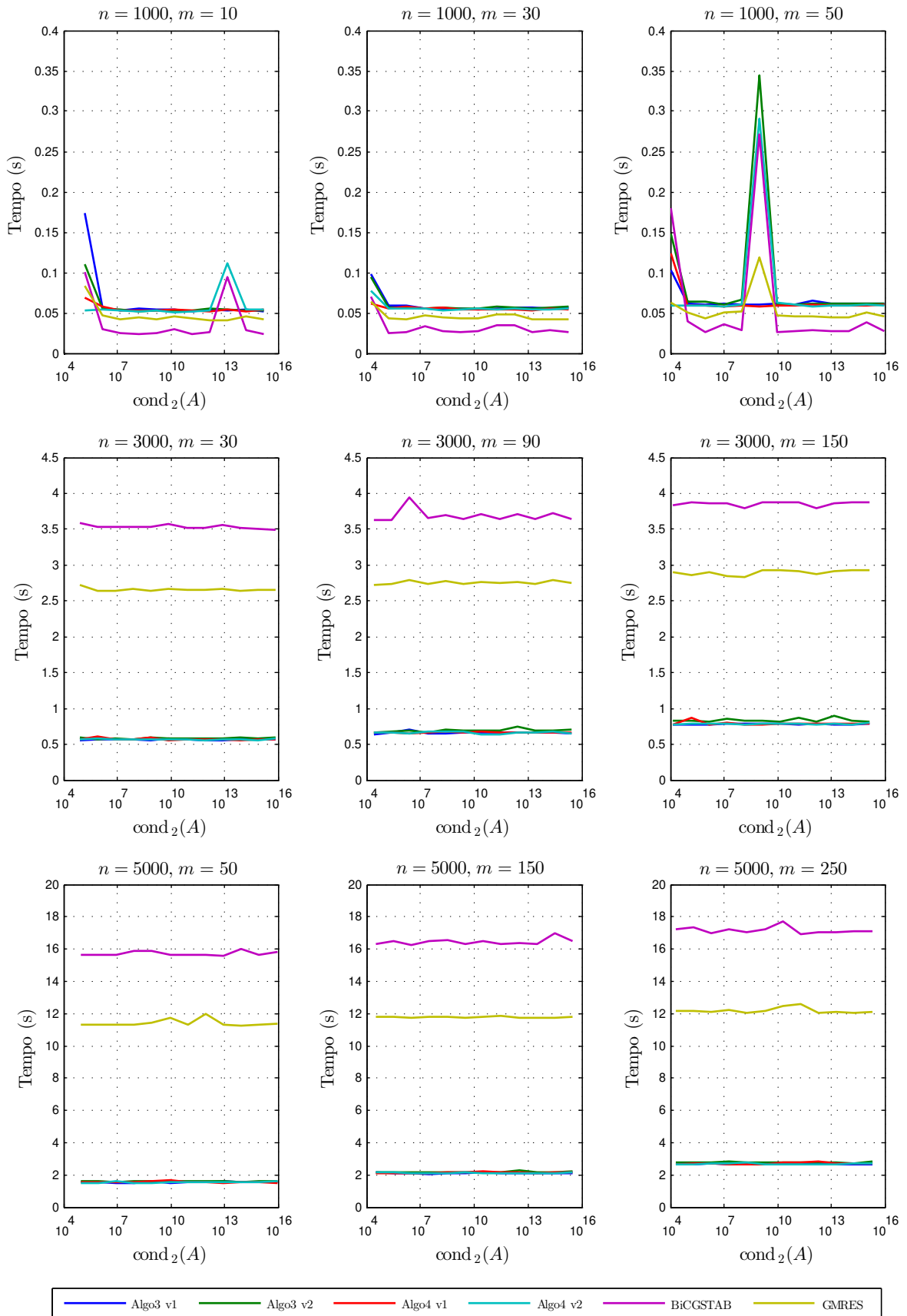


Figura 10 – Variação do número de condição de várias matrizes importantes aos algoritmos, em função de n , m e $\text{cond}_2(A)$. Função base: $\phi(r) = r^3$.

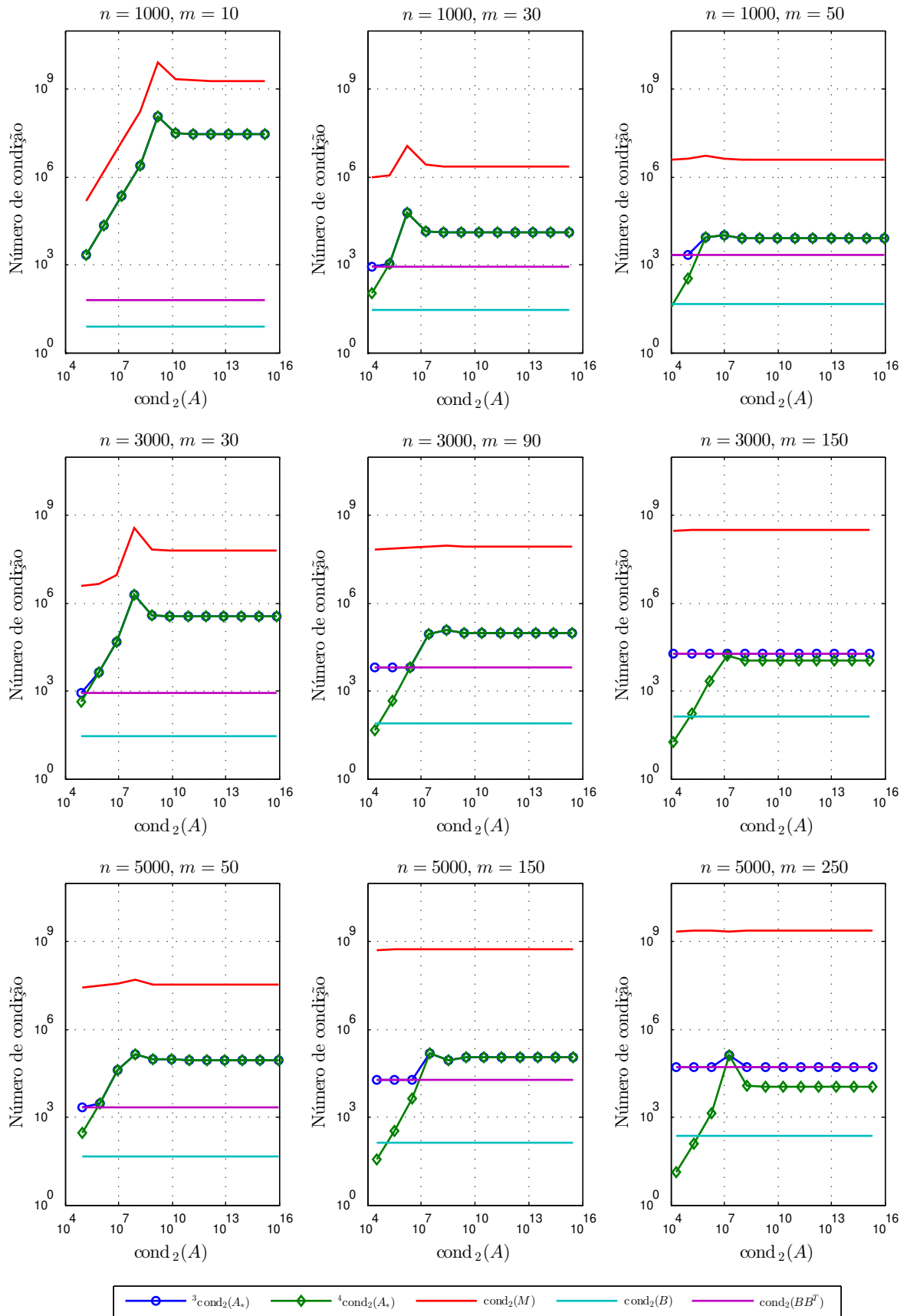


Figura 11 – Variação dos erros relativos associados às soluções calculadas pelos algoritmos, em função de n , m e $\text{cond}_2(B)$. Função base: $\phi(r) = r^3$.

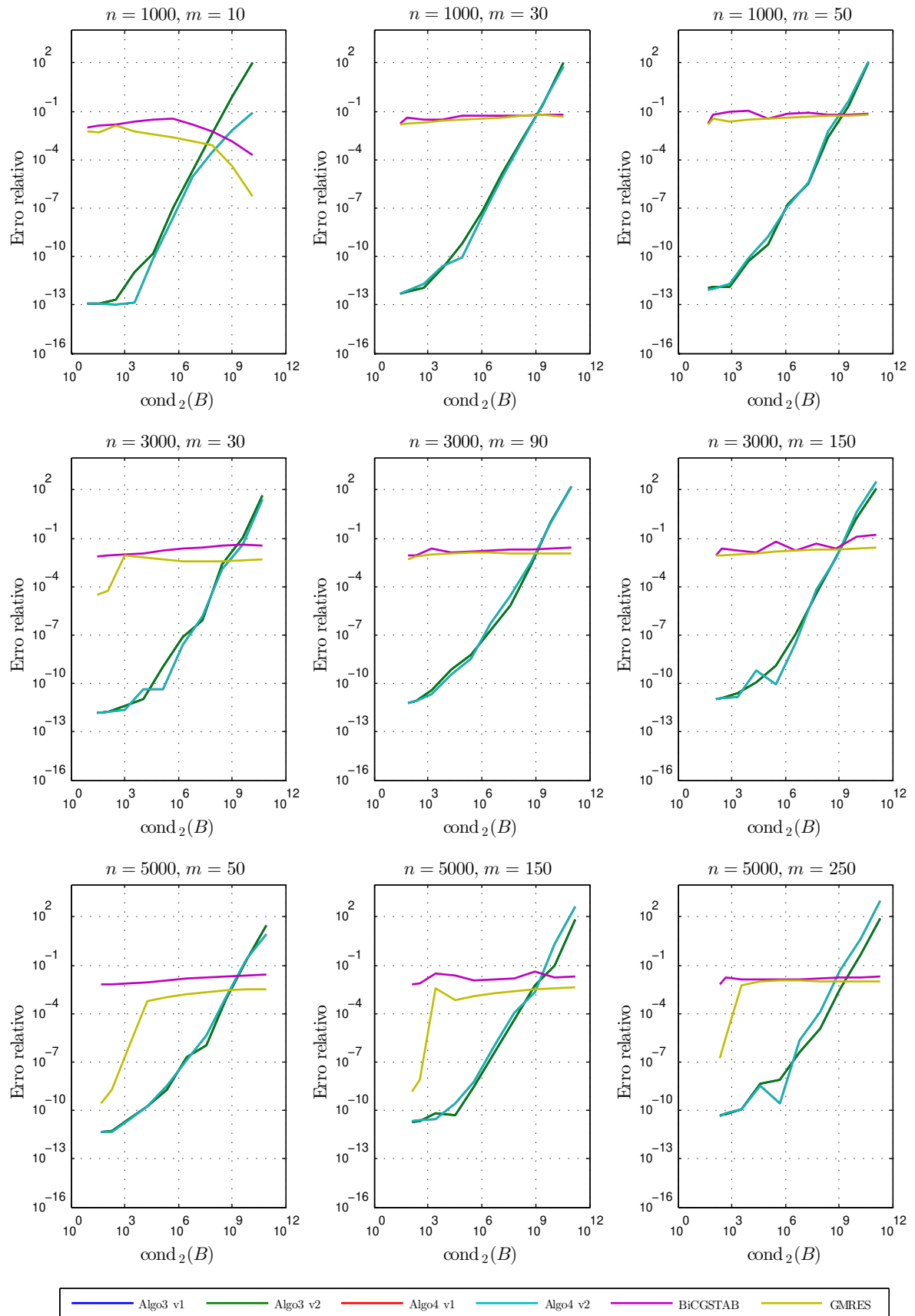


Figura 12 – Variação dos resíduos relativos associados às soluções calculadas pelos algoritmos, em função de n , m e $\text{cond}_2(B)$. Função base: $\phi(r) = r^3$.

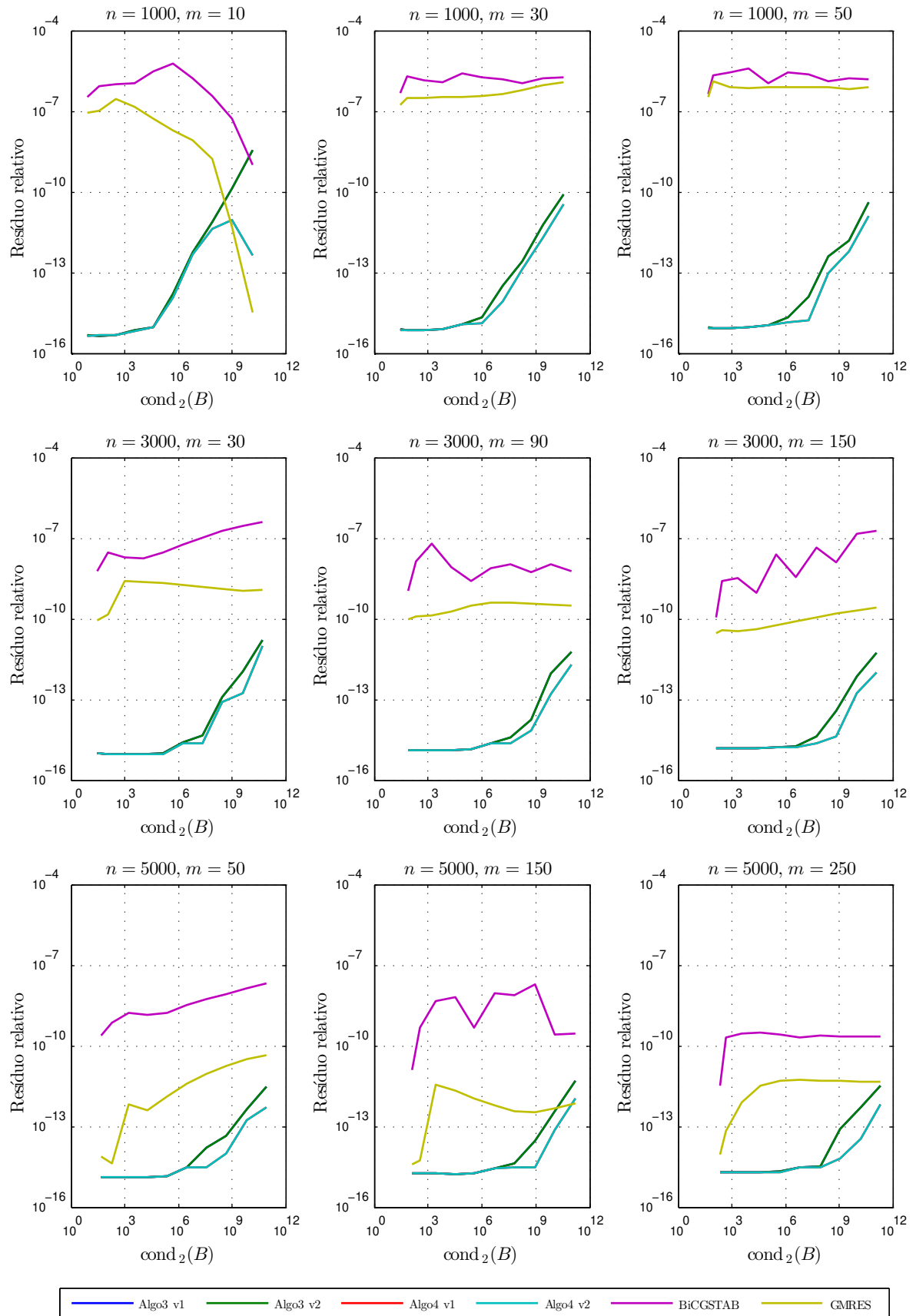


Figura 13 – Variação dos tempos gastos pelos algoritmos para resolução dos problemas gerados, em função de n , m e $\text{cond}_2(B)$. Função base: $\phi(r) = r^3$.

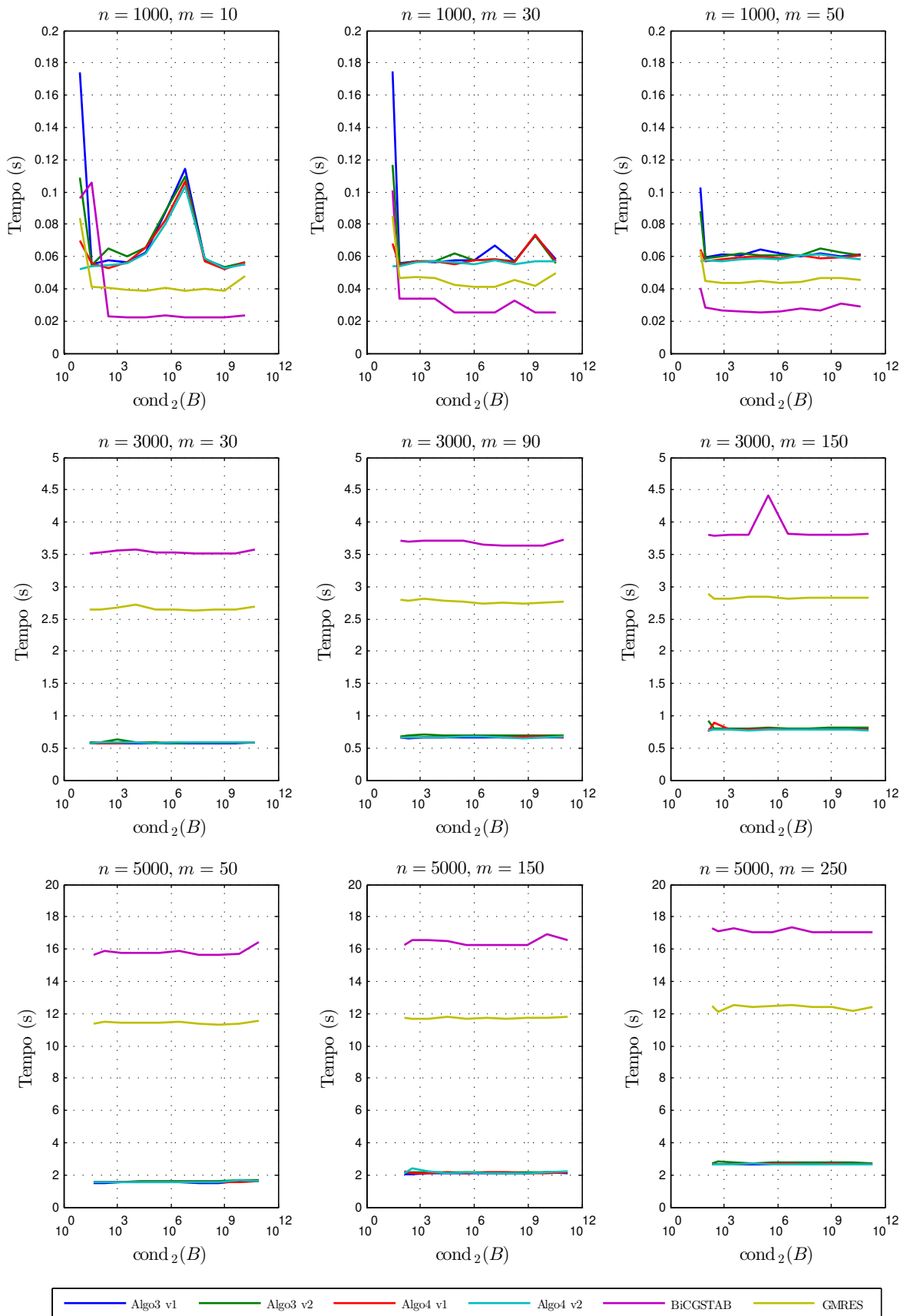


Figura 14 – Variação do número de condição de várias matrizes importantes aos algoritmos, em função de n , m e $\text{cond}_2(B)$. Função base: $\phi(r) = r^3$.

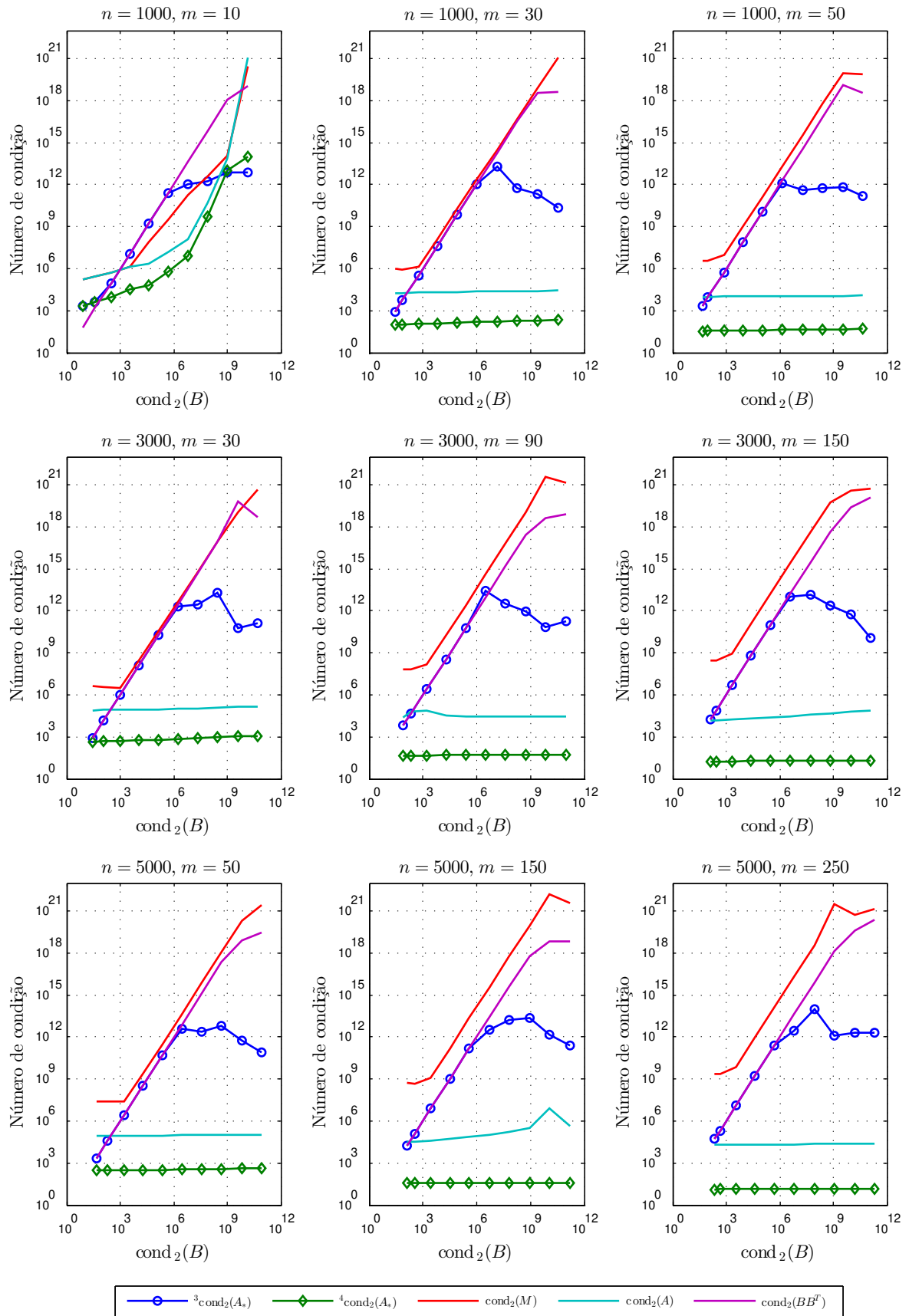


Figura 15 – Variação dos erros relativos associados às soluções calculadas pelos algoritmos, em função de n , m e $\text{cond}_2(A)$. Função base: $\phi(r) = (1 - r)_+^{[(m-1)/2]+1}$.

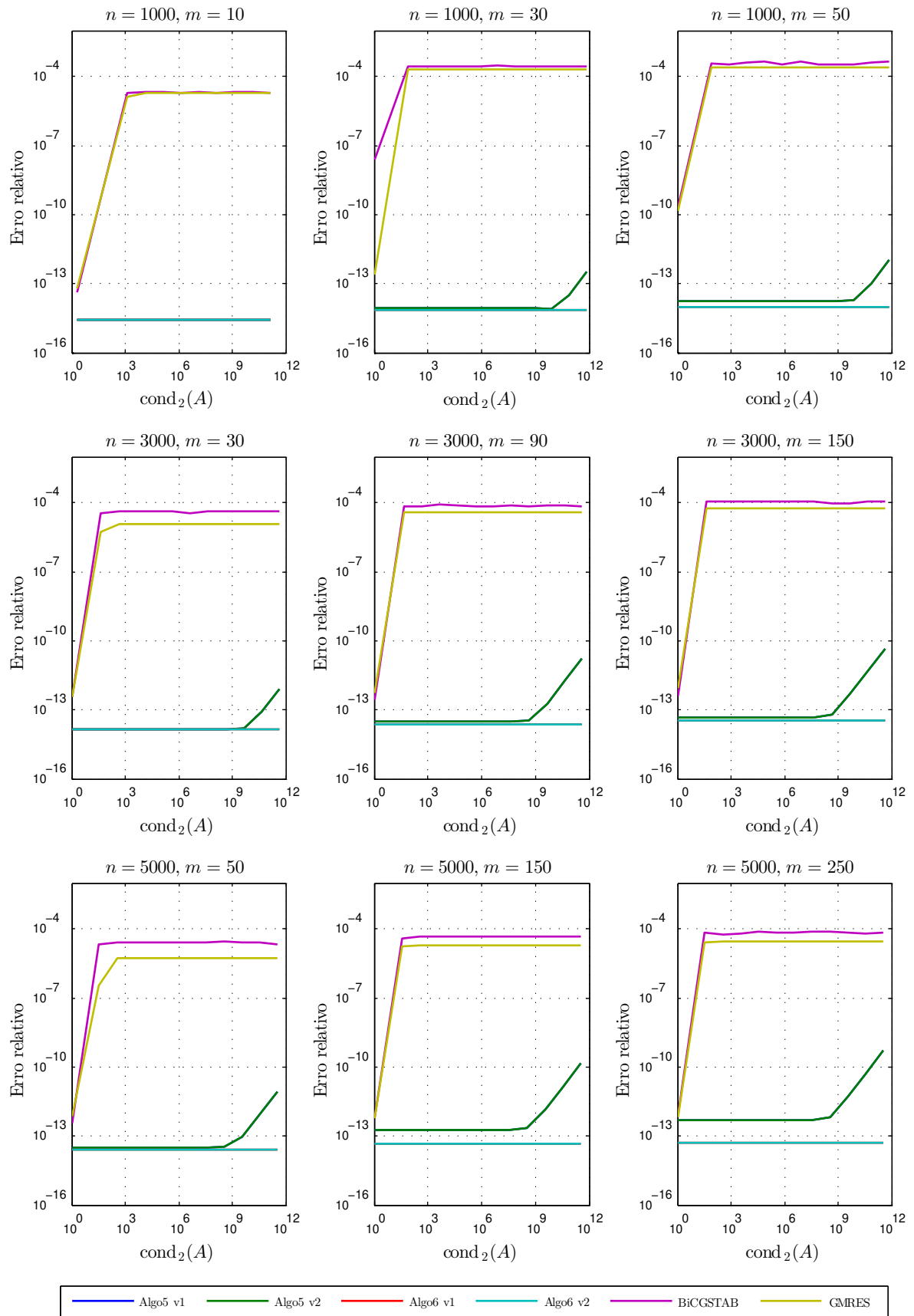


Figura 16 – Variação dos resíduos relativos associados às soluções calculadas pelos algoritmos, em função de n , m e $\text{cond}_2(A)$. Função base: $\phi(r) = (1 - r)_+^{\lfloor (m-1)/2 \rfloor + 1}$.

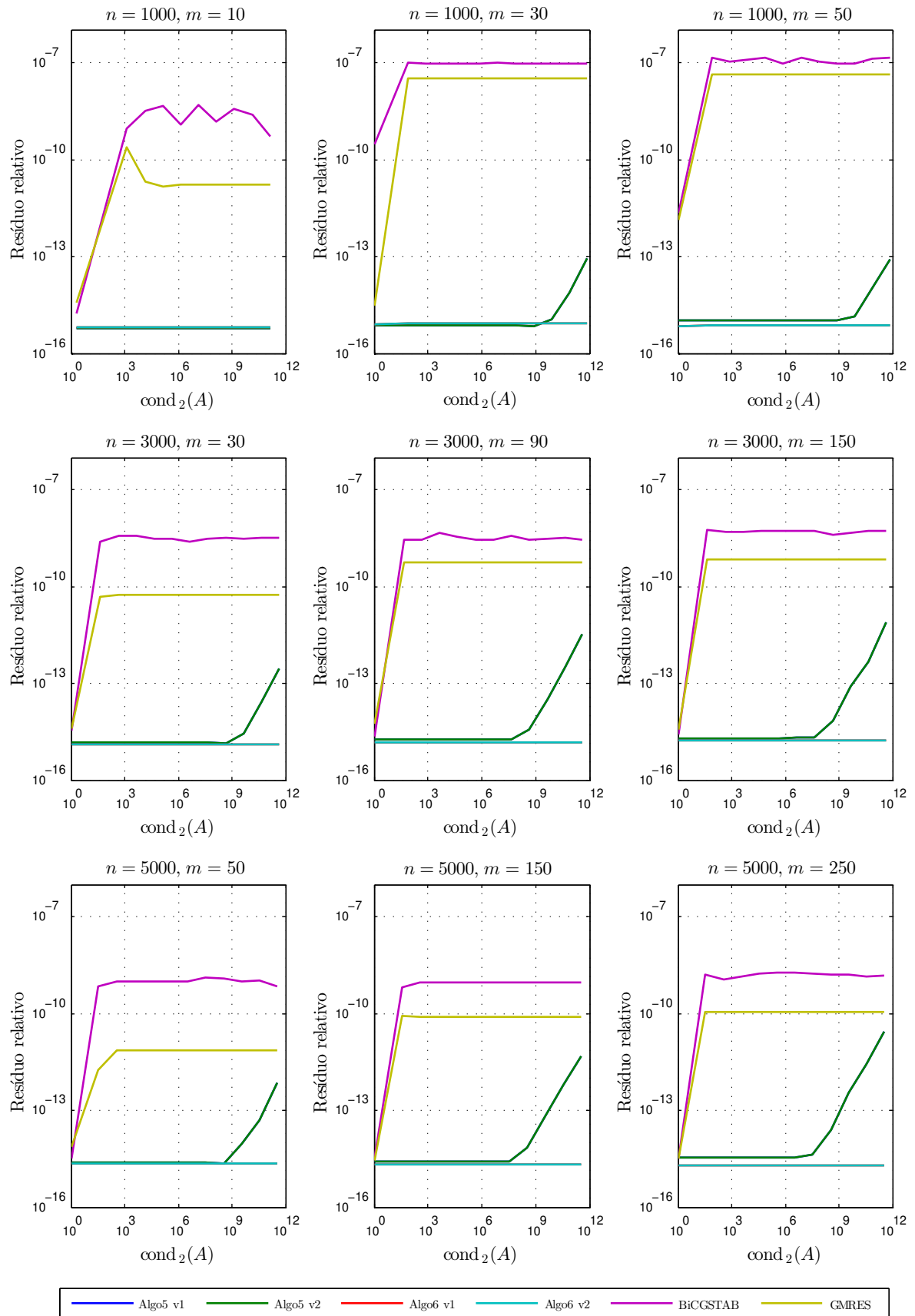


Figura 17 – Variação dos tempos gastos pelos algoritmos para resolução dos problemas gerados, em função de n , m e $\text{cond}_2(A)$. Função base: $\phi(r) = (1 - r)_+^{[(m-1)/2]+1}$.

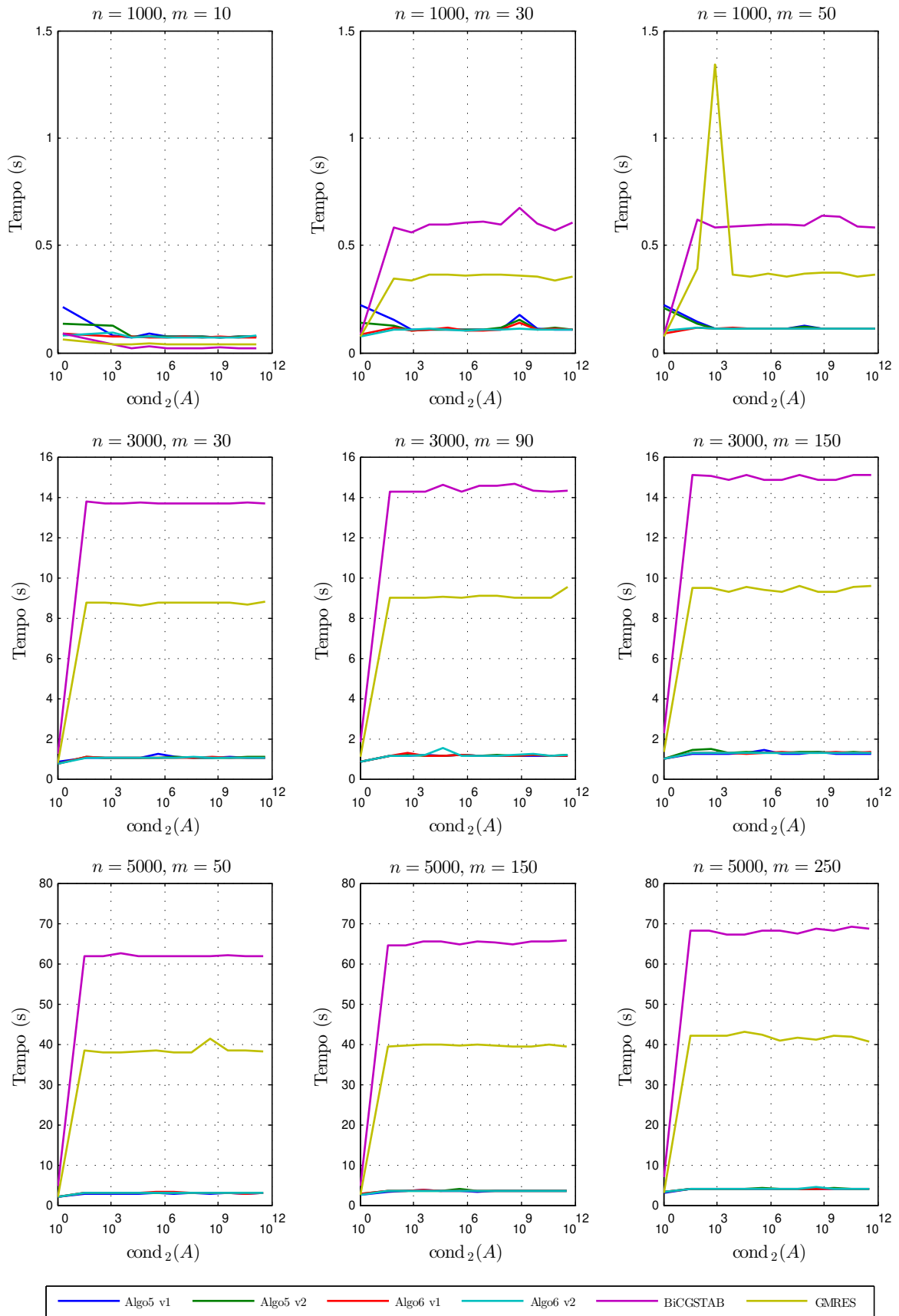


Figura 18 – Variação do número de condição de várias matrizes importantes aos algoritmos, em função de n , m e $\text{cond}_2(A)$. Função base: $\phi(r) = (1 - r)_+^{[(m-1)/2]+1}$.

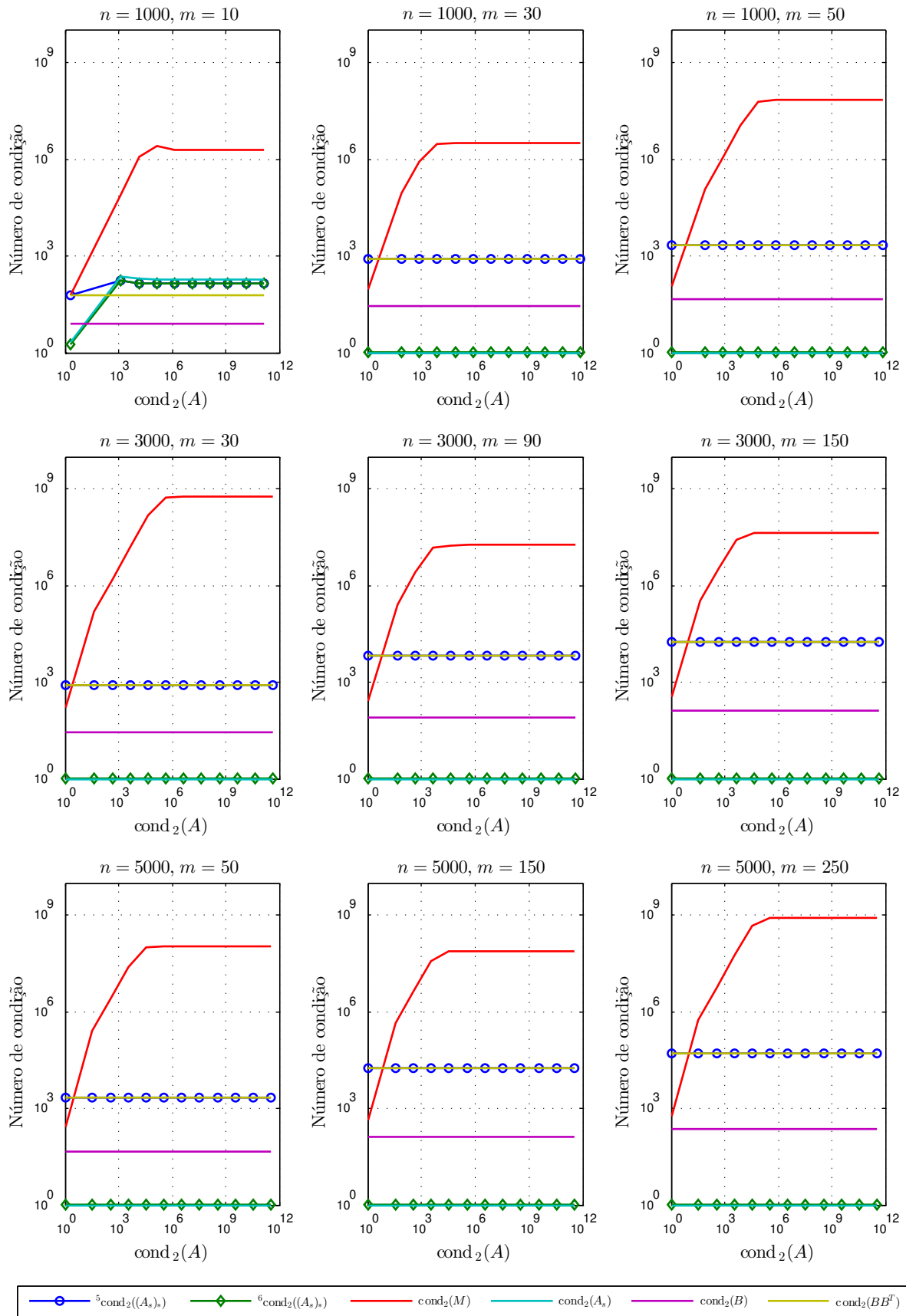


Figura 19 – Variação dos erros relativos associados às soluções calculadas pelos algoritmos, em função de n , m e $\text{cond}_2(B)$. Função base: $\phi(r) = (1 - r)_{+}^{[(m-1)/2]+1}$.

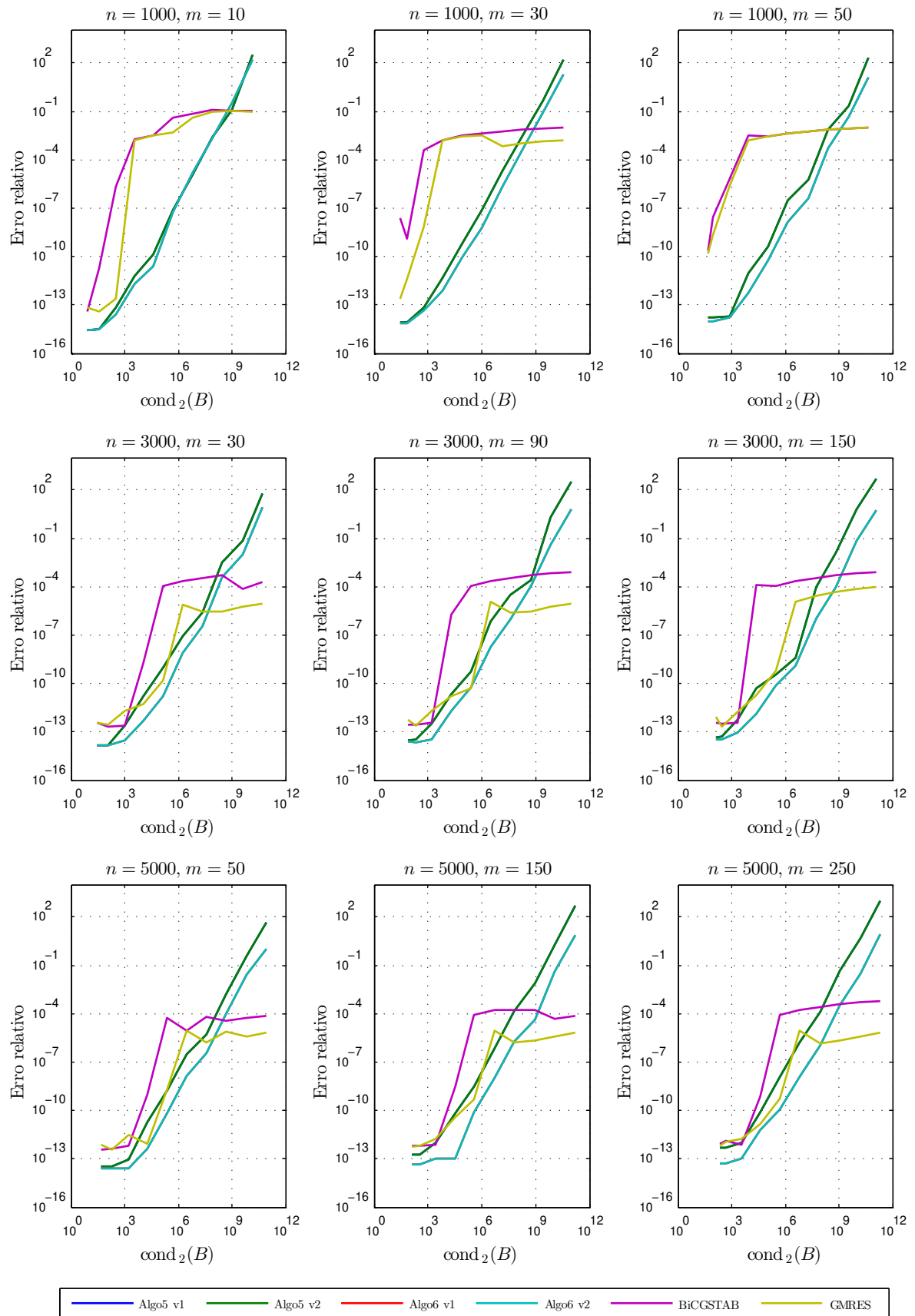


Figura 20 – Variação dos resíduos relativos associados às soluções calculadas pelos algoritmos, em função de n , m e $\text{cond}_2(B)$. Função base: $\phi(r) = (1 - r)_{+}^{[(m-1)/2]+1}$.

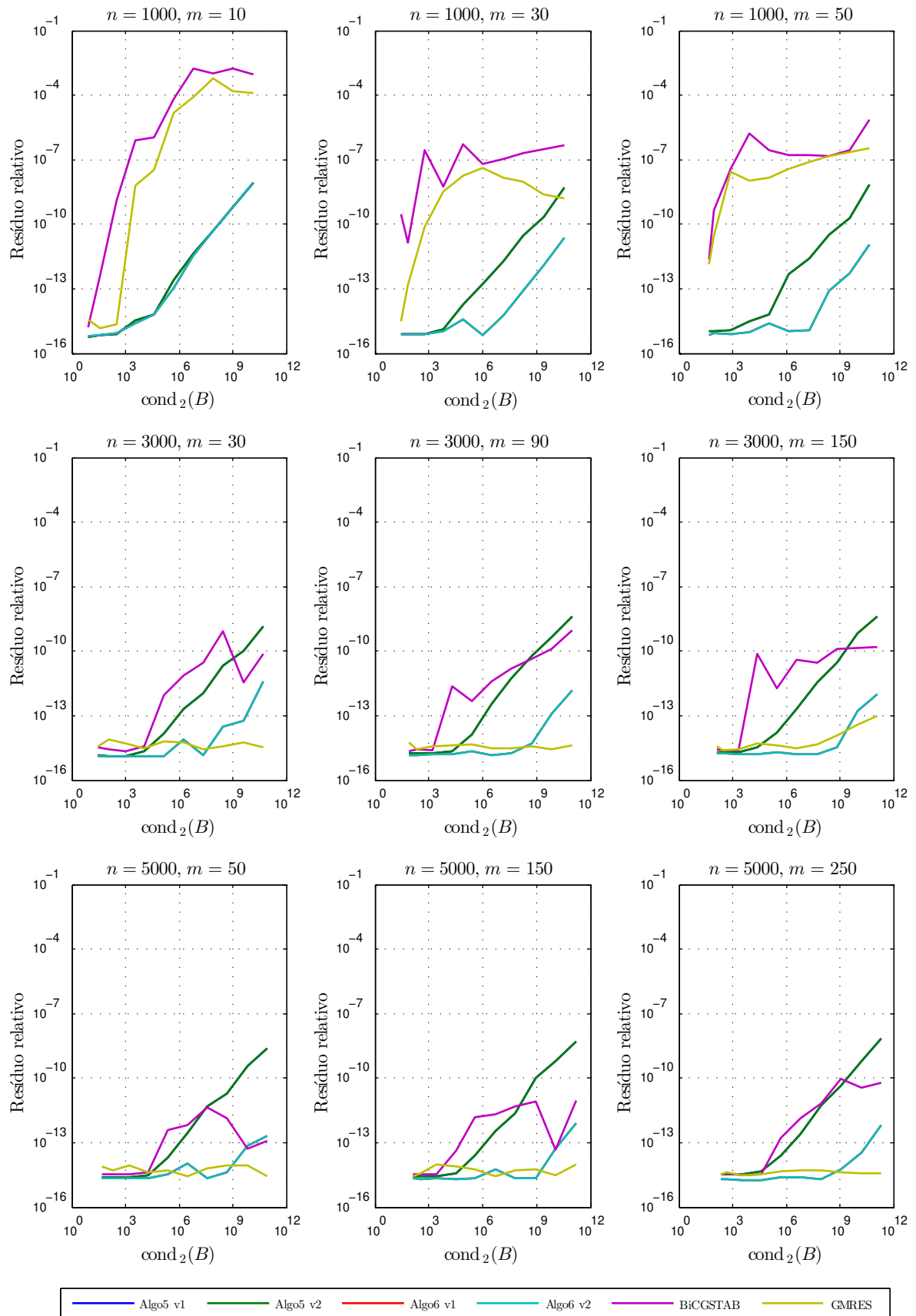


Figura 21 – Variação dos tempos gastos pelos algoritmos para resolução dos problemas gerados, em função de n , m e $\text{cond}_2(B)$. Função base: $\phi(r) = (1-r)_+^{[(m-1)/2]+1}$.

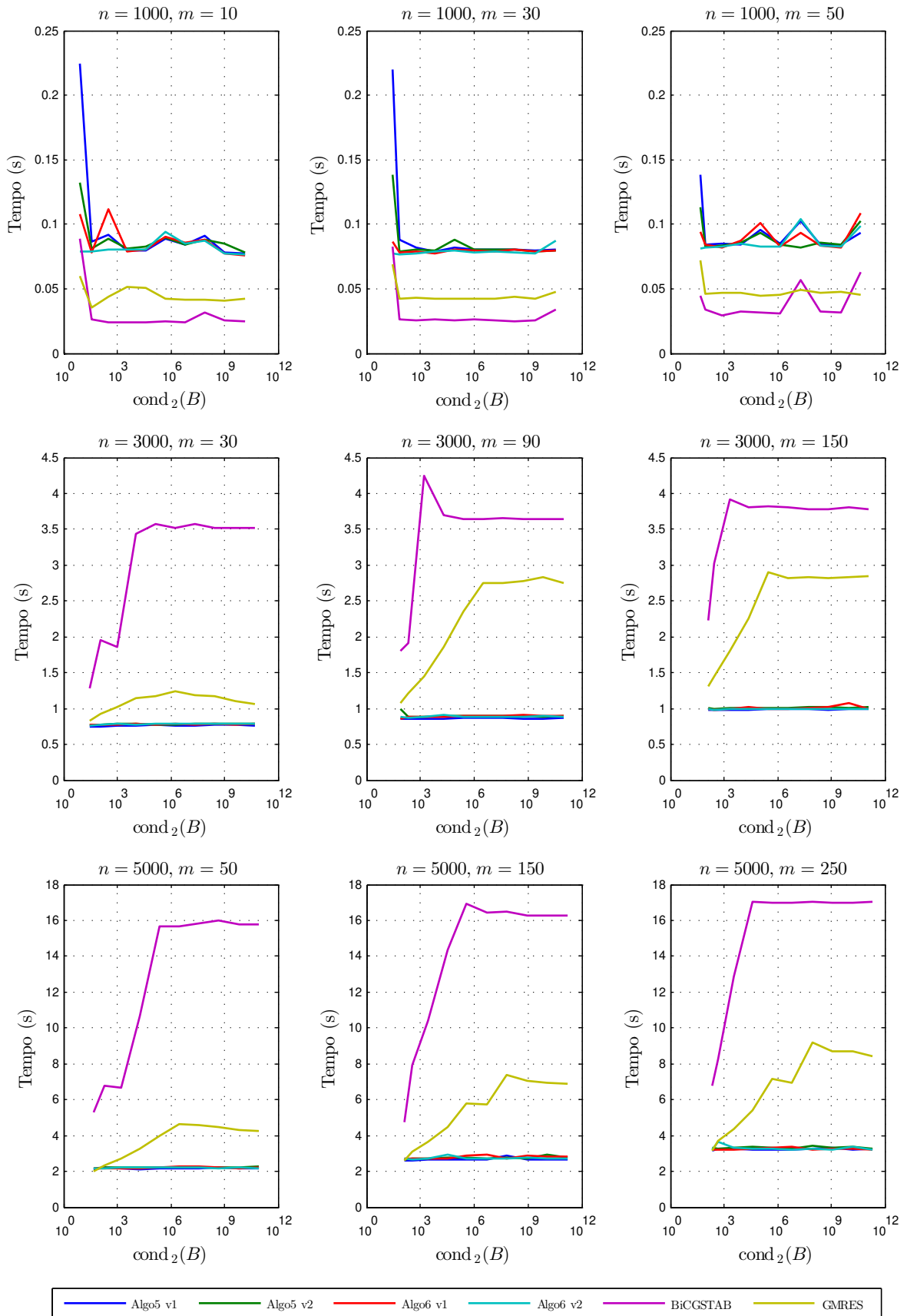
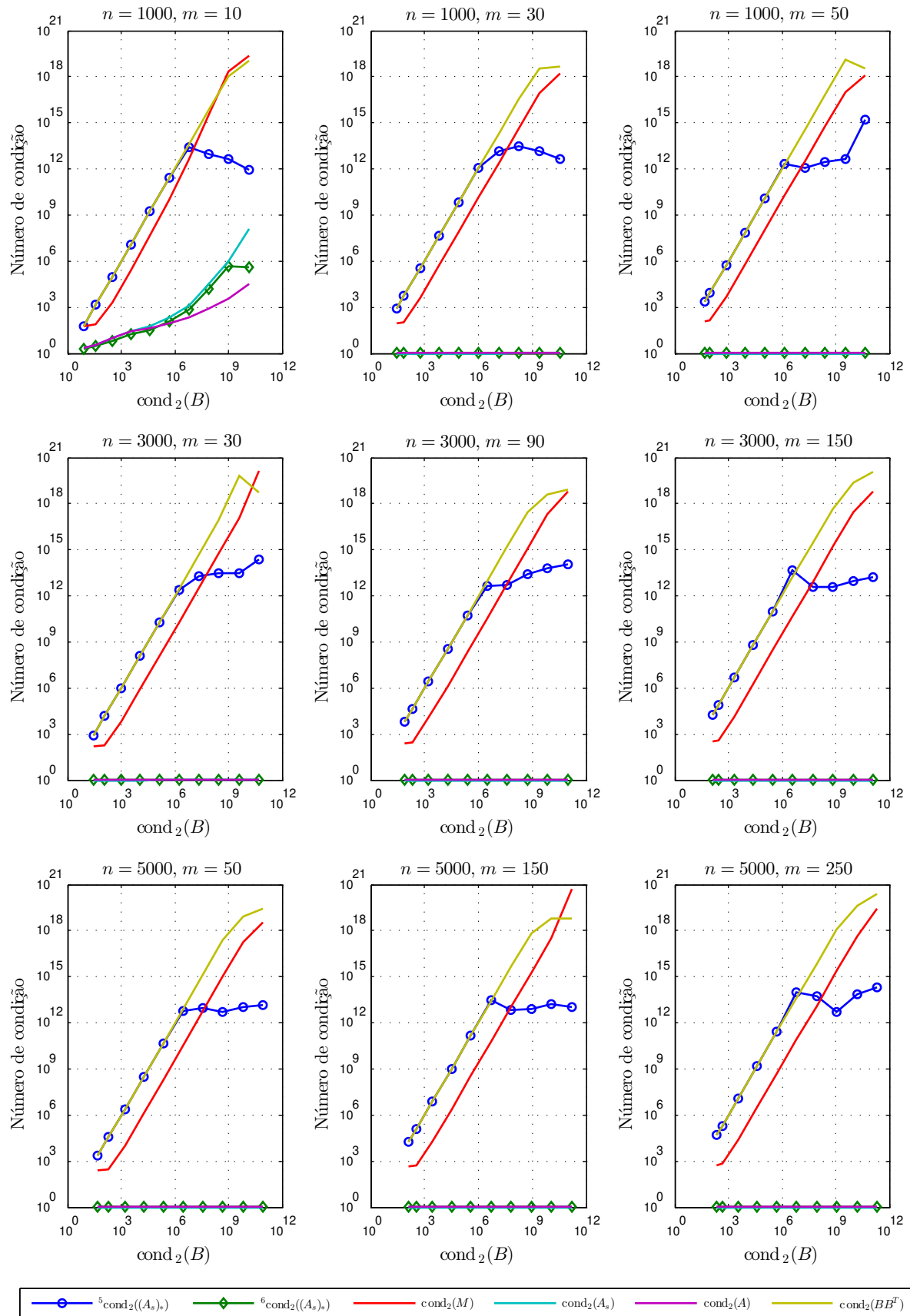


Figura 22 – Variação do número de condição de várias matrizes importantes aos algoritmos, em função de n , m e $\text{cond}_2(B)$. Função base: $\phi(r) = (1 - r)_+^{[(m-1)/2]+1}$.



5 Considerações finais

Muitas aplicações reais levam a problemas de ponto-de-sela. O conhecimento de suas soluções exatas — expressas com considerações numéricas em mente — é então relevante, uma vez que ele pode levar ao desenvolvimento de métodos eficientes de resolução de tais problemas. Sob condições modestas, nós resolvemos analiticamente nesta tese uma ampla classe de problemas de ponto-de-sela, que incluem tanto problemas simétricos, quanto não-simétricos. Cuidado foi tomado durante a determinação da solução de ambos os problemas para melhorar a sensibilidade dos sistemas lineares que os compõem. Isto nos guiou a uma variação robusta e precisa do método de espaço nulo, que é livre de bases. Fornecemos vários algoritmos neste trabalho para calcular as soluções mencionadas e ainda identificamos as condições mais fracas possíveis para a resolução dos problemas de ponto-de-sela considerados aqui. Um *framework* de otimização também foi especialmente concebido para fazer uso desses nossos métodos.

Há um único fenômeno numérico em todo este trabalho, que ainda não conseguimos entender completamente. E ele diz respeito ao comportamento dos nossos algoritmos, quando fazemos o parâmetro δ deles, tender a infinito. Felizmente trata-se de um fenômeno positivo, que aumenta significativamente a precisão das soluções calculadas com os algoritmos. Assim não enxergamos razões para que o entendimento parcial deste único fenômeno desabone de alguma maneira nosso trabalho. Em primeiro lugar, porque autores consagrados nesta área também já se viram em uma situação parecida (([ROZLOŽNÍK; SIMONCINI, 2002](#))). Inclusive por razões muito similares às nossas, como já explicamos durante a análise dos experimentos numéricos realizados. Em segundo lugar, porque o foco do trabalho foi em reduzir o mau-condicionamento de determinadas matrizes, que é uma propriedade delas, e não em tratar especificamente de fenômenos adversos provenientes da representação de números em aritmética de precisão finita. Esta redução de fato ocorreu, como demonstramos analiticamente e empiricamente ao longo deste trabalho. Depois, porque os métodos que propomos funcionam muito bem ao fazermos $\delta \rightarrow \infty$ e toda a teoria que desenvolvemos aqui prediz esse bom funcionamento, inclusive valendo-se de trabalhos de outros autores consagrados (([GOLUB; GREIF, 2003](#)) e ([GOLUB; GREIF; VARAH, 2006](#))). Por fim, porque ainda existe a possibilidade deste fenômeno ser, em parte, externo aos nossos algoritmos e estar relacionado a um comportamento dos *solvers* BiCGSTAB e GMRES que empregamos para a resolução dos sistemas lineares internos a eles. De fato, pois o parâmetro δ dos algoritmos afeta diretamente a performance destes *solvers* por intermédio do número de condição e do espectro das matrizes $A_{*,\delta}$ e $(A_s)_{*,\delta}$.

Os métodos propostos até este momento são todos diretos. Assim existe naturalmente uma limitação a aplicabilidade deles a problemas muito grandes. Seja devido

ao acúmulo de erros de arredondamento, que ocorre mais rapidamente com eles, ou à maior necessidade de recursos computacionais para a execução dos mesmos, como memória. Em particular, nós identificamos para os nossos algoritmos que eles são competitivos para problemas de ordem até mais ou menos 15000. E como em todo método de espaço nulo, livre de bases ou não, é importante que a matriz B dos sistemas lineares a serem resolvidos seja bem-condicionada. Em seguida deduzimos um método iterativo de resolução de problemas de ponto-de-sela simétricos, que provavelmente não conta com esta limitação em relação à ordem dos problemas. A dedução deste outro método, em si, é bastante interessante, porque ela demonstra como podemos fazer bom uso de todas as contribuições desta tese. Em especial, das expressões das soluções exatas dos problemas de ponto-de-sela.

Dos quatro algoritmos que propomos aqui, sugerimos a utilização dos Algoritmos 4 e 6, em detrimento aos demais, simplesmente porque eles exibiram uma melhor performance numérica. Em qualquer caso, devemos sempre nos lembrar de tomar um valor positivo grande para o parâmetro δ .

Como perspectivas futuras de trabalho, nós propomos:

- Analisar com maiores cuidados o efeito do parâmetro δ nos algoritmos, procurando justificar com argumentos analíticos a razão pela qual fazer $\delta \rightarrow \infty$ torna os cálculos dos algoritmos pouco susceptíveis a erros de arredondamento, ao mesmo tempo em que não altera fortemente a solução encontrada com eles, a ponto dela não mais corresponder à solução do problema de ponto-de-sela original e não-perturbado com δ . Parte deste fenômeno já está compreendido para a parte não-simétrica dos Algoritmos 5 e 6, uma vez que sabemos o que acontece com a matriz E , fundamental para eles, com essa transformação em δ : ela se anula. Assim, para que o fenômeno seja compreendido para todos os algoritmos, basta que ele seja compreendido para os Algoritmos 3 e 4, devido à relação existente entre todos eles;
- Determinar uma matriz não-singular $\Theta \in \mathbb{R}^{m \times m}$, computacionalmente barata de ser obtida e aplicada, que modifique o posicionamento relativo das linhas de B , com o objetivo de melhorar o número de condição desta matriz. Pois isso permite relaxar um pouco a exigência de que B tem de ser uma matriz bem-condicionada para aplicação dos algoritmos propostos, já que daí os problemas de ponto-de-sela a serem resolvidos são da forma

$$\begin{bmatrix} A & (\Theta B)^T \\ \Theta B & O \end{bmatrix} \begin{bmatrix} x \\ y_\Theta \end{bmatrix} = \begin{bmatrix} a \\ \Theta b \end{bmatrix}, \quad y = \Theta^T y_\Theta,$$

e o melhor condicionamento da matriz ΘB implica em um melhor condicionamento do fator R da fatoração QR reduzida de $(\Theta B)^T$. Como estamos assumindo que

$m \ll n$, esta transformação pode naturalmente ser baseada na matriz

$$\begin{bmatrix} \frac{B_1 B_1^T}{\|B_1\|_2 \|B_1\|_2} & \cdots & \frac{B_1 B_m^T}{\|B_1\|_2 \|B_m\|_2} \\ \vdots & \ddots & \vdots \\ \frac{B_m B_1^T}{\|B_m\|_2 \|B_1\|_2} & \cdots & \frac{B_m B_m^T}{\|B_m\|_2 \|B_m\|_2} \end{bmatrix},$$

ou em parte dela, cujas entradas são exatamente os cossenos dos ângulos entre as linhas B_1, \dots, B_m da matriz B ;

- Supor que as matrizes A e B têm, elas mesmas, estruturas especiais, como A ser diagonal ou B ser uma matriz banda, e daí tentar especializar os cálculos dos nossos métodos para estas situações. No advento de ainda podermos garantir que a matriz A_* é esparsa sob estas suposições, estudar a que conclusões poderíamos chegar se os espaços $\ker(B)$ e $\text{ran}(B^T)$ fossem invariantes pela transformação linear induzida pelo fator de Cholesky de A_* (que agora pode ser calculado). Dizemos isso porque desconfiamos que estas propriedades ocorram;
- Com base na expressão explícita encontrada para a inversa da matriz de coeficientes do problema de ponto-de-sela generalizado, identificar um pré-condicionador apropriado para a resolução das equações linearizadas de Navier-Stokes. Ou seja: proceder no caso não-simétrico mais ou menos da mesma maneira como [Estrin e Greif \(2015\)](#) procederam no caso simétrico para encontrar um pré-condicionador adequado à resolução numérica das equações de Maxwell harmônicas no tempo;
- Investigar o desempenho numérico do método iterativo (2.5), implicitamente encontrado para a resolução de problemas de ponto-de-sela simétricos, resolvendo cada uma de suas iterações com o método de Arrow-Hurwicz (([BENZI; GOLUB; LIESEN, 2005](#), p. 46)). Sem um pré-condicionamento adequado, este é um método iterativo de convergência usualmente lenta. Mas as suas iterações,

$$\begin{cases} x^{((\ell+1))} = x^{((\ell))} + \alpha(a_* - A_* x^{((\ell))} - B^T y^{((\ell))}) \\ y^{((\ell+1))} = y^{((\ell))} + \omega(Bx^{((\ell+1))} - b) \end{cases}, \quad \ell \in \mathbb{N}_0,$$

onde $x := x^{(k+1)}$, $y := y^{(k+1)}$, $a_* := a_*^{(k)}$, $\alpha > 0$ e $\omega > 0$, são computacionalmente baratas de serem avaliadas, pois envolvem apenas produtos de matrizes por vetores. Conseguimos conceber uma estratégia que possivelmente acelera a convergência dessas iterações mais internas a (2.5), além de simplificá-las, baseando-nos justamente no conhecimento da expressão exata da solução do problema de ponto-de-sela simétrico (2.1). Ela é a seguinte. Vamos substituir $x^{((\ell+1))}$ na expressão de $y^{((\ell+1))}$ para ver que

$$y^{((\ell+1))} = y^{((\ell))} + \omega[Bx^{((\ell))} + \alpha(Ba_* - BA_* x^{((\ell))} - BB^T y^{((\ell))})].$$

Vamos ainda supor que $(x^{((\ell))}, y^{((\ell))}) \rightarrow (x, y)$, quando $\ell \rightarrow \infty$ (o que de fato ocorre, para parâmetros α e ω escolhidos apropriadamente). Nesta situação, temos que

$$Bx^{((\ell))} \rightarrow Bx \quad \text{e} \quad BA_*x^{((\ell))} \rightarrow BA_*x.$$

Mas a expressão de x é conhecida do Teorema 2.8. Utilizando-a, encontramos que

$$Bx = b \quad \text{e} \quad BA_*x = \gamma_* BB^T b,$$

porque $B(I - P) = O$, $BA_*(I - P) = O$ e

$$BA_*B^\dagger = B[(I - P)A(I - P) + \gamma_* B^T B]B^\dagger = \gamma_* BB^T,$$

sendo esta última identidade devida a definição da matriz A_* . Assim podemos fazer

$$y^{((\ell+1))} = y^{((\ell))} + \omega [b + \alpha(Ba_* - \gamma_* BB^T b - BB^T y^{((\ell))})].$$

Mas da definição de a_* também ocorre que

$$Ba_* - \gamma_* BB^T b = B(a + \tilde{W}b - A_{**}x^{(k)}),$$

onde \tilde{W} é como em (2.2) e $x^{(k)}$ é conhecido da iteração externa anterior deste método. Simplificamos a equação acima para encontrar que

$$\begin{aligned} Ba_* - \gamma_* BB^T b &= Ba - BAB^\dagger b - BA(I - P)x^{(k)} \\ &= B\{a - A[(I - P)x^{(k)} + B^\dagger b]\}. \end{aligned}$$

Com isso descobrimos que

$$\begin{aligned} y^{((\ell+1))} &= y^{((\ell))} + \omega [b + \alpha B\{a - A[(I - P)x^{(k)} + B^\dagger b]\} - \alpha BB^T y^{((\ell))}] \\ &= [I - (\alpha\omega)BB^T]y^{((\ell))} + \omega [b + \alpha B\{a - A[(I - P)x^{(k)} + B^\dagger b]\}]. \end{aligned}$$

No limite, nós devemos ter que y é solução do sistema linear

$$(BB^T)y = \frac{1}{\alpha}b + B\{a - A[(I - P)x^{(k)} + B^\dagger b]\}. \quad (5.1)$$

Vemos daí que a determinação de y não depende de elementos externos ao problema de ponto-de-sela original, como γ_* e A_* , o que confirma que ainda há margem neste método para a melhora do número de condição da matriz B (talvez até mesmo com a transformação T_Θ induzida pela matriz Θ de alguns parágrafos atrás). Enxergamos daí, também, que a mesma fatoração QR reduzida da matriz B^T , empregada para se calcular, por exemplo, a matriz A_* , pode ser agora reaproveitada para se calcular o vetor y , a um custo de apenas $\mathcal{O}(m^2)$ operações de ponto flutuante adicionais ao custo de $\mathcal{O}(mn)$ operações necessárias ao cálculo do vetor do lado direito de (5.1). Se o cálculo de y ocorresse iteradamente, como originalmente sugerido pelo método

de Arrow-Hurwicz, teríamos um custo de $\mathcal{O}(mn)$ operações de ponto flutuante por cada iteração, para um número indefinido delas. Como $m \ll n$, a nossa alternativa então já se mostra menos dispendiosa a partir da quinta ou sexta iteração do método desses autores, mais ou menos. Substituímos y assim obtido na expressão de $x^{((\ell+1))}$ para encontrar um esquema iterativo mais simples para a determinação de x :

$$x^{((\ell+1))} = x^{((\ell))} + \alpha[(a_* - B^T y) - A_* x^{((\ell))}], \quad \ell \in \mathbb{N}_0. \quad (5.2)$$

Se $\alpha \ll 1$, a sequência acima claramente termina rapidamente. Como (2.5) converge em exatamente três iterações, acabamos de construir um método iterativo provavelmente muito eficiente para a resolução de problemas de ponto-de-sela simétricos. Este novo método certamente é robusto, porque ele apenas exige dos problemas que a matriz B tenha posto completo, o que sabemos ser uma condição necessária para a existência e unicidade de solução para os mesmos. O método também deve ser preciso, por causa do uso da matriz A_* em suas iterações. Por se tratar de um método iterativo, estamos confiantes de que ele pode ser aplicado competitivamente a problemas de ponto-de-sela de ordem maior do que 15000, ao contrário dos nossos métodos diretos. Utilizar este novo método para se conseguir um método iterativo de resolução de problemas de ponto-de-sela generalizados é imediato da expressão da solução exata deles, fornecida no Teorema 3.2. Porém ainda não estamos certos de que isto resultaria em um método estável, por causa da resolução do sistema linear (3.11), que talvez não admita a aproximação $E \approx O$, pela possível impossibilidade de podermos fazer $\delta \rightarrow \infty$ neste caso;

- Desenvolver um método de Krylov para a resolução de problemas de ponto-de-sela que utilize todas as ideias apresentadas nesta tese, porque o método iterativo logo acima é simplesmente um método estacionário e, da literatura, sabemos que os métodos do primeiro tipo costumam ser mais eficientes do que os do último tipo;
- Por à prova o *framework* proposto no Capítulo 1 para a resolução de um problema geral de otimização com restrições, fixando diferentes estratégias de conjuntos ativos para ele e ainda procurando analisar o seu desempenho a partir de métodos diversificados de atualização das aproximações das hessianas das funções envolvidas no problema.

Referências

- BARLOW, J. L.; NICHOLS, N. K.; PLEMMONS, R. J. Iterative methods for equality-constrained least squares problems. *SIAM J. Sci. Stat. Comput.*, v. 9, n. 5, p. 892–906, 1988.
- BENZI, M. Preconditioning techniques for large linear systems: A survey. *Journal of Computational Physics*, v. 182, n. 2, p. 418–477, 2002.
- BENZI, M.; GOLUB, G. H. A preconditioner for generalized saddle point problems. *SIAM J. Matrix Anal. Appl.*, v. 26, n. 1, p. 20–41, 2004.
- BENZI, M.; GOLUB, G. H.; LIESEN, J. Numerical solution of saddle point problems. *Acta Numerica*, v. 14, p. 1–137, 2005.
- BENZI, M.; OLSHANSKII, M. A. An augmented Lagrangian-based approach to the Oseen problem. *SIAM J. Sci. Comput.*, v. 28, n. 6, p. 2095–2113, 2006.
- BERTSEKAS, D. P. *Constrained Optimization and Lagrange Multiplier Methods*. Belmont, MA: Athena Scientific, 1996.
- _____. *Convex Optimization Theory*. Belmont, MA: Athena Scientific, 2009.
- BIRGIN, E. G.; MARTÍNEZ, J. M. *Practical Augmented Lagrangian Methods for Constrained Optimization*. Filadélfia, PA: SIAM, 2014.
- BOCHEV, P.; LEHOUCQ, R. B. On the finite element solution of the pure Neumann problem. *SIAM Rev.*, v. 47, n. 1, p. 50–66, 2005.
- BURDEN, R. L.; FAIRES, J. D. *Numerical Analysis*. 8. ed. Belmont, CA: Thomson Brooks/Cole, 2005.
- CARR, J. C.; BEATSON, R. K.; CHERRIE, J. B.; MITCHELL, T. J.; FRIGHT, W. R.; MCCALLUM, B. C.; EVANS, T. R. Reconstruction and representation of 3D objects with radial basis functions. In: SIGGRAPH 2001, 28., 2001, Los Angeles. *SIGGRAPH '01 Proceedings of the 28th annual conference on computer graphics and interactive techniques*. Nova Iorque, NY: ACM, 2001. p. 67–76.
- DENNIS, J. J. E.; SCHNABEL, R. B. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Filadélfia, PA: SIAM, 1996.
- ESTRIN, R.; GREIF, C. On nonsingular saddle-point systems with a maximally rank deficient leading block. *SIAM J. Matrix Anal. Appl.*, v. 36, n. 2, p. 367–384, 2015.
- _____. Towards an optimal condition number of certain augmented Lagrangian-type saddle-point matrices. *Numer. Linear Algebra Appl.*, v. 23, p. 693–705, 2016.
- FASSHAUER, G. E. *Meshfree Approximation Methods with MATLAB*. Hackensack, NJ: World Scientific, 2007.
- FLETCHER, R.; JOHNSON, T. *On the stability of null-space methods for KKT systems*. Relatório Técnico NA/167. Dundee, Escócia, UK: University of Dundee, 1995. 22 p.

- FORSGREN, A.; GILL, P. E.; SHINNERL, J. R. Stability of symmetric ill-conditioned systems arising in interior methods for constrained optimization. *SIAM J. Matrix Anal. Appl.*, v. 17, n. 1, p. 187–211, 1996.
- FORTIN, M.; GLOWINSKI, R. *Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems*. Amsterdã, NL: North-Holland, 1983.
- FRANK, P. D.; HEALY, M. J.; MASTRO, R. A. A range-space implementation for large quadratic programs with small active sets. *Journal of Optimization Theory and Applications*, v. 69, n. 1, p. 109–127, 1991.
- GANSTERER, W. N.; SCHNEID, J.; UEBERHUBER, C. W. *Mathematical properties of equilibrium systems*. Relatório Técnico AURORA TR2003-13. Viena, AT: University of Vienna e Vienna University of Technology, 2003. 24 p.
- GLOWINSKI, R. *Numerical Methods for Nonlinear Variational Problems*. Nova Iorque, NY: Springer-Verlag, 1984.
- GLOWINSKI, R.; TALLEC, P. L. *Augmented Lagrangian and Operator-Splitting Methods in Nonlinear Mechanics*. Filadélfia, PA: SIAM, 1989.
- GOLUB, G. H.; GREIF, C. On solving block-structured indefinite linear systems. *SIAM J. Sci. Comput.*, v. 24, n. 6, p. 2076–2092, 2003.
- GOLUB, G. H.; GREIF, C.; VARAH, J. M. An algebraic analysis of a block diagonal preconditioner for saddle point systems. *SIAM J. Matrix Anal. Appl.*, v. 27, n. 3, p. 779–792, 2006.
- GOLUB, G. H.; VAN LOAN, C. F. *Matrix Computations*. 4. ed. Baltimore, MD: The Johns Hopkins University Press, 2013.
- GOLUB, G. H.; WATHEN, A. J. An iteration for indefinite systems and its application to the Navier-Stokes equations. *SIAM J. Sci. Comput.*, v. 19, n. 2, p. 530–539, 1998.
- GOULD, N.; ORBAN, D.; REES, T. Projected Krylov methods for saddle-point systems. *SIAM J. Matrix Anal. Appl.*, v. 35, n. 4, p. 1329–1343, 2014.
- GOULD, N. I. M. On practical conditions for the existence and uniqueness of solutions to the general equality quadratic programming problem. *Math. Program.*, v. 32, n. 1, p. 90–99, 1985.
- GOULD, N. I. M.; HRIBAR, M. E.; NOCEDAL, J. On the solution of equality constrained quadratic programming problems arising in optimization. *SIAM J. Sci. Comput.*, v. 23, n. 4, p. 1376–1395, 2001.
- GUNZBURGER, M. D. *Finite Element Methods for Viscous Incompressible Flows: A Guide to Theory, Practice, and Algorithms*. San Diego, CA: Academic Press, 1989.
- HESTENES, M. R. *Conjugate Direction Methods in Optimization*. Nova Iorque, NY: Springer-Verlag, 1980.
- HORN, R. A.; JOHNSON, C. R. *Matrix Analysis*. 2. ed. Nova Iorque, NY: Cambridge University Press, 2013.

- HU, Q.; SHI, Z.; YU, D. Efficient solvers for saddle-point problems arising from domain decompositions with Lagrange multipliers. *SIAM J. Numer. Anal.*, v. 42, n. 3, p. 905–933, 2004.
- ISAACSON, E.; KELLER, H. B. *Analysis of Numerical Methods*. Mineola, NY: Dover Publications, Inc., 1994.
- JAMES, D.; PLEMMONS, R. J. An iterative substructuring algorithm for equilibrium equations. *Numer. Math.*, v. 57, n. 1, p. 625–633, 1990.
- KELLEY, C. T. *Iterative Methods for Linear and Nonlinear Equations*. Filadélfia, PA: SIAM, 1995.
- KREYSZIG, E. *Introductory Functional Analysis with Applications*. Nova Iorque, NY: John Wiley & Sons, Inc., 1978.
- LUENBERGER, D. G. *Linear and Nonlinear Programming*. 2. ed. Reading, MA: Addison-Wesley, 1984.
- MENDES, D.; DINIZ-EHRHARDT, M. A.; PEDROSO, L. G. A method based on the augmented lagrangian technique for the analytical resolution of saddle point problems. In: MACI 2015, 5., 2015, Tandil. *Proceedings of the V MACI 2015*. Tandil, Argentina: ASAMACI, 2015. p. 143–146.
- NOCEDAL, J.; WRIGHT, S. J. *Numerical Optimization*. 2. ed. Nova Iorque, NY: Springer-Verlag, 2006.
- NOTAY, Y. A new analysis of block preconditioners for saddle point problems. *SIAM J. Matrix Anal. Appl.*, v. 35, n. 1, p. 143–173, 2014.
- PEARSON, J. W.; WATHEN, A. J. A new approximation of the Schur complement in preconditioners for PDE-constrained optimization. *Numer. Linear Algebra Appl.*, v. 19, n. 5, p. 816–829, 2012.
- PERUGIA, I.; SIMONCINI, V.; ARIOLI, M. Linear algebra methods in a mixed approximation of magnetostatic problems. *SIAM J. Sci. Comput.*, v. 21, n. 3, p. 1085–1101, 1999.
- POWELL, M. J. D. *On updating the inverse of a KKT matrix*. Relatório Técnico DAMTP 2004/NA01. Cambridge, UK: Department of Applied Mathematics and Theoretical Physics, Cambridge University, 2004. 25 p.
- _____. The NEWUOA software for unconstrained optimization without derivatives. In: DI PILLO, G.; ROMA, M. (Org.). *Large-Scale Nonlinear Optimization*. Nova Iorque, NY: Springer, 2006. p. 255–297.
- ROZLOŽNÍK, M.; SIMONCINI, V. Krylov subspace methods for saddle point problems with indefinite preconditioning. *SIAM J. Matrix Anal. Appl.*, v. 24, n. 2, p. 368–391, 2002.
- SAAD, Y.; SCHULTZ, M. H. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. and Stat. Comput.*, v. 7, n. 3, p. 856–869, 1986.

- STEWART, G. W. On scaled projections and pseudoinverses. *Linear Algebra and its Applications*, v. 112, p. 189–193, 1989.
- TREFETHEN, L. N.; BAU III, D. *Numerical Linear Algebra*. Filadélfia, PA: SIAM, 1997.
- VAN DER VORST, H. A. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. and Stat. Comput.*, v. 13, n. 2, p. 631–644, 1992.
- VAVASIS, S. A. Stable numerical algorithms for equilibriums systems. *SIAM J. Matrix Anal. Appl.*, v. 15, n. 4, p. 1108–1131, 1994.
- WENDLAND, H. *Scattered Data Approximation*. Nova Iorque, NY: Cambridge University Press, 2005.
- WRIGHT, M. H. Interior methods for constrained optimization. *Acta Numerica*, v. 1, p. 341–407, 1992.

APÊNDICE A – Métodos estacionários para sistemas lineares

Sejam $A \in \mathbb{R}^{n \times n}$ uma matriz não-singular e $b \in \mathbb{R}^n$ um vetor qualquer. Nós queremos encontrar a solução $x^* \in \mathbb{R}^n$ de um sistema linear

$$Ax = b \quad (\text{A.1})$$

construindo uma sequência $(x^{(k)})$ em \mathbb{R}^n que convirja para x^* , isto é, para a qual

$$x^* = \lim_{k \rightarrow \infty} x^{(k)}.$$

Uma maneira relativamente simples para obtermos x^* é com procedimentos que constroem sequências de pontos em \mathbb{R}^n a partir de um dado ponto $x^{(0)} \in \mathbb{R}^n$, determinando sucessivamente os termos da sequência através de alguma relação de recorrência

$$x^{(k+1)} = T(x^{(k)}), \quad k \in \mathbb{N}_0, \quad (\text{A.2})$$

expressa por meio de uma aplicação $T : \mathbb{R}^n \rightarrow \mathbb{R}^n$. Chamamos um tal procedimento de **método iterativo**, o processo de obtenção de um termo da sequência, de **iteração** e o próprio termo, de **iterando**.

Os métodos iterativos mais simples de serem deduzidos e analisados são talvez os chamados **métodos estacionários**. Eles recebem esse nome porque a resolução de um sistema linear com eles é equivalente à determinação do único **ponto fixo (estacionário)** da aplicação $T : \mathbb{R}^n \rightarrow \mathbb{R}^n$ que os definem, isto é, do único ponto $x^* \in \mathbb{R}^n$ tal que

$$T(x^*) = x^*.$$

Usualmente utilizamos o teorema do ponto fixo de Banach para garantir que uma aplicação $T : \mathbb{R}^n \rightarrow \mathbb{R}^n$ tem um único ponto fixo x^* , pois ele assegura que isto ocorre se existe uma constante $\alpha \in (0, 1)$ tal que

$$\|T(x) - T(y)\|_2 \leq \alpha \|x - y\|_2 \quad \text{para todo } x, y \in \mathbb{R}^n. \quad (\text{A.3})$$

A demonstração deste teorema garante ainda que a sequência $(x^{(k)})$, obtida através de (A.2), converge para o ponto fixo de T independentemente da escolha inicial de $x^{(0)}$ e que o erro que se comete ao se aproximar x^* por $x^{(k)}$ é tal que

$$\|x^{(k)} - x^*\|_2 \leq \frac{\alpha^k}{1 - \alpha} \|x^{(1)} - x^{(0)}\|_2.$$

((KREYSZIG, 1978, p. 300-302)). Claramente, quanto menor for a constante α , mais rápida será a convergência de um tal método estacionário para uma dada precisão $\epsilon > 0$.

Podemos construir uma classe bastante ampla de métodos estacionários, com base em decomposições da matriz A da forma

$$A = P - N,$$

onde P e N são matrizes reais de ordem n , com P não-singular. Para tanto, reescrevemos (A.1) como $Px = Nx + b$ e então definimos a relação de recorrência

$$x^{(k+1)} = P^{-1}(Nx^{(k)} + b), \quad k \in \mathbb{N}_0,$$

motivados por esta equação. Esta relação também é interessante porque ela gera uma única sequência $(x^{(k)})$ em \mathbb{R}^n para cada par de pontos $x^{(0)}$ e b fixados, devido a não-singularidade da matriz P . Fazemos $M = P^{-1}N$ e $c = P^{-1}b$ para colocá-la na forma

$$x^{(k+1)} = Mx^{(k)} + c, \quad k \in \mathbb{N}_0, \quad (\text{A.4})$$

que corresponde à aplicação afim $T(x) = Mx + c$. O teorema do ponto fixo de Banach e o corolário de sua demonstração podem ser aplicados se garantirmos que T satisfaz a propriedade (A.3). Evidentemente isto envolve a matriz M , porque

$$\|T(x) - T(y)\|_2 = \|M(x - y)\|_2 \leq \|M\|_2 \|x - y\|_2.$$

Uma condição suficiente para a convergência de $(x^{(k)})$ para x^* é então $\|M\|_2 < 1$.

Seja $\rho(M)$ o **raio espectral** da matriz M , isto é, o maior dos autovalores de M , em valor absoluto. Um resultado melhor sobre a convergência de $(x^{(k)})$ é o seguinte.

Proposição A.1. *A sequência $(x^{(k)})$ determinada por (A.4) converge para a solução x^* de (A.1), independentemente da escolha de $x^{(0)}$, se, e somente se, $\rho(M) < 1$.*

Demonstração. Pode ser encontrada em (ISAACSON; KELLER, 1994, p. 14, 62-63). \square

Uma vez que a convergência da sequência $(x^{(k)})$ depende do raio espectral da matriz M , que a constante $\|M\|_2$ está relacionada à velocidade dessa convergência e que $\rho(M) \leq \|M\|_2$, é razoável esperar que o número mínimo de iterações necessário para um desses métodos estacionários atingir uma certa precisão $\epsilon = 10^{-\ell}$, $\ell > 0$, também dependa de $\rho(M)$. De fato, conforme mostram Isaacson e Keller (1994, p. 64),

$$k \geq \frac{\ell}{-\log(\rho(M))}$$

para que haja uma redução no erro $\|x^{(k)} - x^*\|_2$ por um fator de pelo menos ϵ .

Métodos iterativos como os acima são chamados algumas vezes na literatura de **métodos de decomposição**. Exemplos clássicos de métodos de decomposição são os métodos de Jacobi, Gauss-Seidel e SOR ((KELLEY, 1995, p. 7-9)).

APÊNDICE B – Alternativas para interpolação de dados

Em aplicações práticas de matemática, é frequente o problema em que uma função escalar e real f precisa ser aproximada por uma outra função escalar e real g , usualmente contínua, em razão de f não ser totalmente conhecida ou apresentar uma expressão muito complicada e de poucas propriedades algébricas e analíticas. Como é de se imaginar, existem diversas maneiras de se fazer uma tal aproximação, sendo a interpolação uma das mais simples e conhecidas de todas elas. Neste apêndice fazemos uma breve introdução a essa técnica, procurando explorar em seu contexto uma classe menos tradicional de funções geradoras da aproximação g , a saber: a classe das funções condicionalmente definidas positivas.

Os tópicos deste apêndice são como seguem: na Seção B.1 nós apresentamos o problema de interpolação em si, enquanto nas Seções B.2 e B.3 exibimos esquemas menos populares de interpolação, respectivamente dados a partir de funções definidas positivas e funções condicionalmente definidas positivas.

B.1 O problema de interpolação de dados

Seja f uma função escalar e real definida em uma região U de \mathbb{R}^s . Suponha que U contém um conjunto $Z = \{z_1, \dots, z_n\}$ de n pontos distintos, onde os valores de f são conhecidos. O problema de **interpolação** é determinar uma função $g : U \rightarrow \mathbb{R}$ que coincida com f em todos os pontos de Z , isto é, para a qual

$$g(z_i) = f(z_i), \quad i \in \{1, \dots, n\}.$$

Quando isso é possível, dizemos que g é um **interpolador** de f com relação a Z , ou ainda, que g **interpola** o conjunto de dados

$$\{(z_i, f(z_i)) \mid i \in \{1, \dots, n\}\}.$$

Os pontos z_1, \dots, z_n são por isso chamados **pontos de interpolação** e as relações acima, **condições de interpolação**. A notação que utilizamos para um interpolador de f com relação a Z é \tilde{f}_Z , para que fique claro sua dependência com relação a este conjunto.

Fica claro da maneira como foi formulado que uma grande variedade de funções interpoladoras pode ser considerada no problema de interpolação. A seleção de um ou outro tipo de interpolador depende, no entanto, de vários fatores, alguns dos quais exploraremos

mais adiante. Dito isso, passamos a considerar a partir de agora o caso em que o interpolador \tilde{f}_Z pertence a um espaço real e n -dimensional de funções \mathcal{F}_U gerado por funções escalares definidas em U . Pois, nesta situação, é fácil garantir ao interpolador propriedades como continuidade e diferenciabilidade, além de uma forma relativamente descomplicada, dada pela combinação linear de algumas funções base b_1, \dots, b_n :

$$\tilde{f}_Z(x) = \sum_{j=1}^n \xi_j b_j(x),$$

onde $\xi_1, \dots, \xi_n \in \mathbb{R}$. A caracterização de um interpolador desta forma fica obviamente reduzida à determinação dos coeficientes ξ_1, \dots, ξ_n , que são, em número, iguais às condições de interpolação. Estas passam a ser simplesmente

$$\sum_{j=1}^n \xi_j b_j(z_i) = f(z_i), \quad i \in \{1, \dots, n\}.$$

Juntas, todas elas formam um sistema linear de n equações em n incógnitas, cuja representação matricial

$$\begin{bmatrix} b_1(z_1) & \cdots & b_n(z_1) \\ \vdots & \ddots & \vdots \\ b_1(z_n) & \cdots & b_n(z_n) \end{bmatrix} \begin{bmatrix} \xi_1 \\ \vdots \\ \xi_n \end{bmatrix} = \begin{bmatrix} f(z_1) \\ \vdots \\ f(z_n) \end{bmatrix} \quad (\text{B.1})$$

nós denotamos simplificadaamente por $M\xi = a$, sem exibir explicitamente nela qualquer dependência nas funções base ou pontos de interpolação. A matriz M , de entradas

$$M_{ij} = b_j(z_i), \quad i, j \in \{1, \dots, n\},$$

é chamada **matriz de interpolação**. Dizemos que um problema de interpolação está **bem-colocado** quando ele admite uma única solução. Uma vez fixada uma base do espaço de funções \mathcal{F}_U , a boa-colocação do problema de interpolação claramente é equivalente a não-singularidade da matriz M . Além disso, ela independe de base, pois quaisquer duas representações de um mesmo interpolador estão relacionadas por uma mudança de bases.

B.2 Interpolação com funções definidas positivas

Seja dado um conjunto $Z = \{z_1, \dots, z_n\} \subset \mathbb{R}^s$. Nesta seção vamos considerar interpoladores de uma função $f : \mathbb{R}^s \rightarrow \mathbb{R}$ com relação a Z , que são gerados a partir de funções base que dependem dos pontos de interpolação e que ainda dão origem a matrizes de interpolação definidas positivas. A dependência das referidas funções nos pontos de Z é uma condição necessária para que possamos ter problemas bem-colocados de interpolação, em várias variáveis (Teorema de Mairhuber-Curtis em (FASSHAEUER, 2007, p. 4), se não desejamos limitar os pontos de interpolação a determinados subconjuntos de \mathbb{R}^s . Já a

positividade da matriz de interpolação é que de fato assegura a boa-colocação do problema de interpolação. Para tanto, assumimos que $b_j(x) = \Phi(x - z_j)$, $j \in \{1, \dots, n\}$, para alguma função $\Phi : \mathbb{R}^s \rightarrow \mathbb{R}$ fixada, porém ainda a ser definida. A forma do interpolador resultante passa a ser então

$$\tilde{f}_Z(x) = \sum_{j=1}^n \xi_j \Phi(x - z_j). \quad (\text{B.2})$$

A determinação desse interpolador novamente se resume a resolução do sistema linear (B.1), cuja matriz de interpolação M tem agora entradas dadas por

$$M_{ij} = \Phi(z_i - z_j), \quad i, j \in \{1, \dots, n\}.$$

Garantimos que esta matriz é não-singular tomando Φ entre as funções que fazem de M uma matriz definida positiva.

Definição B.1. Dizemos que uma função contínua $\Phi : \mathbb{R}^s \rightarrow \mathbb{R}$ é **definida positiva**, se Φ é uma função par e, para todo $n \in \mathbb{N}$, todo conjunto $Z = \{z_1, \dots, z_n\} \subset \mathbb{R}^s$ de pontos distintos e todo vetor $(\xi_1, \dots, \xi_n) \in \mathbb{R}^n \setminus \{0\}$, a forma quadrática

$$\sum_{i=1}^n \sum_{j=1}^n \xi_i \xi_j \Phi(z_i - z_j)$$

é positiva.

Existe caracterização integral para funções definidas positivas, o que nos mostra que elas são objetos bem compreendidos (Teoremas 6.6, 6.8 e 6.11 em (WENDLAND, 2005, p. 70-75)). No entanto, trata-se de uma caracterização que requer conhecimentos bastante avançados de teoria da medida e de transformadas de Fourier. Por isso, ao invés de considerá-la em toda a sua extensão, procuramos simplificar um pouco esta introdução ao assunto, prestando atenção somente às **funções definidas positivas radiais**, isto é, àquelas funções definidas positivas para as quais existe uma função $\phi : [0, \infty) \rightarrow \mathbb{R}$ tal que $\Phi(x) = \phi(\|x\|_2)$ para todo $x \in \mathbb{R}^s$. Fazemos isso porque contamos com dois critérios alternativos simples que nos deixam decidir quando uma dada função é definida positiva e radial em \mathbb{R}^s .

Definição B.2. Dizemos que uma função $\psi : [0, \infty) \rightarrow \mathbb{R}$ é **completamente monótona em** $(0, \infty)$ se ψ é suave em $(0, \infty)$ e tal que

$$(-1)^\ell \psi^{(\ell)}(r) \geq 0$$

para todo $r > 0$ e todo $\ell \in \mathbb{N}_0$. Se a função ψ ainda é contínua em $[0, \infty)$, então também dizemos que ela é **completamente monótona em** $[0, \infty)$.

Teorema B.1. Considere uma função $\phi : [0, \infty) \rightarrow \mathbb{R}$. Então, a função $\Phi(x) = \phi(\|x\|_2)$ é definida positiva e radial em \mathbb{R}^s , para todo s , se, e somente se, $\psi(r) = \phi(\sqrt{r})$ é completamente monótona em $[0, \infty)$ e não-constante.

Demonstração. Pode ser encontrada em (WENDLAND, 2005, p. 95). \square

Exemplo B.1. A função $\Phi(x) = \phi(\|x\|_2)$ determinada por $\phi(r) = e^{-\alpha r^2}$, $\alpha > 0$, é definida positiva e radial em \mathbb{R}^s , para todo s , porque $\psi(r) = e^{-\alpha r}$ é uma função não-constante e completamente monótona em $[0, \infty)$. De fato, já que

$$(-1)^\ell \psi^{(\ell)}(r) = \alpha^\ell e^{-\alpha r} \geq 0$$

para todo $\ell \in \mathbb{N}_0$. A função Φ assim dada é chamada de **gaussiana**.

Exemplo B.2. A função $\Phi(x) = \phi(\|x\|_2)$ dada por $\phi(r) = (1 + r^2)^{-\beta}$, $\beta \geq 0$, é definida positiva e radial em \mathbb{R}^s , para todo s , porque $\psi(r) = (1 + r)^{-\beta}$ é uma função não-constante e completamente monótona em $[0, \infty)$. De fato, já que

$$(-1)^\ell \psi^{(\ell)}(r) = (-1)^{2\ell} \beta(\beta + 1) \cdots (\beta + \ell - 1)(1 + r)^{-(\beta + \ell)} \geq 0$$

para todo $\ell \in \mathbb{N}_0$. A função Φ assim dada é chamada de **multiquadrática inversa**.

Definição B.3. Seja $k \in \mathbb{N}$ tal que $k \geq 2$. Dizemos que uma função $\phi : (0, \infty) \rightarrow \mathbb{R}$ é **k -vezes monótona** em $(0, \infty)$ se ϕ é $k - 2$ vezes diferenciável neste intervalo e tal que $(-1)^\ell \phi^{(\ell)}(r)$ é não-negativa, não-crescente e convexa para todo $\ell \in \{0, 1, \dots, k - 2\}$. Se $k = 1$, apenas exigimos que $\phi : (0, \infty) \rightarrow \mathbb{R}$ seja contínua, não-negativa e não-crescente.

No que segue, $\lfloor r \rfloor$ denota o maior inteiro menor ou igual a um número real r e $\lceil r \rceil$, o menor inteiro maior ou igual a r .

Teorema B.2. Seja $k = \lfloor s/2 \rfloor + 2$ um inteiro positivo. Se $\phi : [0, \infty) \rightarrow \mathbb{R}$ é uma função contínua, não-constante e k -vezes monótona em $(0, \infty)$, então $\Phi(x) = \phi(\|x\|_2)$ é definida positiva e radial em \mathbb{R}^s , para qualquer s que satisfaça a desigualdade $\lfloor s/2 \rfloor \leq k - 2$.

Demonstração. Pode ser encontrada em (FASSHAUER, 2007, p. 51). \square

Exemplo B.3. A função $\Phi(x) = \phi(\|x\|_2)$, dada por

$$\phi_k(r) = (1 - r)_+^k = \begin{cases} (1 - r)^k, & \text{se } r \leq 1, \\ 0, & \text{se } r > 1, \end{cases}$$

é definida positiva e radial em \mathbb{R}^s , qualquer que seja $k \geq \lfloor s/2 \rfloor + 2$, porque ϕ é uma função contínua, não-constante e k -vezes monótona em $(0, \infty)$. De fato, pois como

$$(-1)^\ell \phi_k^{(\ell)}(r) = k(k - 1) \cdots (k - \ell + 1)(1 - r)_+^{k - \ell} \geq 0,$$

para todo $\ell \in \{0, 1, \dots, k\}$, nós vemos que as primeiras $k + 1$ derivadas são não-negativas e não-crescentes. E a convexidade das mesmas segue simplesmente do fato de que

$$(-1)^\ell \phi_k^{(\ell+2)}(r) = (-1)^{\ell+2} \phi_k^{(\ell+2)}(r) \geq 0,$$

isto é, de que a segunda derivada das próprias derivadas são não-negativas. A função ϕ_k é chamada de **função de potência truncada**.

Exemplo B.4. Todas as seguintes funções, $\phi_{s,k}$, $k \in \{0, 1, 2, 3\}$, possuem suporte compacto e dão origem a funções definidas positivas radiais $\Phi : \mathbb{R}^s \rightarrow \mathbb{R}$:

$$\begin{aligned}\phi_{s,0}(r) &= (1-r)_+^\ell, \\ \phi_{s,1}(r) &= (1-r)_+^{\ell+1}[(\ell+1)r+1], \\ \phi_{s,2}(r) &= (1-r)_+^{\ell+2}[(\ell^2+4\ell+3)r^2+(3\ell+6)r+3], \\ \phi_{s,3}(r) &= (1-r)_+^{\ell+3}[(\ell^3+9\ell^2+23\ell+15)r^3+(6\ell^2+36\ell+45)r^2+(15\ell+45)r+15],\end{aligned}$$

onde $\ell = \lfloor s/2 \rfloor + k + 1$ e o símbolo $_+$ indica, exclusivamente aqui, igualdade a menos de uma constante positiva. A demonstração destas afirmações não é nem um pouco óbvia e, por isso, nos referimos a (WENDLAND, 2005, p. 119-129) para maiores detalhes sobre ela. As funções indicadas são os exemplos mais importantes das chamadas **funções de Wendland de suporte compacto**.

B.3 Interpolação com funções condicionalmente definidas positivas

Polinômios em várias variáveis não são totalmente adequados à interpolação de qualquer conjunto de dados, porque somente alguns posicionamentos relativos dos pontos de interpolação é que garantem a boa-colocação do problema de interpolação neste caso. Seja $\mathcal{P}_d(\mathbb{R}^s)$ o espaço vetorial real de todos os polinômios em s variáveis, cujo grau não é superior a d . Pode-se mostrar por indução que $m = \dim(\mathcal{P}_d(\mathbb{R}^s)) = \binom{d+s}{s}$. A noção a que estamos nos referindo é formalmente expressa por meio da seguinte definição.

Definição B.4. Dizemos que um conjunto $Z = \{z_1, \dots, z_n\}$ de \mathbb{R}^s com $n \geq m$ pontos é $\mathcal{P}_d(\mathbb{R}^s)$ -**unisolvante** se o polinômio nulo é o único polinômio de $\mathcal{P}_d(\mathbb{R}^s)$ que se anula em todos os pontos de Z .

Seja $\{p_1, \dots, p_m\}$ uma base qualquer de $\mathcal{P}_d(\mathbb{R}^s)$. Claramente, afirmar que um conjunto $Z = \{z_1, \dots, z_n\}$ de n pontos de \mathbb{R}^s é $\mathcal{P}_d(\mathbb{R}^s)$ -unisolvante é equivalente a dizer que o sistema linear homogêneo de matriz de coeficientes $\bar{M} = (\bar{M}_{ij})$, dada por

$$\bar{M}_{ij} = p_j(z_i), \quad i \in \{1, \dots, n\}, \quad j \in \{1, \dots, m\},$$

admite somente a solução trivial. Ou ainda: que a matriz \bar{M} tem posto-coluna completo. Quando $n = m$, isto também significa que \bar{M} é não-singular. Nesta situação podemos tomar $b_j = p_j$, para todo $j = 1, \dots, n$, para ver que a matriz \bar{M} coincide com a matriz M do problema de interpolação (B.1). Assim a definição anterior conduz à boa-colocação do problema de interpolação polinomial, dado $n = m$, e vice-versa. Mas termos de localizar um conjunto de cardinalidade m , que ainda seja $\mathcal{P}_d(\mathbb{R}^s)$ -unisolvante, para poder garantir a existência e unicidade de um interpolador é muito restritivo. E impossível em outros casos.

Exemplo B.5. Suponha que um conjunto $Z \subset \mathbb{R}^s$ contém 5 pontos. Então, quando é possível, interpolação polinomial com relação a Z não está unicamente determinada, porque não existe espaço de polinômios em duas variáveis de dimensão $m = 5$.

Exemplo B.6. Não existe conjunto de $n = 6$ pontos do círculo unitário

$$S^1 = \{(x_1, x_2) \in \mathbb{R}^2 \mid x_1^2 + x_2^2 = 1\}$$

que seja $\mathcal{P}_2(\mathbb{R}^2)$ -unisolvante com relação à base canônica de $\mathcal{P}_2(\mathbb{R}^2)$. De outra forma: quando é possível, interpolação quadrática a partir de $\{1, x_1, x_2, x_1x_2, x_1^2, x_2^2\}$ e de $n = \dim(\mathcal{P}_2(\mathbb{R}^2))$ pontos de S^1 não está unicamente determinada. De fato, pois a coluna de entradas unitárias da matriz de interpolação M pode, neste caso, ser escrita como uma combinação linear das colunas de entradas da forma x_1^2 e x_2^2 , por causa da relação $x_1^2 + x_2^2 = 1$ que é satisfeita pelos pontos de interpolação. Em vista disso e da multilinearidade da função determinante, ocorre que o $\det(M) = 0$.

Funções definidas positivas, por outro lado, não são totalmente adequadas à interpolação de funções polinomiais, pois, por exemplo, claramente se vê que o gráfico de um interpolador como (B.2) para a função $f(x_1, x_2) = x_1 + x_2$, construído a partir da função gaussiana $\Phi(x) = e^{-36\|x\|_2^2}$ e qualquer conjunto $Z \subset \mathbb{R}^2$, se assemelha muito pouco ao plano que f representa em \mathbb{R}^3 , já que esse gráfico possui picos acentuados centrados em cada ponto de Z . Incorporamos ao problema de interpolação a necessidade de poder reproduzir adequadamente polinômios, considerando para uma dada função $f : \mathbb{R}^s \rightarrow \mathbb{R}$ e um dado conjunto $Z = \{z_1, \dots, z_n\} \subset \mathbb{R}^s$, interpoladores da forma

$$\tilde{f}_Z(x) = \sum_{j=1}^n \xi_j \Phi(x - z_j) + \sum_{k=1}^m v_k p_k(x), \quad (\text{B.3})$$

onde p_1, \dots, p_m constituem uma base de $\mathcal{P}_{d-1}(\mathbb{R}^s)$. Exigimos que eles satisfaçam as condições de interpolação usuais,

$$\sum_{j=1}^n \xi_j \Phi(z_i - z_j) + \sum_{k=1}^m v_k p_k(z_i) = f(z_i), \quad i \in \{1, \dots, n\}, \quad (\text{B.4})$$

e eliminamos os seus graus de liberdade adicionais, impondo para eles as condições de que

$$\sum_{j=1}^n \xi_j p_k(z_j) = 0, \quad k \in \{1, \dots, m\}. \quad (\text{B.5})$$

O uso de polinômios de grau menor do que s em \tilde{f}_Z assegura precisão polinomial a este interpolador. Informalmente, isto quer dizer que f e \tilde{f}_Z coincidem localmente quando \tilde{f}_Z pode ser calculado e a própria função f é um polinômio de grau menor do que s .

A determinação de um interpolador da forma (B.3), baseado em (B.4) e (B.5), é equivalente a resolução do sistema linear

$$M \begin{bmatrix} \xi \\ v \end{bmatrix} := \begin{bmatrix} A & B^T \\ B & O \end{bmatrix} \begin{bmatrix} \xi \\ v \end{bmatrix} = \begin{bmatrix} a \\ 0 \end{bmatrix},$$

onde as matrizes $A = (A_{ij}) \in \mathbb{R}^{n \times n}$ e $B = (B_{kj}) \in \mathbb{R}^{m \times n}$ e os vetores $a \in \mathbb{R}^n$, $\xi \in \mathbb{R}^n$ e $v \in \mathbb{R}^m$ são assim definidos:

$$\begin{aligned} A_{ij} &= \Phi(z_i - z_j), \quad B_{kj} = p_k(z_j), \quad i, j \in \{1, \dots, n\}, \quad k \in \{1, \dots, m\}, \\ a &= (f(z_1), \dots, f(z_n)), \quad \xi = (\xi_1, \dots, \xi_n) \quad \text{e} \quad v = (v_1, \dots, v_m). \end{aligned}$$

A matriz de interpolação $M \in \mathbb{R}^{(n+m) \times (n+m)}$ deste problema é não-singular quando, por exemplo, B tem posto completo e A é uma matriz simétrica definida positiva no $\ker(B)$ (Teorema 3.2 em (BENZI; GOLUB; LIESEN, 2005, p. 16)). Garantimos essas propriedades a ela, empregando um conjunto Z de pontos de interpolação que é $\mathcal{P}_{d-1}(\mathbb{R}^s)$ -unisolvante e tomando Φ entre as funções que fazem de A uma matriz definida positiva no $\ker(B)$.

Definição B.5. Dizemos que uma função contínua $\Phi : \mathbb{R}^s \rightarrow \mathbb{R}$ é **condicionalmente definida positiva de ordem d** , se Φ é uma função par e, para todo $n \in \mathbb{N}$, todo conjunto $Z = \{z_1, \dots, z_n\} \subset \mathbb{R}^s$ de pontos distintos e todo vetor $(\xi_1, \dots, \xi_n) \in \mathbb{R}^n \setminus \{0\}$ satisfazendo a relação

$$\sum_{j=1}^n \xi_j p(z_j) = 0,$$

para todo $p \in \mathcal{P}_{d-1}(\mathbb{R}^s)$, a forma quadrática

$$\sum_{i=1}^n \sum_{j=1}^n \xi_i \xi_j \Phi(z_i - z_j)$$

é positiva. Em geral, a ordem d indicada sempre é a menor possível para Φ .

Assim como ocorre com as funções definidas positivas, existe caracterização integral para funções condicionalmente definidas positivas (Teorema 8.14 em (WENDLAND, 2005, p. 108)). Novamente não a mencionamos aqui, devido a necessidade de conhecimentos mais avançados de matemática. Observamos, no entanto, que toda função definida positiva é também naturalmente uma função condicionalmente definida positiva de qualquer ordem. E novamente procuramos simplificar esta introdução ao assunto, prestando atenção somente às **funções condicionalmente definidas positivas radiais**, ou seja, àquelas funções condicionalmente definidas positivas para as quais existe uma função $\phi : [0, \infty) \rightarrow \mathbb{R}$ tal que $\Phi(x) = \phi(\|x\|_2)$ para todo $x \in \mathbb{R}^s$. Fazemos isso, porque, como antes, contamos com um critério simples para a identificação de tais funções.

Teorema B.3. Considere uma função contínua $\phi : [0, \infty) \rightarrow \mathbb{R}$, suave em $(0, \infty)$, e suponha que ϕ não é um polinômio de grau menor ou igual a d . Então, a função $\Phi(x) = \phi(\|x\|_2)$ é condicionalmente definida positiva de ordem $d \in \mathbb{N}_0$ e radial em \mathbb{R}^s , para todo s , se, e somente se, $\psi(r) = \phi(\sqrt{r})$ é tal que $(-1)^d \psi^{(d)}$ é completamente monótona em $(0, \infty)$.

Demonstração. Pode ser encontrada em (WENDLAND, 2005, p. 113-115). □

Exemplo B.7. A função $\Phi(x) = \phi(\|x\|_2)$ determinada por $\phi(r) = (-1)^{[\beta]}(1+r^2)^\beta$, $\beta > 0$, $\beta \notin \mathbb{N}$, é condicionalmente definida positiva de ordem $d = [\beta]$ e radial em \mathbb{R}^s , para todo s , porque $\psi(r) = (-1)^{[\beta]}(1+r)^\beta$ não é um polinômio, uma vez que $\beta \notin \mathbb{N}$, e $(-1)^{[\beta]}\psi^{([\beta])}$ é uma função completamente monótona em $(0, \infty)$. De fato, já que

$$\psi^{(\ell)}(r) = (-1)^{[\beta]}\beta(\beta-1)\cdots(\beta-\ell+1)(1+r)^{\beta-\ell}$$

para todo $\ell \in \mathbb{N}_0$, o que implica que

$$(-1)^{[\beta]}\psi^{([\beta])}(r) = \beta(\beta-1)\cdots(\beta-[\beta]+1)(1+r)^{\beta-[\beta]}$$

é completamente monótona. Além disso, $d = [\beta]$ é o menor inteiro não-negativo tal que $(-1)^d\psi^{(d)}$ é completamente monótona. A função Φ assim dada é chamada de **multiquadrática generalizada**.

Exemplo B.8. A função $\Phi(x) = \phi(\|x\|_2)$ determinada por $\phi(r) = (-1)^{[\beta/2]}r^\beta$, $\beta > 0$, $\beta \notin 2\mathbb{N} = \{2, 4, 6, \dots\}$, é condicionalmente definida positiva de ordem $d = [\beta/2]$ e radial em \mathbb{R}^s , para todo s , porque $\psi(r) = (-1)^{[\beta/2]}r^{\beta/2}$ não é um polinômio, uma vez que $\beta \notin 2\mathbb{N}$, e $(-1)^{[\beta/2]}\psi^{([\beta/2])}$ é uma função completamente monótona em $(0, \infty)$. De fato, já que

$$\psi^{(\ell)}(r) = (-1)^{[\beta/2]}\frac{\beta}{2}\left(\frac{\beta}{2}-1\right)\cdots\left(\frac{\beta}{2}-\ell+1\right)r^{\beta/2-\ell}$$

para todo $\ell \in \mathbb{N}_0$, o que implica que $(-1)^{[\beta/2]}\psi^{([\beta/2])}$ é completamente monótona. Além disso, $d = [\beta/2]$ é o menor inteiro não-negativo tal que $(-1)^d\psi^{(d)}$ é completamente monótona. A função Φ assim dada é chamada de **potência radial**.

Exemplo B.9. A função $\Phi(x) = \phi(\|x\|_2)$ determinada por $\phi(r) = (-1)^{\beta+1}r^{2\beta}\log r$, $\beta \in \mathbb{N}$, é condicionalmente definida positiva de ordem $d = \beta + 1$ e radial em \mathbb{R}^s , para todo s , pois $\psi(r) = (-1)^{\beta+1}r^\beta\log(\sqrt{r})$ claramente não é um polinômio e $(-1)^{\beta+1}\psi^{(\beta+1)}$ é uma função completamente monótona em $(0, \infty)$. De fato, já que $2\psi(r) = (-1)^{\beta+1}r^\beta\log r$, e então

$$2\psi^{(\ell)}(r) = (-1)^{\beta+1}\beta(\beta-1)\cdots(\beta-\ell+1)r^{\beta-\ell}\log r + p_\ell(r), \quad \ell \in \{1, \dots, \beta\},$$

onde p_ℓ é um polinômio de grau $\beta - \ell$. Assim, tomando $\ell = \beta$, encontramos que

$$2\psi^{(\beta)}(r) = (-1)^{\beta+1}\beta!\log r + \gamma,$$

onde $\gamma \in \mathbb{R}$. Logo, $(-1)^{\beta+1}\psi^{(\beta+1)}(r) = (\beta!/2)r^{-1}$, que é obviamente uma função completamente monótona em $(0, \infty)$, como gostaríamos de demonstrar. A função Φ assim dada é chamada de **thin-plate spline**.

APÊNDICE C – Códigos-fontes

Código C.1 – Configuração de parâmetros comuns de várias funções

```
function [nvec, pvec, svec, xpA, xpB, del] = initparam
%%
% Purpose:
%   initparam fixes the common parameters of the functions genip,
%   gendata and genplots, so the three of them work properly as a set.
%
% Format:
%   [nvec, pvec, svec, xpA, xpB, del] = initparam
%
% Input data:
%   None.
%
% Return data:
%   nvec: A row vector with values for n.
%   pvec: A row vector with percentages to define the values of m as
%         fractions of those of n.
%   svec: A row vector with seeds for the calls of Matlab's random
%         number generator.
%   xpA: A state variable that identifies when to change the (1,1)
%         block of the saddle point systems or not. Possible values
%         are 0 or 1.
%   xpB: A state variable that identifies when to change the (2,1)
%         block of the saddle point systems or not. Possible values
%         are 0 or 1.
%   del: A nonnegative parameter to turn the (1,1) block of
%         'preconditioned' saddle point systems into a more
%         diagonally dominant matrix.
%
% Remarks:
%   1) Unless you have a very, very powerful cpu, do not set both
%       parameters xpA and xpB to 1.
%   2) It is possible to skip all the computations associated with a
%       component of nvec or pvec by setting it to -1.

nvec = [1000 3000 5000];
pvec = [0.01 0.03 0.05];
svec = 1;
xpA   = 1;
xpB   = 0;
del   = 1e16;
end
```

Código C.2 – Geração de conjunto $\mathcal{P}_1(\mathbb{R}^d)$ -unisolvante de pontos

```

function X = genip(theta)
%%
% Purpose:
%   genip generates n random interpolation points x1,...,xn in the
%   d-dimensional Euclidean space in a way the angle between any two
%   distinct points of the form [1; xi] and [1; xj] are at least equal
%   to the given value of theta. If theta is negative, then the points
%   genip generates are just randomly generated points.
%
%   The values of n and d depend on the vectors nvec and pvec from the
%   function initparam and are related: d = round(nvec(i)*pvec(j)) - 1,
%   for some i and j previously fixated.
%
%   genip generates a different set of interpolation points for each
%   component of the svec vector from initparam.
%
% Format:
%   X = genip(theta)
%
% Example:
%   X = genip(pi/10)
%
% Input data:
%   theta: The minimum angle, in radians and in the interval [0, pi/2],
%           between any two points.
%
% Return data:
%   X: A max(svec)-by-1 cell array, whose nonempty cells contain a
%       length(nvec)-by-length(pvec) cell array, whose nonempty cells
%       contain a n-by-(d+1) matrix, whose columns correspond to
%       interpolation points for some nonpolynomial function to be
%       specified by the function gensymspp.

%%
% Initializing parameters.

[nvec, pvec, svec, ~, ~, ~] = initparam;

%%
% Generating interpolation points.

X = cell(max(svec), 1);

costheta = cos(theta);

```

```

for seed = svec
    X{seed} = cell(length(nvec), length(pvec));

    for n = nvec
        if n < 0, continue; end

        i = find(nvec == n);
        mvec = round(n*pvec);

        for m = mvec
            if m < 0, continue; end

            j = find(mvec == m);
            d = m-1;

            rng(seed, 'twister');

            % If the value of theta is negative, randomly generates the set
            % of interpolation points without any angle control.
            if theta < 0
                X{seed}{i,j} = rand(d, n);
                continue;
            else
                X{seed}{i,j} = zeros(d+1, n);
                nrm = zeros(1, n);
            end

            str = sprintf(['\nGenerating interpolation points.\n\n' ...
                           'theta = %.2f deg, seed = %d, n = %d,' ...
                           ' d = %d\n'], rad2deg(theta), seed, n, d);
            wb = waitbar(0, str, 'Name', 'genip', 'CreateCancelBtn', ...
                        'setappdata(gcf, ''Cancel'', 1)');
            wbcl = findobj(wb, 'Type', 'Patch');
            set(wbcl, 'EdgeColor', [0 0 0], 'FaceColor', [0 0.5 1]);

            % A point is generated and accepted as the first 'interpolation
            % point'.
            X{seed}{i,j}(:,1) = [1; rand(d, 1)];
            nrm(1) = norm(X{seed}{i,j}(:,1));

            waitbar(1/n);

            % The other n-1 points are generated with angle control.
            for r = 2 : n
                flag1 = true;
                flag2 = false;

                % Keep generating a new point until the angle between this
                % point and any other point already accepted as an

```

```

% 'interpolation point' is as desired.
while flag1
    Xr      = [1; rand(d, 1)];
    nrm(r) = norm(Xr);

    for s = 1 : r-1
        aux = dot(Xr, X{seed}{i,j}(:,s)) / (nrm(r)*nrm(s));

        if aux > costheta
            flag2 = true;
            break;
        end
    end

    if ~flag2
        % Another point is accepted as an 'interpolation point'.
        X{seed}{i,j}(:,r) = Xr;
        flag1              = false;

        waitbar(r/n);
    else
        flag2 = false;
    end

    if getappdata(wb, 'Cancel')
        delete(wb);
        return;
    end
end

% The first component of each column of X is stripped out so the
% vectors that remain are truly interpolation points in a
% d-dimensional Euclidean space.
X{seed}{i,j} = X{seed}{i,j}(2:d+1,:);

delete(wb);
end
end
end
end
end

```


Código C.3 – Geração de problema de ponto-de-sela simétrico

```

function [A, B, a, b] = gensymspp(X, npoly, zeta, tau, seed, varA, varB)
%%
% Purpose:
% gensymspp generates a symmetric saddle point system of the form
% [A B'; B 0] [x; y] = [a; b], where the n-by-n matrix A is symmetric
% and positive definite on ker(B) and the m-by-n matrix B has full
% rank.
%
% To obtain such a linear system, gensymspp employs a function that
% has two parts, one polynomial and other that depends on a
% (conditionally) definite positive basis function, as if this
% function interpolate the columns of X.
%
% By modifying the value of varB, it is possible to take some rows
% of the block B as rows of the Hilbert matrix of order n, so
% increasing the ill-conditioning of B.
%
% The solution of the linear system thus generated is a vector of
% ones.
%
% Format:
% [A, B, a, b] = gensymspp(X, npoly, zeta, tau, seed, varA, varB)
%
% Example:
% [A, B, a, b] = gensymspp(X, 'gauss', 0, 1, 4, 3, 2)
%
% Input data:
% X: A (m-1)-by-n matrix.
% npoly: A string to identify the nonpolynomial part of the
% implicit interpolation function. Possible values are:
% 'gauss' = Gaussian;
% 'invmultquad' = inverse multiquadric;
% 'multquad' = multiquadric;
% 'radpow' = radial power;
% 'thin' = thin plate spline;
% 'trunpow' = truncated power;
% 'wend' = Wendland's compactly supported function.
% zeta: The nonexponential parameter of npoly. Set it to any value
% if npoly does not have a nonexponential parameter.
% tau: The exponential parameter of npoly. Set it to any value in
% the case of the 'wend' npoly function.
% seed: A nonnegative integer used here as a mean of identification.
% varA: A nonnegative integer used here as a mean of identification.
% varB: A nonnegative integer used to take the last varB-1 rows of
% the block B as the first varB-1 rows of the Hilbert matrix
% of order n.

```

```

%
% Return data:
%   A: The (1,1) block of the saddle point system generated.
%   B: The (2,1) block of the saddle point system generated.
%   a: The first n components of the saddle point system's rhs
%       generated.
%   b: The last m components of the saddle point system's rhs
%       generated.

%%
% Initializing settings.

[d, n] = size(X);
m      = d+1;

%%
% Setting the nonpolynomial part of the interpolation function.

switch upper(npoly)
case 'GAUSS'
    if tau <= -2*eps
        fprintf(2, '\nThe given value to tau is not positive.\n\n');
        return
    else
        tau = abs(tau);
    end

    npolyh = @(r) exp(-tau*r^2);

case 'INVMULTQUAD'
    if zeta <= -2*eps
        fprintf(2, '\nThe given value to zeta is not positive.\n\n');
        return
    else
        zeta = abs(zeta);
    end

    if tau <= -2*eps
        fprintf(2, '\nThe given value to tau is not positive.\n\n');
        return
    else
        tau = abs(tau);
    end

    npolyh = @(r) (zeta^2 + r^2)^(-tau);

```

```

case 'MULTQUAD'
    if abs(round(tau)-tau) < 2*eps
        fprintf(2, '\nThe given value to tau is an integer.\n\n');
        return
    elseif tau <= -2*eps
        fprintf(2, '\nThe given value to tau is not positive.\n\n');
        return
    elseif ceil(abs(tau)) ~= 2
        fprintf(2, ['\nConsidering a linear polynomial tail to the ' ...
                    'interpolation model, the value\nof tau has to ' ...
                    'be in the interval (1,2).\n\n']);
        return
    else
        tau = abs(tau);
    end

    npolyh = @(r) (-1)^ceil(tau) * (zeta^2 + r^2)^tau;

case 'RADPOW'
    if mod(abs(tau),2) == 0
        fprintf(2, '\nThe given value to tau is an even integer.\n\n');
        return
    elseif tau <= -2*eps
        fprintf(2, '\nThe given value to tau is not positive.\n\n');
        return
    elseif ceil(abs(tau/2)) ~= 2
        fprintf(2, ['\nConsidering a linear polynomial tail to the ' ...
                    'interpolation model, the value\nof tau has to ' ...
                    'be in the interval (2,4).\n\n']);
        return
    else
        tau = abs(tau);
    end

    npolyh = @(r) (-1)^ceil(tau/2) * r^tau;

case 'THIN'
    npolyh = @(r) r^2 * log10(r);

case 'TRUNPOW'
    if abs(round(tau)-tau) >= 2*eps
        fprintf(2, '\nThe given value to tau is not an integer.\n\n');
        return
    elseif tau < 2*eps
        fprintf(2, '\nThe given value to tau is not positive.\n\n');
        return

```

```

elseif tau < floor(d/2)+1-2*eps
    fprintf(2, ['\nThe given value to tau is not greater or ' ...
               'equal to %d.\n\n'], floor(d/2)+1);
    return
end

npolyh = @(r) (r<=1) * (1-r)^tau;

case 'WEND'
    tau = floor(d/2) + 3;
    npolyh = @(r) (r<=1) * (1-r)^tau * (tau*r + 1);

otherwise
    fprintf(2, ['\nThe given npoly does not correspond to any ' ...
               'implemented nonpolynomial basis\nfunction.\n\n']);
    return
end

%%
% Generating the blocks A and B of a symmetric saddle point system.

str = sprintf(['\nGenerating a symmetric saddle point system.' ...
               '\n\nseed = %d,    varA = %d,    varB = %d,    ' ...
               '\n = %d,    m = %d\n'], seed, varA, varB, n, m);
wb = waitbar(0, str, 'Name', 'gensymspp', 'CreateCancelBtn', ...
            'setappdata(gcf, ''Cancel'', 1)');
wbcl = findobj(wb, 'Type', 'Patch');
set(wbcl, 'EdgeColor', [0 0 0], 'FaceColor', [0 0.5 1]);

% Replacing the last varB-1 rows of the given matrix X by the first
% varB-1 rows of the Hilbert matrix of order n.
H = hilb(n);
X(d-varB+2:d,:) = H(1:varB-1,:);

% Generating the block B;
B = [ones(1, n); X];

% Generating the block A;
A = zeros(n);

if strcmpi(npoly, 'THIN')
    A(1,1) = 0;
else
    A(1,1) = npolyh(0);
end

```

```

for i = 1 : n
    A(i,i) = A(1,1);

    Xi = X(:,i);

    for j = i+1 : n
        A(i,j) = npolyh(norm(Xi - X(:,j)));
        A(j,i) = A(i,j);
    end

    waitbar(i/(n+1));
    if getappdata(wb, 'Cancel')
        delete(wb);
        return;
    end
end

%%
% Generating the right hand side of a symmetric saddle point system,
% so its solution is a vector of ones: (1,...,1).
rhs = [A B'; B zeros(m)] * ones(n+m, 1);
a    = rhs(1:n);
b    = rhs(n+1:n+m);

%%
% Ensuring the full rankness of the matrix B when none of the rows of
% X were replaced by rows of a Hilbert matrix.

if (varB == 1) && (abs(rank(B)-m) >= 2*eps)
    fprintf(2, '\nThe matrix B has not full rank.\n\n');

    delete(wb);
    return;
end

waitbar(1);
delete(wb);
end

```

Código C.4 – Conversão de problema de ponto-de-sela simétrico em generalizado

```

function [A, a, b] = sym2genspp(A, B, gmode, seed)
%%
% Purpose:
%   sym2genspp converts a symmetric saddle point system of the form
%   [A B'; B 0] [x; y] = [a; b] that was generated by the function
%   gensymspp into a generalized saddle point system of the same form,
%   by changing A into a (1,1) block whose symmetric part is still
%   positive definite on ker(B).
%
%   The right hand side vector of the original linear system is
%   recomputed, so the new problem also has a vector of ones as
%   solution.
%
% Format:
%   [A, a, b] = sym2genspp(A, B, gmode, seed)
%
% Example:
%   [A, a, b] = sym2genspp(A, B, 'fast', 6)
%
% Input data:
%   A: The (1,1) block of a symmetric saddle point system.
%   B: The (2,1) block of a symmetric saddle point system.
%   gmode: A string to identify how to generate the skew-symmetric
%           part of the new block A. Possible values are 'fast', 'slow'
%           and 'other'.
%   seed: A nonnegative integer used here as a mean of identification.
%
% Return data:
%   A: The (1,1) block of the generalized saddle point system.
%   a: The first n components of the generalized saddle point system's
%       right hand side.
%   b: The last m components of the generalized saddle point system's
%       right hand side.

%%
% Initializing settings.

opts1.LT = true;
opts2.UT = true;

[m, n] = size(B);

%%
% Updating the matrix A and generating the new rhs of the system.

```

```

rng(seed+10, 'twister');

% Generating any random skew-symmetric matrix.
Ass = rand(n);
Ass = Ass - Ass';
Ass = 0.5 * Ass;

% The mean order of magnitude of the entries of the first possible
% candidate to skew-symmetric part of A is set to be the same as the
% mean order of magnitude of the entries of the symmetric part of A.
eta = meanabs(A) / meanabs(Ass);
Ass = eta * Ass;

if strcmpi(gmode, 'FAST') || strcmpi(gmode, 'SLOW')
    A = A + Ass;
else
    % Building auxiliary objects invAst1 and invAst2.
    As      = A;
    Wnm     = B';
    [Q, R] = qr(Wnm, 0);

    sig     = normest(B);
    sig     = sig * sig;
    R       = R';
    Wmn     = linsolve(R, B, opts1);
    R       = R';
    Wmn     = linsolve(R, Wmn, opts2);
    Vmn     = Wmn * As;
    Wnn     = Wnm * Vmn;
    Wnn     = As - Wnn;
    Vnm     = Wnn * Wnm;
    Ast     = Vnm * Wmn;
    Ast     = Wnn - Ast;
    gamst   = normest(Ast);
    gamst   = gamst / sig;
    Vnm     = gamst * Wnm;
    Wnn     = Vnm * B;
    Ast     = Ast + Wnn;

    invAst1 = Ast \ eye(n);

    Wmn     = Q';
    Vmn     = Wmn * As;
    Wnn     = Q * Vmn;
    Wnn     = As - Wnn;
    Wnm     = Wnn * Q;
    Ast     = Wnm * Wmn;
    Ast     = Wnn - Ast;
    gamst   = normest(Ast);

```

```

Vnm    = gamst * Q;
Wnn    = Vnm * Wmn;
Ast    = Ast + Wnn;

invAst2 = Ast \ eye(n);

% Approximately finding how much reducing in the magnitude of the
% entries of Ass is needed to force the spectral radius of the matrix
% (I-P)*Ass*invAst1*(I-P) to be less than one.
omega1 = 1;
auxAss = Ass;
rho1    = normest(auxAss*invAst1);

while rho1 >= 1-2*eps
    omega1 = 0.9572 * omega1;
    auxAss = omega1 * auxAss;
    rho1    = normest(auxAss*invAst1);
end

% Approximately finding how much reducing in the magnitude of the
% entries of Ass is needed to force the spectral radius of the matrix
% (I-P)*Ass*invAst2*(I-P) to be less than one.
omega2 = 1;
auxAss = Ass;
rho2    = normest(auxAss*invAst2);

while rho2 >= 1-2*eps
    omega2 = 0.9572 * omega2;
    auxAss = omega2 * auxAss;
    rho2    = normest(auxAss*invAst2);
end

% Defining a matrix A with skew-symmetric part Ass such that the
% spectral radius of both matrices (I-P)*Ass*invAst1*(I-P) and
% (I-P)*Ass*invAst2*(I-P) are less than one.
omega = min(omega1, omega2);
Ass    = omega * Ass;
A      = As + Ass;
end

%%
% Generating the right hand side of the generalized saddle point
% system, so its solution is a vector of ones: (1,...,1).

rhs = [A B'; B zeros(m)] * ones(n+m,1);
a    = rhs(1:n);
b    = rhs(n+1:n+m);
end

```


Código C.5 – Atualização do bloco (1,1) de problemas de ponto-de-sela

```

function [A, a, b, cA] = updspp(A0, B, lmin0, cA, npoly, seed, varA,varB)
%%
% Purpose:
%   updspp updates the (1,1) block and the right hand side of a saddle
%   point system so the new (1,1) block has a worse 2-norm condition
%   number than the original one.
%
%   The system's new right hand side vector is still such that the
%   solution of the system is a vector of ones: (1,...,1).
%
% Format:
%   [A, a, b, cA] = updspp(A0, B, lmin0, cA, npoly, seed, varA, varB)
%
% Example:
%   [A, a, b, cA] = updspp(A0, B, lmin0, 2.34*1e8, 'gauss', 9, 8, 1)
%
% Input data:
%   A0: The (1,1) block of a saddle point system.
%   B: The (2,1) block of a saddle point system.
%   lmin0: The smallest eigenvalue of A0 in absolute value. It can be a
%   complex eigenvalue.
%   cA: The 2-norm condition number of the current matrix A being
%   used by the gendata function when the call to updspp occurs.
%   npoly: Different nonpolynomial functions sometimes require
%   particular calculations to speed up the updspp function.
%   This string identifies the nonpolynomial function in use by
%   the gendata function in order to achieve this speed up.
%   seed: Some nonnegative integer used here as a mean of
%   identification.
%   varA: A nonnegative integer that brings lmin near to zero. The
%   updspp function enters in a infinite loop for big values of
%   varA due to the stagnation of Matlab's function cond.
%   varB: Some nonnegative integer used here as a mean of
%   identification.
%
% Return data:
%   A: The updated (1,1) block of a saddle point system.
%   a: The first n components of the new saddle point system's rhs.
%   b: The last m components of the new saddle point system's rhs.
%   cA: The 2-norm condition number of the matrix A.
%
% Remark:
%   There is no guarantee this function work to update a (1,1) block
%   whose skew-symmetric part were generated by the sym2genspp function
%   using the gmode option 'other'.

```

```

%%
% Initializing settings.

[m, n] = size(B);

%%
% Updating A so to increase its condition number.

str = sprintf(['\nUpdating the (1,1) block of the saddle point ' ...
              'system given.\n\n' ...
              'seed = %d,    varA = %d,    varB = %d,    n = %d,' ...
              '    m = %d\n'], seed, varA, varB, n, m);
wb = waitbar(0, str, 'Name', 'upspp', 'CreateCancelBtn', ...
            'setappdata(gcf, ''Cancel'', 1)');
wbcl = findobj(wb, 'Type', 'Patch');
set(wbcl, 'EdgeColor', [0 0 0], 'FaceColor', [0 0.5 1]);

ord1 = floor(log10(cA));
ord2 = ord1 + 1;

switch upper(npoly)
case 'GAUSS'
    epsilon = varA + 1.125;

case 'TRUNPOW'
    epsilon = varA + 1;

case 'WEND'
    epsilon = varA + 1.125;

otherwise
    epsilon = varA - 1;
end

% Keep bringing the smallest eigenvalue of the matrix A0, in absolute
% value, to zero, until the order of magnitude of the original constant
% cA passed to this function increases at least by one.
while ord1 < ord2
    A = A0 - ((1-1/10^epsilon)*lmin0)*eye(n);
    cA = cond(A);
    ord1 = floor(log10(cA));
    epsilon = epsilon + 1;

    if getappdata(wb, 'Cancel')
        delete(wb);
        return;
    end
end
end

```

```
waitbar(1/2);

%%
% Generating the right hand side of the new saddle point system, so its
% solution is a vector of ones: (1,...,1).
rhs = [A B'; B zeros(m)] * ones(n+m, 1);
a    = rhs(1:n);
b    = rhs(n+1:n+m);

waitbar(1);
delete(wb);
end
```

Código C.6 – Geração de dados para análise dos algoritmos

```

function [D1, D2] = gendata(X, npoly, zeta, tau, gmode, del, imax)
%%
% Purpose:
%   gendata generates many measures for Algorithms 1 and 2 or
%   Algorithms 3 and 4, which are all stored at 'data sets' and can be
%   accessed by the following structure:
%   Dt{seed}{varA,varB}(i,j,k), t = 1,2,
%   where
%   k = 1: condition number of the matrix M;
%   k = 2: condition number of the matrix A;
%   k = 3: condition number of the matrix As;
%   k = 4: condition number of the matrix B;
%   k = 5: condition number of the matrix blkdiag(Ast1, gamst1*(B*B'))
%           or of the matrix blkdiag(Ast2, gamst2*eye(m));
%   k = 6: condition number of the matrix Ast_t, t = 1,2;
%   k = 7: condition number of the matrix B*B';
%   k = 8: mean absolute value of the entries of the Schur complement
%           of Ast_t in [Ast_t B'; B 0];
%   k = 9: Percentage of sparsity of the matrix A;
%   k = 10: Algorithm t/t+2 v1's relative error, t = 1,2;
%   k = 11: Algorithm t/t+2 v2's relative error, t = 1,2;
%   k = 12: BiCGSTAB's relative error;
%   k = 13: GMRES' relative error;
%   k = 14: Algorithm t/t+2 v1's relative residual, t = 1,2;
%   k = 15: Algorithm t/t+2 v2's relative residual, t = 1,2;
%   k = 16: BiCGSTAB's relative residual;
%   k = 17: GMRES' relative residual;
%   k = 18: Algorithm t/t+2 v1's time, t = 1,2;
%   k = 19: Algorithm t/t+2 v2's time, t = 1,2;
%   k = 20: BiCGSTAB's time;
%   k = 21: GMRES' time;
%
% Format:
%   [D1, D2] = gendata(X, npoly, zeta, tau, gmode, del, imax)
%
% Example:
%   [D1, D2] = gendata(X, 'gauss', 0, 1, 'slow', 1e12, 4)
%
% Input data:
%   X: A (m-1)-by-n matrix.
%   npoly: A string to identify the nonpolynomial part of the implicit
%           interpolation function. Possible values are:
%           'gauss'           = Gaussian;
%           'invmultquad'     = inverse multiquadric;
%           'multquad'        = multiquadric;
%           'radpow'          = radial power;

```

```

%          'thin'          = thin plate spline;
%          'trunpow'       = truncated power;
%          'wend'         = Wendland's compactly supported function.
%  zeta: The nonexponential parameter of npoly. Set it to any value
%        if npoly does not have a nonexponential parameter.
%  tau: The exponential parameter of npoly. Set it to any value in
%       the case of the 'wend' npoly function.
%  gmode: (Optional) A string that specifies the method to solve the
%          given problem. Possible values are: 'fast', 'slow' and
%          'other'.
%  del: (Optional) A nonnegative parameter to turn the (1,1) block
%        of 'preconditioned' saddle point systems into a more
%        diagonally dominant matrix.
%  imax: (Optional) The number of terms minus one in the truncation
%        of the Neumann series of the matrix I+E.
%
% Return data:
%  D1: Data set to Algorithm 1 or 3, accordingly to the gmode option.
%  D2: Data set to Algorithm 2 or 4, accordingly to the gmode option.

%%
% Initializing parameters.

switch nargin
case 4
    gmode = 'SYM';
    [nvec, pvec, svec, xpA, xpB, del] = initparam;
    imax = 0;

case 5
    [nvec, pvec, svec, xpA, xpB, del] = initparam;
    imax = 0;

case 6
    [nvec, pvec, svec, xpA, xpB, ~] = initparam;
    imax = 0;

case 7
    [nvec, pvec, svec, xpA, xpB, ~] = initparam;
end

if strcmpi(gmode, 'OTHER') && (nargin < 7)
    fprintf(2, ['\nWith the ''other'' gmode option it is mandatory ' ...
               'to specify the values of \nthe parameters del and ' ...
               '\nimax.\n\n']);
    return;
end

```

```

if xpA == 1, vAvec = 1:17; else vAvec = 1; end
if xpB == 1, vBvec = 1:10; else vBvec = 1; end

opts1.LT = true;
opts2.UT = true;

%%
% Creating data matrices for Algorithms 1 or 3 and 2 or 4.

D1 = cell(max(svec), 1);
D2 = cell(max(svec), 1);

for seed = svec
    D1{seed} = cell(max(vAvec), max(vBvec));
    D2{seed} = cell(max(vAvec), max(vBvec));

    for varA = vAvec
        for varB = vBvec
            D1{seed}{varA,varB} = zeros(length(nvec), length(pvec), 21);
            D2{seed}{varA,varB} = zeros(length(nvec), length(pvec), 21);
        end
    end
end

%%
% Generating data.

for seed = svec
    for varB = vBvec
        for n = nvec
            if n < 0, continue; end

            i = find(nvec == n);
            mvec = round(n*pvec);

            for m = mvec
                if m < 0, continue; end

                j = find(mvec == m);

                for varA = vAvec
                    if find(vAvec(1) == varA)
                        % Generating a symmetric saddle point problem of order n+m
                        % and known solution: (1,...,1).
                        if strcmpi(npoly, 'TRUNPOW')
                            tau = floor((m-1)/2) + 1;
                        end
                    end
                end
            end
        end
    end
end

```

```

[A, B, a, b] = ...
gensymspp(X{seed}{i,j}, npoly, zeta, tau, seed,varA,varB);

% Changing the previous symmetric saddle point problem into
% a generalized saddle point problem for which the solution
% still is a vector of ones.
if ~strcmpi(gmode, 'SYM')
    [A, a, b] = sym2genspp(A, B, gmode, seed);
end

% Determining the (complex) eigenvalue of A nearest to
% zero.
A0          = A;
lvec         = eig(A0);
abslvec      = abs(lvec);
minabslvec   = min(abslvec);
lmin0        = lvec(find(abslvec == minabslvec, 1));

switch upper(npoly)
case 'GAUSS'
    maxcA = 12;

case 'TRUNPOW'
    maxcA = 12;

case 'WEND'
    maxcA = 12;

otherwise
    maxcA = 16;
end

D1{seed}{varA,varB}(i,j,2) = cond(A);
else
    varAold = vAvec(find(vAvec == varA)-1);

if floor(log10(D1{seed}{varAold,varB}(i,j,2))) < maxcA
    % Updating the (1,1) block of a saddle point system in
    % order to make it more ill-conditioned. The right hand
    % side of the system is recalculated, so its solution
    % remains being (1,...,1).
    [A, a, b, D1{seed}{varA,varB}(i,j,2)] = ...
        updspp(A0, B, lmin0, D1{seed}{varAold,varB}(i,j,2), ...
            npoly, seed, varA, varB);
else
    D1{seed}{varA,varB}(i,j,2) = Inf;
    continue;
end
end
end

```

```

%%
% Building some objects that are needed to test each
% algorithm. The value of del is zero at this step, so the
% matrices Ast1 and Ast2 built correspond to their
% theoretical versions.
str = sprintf(['\nBuilding some objects to test the ' ...
              'algorithms.\n\nseed = %d,    varA = %d,' ...
              '    varB = %d,    n = %d,    m = %d\n'], ...
              seed, varA, varB, n, m);
wb = waitbar(0, str, 'Name', 'gendata', ...
            'CreateCancelBtn', ...
            'setappdata(gcf, ''Cancel'', 1)');
wbcl = findobj(wb, 'Type', 'Patch');
set(wbcl, 'EdgeColor', [0 0 0], 'FaceColor', [0 0.5 1]);

M = [A B'; B zeros(m)];
rhs = [a; b];
nrhs = norm(rhs);

if strcmpi(gmode, 'SYM')
    As = A;
else
    As = 0.5 * (A + A');
end

Wnm = B';
[Q, R] = qr(Wnm, 0);

waitbar(1/3);
if getappdata(wb, 'Cancel')
    delete(wb);
    return;
end

% Auxiliary matrix Ast1 associated with Algorithms 1 and 3.
sig = normest(B);
sig = sig * sig;
R = R';
Wmn = linsolve(R, B, opts1);
R = R';
Wmn = linsolve(R, Wmn, opts2);
Vmn = Wmn * As;
Wnn = Wnm * Vmn;
Wnn = As - Wnn;
Vnm = Wnn * Wnm;
Ast1 = Vnm * Wmn;
Ast1 = Wnn - Ast1;
gamst1 = normest(Ast1);
gamst1 = gamst1 / sig;

```



```

Vnm    = gamst1 * Wnm;
Wnn    = Vnm * B;
Ast1   = Ast1 + Wnn;

waitbar(2/3);
if getappdata(wb, 'Cancel')
    delete(wb);
    return;
end

% Auxiliary matrix Ast2 associated with Algorithms 2 and 4.
Wmn    = Q';
Vmn    = Wmn * As;
Wnn    = Q * Vmn;
Wnn    = As - Wnn;
Wnm    = Wnn * Q;
Ast2   = Wnm * Wmn;
Ast2   = Wnn - Ast2;
gamst2 = normest(Ast2);
Vnm    = gamst2 * Q;
Wnn    = Vnm * Wmn;
Ast2   = Ast2 + Wnn;

waitbar(1);
delete(wb);

%%
% Collecting some initial data.

str = sprintf(['\nCollecting data.\n\n' ...
              'seed = %d,    varA = %d,    varB = %d,' ...
              '    n = %d,    m = %d\n'], ...
              seed, varA, varB, n, m);
wb = waitbar(0, str, 'Name', 'main', 'CreateCancelBtn', ...
            'setappdata(gcbf, ''Cancel'', 1)');
wbcl = findobj(wb, 'Type', 'Patch');
set(wbcl, 'EdgeColor', [0 0 0], 'FaceColor', [0 0.5 1]);

D1{seed}{varA,varB}(i,j,1) = cond(M);
D2{seed}{varA,varB}(i,j,1) = D1{seed}{varA,varB}(i,j,1);

waitbar(2/16);
if getappdata(wb, 'Cancel')
    delete(wb);
    return;
end

D2{seed}{varA,varB}(i,j,2) = D1{seed}{varA,varB}(i,j,2);

```

```

if strcmpi(gmode, 'SYM')
    D1{seed}{varA,varB}(i,j,3) = D1{seed}{varA,varB}(i,j,2);
else
    D1{seed}{varA,varB}(i,j,3) = cond(As);
end
D2{seed}{varA,varB}(i,j,3) = D1{seed}{varA,varB}(i,j,3);

D1{seed}{varA,varB}(i,j,4) = cond(B);
D2{seed}{varA,varB}(i,j,4) = D1{seed}{varA,varB}(i,j,4);

waitbar(3/16);
if getappdata(wb, 'Cancel')
    delete(wb);
    return;
end

C = B*B';

D1{seed}{varA,varB}(i,j,5) = cond(blkdiag(Ast1, gamst1*C));
D1{seed}{varA,varB}(i,j,6) = cond(Ast1);
D1{seed}{varA,varB}(i,j,7) = cond(C);

waitbar(8/16);
if getappdata(wb, 'Cancel')
    delete(wb);
    return;
end

D2{seed}{varA,varB}(i,j,5) = ...
    cond(blkdiag(Ast2, gamst2*eye(m)));
D2{seed}{varA,varB}(i,j,6) = cond(Ast2);
D2{seed}{varA,varB}(i,j,7) = D1{seed}{varA,varB}(i,j,7);

waitbar(12/16);
if getappdata(wb, 'Cancel')
    delete(wb);
    return;
end

%%
% Checking that the Schur complement of Ast in Mst is,
% numerically, a multiple of the identity matrix, as expected
% from the theory.

C = B * (Ast1 \ B');
D1{seed}{varA,varB}(i,j,8) = meanabs((-gamst1)*C + eye(m));

C = Q' * (Ast2 \ Q);

```

```

D2{seed}{varA,varB}(i,j,8) = meanabs((-gamst2)*C + eye(m));

waitbar(13/16);
if getappdata(wb, 'Cancel')
    delete(wb);
    return;
end

%%
% Computing the percentage of sparsity of the matrix A.

D1{seed}{varA,varB}(i,j,9) = nnz(A) / n^2;
D2{seed}{varA,varB}(i,j,9) = D1{seed}{varA,varB}(i,j,9);

%%
% Computing with different solvers the solution of the saddle
% point problem generated above and the many relative errors,
% residuals and times spent obtaining it.

if strcmpi(gmode, 'SYM')
    set1 = {
        '[x, y] = algo1(A, B, a, b, del, ''bicgstab'');', ...
        '[x, y] = algo1(A, B, a, b, del, ''gmres'');'
    };

    set2 = {
        '[x, y] = algo2(A, B, a, b, del, ''bicgstab'');', ...
        '[x, y] = algo2(A, B, a, b, del, ''gmres'');'
    };
else
    set1 = {
        '[x, y] = algo3(A, B, a, b, gmode, del, imax, ''bicgstab'');', ...
        '[x, y] = algo3(A, B, a, b, gmode, del, imax, ''gmres'');'
    };

    set2 = {
        '[x, y] = algo4(A, B, a, b, gmode, del, imax, ''bicgstab'');', ...
        '[x, y] = algo4(A, B, a, b, gmode, del, imax, ''gmres'');'
    };
end

set3 = {'[z, ~] = bicgstab(M, rhs, tol1, round(n/20));', ...
        '[z, ~] = gmres(M, rhs, [], tol2, round(n/20));'};

e = ones(n+m, 1);

for k = 1 : 2

```

```

eval(strcat('tic; ', set1{k}, ...
    ' D1{seed}{varA,varB}(i,j,17+k) = toc;'));

D1{seed}{varA,varB}(i,j,9+k) = norm([x;y]-e) / sqrt(n+m);
D1{seed}{varA,varB}(i,j,13+k) = norm(M*[x;y]-rhs) / nrhs;

eval(strcat('tic; ', set2{k}, ...
    ' D2{seed}{varA,varB}(i,j,17+k) = toc;'));

D2{seed}{varA,varB}(i,j,9+k) = norm([x;y]-e) / sqrt(n+m);
D2{seed}{varA,varB}(i,j,13+k) = norm(M*[x;y]-rhs) / nrhs;
end

waitbar(14/16);
if getappdata(wb, 'Cancel')
    delete(wb);
    return;
end

% Finding the smallest order of magnitude of the relative
% residuals obtained with Algorithms 1 and 2 or 3 and 4 for
% both inner solvers: BiCGSTAB and GMRES.
tol1 = min(floor(log10(D1{seed}{varA,varB}(i,j,14))), ...
    floor(log10(D2{seed}{varA,varB}(i,j,14))));
tol2 = min(floor(log10(D1{seed}{varA,varB}(i,j,15))), ...
    floor(log10(D2{seed}{varA,varB}(i,j,15))));

for k = 1 : 2
    eval(strcat('tic; ', set3{k}, ...
        ' D1{seed}{varA,varB}(i,j,19+k) = toc;'));

    D1{seed}{varA,varB}(i,j,11+k) = norm(z - e) / sqrt(n+m);
    D1{seed}{varA,varB}(i,j,15+k) = norm(M*z - rhs) / nrhs;

    D2{seed}{varA,varB}(i,j,11+k) = ...
        D1{seed}{varA,varB}(i,j,11+k);
    D2{seed}{varA,varB}(i,j,15+k) = ...
        D1{seed}{varA,varB}(i,j,15+k);
    D2{seed}{varA,varB}(i,j,19+k) = ...
        D1{seed}{varA,varB}(i,j,19+k);

    waitbar((14+k)/16);
    if getappdata(wb, 'Cancel')
        delete(wb);
        return;
    end
end

waitbar(1);

```

```
        delete(wb);  
    end  
end  
end  
end  
end  
end
```

Código C.7 – Geração de gráficos dos dados dos algoritmos

```

function genplots(D1, D2, xp, vA, vB, x, y, z, ...
                xLimits, xSpc, yLimits, ySpc, xLab, yLab)
%%
% Purpose:
%   genplots generates a grid of 3-by-3 plots corresponding to the
%   experiments performed by the gendata function. Since both cond(A)
%   and cond(B) can change during these experiments, there is need to
%   fix the value of one the the inner variables varA and varB of this
%   function, so the plots correspond to a 'cross-section' of the
%   results of the experiments. This is done through the parameters vA
%   and vB as explained below.
%
% Format:
%   genplots(D1, D2, xp, vA, vB, x, y, z, ...
%           xLimits, xSpc, yLimits, ySpc, xLab, yLab)
%
% Example:
%   genplots(D1, D2, 1, [], 1, 2, [10 11], [12 13], ...
%       [1e0 1e16], 4, [1e-16 1e0], 2, '$cond(A)$', 'Relative error')
%
% Input data:
%   D1: 'Data set' 1.
%   D2: 'Data set' 2.
%   xp: Identification of the experiment performed by the function
%       gendata, for which the 'data sets' D1 and D2 correspond.
%       Possible values are: 1 (for the experiments where cond(A)
%       changes) and 2 (for the experiments where cond(B)
%       changes).
%   vA: If xp==1, vA has to be set to []. Else, vA is any value
%       between 1 and 17 with which the value of the inner
%       variable varA is set.
%   vB: If xp==2, vB has to be set to []. Else, vB is any value
%       between 1 and 10 with which the value of the inner
%       variable varB is set.
%   x: A positive integer that represents only one entry of the
%       'data sets', which expresses the quantity in the x-axis of
%       the plots.
%   y: Any number of positive integers that represent the entries
%       of the 'data sets', which express the quantities in the y
%       coordinates of the curves on the plots. Each quantity is
%       plotted twice, one for each version of our algorithms.
%       This parameter can be set to empty.
%   z: Any number of positive integers that represent the entries
%       of the 'data sets', which express the quantities in the y
%       coordinates of the curves on the plots. Each quantity is
%       plotted only once. This parameter can be set to empty.

```

```

%   xLimits: A 1-by-2 row vector with the x-axis limits.
%   xSpc: The 'power-space' between tick marks on the x-axis.
%   yLimits: A 1-by-2 row vector with the y-axis limits.
%   ySpc: The 'power-space' between tick marks on the y-axis.
%   xLab: The label of the x-axis.
%   yLab: The label of the y-axis.
%
% Return data:
%   None.

%%
% Initializing parameters.

[nvec, pvec, svec, xpA, xpB, ~] = initparam;

if xpA == 1, vAvec = 1:17; else vAvec = 1; end
if xpB == 1, vBvec = 1:10; else vBvec = 1; end

ordmax = floor(log10(xLimits(2)));
scl     = 16.8/24.5;

close all;

figure('Units', 'centimeters', ...
       'Position', [0 0 16.8 24.5], ...
       'PaperUnits', 'centimeters', ...
       'PaperPosition', [0 0 16.8 24.5], ...
       'PaperPositionMode', 'manual');

%%
% Averaging measures which correspond to all seed values.

ln = length(nvec);
lm = length(pvec);
ls = length(svec);
lvA = length(vAvec);
lvB = length(vBvec);
ly = length(y);
lz = length(z);

Avg1 = cell(max(vAvec), max(vBvec));
Avg2 = cell(max(vAvec), max(vBvec));

for varA = vAvec
    for varB = vBvec
        Avg1{varA,varB} = zeros(ln, lm, 21);
        Avg2{varA,varB} = zeros(ln, lm, 21);
    end
end

```

```

for k = 1 : 21
    for seed = svec
        Avg1{varA,varB}(:, :, k) = Avg1{varA,varB}(:, :, k) ...
                                + D1{seed}{varA,varB}(:, :, k);
        Avg2{varA,varB}(:, :, k) = Avg2{varA,varB}(:, :, k) ...
                                + D2{seed}{varA,varB}(:, :, k);
    end

    Avg1{varA,varB}(:, :, k) = ...
        Avg1{varA,varB}(:, :, k) ./ (ls*ones(ln,lm));
    Avg2{varA,varB}(:, :, k) = ...
        Avg2{varA,varB}(:, :, k) ./ (ls*ones(ln,lm));
end
end
end

%%
% Generating x and y vectors to plot curves of points (x(i),y(i)).

G = cell(ln, lm);

if xp == 1
    for i = 1 : ln
        for j = 1 : lm
            G{i,j} = zeros(lvA, 2*ly+lz+1);

            for varA = vAvec
                ii = find(vAvec == varA);

                G{i,j}(ii,1) = Avg1{varA,vB}(i,j,x);

                for k = 1 : ly
                    G{i,j}(ii,k+1) = Avg1{varA,vB}(i,j,y(k));
                    G{i,j}(ii,k+ly+1) = Avg2{varA,vB}(i,j,y(k));
                end

                for k = 1 : lz
                    G{i,j}(ii,k+2*ly+1) = Avg1{varA,vB}(i,j,z(k));
                end
            end
        end
    end
else
    for i = 1 : ln
        for j = 1 : lm
            G{i,j} = zeros(lvB, 2*ly+lz+1);

            for varB = vBvec

```



```

        ii = find(vBvec == varB);

        G{i,j}(ii,1) = Avg1{vA,varB}(i,j,x);

        for k = 1 : ly
            G{i,j}(ii,k+1) = Avg1{vA,varB}(i,j,y(k));
            G{i,j}(ii,k+ly+1) = Avg2{vA,varB}(i,j,y(k));
        end

        for k = 1 : lz
            G{i,j}(ii,k+2*ly+1) = Avg1{vA,varB}(i,j,z(k));
        end
    end
end
end
end

%%
% Plotting curves and configuring the 3-by-3 grid of plots.

ax1 = zeros(ln*lm, 1);
ax2 = ax1;

ch = zeros(2*ly+lz, ln*lm);

mkrs = {'-', '-', '-', '-', '-', '-'};

for k = 1 : ln*lm
    i = floor((k-1)/lm) + 1;
    j = mod(k-1, lm) + 1;

    ax1(k) = subplot(ln,lm, k);
    p = get(ax1(k), 'Position');
    p(1) = p(1) + (j-1)*0.03;
    p(3) = 0.22;
    p(4) = scl * ((6/4) * p(3));
    set(ax1(k), 'Position', p);

    if strcmpi(yLab, 'Tempo (s)')
        str = 'ch(:,k) = semilogx(';
    else
        str = 'ch(:,k) = loglog(';
    end

    for kk = 1 : length(ch(:,1))
        ordvec = floor(log10(G{i,j}(:,1)));
        cend = max(find(ordvec < ordmax));
        str = sprintf('%sG{i,j}(1:%d,1), G{i,j}(1:%d,%d), mkrs{%d}, ', ...

```

```

        str, cend, cend, kk+1, kk);
end

str = strcat(str(1:length(str)-2), ');');
eval(str);

xlabel(ax1(k), {xLab}, 'FontSize', 9, ...
               'FontUnits', 'points', ...
               'interpreter', 'latex');

ylabel(ax1(k), {yLab}, 'FontSize', 9, ...
        'FontUnits', 'points', ...
        'interpreter', 'latex');

grid(ax1(k), 'on');

ax2(k) = axes('Position', get(ax1(k), 'Position'), ...
              'XAxisLocation', 'top', 'YAxisLocation', 'right', ...
              'XScale', 'log', 'YScale', 'log', ...
              'Color', 'none');

str = sprintf('$n = %d$, $m = %d$', nvec(i), round(pvec(j)*nvec(i)));
xlabel(ax2(k), {str}, 'FontSize', 9, ...
        'FontUnits', 'points', ...
        'interpreter', 'latex');
end

set(ax1, 'Xlim', xLimits, 'YLim', yLimits);
if strcmpi(yLab, 'Tempo (s)')
    set(ax1, 'XTick', 10.^(log10(xLimits(1)):xSpc:log10(xLimits(2))));
else
    set(ax1, 'XTick', 10.^(log10(xLimits(1)):xSpc:log10(xLimits(2))), ...
          'YTick', 10.^(log10(yLimits(1)):ySpc:log10(yLimits(2))));
end
set(ax1, 'FontSize', 6);

set(ax2, 'Xlim', xLimits, 'YLim', yLimits);
set(ax2, 'XTick', [], 'YTick', []);
set(ax2, 'FontSize', 6);

set(ax2, 'Box', 'off', 'Visible', 'off')
set(findall(ax2, 'Type', 'text'), 'Visible', 'on')

% Manually changing the scales of the second and third rows of time
% curves.
% if strcmpi(yLab, 'Tempo (s)')
%     set(ax1(4:6), 'YLim', [0 16]);
%     set(ax2(4:6), 'YLim', [0 16]);
%     set(ax1(7:9), 'YLim', [0 80]);

```

```

%      set(ax2(7:9), 'YLim', [0 80]);
%      end

set(ch, 'LineWidth', 1.00, 'MarkerSize', 4);

% Manually changing the legend of the plots.
legh = legend(ch(:,5), {'Algo1 v1', 'Algo1 v2', ...
                        'Algo2 v1', 'Algo2 v2', ...
                        'BiCGSTAB', 'GMRES'});

%      legh = legend(ch(:,5), {'Algo3 v1', 'Algo3 v2', ...
%                              'Algo4 v1', 'Algo4 v2', ...
%                              'BiCGSTAB', 'GMRES'});
%      legh = legend(ch(:,5), {'$\space^1\mathrm{cond}_2(A\_ast)$', ...
%                              '$\space^2\mathrm{cond}_2(A\_ast)$', ...
%                              '$\mathrm{cond}_2(M)$', ...
%                              '$\mathrm{cond}_2(B)$', ...
%                              '$\mathrm{cond}_2(BB^T)$'});
%      legh = legend(ch(:,5), {'$\space^1\mathrm{cond}_2(A\_ast)$', ...
%                              '$\space^2\mathrm{cond}_2(A\_ast)$', ...
%                              '$\mathrm{cond}_2(M)$', ...
%                              '$\mathrm{cond}_2(A)$', ...
%                              '$\mathrm{cond}_2(BB^T)$'});
%      legh = legend(ch(:,5), {'$\space^3\mathrm{cond}_2((A\_s)\_ast)$', ...
%                              '$\space^4\mathrm{cond}_2((A\_s)\_ast)$', ...
%                              '$\mathrm{cond}_2(M)$', ...
%                              '$\mathrm{cond}_2(A\_s)$', ...
%                              '$\mathrm{cond}_2(B)$', ...
%                              '$\mathrm{cond}_2(BB^T)$'});
%      legh = legend(ch(:,5), {'$\space^3\mathrm{cond}_2((A\_s)\_ast)$', ...
%                              '$\space^4\mathrm{cond}_2((A\_s)\_ast)$', ...
%                              '$\mathrm{cond}_2(M)$', ...
%                              '$\mathrm{cond}_2(A\_s)$', ...
%                              '$\mathrm{cond}_2(A)$', ...
%                              '$\mathrm{cond}_2(BB^T)$'});

set(legh, 'Location', 'south', ...
        'Orientation', 'horizontal', ...
        'FontSize', 6, ...
        'FontUnits', 'points', ...
        'Interpreter', 'latex');

% Manually changing the position of the legend of the plots.
set(legh, 'Position', get(legh, 'Position')+[-0.01 -0.39 0.00 0.00]);
%      set(legh, 'Position', get(legh, 'Position')+[0.00 -0.39 0.00 0.00]);
end

```

Código C.8 – Implementação do Algoritmo 3

```

function [x, y] = algo1(A, B, a, b, del, solver)
%%
% Purpose:
%   algo1 solves a symmetric saddle point system of the form
%    $[A \ B'; \ B \ 0] \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix}$ ,
%   where the n-by-n matrix A is assumed to be symmetric and positive
%   definite on  $\ker(B)$  and the m-by-n matrix B is assumed to have full
%   rank.
%
% Format:
%   [x, y] = algo1(A, B, a, b, del, solver)
%
% Example:
%   [x, y] = algo1(A, B, a, b, 1e16, 'bicgstab')
%
% Input data:
%       A: The (1,1) block of a saddle point system.
%       B: The (2,1) block of a saddle point system.
%       a: The first n components of a saddle point system's rhs.
%       b: The last m components of a saddle point system's rhs.
%       del: A nonnegative parameter to turn the (1,1) block of
%            'preconditioned' saddle point systems into a more
%            diagonally dominant matrix.
%       solver: A string that specifies with which solver, BiCGSTAB or
%              GMRES, a inner linear system will be solved. Possible
%              values are: 'bicgstab' and 'gmres'.
%
% Return data:
%       x: The first n components of a saddle point system's solution.
%       y: The last m components of a saddle point system's solution.

%%
% Initializing parameters.

warning('off', 'all');

n = size(B, 2);

opts1.LT = true;
opts2.UT = true;

%%
% Solving the saddle point problem given.

```

```

% Step 1:
sig    = normest(B);
sig    = sig * sig;
Wnm    = B';
[~, R] = qr(Wnm, 0);
R      = R';
Wmn    = linsolve(R, B, opts1);
R      = R';
Wmn    = linsolve(R, Wmn, opts2);

% Step 2:
Vmn = Wmn * A;
Wnn = Wnm * Vmn;
Wnn = A - Wnn;
Vnm = Wnn * Wnm;
At   = Vnm * Wmn;
At   = Wnn - At;

% Step 3:
gamt = normest(At);
gamt = gamt / sig;
Vnm   = gamt * Wnm;
Wnn   = Vnm * B;
At    = At + Wnn;

if del > 2*eps
    del = del * gamt;

    for i = 1 : n
        At(i,i) = At(i,i) + del;
    end
end

% Step 4:
Vnm = Wmn';
wn   = Vnm * b;
vn   = A * wn;

% Step 5:
vn = a - vn;
y  = Wmn * vn;
a  = Wnm * y;
a  = vn - a;

if strcmpi(solver, 'BICGSTAB')
    [a, ~] = bicgstab(At, a);
else
    [a, ~] = gmres(At, a);
end

```

```
x = a + wn;  
  
% Step 6:  
a = A * a;  
vn = vn - a;  
y = Wmn * vn;  
end
```

Código C.9 – Implementação do Algoritmo 4

```

function [x, y] = algo2(A, B, a, b, del, solver)
%%
% Purpose:
%   algo2 solves a symmetric saddle point system of the form
%    $[A \ B'; \ B \ 0] \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix}$ ,
%   where the n-by-n matrix A is assumed to be symmetric and positive
%   definite on  $\ker(B)$  and the m-by-n matrix B is assumed to have full
%   rank.
%
% Format:
%   [x, y] = algo2(A, B, a, b, del, solver)
%
% Example:
%   [x, y] = algo2(A, B, a, b, 1e8, 'gmres')
%
% Input data:
%       A: The (1,1) block of a saddle point system.
%       B: The (2,1) block of a saddle point system.
%       a: The first n components of a saddle point system's rhs.
%       b: The last m components of a saddle point system's rhs.
%       del: A nonnegative parameter to turn the (1,1) block of
%            'preconditioned' saddle point systems into a more
%            diagonally dominant matrix.
%       solver: A string that specifies with which solver, BiCGSTAB or
%              GMRES, a inner linear system will be solved. Possible
%              values are: 'bicgstab' and 'gmres'.
%
% Return data:
%       x: The first n components of a saddle point system's solution.
%       y: The last m components of a saddle point system's solution.
%
% Remark:
%       The thin QR factorization of B' is computed using Matlab's qr
%       command. Ideally, we would like to employ Householder
%       transformations in this procedure, but we are not sure if Matlab
%       does this.

%%
% Initializing parameters.

warning('off', 'all');

n = size(B, 2);

opts1.LT = true;

```

```

opts2.UT = true;

%%
% Solving the saddle point problem given.

% Step 1:
Wnm = B';
[Q, R] = qr(Wnm, 0);
B = Q';
P = Q * B;

% Step 2:
Wmn = B * A;
Wnn = Q * Wmn;
Wnn = A - Wnn;
Wnm = Wnn * Q;
At = Wnm * B;
At = Wnn - At;

% Step 3:
gamt = normest(At);
At = At + gamt*P;

if del > 2*eps
    del = del * gamt;
    for i = 1 : n
        At(i,i) = At(i,i) + del;
    end
end

% Step 4:
R = R';
y = linsolve(R, b, opts1);
wn = Q * y;
vn = A * wn;

% Step 5:
vn = a - vn;
y = B * vn;
a = Q * y;
a = vn - a;

if strcmpi(solver, 'BICGSTAB')
    [a, ~] = bicgstab(At, a);
else
    [a, ~] = gmres(At, a);
end

```



```
x = a + wn;  
  
% Step 6:  
a = A * a;  
vn = vn - a;  
y = B * vn;  
R = R';  
y = linsolve(R, y, opts2);  
end
```

Código C.10 – Implementação do Algoritmo 5

```

function [x, y] = algo3(As, B, a, b, gmode, del, imax, solver)
%%
% Purpose:
%   algo3 solves a generalized saddle point system of the form
%   [As B'; B 0] [x; y] = [a; b],
%   where the symmetric part of the n-by-n matrix As is assumed to be
%   positive definite on ker(B) and the m-by-n matrix B is assumed to
%   have full rank. Given a certain n-by-n matrix Ast related with the
%   blocks As and B, the matrix E=(I-P)*Ass*inv(Ast)*(I-P) is also
%   assumed to do not have any eigenvalue -1.
%
% Format:
%   [x, y] = algo3(As, B, a, b, gmode, del, imax, solver)
%
% Example:
%   [x, y] = algo3(As, B, a, b, 'other', 1e16, 4, 'bicgstab')
%
% Input data:
%   As: The (1,1) block of a saddle point system.
%   B: The (2,1) block of a saddle point system.
%   a: The first n components of a saddle point system's rhs.
%   b: The last m components of a saddle point system's rhs.
%   gmode: A string that specifies the method to solve the given
%   problem. Possible values are: 'fast', 'slow' and 'other'.
%   del: A nonnegative parameter to turn the (1,1) block of
%   'preconditioned' saddle point systems into a more
%   diagonally dominant matrix.
%   imax: The number of terms minus one in the truncation of the
%   Neumann series of the matrix I+E.
%   solver: A string that specifies with which solver, BiCGSTAB or
%   GMRES, a inner linear system will be solved. Possible
%   values are: 'bicgstab' and 'gmres'.
%
% Return data:
%   x: The first n components of a saddle point system's solution.
%   y: The last m components of a saddle point system's solution.

%%
% Initializing parameters.

warning('off', 'all');

n = size(B, 2);

opts1.LT = true;

```

```

opts2.UT = true;

flag = 0;

%%
% Solving the saddle point problem given.

% Step 0:
Ass = As';
Ass = As - Ass;
Ass = 0.5 * Ass;
As = As - Ass;

% Step 1:
sig = normest(B);
sig = sig * sig;
Wnm = B';
[~, R] = qr(Wnm, 0);
R = R';
Wmn = linsolve(R, B, opts1);
R = R';
Wmn = linsolve(R, Wmn, opts2);

% Step 2:
Vmn = Wmn * As;
Wnn = Wnm * Vmn;
Wnn = As - Wnn;
Vnm = Wnn * Wnm;
Ast = Vnm * Wmn;
Ast = Wnn - Ast;

% Step 3:
gamst = normest(Ast);
gamst = gamst / sig;
Vnm = gamst * Wnm;
Wnn = Vnm * B;
Ast = Ast + Wnn;

if del > 2*eps
    del = del * gamst;

    for i = 1 : n
        Ast(i,i) = Ast(i,i) + del;
    end
end

% Step 4a:
Vnm = Wmn';

```

```

wn = Vnm * b;
vn = As * wn;

% Step 4b:
y = Wmn * vn;
un = Wnm * y;
un = vn - un;

if strcmpi(gmode, 'SLOW')
    try
        C = chol(Ast);
        Ct = C';

        un = linsolve(Ct, un, opts1);
        un = linsolve(C, un, opts2);

        flag = 1;
    catch
        [L, D, p] = ldl(Ast, 'vector');
        Lt = L';

        un = un(p,:);
        un = linsolve(L, un, opts1);
        un = D \ un;
        un = linsolve(Lt, un, opts2);
        un = un(p,:);
    end
else
    if strcmpi(solver, 'BICGSTAB')
        [un, ~] = bicgstab(Ast, un);
    else
        [un, ~] = gmres(Ast, un);
    end
end

un = wn - un;
un = Ass * un;
y = Wmn * un;
x = Wnm * y;
un = un - x;
a = a - un;

% Step 4c:
switch upper(gmode)
    case 'SLOW'
        if flag == 1
            Wnn = linsolve(Ct, Ass, opts1);
            Wnn = linsolve(C, Wnn, opts2);
        else

```

```

    Wnn = Ass(:, p);
    Wnn = linsolve(L, Wnn, opts1);
    Wnn = D \ Wnn;
    Wnn = linsolve(Lt, Wnn, opts2);
    Wnn = Wnn(:, p);
end

Wnn = Wnn';
Vmn = Wmn * Wnn;
Ast = Wnm * Vmn;
Wnn = Wnn - Ast;
Vnm = Wnn * Wnm;
Ast = Vnm * Wmn;
Wnn = Wnn - Ast;

for i = 1 : n
    Wnn(i,i) = Wnn(i,i) - 1;
end

a = -a;

if strcmpi(solver, 'BICGSTAB')
    [a, ~] = bicgstab(Wnn, a);
else
    [a, ~] = gmres(Wnn, a);
end

case 'OTHER'
    un = a;
    sn = zeros(n, 1);

    for i = 1 : imax
        y = Wmn * un;
        x = Wnm * y;
        un = un - x;

        if strcmpi(solver, 'BICGSTAB')
            [un, ~] = bicgstab(Ast, un);
        else
            [un, ~] = gmres(Ast, un);
        end

        un = Ass * un;
        x = power(-1, i) * un;
        sn = sn + x;
    end

    y = Wmn * sn;
    x = Wnm * y;

```

```
    a = a + sn - x;
end

% Step 5:
vn = a - vn;
y  = Wmn * vn;
a  = Wnm * y;
a  = vn - a;

if strcmpi(gmode, 'SLOW')
    if flag == 1
        a = linsolve(Ct, a, opts1);
        a = linsolve(C, a, opts2);
    else
        a = a(p,:);
        a = linsolve(L, a, opts1);
        a = D \ a;
        a = linsolve(Lt, a, opts2);
        a = a(p,:);
    end
else
    if strcmpi(solver, 'BICGSTAB')
        [a, ~] = bicgstab(Ast, a);
    else
        [a, ~] = gmres(Ast, a);
    end
end

x = a + wn;

% Step 6:
un = Ass * a;
a  = As * a;
a  = a + un;
vn = vn - a;
un = Ass * wn;
vn = vn - un;
y  = Wmn * vn;
end
```

Código C.11 – Implementação do Algoritmo 6

```

function [x, y] = algo4(As, B, a, b, gmode, del, imax, solver)
%%
% Purpose:
%   algo4 solves a generalized saddle point system of the form
%   [As B'; B 0] [x; y] = [a; b],
%   where the symmetric part of the n-by-n matrix As is assumed to be
%   positive definite on ker(B) and the m-by-n matrix B is assumed to
%   have full rank. Given a certain n-by-n matrix Ast related with the
%   blocks As and B, the matrix E=(I-P)*Ass*inv(Ast)*(I-P) is also
%   assumed to do not have any eigenvalue -1.
%
% Format:
%   [x, y] = algo4(As, B, a, b, gmode, del, imax, solver)
%
% Example:
%   [x, y] = algo4(As, B, a, b, 'fast', 1e8, 0, 'gmres')
%
% Input data:
%   As: The (1,1) block of a saddle point system.
%   B: The (2,1) block of a saddle point system.
%   a: The first n components of a saddle point system's rhs.
%   b: The last m components of a saddle point system's rhs.
%   gmode: A string that specifies the method to solve the given
%   problem. Possible values are: 'fast', 'slow' and 'other'.
%   del: A nonnegative parameter to turn the (1,1) block of
%   'preconditioned' saddle point systems into a more
%   diagonally dominant matrix.
%   imax: The number of terms minus one in the truncation of the
%   Neumann series of the matrix I+E.
%   solver: A string that specifies with which solver, BiCGSTAB or
%   GMRES, a inner linear system will be solved. Possible
%   values are: 'bicgstab' and 'gmres'.
%
% Return data:
%   x: The first n components of a saddle point system's solution.
%   y: The last m components of a saddle point system's solution.
%
% Remark:
%   The thin QR factorization of B' is computed using Matlab's qr
%   command. Ideally, we would like to employ Householder
%   transformations in this procedure, but we are not sure if Matlab
%   does this.

%%
% Initializing parameters.

```

```

warning('off', 'all');

n = size(B, 2);

opts1.LT = true;
opts2.UT = true;

flag = 0;

%%
% Solving the saddle point problem given.

% Step 0:
Ass = As';
Ass = As - Ass;
Ass = 0.5 * Ass;
As = As - Ass;

% Step 1:
Wnm = B';
[Q, R] = qr(Wnm, 0);
B = Q';

% Step 2:
Wmn = B * As;
Wnn = Q * Wmn;
Wnn = As - Wnn;
Wnm = Wnn * Q;
Ast = Wnm * B;
Ast = Wnn - Ast;

% Step 3:
gamst = normest(Ast);
Wnm = gamst * Q;
Wnn = Wnm * B;
Ast = Ast + Wnn;

if del > 2*eps
    del = del * gamst;

    for i = 1 : n
        Ast(i,i) = Ast(i,i) + del;
    end
end

% Step 4a:
R = R';
y = linsolve(R, b, opts1);

```



```

wn = Q * y;
vn = As * wn;

% Step 4b:
y = B * vn;
un = Q * y;
un = vn - un;

if strcmpi(gmode, 'SLOW')
    try
        C = chol(Ast);
        Ct = C';

        un = linsolve(Ct, un, opts1);
        un = linsolve(C, un, opts2);

        flag = 1;
    catch
        [L, D, p] = ldl(Ast, 'vector');
        Lt = L';

        un = un(p,:);
        un = linsolve(L, un, opts1);
        un = D \ un;
        un = linsolve(Lt, un, opts2);
        un = un(p,:);
    end
else
    if strcmpi(solver, 'BICGSTAB')
        [un, ~] = bicgstab(Ast, un);
    else
        [un, ~] = gmres(Ast, un);
    end
end

un = wn - un;
un = Ass * un;
y = B * un;
x = Q * y;
un = un - x;
a = a - un;

% Step 4c:
switch upper(gmode)
    case 'SLOW'
        if flag == 1
            Wnn = linsolve(Ct, Ass, opts1);
            Wnn = linsolve(C, Wnn, opts2);
        else

```

```

    Wnn = Ass(:, p);
    Wnn = linsolve(L, Wnn, opts1);
    Wnn = D \ Wnn;
    Wnn = linsolve(Lt, Wnn, opts2);
    Wnn = Wnn(:, p);
end

Wnn = Wnn';
Wmn = B * Wnn;
Ast = Q * Wmn;
Wnn = Wnn - Ast;
Wnm = Wnn * Q;
Ast = Wnm * B;
Wnn = Wnn - Ast;

for i = 1 : n
    Wnn(i,i) = Wnn(i,i) - 1;
end

a = -a;

if strcmpi(solver, 'BICGSTAB')
    [a, ~] = bicgstab(Wnn, a);
else
    [a, ~] = gmres(Wnn, a);
end

case 'OTHER'
    un = a;
    sn = zeros(n, 1);

    for i = 1 : imax
        y = B * un;
        x = Q * y;
        un = un - x;

        if strcmpi(solver, 'BICGSTAB')
            [un, ~] = bicgstab(Ast, un);
        else
            [un, ~] = gmres(Ast, un);
        end

        un = Ass * un;
        x = power(-1, i) * un;
        sn = sn + x;
    end

y = B * sn;
x = Q * y;

```

```
    a = a + sn - x;
end

% Step 5:
vn = a - vn;
y  = B * vn;
a  = Q * y;
a  = vn - a;

if strcmpi(gmode, 'SLOW')
    if flag == 1
        a = linsolve(Ct, a, opts1);
        a = linsolve(C, a, opts2);
    else
        a = a(p,:);
        a = linsolve(L, a, opts1);
        a = D \ a;
        a = linsolve(Lt, a, opts2);
        a = a(p,:);
    end
else
    if strcmpi(solver, 'BICGSTAB')
        [a, ~] = bicgstab(Ast, a);
    else
        [a, ~] = gmres(Ast, a);
    end
end

x = a + wn;

% Step 6:
un = Ass * a;
a  = As * a;
a  = a + un;
vn = vn - a;
un = Ass * wn;
vn = vn - un;
y  = B * vn;
R  = R';
y  = linsolve(R, y, opts2);
end
```

Código C.12 – *Solver* final para resolução de problemas de ponto-de-sela

```

function [x, y, varargout] = algofinal(As, B, a, b, smode,gmode,del,imax)
%%
% Purpose:
%   algofinal solves a generalized saddle point system of the form
%   [As B'; B 0] [x; y] = [a; b],
%   where the symmetric part of the n-by-n matrix As is assumed to be
%   positive definite on ker(B) and the m-by-n matrix B is assumed to
%   have full rank. Given a certain n-by-n matrix Ast related with the
%   blocks As and B, the matrix E=(I-P)*Ass*inv(Ast)*(I-P) is also
%   assumed to do not have any eigenvalue -1.
%
% Format:
%   [x, y, scl] = algofinal(As, B, a, b, smode, gmode, del, imax)
%
% Examples:
%   [x, y]          = algofinal(As, B, a, b)
%   [x, y, scl]     = algofinal(As, B, a, b, 'scale')
%   [x, y]          = algofinal(As, B, a, b, 'nonscale', 'fast')
%   [x, y, scl]     = algofinal(As, B, a, b, 'scale', 'fast', 1e8)
%
% Input data:
%   As: The (1,1) block of a saddle point system.
%   B: The (2,1) block of a saddle point system.
%   a: The first n components of a saddle point system's rhs.
%   b: The last m components of a saddle point system's rhs.
%   smode: (optional) A string that informs to scale the given problem
%           or not. Possible values are: 'scale' or 'nonscale'. Default
%           value: 'nonscale'.
%   gmode: (Optional) A string that specifies the method to solve the
%           given problem. Possible values are: 'sym', 'fast', 'slow'
%           and 'other'. Default value: 'sym'.
%   del: (Optional) A nonnegative parameter to turn the (1,1) block
%         of 'preconditioned' saddle point systems into a more
%         diagonally dominant matrix. Default value: 1e16.
%   imax: (Optional) The number of terms minus one in the truncation
%         of the Neumann series of the matrix I+E. Default value: 0.
%
% Return data:
%   x: The first n components of a saddle point system's solution.
%   y: The last m components of a saddle point system's solution.
%   scl: If given the option smode='scale', then scl is a constant for
%         which (x,y) is the solution of the saddle point problem
%         [As (scl*B)'; scl*B 0] [x; y] = [a; scl*b].
%
% Remark:
%   The thin QR factorization of B' is computed using Matlab's qr

```

```
%  command. Ideally, we would like to employ Householder
%  transformations in this procedure, but we are not sure if Matlab
%  does this.

%%
% Initializing parameters.

warning('off', 'all');

if nargin < 4
    fprintf(2, '\nToo few input arguments.\n\n');
    return;
end

if nargin > 8
    fprintf(2, '\nToo many input arguments.\n\n');
    return;
end

switch nargin
    case 4
        smode = 'NONSCALE';
        gmode = 'SYM';
        del    = 1e16;
        imax   = 0;

    case 5
        gmode = 'SYM';
        del    = 1e16;
        imax   = 0;

    case 6
        del    = 1e16;
        imax   = 0;

    case 7
        imax = 0;
end

switch upper(smode)
    case 'NONSCALE'
        scl = 1;

    case 'SCALE'
        scl = meanabs(As) / meanabs(B);

    otherwise
        fprintf(2, '\nImproper smode option.\n\n');
```

```

        return;
    end

    switch upper(gmode)
        case 'SYM'
        case 'FAST'
        case 'SLOW'
        case 'OTHER'
            if nargin < 8
                fprintf(2, ['\nWith the ''other'' gmode option it is ' ...
                    'mandatory to specify the values of \nthe ' ...
                    'parameters del and imax.\n\n']);
                return;
            end

            otherwise
                fprintf(2, '\nImproper gmode option.\n\n');
                return;
            end

        if nargin < 2
            fprintf(2, '\nToo few output arguments.\n\n');
            return;
        end

        if nargin > 3
            fprintf(2, '\nToo many output arguments.\n\n');
            return;
        end

        [m, n] = size(B);

        if m >= n
            fprintf(2, ['\nImproper dimensions of the blocks of the saddle ' ...
                'point problem.\n\n']);
            return;
        end

        opts1.LT = true;
        opts2.UT = true;

        flag = 0;

        %%
        % Solving the saddle point problem given.

        % Step 0:
        if strcmpi(smode, 'SCALE')

```

```

    B = scl * B;
    b = scl * b;
end

if ~strcmpi(gmode, 'SYM')
    Ass = As';
    Ass = As - Ass;
    Ass = 0.5 * Ass;
    As = As - Ass;
end

% Step 1:
Wnm = B';
[Q, R] = qr(Wnm, 0);
B = Q';

% Step 2:
Wmn = B * As;
Wnn = Q * Wmn;
Wnn = As - Wnn;
Wnm = Wnn * Q;
Ast = Wnm * B;
Ast = Wnn - Ast;

% Step 3:
gamst = normest(Ast);
Wnm = gamst * Q;
Wnn = Wnm * B;
Ast = Ast + Wnn;

if del > 2*eps
    del = del * gamst;

    for i = 1 : n
        Ast(i,i) = Ast(i,i) + del;
    end
end

% Step 4a:
R = R';
y = linsolve(R, b, opts1);
wn = Q * y;
vn = As * wn;

% Step 4b:
if ~strcmpi(gmode, 'SYM')
    y = B * vn;
    un = Q * y;
    un = vn - un;
end

```

```

if strcmpi(gmode, 'SLOW')
    try
        C = chol(Ast);
        Ct = C';

        un = linsolve(Ct, un, opts1);
        un = linsolve(C, un, opts2);

        flag = 1;
    catch
        [L, D, p] = ld1(Ast, 'vector');
        Lt = L';

        un = un(p,:);
        un = linsolve(L, un, opts1);
        un = D \ un;
        un = linsolve(Lt, un, opts2);
        un = un(p,:);
    end
else
    [un, ~] = gmres(Ast, un);
end

un = wn - un;
un = Ass * un;
y = B * un;
x = Q * y;
un = un - x;
a = a - un;
end

% Step 4c:
switch upper(gmode)
case 'SLOW'
    if flag == 1
        Wnn = linsolve(Ct, Ass, opts1);
        Wnn = linsolve(C, Wnn, opts2);
    else
        Wnn = Ass(:, p);
        Wnn = linsolve(L, Wnn, opts1);
        Wnn = D \ Wnn;
        Wnn = linsolve(Lt, Wnn, opts2);
        Wnn = Wnn(:, p);
    end

    Wnn = Wnn';
    Wmn = B * Wnn;
    Ast = Q * Wmn;
    Wnn = Wnn - Ast;

```



```

Wnm = Wnn * Q;
Ast = Wnm * B;
Wnn = Wnn - Ast;

for i = 1 : n
    Wnn(i,i) = Wnn(i,i) - 1;
end

a      = -a;
[a, ~] = gmres(Wnn, a);

case 'OTHER'
    un = a;
    sn = zeros(n, 1);

    for i = 1 : imax
        y      = B * un;
        x      = Q * y;
        un     = un - x;
        [un, ~] = gmres(Ast, un);
        un     = Ass * un;
        x      = power(-1, i) * un;
        sn     = sn + x;
    end

    y = B * sn;
    x = Q * y;
    a = a + sn - x;
end

% Step 5:
vn = a - vn;
y  = B * vn;
a  = Q * y;
a  = vn - a;

if strcmpi(gmode, 'SLOW')
    if flag == 1
        a = linsolve(Ct, a, opts1);
        a = linsolve(C, a, opts2);
    else
        a = a(p,:);
        a = linsolve(L, a, opts1);
        a = D \ a;
        a = linsolve(Lt, a, opts2);
        a = a(p,:);
    end
else
    [a, ~] = gmres(Ast, a);
end

```

```
end

x = a + wn;

% Step 6:
if strcmpi(gmode, 'SYM')
    a = As * a;
else
    un = Ass * a;
    a = As * a;
    a = a + un;
end

vn = vn - a;

if ~strcmpi(gmode, 'SYM')
    un = Ass * wn;
    vn = vn - un;
end

y = B * vn;
R = R';
y = linsolve(R, y, opts2);

if nargout > 2
    varargout{1} = scl;
end
end
```