# Aplicações, métodos e ferramentas para programação semi-infinita não linear

António Ismael de Freitas Vaz

Universidade do Minho
Braga, Fevereiro 2003

# Aplicações, métodos e ferramentas para programação semi-infinita não linear

**António Ismael de Freitas Vaz**

Dissertação submetida à Universidade do Minho
para obtenção do grau de doutor
no Ramo de Engenharia de Produção e Sistemas
Área de Métodos Numéricos e Estatísticos

**Universidade do Minho**
**Braga, Fevereiro de 2003**

# Resumo

Esta tese dedica-se à modelação e resolução de problemas de programação semi-infinita (PSI) não linear. Estes problemas são caracterizados por possuírem um número finito de variáveis e um conjunto infinito de restrições.

A partir de uma compilação de problemas existentes na literatura da especialidade e com base na linguagem de modelação AMPL foi construída uma base de dados que contém actualmente cento e quarenta e quatro problemas. Adicionalmente foram integrados no pacote de software designado por SIPAMPL, uma interface que permite a ligação de um software de resolução à base de dados e um conjunto de ferramentas.

Problemas de optimização de trajectórias de robôs e de desenho óptimo de conjuntos de sinais foram formulados como problemas de PSI e resolvidos por um método de discretização. Foi desenvolvida uma extensão da técnica de programação quadrática sequencial para a resolução de problemas de PSI, a qual se baseia numa parametrização linear das variáveis duais para obter a solução do problema quadrático dual. Através de uma estratégia de transcrição das restrições infinitas em restrições finitas que envolvem o uso de integrais, foram ainda desenvolvidos um método de penalidade e um método primal-dual de pontos interiores não admissível que requerem, pela natureza do problema finito definido, a utilização de fórmulas numéricas adaptativas para o cálculo de integrais.

Para a implementação destes métodos e aproveitando as potencialidades do pacote SIPAMPL, foi criado o software NSIPS para a resolução de problemas de PSI. Este software permite que o utilizador altere de uma forma simples e eficaz os parâmetros dos métodos implementados, afinando-os para os problemas que pretende resolver.

Os pacotes de software SIPAMPL e NSIPS encontram-se disponíveis ao público em geral e em conjunto suportam a PSI desde a modelação de um problema até a sua resolução.

# Abstract

This thesis is devoted to modeling and solving nonlinear semi-infinite programming (SIP) problems. These problems have a finite number of variables and an infinite set of constraints.

By collecting a set of problems from the SIP literature and using the AMPL modeling language, a database of problems was developed containing at present one hundred and forty four problems. Additionally an interface that connects a solver to the database and a set of tools were integrated in the software package called SIPAMPL.

Robot trajectory planning and optimal signals set problems were formulated as SIP problems and solved by a discretization method. An extension of the sequential quadratic programming technique for solving SIP problems was developed which is based on a linear parametrization of the dual variables to obtain the solution of the quadratic dual problem. Using a transcription of the infinite constraints into finite constraints involving integrals, we also developed a penalty method and an infeasible primal-dual interior point method that require, due to the defined finite problem, the use of adaptative numerical formulae for integral computations.

To implement these methods and using the SIPAMPL facilities, a package called NSIPS was created for solving SIP problems. This software allows parameter modifications in a simple and powerful way to suit problems request.

The software packages SIPAMPL and NSIPS are freely available and together they support SIP from problem modeling to its resolution.

# Agradecimentos

Os primeiros e maiores agradecimentos são para a orientadora principal deste projecto, Doutora Edite Fernandes, pela sua incansável disponibilidade e trabalho. À orientadora Doutora Paula Gomes, embora fisicamente mais afastada, o meu agradecimento por ter sempre manifestado o seu empenho e disponibilidade.

Para os colegas do subgrupo de Métodos Numéricos e Estatísticos o meu agradecimento pelo apoio sempre manifestado.

Agradeço à Universidade do Minho pelas facilidades concedidas, nomeadamente em relação à equiparação a bolseiro.

É de referir que este trabalho foi suportado pelo Programa PRODEP, Concurso Nº4/5.3/-PRODEP/2000, PRODEP III.

Agradecimentos a toda a minha família que permitiu a estabilidade necessária para a execução deste trabalho.

# Dedicatória

**À minha esposa Mónica e aos meus filhos Gabriela e Henrique**

# Conteúdo

# Lista de Tabelas

# Lista de Figuras

# Capítulo 1

# Introdução

---

Neste capítulo faz-se uma breve introdução à programação semi-infinita (PSI). Problemas de programação semi-infinita aparecem em várias áreas da engenharia, como por exemplo, no controlo da poluição [58], planeamento da produção [75, 161], teoria da aproximação de Chebyshev [55, 57, 116], cálculo de trajectórias de robôs [52, 82, 137, 153], desenho óptimo de conjuntos de sinais [42, 68, 156] e desenhos de filtros digitais [46, 110, 106, 107, 108, 109, 120]. Uma breve referência a alguns destes problemas pode ser encontrada em [58].

---

A primeira secção apresenta uma breve descrição do problema de programação semi-infinita. A Secção 1.2 apresenta a motivação do trabalho desenvolvido e na última secção é apresentada a estrutura da tese.

## 1.1 Descrição do problema

Assim como em programação finita, a programação semi-infinita também pode ser classificada de acordo com as características matemáticas das funções envolvidas no problema. Os problemas aqui abordados são de programação semi-infinita não linear, embora se faça referência, quando necessário, à PSI linear e quadrática. Os algoritmos desenvolvidos são pois indicados para problemas de PSI não linear.

O problema de programação semi-infinita pode ser descrito da seguinte forma

$$\begin{aligned} & \min_{x \in R^n} f(x) \\ \text{s.a} \ & g_i(x,t) \leq 0 \quad i = 1, \ldots, m \\ & h_i(x) = 0 \quad i = 1, \ldots, o \\ & h_i(x) \leq 0 \quad i = o+1, \ldots, q \\ & \forall t \in T, \end{aligned} \tag{1.1.1}$$

onde $T$ é um conjunto infinito, usualmente um produto cartesiano de intervalos de limites finitos ($T = [\alpha_1, \beta_1] \times [\alpha_2, \beta_2] \times \cdots \times [\alpha_p, \beta_p]$), e um subconjunto de $R^p$.

A função $f(x)$ denota a função objectivo, $g_i(x,t) \leq 0$, $i = 1, \ldots, m$, são as restrições do tipo infinito (designadas de restrições infinitas), e $h_i(x) = 0$, $i = 1, \ldots, o$, $h_i(x) \leq 0$, $i = o+1, \ldots, q$, são respectivamente as restrições de igualdade e desigualdade finitas. Num problema de PSI tem-se $m \geq 1$, existindo pelo menos uma restrição infinita, e $o \geq 0$, $q \geq o$, podendo não existir nenhuma restrição finita. Em PSI não linear supõe-se que pelo menos uma das funções $f$, $g_i$, $i = 1, \ldots, m$, e $h_i$, $i = 1, \ldots, q$, é não linear em $x$.

A designação de programação semi-infinita deriva do facto de que cada restrição $g_i(x,t) \leq 0$ do problema (1.1.1) poder ser vista como uma restrição indexada por cada valor de $t \in T$, com $T$ infinito, resultando num problema com um número finito de variáveis sujeito a um conjunto infinito de restrições.

Quando o conjunto $T$ é função das variáveis ($T := T(x)$) o problema pertence à classe da PSI generalizada ([47, 73, 132, 134, 133, 130, 131, 122, 135, 165]). Se, pelo contrário, o conjunto $T$ não é função das variáveis o problema pertence à classe da PSI padrão (*standard*). Este trabalho refere-se exclusivamente à PSI padrão, fazendo-se pontualmente referência à PSI generalizada.

A programação semi-infinita paramétrica ([56, 65, 124, 171]) resulta da dependência das funções $f$ e $g$ de mais uma variável que é vista como um parâmetro. A substituição do parâmetro por um valor fixo resulta num problema de PSI.

O problema de programação semi-infinita descrito em (1.1.1) não se encontra na sua forma mais geral, uma vez que o problema pode conter ainda restrições do tipo limites simples. No entanto, estas podem ser manipuladas de forma a gerar restrições do tipo já apresentado.

Para mostrar a dificuldade de resolução de problemas de PSI basta lembrar que a determinação do máximo global de uma função não linear

$$\max_{t \in T} g(t) \tag{1.1.2}$$

pode ser equacionada como um problema de PSI linear

$$\begin{aligned} &\min_{d \in R} d \\ \text{s.a} \quad &g(t) - d \leq 0 \\ &\forall t \in T. \end{aligned} \tag{1.1.3}$$

## 1.2 Motivação

A existência de problemas teste, académicos ou reais, é fundamental para o desenvolvimento de software numérico, não só para demonstrar a sua capacidade de resolução mas também para testar a sua eficiência.

Em programação matemática finita, linear e não linear, existem bases de dados com problemas, reais e académicos, disponíveis ao público. O pacote CUTE [11] (*Constrained and Unconstrained Test Environment*) possui uma dessas bases de dados com inúmeros problemas teste para programação finita. Este ambiente possui ainda uma interface que

permite a interligação dessa base de dados com software já comercializado (NPSOL [40], LANCELOT [21]) e com outros códigos que eventualmente possam vir a ser desenvolvidos para a resolução de problemas de programação finita. O pacote CUTE, embora seja o mais conhecido, não é o único do seu género. O AMPL (*A Modeling Programming Language*) [36] e o GAMS (*General Algebraic Modeling System*) [14] são dois exemplos que têm vindo a ser utilizados por investigadores da área da programação matemática.

Para a PSI não existia, no início deste trabalho, nenhuma base de dados com problemas que estivesse disponível ao público. Com o objectivo de facilitar o desenvolvimento de software numérico para a PSI foi construída uma base de dados que contém, neste momento 144 problemas retirados da literatura. Usou-se o software AMPL, que fornece uma linguagem de modelação para problemas de programação matemática, para a codificação destes problemas. Além da base de dados, foi construída uma interface que permite a ligação da base de dados ao software de resolução de problemas de PSI (*solver*). Foram também desenvolvidos o software que permite consultar a base de dados (ferramenta *select*) e uma outra interface para a ligação da base de dados ao MATLAB. A inclusão de problemas de robótica na base de dados levou ao desenvolvimento de uma biblioteca dinâmica de "B-Splines" para o AMPL. Este pacote de software está disponível ao público no endereço *http://www.norg.uminho.pt/aivaz/* e foi designado por SIPAMPL.

Embora se conheçam vários algoritmos para a resolução de problemas de PSI, não existia nenhum software comercial ou livre que permitisse resolver o tipo de problemas tratados nesta tese. Além disso, uma grande parte desses algoritmos foi desenvolvida para a PSI linear e quadrática e não pode ser directamente aplicável à PSI não linear. Assim, com base nas técnicas mais conhecidas para programação não linear finita, foram desenvolvidos vários métodos para a PSI não linear, dos quais se destacam, um método de programação quadrática sequencial que resolve o problema quadrático dual através de uma parametrização linear das variáveis duais, um método de penalidade e um método primal-dual de pontos interiores não admissível que recorrem à transcrição das restrições infinitas do problema semi-infinito em restrições finitas que envolvem o uso de integrais. A resolução dos integrais foi facilmente ultrapassada pela implementação de uma fórmula numérica adaptativa.

Em complemento ao software SIPAMPL, e apoiado por este, foram implementados todos os algoritmos desenvolvidos neste trabalho para a programação semi-infinita não linear e criado o software de resolução NSIPS.

## 1.3 Estrutura da tese

Esta tese é composta por dois suportes diferentes, um em papel e outro digital (CDROM).

O suporte digital em CDROM é apresentado com mais detalhe no Apêndice A. Com este tipo de suporte é possível adicionar à tese material que de outra forma seria mais fastidioso. O CDROM foi usado para incluir o software desenvolvido, tanto em formato de código fonte como em binário, permitindo ao leitor uma completa apreciação do trabalho desenvolvido. Evita-se assim a inclusão de uma enorme quantidade de papel que seria

necessário para apresentar de forma exaustiva o trabalho realizado.

No próximo capítulo faz-se uma breve introdução à optimização finita não linear com e sem restrições e no Capítulo 3 descrevem-se as condições de optimalidade de um problema de PSI. Estas condições são essenciais para mostrar o aspecto das soluções dos problemas de PSI que nos propomos abordar e são uma motivação para as abordagem clássicas descritas no Capítulo 4.

A recolha bibliográfica permitiu obter vários problemas, quer académicos, quer da vida real e que foram usados para construir uma base de dados de problemas de PSI. A existência desta base de dados levou à criação de uma interface para a sua ligação a um futuro software de resolução de problemas de PSI. Foi usada a linguagem de modelação matemática AMPL [36] para a construção deste software, designado de SIPAMPL. No Capítulo 5 descreve-se em detalhe o pacote SIPAMPL.

O Capítulo 6 descreve um algoritmo baseado numa modificação do método de discretização para a resolução de problemas de PSI não linear.

Os problemas práticos comprovam a utilidade da teoria e software desenvolvidos. O Capítulo 7 apresenta dois casos da aplicação da PSI a casos reais de optimização de trajectórias de robôs e ao desenho óptimo de conjuntos de sinais.

Foram ainda desenvolvidos vários métodos que permitem resolver quase todos os problemas codificados. A teoria que está subjacente aos métodos desenvolvidos e os correspondentes algoritmos são detalhadamente descritos no Capítulo 8.

O Capítulo 9 apresenta o software que inclui os métodos implementados. O NSIPS (*Nonlinear Semi-infinite Programming Solver*) permite resolver os problemas codificados em AMPL através do uso da interface do SIPAMPL. O NSIPS é o único software (do nosso conhecimento) que resolve problemas de PSI de uma forma modular e é distribuído ao público no formato de código fonte e binário.

Os resultados numéricos obtidos com os métodos implementados são apresentados no Capítulo 10. Sempre que possível é feita uma comparação com resultados numéricos fornecidos por outros autores.

O Capítulo 11 apresenta as conclusões e aponta futuras direcções de trabalho.

Para os apêndices são relegados os assuntos que auxiliam as matérias abordadas nos capítulos anteriores e são de leitura mais ou menos facultativa. São apresentadas sessões interactivas com o software desenvolvido.

# Capítulo 2

# Optimização finita

---

A programação semi-infinita é uma generalização da programação finita, em que o número de restrições deixa de ser finito para ser infinito. Neste capítulo faz-se uma breve referência aos métodos mais usados em optimização finita sem e com restrições. Estes métodos são de vital importância para a PSI, uma vez que as técnicas usadas em PSI são frequentemente extensões das técnicas usadas em programação finita.

---

Na Secção 2.1 faz-se uma introdução aos dois métodos mais usados para problemas sem restrições. O método de Newton exige o cálculo analítico ou numérico das segundas derivadas da função objectivo e o método quasi-Newton necessita apenas das primeiras derivadas para construir uma aproximação à matriz das segundas derivadas. Na Secção 2.2 descrevem-se três técnicas que são geralmente usadas em optimização com restrições e que foram alvo de extensão para a PSI. As referências [6, 96, 99] contêm mais detalhes sobre este assunto.

## 2.1 Optimização sem restrições

Na optimização sem restrições procura-se determinar a solução do problema

$$\min_{x \in R^n} f(x) \tag{2.1.1}$$

em que $f$ é a função objectivo. As propriedades da função $f$ determinam muitas vezes o tipo de algoritmo que se pode aplicar. Se $f$ for uma função não diferenciável deve usar-se um algoritmo que não necessite do cálculo das derivadas, como por exemplo o método de Powell [167], método simplex de Nelder-Mead [97], método de procura em padrão (*Pattern search*) [74, 143, 144] ou algoritmos genéticos [93]. Se $f$ for duas vezes continuamente diferenciável, o método mais indicado é o método de Newton que possui convergência quadrática local. Se o cálculo das segundas derivadas não for possível, ou for demasiado penoso, pode usar-se uma variante quasi-Newton que consiste na aproximação

da matriz das segundas derivadas por informação de primeira ordem, com prejuízo para a convergência, que passa a ser superlinear. Estas duas abordagens serão descritas com mais detalhe nas subsecções seguintes.

### 2.1.1 Método de Newton

O método de Newton na sua forma mais simples gera uma sucessão de pontos

$$x_{k+1} = x_k + d_{k,N},$$

em que $x_k$ representa o valor de $x$ na iteração $k$, e $d_{k,N}$ é a direcção Newton obtida da resolução do sistema

$$\nabla^2 f(x_k) d_{k,N} = -\nabla f(x_k). \tag{2.1.2}$$

$\nabla f(x)$ é o vector gradiente e $\nabla^2 f(x)$ é a matriz Hessiana de $f(x)$.

Seja $x^*$ uma solução de (2.1.1). Dado um valor inicial $x_0$, suficientemente próximo de $x^*$, o método de Newton converge com razão q-quadrática para um ponto estacionário de $f$.

A implementação do método de Newton possui algumas limitações, nomeadamente quando a aproximação inicial está longe de um ponto estacionário ou a matriz Hessiana é singular.

O problema da aproximação inicial estar longe da solução pode ser resolvido usando uma técnica de procura unidimensional ou através do uso de regiões de confiança. O uso destas técnicas obriga à introdução de uma função mérito que mede o progresso da iteração corrente para a iteração seguinte. A obrigatoriedade do uso de funções mérito foi recentemente ultrapassada através do aparecimento dos métodos dos filtros [32, 33, 34, 146, 163].

Quando a matriz Hessiana, em $x^*$, é definida positiva está-se na presença de um minimizante local do problema (2.1.1). Quando a matriz em (2.1.2) não é definida positiva, o método de Newton pode convergir para pontos estacionários que não sejam minimizantes. Uma das estratégias usadas para evitar este tipo de comportamento consiste em somar um múltiplo da matriz identidade à matriz Hessiana por forma a torná-la definida positiva. Outra estratégia consiste em usar uma factorização de Cholesky modificada para resolver o sistema (2.1.2), que modifica os elementos dos factores por forma a que a direcção calculada seja equivalente à direcção obtida com a Hessiana modificada.

### 2.1.2 Método quasi-Newton

Uma das desvantagens do método de Newton está relacionada com a necessidade de calcular a matriz Hessiana. Embora actualmente exista software de derivação automática (por exemplo CUTE [11], AMPL [36] e GAMS [14]) que fornece as derivadas da matriz Hessiana, sem intervenção do utilizador, esta pode ser de tal dimensão que os recursos informáticos se tornem proibitivos.

A impossibilidade de calcular a matriz Hessiana ou o seu peso de cálculo levaram ao aparecimento do método quasi-Newton. Este método tem como base a construção de uma aproximação à matriz Hessiana, que requer apenas informação relativa às primeiras derivadas de $f$. A imposição de que a matriz aproximação deva ser definida positiva ao longo do processo iterativo e a verificação da condição secante deram origem a várias fórmulas de actualização, das quais a BFGS (proposta por Broyden, Fletcher, Goldfarb e Shanno) é a mais popular.

Existem também variantes do método quasi-Newton que geram aproximações à inversa da matriz Hessiana, permitindo que se faça apenas um produto matriz vector para o cálculo da direcção, em vez de se resolver o sistema (2.1.2).

Existem também variantes de memória limitada que permitem que a aproximação à matriz Hessiana seja feita com base em alguns vectores. Estes métodos permitem uma redução da memória usada para guardar a matriz aproximação e são pois próprios para a resolução de problemas de grandes dimensões.

## 2.2 Optimização com restrições

Na optimização com restrições procura-se determinar a solução do problema

$$\begin{aligned} &\min_{x \in R^n} f(x) \\ s.a\ \ &h_i(x) = 0, i \in I \\ &h_d(x) \leq 0, d \in D \end{aligned} \tag{2.2.1}$$

em que $f$ é a função objectivo, $h_l$, $l \in I \cup D$, são as funções das restrições em que $I$ é o conjunto dos índices das restrições de igualdade e $D$ é o conjunto dos índices das restrições de desigualdade.

Uma das técnicas usadas para resolver problemas da forma (2.2.1) consiste na substituição do problema com restrições por uma sequência de problemas sem restrições.

A outra técnica também muito usada baseia-se na substituição do problema (2.2.1) por uma sequência de problemas com restrições, mas de resolução mais fácil. Nestas técnicas está incluída a programação quadrática sequencial em que o problema (2.2.1) é localmente aproximado por um problema quadrático.

A técnica baseada em pontos interiores consiste em aplicar o método de Newton ao sistema das condições de Karush-Kuhn-Tucker (KKT) perturbado.

### 2.2.1 Técnicas de penalidade

As técnicas de penalidade têm como base a definição de um problema sem restrições, cuja função objectivo é obtida adicionando à função objectivo do problema original termos que penalizam a violação das restrições. A técnica consiste na substituição do problema (2.2.1) por uma sequência de problemas da forma

$$\min_{x \in R^n} \phi(x, \mu) \equiv f(x) + \mu F_1(h_i(x)) + \mu F_2(h_d(x)), \quad i \in I, \quad d \in D \tag{2.2.2}$$

parametrizados por $\mu$, em que $\mu > 0$ é o parâmetro de penalidade. A função de penalidade é pois uma função que depende da função objectivo e das restrições do problema original. As funções $F_1$ e $F_2$ são em geral nulas para pontos na região admissível do problema original e positivas fora dela.

As funções de penalidade podem ser divididas em funções de penalidade exteriores e interiores. As exteriores permitem que as sucessivas aproximações à solução estejam fora da região admissível, enquanto que nas interiores as sucessivas aproximações têm de pertencer à região admissível.

Um dos problemas que surge da implementação da técnica de penalidade está relacionado com a instabilidade de algumas funções de penalidade, uma vez que as soluções da sequência de problemas sem restrições, $x^*_\mu$, convergem para a solução do problema original, quando o parâmetro de penalidade tende para infinito. Nesta situação a matriz Hessiana da função de penalidade torna-se mal condicionada, e a resolução do problema (2.2.2) é cada vez mais difícil.

Este inconveniente é facilmente ultrapassado utilizando funções de penalidade exactas.

**Definição 2.2.1** *Uma função de penalidade diz-se exacta se existe $\overline{\mu} > 0$ tal que a solução de* (2.2.2)*, $x^*_\mu$, é solução de* (2.2.1)*, $x^*$, para todo o $\mu > \overline{\mu}$.*

Neste caso, se for possível calcular $\overline{\mu}$, basta resolver uma vez o problema (2.2.2) para se conhecer a solução do problema original. Como o valor de $\overline{\mu}$ depende da solução que não é conhecida, na prática resolve-se uma sequência de problemas sem restrições, ajustando sucessivamente as estimativas de $\overline{\mu}$.

Algumas funções de penalidade também incluem termos que envolvem os multiplicadores de Lagrange, sendo, nestes casos, necessário obter estimativas do vector dos multiplicadores óptimo. Na literatura, estes métodos são conhecidos por métodos dos multiplicadores.

### 2.2.2 Programação quadrática sequencial

Na programação quadrática sequencial resolve-se uma sequência de problemas quadráticos para obter uma sequência de direcções de procura que geram as aproximações à solução do problema (2.2.1). Os problemas quadráticos são aproximações ao problema original em que a função objectivo é uma aproximação quadrática à função $f$ e as restrições são aproximações lineares às funções $h_l$, $l \in I \cup D$.

Através da resolução do problema quadrático obtém-se uma direcção de procura para gerar aproximações às variáveis primais e duais.

Um dos problemas deste tipo de técnica reside no facto do problema quadrático poder não possuir solução admissível, mesmo que o problema original possua.

A aproximação quadrática à função $f$ é, em geral, construída com base na Hessiana da função Lagrangeana associada ao problema original. Para ultrapassar o inconveniente que resulta do cálculo das segundas derivadas das funções envolvidas no problema, pode implementar-se uma estratégia quasi-Newton para obter aproximações à Hessiana da Lagrangeana.

### 2.2.3 Pontos interiores

O método de pontos interiores resulta da resolução do sistema das condições de optimalidade de primeira ordem do problema barreira que está associado ao problema original com restrições.

Os métodos de pontos interiores podem ser divididos em admissíveis e não admissíveis.

Nos métodos admissíveis, o processo requer uma aproximação inicial estritamente admissível e gera sucessivas aproximações estritamente admissíveis para as restrições do problema original. Este procedimento é conseguido através da reformulação do problema (2.2.1) como um problema barreira em que as restrições de desigualdade são colocadas como argumentos de uma função barreira. O problema é então reformulado como

$$\begin{aligned} &\min_{x\in R^n} f(x) - \mu_b \sum_{d\in D} \mathcal{B}(h_d(x)) \\ s.a\ \ &h_i(x) = 0, i \in I \end{aligned} \tag{2.2.3}$$

em que $\mu_b$ é o parâmetro barreira e $\mathcal{B}$ é a função barreira (veja-se por exemplo [35, 38, 141]). Entre as propriedades desejadas para a função barreira está a da auto-concordância (veja-se em [98]).

No caso dos métodos de pontos interiores não admissíveis são adicionadas variáveis de folga não negativas às restrições de desigualdade e são essas que são incluídas na função barreira. O problema barreira associado ao problema (2.2.1) é então

$$\begin{aligned} &\min_{x\in R^n, s} f(x) - \mu_b \sum_{d\in D} \mathcal{B}(s_d) \\ s.a\ \ &h_i(x) = 0, i \in I \\ &h_d(x) + s_d = 0, d \in D \end{aligned} \tag{2.2.4}$$

em que $s$ é o vector das variáveis de folga, $\mu_b$ é o parâmetro barreira e $\mathcal{B}$ é a função barreira. Lista-se de seguida um conjunto de referências que abordam este tipo de métodos [15, 16, 24, 39, 86, 87, 88, 91, 123, 147, 149]. Os trabalhos [26, 114, 115, 170] consideram ainda aspectos teóricos desta abordagem.

A resolução do sistema KKT de primeira ordem do problema barreira requer técnicas especiais de resolução, especialmente quando este é de grandes dimensões, que consideram a utilização do método de Newton na resolução do sistema KKT e a implementação de uma factorização Cholesky modificada ao sistema Newton [89, 92, 168] combinada com técnicas para lidar com matrizes esparsas.

Quando a estratégia de globalização do algoritmo é a procura unidimensional e o problema é linear, usam-se dois comprimentos de passos distintos para as variáveis primais e duais. No caso não linear, usa-se o mesmo comprimento de passo para ambas as variáveis. Em [90] o autor descreve um esquema em que se pode usar passos diferentes para as variáveis primais e duais no caso quadrático.

A sequência de soluções $x^*_{\mu_b}$ dos diferentes problemas barreira, parametrizados por $\mu_b$, define um caminho denominado de caminho central. A função barreira força as sucessivas

aproximações à solução a permaneçam o mais perto possível do caminho central, de modo a que a solução óptima do problema original seja obtida quando $\mu_b \approx 0$.

# Capítulo 3

# Condições de optimalidade

---

As condições de optimalidade são de extrema importância para todos os problemas de programação matemática, uma vez que caracterizam as soluções do problema. A dedução das condições de optimalidade indicam também possíveis caminhos na resolução do problema apresentado.

---

A próxima secção apresenta uma simplificação do problema mais geral apresentado em (1.1.1). Este problema mais simples permite-nos descrever com mais simplicidade as condições de optimalidade de primeira e segunda ordem, sendo directa a extrapolação para o problema geral. As condições de optimalidade de primeira ordem são apresentadas na Secção 3.2. Como as condições de segunda ordem são discutidas com base na redução local do problema de PSI a um problema finito, descreve-se na Secção 3.3 a respectiva redução local e na Secção 3.4 as condições de optimalidade de segunda ordem.

## 3.1 Simplificação do problema

O problema de programação semi-infinita descrito em (1.1.1) encontra-se na sua forma geral, uma vez que o problema de PSI contém várias ($m$) restrições do tipo infinito e pode conter restrições finitas de igualdade e desigualdade. A simplificação do problema geral de PSI não reduz o grau de complexidade de resolução, pelo que a seguinte descrição mais simples será preferencialmente usada,

$$\begin{aligned} &\min_{x \in R^n} \ f(x) \\ s.a \ \ &g(x,t) \leq 0 \\ &\forall t \in T \ , \end{aligned} \tag{3.1.1}$$

sendo directa a dedução das condições de optimalidade para o caso geral.

## 3.2 Condições de optimalidade de primeira ordem

A seguinte definição caracteriza um ponto estacionário do problema de PSI [111, 113]. Assume-se que $f$ e $g$ são duas vezes continuamente diferenciáveis, em relação a $x$ e $t$, respectivamente em $R^n$ e $R^n \times R^p$, e $T$ é um conjunto compacto.

**Definição 3.2.1** *Seja $x^* \in R^n$ um ponto tal que*

$$g(x^*, t) \leq 0, \ \forall t \in T,$$

*e suponha que existem $t_1, t_2, \ldots, t_{m^*} \in T$ e números não negativos $\lambda_0^*, \lambda_1^*, \lambda_2^*, \ldots, \lambda_{m^*}^*$ tais que*

$$\lambda_0^* \nabla_x f(x^*) + \sum_{i=1}^{m^*} \lambda_i^* \nabla_x g(x^*, t_i) = 0 \tag{3.2.1}$$

*com*

$$g(x^*, t_i) = 0, \quad i = 1, \ldots, m^*. \tag{3.2.2}$$

*Então $x^*$ é um ponto estacionário para o problema de PSI* (3.1.1).

**Hipótese 3.2.1** *Suponha-se que a seguinte condição de regularidade é válida*

$$\exists u \in R^n \ \text{ tal que } \ g(x^*, t) + u^T \nabla_x g(x^*, t) < 0, \forall t \in T. \tag{3.2.3}$$

Um ponto que satisfaz a Definição 3.2.1 e a Hipótese 3.2.1 chama-se ponto Karush-Kuhn-Tucker. De facto, a Hipótese 3.2.1 não é a única condição (*constraint qualification*) sobre a qual se podem deduzir as condições de optimalidade de primeira ordem. Em [111] é apresentada uma dedução das condições de optimalidade de primeira ordem baseada em integrais de Lebesgue. Em [51] é proposta uma dedução baseada no integral de Riemann.

A seguinte hipótese é imposta para que o problema de PSI seja tratável.

**Hipótese 3.2.2** *Para todo o $x \in R^n$ o conjunto dos máximos globais ($\Gamma(x)$) de $g(x, t)$ para todo o $t \in T$ é finito.*

As condições de optimalidade (3.2.1) e (3.2.2) e a não negatividade dos multiplicadores de Lagrange são de facto as condições de optimalidade de primeira ordem do seguinte problema:

$$\min_{x \in R^n} f(x) \ \text{ s.a } \ g(x, t_i) \leq 0 \quad \text{para todo o } i \text{ do conjunto } \{1, \ldots, m^*\}.$$

## 3.3 Redução local a um problema finito

Considere o seguinte problema definido para cada $\overline{x} \in R^n$:

$$\max_{t \in T} g(\overline{x}, t). \tag{3.3.1}$$

**Definição 3.3.1** *Sejam $\overline{t}_i \in \Gamma(\overline{x})$, $i = 1, \ldots, \overline{m}$ as soluções óptimas do problema* (3.3.1) *para um dado $\overline{x} \in R^n$.*

A redução local consiste na substituição das restrições infinitas no problema de PSI por um número finito de restrições considerando os máximos globais de (3.3.1) como funções dependentes de $x$. A seguinte hipótese é necessária para que o problema possa ser reduzido.

**Hipótese 3.3.1** *Em conjunto com a Hipótese 3.2.2, existem vizinhanças $U_{\overline{x}}$ de $\overline{x}$, e $U_{\overline{t}_i}$ de $\overline{t}_i$, $i = 1, \ldots, \overline{m}$, e funções contínuas*

$$t_i : U_{\overline{x}} \to U_{\overline{t}_i} \cap T$$

*tais que*

- $t_i(\overline{x}) = \overline{t}_i$;
- $t_i(\overline{x})$ *é a única solução local para o problema* (3.3.1) *em $U_{\overline{t}_i} \cap T$, para $i = 1, \ldots, \overline{m}$ e para todo o $x \in U_{\overline{x}}$.*

Se a Hipótese 3.3.1 for verificada em $\overline{x}$, então o problema de PSI pode ser localmente aproximado, na vizinhança $U_{\overline{x}}$, por um problema reduzido finito.

**Lema 3.3.1** *(Lema 4.8 em [58]) Seja válida a Hipótese 3.3.1. Seja dado $\overline{x} \in R^n$, então existe uma vizinhança $U_{\overline{x}}$ de $\overline{x}$ tal que para todo o $x \in U_{\overline{x}}$ existe um ponto admissível $x \in R^n$ se e só se*

$$G_i(x) := g(x, t_i(x)) \leq 0, \quad i = 1, \ldots, \overline{m}.$$

O problema reduzido é então definido da seguinte forma

$$\begin{aligned} &\min_{x \in U_{\overline{x}}} \ f(x) \\ &s.a \ \ G_i(x) \leq 0, \quad i = 1, \ldots, \overline{m}. \end{aligned} \tag{3.3.2}$$

O estudo que se segue é restrito aos problemas de PSI que verificam as duas hipóteses seguintes.

**Hipótese 3.3.2** *Suponha que o conjunto compacto $T \subseteq R^p$ é definido por*

$$T = \{t | h_j(t) \leq 0, j = 1, \ldots, J\} \tag{3.3.3}$$

*onde $h_j(t)$ são funções duas vezes continuamente diferenciáveis em $R^p$. Além disso, em cada ponto de $T$ a condição (*constraint qualification*) de independência linear é válida, i.e., os vectores $\nabla_t h_k(t)$, $k \in \{j | h_j(t) = 0, j = 1, \ldots, J\}$ são linearmente independentes.*

O seguinte resultado é então imediato.

**Lema 3.3.2** *(Lema 4.12 em [58]) Seja válida a Hipótese 3.3.2. Para $\overline{x} \in R^n$ e para todo o $\overline{t}_i$, $i = 1, \ldots, \overline{m}$ sejam*

$$\mathcal{M}^i = \{j | h_j(\overline{t}_i) = 0, j = 1, \ldots, J\}$$

*e a função Lagrangeana associada ao problema* (3.3.1) *com respeito a $\overline{t}_i$:*

$$\mathcal{L}^i(t, \alpha^i) := g(\overline{x}, t) - \sum_{j \in \mathcal{M}^i} \alpha^i_j h_j(t).$$

*Então existem multiplicadores únicos $\overline{\alpha}^i_j \geq 0$ tais que*

$$\nabla_t \mathcal{L}^i(\overline{t}_i, \overline{\alpha}^i) = 0.$$

**Hipótese 3.3.3** *Seja $g$ uma função duas vezes continuamente diferenciável em $R^n \times R^p$ e seja válida a Hipótese 3.3.2 para todo o $\overline{t}_i$, $i = 1, \ldots, \overline{m}$. A condição suficiente de segunda ordem para que $\overline{t}_i$ seja um maximizante local do problema* (3.3.1) *é:*

$\nabla^2_{tt} \mathcal{L}^i(\overline{t}_i, \overline{\alpha}^i)$ *é definida negativa no espaço tangente*

$$\mathcal{T}^i = \{\eta | \overline{\alpha}^i_j \eta^T \nabla_t h_j(\overline{t}_i) = 0, j \in \mathcal{M}^i\}.$$

## 3.4 Condições de optimalidade de segunda ordem

As condições de segunda ordem para o problema de PSI são basicamente as condições de segunda ordem para o problema reduzido (3.3.2)

De acordo com as Hipóteses 3.3.2 e 3.3.3 é possível provar o seguinte teorema (veja-se em [58]).

**Teorema 3.4.1** *(Teorema 5.1 em [58]) Suponha que $\overline{x} \in R^n$ é um ponto admissível e que a Hipótese 3.3.3 é válida. Tem-se então*

*(a) (Condição necessária) Se $\overline{x}$ é solução de* (3.1.1)*, então para todo o $\xi$ em $\mathcal{T}$,*

$$\mathcal{T} = \left\{\xi | \xi^T \nabla_x f(x) \leq 0, \xi^T \nabla_x g(x, t_i) \leq 0, i = 1, \ldots, \overline{m}\right\},$$

*existem $\lambda_0(\xi) \geq 0$, $\lambda(\xi) \in R^{\overline{m}}_+$, $\sup(\lambda(\xi)) \in \Gamma((\lambda_0(\xi), \lambda(\xi)) \neq (0, 0))$, tais que, para*

$$L(x, \lambda_0, \lambda, t) := \lambda_0 f(x) + \sum_{i=1}^{\overline{m}} \lambda_i g(x, t_i),$$

*tem-se*

$$\nabla_x L(\overline{x}, \lambda_0(\xi), \lambda(\xi), \overline{t}) = 0 \tag{3.4.1}$$

*e*

$$q(\xi) := \xi^T \nabla^2_{xx} L(\overline{x}, \lambda_0(\xi), \lambda(\xi), \overline{t})\xi - \sum_{i=1}^{\overline{m}} \lambda_i(\xi)(\nabla_x t_i(\overline{x})\xi)^T \nabla^2_{tt} \mathcal{L}^i(\overline{t}_i, \alpha(\overline{x}))(\nabla_x t_i(\overline{x})\xi) \geq 0$$

*(b) (Condição suficiente) Se para todo o* $\xi \in \mathcal{T}$, *existem* $\lambda_0(\xi)$ *e* $\lambda(\xi)$ *que verificam as condições referidas em (a) e em* (3.4.1) *e se para* $\xi \neq 0$,

$$q(\xi) > 0,$$

*então* $\overline{x}$ *é uma solução local estrita de* (3.1.1).

*(c) (Condição suficiente forte) A alínea (b) é especialmente verificada se*

- *os vectores* $\nabla_x g(x, t_i),\ i = 1, \ldots, \overline{m}$ *são linearmente independentes;*
- *existem* $\lambda_i \geq 0$ *únicos tais que a condição* (3.4.1) *é válida com* $\lambda_0(\xi) = 1$ *e* $\lambda(\xi) = \lambda$*;*
- $q(\xi) > 0$, *com* $\lambda_0(\xi) = 1$ *e* $\lambda(\xi) = \lambda$, *para todo o* $\xi \in \Sigma$, *onde*

$$\Sigma = \left\{ \xi | \lambda_i (\nabla_x g(x, t_i))^T \xi = 0,\ i = 1, \ldots, \overline{m} \right\}.$$

**Nota 3.4.1** *O primeiro termo em* $q(\xi)$ *é o termo de segunda ordem que se obtém para um problema finito regular e o segundo termo reflecte a estrutura semi-infinita do problema e é gerado pelo desvio das restrições activas* $\overline{t}_i(x)$ *como funções de* $x$.

# Capítulo 4

# Abordagens clássicas

---

Existem vários métodos numéricos para a resolução de problemas de PSI. Os diferentes métodos definem quatro classes principais: métodos de discretização, métodos de trocas, métodos baseados na redução local e métodos de transcrição de restrições. As três primeiras classes baseiam-se na substituição do conjunto infinito de restrições por um conjunto finito e o modo como essa substituição é feita caracteriza a classe. A classe da transcrição de restrições reformula o problema de PSI como um problema finito em que as restrições infinitas são transcritas como restrições finitas que envolvem o uso de integrais.

---

Este capítulo inclui quatro secções dedicadas às quatro abordagens clássicas. A Secção 4.1 aborda o método de discretização, a Secção 4.2 descreve o método de trocas e a Secção 4.3 trata do método de redução local. Na última secção apresenta-se o método de transcrição de restrições.

Nas secções que se seguem é necessária a seguinte definição.

**Definição 4.0.1** *Define-se o problema finito (PF) da seguinte forma:*

$$PF(\overline{T}) \equiv \begin{array}{c} \min\limits_{x \in R^n} f(x) \\ s.a \;\; g_i(x,t) \leq 0 \quad i = 1, \ldots, m \\ \forall t \in \overline{T}, \quad \left|\overline{T}\right| < \infty. \end{array} \tag{4.0.1}$$

A inclusão de restrições finitas de igualdade e desigualdade na definição (4.0.1) é imediata e não implica um aumento do grau de dificuldade na resolução do problema, pelo que, por simplicidade, usa-se esta definição.

## 4.1 Método de discretização

A ideia base de um método de discretização consiste na substituição do conjunto infinito $T$ por um conjunto finito de pontos $T_k \subset T$ que é usualmente uma grelha de pontos. A implementação de um método de discretização traduz-se na resolução de uma sucessão de subproblemas finitos baseados nos conjuntos $T_k$ ( $T_0 \subset T_1 \subset \cdots \subset T_k \subset T$), em que $T_k \to T$ com $k \to \infty$.

A descrição do método será baseada na hipótese de que o conjunto $T_k$ é uma grelha de pontos. A seguinte definição apresenta o conceito de grelha de uma forma mais formal.

**Definição 4.1.1** *Seja $h = (h_1, \ldots, h_p)$ um vector de $R^p$ e $\tau = (\tau_1, \ldots, \tau_p)$ um ponto de $R^p$. Define-se uma grelha de pontos $T_h$ como*

$$T_h = G_h \cap T$$

*em que*

$$G_h = \{t = (t_1, \ldots, t_p) | t_j = \tau_j + \alpha_j h_j,\ \alpha_j \in Z,\ j = 1, \ldots, p\}\,.$$

O seguinte algoritmo descreve de uma forma geral um método de discretização.

**Algoritmo 4.1.1** *Método de discretização.*

> ***Passo(i)*** *Dados $h^i$, uma selecção de pontos da grelha $\widetilde{T}_{h^i} \subset T_{h^i}$ e a solução $\widetilde{x}_i$ do problema PF($\widetilde{T}_{h^i}$).*
>
> 1. *Seja $h^{i+1} = \frac{h^i}{n_i}$ ($n_i \in N,\ n_i \geq 2$).*
> 2. *Seleccionar um conjunto de pontos da grelha $\widetilde{T}_{h^{i+1}} \subset T_{h^{i+1}}$ (com base em $\widetilde{x}_i$, $T_{h^i}$, e eventualmente nos valores já testados para $\widetilde{T}_{h^{i+1}}$ e $\widetilde{x}_{i+1}$).*
> 3. *Calcular a solução $\widetilde{x}_{i+1}$ de PF($\widetilde{T}_{h^{i+1}}$). Se $\widetilde{x}_{i+1}$ não é admissível para $T_{h^{i+1}}$ então continuar com 2.*
> 4. *Se $i > i_0$ (número máximo de refinamentos) então parar, senão executar o* ***Passo($i+1$)****.*

Uma parte significativa das implementações descritas na bibliografia são para programação semi-infinita linear e quadrática. Um problema de programação semi-infinita linear é um problema do tipo (1.1.1) em que a função objectivo é linear ($c^T x$, com $c, x \in R^n$) e as restrições infinitas são do tipo

$$A^T(t)x - b(t) \leq 0, \quad \forall t \in T,$$

em que $A(t)$ é uma matriz $n \times m$ e $b(t)$ é um vector de dimensão $m$. A substituição de $T$ pela grelha $\widetilde{T}_h$ origina o seguinte problema linear finito (PL)

$$\mathrm{PL}(\widetilde{T}_h) \equiv \quad \begin{array}{ll} \min\limits_{x \in R^n} c^T x \\ s.a & A(t)^T x - b(t) \leq 0, \\ & \forall t \in \widetilde{T}_h, \ |\widetilde{T}_h| < \infty. \end{array}$$

Um problema de PSI quadrático caracteriza-se por ter uma função objectivo quadrática e possuir apenas restrições finitas/infinitas lineares.

Hettich em [55] descreve um método de discretização para programação semi-infinita linear, sem restrições finitas. A extensão para problemas de PSI quadráticos é mais tarde proposta por Hettich e Gramlich em [57]. Este algoritmo é basicamente o descrito em [55] embora o problema discretizado seja quadrático. Reemtsen [116] descreve também um método de discretização para programação semi-infinita quadrática. No âmbito deste trabalho foram propostas extensões dos algoritmo descritos em [57] e [116] para PSI não linear [151]. No Capítulo 8 é feita uma descrição mais pormenorizada dos correspondentes algoritmos, bem como de uma outra versão em que $T_k$ é um conjunto de pontos pseudo-aleatórios.

Em [1] os autores propõem um método de discretização para programação linear em que é definida uma sucessão de grelhas de pontos. A selecção dos pontos que irão ser usados para resolver o problema finito é determinada através de uma função alcance (*reach function*) que é calculada com base nas funções das restrições.

Os métodos propostos em [44] e [105] baseiam-se na construção de um problema quadrático reduzido usando as funções $t_i(x)$, $i = 1, \ldots, \overline{m}$, definidas na Hipótese 3.3.1. Como estas funções são difíceis de obter, os algoritmos usam uma grelha de pontos para as aproximar.

A paralelização dos métodos de discretização é abordada por Kaliski *et al.* em [66] que usa um algoritmo baseado na função barreira logarítmica. A implementação foi efectuada numa versão série e numa versão paralela usando uma arquitectura SIMD (*Single Instruction, Multiple Data*). Na versão paralela, apenas foi paralelizada a rotina de verificação da grelha. Os autores também efectuaram um estudo comparativo entre o uso de uma grelha de pontos estática e uma grelha de pontos dinâmica.

## 4.2 Métodos de trocas

Os métodos de trocas [10, 27, 48, 59, 117, 118, 121, 162, 169] são métodos iterativos que substituem, em cada iteração, as restrições infinitas por um conjunto de restrições finitas. Estes métodos procedem à remoção e à inserção de novas restrições no conjunto das restrições finitas de iteração para iteração. Em termos algoritmicos, estes métodos podem ser descritos do seguinte modo:

**Algoritmo 4.2.1** *Método das trocas.*

***Passo** (i) Dado $T_{i-1} \subset T$, $|T_{i-1}| < \infty$.*

1. *Determinar a solução $x_i$ de PF($T_{i-1}$) e soluções $t_1, \ldots, t_{m_i}$ aproximadas para o problema* (3.3.1)
2. *Se $g(x_i, t_l) \leq 0, l = 1, \ldots, m_i$ então parar.*
3. *Escolher $T_i$ tal que*

$$T_i \subset T_{i-1} \cup \{t_1, \ldots, t_{m_i}\} .$$

4. *Executar o **Passo**$(i+1)$.*

No ponto 3 podem ser adicionadas restrições correspondentes aos pontos do conjunto $\{t_1, \ldots, t_{m_i}\}$ e removidas restrições que correspondem a pontos do conjunto $T_{i-1}$.

Este tipo de métodos tem sido preferencialmente aplicados a problemas de PSI linear. Neste contexto Gribik em [48] propõe um algoritmo conceptual com base em planos de corte (*cutting plane*). A estratégia utilizada por Gribik para a inserção de novas restrições é a seguinte. Seja

$$\begin{aligned} f = & \min_x c^T x \\ s.a \;\; & a^T(t)x - b(t) \leq 0 \;\; \forall t \in T \\ & x \in X \end{aligned} \tag{4.2.1}$$

o problema de PSI linear, onde $c \neq \mathbf{0}$ e $a(t)$ são vectores de dimensão $n$, $b(t)$ é um escalar e $X \in R^n$. O algoritmo começa por resolver o problema finito relaxado

$$\begin{aligned} & \max \sigma \\ s.a \;\; & c^T x + \|c\|\sigma \leq \bar{f} \\ & x \in X \end{aligned} \tag{4.2.2}$$

em que $\bar{f}$ é uma estimativa, por excesso, de $f$. Se $x_k$ for a solução do problema (4.2.2), o algoritmo adiciona a restrição

$$c^T x + \|c\|\sigma \leq c^T x_k, \tag{4.2.3}$$

se $x_k$ for admissível, isto é, se

$$a^T(t)x_k - b(t) \leq 0 \;\; \forall t \in T, \tag{4.2.4}$$

e adiciona a restrição

$$a^T(t_k)x - \|a(t_k)\|\sigma - b(t_k) \leq 0 \tag{4.2.5}$$

caso contrário, onde $t_k \in T$ é tal que

$$a^T(t_k)x_k - b(t_k) \geq 0. \tag{4.2.6}$$

O algoritmo resolve o novo problema finito resultante da adição da restrição e procede no final à remoção das restrições não necessárias.

Um algoritmo com planos de corte foi também usado por Reemtsen em [117, 118]. No primeiro artigo, o algoritmo é aplicado a um problema de aproximação de funções complexas. Na solução do problema linear finito resultante do método das trocas, Roleff [121] propõe um método baseado no método simplex para programação linear que permite mais do que uma troca na matriz base. Na grelha inicial é proposto um espaçamento de uma centésima para determinar a matriz base inicial e posteriormente é aplicado o método de Newton ao problema (3.3.1).

Um híbrido do método do tipo adição de restrição (*adding constraint*) e do método do tipo perturbação para programação finita linear é proposto em [162] para a resolução de problemas de PSI linear. A restrição que é adicionada, em cada iteração, corresponde à solução do problema (3.3.1).

No contexto da PSI linear, propõe-se em [159] um algoritmo estocástico para a resolução do problema (3.3.1).

Hu em [59] aplica um método de descida máxima para programação linear. Com base em $\epsilon_k > 0$, $\lambda_k > 0$ e $x_k$ o algoritmo determina a $\epsilon_k$-solução do problema não linear

$$\sup\{a^T(t)(x_k + \lambda_k c) - b(t) : t \in T\}, \tag{4.2.7}$$

ou seja, determina um $t_k$ que satisfaz

$$a^T(t_k)(x_k + \lambda_k c) - b(t_k) \geq \sup\{a^T(t)(x_k + \lambda_k c) - b(t) : t \in T\} - \epsilon_k.$$

Se $x_k + \lambda_k c$ satisfaz a restrição $a^T(t_k)(x_k + \lambda_k c) - b(t_k) \leq 0$, pode concluir-se que $x_k + \lambda_k c$ está suficientemente perto da região admissível e considera-se $x_{k+1} = x_k + \lambda_k c$. Caso contrário, calcula-se $x_{k+1}$ como a projecção de $x_k + \lambda_k c$ em $a^T(t_k)x = b(t_k)$. O problema exemplo (4.2.7) usado em [59] é equivalente a determinar o maior valor próprio e um vector próprio unitário de uma matriz.

Uma aproximação à solução de (3.3.1) é obtida por Wu e Fang em [169] através de um problema relaxado em que a convergência é garantida para pontos $t_{k+1}$ que verifiquem $a^T(t_{k+1})x_k - b(t_{k+1}) < -\delta$, sendo $\delta > 0$ uma constante suficientemente pequena.

Problemas de desenho óptimo de filtros FIR podem ser formulados como problemas de PSI [110]. Neste artigo, o autor resolve o problema (3.3.1) usando a rotina IMSL-"DUVMIF" de [100] e considera para a definição do problema finito todos os máximos locais que têm valores superiores a $\zeta \geq 0$, ou seja, $g(\bar{x}, t_i) \geq \zeta$, $i = 1, \ldots, m$.

No contexto da PSI quadrática, Fang *et al.* [27] propõem um método baseado na função entrópica $-x \ln x$. O algoritmo baseia-se na resolução do problema de optimização entrópico (problema em que a função objectivo possui um termo extra com a função entrópica) em que é adicionada, em cada iteração, a restrição que corresponde a um máximo global do problema (3.3.1). A resolução do problema (3.3.1) é baseada na divisão do intervalo $T$ em 100 subintervalos e na utilização da sub-rotina do IMSL [100] para encontrar um máximo em cada subintervalo. Como a sub-rotina não garante a determinação do máximo global, o algoritmo usa o maior dos 100 máximos locais como aproximação ao máximo global.

Blankenship e Falk em [10] propõem também um algoritmo baseado em planos de corte que adiciona em cada iteração, uma restrição não linear, que corresponde à solução do problema (3.3.1). O algoritmo permite a remoção de restrições anteriormente introduzidas que não estejam activas naquela iteração. Para a resolução do problema finito, os autores propõem duas abordagens: uma baseada em funções de penalidade e a outra no método de Wolfe denominado de linearização de grelha. Os problemas (3.3.1) foram resolvidos de uma forma analítica para os exemplos numéricos apresentados.

## 4.3 Métodos baseados na redução local

Os métodos de redução local ([58]) usam a teoria de redução local descrita na Secção 3.3, para gerarem o problema (3.3.2). O algoritmo baseado na redução local pode ser descrito da seguinte forma:

**Algoritmo 4.3.1** *Método de redução local.*

***Passo**(i) Dado um $x_i$.*

1. *(procura multi-local)*
   *Determinar todos os máximos locais e globais $T_{red} = \{t_1, \ldots, t_{m_i}\}$ do problema* (3.3.1).
2. *Resolver o problema reduzido PF($T_{red}$). Seja $x_{i+1}$ a solução.*
3. *Continuar com o **Passo**$(i+1)$.*

De seguida faz-se uma breve descrição dos métodos que têm sido propostos neste âmbito, começando pelos vocacionados para PSI linear.

Em [71] os autores sugerem um método do tipo simplex com uma estratégia de pivotagem combinado com um esquema de direcções admissíveis. É incluída também uma estratégia de purificação (veja-se em [72]) que permite progredir de uma solução admissível para um ponto extremo. O algoritmo multi-local proposto por León *et al.* em [70] consiste numa estratégia passiva de partição do intervalo $T$. O método baseia-se no cálculo das derivadas da função num número de pontos igualmente espaçados do intervalo $T$. Os subintervalos que não contenham um óptimo local são abandonados, enquanto que se procede a uma procura unidimensional não exacta em cada subintervalo que contêm um óptimo local. Em [3] os autores propõem uma extensão do método simplex para PSI linear e propõem também uma estratégia de purificação. No artigo não é apresentada nenhuma estratégia em particular para abordar o problema (3.3.1).

Em 1985, Coope e Watson [22] sugerem um algoritmo para programação semi-infinita baseado na Lagrangeana projectada. A Lagrangeana de um problema de PSI, com apenas uma restrição infinita, é definida da seguinte forma

$$L(x, \lambda) = f(x) + \sum_{i=1}^{\overline{m}} \lambda_i g(x, t_i)$$

onde $t_i, i = 1, \ldots, \overline{m}$ são os máximos locais e globais de $g(x,t)$ (para um dado $x$). O procedimento utilizado para determinar todos estes máximos (procura multi-local) consistiu na construção de uma grelha uniforme de pontos e na identificação dos máximos através da aplicação de um método do tipo Newton.

A direcção de procura, usada para obter uma melhor aproximação à solução, é calculada resolvendo o seguinte problema quadrático

$$\begin{aligned} &\min_{d \in R^n} d^T \nabla f(x) + \frac{1}{2} d^T H d \\ &\text{s.a} \quad g(x,t_i) + d^T \nabla_x g(x,t_i) = 0, \quad i = 1, \ldots, \overline{m}, \end{aligned} \tag{4.3.1}$$

e a função mérito utilizada é a função de penalidade exacta (veja-se, por exemplo, em [13])

$$P(x) = f(x) + \mu \sum_{i=1}^{\overline{m}} [g(x,t_i)]_+ \tag{4.3.2}$$

onde $[g]_+ = \max\{0, g\}$.

Para $\mu \geq \|\lambda\|_\infty$, a direcção obtida por (4.3.1) é uma direcção de descida para a função (4.3.2). Quando $d = 0$, a solução de (4.3.1) é um ponto estacionário para o problema de PSI. A Hessiana da função quadrática de (4.3.1) é mantida definida positiva através do esquema $H = \nabla^2 L(x,\lambda) + \mu I$. Para a procura unidimensional, o comprimento do passo $\alpha$ é o maior valor da sequência $\left\{1, \frac{1}{2}, \frac{1}{4}, \ldots\right\}$ que verifica $T(\alpha, x) \geq \rho$ ($\rho$ um valor fixo dado) sendo

$$T(\alpha, x) = \frac{P(x + \alpha d) - P(x)}{\alpha P'(x;d)}$$

em que $P'(x;d) < 0$ é a derivada direccional da função de penalidade em $x$ segundo a direcção $d$.

O algoritmo proposto por Price [112, 113] usa apenas primeiras derivadas e o algoritmo baseia-se na função de penalidade exacta

$$\phi(\mu, \nu; x) = f(x) + \mu\theta + \frac{1}{2}\nu\theta^2 \text{ onde } \theta = \max_{t \in T} [g(x,t)]_+ .$$

Na procura multi-local é usado um algoritmo estocástico que consiste em gerar pontos pseudo-aleatórios no conjunto $T$ e atribuir a esses pontos ligações com outros pontos da vizinhança com maior valor da função objectivo. Às ligações são atribuídos pesos que posteriormente são usados como critério de paragem para o processo. Um algoritmo do tipo quasi-Newton é então implementado a partir dos pontos da vizinhança com maior valor da função objectivo.

A direcção de procura $d$ é calculada com base numa aproximação quadrática de $\phi$ definida por

$$\psi(x_k, A; \mu, \nu; d) = f(x_k) + d^T \nabla f(x_k) + \frac{1}{2} d^T H d + \mu\zeta(d) + \frac{1}{2}\nu\zeta^2(d),$$

onde

$$\zeta(d) = \max_{t \in A} \left[ g(x_k, t) + d^T \nabla_x g(x_k, t) \right]_+$$

e $A \in T$ é um conjunto finito. Mostra-se ([111]) que para $d$ suficientemente pequeno

$$\phi(\mu, \nu; x_k + d) = \psi(x_k, A; \mu, \nu; d) + o\left(\|s\|\right).$$

Como o problema de minimizar $\psi$ pode ser reescrito na forma

$$\begin{aligned} \min_{d \in R^n, \zeta \in R} \; & d^T \nabla f + \frac{1}{2} d^T H d + \mu\zeta + \frac{1}{2}\nu\zeta^2 \\ \text{s.a} \quad & g(x_k, t) + d^T \nabla_x g(x_k, t) - \zeta \leq 0 \\ & \zeta \geq 0 \\ & \forall t \in A, \end{aligned}$$

os multiplicadores de Lagrange deste problema são usados não só como estimativas para os multiplicadores de Lagrange óptimos como para actualizar $H$, $\mu$, e $\nu$. A procura de uma nova aproximação é baseada no critério de Armijo e no arco definido por

$$x_{k+1} = x_k + \alpha d + \alpha^2 c.$$

O vector $c$ é usado para evitar o efeito *Maratos* e consequentemente garantir a convergência superlinear.

Em programação não linear finita pode usar-se o método de Newton para resolver o sistema das condições de optimalidade de primeira ordem, sendo a convergência local e superlinear. Esta abordagem ([111]) aplicada a um problema de PSI resulta num sistema que pode ser de grandes dimensões

$$\begin{aligned} \nabla f(x) + \sum_{i=1}^{\overline{m}} \lambda_i \nabla_x g(x, t_i) &= 0 \\ g(x, t_i) &= 0, \quad i = 1, \ldots, \overline{m} \\ \nabla_t g(x, t_i) + \sum_{j \in \mathcal{H}_T(i)} \xi_{ij} \nabla h_j(t_i) &= 0, \quad i = 1, \ldots, \overline{m} \\ h_j(t_i) &= 0, \quad i = 1, \ldots, \overline{m} \text{ e } \forall j \in \mathcal{H}_T(i) \end{aligned}$$

onde $\mathcal{H}_T(i) = \{j | h_j(t_i) = 0, \quad i = 1, \ldots, \overline{m}\}$. As variáveis $\xi_{ij}$ são os multiplicadores de Lagrange das restrições $h_j(t) \leq 0$ que estão activas nos máximos $t_i$, $i = 1, \ldots, \overline{m}$. Note-se que a dimensão do sistema pode ser diferente para diferentes valores de $x$.

Haaren-Retagne em [52] implementa um método de programação quadrática sequencial que usa as condições de optimalidade do problema reduzido (3.3.2).

O algoritmo multi-local proposto por Haaren-Retagne em [52], para um dado $\overline{x}$, consiste em aplicar uma procura local no intervalo $[t_{i-1}, t_{i+1}]$, em que $t_\kappa$, $\kappa = 1, \ldots, q$ é uma grelha

unidimensional de pontos e $g(\overline{x}, t_i) > \max\{g(\overline{x}, t_{i-1}), g(\overline{x}, t_{i+1})\}$. Na grelha $t_0 = t_1 = 0$, $t_{q+1} = t_q = 1$ e $g(\overline{x}, t_0) = g(\overline{x}, t_{q+1}) = -\infty$. A admissibilidade da iteração é avaliada para uma grelha fina de pontos equidistantes. A grelha é refinada localmente sempre que se verifique que um óptimo local foi perdido.

Os algoritmos propostos por León *et al.* em [70] e por Haaren-Rectagne em [52] para a procura multi-local foram aplicados ao caso unidimensional ($p = 1$), enquanto que o algoritmo estocástico proposto por Price em [111] foi aplicado a problemas com $p \geq 1$.

## 4.4 Métodos de transcrição de restrições

Os métodos pertencentes a esta classes são caracterizados por transcreverem as restrições infinitas em restrições finitas que envolvem o uso de integrais. O cálculo numérico das restrições finitas pode ser estático ou dinâmico, mas os pontos usados para calcular os integrais podem ser diferentes de restrição para restrição e de iteração para iteração. As técnicas de programação finita podem agora ser usadas para resolver o problema com as restrições finitas.

Apresenta-se de seguida alguma notação que será usada ao longo desta secção.

Para $z \in R$ seja,

$$z_+ = max\{0, z\}$$

e

$$sgn(z) = \begin{cases} +1 & \text{se} \quad z > 0 \\ 0 & \text{se} \quad z = 0 \\ -1 & \text{se} \quad z < 0. \end{cases}$$

Segue-se uma breve descrição de algumas técnicas que têm sido usadas no contexto de transcrição de restrições.

Lin *et al.* em [77] resolve o problema de PSI linear

$$\begin{aligned} &\max_{x \in R^n} c^T x \\ &s.a \;\; a^T(t)x - b(t) \leq 0 \;\; \forall t \in T \end{aligned} \tag{4.4.1}$$

através do problema sem restrições

$$\max_{x \in R^n} c^T x - \mu \int_T e^{\frac{a^T(t)x - b(t)}{\mu} - 1} d\lambda(t), \tag{4.4.2}$$

onde $\lambda(t)$ é uma medida de Lebesgue (veja-se em [166]). Na resolução numérica do integral, os autores usaram a regra de Simpson com base numa partição do intervalo $[0, 1]$ em 400 000 subintervalos. Para os problemas em que $p = 2$ o intervalo $[0, 1] \times [0, 1]$ foi dividido em $1500^2$ subintervalos.

A extensão do método de escalonamento afim (*affine scaling*) para a programação semi-infinita foi proposta por Ferris e Philpott em [29, 30]. O algoritmo introduz na formulação do problema uma infinidade de variáveis de folga que levam ao aparecimento de integrais em $T$, calculados através da regra de Simpson.

No contexto dos métodos de pontos interiores, em [60] demonstra-se que o integral da função barreira, que surge directamente da formulação semi-infinita não é uma função auto-concordante e consequentemente exibe uma convergência local pobre. O autor sugere que o problema pode ser resolvido mais eficientemente através da aplicação da técnica de pontos interiores ao problema reduzido.

Em relação à PSI quadrática, Liu e Teo em [79] resolvem o problema

$$\begin{aligned} & \min_{x \in R^n} \frac{1}{2} x^T Q x + c^T x \\ s.a \quad & a(t)^T x - b(t) \leq 0 \\ & \forall t \in T \end{aligned}$$

através do problema dual

$$\begin{aligned} & \min_{x \in R^n} \frac{1}{2} x^T Q x + \int_T b(t) d\lambda(t) \\ s.a \quad & Qx + c + \int_T a(t) d\lambda(t) = 0 \\ & \lambda(t) \geq 0 \quad \forall t \in T. \end{aligned}$$

O cálculo do integral é baseado numa sucessão de números inteiros $\{k_i\}$, $i = 1, 2, \ldots$ e de pontos $\tau^i = (t_1^i, t_2^i, \ldots, t_{k_i}^i)^T$ escolhidos da seguinte forma: para $k_1$, escolhem-se $k_1$ pontos igualmente espaçados do intervalo $T$ (incluindo os limites). Na iteração seguinte, além dos pontos da iteração anterior são também incluídos os pontos médios dos subintervalos definidos pelos pontos anteriores. A seguinte sucessão de problemas é então resolvida

$$\begin{aligned} & \min_{x \in R^n} \frac{1}{2} x^T Q x + \sum_{j=1}^{k_i} b(t_j) \lambda_j \\ s.a \quad & Qx + c + \sum_{j=1}^{k_i} a(t_i) \lambda_j = 0 \\ & \lambda_j \geq 0 \quad j = 1, \ldots, k_i, \end{aligned}$$

onde $\lambda(t_j) = \lambda_j$, até que um determinado critério de paragem seja satisfeito.

Para a PSI não linear, Teo e Goh em [139] propuseram uma transcrição das restrições infinitas do problema

$$\begin{aligned} & \min_{x \in R^n} f(x) \\ s.a \quad & g_i(x, t) \leq 0 \\ & \forall t \in T, i = 1, \ldots, m, \end{aligned} \tag{4.4.3}$$

transformando-o num problema de programação não linear finito (PNL). As restrições de desigualdade infinitas são transformadas em restrições de igualdade finitas da seguinte forma:

$$G_i(x) \equiv c \int_T [g_i(x,t)]_+^2 \, dt = 0, \quad i = 1, \ldots, m$$

onde $c$ é uma constante empírica que é usada para melhorar a precisão numérica. No entanto, as novas restrições não satisfazem a condição usual de independência linear (*constraint qualification*) e a convergência, com os métodos típicos para PNL, não é garantida. Apesar disso Teo e Goh apresentaram resultados numéricos para dois exemplos.

Neste contexto, para a resolução do problema de PSI, Pietrzykowski propõe uma função de penalidade definida por

$$p(x,\mu) = \mu f(x) + \sum_{i=1}^{m} \sum_{j=1}^{s_i} \int_{\Omega_{ij}(x)} g_i(x,t) dt \tag{4.4.4}$$

onde $\mu$ é um escalar positivo e $\Omega_{ij}(x)$, $i = 1, \ldots, m$, $j = 1, \ldots, s_i$ são conjuntos que satisfazem as seguintes propriedades:

(i) $\Omega_{ij} \subseteq T$, $1 \leq j \leq s_i = s_i(x) < \infty$,

(ii) $g_i(x,t) \geq 0$, $\forall t \in \Omega_{ij}(x)$ e $g_i(x,t) < 0$, $\forall t \in T \setminus \cup_{j=1}^{s_i} \Omega_{ij}(x)$,

(iii) $\Omega_{ij}(x) \cap \Omega_{ik}(x) = \emptyset$ se $j \neq k$, e

(iv) $\Omega_{ij}(x)$ é conexo e não trivial, i.e., $\int_{\Omega_{ij}(x)} dt > 0$.

Como esta função não é exacta (veja-se a Nota 2.2.1), Conn e Gould [20] usam a seguinte alternativa

$$\phi_{CG}(x,\mu) = f(x) + \mu \sum_{i=1}^{m} \left( \frac{\sum_{j=1}^{s_i} \int_{\Omega_{ij}(x)} g_i(x,t) dt}{\sum_{j=1}^{s_i} \int_{\Omega_{ij}(x)} dt} \right), \tag{4.4.5}$$

em que o denominador do termo de penalidade serve para tornar a penalização suficientemente forte no sentido de que a função de penalidade seja exacta. Como os conjuntos $\Omega_{ij}$, $i = 1, \ldots, m$, $j = 1, \ldots, s_i$ não são fáceis de calcular, pode usar-se em alternativa a função

$$\phi_{CG}(x,\mu) = f(x) + \mu \sum_{i=1}^{m} \frac{\int_T [g_i(x,t)]_+ dt}{\int_T [sgn\,(g_i(x,t))]_+ dt} \tag{4.4.6}$$

que de certa forma é equivalente à apresentada em (4.4.5) ([111]).

Como a função (4.4.6) não é diferenciável, a sua minimização exige a implementação de técnicas que não recorram às derivadas. Veja-se por exemplo, Polak [103] para uma revisão sobre optimização não diferenciável e Wolfe [167] para o método de Powell que não usa derivadas. Em [19] é proposta outra abordagem para funções não diferenciáveis.

Mais tarde, Jennings e Teo [63] mostraram como o problema (4.4.3) pode ser substituído por um problema aproximado (de uma forma semelhante ao de [139]), com restrições uma vez continuamente diferenciáveis em $x$ da forma

$$\begin{aligned} &\min_{x\in R^n} \ f(x) \\ s.a \ \ &G_{i,\epsilon}(x) \equiv \int_T g_{i,\epsilon}(x,t)dt = 0, \quad i = 1,\ldots,m \end{aligned} \tag{4.4.7}$$

em que

$$g_{i,\epsilon}(x,t) = \begin{cases} 0, & \text{se } \ g_i(x,t) < -\epsilon; \\ \left(g_i(x,t)+\epsilon\right)^2/4\epsilon, & \text{se } \ -\epsilon \le g_i(x,t) \le \epsilon; \\ g_i(x,t), & \text{se } \ g_i(x,t) > \epsilon, \end{cases} \tag{4.4.8}$$

para $\epsilon$ real positivo e suficientemente pequeno.

A aproximação (4.4.8) é usada (com uma alteração apropriada para o caso de restrições $\geq$) no software de controlo óptimo MISER3 [62] para tratar restrições de estado, contínuas e de desigualdade que são independentes da função de controlo. O parâmetro $\epsilon$ é decrementado ao longo do processo iterativo até atingir o valor $\epsilon = 10^{-4}$.

De facto, as restrições de igualdade $G_{i,\epsilon}(x) = 0$ do problema (4.4.7) não satisfazem novamente a condição usual de independência linear, não sendo aconselhável a resolução do problema na formulação (4.4.7). Em alternativa pode resolver-se o seguinte problema aproximado

$$\begin{aligned} &\min_{x\in R^n} \ f(x) \\ s.a \ \ &G_{i,\epsilon}(x) < \tau, \\ &\qquad i = 1,\ldots,m \end{aligned} \tag{4.4.9}$$

para $\tau$ positivo e suficientemente pequeno. O problema (4.4.9) foi resolvido, em [63], usando a rotina E04VCF da biblioteca da NAG [76].

Teo *et al.* em [140] desenvolveram também um novo algoritmo para a PSI baseado na transcrição das restrições e na função de penalidade exacta $L_1$

$$\phi_1(x,\mu) = f(x) + \mu \sum_{i=1}^{m} \int_T g_{i,\epsilon}(x,t)dt.$$

O problema minimax da forma

$$\begin{aligned} &\min_{x\in R^n} \max_{t\in T} h(x,t) \\ s.a \ \ &g_i(x,t) \le 0, \quad i = 1,\ldots,m \\ &\forall t \in T \end{aligned}$$

foi equacionado por Jiang *et al.* [64] como um problema de PSI

$$\begin{aligned} & \min_{x \in R^n,\, \alpha} \ \alpha \\ s.a \quad & g_0(x,t,\alpha) \equiv h(x,t) - \alpha \leq 0 \\ & g_i(x,t) \leq 0, \quad i = 1, \ldots, m \\ & \forall t \in T \ , \end{aligned} \tag{4.4.10}$$

em que a função auxiliar

$$J(\epsilon, \alpha) = \min_{x \in R^n} \int_T \left( g_{0,\epsilon}(x,t,\alpha) + \sum_{i=1}^{m} g_{i,\epsilon}(x,t) \right) dt$$

é usada para determinar o valor de $\alpha$ óptimo. Com base em dois valores de $\alpha$ e comparando os valores da função auxiliar em diversos pontos do intervalo definido por esses dois valores, o algoritmo progride em direcção ao óptimo que pertence ao intervalo.

# Capítulo 5

# SIPAMPL

---

O SIPAMPL (*Semi-Infinite Programming with AMPL*) é um pacote de software que começou a ser desenvolvido nos finais de 1999. Na altura não existia uma base de dados com problemas de PSI que pudesse ser usada para avaliar o desempenho dos algoritmos numéricos, à semelhança do CUTE [11] em programação finita. Com base na linguagem de modelação AMPL [36], muito mais simples que a linguagem SIF (*Standard Input Format*) do CUTE, e nas suas ferramentas de diferenciação automática, o SIPAMPL contém actualmente uma base de dados com cento e quarenta e quatro problemas de PSI codificados e uma interface para ligação entre as rotinas do AMPL e qualquer software de resolução de problemas de PSI. A codificação de alguns problemas de robótica exigiu a construção de uma biblioteca dinâmica de "B-Splines" (curvas paramétricas) para o AMPL. A ferramenta *select* do SIPAMPL permite fazer uma selecção de problemas de PSI de acordo com as suas características matemáticas.

---

A Secção 5.1 apresenta uma breve descrição da base de dados e ilustra como um problema de PSI pode ser codificado na linguagem de modelação do AMPL. A descrição da interface que liga a base de dados do SIPAMPL a qualquer software de resolução de problemas de PSI é feita na Secção 5.2. O relatório técnico, usado como manual, e o respectivo software podem ser obtidos através do endereço de internet indicado na referência [157]. A ferramenta *select* é apresentada na Secção 5.3 e um exemplo do seu uso no Apêndice D.

As instruções de instalação, bem como a estrutura de directórios usada no software distribuído são apresentadas no Apêndice B.

## 5.1 Base de dados do SIPAMPL

Ao longo dos últimos anos foram codificados vários problemas de PSI retirados da literatura da especialidade. Como é boa prática, os problemas foram codificados contendo, como comentário, no início de cada ficheiro (modelo do problema) a referência de onde o problema

foi extraído. Uma vez que a maioria dos autores não especifica a origem do problema optou-se por identificar o ficheiro com o nome do primeiro autor do trabalho.

A existência desta base de dados permitirá, no futuro, referenciar o problema usando o nome do ficheiro na base de dados do SIPAMPL, sem ter de reproduzir o problema em si, evitando assim erros frequentes na sua escrita.

O SIPAMPL permite ainda a inclusão de novos problemas codificados em AMPL e a sua rápida resolução através do acesso aos programas de resolução (*solver*). A codificação de problemas de PSI na linguagem de modelação AMPL é fácil e apenas requer a definição das funções envolvidas no problema. A diferenciação automática do AMPL fornece as primeiras e segundas derivadas de todas as funções codificadas (função objectivo e restrições).

A base de dados possui até ao momento 144 problemas codificados e a sua actualização será feita sempre que possível. Referências a novos problemas de PSI e problemas já codificados são, naturalmente, bem vindos.

Como exemplo de um problema codificado, considere-se o seguinte problema de PSI

$$\begin{aligned} & \min_{x \in R^2} \quad x_1^2 + x_2^2 \\ s.a \quad & x_1 t + x_2 t^2 \leq 0 \\ & -10 \leq x_1 + x_2 \leq 10 \\ & \forall t \in [0,1] \end{aligned}$$

cuja correspondente codificação em (SIP) AMPL é apresentada de seguida.

```
#########################################################################
# Objective: Quadratic
# Constraints: Linear
#########################################################################
# Problema exemplo
# aivaz@dps.uminho.pt
#########################################################################

var x {1..2}; # o nome das variáveis finitas não pode começar por t

var t;        # o nome das variáveis infinitas tem de começar por t


minimize fx:  # função objectivo
  x[1]^2+x[2]^2;

subject to tcons:         #   restrição infinita, logo o nome tem
  x[1]*t+x[2]*t^2 <=  0;  # de começar por t

subject to constraint:      #   restrição finita, logo o nome não
  -10  <= x[1]+x[2] <=  10; # pode começar por t
```

```
subject to bounds:          # limites simples na variável t
  0 <= t <= 1;


#################################
# Fim da codificação do problema #
#################################


option mysolver_auxfiles rc;   #   não esquecer de forçar a escrita
                               # dos ficheiros .col e .row
option reset_initial_guesses 1;#   este problema não tem
                               # aproximação inicial
option solver mysolver;        #   indicar o software de resolução
                               # do problema
solve;                         # resolver o problema

######################
# Solução encontrada #
######################
printf "Solução encontrada\n";
display x;
display fx;
```

# é o caracter que indica o início de um comentário em AMPL sendo ignorado o texto que surge a seguir ao caracter.

`mysolver` é o nome do software que resolve o problema de PSI e pode ser substituído para satisfazer as necessidades de cada utilizador (veja-se na Subsecção 5.1.1 como substituí-lo).

As restrições impostas para a codificação de problemas de PSI em AMPL são as seguintes:

- as variáveis infinitas são codificadas com nomes que começam por t e inversamente todas as variáveis cujos nomes começam por t são consideradas infinitas;
- as restrições infinitas são codificadas com nomes que começam por t e inversamente todas as restrições cujos nomes começam por t são consideradas infinitas;
- os ficheiros AMPL `.row` e `.col` têm de ser fornecidos através do uso da opção *auxfiles*.

### 5.1.1 Substituição do executável na base de dados

Cada ficheiro que contém um problema de PSI deve fazer referência ao software que vai resolver o problema. A selecção é feita através do comando `option solver nsips;` (`nsips`

é o software usado por omissão em todos os problemas codificados). Se o utilizador desejar utilizar o seu próprio software então um dos seguintes procedimentos tem de ser seguido:

- chamar `nsips` (nome do executável) ao seu software, ou renomeá-lo como `nsips`.
- editar todos os ficheiros de problemas que pretende usar e alterar `nsips` para o nome do seu próprio software.

Para efectuar o segundo procedimento, de uma forma automática, é necessário executar o seguinte conjunto de comandos para a concha (*shell*):

*for i in* `para os ficheiros que quer alterar`
*do*
*sed s/nsips/*`modificado`*/ < $i > $i.novo*
*done*

Estes comandos editam os ficheiros `para os ficheiros que quer alterar`, substituindo todas as ocorrências da palavra *nsips* por `modificado` (nome do novo software), dando origem aos novos ficheiros `para.novo os.novo ficheiros.novo que.novo quer.novo alterar.novo`.

A palavra `modificado` deve ser reservada no sentido de que não deve existir na definição do problema.

### 5.1.2 A biblioteca de "B-Splines"

O AMPL permite, além do uso de funções intrínsecas (sin, cos, *exp*, etc), o uso de funções externas. As funções externas são carregadas dinamicamente assim que o AMPL é usado.

A biblioteca para o AMPL que permite a codificação de problemas matemáticos que necessitam de "B-Splines" está disponível e será descrita de seguida.

O AMPL possui uma linguagem de descrição de modelos matemáticos que não permite recursividade. A diferenciação automática é usada apenas para obter as derivadas da função objectivo e restrições. Alguns problemas em programação matemática usam "B-Splines" para aproximar funções. A codificação desse tipo de funções pode ser tenebrosa, nomeadamente se for necessário usar várias "B-Splines", variando o grau dos polinómios base, número de coeficientes e o número de nós.

Uma "B-Spline" é formada por uma combinação linear de funções base e é representada por

$$B_{k,\xi}(t) = \sum_{i=1}^{n} x_i B_{i,k,\xi}(t) \tag{5.1.1}$$

onde $B_{i,k,\xi}(t)$, para todo o $i$, são as funções base para a "B-Spline". Estas funções base são polinómios de grau $k-1$ numa variável $(t)$ e formam uma base. $x_i$ são os coeficientes da "B-Spline" e $\xi$ é o vector dos nós.

Quando se usam "B-Splines" num problema matemático, pode ser necessário recorrer às derivadas relativas às variáveis $t$ e/ou $x_i$. O Apêndice E.1 contém as fórmulas destas derivadas.

A biblioteca de "B-Splines" está disponível através da internet[1] juntamente com o pacote do SIPAMPL. O Apêndice E.2 contém os passo para a instalação desta biblioteca.

A biblioteca fornece duas funções. `bspline` é usada para calcular valores para a "B-Spline" e `dbspline` para calcular valores para as derivadas da B-Spline com respeito à variável $t$. A sintaxe das funções é:

$$\texttt{bspline}\ (t, k, n, x1, x2, ..., xn, k1, k2, ..., kn)$$

$$\texttt{dbspline}\ (t, j, k, n, x1, x2, ..., xn, k1, k2, ..., kn)$$

onde $t$ e $x1$, ... ,$xn$ são variáveis do AMPL e correspondem respectivamente ao parâmetro das funções base e aos coeficientes da "B-Spline". $k$, $n$, $k1, \ldots, kn$ são constantes, onde $k$ é o grau da "B-Spline", $n$ é o número de coeficientes e $k1, \ldots, kn$ são os nós. $j$ indica a $j$-ésima derivada da "B-Spline" com respeito à variável $t$ $\left(\frac{\partial^j B}{\partial t^j}\right)$.

As funções `bspline` e `dbspline` retornam ao AMPL o valor da "B-Spline" em $t$ e, se pedido, as primeiras e segundas derivadas da "B-Spline" com respeito ao parâmetro e coeficientes. Quando no AMPL se usam funções externas, carregadas dinamicamente, tem de usar-se o comando `function` para declarar as funções usadas antes das chamadas às funções externas.

### 5.1.3 A base de dados dos problemas

A base de dados do SIPAMPL contém uma grande variedade de problemas, desde os de programação linear até aos não lineares. Alguns problemas dependem de parâmetros que podem ser modificados pelo utilizador de forma a gerar outros problemas resultando por vezes em problemas de maior dimensão. A maior parte destes problemas é de pequena dimensão, com menos de 50 variáveis finitas e 10 variáveis infinitas. A Tabela 5.1 lista os problemas da base de dados. Na tabela, "Problema" indica o nome do ficheiro, de extensão `.mod`, que contém o problema, "nx" é o número de variáveis finitas ($n$), "nt" é o número de variáveis infinitas ($p$), "nrx" é o número de restrições finitas ($q$) e "nrt" é o número de restrições infinitas ($m$) (veja-se a definição (1.1.1)).

Foram codificados dez problemas de robótica descritos em [52] assim como os dez problemas de desenho óptimo de sinais de [156]. Dezasseis problemas do conjunto de problemas de Watson (veja-se [164] e [22]) e sete problemas, quadráticos e não lineares, descritos por Price ([111]) foram também codificados. No problema S, identificado como *priceS*, o autor considerou $T$ como sendo o conjunto $[0, 1]^p$, mas as soluções apresentadas são do conjunto $[0, 2]^p$. Este foi o conjunto codificado. Os três problemas de classe $C^1$, identificados como *coopeL*, *coopeM* e *coopeN*, foram retirados de Price e Coope ([113]). Quatorze

[1] *http://www.norg.uminho.pt/aivaz/*

problemas baseados em aproximações de Chebyshev foram retirados de Hettich [54, 55], Reemtsen [116] e Hettich e Gramlich [57]. O problema *hettich8* foi classificado por Hettich [54] como sendo quadrático, mas não existe nenhum termo quadrático na função objectivo. O problema *hettich9* depende de um conjunto de pontos que não foram indicados pelos autores em [57]. Três problemas quadráticos foram retirados de Liu, Teo e Ito [80] e um outro linear de Lin, Fang e Wu ([77, Secção 5.2]). Neste artigo a solução do problema é igual à indicada em [22], mas o problema não é o mesmo, sendo um coeficiente na função objectivo, uma restrição e um limite simples numa das variáveis diferentes do original. O problema foi codificado como está descrito em [77]. Foram também codificados três problemas descritos em Fang and Wu [28] e dois em Ferris e Philpott [29]. Dezoito problemas lineares foram retirados do artigo de León, Sanmatias e Vercher [71]. Os problemas *leon13* e *leon18* correspondem respectivamente aos problemas 1 e 2 de [101]. Dois problemas de planeamento da produção foram codificados de acordo com a forma descrita por Li e Wang [75] e Wang e Fang [161]. Um problema de Tanaka [138] e dois de Polak [104] foram também codificados. O problema *hettich10* foi retirado de [69]. Os exemplos 1 e 2 de [139] também foram codificados. De facto, o exemplo 2 (*teo2*) apareceu num artigo anterior de Gonzaga *et al.* [44] e está incorrectamente descrito em [139]. O exemplo 2 também surge correctamente descrito em [63, 140]. A base de dados contém também os três problemas de Blankenship e Falk [10] e os problemas de desenho de filtros, num total de sete, propostos por Potchinkov em [110]. Nos problemas *blankenship2* e *blankenship3*, obtidos de [10], o intervalo $T$ é um intervalo do tipo $[0, +\infty[$. O problema usado por Powell para mostrar que o seu algoritmo generalizado de Karmarkar [2] pode convergir para uma solução não óptima, também foi codificado da forma apresentada em [142]. Foi também codificado o problema exemplo, descrito em [136], usado por Still para demonstrar a importância da inclusão dos pontos de fronteira do conjunto $T$ na discretização do problema de PSI. Foi também codificado um problema proposto por Anderson e Lewis em [3, pág 267]. Em [45] o autor propõe 24 problemas de PSI, dos quais 17 já se encontravam codificados. Os restantes 7 problemas foram codificados. Foram codificados 17 problemas obtidos de [50] e um de [172]. Em [172] os autores apresentam uma versão correcta do problema `hettich10` que foi novamente codificado, dando origem ao problema `hettich10c`. Quatro problemas de [67] foram codificados e um de [70]. Os dois exemplos descritos no manual do MATLAB [85] também se encontram codificados.

Pontos iniciais e resultados, quando indicados pelos autores, foram incluídos nos ficheiros dos problemas.

Nos problemas de robótica as restrições infinitas são do tipo

$$lb \leq g(x,t) \leq ub.$$

Uma vez que alguns dos métodos implementados apenas permitem restrições simples do tipo $\leq$ estas foram substituídas por duas restrições do tipo

$$\begin{aligned} -g(x,t) &\leq -lb \\ g(x,t) &\leq ub. \end{aligned}$$

Esta substituição deu origem aos problemas *elke1std*, ... , *elke7std* que correspondem aos problemas *elke1*, ... , *elke7*.

| Problema | nx | nt | nrx | nrt | Problema | nx | nt | nrx | nrt |
|---|---|---|---|---|---|---|---|---|---|
| andreson1 | 3 | 2 | 0 | 1 | blankenship1 | 2 | 1 | 0 | 1 |
| blankenship2 | 2 | 4 | 0 | 4 | blankenship3 | 3 | 2 | 3 | 3 |
| coopeL | 2 | 1 | 0 | 1 | coopeM | 2 | 1 | 1 | 1 |
| coopeN | 2 | 1 | 0 | 1 | elke1 | 9 | 1 | 0 | 10 |
| elke2 | 9 | 1 | 0 | 10 | elke3 | 9 | 1 | 0 | 10 |
| elke4 | 9 | 1 | 0 | 4 | elke5 | 9 | 1 | 0 | 10 |
| elke6 | 9 | 1 | 0 | 10 | elke7 | 9 | 1 | 0 | 10 |
| elke8 | 9 | 1 | 0 | 7 | elke9 | 9 | 1 | 0 | 7 |
| elke10 | 9 | 1 | 0 | 7 | elke1std | 9 | 1 | 0 | 19 |
| elke2std | 9 | 1 | 0 | 19 | elke3std | 9 | 1 | 0 | 19 |
| elke4std | 9 | 1 | 0 | 7 | elke5std | 9 | 1 | 0 | 19 |
| elke6std | 9 | 1 | 0 | 19 | elke7std | 9 | 1 | 0 | 19 |
| fang1 | 50 | 1 | 0 | 1 | fang2 | 50 | 1 | 0 | 1 |
| fang3 | 50 | 1 | 0 | 1 | ferris1 | 7 | 1 | 0 | 2 |
| ferris2 | 7 | 1 | 0 | 1 | gockenbach1 | 33 | 1 | 120 | 16 |
| gockenbach2 | 33 | 1 | 120 | 16 | gockenbach3 | 33 | 1 | 120 | 16 |
| gockenbach4 | 33 | 1 | 120 | 16 | gockenbach5 | 33 | 1 | 120 | 16 |
| gockenbach6 | 33 | 1 | 120 | 16 | gockenbach7 | 33 | 1 | 120 | 16 |
| gockenbach8 | 33 | 1 | 120 | 16 | gockenbach9 | 33 | 1 | 120 | 16 |
| gockenbach10 | 33 | 1 | 120 | 16 | goerner1 | 4 | 1 | 0 | 2 |
| goerner2 | 5 | 1 | 0 | 2 | goerner3 | 7 | 1 | 0 | 2 |
| goerner4 | 7 | 2 | 0 | 2 | goerner5 | 7 | 2 | 0 | 2 |
| goerner6 | 16 | 2 | 0 | 2 | goerner7 | 8 | 2 | 0 | 2 |
| gugat1 | 9 | 1 | 0 | 4 | gugat2 | 9 | 1 | 0 | 4 |
| gugat3 | 7 | 1 | 2 | 2 | gugat4a | 9 | 1 | 0 | 4 |
| gugat4b | 9 | 1 | 0 | 4 | gugat4c | 9 | 1 | 0 | 4 |
| gugat4d | 9 | 1 | 0 | 4 | gugat4e | 9 | 1 | 0 | 4 |
| gugat4f | 9 | 1 | 0 | 4 | gugat5a | 7 | 1 | 0 | 4 |
| gugat5b | 7 | 1 | 0 | 4 | gugat5c | 7 | 1 | 0 | 4 |
| gugat5d | 7 | 1 | 0 | 4 | gugat5e | 7 | 1 | 0 | 4 |
| gugat5f | 7 | 1 | 0 | 4 | gugat6 | 6 | 1 | 0 | 4 |
| gugat7 | 4 | 1 | 0 | 4 | hettich1 | 9 | 2 | 0 | 2 |
| hettich2 | 3 | 1 | 0 | 2 | hettich3 | 5 | 1 | 0 | 2 |
| hettich4 | 2 | 1 | 0 | 2 | hettich5 | 3 | 2 | 0 | 2 |
| hettich6 | 7 | 2 | 0 | 2 | hettich7 | 7 | 2 | 0 | 2 |
| hettich8 | 5 | 1 | 0 | 2 | hettich9 | 11 | 2 | 0 | 2 |
| hettich10 | 2 | 1 | 0 | 2 | hettich10c | 2 | 1 | 0 | 2 |
| kortanek1 | 2 | 1 | 0 | 1 | kortanek2 | 2 | 2 | 0 | 1 |

Continua na próxima página.

Continuação da Tabela 5.1.

| Problema | nx | nt | nrx | nrt | Problema | nx | nt | nrx | nrt |
|---|---|---|---|---|---|---|---|---|---|
| kortanek3 | 7 | 1 | 0 | 1 | kortanek4 | 8 | 1 | 0 | 1 |
| leon1 | 4 | 1 | 0 | 2 | leon2 | 6 | 1 | 0 | 2 |
| leon3 | 6 | 1 | 0 | 2 | leon4 | 7 | 1 | 0 | 2 |
| leon5 | 8 | 1 | 0 | 2 | leon6 | 5 | 1 | 0 | 2 |
| leon7 | 5 | 1 | 0 | 2 | leon8 | 7 | 1 | 0 | 2 |
| leon9 | 7 | 1 | 0 | 2 | leon10 | 3 | 1 | 0 | 2 |
| leon11 | 3 | 1 | 0 | 2 | leon12 | 2 | 1 | 0 | 1 |
| leon13 | 2 | 1 | 0 | 1 | leon14 | 2 | 1 | 0 | 1 |
| leon15 | 2 | 1 | 0 | 1 | leon16 | 3 | 1 | 0 | 1 |
| leon17 | 3 | 1 | 0 | 1 | leon18 | 2 | 1 | 0 | 1 |
| leon19 | 5 | 1 | 0 | 1 | li1 | 10 | 1 | 0 | 1 |
| li2 | 6 | 1 | 0 | 1 | lin1 | 6 | 2 | 0 | 1 |
| liu1 | 2 | 1 | 0 | 1 | liu2 | 2 | 1 | 0 | 1 |
| liu3 | 16 | 1 | 0 | 2 | matlab1 | 3 | 1 | 0 | 2 |
| matlab2 | 3 | 2 | 0 | 1 | polak1 | 4 | 2 | 0 | 2 |
| polak2 | 4 | 2 | 0 | 2 | potchinkov1 | 298 | 2 | 0 | 4 |
| potchinkov2 | 65 | 3 | 0 | 6 | potchinkov3 | 66 | 2 | 0 | 4 |
| potchinkov4a | 67 | 1 | 19 | 2 | potchinkov4b | 65 | 1 | 19 | 1 |
| potchinkovPL | 122 | 2 | 0 | 4 | potchinkovPLR | 122 | 2 | 0 | 4 |
| powell1 | 2 | 1 | 0 | 1 | priceK | 2 | 1 | 0 | 1 |
| priceS3 | 4 | 3 | 0 | 1 | priceS4 | 4 | 4 | 0 | 1 |
| priceS5 | 4 | 5 | 0 | 1 | priceS6 | 4 | 6 | 0 | 1 |
| priceT | 4 | 3 | 0 | 1 | priceU | 4 | 6 | 0 | 1 |
| reemtsen1 | 11 | 3 | 0 | 2 | reemtsen2 | 10 | 2 | 0 | 2 |
| reemtsen3 | 10 | 2 | 0 | 2 | reemtsen4 | 37 | 2 | 0 | 2 |
| reemtsen5 | 11 | 3 | 0 | 2 | still1 | 2 | 1 | 0 | 1 |
| tanaka1 | 2 | 1 | 1 | 1 | teo1 | 3 | 1 | 0 | 1 |
| teo2 | 3 | 1 | 0 | 1 | userman | 2 | 1 | 1 | 1 |
| watson1 | 2 | 1 | 0 | 1 | watson2 | 2 | 1 | 0 | 1 |
| watson3 | 3 | 1 | 0 | 1 | watson4a | 3 | 1 | 0 | 1 |
| watson4b | 6 | 1 | 0 | 1 | watson4c | 8 | 1 | 0 | 1 |
| watson5 | 3 | 1 | 0 | 1 | watson6 | 2 | 1 | 0 | 1 |
| watson7 | 3 | 2 | 0 | 1 | watson8 | 6 | 2 | 0 | 1 |
| watson9 | 6 | 2 | 0 | 1 | watson10 | 3 | 2 | 0 | 1 |
| watson11 | 3 | 2 | 0 | 1 | watson12 | 3 | 2 | 0 | 1 |
| watson13 | 3 | 2 | 0 | 1 | watson14 | 2 | 1 | 0 | 1 |
| zhou1 | 2 | 1 | 0 | 1 | | | | | |

Tabela 5.1: Problemas na base de dados do SIPAMPL

| Variável | Descrição | PSI |
|---|---|---|
| int nxsip | número de variáveis finitas | $n$ |
| int ntsip | número de variáveis infinitas | $p$ |
| int nxsipc | número de restrições finitas | $q$ |
| int ntsipc | número de restrições infinitas | $m$ |
| real *XBU | Limite superior das variáveis finitas | |
| real *XBL | Limite inferior das variáveis finitas | |
| real *TBU | Limite superior das variáveis infinitas | $\alpha_j$ |
| real *TBL | Limite inferior das variáveis infinitas | $\beta_j$ |
| real *XCBU | Limite superior das restrições finitas | |
| real *XCBL | Limite inferior das restrições finitas | |
| real *TCBU | Limite superior das restrições infinitas | |
| real *TCBL | Limite inferior das restrições infinitas | |

Tabela 5.2: Dados fornecidos pela interface do SIPAMPL

## 5.2 A interface do SIPAMPL

A interface do SIPAMPL é uma extensão da interface do AMPL, permitindo ao utilizador avaliar as restrições finitas/infinitas e aceder a outros dados relacionados com o problema de PSI (número de variáveis finitas, número de variáveis infinitas, etc). Veja-se em [37] como utilizar a interface do AMPL e o Apêndice B.1 que contém passos da instalação da interface do AMPL. A interface do SIPAMPL chama as rotinas de interface do AMPL e com a informação adicional disponível nos ficheiros `.row` e `.col` preenche a estrutura de dados da Tabela 5.2 com os valores respectivos. Na tabela as colunas indicam: "Variável" o nome da variável e o tipo; "Descrição" uma breve descrição da variável e "PSI" a notação usada na definição do problema de PSI (1.1.1).

Para descrever as rotinas de interface do SIPAMPL são necessárias as seguintes definições. Uma vez que o AMPL não suporta problemas de PSI e o processamento inicial do problema (*presolver*) efectuado não tem cuidados especiais com a ordem das declarações de variáveis, as variáveis cujos nomes começam por t e as outras são misturadas. A natureza da PSI requer que estes componentes estejam separados (o algoritmo de resolução de problemas de PSI pode necessitar de alterar umas variáveis, enquanto as outras se encontram fixas). Seja $x_{NLP} = (x_1, x_2, \ldots, x_{\texttt{nvar}})$ o vector das variáveis originais, onde `nvar` é uma variável do AMPL que indica o número total de variáveis do problema. Os componentes $x$ e $t$ têm de ser obtidos de $x_{NLP}$. Sejam esses componentes designados de $x_{SIP}$ e $t_{SIP}$, respectivamente.

Segue-se uma breve descrição das funções do SIPAMPL que suportam a avaliação das funções do problema de PSI. Note-se que os gradientes, Hessianas e Jacobianos são sempre no formato denso, tal como em FORTRAN (formato denso do AMPL).

- `sip_extractx` - Extrai o componente finito da variável inicial completa;
- `sip_extractt` - Como em `sip_extractx` mas para o componente infinito;

- `sip_joinxt` - Une os componentes finito e infinito novamente na variável completa;
- `sip_init` - Inicia as variáveis do problema, aloca memória para os limites simples e copia os limites para a nova zona de memória; esta função procura os nomes das variáveis e restrições para calcular as posições originais das variáveis e restrições finitas e infinitas;
- `sip_free` - Liberta a memória alocada na chamada à função `sip_init`. Apenas a memória é libertada, as variáveis na estrutura `sip` não são reiniciadas;
- `sip_objval` - Calcula o valor da função objectivo;
- `sip_objgrd` - Calcula o vector gradiente da função objectivo;
- `sip_objhes` - Calcula a matriz Hessiana da função objectivo;
- `sip_conval` - Calcula o valor das restrições;
- `sip_jacval` - Calcula o Jacobiano das restrições;
- `sip_conxval` - Calcula as restrições finitas;
- `sip_contval` - Calcula as restrições infinitas;
- `sip_conxgrd` - Calcula o vector gradiente de uma restrição finita;
- `sip_contgrd` - Calcula o vector gradiente de uma restrição infinita;
- `sip_conxhes` - Calcula a matriz Hessiana de uma restrição finita;
- `sip_conthes` - Calcula a matriz Hessiana de uma restrição infinita;

Se as rotinas da interface do AMPL forem necessárias, também podem ser usadas.

### 5.2.1 Interface do MATLAB com o SIPAMPL

Nesta subsecção é feita uma descrição da interface entre o MATLAB [84] e o SIPAMPL. Esta interface permite que problemas codificados em AMPL possam ser resolvidos com o MATLAB. O MATLAB no seu pacote de optimização ([18]) fornece uma função (`fseminf`) para programação semi-infinita. O conjunto de problemas que podem ser resolvidos pelo MATLAB é limitado, uma vez que cada restrição apenas pode conter, no máximo, duas variáveis infinitas. A interface com o MATLAB fica então limitada a problemas com apenas duas variáveis infinitas, uma vez que a formulação apresentada aqui supõem que as variáveis infinitas estão presentes em todas as restrições infinitas.

A interface desenvolvida inclui dois ficheiros. `sipampl.c` é o programa principal que fornece a função MEX, biblioteca executável com a função `sipampl` para o MATLAB ([83]). `sipsolve.m` é um ficheiro MATLAB com um pequeno exemplo de como usar o MATLAB para resolver problemas codificados em SIPAMPL.

A sintaxe da função `sipampl` em MATLAB é:

```
[x0,ntc,xbl,xbu] = sipampl('stub')
                 f = sipampl(x)
           [f,Grad] = sipampl(x)
[c,ceq,K1,...,Kntc,s] = sipampl(x,s)
                        sipampl('msg',x)
```

O comportamento da função MATLAB `sipampl` depende do número de argumentos de entrada e saída. Assim,

- `[x0,ntc,xbl,xbu] = sipampl('stub')` lê o problema de nome `stub` (ficheiro `stub.nl`) e retorna `x0` a aproximação inicial, `ntc` número de restrições infinitas, e os limites simples inferiores e superiores das variáveis $x$, `xbl` e `xbu`, respectivamente;
- `f = sipampl(x)` retorna o valor da função objectivo em `x`.
- `[f,Grad] = sipampl(x)` retorna o valor da função objectivo e o vector gradiente em `x`;
- `[c,ceq,K1,...,Kntc,s] = sipampl(x,s)` calcula as restrições em `x`. `s` é o tamanho do passo para a grelha onde as restrições infinitas são avaliadas (veja-se [18, 85] para mais detalhes), `c` e `ceq` são os vectores com os valores das restrições finitas, `c` para as de desigualdade e `ceq` para as de igualdade. Os `K1,...,Kntc` são `ntc` vectores (ou matrizes) com as restrições infinitas avaliadas na grelha;
- `sipampl('msg',x)` escreve a solução encontrada `x` (através do AMPL) com a mensagem `msg`.

Um exemplo simples do uso desta função é:

```
>> [x0,ntc,xbl,xbu] = sipampl('userman');
>> options = optimset('GradObj', 'on');
>> x = fseminf('sipampl',x0,ntc,'sipampl',[],[],[],[],...
              xbl,xbu,options);
>> sipampl('Solução encontrada pelo MATLAB',x);
```

Estes ficheiros estão também disponíveis através da internet juntamente com a base de dados do SIPAMPL. A instalação da interface com o MATLAB é descrita no Apêndice B.2.

## 5.3 A ferramenta *select*

Em programação semi-infinita, assim como em programação finita, alguns algoritmos são desenvolvidos para resolver determinados problemas com estruturas específicas. Por exemplo, para resolver problemas de programação quadrática deve usar-se um algoritmo específico para programação quadrática. Sendo a base de dados do SIPAMPL geral para a

PSI pode ser desejável a selecção de determinados problemas com uma estrutura própria. A ferramenta *select* permite a selecção de problemas com as características indicadas na Tabela 5.3. Na tabela, "Opção" é a característica do problema que pode ser questionada; "Tipo" é o tipo de dados que a ferramenta *select* espera. Em *lista* a ferramenta *select* apresenta os valores permitidos de uma lista e espera pela selecção. Em *intervalo* a ferramenta *select* pede dois valores, uma para o limite inferior e outro para o limite superior. "Valores permitidos" são os valores aceites para a opção seleccionada. "Omissão" são os valores usados por defeito e "PSI" é a terminologia na definição do problema de PSI (1.1.1).

O exemplo de uma sessão interactiva com a ferramenta *select* pode ser consultada no Apêndice D.

Na codificação dos problemas de PSI, foi colocado, como comentário, o tipo de função objectivo e o tipo de restrições envolvidas. As linhas em comentário

```
# Objective: Quadratic
# Constraints: Linear
```

do exemplo apresentado na Secção 5.1 indicam que a função objectivo é do tipo quadrático e as restrições lineares. Esta informação é a única que deve existir na codificação de problemas de PSI, uma vez que a ferramenta *select* obtém a restante a partir do problema codificado. Esta informação é obrigatória uma vez que em PSI a restrição $x_1 e^t \leq 0$ é considerada linear, no entanto, o AMPL ao testar a não linearidade em todas as funções concluiria que esta mesma restrição em programação finita, sendo todas as variáveis finitas, seria não linear devido ao factor $e^t$.

O facto de a ferramenta *select* não necessitar de mais dados fornecidos pelo utilizador significa que este não precisa de mais cuidados adicionais na classificação dos problemas, evitando possíveis erros. Como a ferramenta *select* chama as rotinas do AMPL, o processo de selecção torna-se mais lento, sendo posteriormente compensado ao nível da exactidão dos resultados e da simplicidade das tarefas exigidas ao utilizador.

As interligações entre a ferramenta *select* e o SIPAMPL são ilustradas na Figura 5.1.

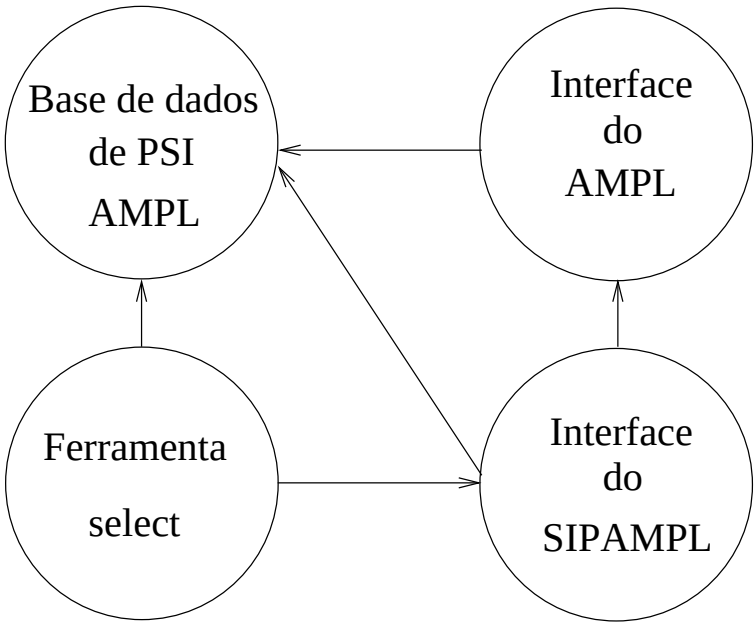


Figura 5.1: Interligações entre a ferramenta *select* e o SIPAMPL

| Opção | Tipo | Valores permitidos | Omissão | PSI |
|---|---|---|---|---|
| Tipo de função objectivo | lista | Linear | Todo tipo | |
| | | Quadrática | | |
| | | Polinomial | | |
| | | Genérica | | |
| | | Todo tipo | | |
| Tipo de funções restrições | lista | Linear | Todo tipo | |
| | | Quadrática | | |
| | | Polinomial | | |
| | | Genérica | | |
| | | Todo tipo | | |
| Número de variáveis finitas | intervalo | Inteiros não negativos | $[0,+\infty]$ | $n$ |
| Número de variáveis infinitas | intervalo | Inteiros não negativos | $[0,+\infty]$ | $p$ |
| Número de restrições finitas | intervalo | Inteiros não negativos | $[0,+\infty]$ | $q$ |
| Número de restrições infinitas | intervalo | Inteiros não negativos | $[0,+\infty]$ | $m$ |
| Limites variáveis finitas | lista | Ambos limites finitos | Tudo | |
| | | Limite inferior finito | | |
| | | Limite superior finito | | |
| | | Nenhum limite finito | | |
| | | Tudo | | |
| Limites variáveis infinitas | lista | Ambos limites finitos | Tudo | $[\alpha, \beta]$ |
| | | Limite inferior finito | | |
| | | Limite superior finito | | |
| | | Nenhum limite finito | | |
| | | Tudo | | |
| Aproximação inicial | lista | Com aproximação inicial | Com ou sem aproximação inicial | |
| | | Sem aproximação inicial | | |
| | | Com ou sem aproximação inicial | | |

Tabela 5.3: Características questionáveis do problema

# Capítulo 6

# Método de discretização implementado

---

O método de discretização, já apresentado na Secção 4.1, foi modificado e implementado para a resolução de problemas de PSI não lineares e merece desde já uma apresentação mais detalhada. Este método foi implementado com base em três versões. Do conjunto dos métodos desenvolvidos, este é o que permite resolver um conjunto maior de problemas, uma vez que pode ser aplicado a problemas que possuam restrições finitas e que tenham mais do que uma variável infinita. A sua única limitação é a de que o intervalo das variáveis infinitas ($T$) seja do tipo produto cartesiano de intervalos com limites finitos.

---

Os primeiros resultados numéricos obtidos pela implementação do método de discretização nas versões modificadas de Hettich e Reemtsen foram apresentados em [150, 151]. A Secção 6.1 apresenta alguma notação e definições necessárias para a descrição dos algoritmos implementados, que são descritos na Secção 6.3. Na Secção 6.2 fazem-se alguns comentários acerca dos problemas codificados que não possuem aproximações iniciais.

## 6.1 Notação e definições

As seguintes definições são necessárias para descrever os algoritmos de discretização apresentados nas próximas secções. Seja $T = [\alpha_1, \beta_1] \times \cdots \times [\alpha_p, \beta_p]$ o produto cartesiano de intervalos.

**Definição 6.1.1** *(Alternativa à Definição 4.1.1) Uma grelha de pontos é um conjunto da forma* $T[h = (h_1, h_2, \ldots, h_p)] = T \cap \{t = (t_1, t_2, \ldots, t_p) : t_i = \alpha_i + jh_i, j = 0, \ldots, n_i; i = 1, \ldots, p\}$, *onde* $n_i = (\beta_i - \alpha_i)/h_i$.

**Definição 6.1.2** *PNL(T[h]) é o seguinte subproblema de programação não linear finito:*

$$\begin{aligned} & \min_{x \in R^n} f(x) \\ s.a \;\; & g_i(x,t) \leq 0, \; i = 1, \ldots, m \\ & h_i(x) = 0, \; i = 1, \ldots, o \\ & h_i(x) \leq 0, \; i = o+1, \ldots, q \\ & \forall t \in T[h] \, , \end{aligned} \tag{6.1.1}$$

**Definição 6.1.3** *Defina-se $R(x_k)$ como o conjunto de todos os pontos da grelha na iteração $k$ que torna pelo menos uma restrição infinita activa, i.e., $R(x_k) = \{t \in T[h^k] : g_i(x_k, t) = 0\}$.*

**Definição 6.1.4** *Defina-se $S(x_k)$ como o conjunto de todos os pontos da grelha na iteração $k$ que torna pelo menos uma restrição infinita activa ou violada, i.e., $S(x_k) = \{t \in T[h^k] : g_i(x_k, t) \geq 0\}$.*

## 6.2 Aproximação inicial

As aproximações iniciais foram codificadas nos ficheiros dos problemas da base de dados, sempre que essa informação estava disponível. No entanto, existem alguns problemas na base de dados do SIPAMPL que não possuem aproximações iniciais. Quando não é disponibilizada uma aproximação inicial à solução de um problema, resolve-se um problema finito obtido a partir do problema de PSI, em que as restrições infinitas são substituídas por um conjunto de restrições finitas que correspondem às restrições infinitas avaliadas numa grelha formada por cinco pontos, por dimensão. A aproximação inicial às variáveis do problema finito é então gerada aleatoriamente entre 0 e 1. A solução encontrada para o problema finito é usada como aproximação inicial para o problema de PSI. Para a resolução do problema finito usou-se o software NPSOL.

## 6.3 Os algoritmos

Duas das versões implementadas do método de discretização resolvem o problema de PSI substituindo o conjunto infinito $T$ por uma grelha de pontos. O método de discretização em geral não garante uma solução exacta, ou quase exacta, para o problema de PSI, no entanto, resolve o problema na grelha final, que é a mais fina. Estas versões começam com uma aproximação inicial para a solução do problema de PSI e resolvem uma sequência de subproblemas finitos baseados numa sequência de grelhas de pontos cada vez mais finas. Em cada iteração exterior o método refina a grelha de uma maneira pré-determinada. Para minimizar o número de restrições em cada subproblema finito, o algoritmo usa apenas uma selecção de pontos em cada grelha. Hettich [55, 57] e Reemtsen [116] apresentam dois

algoritmos que pertencem a classe dos métodos de discretização, e que são aqui modificados com o objectivo de resolverem problemas não lineares de PSI.

Foi ainda implementada outra versão do método proposto por Hettich na qual a grelha de pontos foi substituída por um conjunto de pontos gerados pseudo-aleatoriamente. Este conjunto forma uma sequência de pontos de Halton (veja-se em [111]). O algoritmo começa com um determinado número de pontos pseudo-aleatórios e adiciona em cada refinamento, um segundo conjunto de pontos até o número máximo de pontos ser atingido.

A versão modificada do algoritmo de Hettich é apresentada a seguir.

**Algoritmo 6.3.1** *Versão modificada de Hettich.*

***Passo 0:*** *Defina* $T[h^0]$, *seja* $\widetilde{T}[h^0] = T[h^0]$. *Resolver PNL(*$\widetilde{T}[h^0]$*) e seja* $x_0$ *a solução encontrada. Defina* $R = R(x_0)$.

***Passo k:*** *Se* $S(x_{k-1}) \nsubseteq \widetilde{T}[h^{k-1}]$

*então: Faça* $\widetilde{T}[h^{k-1}] = R \cup S(x_{k-1})$. *Resolva PNL(*$\widetilde{T}[h^{k-1}]$*) e seja* $x_{k-1}$ *a solução encontrada. Se* $R(x_{k-1}) \nsubseteq R$

*então: Seja* $R = R \cup R(x_{k-1})$.

*senão: Adicione um ponto de* $\widetilde{T}[h^{k-1}] \backslash R$ *a* $R$.

*Continue com o* ***Passo k.***

*senão: Se* $k > r$ *pare, senão faça* $\widetilde{T}[h^k] = R \cup S(x_{k-1}) \cup N(S(x_{k-1}))$. *Resolva PNL(*$\widetilde{T}[h^k]$*) e seja* $x_k$ *a solução encontrada. Seja* $R = R \cup R(x_k)$. *Vá para o* ***Passo k+1.***

$r$ é o número máximo de refinamentos e $N(S(x_{k-1}))$ é um conjunto que contém os pontos vizinhos de $S(x_{k-1})$ na grelha total $T[h^k]$ que tornam as restrições infinitas activas ou violadas. Neste algoritmo, o conjunto finito $T[h^k]$ pode surgir de uma grelha de pontos ou de um conjunto de pontos pseudo-aleatórios. $\widetilde{T}[h^k]$ corresponde à selecção efectuada de pontos da grelha (ou do conjunto pseudo-aleatório). PNL($\widetilde{T}[h^k]$) indica o subproblema finito não linear baseado nos pontos seleccionados (Definição 6.1.2).

O algoritmo da versão modificada de Reemtsen é apresentado de seguida com uma notação que corresponde à usada nesta tese.

**Algoritmo 6.3.2** *Versão modificada de Reemtsen.*

***Passo 0:*** *Escolha* $\widetilde{\varepsilon}_k \in (0,1)$ *e* $\widetilde{\delta}_k \geq 0$ $(k = 1, \ldots, r)$. *Seja* $D_0 = T[h^0]$. *Seja também* $i = 0$ *e* $k = 1$.

***Passo 1:*** *Resolva PNL(*$D_i$*) e seja* $x_i$ *a solução encontrada.*

***Passo 2:*** *Se* $k > r$ *pare, senão seja* $B_{i+1} = T[h^k]$, $\varepsilon_{i+1} = \widetilde{\varepsilon}_k$ *e* $\delta_{i+1} = \widetilde{\delta}_k$.
*Se* $x_i$ *resolve a grelha total, i.e.,* $g_j(x_i, t) \leq \delta_{i+1}$ *para todo* $t \in B_{i+1}$ *e* $j = 1, \ldots, m$

*então: Incremente* $k$ *de uma unidade e continue com o* ***Passo 2.***

*senão: Seja* $D_{i+1} = \{t \in B_{i+1} : g_j(x_i, t) \geq -\varepsilon_{i+1}|f(x_i)|, j = 1, \ldots, m\}$. *Incremente* $i$ *de uma unidade e vá para o* ***Passo 1***.

PNL($D_i$) corresponde ao subproblema finito não linear baseado no conjunto de pontos $D_i$. $f(x_i)$ é o valor da função objectivo em $x_i$ (solução do PNL($D_i$)).

Foram realizados dois tipos de refinamentos da grelha, sendo $h^1 = h^0/2$ e $h^{k+1} = h^k/3$ (Hettich [55]). Na versão modificada de Reemtsen o parâmetro $\widetilde{\epsilon}_k$ é actualizado da seguinte forma:

$$\begin{cases} \widetilde{\epsilon}_1 = \widetilde{\epsilon}_0/2^p & \text{para } k = 0 \\ \widetilde{\epsilon}_{k+1} = \widetilde{\epsilon}_k/3^p & \text{para } k \neq 0 \ , \end{cases}$$

onde $p$ é o número de variáveis infinitas.

Nas três versões implementadas usa-se o software NPSOL para resolver os subproblemas finitos.

# Capítulo 7

# Aplicações a casos práticos

---

De entre as aplicações da PSI a casos práticos já mencionadas, duas são descritas com mais detalhe neste capítulo. A primeira aplicação é um caso de planeamento de trajectórias de robôs. A segunda aplicação surge no desenho óptimo de sinais e ilustra como estes podem ser formulados em problemas de PSI.

---

Este capítulo está dividido em duas secções que abordam problemas práticos que podem ser formulados como problemas de PSI. Na Secção 7.1 é apresentado um problema de robótica e na Secção 7.2 um problema de desenho óptimo de conjunto de sinais. Os resultados numéricos aqui apresentados foram obtidos num computador com processador Pentium III a 450Mhz com 128MB de memória RAM e num sistema operativo Linux (Red Hat 5.2) com a versão limitada do AMPL (*Student Version*) número 19991027 (Linux 2.0.18).

## 7.1 Problemas de robótica

Começa-se por introduzir alguma notação e termos usados em robótica para descrever trajectórias de braços robotizados (*manipulators*). Posteriormente apresenta-se o problema de planeamento de trajectórias em que se consideram dois tipos de restrições, dando origem a duas formulações diferentes.

### 7.1.1 Definição de trajectória

Para uma introdução à mecânica e controlo de robôs sugerem-se as referências [23, 102].

Um robô pode ser esquematicamente representado por um conjunto de ligações. O robô representado na Figura 7.1 tem duas ligações. Cada ligação tem um comprimento e massa associados, $d_1$, $d_2$, $m_1$, $m_2$ respectivamente para os comprimentos e massas das duas ligações. Associado ao robô existem aparelhos mecânicos que proporcionam de alguma forma movimento às ligações. A capacidade que o robô tem de posicionar as ligações

indica os *graus de liberdade* (g.d.l.) do robô. No caso representado na Figura 7.1, o robô possui três g.d.l.: um na base da primeira ligação que permite a rotação à volta do eixo $Y$ (indicado pela variável $\theta_1$), outro que controla a abertura da primeira ligação (indicado pela variável $\theta_2$), e o terceiro que controla a abertura da segunda ligação (indicado pela variável $\theta_3$).

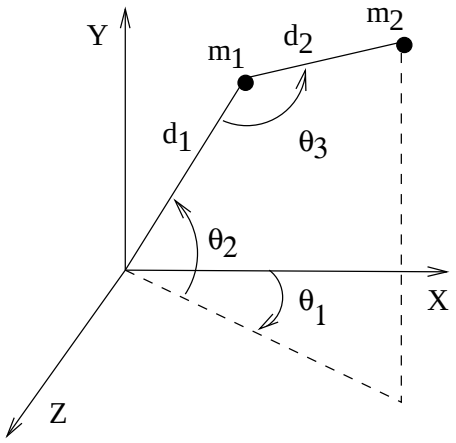


Figura 7.1: Robô com três graus de liberdade

No espaço cartesiano (*cartesian space*) são conhecidas as coordenadas $(x, y, z)$ da posição da parte terminal do robô (*end effector*) e a sua orientação.

No espaço das junções (*joint space*), a especificação dos valores para cada g.d.l. é suficiente para descrever a posição do robô no espaço. Uma trajectória do robô define uma curva paramétrica

$$\theta(\tau) = (\theta_1(\tau), \theta_2(\tau), \ldots, \theta_l(\tau))^T \quad \tau \in [0, \tau_f] \tag{7.1.1}$$

em que $\tau$ é o parâmetro e $l$ é o número de g.d.l..

**Definição 7.1.1** *O Jacobiano, no contexto da robótica, é o produto das matrizes transformação (translação e rotação) que resultam da aplicação das deslocações (comprimento dos braços e ângulos) dos respectivos braços. O Jacobiano é pois a matriz que multiplicada pela coordenada da base do robô resulta na coordenada da parte terminal do mesmo.*

Através do uso do Jacobiano e dada a posição do robô descrita no espaço das junções, é fácil obter a posição no espaço cartesiano. No entanto, dada a posição no espaço cartesiano, pode não ser fácil calcular a posição no espaço das junções, uma vez que o Jacobiano invertido pode ser nulo em alguns pontos.

As condições naturalmente impostas à curva paramétrica são:

$$\sum_{i=1}^{l} \left( \frac{d\theta_i}{d\tau} \right)^2 > 0, \quad \tau \in (0, \tau_f) \tag{7.1.2}$$

$$\frac{d\theta}{d\tau}(0) = \frac{d\theta}{d\tau}(\tau_f) = \mathbf{0} \tag{7.1.3}$$

e

$$\frac{d^2\theta}{d\tau^2}(0), \frac{d^2\theta}{d\tau^2}(\tau_f) \neq \mathbf{0}. \tag{7.1.4}$$

A equação (7.1.2) significa que o robô tem de se encontrar em movimento em pelo menos uma junção. A equação (7.1.3) significa que a posição inicial/final do robô é a posição de descanso (apesar de poder tomar outros valores) e a equação (7.1.4) significa que o robô na posição inicial/final tem de estar em aceleração/desaceleração.

Para simplificar a notação, nas próximas subsecções, usa-se a seguinte notação para as derivadas:

$$\dot{f} = \frac{df}{dt}, \ f' = \frac{df}{d\tau}.$$

### 7.1.2 Trajectórias óptimas com polinómios cúbicos

Em casos práticos, no cálculo da trajectória óptima de robôs, apenas se conhecem determinados pontos de passagem do robô. Esses pontos de passagem, mesmo que conhecidos no espaço cartesiano, podem ser obtidos no espaço das junções, através do uso do Jacobiano invertido. Neste tipo de problemas assume-se que a expressão da curva paramétrica $\theta(\tau)$ (7.1.1) é desconhecida.

Seja $[\theta_1(\tau_1), \theta_1(\tau_2), \ldots, \theta_1(\tau_n)]$, $[\theta_2(\tau_1), \theta_2(\tau_2), \ldots, \theta_2(\tau_n)]$, $\ldots$, $[\theta_l(\tau_1), \theta_l(\tau_2), \ldots, \theta_l(\tau_n)]$ os vectores de pontos onde a trajectória no espaço das junções passa (pontos de passagem na trajectória no espaço das junções). Cada vector contém $n$ pontos na trajectória do espaço das junções que serão referidos como nós. A optimização consiste em encontrar a distribuição de tempo óptima que satisfaça a trajectória no espaço das junções, usando polinómios cúbicos para aproximar a trajectória, sujeito a restrições nos limites da velocidade, aceleração e velocidade de aceleração. Seja $t_1 < t_2 < \cdots < t_n$ uma sequência de instantes onde $t_i$ é o instante em que o robô está na posição $[\theta_1(\tau_i), \theta_2(\tau_i), \ldots, \theta_l(\tau_i)]$. Sejam $d_1 = t_2 - t_1$, $d_2 = t_3 - t_2$, $\ldots$, $d_{n-1} = t_n - t_{n-1}$ os intervalos entre instantes. Na Figura 7.2 apresenta-se uma trajectória de um robô com três g.d.l. em três posições. Seja $Q_{ij}(t)$ o segmento cúbico para a junção $i$ que aproxima $\theta_i(t)$ no intervalo $[t_j, t_{j+1}]$.

Este problema de aproximação de trajectórias por polinómios cúbicos pode ser formu-

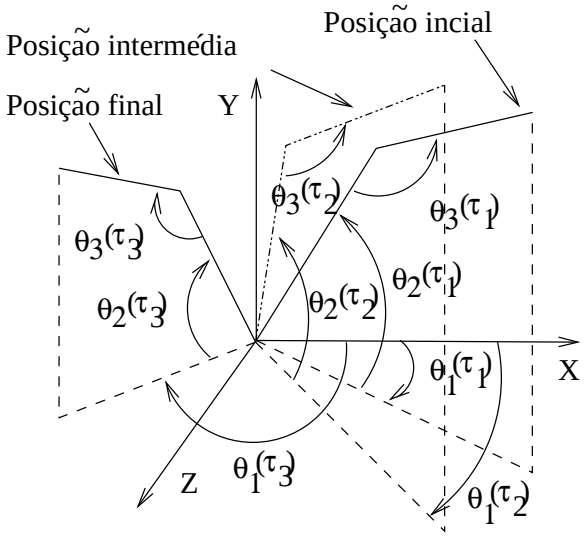


Figura 7.2: Trajectória de robô com três g.d.l.

lado como um problema de PSI com a seguinte forma matemática:

$$\begin{aligned}
&\min_{\mathbf{d}\in R^{n-1}} \sum_{k=1}^{n-1} d_k \\
&s.a \quad (i = 1, \ldots, l; j = 1, \ldots, n-1) \\
&\qquad \left|\dot{Q}_{ij}(t)\right| \leq C_{i,1}, \quad \forall t \in [t_j, t_{j+1}] \\
&\qquad \left|\ddot{Q}_{ij}(t)\right| \leq C_{i,2}, \quad \forall t \in [t_j, t_{j+1}] \\
&\qquad \left|\dddot{Q}_{ij}(t)\right| \leq C_{i,3}, \quad \forall t \in [t_j, t_{j+1}] \\
&\qquad d_j > 0
\end{aligned} \tag{7.1.5}$$

onde $\mathbf{d} = [d_1, \ldots, d_{n-1}]^T$ é o vector dos intervalos entre instantes, $C_{i,1}$, $C_{i,2}$ e $C_{i,3}$ são os limites respectivamente na velocidade, aceleração e velocidade de aceleração, na junção $i$.

A formulação em (7.1.5) é to tipo semi-infinita generalizada uma vez que os intervalos infinitos ($[t_j, t_{j+1}]$, $j = 1, \ldots, n-1$) dependem das variáveis do problema ($d_j$, $j = 1, \ldots, n-1$).

É possível redefinir o problema (7.1.5) como um problema de PSI padrão, através do uso de uma reparametrização da variável $t$. Este assunto será objecto de um estudo posterior.

Note-se que para a definição das funções $Q_{ij}(t)$ será necessário resolver um sistema de equações lineares para o cálculo dos coeficientes do polinómio.

Em [78] foi proposta uma formulação e optimização de trajectórias baseada em polinómios cúbicos idêntica à aqui apresentada. O problema de optimização não foi formulado como um problema de PSI. O problema foi resolvido com o método de Nelder-Mead ([167])

aplicado à função objectivo e uma técnica que converte os vértices não admissíveis em vértices admissíveis através da multiplicação dos vértices não admissíveis por um escalar $\lambda$.

Em [81] e [9] foram propostas formulações baseadas em polinómios cúbicos originando um problema de PSI. Em [81] foi usado o método de discretização de Gonzaga *et al.* [44] para resolver o PSI. As restrições diferem das apresentadas neste trabalho, uma vez que o autor considera restrições na velocidade, momento de torção e intervalo de tempo. Em [9] é usado um algoritmo genético aplicado a uma função de penalidade em que a análise intervalar [94] é usada no cálculo das restrições infinitas.

Em [78] pode encontrar-se um exemplo numérico relacionado com o robô tipo Unimate PUMA 560 com seis junções de rotação. Em [81] é usado um robô Scara com duas ligações e em [9] um robô planar com duas ligações.

### 7.1.3 Parametrização óptima de curvas para desenho de trajectórias de robôs

No problema de parametrização óptima de curvas assume-se que a curva $\theta(\tau)$ é conhecida. Assume-se ainda que a curva dada satisfaz as condições (7.1.2), (7.1.3), (7.1.4) e que, sem perda de generalidade, $\tau_f = 1$.

O problema consiste em determinar a reparametrização óptima para o parâmetro $\tau$, de tal forma que o tempo necessário para percorrer a trajectória seja mínimo. Entretanto, a reparametrização tem de obedecer a determinados limites físicos impostos pelo robô. Dois problemas de optimização podem então ser equacionados dependendo das restrições usadas. Chame-se *Modelo 1* ao problema com limites constantes nas derivadas, que consiste em encontrar uma mudança de variável

$$t = h(\tau), \quad \tau \in [0,1], \tag{7.1.6}$$

i.e.,

$$\tau = h^{-1}(t), \quad t \in [0, t_f] \tag{7.1.7}$$

tal que

$$h(1) \text{ é mínimo} \tag{7.1.8}$$
$$h(0) = 0 \tag{7.1.9}$$
$$h'(\tau) > 0 \quad \tau \in [0,1] \tag{7.1.10}$$
$$\left|\dot{\theta}_i(\tau)\right| \leq C_{i,1} \tag{7.1.11}$$
$$\left|\ddot{\theta}_i(\tau)\right| \leq C_{i,2} \tag{7.1.12}$$
$$|\dddot{\theta}_i(\tau)| \leq C_{i,3} \;\; i = 1, \ldots, l \;\; \forall \tau \in [0,1] \tag{7.1.13}$$

onde $C_{i,1}$, $C_{i,2}$, $C_{i,3}$ são constantes e representam respectivamente os limites da velocidade, aceleração e velocidade de aceleração na junção $i$.

O *Modelo 2* é obtido através da substituição das equações (7.1.11), (7.1.12), (7.1.13) por

$$|F_i(\tau)| \leq C_i \quad \tau \in [0,1], \quad i = 1, \ldots, l \tag{7.1.14}$$

onde $F_i$ é o momento de torção da junção $i$ do robô.

O problema pode ser formulado em termos das derivadas da função desconhecida $h$. Seja

$$g(\tau) = h'(\tau). \tag{7.1.15}$$

A expressão para o momento de torção do robô é

$$F_i(\tau) = J_i n_i \ddot{\theta}_i(\tau) + B_i n_i \dot{\theta}_i(\tau) + \frac{1}{n_i}\left(\sum_{j=1}^{l} I_{ij}(\theta)\ddot{\theta}_j(\tau) + \sum_{j=1}^{l}\sum_{k=1}^{l} C_{ijk}(\theta)\dot{\theta}_j(\tau)\dot{\theta}_k(\tau) + d_i(\theta)\right) \tag{7.1.16}$$

onde para cada junção $i$ do robô:

$J_i$=inércia do motor ($J_i > 0,\ i = 1, \ldots, l$);
$n_i$=razão de engrenagem; (*Gear ratio*)
$B_i$=coeficiente de viscosidade ($B_i > 0,\ i = 1, \ldots, l$);
$(I_{ij}(\theta))_{i,j=1,\ldots,l}$=matriz inércia (definida positiva);
$(C_{ijk}(\theta))_{i,j,k=1,\ldots,l}$=Tensão de Coriolis;
$d_i(\theta)$=Torção gravitacional.

Usando a regra de derivação composta e as igualdades (7.1.7) e (7.1.15) tem-se

$$\dot{\theta}_i(\tau) = \frac{d}{dt}\left\{\theta_i(h^{-1}(t))\right\} = \frac{\theta_i'(\tau)}{h'(\tau)} = \frac{\theta_i'(\tau)}{g(\tau)}, \tag{7.1.17}$$

$$\ddot{\theta}_i(\tau) = \frac{\theta_i''(\tau) - \theta_i'(\tau)\frac{h''(\tau)}{h'(\tau)}}{(h'(\tau))^2} = \frac{\theta_i''(\tau) - \theta_i'(\tau)\frac{g'(\tau)}{g(\tau)}}{g^2(\tau)} \tag{7.1.18}$$

e

$$\begin{aligned}\dddot{\theta}_i(\tau) &= \frac{\theta_i'''(\tau) - 3\theta_i''(\tau)\frac{h''(\tau)}{h'(\tau)} + \theta_i'\frac{3(h''(\tau))^2 - h'(\tau)h'''(\tau)}{(h'(\tau))^2}}{(h'(\tau))^3} \\ &= \frac{\theta_i'''(\tau) - 3\theta_i''(\tau)\frac{g'(\tau)}{g(\tau)} + \theta_i'\frac{3(g'(\tau))^2 - g(\tau)g''(\tau)}{g^2(\tau)}}{g^3(\tau)}.\end{aligned} \tag{7.1.19}$$

Usando (7.1.17) e (7.1.18), a equação (7.1.16) pode ser reescrita como

$$F_i(\tau) = J_i n_i \left( \frac{\theta_i''(\tau) - \theta_i'(\tau)\frac{g'(\tau)}{g(\tau)}}{g^2(\tau)} \right) + B_i n_i \frac{\theta_i'(\tau)}{g(\tau)} + \frac{1}{n_i} \left( \frac{u_i(\tau) - v_i(\tau)\frac{g'(\tau)}{g(\tau)}}{g^2(\tau)} + w_i(\tau) \right) \tag{7.1.20}$$

onde

$$\left.\begin{aligned} u_i(\tau) &= \sum_{j=1}^{l} I_{ij}(\theta(\tau))\theta_j''(\tau) + \sum_{j=1}^{l}\sum_{k=1}^{l} C_{ijk}(\theta(\tau))\theta_j'(\tau)\theta_k' \\ v_i(\tau) &= \sum_{j=1}^{l} I_{ij}(\theta(\tau))\theta_j'(\tau) \\ w_i(\tau) &= d_i(\theta(\tau)). \end{aligned}\right\} \tag{7.1.21}$$

Assim o problema referente ao *Modelo 1* pode ser reescrito na forma de PSI padrão como

$$\begin{aligned} \min_{x\in R^n} \ & \int_0^1 g(\tau)d\tau \\ s.a \ \ & g(\tau) > 0 \\ & \left|\dot{\theta}_i(\tau)\right| \leq C_{i,1} \\ & \left|\ddot{\theta}_i(\tau)\right| \leq C_{i,2} \\ & |\dddot{\theta}_i(\tau)| \leq C_{i,3} \ \ i = 1, \ldots, l \\ & \forall \tau \in [0,1] \ , \end{aligned} \tag{7.1.22}$$

e o *Modelo 2* como

$$\begin{aligned} \min_{x\in R^n} \ & \int_0^1 g(\tau)d\tau \\ s.a \ \ & g(\tau) > 0 \\ & |F_i(\tau)| \leq C_i \ \ i = 1, \ldots, l \\ & \forall \tau \in [0,1] \ . \end{aligned} \tag{7.1.23}$$

### 7.1.4 Problemas codificados oriundos da robótica

Por uma questão prática a função $g(\tau)$, em (7.1.15), na definição da função objectivo dos problemas (7.1.22) e (7.1.23), será aproximada por uma "B-Spline" ([12]). Tem-se assim,

$$\begin{aligned}\int_0^1 g(\tau)d\tau &= \int_0^1 B_{k,\xi}(\tau)d\tau \\ &= \int_0^1 \sum_{i=1}^{n} x_i B_{i,k,\xi}(\tau)d\tau \qquad \text{por (5.1.1)} \\ &= \sum_{i=1}^{n} x_i \int_0^1 B_{i,k,\xi}(\tau)d\tau \\ &= \frac{1}{k}\sum_{i=1}^{n} x_i(\xi_{i+k} - \xi_i).\end{aligned}$$

Usando a biblioteca de "B-Splines", já descrita na Subsecção 5.1.2, foram testadas três trajectórias. As trajectórias são as descritas por Haaren-Retagne em [52].

- **Exemplo 1**: Formulação *Modelo 1* com a trajectória definida por

  $$\begin{aligned}\theta_1(\tau) &= 1.5\zeta(\tau) \\ \theta_2(\tau) &= -0.5\sin(4.7\zeta(\tau)) \\ \theta_3(\tau) &= -1.3\zeta(\tau) + 2.6\end{aligned}$$

  onde $\zeta(\tau) = \tau^3(6\tau^2 - 15\tau + 10)$.

- **Exemplo 2**: Formulação *Modelo 2* com a trajectória definida por

  $$\begin{aligned}\theta_1(\tau) &= \frac{4}{3}\pi(\zeta(\tau) - 0.5)^2 \\ \theta_2(\tau) &= -\frac{\pi}{2} + \frac{\pi}{4}\sin(2\pi\zeta(\tau)) \\ \theta_3(\tau) &= \frac{\pi}{2}\zeta(\tau)\end{aligned}$$

  onde $\zeta(\tau) = -2\tau^3 + 3\tau^2$.

- **Exemplo 3**: Formulação *Modelo 1* com as coordenadas do robô dadas por uma "B-Spline" de grau 6

  $$\theta_j(\tau) = \sum_{i=1}^{11} a_{ij} B_{i,6,\xi}(\tau), \quad j = 1, 2, 3$$

  onde os nós $\xi$ são $\xi_1 = \cdots = \xi_6 = 0$, $\xi_7 = 0.1754653$, $\xi_8 = 0.3552259$, $\xi_9 = 0.5061700$, $\xi_{10} = 0.6877756$, $\xi_{11} = 0.8319530$, $\xi_{12} = \cdots = \xi_{17} = 1$.

  Os coeficientes $a_{ij}$ são apresentados na Tabela 7.1.

| $i$ | $j$=1 | $j$=2 | $j$=3 |
|---|---|---|---|
| 1 | 0.8353216 | -0.4971999 | 2.803471 |
| 2 | 0.8353216 | -0.4971999 | 2.803471 |
| 3 | 0.8353216 | -0.4971999 | 2.803471 |
| 4 | 2.294404 | -1.289908 | 2.137097 |
| 5 | 1.506865 | -0.2470876 | 1.881815 |
| 6 | 0.8491478 | 0.1336621 | 2.158783 |
| 7 | 0.5452022 | 0.7745534 | 1.821837 |
| 8 | 0.1477306 | -0.07365886 | 3.067521 |
| 9 | 1.227333 | -0.4258067 | 2.340066 |
| 10 | 1.227333 | -0.4258067 | 2.340066 |
| 11 | 1.227333 | -0.4258067 | 2.340066 |

Tabela 7.1: Coeficientes da "B-Spline" que indicam as coordenadas do robô

| | $\theta_1$ | | | $\theta_2$ | | | $\theta_3$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | $C_{11}$ | $C_{12}$ | $C_{13}$ | $C_{21}$ | $C_{22}$ | $C_{23}$ | $C_{31}$ | $C_{32}$ | $C_{33}$ |
| A | 2 | 8 | 250 | 3 | 18 | 650 | 4 | 50 | 1000 |
| B | 1 | 3 | 100 | 1 | 3 | 100 | 1 | 3 | 100 |
| C | 2 | 8 | 25 | 3 | 18 | 65 | 4 | 50 | 100 |

Tabela 7.2: Limites das derivadas para a formulação *Modelo 1*

As três diferentes possibilidades para os limites das derivadas da formulação *Modelo 1* são indicadas na Tabela 7.2.

As constantes que definem o robô para a formulação *Modelo 2* são dadas na Tabela 7.3, as massas e os comprimentos dos braços são respectivamente $m_1 = 5kg$, $m_2 = 2kg$, $d_1 = 1.5m$, $d_2 = 0.5m$. As equações dinâmicas do robô são omitidas, por uma questão de simplicidade, mas podem ser obtidas do problema codificado.

A combinação dos Exemplos 1, 2 e 3 com os vários limites das derivadas originaram os sete problemas codificados no SIPAMPL. As designações, bem como a correspondência à terminologia usada em [52] são apresentadas na Tabela 7.4. "Problema" é o nome do ficheiro com extensão `.mod`, "Modelo" é o tipo de formulação, tal como foi apresentado na

| $i$ | $J_i$ | $n_i$ | $B_i$ | $C_i$ |
|---|---|---|---|---|
| 1 | $1.98 \times 10^{-4}$ | 62.5 | $1.744 \times 10^{-3}$ | 2.090 |
| 2 | $1.98 \times 10^{-4}$ | 62.5 | $1.744 \times 10^{-3}$ | 0.523 |
| 3 | $1.98 \times 10^{-4}$ | 62.5 | $1.744 \times 10^{-3}$ | 1.045 |

Tabela 7.3: Constantes para a formulação *Modelo 2*

Subsecção 7.1.3 e a última coluna identifica a terminologia usada em [52]. Os problemas foram resolvidos usando nove coeficientes e um parâmetro na "B-Spline". O número de coeficientes pode ser modificado, introduzindo mais nós à "B-Spline". Foram usados cinco nós interiores e a restrição $g(\tau) > 0$, nos problemas (7.1.22) e (7.1.23), foi substituída por $g(\tau) > \varepsilon$, com $\varepsilon = 10^{-2}$ (como em [52]).

| Problema | Modelo | terminologia em [52] |
|---|---|---|
| elke1 | 1 | Ex1A |
| elke2 | 1 | Ex1B |
| elke3 | 1 | Ex1C |
| elke4 | 2 | Ex2 |
| elke5 | 1 | Ex3A |
| elke6 | 1 | Ex3B |
| elke7 | 1 | Ex3C |

Tabela 7.4: Problemas de robótica codificados em AMPL

Foi também codificado o problema de planeamento de trajectórias para o robô Bendix PACS, de [52]. Os problemas foram usados originalmente em [125, 126, 127]. Foram consideradas três trajectórias diferentes, usando a formulação *Modelo 2*. As trajectórias consideradas foram:

- **Trajectória 1**: Interpolação de junções
$$\theta_j(\tau) = q_j^s + \zeta(\tau)(q_j^f - q_j^s)$$
onde $q^s = (0.7, 0.7, 0.1)^T$ e $q^f = (0.4, -0.4, 0.4)^T$.

- **Trajectória 2**: Quarto de círculo
$$\begin{aligned}\theta_1(\tau) =& \frac{\pi}{4} - \frac{\pi}{2}\zeta(\tau)\\ \theta_2(\tau) =& 1\\ \theta_3(\tau) =& 1\end{aligned}$$

- **Trajectória 3**: Linha
$$\begin{aligned}\theta_1(\tau) =& \frac{\pi}{4} - \frac{\pi}{2}\zeta(\tau)\\ \theta_2(\tau) =& \sec\left(-\frac{\pi}{4} - \frac{\pi}{2}\zeta(\tau)\right)\\ \theta_3(\tau) =& 1\end{aligned}$$

com $\zeta(\tau) = -\tau^2(-2\tau + 3)$, para todo o $\tau \in [0, 1]$. Existe um erro em [52] na descrição de $\theta_2(\zeta)$ para a Trajectória 3. É aqui apresentada a versão correcta como descrita em [125].

Mais uma vez, por questões de simplicidade, omitem-se as equações dinâmicas que podem ser consultadas nos problemas codificados.

Este problema origina três novos ficheiros de acordo com o tipo de trajectória, indicados na Tabela 7.5.

| Problema | trajectórias em [52] |
|---|---|
| elke8 | Interpolação de junções |
| elke9 | Quarto de círculo |
| elke10 | Linha |

Tabela 7.5: Problemas do Robô Bendix codificados em AMPL

### 7.1.5 Um exemplo de uso da biblioteca de "B-Splines" em robótica

Alguns problemas de robótica podem ser formulados como uma parametrização de trajectórias de tempo mínimo no espaço das junções [52, 82]. Esta parametrização consiste em encontrar uma função que minimiza o tempo total de viagem de um robô para uma dada trajectória (no espaço de junções do robô). Esta função é usualmente aproximada por uma "B-Spline". O problema pode ser equacionado da seguinte forma (apenas é considerada uma restrição, uma vez que o propósito é ilustrar como as "B-Splines" podem ser usadas no AMPL e não como resolver um problema real de robótica, já analisado na Subsecção 7.1.3)

$$\begin{aligned} \min_{x \in R^n} \quad & \int_0^1 B_{k,\xi}(t)dt \equiv \frac{1}{k} \sum_{i=1}^{n} x_i \left(\xi_{i+k} - \xi_i\right) \\ s.a \quad & B_{k,\xi}(t) \geq 0 \\ & \forall t \in [0,1] \end{aligned}$$

como um problema de PSI. Note-se que a função objectivo não depende da variável $t$. A restrição obriga a que a "B-Spline" seja positiva no intervalo $[0,1]$. Uma possível codificação em AMPL é a seguinte:

```
function bspline;          # declarar as funções externas usadas
#function dbspline; (não usada neste exemplo)
param n:=7;                # número de coeficientes
param k:=4;                # grau da B-Spline
param nk:=11;              # número de nós
param knots{1..nk};        # vector dos nós
var x{1..n};               # coeficientes
var t;                     # parâmetro da ‘‘B-Spline’’

# para poupar algum espaço definimos uma variável
#  que é uma ‘‘B-Spline’’
var g=bspline(t,k,n,{i in 1..n}x[i],
                  {i in 1..nk}knots[i]);

minimize obj:              # função objectivo
  (1/k)*(sum {i in 1..n} (x[i]
```

```
                *(knots[i+k]-knots[i])));

subject to tcons:          # restrição infinita
  g >= 0;

data;

# dados para o problema
param knots :=             # nós da ‘‘B-Spline’’
  1   0
  2   0
  3   0
  4   0
  5   0.25
  6   0.5
  7   0.75
  8   1
  9   1
  10  1
  11  1;
```

### 7.1.6 Resultados numéricos

Os problemas de robótica são os problemas que se encontram na base de dados do SIPAMPL de mais difícil resolução. Para avaliar a robustez do método de discretização, as versões modificadas do método de Hettich [55, 57] e Reemtsen [116] foram usadas para resolver os problemas de robótica da base de dados do SIPAMPL.

A notação usada na Tabela 7.6 é a seguinte:

| | |
|---|---|
| Problema | Nome do problema (ficheiro com a extensão `.mod` na base de dados do SIPAMPL) |
| MRH/MRR | Número médio de restrições nos subproblemas de PNL resolvidos nas versões Hettich/Reemtsen do método. O número de restrições no primeiro problema de PNL não é contabilizado |
| NH/NR | Número de subproblemas de PNL resolvidos nas versões Hettich/Reemtsen |
| fxH/fxR | Valor da função objectivo na solução encontrada na grelha final nas versões Hettich/Reemtsen. |

Em todos os problemas o passo inicial foi de 0.01, resultando em 100 pontos na grelha inicial. Foram feitos dois refinamentos resultando em 600 pontos na grelha final.

Para o problema *elke2* ambos os métodos falharam na convergência para a solução, uma vez que o NPSOL não conseguiu encontrar uma solução admissível porque os limites na velocidade, aceleração e velocidade de aceleração são muito pequenos. Pa-

| Problema | MRH | MRR | NH | NR | fxH | fxR |
|---|---|---|---|---|---|---|
| elke1 | 150 | 545 | 7 | 3 | 1.08 | 1.08 |
| elke2 | — | — | — | — | — | — |
| elke3 | 116.7 | 540 | 4 | 3 | 1.52 | 1.52 |
| elke4 | 50 | 606 | 5 | 3 | 2.50 | 2.50 |
| elke5 | 130 | 675 | 4 | 3 | 2.40 | 2.40 |
| elke6 | 122.5 | 755 | 5 | 3 | 4.49 | 4.52 |
| elke7 | 130 | 455 | 5 | 3 | 2.91 | 2.91 |

Tabela 7.6: Resultados numéricos para os problemas de robótica (método de discretização)

ra o problema *elke7* ambos os métodos não convergem a partir da aproximação inicial $x_0 = (4.85386, \ldots, 4.85386)^T$ proposta por Haaren-Retagne em [52]. Foi usada outra aproximação inicial $x_0 = (2.85386, \ldots, 2.85386)^T$.

Os tempos, em segundos, de utilizador para a resolução dos problemas são apresentados na Tabela 7.7 para ambas as versões implementadas ("H" - versão modificada de Hettich, "R" - versão modificada de Reemtsen).

| Problema | H | R |
|---|---|---|
| elke1 | 2.31 | 2.39 |
| elke3 | 1.43 | 1.96 |
| elke4 | 1.32 | 2.20 |
| elke5 | 10.56 | 17.58 |
| elke6 | 9.35 | 13.79 |
| elke7 | 9.36 | 13.17 |

Tabela 7.7: Tempos de resolução dos problemas de robótica (em segundos)

Os tempos reportados por Haaren-Retagne em [52] foram da ordem de 8 segundos.

Em [52] os problemas foram resolvidos por um método de redução local (veja-se na Secção 4.3) com a implementação de uma técnica de programação quadrática sequencial para resolver os subproblemas finitos resultantes. Os resultados numéricos obtidos por Haaren-Retagne, na Tabela 7.8, são semelhantes aos obtidos pelo método de discretização.

Os resultados numéricos referentes aos problemas *elke8*, *elke9* e *elke10* não devem ser comparados com os obtidos por Haaren-Retagne em [52], uma vez que não são indicadas as aproximações iniciais usadas. Os resultados numéricos obtidos para estes problemas estão indicados na Tabela 7.9, em que $h = 0.01$, resultando em 100 pontos na grelha inicial e 600 na grelha final. Na Tabela 7.9: "Problema" é o nome do problema, "MRH/MRR" é o número médio de restrições na versão Hettich/Reemtsen, "NH/NR" é o número de subproblemas resolvidos, "fxH/fxR" é o valor da função objectivo na solução encontrada e "tH/tR" é o tempo de utilizador, em segundos, da resolução do problema.

Na Tabela 7.10 apresentam-se os valores da função objectivo, na solução encontrada, obtidos por Haaren-Retagne. Esses valores correspondem aos valores obtidos com maior

| Problema | terminologia em [52] | Solução (função objectivo) |
|---|---|---|
| elke1 | Ex1A | 1.08351 |
| elke2 | Ex1B | 2.55783 |
| elke3 | Ex1C | 1.52472 |
| elke4 | Ex2 | 2.48325 |
| elke5 | Ex3A | 2.39669 |
| elke6 | Ex3B | 4.52595 |
| elke7 | Ex3C | 2.91143 |

Tabela 7.8: Resultados numéricos obtidos por Haaren-Retagne em [52]

| Problema | MRH | MRR | NH | NR | fxH | fxR | tH | tR |
|---|---|---|---|---|---|---|---|---|
| elke8 | 81.7 | 696.5 | 4 | 3 | 2.152577 | 2.152595 | 0.85 | 1.28 |
| elke9 | 84.0 | 252.0 | 4 | 3 | 1.634611 | 1.634739 | 0.93 | 1.33 |
| elke10 | 77.0 | 595.0 | 4 | 3 | 2.187568 | 2.187581 | 1.00 | 1.85 |

Tabela 7.9: Resultados numéricos para os problemas de robótica *elke8-10* (método de discretização)

precisão (maior número de coeficientes da "B-Spline").

Os resultados numéricos obtidos pelo método de discretização são semelhantes aos obtidos por Haaren-Retagne. Os tempos de execução reportados por Haaren-Retagne são entre 3 (Trajectória 1, menor dimensão) e 153 (Trajectória 3, maior dimensão) segundos.

## 7.2 Problema de desenho óptimo de sinais

Embora os problemas de desenho óptimo de sinais possuam uma estrutura semi-infinita a sua resolução tem sido efectuada através de uma discretização do problema e uma consequente aplicação de uma técnica finita. Descreve-se de seguida a sua formulação como um problema não linear finito e posteriormente a sua formulação como um problema de PSI.

### 7.2.1 Formulação finita

Um sistema simples de comunicações pode ser representado esquematicamente por um transmissor e um receptor. O transmissor recebe um símbolo, $m$, de um conjunto de M

| Problema | fx |
|---|---|
| elke8 | 2.13345 |
| elke9 | 1.30976 |
| elke10 | 2.24970 |

Tabela 7.10: Resultados numéricos obtidos por Haaren-Retagne para os problemas de robótica *elke8-10*

símbolos, e modula o símbolo a ser transmitido, $s_m(t)$, enviado através de um canal com ruído, $n(t)$. O receptor recebe o sinal modulado adicionado com ruído, $y(t)$, e prevê o símbolo que foi transmitido, $\widehat{m}$. O modelo é representado na Figura 7.3.

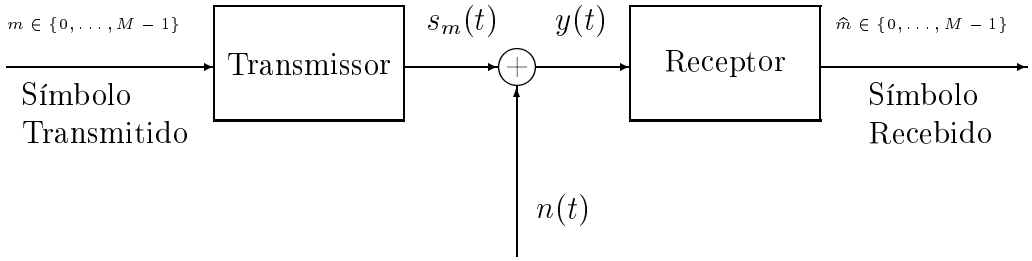


Figura 7.3: Esquema de um sistema simples de comunicações

Em geral, supõe-se que o sinal transmitido $s_m(t)$ é da forma

$$s_m(t) = \sum_{l=1}^{L} \alpha_{ml} \phi_l(t)$$

onde $\phi_l(t)$ são funções base e $\alpha_{ml}$ parâmetros, em que $l = 1, \ldots, L$ e $t \in [0, 1]$, sem perda de generalidade.

O problema de optimização surge quando se pretende encontrar os parâmetros $\alpha_{ml}$ tal que a probabilidade do receptor errar é minimizada ([129]). A minimização da probabilidade é equivalente à maximização da distância mínima entre sinais (veja-se em [42, 68]). Os sinais são forçados a satisfazer algumas restrições, ou no limite do pico da amplitude ou no limite da média de energia. No caso aqui apresentado considera-se uma restrição no limite do pico da amplitude. Segundo Gockenbach e Kearsley ([42]), o problema pode ser equacionado na forma:

$$\min_{d,\alpha} \; -d^2 \tag{7.2.1a}$$

$$\text{s.a} \quad \sum_{t=0}^{\overline{T}-1} K_N(s_{m_1}(t/\overline{T}) - s_{m_2}(t/\overline{T})) \geq d^2, \; m_1 < m_2 \tag{7.2.1b}$$

$$\text{e} \quad s_m(t/\overline{T})^2 \leq C^2, \; t = 0, \ldots, \overline{T}-1, \; m = 0, \ldots, M-1 \tag{7.2.1c}$$

onde $K_N$ é uma função que representa a distância de Kullback-Leibler entre dois sinais e depende da função densidade de probabilidade do ruído. $\overline{T} \in [0, 1]$ é um conjunto discreto

de instantes igualmente espaçados usados no problema discreto. As funções $K_N$ estudadas são as apresentadas em [42, 68] e estão indicadas na Tabela 7.11. Para a função densidade Gaussiana Generalizada a constante $A$ é dada por

$$A = \sqrt{\left(\frac{\sigma^2\Gamma(\frac{1}{4})}{\Gamma(\frac{3}{4})}\right)}$$

onde $\Gamma(.)$ é a função gamma [95].

Este problema de programação não linear finita tem $ML+1$ variáveis e $\frac{M(M-1)}{2} + M\overline{T}$ restrições, podendo originar um problema de grande dimensão.

| Nome | Função densidade | $K_N(\delta)$ |
|---|---|---|
| Gaussiana | $\frac{exp(-t^2/2\sigma^2)}{\sqrt{2\pi\sigma^2}}$ | $\frac{\delta^2}{2\sigma^2}$ |
| Laplaciana | $\frac{exp(-\lvert t\rvert/(\sigma/\sqrt{2}))}{\sqrt{2}\sigma}$ | $\frac{\lvert\delta\rvert}{\sigma/\sqrt{2}} + exp(-\frac{\lvert\delta\rvert}{\sigma/\sqrt{2}}) - 1$ |
| Secante Hiperbólica | $\frac{sech(\pi t/2\sigma)}{2\sigma}$ | $-2ln(sech(\pi\delta/4\sigma))$ |
| Gaussiana Generalizada | $\frac{1}{2\Gamma(5/4)A}exp(-\frac{t^4}{A^4})$ | $\frac{\Gamma^2(3/4)}{\Gamma^2(1/4)}\left(6\frac{\delta^2}{\sigma^2} + \frac{\delta^2}{\sigma^4}\right)$ |
| Cauchy | $\frac{1}{\pi\sigma(1+(t/\sigma)^2)}$ | $ln(1+\delta^2/r\sigma^2)$ |

Tabela 7.11: Funções densidade dos ruídos e distâncias $K_N$

### 7.2.2 Formulação semi-infinita

As restrições do problema (7.2.1) são de facto restrições contínuas e infinitas no tempo, uma vez que devem ser verificadas para todo o $t$ no intervalo $[0,1]$. Assim, o problema original deve ter a forma:

$$\min_{d,\alpha} \ -d^2 \tag{7.2.2a}$$

s.a

$$\int_0^1 K_N(s_{m_1}(z) - s_{m_2}(z))dz \geq d^2, \ m_1 < m_2 \tag{7.2.2b}$$

$$s_m(t)^2 \leq C^2, \ m = 0, \ldots, M-1, \ \forall t \in [0,1] \tag{7.2.2c}$$

definindo um problema de PSI.

A notação com integral é preferida, uma vez que torna o problema mais compacto e claramente o somatório em (7.2.1b) é um integral quando $\overline{T} \to \infty$.

A formulação do problema de PSI tem agora $ML+1$ variáveis e $\frac{M(M+1)}{2}$ restrições.

### 7.2.3 Problemas codificados oriundos do desenho óptimo de sinais

Para codificar o problema de desenho óptimo de sinais, tal como formulado em (7.2.2), o integral (7.2.2b) foi avaliado numericamente. A integração numérica baseada na quadratura Gaussiana não parece ser apropriada, uma vez que o erro de truncatura é elevado. Em vez disso, foi usada a regra do trapézio baseada em $\overline{T}$ pontos.

Foram implementados dois conjuntos de funções base ($L = 2$) e os correspondentes parâmetros [42]:

$$\left\{\sqrt{\tfrac{2}{\overline{T}}}sin(2\pi\omega_1 t), \sqrt{\tfrac{2}{\overline{T}}}sin(2\pi\omega_2 t)\right\}$$

e

$$\left\{\sqrt{\tfrac{2}{\overline{T}}}sin(2\pi\omega_1 t), \sqrt{\tfrac{2}{\overline{T}}}cos(2\pi\omega_1 t)\right\}$$

com $\omega_1 = 10$ e $\omega_2 = 11$. O limite do pico de amplitude é $C = \sqrt{10}$ e $\overline{T} = 50$.

Combinando os dois conjuntos de funções base com as funções densidade do ruído, já apresentadas na Tabela 7.11, resultam dez problemas diferentes que foram codificados na base de dados do SIPAMPL e estão identificados na Tabela 7.12.

| Problema | Funções base | Distâncias de ruído $K_N$ |
|---|---|---|
| gockenbach1 | Seno-Seno | Gaussiana |
| gockenbach2 | Seno-Seno | Laplaciana |
| gockenbach3 | Seno-Seno | Secante Hiperbólica |
| gockenbach4 | Seno-Seno | Gaussiana Generalizada |
| gockenbach5 | Seno-Seno | Cauchy |
| gockenbach6 | Seno-Co-seno | Gaussiana |
| gockenbach7 | Seno-Co-seno | Laplaciana |
| gockenbach8 | Seno-Co-seno | Secante Hiperbólica |
| gockenbach9 | Seno-Co-seno | Gaussiana Generalizada |
| gockenbach10 | Seno-Co-seno | Cauchy |

Tabela 7.12: Problemas codificados na base de dados do SIPAMPL (desenho óptimo de sinais)

Como não são conhecidas aproximações iniciais para estes problemas, a base de dados do SIPAMPL gera aleatoriamente aproximações iniciais para os $\alpha_{ml}$ no intervalo $[-15, 15]$.

### 7.2.4 Resultados numéricos

Os resultados numéricos obtidos permitem concluir que as duas versões Hettich e Reemtsen modificadas do método de discretização produzem resultados idênticos. Apresentam-se portanto alguns resultados numéricos para o método de discretização na versão modificada de Hettich aplicado aos problemas de desenho óptimo de sinais.

| Problema | $f(x^*)$ | MR | NSP |
|---|---|---|---|
| gockenbach1 | -1.30e+0 | 497.3 | 4 |
| gockenbach2 | -1.14e+0 | 497.3 | 4 |
| gockenbach3 | -1.09e+0 | 476.0 | 4 |
| gockenbach4 | -2.06e+0 | 454.7 | 4 |
| gockenbach5 | -4.06e-1 | 454.7 | 4 |
| gockenbach6 | -1.86e+0 | 1348.0 | 6 |
| gockenbach7 | -1.62e+0 | 1156.0 | 4 |
| gockenbach8 | -1.60e+0 | 1156.0 | 4 |
| gockenbach9 | -2.97e+0 | 1348.0 | 6 |
| gockenbach10 | -5.90e-1 | 1348.0 | 6 |

Tabela 7.13: Resultados numéricos com $M = 8$

Os parâmetros $\alpha_{ml}$ dos sinais podem ser representados num mapa. Quando $L = 2$ o mapa correspondente é um gráfico a duas dimensões (designado de constelação). Quatro mapas representando constelações são apresentados nas Figuras 7.4, 7.5, 7.6 e 7.7 onde os eixos representam os parâmetros $\alpha_{m1}$ e $\alpha_{m2}$. As figuras correspondem respectivamente às soluções dos problemas *gockenbach1* e *gockenbach2* com $M = 8$ símbolos e *gockenbach7* e *gockenbach10* com $M = 16$ símbolos.

Alguns dados das experiências realizadas são apresentados nas Tabelas 7.13 e 7.14 respectivamente para $M = 8$ e $M = 16$. A primeira coluna contém o nome do problema na base de dados do SIPAMPL. "$f(x^*)$" é o valor da função objectivo na solução encontrada, "MR" é o número médio de restrições nos subproblemas de PNL resolvidos (as restrições do primeiro subproblema não são contabilizadas) e "NSP" é o número de subproblemas resolvidos. A grelha inicial contém 101 pontos e a grelha final tem 601. Note-se que o problema de PNL (7.2.1) tem 428 restrições finitas e a formulação como problema de PSI (7.2.2) tem 36 restrições infinitas para $M = 8$. Quando $M = 16$, a formulação como problema de PNL (7.2.1) tem 920 restrições finitas e a formulação como problema de PSI (7.2.2) tem 136 restrições infinitas.

Em [42, 68] os autores mostram alguns resultados numéricos obtidos com um método do tipo programação quadrática sequencial aplicado ao problema discretizado. Uma vez que as restrições (7.1.23) são diferentes das (7.2.1b), os valores aqui apresentados da função objectivo e os valores publicados em [42, 68] são diferentes. Em [68], para todos os casos, a execução terminou em menos de 10 ($M = 8$) a 15 ($M = 16$) minutos.

No método de discretização aplicado à PSI o tempo de execução para cada problema é inferior a 20 segundos para $M = 8$ e 100 segundos para $M = 16$ para todas as execuções efectuadas. Nas Tabelas 7.15 e 7.16, "Problema" refere-se ao nome do problema e "Tempo" é o tempo máximo de execução de utilizador, em segundos, de cinco execuções do mesmo problema. Recorde-se que a solução inicial é gerada aleatoriamente e portanto o tempo de execução pode ser diferente para duas execuções do mesmo problema. Optou-se por resolver o problema cinco vezes e indicar o maior tempo obtido.

O desenho óptimo de sinais é um problema com muitas soluções óptimas locais e globais.

| Problema | $f(x^*)$ | MR | NSP |
|---|---|---|---|
| gockenbach1 | -5.54e-1 | 994.7 | 4 |
| gockenbach2 | -6.06e-1 | 994.7 | 4 |
| gockenbach3 | -5.55e-1 | 994.7 | 4 |
| gockenbach4 | -8.65e-1 | 994.7 | 4 |
| gockenbach5 | -2.04e-1 | 1024.0 | 5 |
| gockenbach6 | -7.57e-1 | 2802.7 | 4 |
| gockenbach7 | -8.44e-1 | 2802.7 | 4 |
| gockenbach8 | -7.73e-1 | 2802.7 | 4 |
| gockenbach9 | -1.21e+0 | 2802.7 | 4 |
| gockenbach10 | -2.86e-1 | 2616.0 | 5 |

Tabela 7.14: Resultados numéricos com $M = 16$

| Problema | Tempo (segundos) |
|---|---|
| gockenbach1 | 6.58 |
| gockenbach2 | 8.02 |
| gockenbach3 | 5.68 |
| gockenbach4 | 5.82 |
| gockenbach5 | 6.66 |
| gockenbach6 | 11.30 |
| gockenbach7 | 13.50 |
| gockenbach8 | 12.32 |
| gockenbach9 | 12.13 |
| gockenbach10 | 10.82 |

Tabela 7.15: Tempos de execução de utilizador com $M = 8$

| Problema | Tempo (segundos) |
|---|---|
| gockenbach1 | 60.53 |
| gockenbach2 | 75.20 |
| gockenbach3 | 61.76 |
| gockenbach4 | 60.61 |
| gockenbach5 | 78.15 |
| gockenbach6 | 90.85 |
| gockenbach7 | 96.59 |
| gockenbach8 | 74.24 |
| gockenbach9 | 80.51 |
| gockenbach10 | 81.76 |

Tabela 7.16: Tempos de execução de utilizador com $M = 16$

Para obter a solução global, os autores em [42] aplicaram o algoritmo várias vezes até obterem uma solução que se pensa ser global. Em [75, 161] os autores propuseram dois algoritmos de procura de óptimos globais, baseados em algoritmos genéticos e arrefecimento simulado (*simulated annealing*), reclamando a obtenção dos óptimos globais do problema de PSI para os modelos de planeamento de produção por eles descritos.

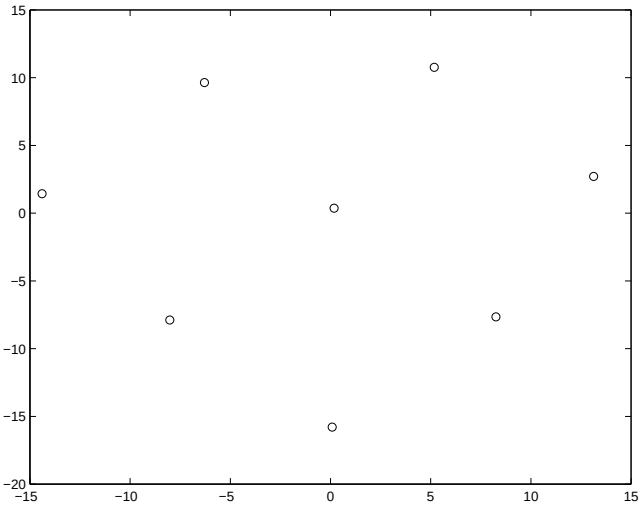


Figura 7.4: Constelação para ruído Gaussiano, $M = 8$, base Seno-Seno

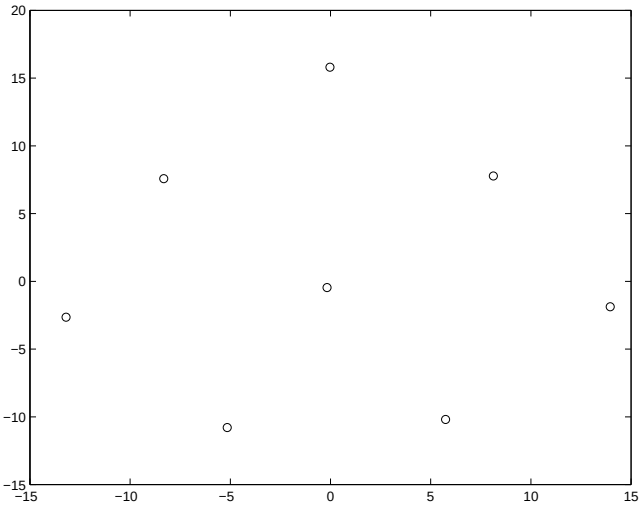


Figura 7.5: Constelação para ruído Laplaciano, $M = 8$, base Seno-Seno

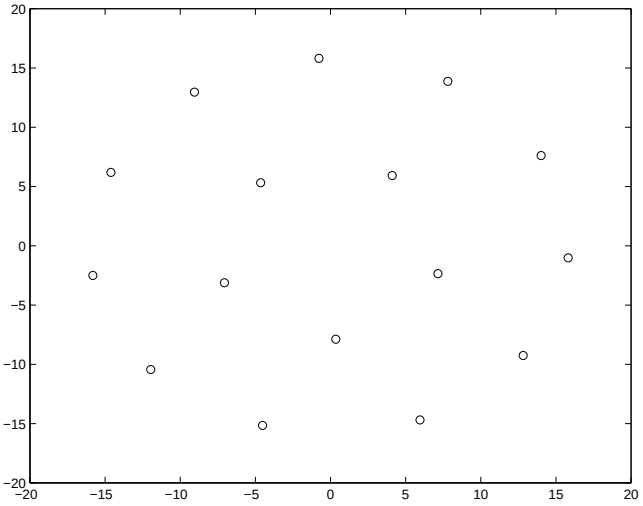


Figura 7.6: Constelação para ruído Secante Hiperbólico, $M = 16$, base Seno-Co-seno

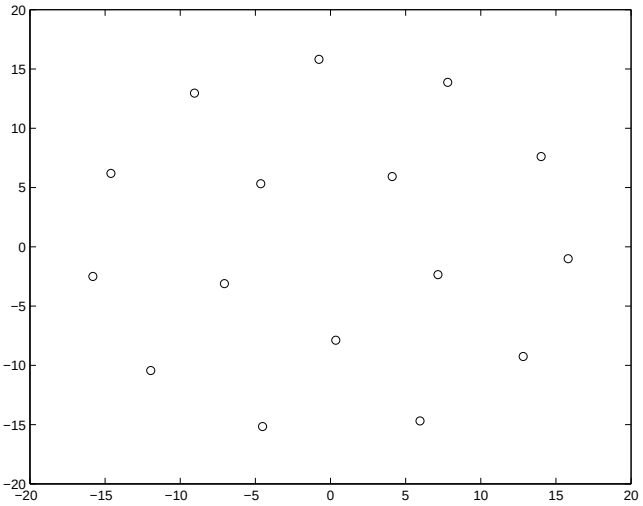


Figura 7.7: Constelação para ruído Cauchy, $M = 16$, base Seno-Co-seno

# Capítulo 8

# Outros métodos para programação semi-infinita não linear

---

Neste trabalho foram desenvolvidos quatro tipos de métodos. O primeiro corresponde ao método de discretização, que foi já apresentado no Capítulo 6. O segundo corresponde ao método de programação quadrática sequencial. O problema quadrático de aproximação ao problema original é resolvido por uma parametrização do problema dual. O terceiro método corresponde à implementação de um método quasi-Newton aplicado a várias funções de penalidade que são obtidas com base na transcrição das restrições infinitas como descrito na Secção 4.4. O quarto é um método primal-dual de pontos interiores aplicado ao problema finito que surge da transcrição das restrições infinitas.

---

Nas três secções seguintes são descritos os três métodos desenvolvidos. Na Secção 8.1 apresenta-se o método de programação quadrática sequencial e na Secção 8.2 o método de penalidade. O método de pontos interiores é descrito na Secção 8.3.

## 8.1 Método de programação quadrática sequencial

O método da programação quadrática sequencial (PQS) para a resolução de um problema não linear consiste em calcular uma sequência de direcções de procura resolvendo uma sequência de subproblemas quadráticos. A função objectivo é usualmente uma aproximação baseada na expansão em série de Taylor da função Lagrangeana, centrada em $(x_k, \lambda_k)$, em que $\lambda_k$ é o vector dos multiplicadores de Lagrange na iteração $k$ e as restrições são aproximações lineares a $g(x_k) \leq 0$. A solução de cada subproblema quadrático é um vector $(d_k, \omega_k)$, com o qual se actualiza $(x_k, \lambda_k)$. O processo repete-se sequencialmente até se atingir uma aproximação razoável à solução do problema não linear. Em programação semi-infinita não linear, o procedimento a seguir é semelhante ao referido embora a resolução do problema quadrático semi-infinito seja feita através da sua formulação dual (veja-se em [154]).

A Subsecção 8.1.1 introduz a técnica de PQS. A aproximação à função dual e o problema dual são apresentados na Subsecção 8.1.2. A função mérito usada para medir o progresso para a solução é descrita na Subsecção 8.1.3. Descreve-se a fórmula BFGS usada para aproximar a inversa da Hessiana da Lagrangeana e o algoritmo implementado na Subsecção 8.1.4.

### 8.1.1 A técnica de programação quadrática sequencial

Considere-se o problema de PSI na sua forma simplificada

$$\begin{aligned} &\min_{x\in R^n} f(x) \\ s.a\ \ &g_i(x,t) \leq 0,\ i=1,\ldots,m, \\ &\forall t \in\ T=[a,b]. \end{aligned} \tag{8.1.1}$$

A função Lagrangeana associada ao problema (8.1.1) pode ser equacionada do seguinte modo

$$\mathcal{L}(x,v) = f(x) + \sum_{j=1}^{m} \int_a^b g_j(x,t) dv_j(t) \tag{8.1.2}$$

onde o integral é do tipo Lebesgue-Stieltjes (veja-se, por exemplo, em [166]) com medidas induzidas pelo vector função $v(t) = (v_1(t), \ldots, v_m(t))^T$, com $v(t) \in \mathcal{V}[a,b;R^m]$ (veja-se a Subsecção 8.1.2).

Seja dado $x_k$, uma aproximação à solução do problema (8.1.1) na iteração $k$. Um problema de programação quadrática, aqui denotado como programação semi-infinita quadrática (PSIQ), é:

$$\begin{aligned} &\min_{d\in R^n} f_Q(d) \equiv \frac{1}{2} d^T H_k d + d^T \nabla f(x_k) \\ &s.a\ \ d^T \nabla_x g_j(x_k,t) + g_j(x_k,t) \leq 0, \\ &j=1,\ldots,m,\ \forall t \in [a,b]\ , \end{aligned} \tag{8.1.3}$$

onde $H_k$ é uma matriz simétrica definida positiva, $\nabla f(x_k)$ é o gradiente da função objectivo e $\nabla_x g_j(x_k,t)$, $j=1,\ldots,m$ são os gradientes das restrições infinitas, com respeito à variável $x$. A técnica de programação semi-infinita quadrática sequencial (PSIQS) resolve uma sequência de subproblemas de PSIQ da forma (8.1.3) para calcular $d_k$, a direcção de procura usada para obter uma nova aproximação, $x_{k+1} = x_k + \alpha_k d_k$, à solução do problema original (8.1.1), onde $\alpha_k$ define o comprimento do passo, calculado por uma estratégia de procura unidimensional.

A técnica de PSIQS é descrita no seguinte algoritmo.

**Algoritmo 8.1.1** *Algoritmo de PSIQS.*

*Passo (a) Dado $x_0$. Seja $k=0$.*

*Passo (b)* *Calcular* $H_k$.

*Passo (c)* *Resolver o problema de PSIQ para obter a direcção de procura* $d_k$.

*Passo (d)* *Se* $d_k = \mathbf{0}$ *então parar.*

*Passo (e)* *Procura unidimensional. Encontrar* $\alpha_k$ *tal que* $x_{k+1} = x_k + \alpha_k d_k$ *produza um decréscimo significativo numa função mérito.*

*Passo (f)* *Se não existir uma diferença significativa entre* $x_{k+1}$ *e* $x_k$ *então parar, sendo* $x_{k+1}$ *uma aproximação à solução. Senão fazer* $k = k + 1$ *e ir para o Passo* (b).

### 8.1.2 Resolução do subproblema de programação semi-infinita quadrática

Alguma da teoria apresentada em [80] é usada para mostrar como o problema de PSIQ pode ser resolvido.

Seja $\mathcal{B}[a, b; R^m]$ um espaço de Banach de todas as funções contínuas de $[a, b]$ em $R^m$ equipado com a norma suprema, e seja $\mathcal{V}[a, b; R^m]$ (dual de $\mathcal{B}[a, b; R^m]$) o espaço de todas as funções de variação normalizadas e limitadas de $[a, b]$ em $R^m$, que são contínuas à direita em $(a, b)$ e nulas em $t = a$. Seja $\mathcal{B}^*$ o cone de $\mathcal{B}[a, b; R^m]$ que consiste em todas as funções não negativas em $\mathcal{B}[a, b; R^m]$. O cone associado $\mathcal{V}^*$ é o conjunto de todas as funções não decrescentes de $\mathcal{V}[a, b; R^m]$. Então tem-se que $v \in \mathcal{V}^*$ se e só se $v$ é uma função de variação não negativa para todas as funções em $\mathcal{B}^*$.

A Lagrangeana associada ao problema (8.1.3) é

$$\mathcal{L}(d, v) = \frac{1}{2} d^T H_k d + d^T \nabla f(x_k) + \sum_{i=1}^{m} \int_a^b \left( d^T \nabla_x g_j(x_k, t) + g_j(x_k, t) \right) dv_j(t) \qquad (8.1.4)$$

onde novamente os integrais são do tipo Lebesgue-Stieltjes com medidas induzidas por $v(t)$.

Assim como na técnica de PQS aplicada à programação finita, o problema quadrático pode resultar num subproblema sem solução admissível, mesmo que o problema original possua pontos admissíveis. Considere-se a seguinte hipótese.

**Hipótese 8.1.1** *Existe algum* $d \in R^n$ *tal que*

$$(CQ) \quad d^T \nabla_x g_j(x_k, t) + g_j(x_k, t) < 0, \quad \forall t \in [a, b] \quad e \quad j = 1, \ldots, m.$$

A região admissível do problema (8.1.3) é convexa e como $H_k$ é definida positiva, a função $f_Q(d)$ é estritamente convexa. Para um $n$ finito tem-se o seguinte lema.

**Lema 8.1.1** *(Lema 3.1 em [80]) Se* $H_k$ *é definida positiva então o problema* (8.1.3) *tem uma solução única* $d^*$.

De acordo com a Hipótese 8.1.1 e usando o Lema 8.1.1, o seguinte teorema obtém-se da teoria da dualidade (veja-se por exemplo [96, Secção 14.8] ou [6, Secção 6.2]).

**Teorema 8.1.1** *(Teorema 3.2 em [80]) Considere válida a Hipótese 8.1.1 e seja $d^*$ a solução única do problema* (8.1.3). *Então*

$$f_Q(d^*) = \max_{v \in \mathcal{V}^*} \min_{d \in R^n} \mathcal{L}(d, v) \tag{8.1.5}$$

*e o máximo do lado direito da equação* (8.1.5) *é atingido para algum $v^* \in \mathcal{V}^*$. Além disso,*

$$f_Q(d^*) = \min_{d \in R^n} \mathcal{L}(d, v^*) \ . \tag{8.1.6}$$

Para um dado $v \in \mathcal{V}\,[a, b; R^m]$ a solução única $d(v)$ do problema

$$\min_{d \in R^n} \ \mathcal{L}(d, v) \tag{8.1.7}$$

é dada por

$$d(v) = -H_k^{-1} \left( \nabla f(x_k) + \sum_{j=1}^{m} \int_a^b \nabla_x g_j(x_k, t) dv_j(t) \right) \tag{8.1.8}$$

que pode ser facilmente obtida resolvendo a equação

$$\nabla_d \mathcal{L}(d, v) = 0 \ .$$

A parte direita da equação (8.1.5) pode ser escrita como um problema dual

$$\min_{v \in \mathcal{V}^*} \ \mathcal{L}^*(v) \tag{8.1.9}$$

onde $\mathcal{L}^*(v)$ é a função dual dada por

$$\begin{aligned} \mathcal{L}^*(v) = & - \mathcal{L}(d(v), v) \\ = & \frac{1}{2} \left( \nabla f(x_k) + \sum_{j=1}^{m} \int_a^b \nabla_x g_j(x_k, t) dv_j(t) \right)^T H_k^{-1} \\ & \left( \nabla f(x_k) + \sum_{j=1}^{m} \int_a^b \nabla_x g_j(x_k, t) dv_j(t) \right) \\ & - \sum_{j=1}^{m} \int_a^b g_j(x_k, t) dv_j(t). \end{aligned} \tag{8.1.10}$$

Esta dedução é apresentada no Apêndice G.

Usando o Teorema 8.1.1 em conjunto com o Lema 8.1.1 obtém-se o seguinte teorema.

**Teorema 8.1.2** *(Teorema 3.3 em [80]) Seja $v^*$ uma solução para o problema* (8.1.9). *Então $d^* = d(v^*)$, dado por* (8.1.8), *é a solução do problema* (8.1.3).

Como as variáveis duais $v$ são medidas de integração em (8.1.9), a sua resolução baseia-se na aproximação de $v(t)$ por uma função linear segmentada em $[a, b]$. Seja $\mathcal{F}^* \subset \mathcal{V}^*$ o conjunto de todas essas funções. Seja ainda

$$a = t_0 < t_1 < \cdots < t_l < t_{l+1} = b \tag{8.1.11}$$

uma partição do conjunto $[a, b]$. Defina-se $|\delta| = \max_{1 \le i \le l+1}(t_i - t_{i-1})$ e seja $v \in \mathcal{V}^*$ uma dada função. Para qualquer partição da forma (8.1.11), defina-se

$$v_\delta(t) = v^{i-1} + \frac{t - t_{i-1}}{t_i - t_{i-1}} \left(v^i - v^{i-1}\right) \tag{8.1.12}$$

para $t \in [t_{i-1}, t_i], i = 1, 2, \ldots, l+1$, com $v^0 = v(t_0)$ e $v^i = v(t_i + 0)$. Tem-se o seguinte lema.

**Lema 8.1.2** *(Lema 4.1 em [80]) Para todo o $v \in \mathcal{V}^*$, seja $v_\delta$ construída de acordo com* (8.1.12). *Se $|\delta| \to 0$, então $v_\delta \xrightarrow{m} v$. Ou seja*

$$\lim_{|\delta| \to 0} \int_a^b \phi(t)^T dv_\delta(t) = \int_a^b \phi(t)^T dv(t), \tag{8.1.13}$$

*para todo o $\phi \in \mathcal{B}[a, b; R^m]$.*

Seja $\mathcal{F}$ o conjunto de todas as funções contínuas de $\mathcal{F}^*$. É claro que

$$\mathcal{F} \subset \mathcal{F}^* \subset \mathcal{V}^*. \tag{8.1.14}$$

**Teorema 8.1.3** *(Teorema 4.2 em [80]) As seguinte igualdades são válidas*

$$\max_{v \in \mathcal{V}^*} \min_{d \in R^n} \mathcal{L}(d, v) = \sup_{w \in \mathcal{F}^*} \min_{d \in R^n} \mathcal{L}(d, w) = \sup_{u \in \mathcal{F}} \min_{d \in R^n} \mathcal{L}(d, u). \tag{8.1.15}$$

Para todo o inteiro $l \ge 1$, seja $\mathcal{F}_l^*$ o conjunto de todas as funções em $\mathcal{F}^*$ que consistem em $l+1$ segmentos lineares em $[a, b]$. O vector função $v(t) \in \mathcal{F}_l^* \subset \mathcal{F}^*$ é definido por

$$v_j(t) = \begin{cases} w_{1j}(t - a), & \text{para } t \in [a, t_1)\,; \\ a_{ij} + w_{i+1j}(t - t_i), & \text{para } t \in [t_i, t_{i+1})\,, i = 1, 2, \ldots, l-1; \\ a_{lj} + w_{l+1j}(t - t_l), & \text{para } t \in [t_l, b]\,; \end{cases} \tag{8.1.16}$$

$j = 1, \ldots, m$, onde

$$a_{ij} = \sum_{p=1}^{i} h_{pj} + \sum_{p=1}^{i} w_{pj}(t_p - t_{p-1}),\ i = 1, \ldots, l \tag{8.1.17}$$

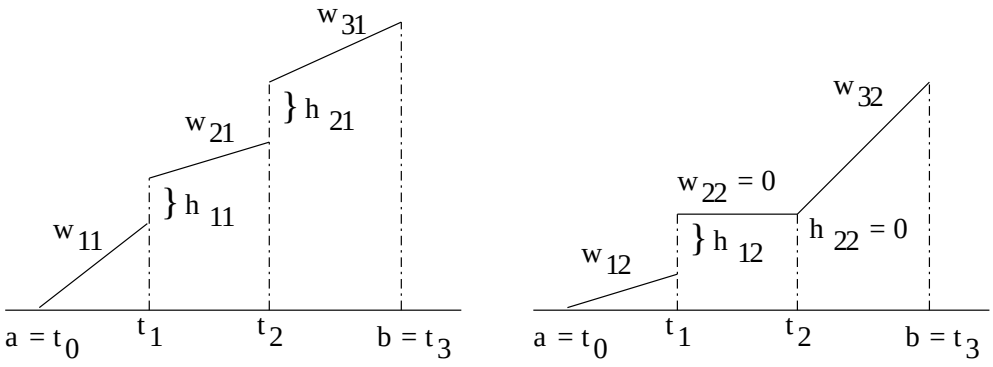


Figura 8.1: Aproximação linear por segmentos de $v(t)$ com $l = 2$ e $m = 2$.

sendo $t_i$, $i = 1, \ldots, l$ os pontos da partição, $h_{ij}$, $i = 1, \ldots, l$, $j = 1, \ldots, m$ os saltos de descontinuidade e $w_{ij}$, $i = 1, \ldots, l+1$, $j = 1, \ldots, m$ os declives dos segmentos lineares. Veja-se a Figura 8.1 que ilustra um caso com $l = 2$ e duas restrições.

Através do Teorema 8.1.3, espera-se que a solução do problema (8.1.9) possa ser aproximada pela minimização de $\mathcal{L}^*(v)$ em $\mathcal{F}_l^*$ com $l$ suficientemente grande.

Com a aproximação (8.1.16) a $v(t)$ e as propriedades dos integrais de Lebesgue-Stieltjes, os integrais em (8.1.10) são aproximados pelas seguintes fórmulas

$$\int_a^b \nabla_x g_j(x,t) dv_j(t) = \sum_{i=1}^{l} \nabla_x g_j(x,t_i) h_{ij} + \sum_{i=1}^{l+1} w_{ij} \int_{t_{i-1}}^{t_i} \nabla_x g_j(x,t) dt \tag{8.1.18a}$$

e

$$\int_a^b g_j(x,t) dv_j(t) = \sum_{i=1}^{l} g_j(x,t_i) h_{ij} + \sum_{i=1}^{l+1} w_{ij} \int_{t_{i-1}}^{t_i} g_j(x,t) dt. \tag{8.1.18b}$$

A solução do problema (8.1.9) é então aproximada por

$$\min_{v \in \mathcal{F}_l^*} \mathcal{L}_l^*(v) \tag{8.1.19}$$

com $l$ suficientemente grande, onde $\mathcal{L}_l^*(v)$ é dado por

$$\begin{aligned} \mathcal{L}_l^*(v) = & \frac{1}{2} \left( \nabla f(x_k) + \sum_{j=1}^{m} c_j \right)^T H_k^{-1} \left( \nabla f(x_k) + \sum_{j=1}^{m} c_j \right) \\ & - \sum_{j=1}^{m} \left( \sum_{i=1}^{l} g_j(x_k,t_i) h_{ij} + \sum_{i=1}^{l+1} w_{ij} \int_{t_{i-1}}^{t_i} g_j(x_k,t) dt \right) \end{aligned} \tag{8.1.20}$$

com

$$c_j = \sum_{i=1}^{l} \nabla_x g_j(x_k,t_i) h_{ij} + \sum_{i=1}^{l+1} w_{ij} \int_{t_{i-1}}^{t_i} \nabla_x g_j(x_k,t) dt.$$

Aconselha-se a referência [80] para obter mais detalhes.

**Nota 8.1.1** *No problema* (8.1.19) *a restrição* $v(t) \in \mathcal{F}_l^*$ *é equivalente às seguintes restrições*

$$\begin{cases} w_{ij} \geq 0 & i = 1, \ldots, l+1, \quad j = 1, \ldots, m \\ h_{ij} \geq 0 & i = 1, \ldots, l, \quad j = 1, \ldots, m \\ t_i \leq t_{i+1} & i = 0, \ldots, l. \end{cases} \tag{8.1.21}$$

**Nota 8.1.2** *O novo problema baseado em* (8.1.20) *e* (8.1.21) *tem* $m + lm + (l+1)m$ *variáveis,* $(l+1)m + lm + 2$ *restrições do tipo limite simples e* $l - 1$ *restrições lineares e deixa de ser um problema quadrático.*

### 8.1.3 Função mérito

O progresso do algoritmo de PSIQS é medido através do uso da seguinte função mérito

$$\phi(x, \mu) = f(x) + \frac{1}{2}\mu \sum_{j=1}^{m} \int_a^b [g_j(x,t)]_+^2 \, dt, \tag{8.1.22}$$

onde $\mu$ é um parâmetro positivo.

Das condições de optimalidade para o problema de PSI quadrático (8.1.3) obtém-se

$$\begin{aligned} d^T \nabla_x \phi(x_k, \mu) &= d^T \nabla f(x_k) + \mu d^T \sum_{j=1}^{m} \int_a^b \nabla_x g_j(x_k, t) \, [g_j(x_k, t)]_+ \, dt \\ &\leq d^T \nabla f(x_k) - \mu \sum_{j=1}^{m} \int_a^b g_j(x_k, t) \, [g_j(x_k, t)]_+ \, dt \\ &= d^T \nabla f(x_k) - \mu \sum_{j=1}^{m} \int_a^b [g_j(x_k, t)]_+^2 \, dt \\ &= \varrho - \mu \sum_{j=1}^{m} \int_a^b [g_j(x_k, t)]_+^2 \, dt \end{aligned}$$

onde $\varrho = -d^T H_k d - \sum_{j=1}^{m} \int_a^b d^T \nabla_x g_j(x_k, t) dv_j(t)$.

O seguinte procedimento foi usado para calcular o parâmetro de penalidade $\mu_{k+1}$ que garante que a direcção $d$ obtida de (8.1.8) é de descida para (8.1.22), em $x_k$.

**Algoritmo 8.1.2** *Cálculo do parâmetro de penalidade.*

*Passo (a) Calcula-se* $\varrho$. *Se* $\varrho$ *é negativo então faz-se* $\mu_{k+1} = \mu_k$, *e termina-se.*

*Passo (b) Calcula-se*

$$\sum_{j=1}^{m} \int_a^b [g_j(x_k,t)]_+^2 \, dt. \tag{8.1.23}$$

*Se* (8.1.23) *é zero, como medida de salvaguarda toma-se* $d = -\nabla_x \phi(x_k, \mu_k)$, *faz-se* $\mu_{k+1} = \mu_k$ *e termina-se.*

*Passo (c) Faz-se*

$$\mu_{k+1} = 10 \max \left\{ \frac{\varrho}{\sum_{j=1}^{m} \int_a^b [g_j(x_k,t)]_+^2 \, dt}, \mu_k \right\} \tag{8.1.24}$$

*e termina-se.*

O tamanho do passo $\alpha_k$ usado é o primeiro elemento da sequência $\{1, \beta, \beta^2, \dots\}$, $\beta = 0.5$, que satisfaz a condição de Armijo

$$\phi(x_k + \alpha_k d, \mu_{k+1}) \leq \phi(x_k, \mu_{k+1}) + \eta \alpha_k d^T \nabla_x \phi(x_k, \mu_{k+1}), \tag{8.1.25}$$

com $\eta \in (0,1)$, fixo.

### 8.1.4 O algoritmo implementado

A matriz $H_k$ é uma aproximação à Hessiana da Lagrangeana. Como se viu em (8.1.20) é necessário calcular uma aproximação à inversa da Hessiana da Lagrangeana, $H_k^{-1}$, em vez de $H_k$. Para aproximar $H_k^{-1}$ foi usada a seguinte fórmula BFGS quasi-Newton

$$H_k^{-1} = \left( I - \frac{s_k y_k^T}{y_k^T s_k} \right) H_{k-1}^{-1} \left( I - \frac{y_k s_k^T}{y_k^T s^k} \right) + \frac{s_k s_k^T}{y_k^T s_k}, \tag{8.1.26}$$

onde

$$s_k = x_k - x_{k-1}, \qquad y_k = \nabla_x \mathcal{L}(x_k, v_k) - \nabla_x \mathcal{L}(x_{k-1}, v_k)$$

sendo $v_k$ os multiplicadores de Lagrange na iteração $k$.

Estamos agora em condições de apresentar o algoritmo completo implementado.

**Algoritmo 8.1.3** *Algoritmo completo de PSIQS.*

*Passo (a) Dados* $x_0$, $\delta_1$ *e* $\delta_2$. *Seja* $k = 0$ *e* $l = 1$.

*Passo (b) Actualizar* $H_k^{-1}$ *usando a fórmula de actualização BFGS quasi-Newton* (8.1.26) *(se* $k = 0$ *então* $H_k$ *=matriz identidade).*

*Passo (c) Seja* $b = \nabla f(x_k)$.

*Passo (d) Resolver o seguinte problema*

$$\min_{\substack{t_i,h_{ij},\, i=1,\ldots,l \\ w_{ij},\, i=1,\ldots,l+1 \\ j=1,\ldots,m}} \mathcal{L}_l^*(t_i,h_{ij},w_{ij}) = \frac{1}{2}\left(b+\sum_{j=1}^{m} c_j\right)^T H_k^{-1}\left(b+\sum_{j=1}^{m} c_j\right) - \sum_{j=1}^{m}\left(\sum_{i=1}^{l} g_j(x_k,t_i)h_{ij} + \sum_{i=1}^{l+1} w_{ij}\int_{t_{i-1}}^{t_i} g_j(x_k,t)dt\right) \tag{8.1.27}$$

$$s.a$$

$$w \geq 0, \quad h \geq 0, \quad t_{i+1}-t_i \geq 0, \quad i=0,\ldots,l$$

*com*

$$c_j = \sum_{i=1}^{l} \nabla_x g_j(x_k,t_i)h_{ij} + \sum_{i=1}^{l+1} w_{ij}\int_{t_{i-1}}^{t_i} \nabla_x g_j(x_k,t)dt$$

*para obter as variáveis da aproximação a* $v(t)$

*Passo (e) Calcular a direcção de procura* $\tilde{d}_l$ *a partir de*

$$\tilde{d}_l(t_i,h_{ij},w_{ij}) = -H_k^{-1}\left(b+\sum_{j=1}^{m} c_j\right) \tag{8.1.28}$$

*onde* $t$, $h$ *e* $w$ *definem a solução do problema* (8.1.27) *na iteração* $l$.

*Passo (f) Se existir uma diferença significativa entre* $\tilde{d}_{l-1}$ *e* $\tilde{d}_l$ *(*$\frac{\|\tilde{d}_{l-1}-\tilde{d}_l\|_2}{\|\tilde{d}_l\|_2} > \delta_1$*) e* $l < n$*, então fazer* $l = l+1$ *e ir para o Passo* (d)*. Senão parar com* $d_k = \tilde{d}_l$ *como uma solução apropriada para o problema* (8.1.3).

*Passo (g) Se* $d_k = \mathbf{0}$ *então parar com* $x_k$ *como solução.*

*Passo (h) Parâmetro de penalidade. Calcular* $\mu_{k+1}$ *usando o Algoritmo 8.1.2.*

*Passo (i) Procura unidimensional. Seja* $x_{k+1} = x_k + \alpha_k d_k$.

*Passo (j) Se não existir uma diferença significativa entre* $x_{k+1}$ *e* $x_k$ *(*$\frac{\|x_{k+1}-x_k\|_2}{\|x_{k+1}\|_2} < \delta_2$*) então parar com* $x_{k+1}$ *como uma solução aproximada. Senão se* $l = n$ *ir para o Passo* (b)*, senão fazer* $l = \max\{1, l-1\}$ *e ir para o Passo* (b).

A iteração interior do algoritmo, indexada por $l$, calcula uma aproximação ao problema dual de PSIQ, usando uma aproximação linear por segmentos às variáveis duais. As iterações exteriores, indexadas por $k$, geram uma sucessão de aproximações ao problema original (8.1.1).

A aproximação inicial para o problema (8.1.27) é a seguinte

$$\begin{cases} t_i = i\frac{b-a}{l+1}, & i = 1, \ldots, l \\ h_{ij} = 1.0, & i = 1, \ldots, l, \quad j = 1, \ldots, m \\ w_{ij} = 1.0, & i = 1, \ldots, l+1, \quad j = 1, \ldots, m \end{cases} \tag{8.1.29}$$

Se para cada resolução do problema (8.1.27) se usar como aproximação inicial os valores (8.1.29), o algoritmo é implementado com uma estratégia de reinício. Por sua vez, se a solução do problema (8.1.27), na iteração $k$, é usada como aproximação inicial do problema, na iteração $k+1$, o algoritmo implementa uma estratégia sem reinício.

O problema (8.1.27) foi resolvido com o pacote de software NPSOL. Quando o cálculo do integral é feito com pouca precisão, o uso de diferenças finitas para aproximar as primeiras derivadas de $\mathcal{L}_l^*$ com respeito às variáveis $t$, $h$, e $w$ não é recomendado. Assim as primeiras derivadas de $\mathcal{L}_l^*$ foram fornecidas ao NPSOL. O Apêndice F contém as expressões destas derivadas.

Outra questão importante na resolução do problema (8.1.27) tem a ver com o cálculo do integral. Uma vez que os intervalos $[t_{i-1}, t_i]$ podem ser de pequena amplitude e para reduzir o número de cálculos de integrais os $l+1$ integrais

$$w_{ij} \int_{t_{i-1}}^{t_i} \nabla_x g_j(x_k, t)dt \quad \text{e} \quad w_{ij} \int_{t_{i-1}}^{t_i} g_j(x_k, t)dt \tag{8.1.30}$$

são substituídos, respectivamente, por

$$\int_a^b \omega_j(t) \nabla_x g_j(x_k, t)dt \quad \text{e} \quad \int_a^b \omega_j(t) g_j(x_k, t)dt \tag{8.1.31}$$

onde

$$\omega_j(t) = w_{ij} \;\text{ se }\; t \in [t_{i-1}, t_i]\,. \tag{8.1.32}$$

### 8.1.5 Cálculo numéricos dos integrais

No cálculo numérico dos integrais usa-se uma fórmula adaptativa do trapézio ou Gaussiana.

A fórmula do trapézio para calcular um integral simples no intervalo $T = [a, b]$, baseada em dois pontos é

$$\int_a^b s(x)dx \approx \frac{b-a}{2}\left[s(a) + s(b)\right].$$

O cálculo do integral com uma dada precisão $\epsilon$ é feito recursivamente da seguinte forma. Começa-se por calcular o integral usando uma fórmula do trapézio de dois pontos no intervalo $[a, b]$ e nos subintervalos $\left[a, \frac{a+b}{2}\right]$, $\left[\frac{a+b}{2}, b\right]$. Se o erro entre as duas aproximações é maior que $\epsilon_T$, i.e., se

$$\left| \int_a^b s(x)dx - \left( \int_a^{\frac{a+b}{2}} s(x)dx + \int_{\frac{a+b}{2}}^b s(x)dx \right) \right| > \epsilon_T$$

então procede-se recursivamente nos subintervalos $\left[a, \frac{a+b}{2}\right]$ e $\left[\frac{a+b}{2}, b\right]$. Caso contrário, aceita-se $\int_a^{\frac{a+b}{2}} s(x)dx + \int_{\frac{a+b}{2}}^b s(x)dx$ como uma boa aproximação do integral.

Desta forma o cálculo da função é feito mais vezes onde a função é menos suave.

Esta ideia adaptativa pode também ser usada com uma fórmula Gaussiana. Apesar de a fórmula Gaussiana ter, em geral, uma melhor precisão, com uma programação cuidada a fórmula do trapézio requer menos cálculos da função, uma vez que na fórmula Gaussiana os valores da função usados num intervalo são descartados quando há uma divisão desse intervalo.

Na Figura 8.2, a primeira restrição infinita para o problema *hettich2* da base de dados do SIPAMPL é usada para ilustrar o cálculo do integral através de uma fórmula Gaussiana adaptativa. A linha sólida representa a função

$$\left[-\left(t^2 - \left(p_1 t + p_2 e^t\right)\right) - d\right]_+$$

com $p_1 = 0.675751$, $p_2 = 0.285805$, $d = 0.536671$ e $t \in [0, 2]$. Os sinais + indicam os pontos onde o intervalo $[0, 2]$ foi dividido e $*$ o valor da função no respectivo ponto. Note-se que no intervalo $[1, 2]$ a função é constante e apenas uma divisão do intervalo foi necessária.

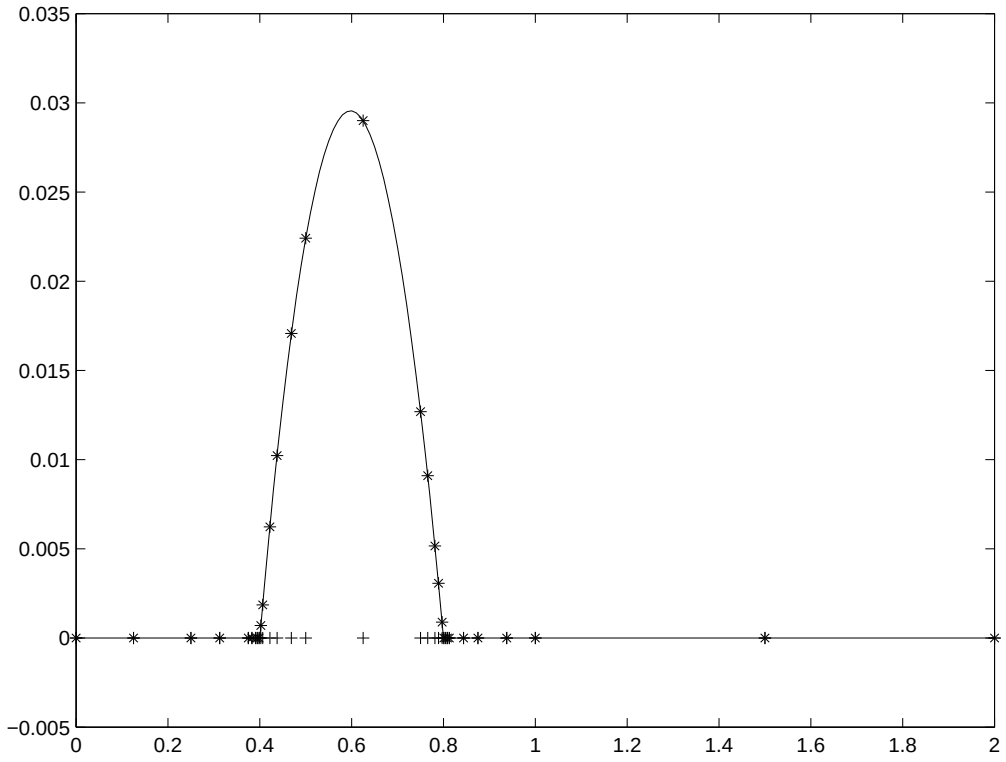


Figura 8.2: Fórmula Gaussiana adaptativa aplicada à primeira restrição infinita do problema *hettich2* da base de dados do SIPAMPL

Na fórmula adaptativa do trapézio o intervalo inicial $[a, b]$ é dividido num pré-determinado

número de subintervalos $ni$ e a fórmula recursiva aplicada a cada subintervalo $\left[a + i * \frac{b-a}{ni}, a + (i+1) * \frac{b-a}{ni}\right]$, com $i = 0, \ldots, ni - 1$.

O número de cálculos da função não é limitado, mas a amplitude mínima de cada intervalo é fixada. O intervalo $[x, y]$ não é subdividido se $y - x < \rho$.

## 8.2 Método de penalidade

O método de penalidade implementado inclui duas versões, ambas baseadas na técnica quasi-Newton (veja-se por exemplo em [96]) aplicada a funções de penalidade. Esta técnica foi apresentada em [152].

Uma das versões implementadas destina-se às funções de penalidade simples, i.e, às funções cujo termo de penalidade não depende dos multiplicadores de Lagrange.

A outra versão implementada destina-se às funções de penalidade em que os multiplicadores de Lagrange estão presentes (Lagrangeana aumentada e exponencial). Neste âmbito, usa-se uma fórmula de actualização para construir aproximações aos multiplicadores de Lagrange.

Na Subsecção 8.2.1 apresenta-se o problema finito que se obtém do problema de PSI através de uma transcrição das restrições infinitas. As funções de penalidade serão apresentadas na Subsecção 8.2.2 e as funções de penalidade baseadas na Lagrangeana aumentada e na função exponencial são apresentadas respectivamente nas Subsecções 8.2.3 e 8.2.4. Na Subsecção 8.2.6 descreve-se o algoritmo implementado.

### 8.2.1 Problema com as restrições transcritas

Considere-se o problema de PSI, apresentado em (8.1.1). As funções $f$ e $g_i$ assumem-se continuamente diferenciáveis em $x$ e $t$.

Foi já referido na Secção 4.4 que o problema (8.1.1) pode ser resolvido usando a transcrição das restrições infinitas em restrições finitas baseadas em integrais. Ainda na sequência do que foi apresentado na Secção 4.4, considere-se o problema aproximado

$$\begin{aligned} &\min_{x \in R^n} \ f(x) \\ s.a \ \ &G_{i,\epsilon}(x) \equiv \int_T g_{i,\epsilon}(x,t)dt = 0, \quad i = 1, \ldots, m \end{aligned} \tag{8.2.1}$$

com $g_{i,\epsilon}(x,t)$ definido em (4.4.8).

Os lemas seguintes são importantes para a análise que se segue.

**Lema 8.2.1** *O seguinte limite é verdadeiro*

$$\lim_{\epsilon \to 0} g_{i,\epsilon}(x,t) = \left[g_i(x,t)\right]_+ . \tag{8.2.2}$$

**Prova.**

$$\begin{aligned}\lim_{\epsilon\to 0} g_{i,\epsilon}(x,t) &= \lim_{\epsilon\to 0}\left(\begin{cases}0, & \text{se } g_i(x,t) < -\epsilon;\\ \frac{(g_i(x,t)+\epsilon)^2}{4\epsilon}, & \text{se } -\epsilon \le g_i(x,t) \le \epsilon;\\ g_i(x,t), & \text{se } g_i(x,t) > \epsilon,\end{cases}\right)\\ &= \begin{cases}0, & \text{se } g_i(x,t) < 0;\\ \lim_{\epsilon\to 0}\frac{(g_i(x,t)+\epsilon)^2}{4\epsilon}, & \text{se } -\epsilon \le g_i(x,t) \le \epsilon;\\ g_i(x,t), & \text{se } g_i(x,t) > 0.\end{cases}\end{aligned} \tag{8.2.3}$$

Uma vez que $\lim_{\epsilon\to 0}\frac{(g_i(x,t)+\epsilon)^2}{4\epsilon} = 0$ quando $-\epsilon < g_i(x,t) < \epsilon$ tem-se que

$$\lim_{\epsilon\to 0} g_{i,\epsilon}(x,t) = \begin{cases}0, & \text{se } g_i(x,t) \le 0;\\ g_i(x,t), & \text{se } g_i(x,t) > 0,\end{cases} = [g_i(x,t)]_+ \tag{8.2.4}$$

■

Considere-se a região admissível para o problema (8.1.1),

$$F = \{x | g_i(x,t) \le 0, i = 1,\ldots,m, \forall t \in T\}, \tag{8.2.5}$$

e a região admissível para o problema (8.2.1),

$$F_\epsilon = \left\{x | \int_T g_{i,\epsilon}(x,t)dt = 0, i = 1,\ldots,m\right\}.$$

**Lema 8.2.2** *A seguinte equação é válida.*

$$\lim_{\epsilon\to 0} F_\epsilon = F. \tag{8.2.6}$$

**Prova.** O resultado segue da aplicação do Lema 8.2.1 e notando-se que

$$g_i(x,t) \le 0 \ \ \forall t \in T \Leftrightarrow [g_i(x,t)]_+ = 0 \ \ \forall t \in T \Leftrightarrow \int_T [g_i(x,t)]_+ \, dt = 0, \tag{8.2.7}$$

para funções contínuas. ■

Considere-se também as seguintes hipóteses:

**Hipótese 8.2.1** $int(F) \neq \emptyset$

**Hipótese 8.2.2** *Seja $x^*$ uma solução óptima para o problema* (8.1.1). *Então existe um ponto $\bar{x} \in int(F)$ tal que*

$$\alpha\bar{x} + (1-\alpha)x^* \in int(F)$$

*para todo o $\alpha \in (0,1]$.*

**Hipótese 8.2.3** *O conjunto $F$ é compacto.*

Os seguintes teoremas relacionam o problema (8.1.1) com o problema (8.2.1).

**Teorema 8.2.1** *(Teorema 2.1 em [63]) Sejam válidas as Hipóteses 8.2.1-8.2.3. Seja $x^*$ uma solução óptima do problema* (8.1.1) *e $x_\epsilon^*$ uma solução óptima do problema* (8.2.1). *Então*

$$\lim_{\epsilon \to 0} f(x_\epsilon^*) = f(x^*).$$

**Teorema 8.2.2** *(Teorema 2.2 em [63]) Sejam válidas as Hipótese 8.2.1-8.2.3. Sejam $x^*$ e $x_\epsilon^*$ como no Teorema 8.2.1. Então a sucessão $\{x_\epsilon^*\}$ tem um ponto de acumulação. Além disso todo o ponto de acumulação da sucessão $\{x_\epsilon^*\}$ é uma solução óptima do problema* (8.1.1).

### 8.2.2 Funções de penalidade simples

Numa técnica de penalidade resolve-se uma sequência de subproblemas sem restrições da forma

$$\min_{x \in R^n} \phi_S(x, \mu) \tag{8.2.8}$$

para uma sequência crescente de valores de $\mu$, onde $\phi_S(x, \mu)$ é uma função genérica de penalidade definida por

$$\phi_S(x, \mu) = f(x) + \mu P(x). \tag{8.2.9}$$

Implementou-se um algoritmo que usa as seguinte funções de penalidade, que são instâncias de (8.2.9) para diferentes $P(x)$

$$\phi_S^1(x, \mu) = f(x) + \mu \sum_{i=1}^{m} \int_T g_{i,\epsilon}(x, t)dt, \tag{8.2.10}$$

$$\phi_S^2(x, \mu) = f(x) + \frac{\mu}{2} \sum_{i=1}^{m} \int_T g_{i,\epsilon}(x, t)^2 dt \tag{8.2.11}$$

e

$$\phi_S^3(x, \mu) = f(x) + \mu \sum_{i=1}^{m} \int_T \left(e^{g_{i,\epsilon}(x,t)} - 1\right) dt. \tag{8.2.12}$$

As seguintes propriedades podem ser provadas para as funções de penalidade (8.2.10-8.2.12).

**Lema 8.2.3** *Para as funções de penalidade (8.2.10-8.2.12) a seguinte propriedade é válida*

$$\begin{cases} P(x)=0 & se \quad x \quad é \ admissível \ para \ o \ problema \ (8.2.1) \\ P(x)>0 & se \quad x \quad não \ é \ admissível \ para \ o \ problema \ (8.2.1). \end{cases}$$

**Prova.** Como $g_{i,\epsilon}(x,t)$ é uma função não negativa tem-se que $P(x) \geq 0$, $\forall x \in R^n$.

Se $x$ é admissível então $x \in F_\epsilon$, i.e., $\int_T g_{i,\epsilon}(x,t)dt = 0$, $i = 1, \ldots, m$ e como $g_{i,\epsilon}(x,t)$ é não negativa e contínua tem-se que $g_{i,\epsilon}(x,t) = 0$, $\forall t \in T$. Claramente $g_{i,\epsilon}(x,t)^2 = 0$ e $e^{g_{i,\epsilon}(x,t)} - 1 = 0$, resultando em $P(x) = 0$ para as funções de penalidade (8.2.10-8.2.12).

Suponha-se agora que $x$ não é admissível. Então, visto que $g_{i,\epsilon}(x,t)$ é não negativa, $g_{i,\epsilon}(x,t) > 0$, para algum $t \in T$ e para algum $i$ em $I = \{1, \ldots, m\}$. Seja $J$ o conjunto dos índices das restrições violadas, i.e.,

$$J = \{i | g_{i,\epsilon}(x,t) > 0 \ \text{ para algum } \ t \in T, \ i \in I\}.$$

Considere-se

$$T_j^+ = \{t \in T | g_{j,\epsilon}(x,t) > 0\}, \ \text{para } j \in J.$$

Para a função de penalidade (8.2.10) (o resultado é semelhante para as outras duas funções de penalidade (8.2.11-8.2.12) e a prova é portanto omitida) tem-se

$$\sum_{i \in I} \int_T g_{i,\epsilon}(x,t)dt = \sum_{j \in J} \int_{T_j^+} g_{j,\epsilon}(x,t)dt$$

que é uma soma de termos positivos e o resultado é imediato. ■

Seja $N_\delta(x_\epsilon^*)$ uma vizinhança de $x_\epsilon^*$ de raio $\delta$, $\delta > 0$.

**Teorema 8.2.3** *Seja $f(x)$ uma função limitada, i.e., $f(x) \neq \infty$, $\forall x \in R^n$, e $x_\epsilon^*$ um minimizante local admissível do problema (8.2.1), i.e., $x_\epsilon^*$ satisfaz as seguintes condições*

$$x_\epsilon^* \in F_\epsilon \Leftrightarrow \int_T g_{i,\epsilon}(x_\epsilon^*, t)dt = 0, \quad i = 1, \ldots, m \tag{8.2.13a}$$

$$f(x_\epsilon^*) \leq f(\hat{x}_\epsilon), \quad \forall \hat{x}_\epsilon \in N_\delta(x_\epsilon^*) \cap F_\epsilon \tag{8.2.13b}$$

*para $\delta > 0$.*

*Então existe um $\bar{\mu} > 0$ tal que para $\mu > \bar{\mu}$, $x_\epsilon^*$ é um minimizante local das funções de penalidade (8.2.10-8.2.12).*

**Prova.** Pretende-se provar que, para $\gamma > 0$

$$\phi_S(x_\epsilon^*, \mu) \leq \phi_S(\bar{x}_\epsilon, \mu), \quad \forall \bar{x}_\epsilon \in N_\gamma(x_\epsilon^*). \tag{8.2.14}$$

Se $\bar{x}_\epsilon$ é admissível então

$$\phi_S(x_\epsilon^*, \mu) = f(x_\epsilon^*) \leq f(\bar{x}_\epsilon) = \phi_S(\bar{x}_\epsilon, \mu), \quad \forall \bar{x}_\epsilon \in N_\gamma(x_\epsilon^*).$$

Se $\bar{x}_\epsilon$ não é admissível e $f(x_\epsilon^*) \leq f(\bar{x}_\epsilon)$, a equação (8.2.14) é verificada para qualquer $\mu \geq 0$. Se $f(x_\epsilon^*) > f(\bar{x}_\epsilon)$, fazendo $\mu > \bar{\mu}$ (finito) em que

$$\bar{\mu} = \frac{f(x_\epsilon^*) - f(\bar{x}_\epsilon)}{P(\bar{x}_\epsilon)}$$

resulta em

$$\phi_S(x_\epsilon^*, \mu) = f(x_\epsilon^*) \leq f(\bar{x}_\epsilon) + \mu P(\bar{x}_\epsilon) = \phi_S(\bar{x}_\epsilon, \mu), \quad \forall \bar{x}_\epsilon \in N_\gamma(x_\epsilon^*).$$

■

**Teorema 8.2.4** *(recíproco do Teorema 8.2.3) Seja $x_\epsilon^*$ uma solução local do problema* (8.2.8) *que seja admissível para o problema* (8.2.1)*, i.e., as duas condições seguintes são válidas*

$$\phi_S(x_\epsilon^*, \mu) \leq \phi_S(\bar{x}_\epsilon, \mu) \quad \forall \bar{x}_\epsilon \in N_\gamma(x_\epsilon^*)$$

*e*

$$P(x_\epsilon^*) = 0.$$

*Então $x_\epsilon^*$ é uma solução local do problema* (8.2.1)*, i.e.,*

$$f(x_\epsilon^*) \leq f(\hat{x}_\epsilon) \quad \forall \hat{x}_\epsilon \in N_\delta(x_\epsilon^*) \cap F_\epsilon.$$

**Prova.** Seja, sem perda de generalidade, $(N_\delta(x_\epsilon^*) \cap F_\epsilon) \subset N_\gamma(x_\epsilon^*)$. Para todo o $\hat{x}_\epsilon$ em $N_\delta(x_\epsilon^*) \cap F_\epsilon$ tem-se que

$$f(x_\epsilon^*) = \phi_S(x_\epsilon^*, \mu) \leq \phi_S(\hat{x}_\epsilon, \mu) = f(\hat{x}_\epsilon)$$

uma vez que $\hat{x}_\epsilon$ é admissível. ■

**Nota 8.2.1** *A função de penalidade* (8.2.11)*, sem o uso da curva de suavização, é uma vez continuamente diferenciável e o uso da aproximação* (4.4.8) *não é necessária. As provas dos Teoremas 8.2.3 e 8.2.4 são semelhantes se a aproximação* (4.4.8) *for omitida.*

### 8.2.3 Função de penalidade baseada na Lagrangeana aumentada

Para ultrapassar a violação da condição de independência linear, Jennings e Teo [63] propuseram a seguinte aproximação ao problema (8.1.1)

$$\begin{aligned} &\min_{x\in R^n} f(x) \\ s.a \quad &\int_T g_{i,\epsilon}(x,t)dt \leq \tau,\ i=1,\ldots,m, \end{aligned} \tag{8.2.15}$$

para $\tau$ suficientemente pequeno e positivo. Seja

$$F_\tau = \left\{ x | \int_T g_{i,\epsilon}(x,t)dt \leq \tau, i=1,\ldots,m \right\},$$

a sua região admissível.

O teorema seguinte relaciona os problemas (8.1.1) e (8.2.15).

**Teorema 8.2.5** *(Teorema 2.3 em [63]) Existe um $\tau(\epsilon)$ tal que para todo o $\tau$, $0 < \tau < \tau(\epsilon)$, toda a solução admissível do problema* (8.2.15) *é também admissível para o problema* (8.1.1).

A função Lagrangeana associada ao problema (8.2.15) é

$$\mathcal{L}(x,\lambda) = f(x) + \sum_{i=1}^{m} \lambda_i \left( \int_T g_{i,\epsilon}(x,t)dt - \tau \right) \tag{8.2.16}$$

onde $\lambda = (\lambda_1,\ldots,\lambda_m)^T$ é o vector dos multiplicadores de Lagrange. Com base nesta função Lagrangeana pode definir-se a seguinte função de penalidade

$$\begin{aligned} \phi_{LA}(x,\lambda,\mu) =& f(x) + \sum_{i=1}^{m} \lambda_i \left( \int_T g_{i,\epsilon}(x,t)dt - \tau \right) \\ &+ \frac{\mu}{2} \sum_{i=1}^{m} \left( \int_T g_{i,\epsilon}(x,t)dt \right)^2, \end{aligned} \tag{8.2.17}$$

e resolver a sequência de subproblemas sem restrições

$$\min_{x\in R^n} \phi_{LA}(x,\lambda,\mu) \tag{8.2.18}$$

para uma sequência crescente de valores positivos de $\mu$.

O método baseado na resolução de (8.2.18) chama-se método dos multiplicadores ou método da Lagrangeana aumentada uma vez que considera a introdução explícita de estimativas dos multiplicadores de Lagrange, em cada etapa da sequência, na função a minimizar. Assim, para cada valor de $\mu$, usa-se uma estimativa do vector $\lambda^*$ e minimiza-se $\phi_{LA}$

em relação a $x$. A solução deste subproblema é depois usada para actualizar a estimativa do vector dos multiplicadores para a etapa seguinte.

A utilização desta função Lagrangeana aumentada, no contexto do método dos multiplicadores, é justificada por duas propriedades. A primeira mostra que quando se conhece o vector dos multiplicadores óptimo $\lambda^*$, um ponto estacionário da função de penalidade (8.2.17) admissível para o problema (8.2.15), é um ponto estacionário de $\phi_{LA}(x, \lambda^*, \mu)$ para qualquer $\mu$ finito. A segunda determina que todo o ponto estacionário de $\phi_{LA}(x, \lambda^*, \mu)$ que verifique as restrições do problema (8.2.15) é um ponto estacionário da Lagrangeana.

Para a análise que se segue defina-se a função de penalidade baseada na Lagrangeana aumentada da seguinte forma geral

$$\phi_{LA}(x, \lambda, \mu) = \mathcal{L}(x, \lambda) + \mu \overline{P}(x). \tag{8.2.19}$$

**Teorema 8.2.6** *Todo o ponto estacionário da função Lagrangeana* (8.2.16) *que seja admissível para* (8.2.15) *é ponto estacionário da função de penalidade* (8.2.17)*, com* $\tau$ *dado pelo Teorema 8.2.5 e* $\epsilon \to 0$*.*

**Prova.** Seja $x_\tau^*$ um ponto admissível e estacionário da Lagrangeana, i.e.,

$$x_\tau^* \in F_\tau \Leftrightarrow \int_T g_{i,\epsilon}(x_\tau^*, t)dt \leq \tau \tag{8.2.20}$$

e para $\lambda^*$,

$$\nabla_x \mathcal{L}(x_\tau^*, \lambda^*) = \nabla f(x_\tau^*) + \sum_{i=1}^{m} \lambda_i^* \int_T \nabla_x g_{i,\epsilon}(x_\tau^*, t)dt = 0. \tag{8.2.21}$$

Para a função de penalidade baseada na Lagrangeana aumentada (8.2.17) tem-se que

$$\nabla \bar{P}(x) = \sum_{i=1}^{m} \int_T g_{i,\epsilon}(x, t)dt \int_T \nabla_x g_{i,\epsilon}(x, t)dt,$$

e como $x_\tau^*$ é um ponto admissível para o problema (8.2.15), usando o Teorema 8.2.5 e com $\epsilon \to 0$ tem-se que $\int_T g_{i,\epsilon}(x_\tau^*, t)dt \to \int_T [g_i(x_\tau^*, t)]_+ dt = 0$, resultando em

$$\nabla \bar{P}(x_\tau^*) \to 0$$

obtendo-se

$$\nabla_x \phi_{LA}(x_\tau^*, \lambda^*, \mu) = \nabla_x \mathcal{L}(x_\tau^*, \lambda^*) + \mu \nabla \bar{P}(x_\tau^*) \to 0,$$

como pretendido. ■

**Teorema 8.2.7** *(recíproco do Teorema 8.2.6) Todo o ponto estacionário da função de penalidade* (8.2.17) *que seja admissível para o problema* (8.2.15) *é ponto estacionário da função Lagrangeana, em que* $\tau$ *é dado pelo Teorema 8.2.5 e* $\epsilon \to 0$*.*

**Prova.** Seja $x_\tau^*$ um ponto estacionário de $\phi_{LA}(x, \lambda, \mu)$,

$$\nabla_x \phi_{LA}(x_\tau^*, \lambda^*, \mu) = \nabla_x \mathcal{L}(x_\tau^*, \lambda^*) + \mu \nabla \bar{P}(x_\tau^*) = 0.$$

Se $x_\tau^*$ é admissível para o problema (8.2.15), usando-se o Teorema 8.2.5, tem-se que

$$\nabla \bar{P}(x_\tau^*) \to 0$$

e o resultado segue de imediato. ■

Para se reconhecer uma solução local do problema (8.2.15) é necessário conhecer o vector dos multiplicadores óptimo ($\lambda^*$). Dado um vector não óptimo de multiplicadores de Lagrange pretende-se calcular aproximações ao vector dos multiplicadores que convergem para $\lambda^*$. Este processo é feito iterativamente através de uma fórmula de actualização.

A fórmula de actualização dos multiplicadores para a função de penalidade (8.2.17) é assim definida por

$$\lambda_{i,k+1} = \lambda_{i,k} + \mu_k \int_T g_{i,\epsilon}(x_k, t)dt, \quad i = 1, \ldots, m. \tag{8.2.22}$$

A motivação desta escolha é baseada no facto de que

$$\begin{aligned}
\nabla_x \mathcal{L}(x_k, \lambda_{k+1}) =& \nabla f(x_k) + \sum_{i=1}^{m} \lambda_{i,k+1} \int_T \nabla_x g_{i,\epsilon}(x_k, t)dt \\
=& \nabla f(x_k) + \sum_{i=1}^{m} \left( \lambda_{i,k} + \mu_k \int_T g_{i,\epsilon}(x_k, t)dt \right) \int_T \nabla_x g_{i,\epsilon}(x_k, t)dt \\
=& \nabla f(x_k) + \sum_{i=1}^{m} \lambda_{i,k} \int_T \nabla_x g_{i,\epsilon}(x_k, t)dt \\
&+ \mu_k \sum_{i=1}^{m} \int_T g_{i,\epsilon}(x_k, t)dt \int_T \nabla_x g_{i,\epsilon}(x_k, t)dt \\
=& \nabla_x \phi_{LA}(x_k, \lambda_k, \mu_k)
\end{aligned}$$

onde

$$\nabla_x g_{i,\epsilon}(x, t) = \begin{cases} 0, & \text{se } g_i(x,t) < -\epsilon; \\ \frac{\nabla_x g_i(x,t)(g_i(x,t)+\epsilon)}{2\epsilon}, & \text{se } -\epsilon \leq g_i(x,t) \leq \epsilon; \\ \nabla_x g_i(x,t), & \text{se } g_i(x,t) > \epsilon, \end{cases} \tag{8.2.23}$$

e $k$ é o índice da iteração.

### 8.2.4 Função de penalidade exponencial

No contexto das técnicas de penalidade, pode definir-se uma função de penalidade exponencial com propriedades mais fortes do que as descritas para a função Lagrangeana aumentada. Assim, à imagem da Subsecção 8.2.3 resolve-se uma sequência de subproblemas

$$\min_{x \in R^n} \phi_E(x, \lambda, \mu) \tag{8.2.24}$$

onde

$$\phi_E(x, \lambda, \mu) = f(x) + \frac{1}{\mu} \sum_{i=1}^{m} \lambda_i \left( e^{\mu \left( \int_T g_{i,\epsilon}(x,t)dt - \tau \right)} - 1 \right) \tag{8.2.25}$$

é a função de penalidade parametrizada por $\mu$, o parâmetro de penalidade. Esta função de penalidade é exacta para o caso finito, se o parâmetro for correctamente escolhido [53].

As condições KKT de primeira ordem para o problema (8.2.15) são

$$\nabla_x \mathcal{L}(x, \lambda) = \nabla f(x) + \sum_{i=1}^{m} \lambda_i \int_T \nabla_x g_{i,\epsilon}(x,t)dt = 0 \tag{8.2.26a}$$

$$\lambda_i \left( \int_T g_{i,\epsilon}(x,t)dt - \tau \right) = 0, \quad i = 1, \ldots, m \tag{8.2.26b}$$

$$\lambda_i \geq 0, \quad i = 1, \ldots, m \tag{8.2.26c}$$

$$\int_T g_{i,\epsilon}(x,t)dt - \tau \leq 0, \quad i = 1, \ldots, m. \tag{8.2.26d}$$

**Teorema 8.2.8** *Seja $x^*$ uma solução local do problema* (8.2.15) *que satisfaz as condições (8.2.26b-8.2.26c). Então existe $\overline{\mu} > 0$ tal que para $\mu > \overline{\mu}$, $x^*$ é um minimizante local da função de penalidade* (8.2.25).

**Prova.** Seja $x^*$ uma solução local do problema (8.2.15), então $\exists \delta > 0$ tal que,

$$f(x^*) \leq f(\hat{x}) \ \ \forall \hat{x} \in N_\delta(x^*) \cap F_\tau. \tag{8.2.27}$$

Pretende-se provar que $\exists \gamma > 0$ tal que,

$$\phi_E(x^*, \lambda^*, \mu) \leq \phi_E(\bar{x}, \lambda^*, \mu) \ \ \forall \bar{x} \in N_\gamma(x^*). \tag{8.2.28}$$

Note-se que, usando (8.2.26b), $\exists \lambda^*$ tal que se tem

$$\begin{aligned}\phi_E(x^*, \lambda^*, \mu) =& f(x^*) + \frac{1}{\mu} \sum_{i=1}^{m} \lambda_i^* \left( e^{\mu \left( \int_T g_{i,\epsilon}(x^*, t) dt - \tau \right)} - 1 \right) \\ =& f(x^*).\end{aligned} \tag{8.2.29}$$

Seja

$$\mathcal{I} = \{1, \ldots, m\}$$

$$\mathcal{J} = \left\{ i \in \mathcal{I} | \int_T g_{i,\epsilon}(\bar{x}, t) dt - \tau > 0 \text{ e } \lambda_i^* > 0 \right\}$$

e

$$\mathcal{K} = \left\{ i \in \mathcal{I} | \int_T g_{i,\epsilon}(\bar{x}, t) dt - \tau < 0 \text{ e } \lambda_i^* > 0 \right\}.$$

No caso em que $\mathcal{K} = \emptyset$ e $\mathcal{J} = \emptyset$ a equação (8.2.28) é trivialmente satisfeita e se $\mathcal{J} \neq \emptyset$ então o termo de penalidade de $\phi_E(\bar{x}, \lambda^*, \mu)$ é positivo e a equação (8.2.28) também é satisfeita.

Considere-se agora o caso em que $\mathcal{K} \neq \emptyset$ ($\mathcal{J} \neq \emptyset$ ou $\mathcal{J} = \emptyset$). A desigualdade

$$\phi_E(x^*, \lambda^*, \mu) \leq \phi_E(\bar{x}, \lambda^*, \mu)$$

é equivalente a

$$f(x^*) \leq f(\bar{x}) + \frac{1}{\mu} \sum_{i \in \mathcal{I}} \lambda_i^* \left( e^{\mu \left( \int_T g_{i,\epsilon}(\bar{x}, t) dt - \tau \right)} - 1 \right), \tag{8.2.30}$$

que pode ser reescrita como

$$\begin{aligned}\frac{1}{\mu} \sum_{j \in \mathcal{J}} \lambda_j^* \left( e^{\mu \left( \int_T g_{j,\epsilon}(\bar{x}, t) dt - \tau \right)} - 1 \right) + \\ \frac{1}{\mu} \sum_{k \in \mathcal{K}} \lambda_k^* \left( e^{\mu \left( \int_T g_{k,\epsilon}(\bar{x}, t) dt - \tau \right)} - 1 \right) \geq f(x^*) - f(\bar{x}).\end{aligned} \tag{8.2.31}$$

Como $\lambda_j^* \left( e^{\mu \left( \int_T g_{j,\epsilon}(\bar{x}, t) dt - \tau \right)} - 1 \right) > 0$ para todo o $j \in \mathcal{J}$, a equação (8.2.31) é satisfeita se

$$\frac{1}{\mu} \sum_{k \in \mathcal{K}} \lambda_k^* \left( e^{\mu \left( \int_T g_{k,\epsilon}(\bar{x}, t) dt - \tau \right)} - 1 \right) \geq f(x^*) - f(\bar{x}). \tag{8.2.32}$$

Mais uma vez a equação (8.2.32) é satisfeita se

$$\frac{1}{\mu}|\mathcal{K}| \min_k\{\lambda_k^*\} \left(e^{\mu \max_k\{\int_T g_{k,\epsilon}(\bar{x},t)dt-\tau\}} - 1\right) \geq f(x^*) - f(\bar{x}), \tag{8.2.33}$$

que é equivalente a

$$\frac{1}{\mu} \left(e^{\mu \max_k\{\int_T g_{k,\epsilon}(\bar{x},t)dt-\tau\}} - 1\right) \geq \frac{f(x^*) - f(\bar{x})}{|\mathcal{K}| \min_k\{\lambda_k^*\}}, \tag{8.2.34}$$

onde $|\mathcal{K}|$ é a cardinalidade de $\mathcal{K}$.

Esta última desigualdade é válida se

$$\frac{1}{e^{\mu}} \left(e^{\mu \max_k\{\int_T g_{k,\epsilon}(\bar{x},t)dt-\tau\}} - e^{\mu}\right) \geq \frac{f(x^*) - f(\bar{x})}{|\mathcal{K}| \min_k\{\lambda_k^*\}}. \tag{8.2.35}$$

Se

$$\frac{f(x^*) - f(\bar{x})}{|\mathcal{K}| \min_k\{\lambda_k^*\}} + 1 \leq 0$$

a equação (8.2.35) é satisfeita para todo o $\mu > 0$, senão o resultado é verificado desde que $\mu > \bar{\mu}$ dado por

$$\bar{\mu} = \frac{\log\left(\frac{f(x^*)-f(\bar{x})}{|\mathcal{K}| \min_k\{\lambda_k^*\}} + 1\right)}{\max_k\{\int_T g_{k,\epsilon}(\bar{x},t)dt - \tau\} - 1}. \tag{8.2.36}$$

■

**Teorema 8.2.9** *Todo o minimizante local da função de penalidade* (8.2.25) *que satisfaz as condições (8.2.26b-8.2.26d) é uma solução local do problema* (8.2.15).

**Prova.** (Semelhante à prova do Teorema 2.2 em [53])

Seja $x^*$ um minimizante local da função de penalidade (8.2.25), com multiplicadores de Lagrange $\lambda_i^* \geq 0$, $i = 1, \ldots, m$ e $\mu > 0$. Então existe um $\delta > 0$ tal que

$$\phi_E(x^*, \lambda^*, \mu) \leq \phi_E(\bar{x}, \lambda^*, \mu), \quad \forall \bar{x} \in N_\delta(x^*). \tag{8.2.37}$$

Pretende-se provar que $f(x^*) \leq f(\hat{x})$, $\forall \hat{x} \in N_\delta(x^*) \cap F_\tau$.

Note-se que

$$\phi_E(x^*, \lambda^*, \mu) = f(x^*) \tag{8.2.38}$$

como mostrado em (8.2.29).

Então, usando (8.2.37) e (8.2.38),

$$f(x^*) \leq f(\hat{x}) + \frac{1}{\mu} \sum_{i=1}^{m} \lambda_i^* \left(e^{\mu(\int_T g_{i,\epsilon}(\hat{x},t)dt-\tau)} - 1\right). \tag{8.2.39}$$

Como $\hat{x}$ é admissível e $\lambda_i^* \geq 0$ tem-se que

$$\sum_{i=1}^{m} \lambda_i^* \left( e^{\mu(\int_T g_{i,\epsilon}(\hat{x},t)dt-\tau)} - 1 \right) \leq 0 \tag{8.2.40}$$

resultando em

$$f(x^*) \leq f(\hat{x}) \ \ \forall \hat{x} \in N_\delta(x^*) \cap F_\tau. \tag{8.2.41}$$

■

A fórmula de actualização para os multiplicadores de Lagrange referente à função de penalidade (8.2.25) é

$$\lambda_{i,k+1} = \lambda_{i,k} e^{\mu_k \left( \int_T g_{i,\epsilon}(x_k,t)dt-\tau \right)}, \quad i = 1, \ldots, m \tag{8.2.42}$$

e é motivada pela seguinte relação

$$\begin{aligned} \nabla_x \mathcal{L}(x_k, \lambda_{k+1}) =& \nabla f(x_k) + \sum_{i=1}^{m} \lambda_{i,k+1} \int_T \nabla_x g_{i,\epsilon}(x_k,t)dt \\ =& \nabla f(x_k) + \sum_{i=1}^{m} \lambda_{i,k} e^{\mu_k \left( \int_T g_{i,\epsilon}(x_k,t)dt-\tau \right)} \int_T \nabla_x g_{i,\epsilon}(x_k,t)dt \\ =& \nabla_x \phi_E(x_k, \lambda_k, \mu_k). \end{aligned}$$

Note-se que os multiplicadores de Lagrange actualizados pela fórmula (8.2.42) aproximam-se de zero quando as restrições não estão activas na solução.

### 8.2.5 Cálculo das funções de penalidade

Os integrais das funções de penalidade são calculados numericamente usando o procedimento já descrito na Subsecção 8.1.5.

### 8.2.6 O algoritmo implementado

No algoritmo de penalidade, baseado nas funções de penalidade propostas, é usada uma estratégia quasi-Newton (QN) com a fórmula BFGS de actualização para a inversa da Hessiana de $\phi$ com o objectivo de calcular soluções dos subproblemas (8.2.8), (8.2.18) e (8.2.24) para cada valor de $\mu$.

Segue-se a descrição do algoritmo.

**Algoritmo 8.2.1** *Algoritmo quasi-Newton para as funções de penalidade.*

*Passo (a)* *Dados $x_0$, $\mu_0$, $\epsilon_0$ (excepto para a função de penalidade (8.2.11)), $\delta_1$, $\delta_2$, $\tau_0$ e $\lambda_0$ (se for usada $\phi_{LA}$ ou $\phi_E$). Seja $i = 0$, $j = 0$, $k = 0$, $y_0 = x_0$.*

*Passo (b)* *Iteração exterior. Seja* $x_0 = y_k$.

*Passo (c)* *Iteração interior. Actualizar* $H_i$ *através da fórmula BFGS (se* $i = 0$ *então* $H_i =$ *Matriz identidade). Seja* $d_i = -H_i \nabla_x \phi$.

*Passo (d)* *Seja* $\alpha_i$ *o comprimento do passo calculado por uma regra do tipo Armijo que reduza significativamente a função de penalidade* $\phi$.

*Passo (e)* *Fazer* $x_{i+1} = x_i + \alpha_i d_i$.

*Passo (f)* *Se não existir uma evolução significativa de* $x_i$ *para* $x_{i+1}$ $\left(\frac{\|x_{i+1}-x_i\|}{\|x_{i+1}\|} < \delta_1\right)$ *então fazer* $k = k+1$, $y_k = x_{i+1}$, $i = 0$, *e ir para o Passo (g). Senão fazer* $i = i+1$ *e ir para o Passo (c).*

*Passo (g)* *Se* $y_k$ *não é admissível então actualizar* $\lambda_k$ *((8.2.22) se for usada a* $\phi_{LA}$ *ou (8.2.42) se for a* $\phi_E$*),* $\mu_k$ *(*$\mu_{k+1} = \mu_f \mu_k$, $\mu_f \geq 1$*) e ir para o Passo (b). Senão, se* $j > 0$ *e não existir uma evolução significativa de* $y_{eps}$ *para* $y_k$ $\left(\frac{\|y_{eps}-y_k\|}{\|y_k\|} < \delta_2\right)$ *então parar com* $y_k$ *como uma solução aproximada de* (8.1.1). *Senão fazer* $j = j+1$, $y_{eps} = y_k$, *actualizar* $\epsilon_j$, $\tau_j$ *(se for usada* $\phi_{LA}$ *ou* $\phi_E$*) e ir para o Passo (b)*

A sucessão $\{y_k\}$, de aproximações aos problemas (8.2.1,8.2.15), é gerada pela iteração exterior. O processo iterativo indexado por $j$ destina-se a gerar aproximações à solução do problema (8.1.1). A sucessão $\{x_i\}$ de aproximações a um minimizante do problema sem restrições, para um dado $\mu$, e $H_i$ são geradas pela iteração interior, onde $H_i$ é uma aproximação à inversa da Hessiana de $\phi$, na iteração $i$. Uma estratégia de não actualização é implementada sempre que o denominador da fórmula de actualização BFGS é suficientemente pequeno. Também foi implementada uma fórmula modificada (*damped*) que lida com o facto da condição de curvatura não ser satisfeita (veja-se em [99]).

## 8.3 Método de pontos interiores

Na Subsecção 2.2.3 fez-se uma breve introdução ao método de pontos interiores para o caso da programação finita.

O método de pontos interiores tem sido usado na PSI para os casos lineares. Em [29] os autores descrevem um método de pontos interiores proveniente do método de Karmarkar ([2]) e em [30] baseado no escalamento afim.

A técnica aqui descrita é não admissível, uma vez que as aproximações à solução do problema podem não ser admissíveis ao longo do processo iterativo, embora se exija a admissibilidade no limite.

Na Subsecção 8.3.1 descreve-se a estratégia usada no método dos pontos interiores, na Subsecção 8.3.2 apresentam-se os detalhes de implementação e na Subsecção 8.3.5 o algoritmo implementado.

### 8.3.1 Pontos interiores não admissíveis

Considere-se novamente o problema de PSI na sua forma mais simples (8.1.1).

A implementação da técnica de pontos interiores não admissível tem como base o problema aproximado (8.2.15) em que

$$g_{i,\epsilon}(x,t) = \frac{g_i(x,t) + \sqrt{(g_i(x,t))^2 + \epsilon^2}}{2}. \tag{8.3.1}$$

Enquanto que a técnica de suavização usada por Jenning e Teo é apenas $C^1$ (não é duas vezes continuamente diferenciável em $-\epsilon$ e $\epsilon$), a aproximação (8.3.1) é $C^\infty$ em todo o seu domínio.

A descrição que se segue é baseada na estratégia utilizada em [123, 147, 149] para programação finita.

Adicionando variáveis de folga não negativas, $\omega_i$, $i = 1, \ldots, m$, o problema (8.2.15) é reformulado como

$$\begin{aligned} &\min_{x \in R^n, \omega \in R^m} f(x) \\ s.a \quad &\int_T g_{i,\epsilon}(x,t)dt - \tau + \omega_i = 0 \\ &\omega_i \geq 0,\ i = 1, \ldots, m. \end{aligned} \tag{8.3.2}$$

sendo $\omega$ o vector com componentes $\omega_i$.

As restrições de desigualdade em (8.3.2) são eliminadas através da sua colocação em termos barreira na função objectivo, resultando no problema barreira associado a (8.2.15)

$$\begin{aligned} &\min_{x \in R^n, s \in R^m} f(x) - \mu \sum_{i=1}^{m} \log(s_i + \tau) \\ s.a \quad &\int_T g_{i,\epsilon}(x,t)dt + s_i = 0, \quad i = 1, \ldots, m, \end{aligned} \tag{8.3.3}$$

onde $s$ é o vector formado pelas variáveis $s_i = \omega_i - \tau$ e $\mu > 0$ é o parâmetro barreira. A função Lagrangeana associada a (8.3.3) é

$$\mathcal{L}_\mu(x,s,\lambda) = f(x) - \mu \sum_{i=1}^{m} \log(s_i + \tau) + \sum_{i=1}^{m} \lambda_i \left( \int_T g_{i,\epsilon}(x,t)dt + s_i \right) \tag{8.3.4}$$

e as condições KKT de primeira ordem para um minimizante são

$$\nabla_x \mathcal{L}_\mu(x,s,\lambda) = \nabla f(x) + \sum_{i=1}^{m} \lambda_i \int_T \nabla_x g_{i,\epsilon}(x,t)dt = 0 \tag{8.3.5a}$$

$$\nabla_s \mathcal{L}_\mu(x,s,\lambda) = -\mu S^{-1} e + \lambda = 0 \tag{8.3.5b}$$

$$\nabla_\lambda \mathcal{L}_\mu(x,s,\lambda) = -\overline{g}(x) - s = 0 \tag{8.3.5c}$$

onde $S$ é a matriz diagonal formada pelos elementos $s_i + \tau$, $i = 1, \ldots, m$, $e$ é o vector unitário, $\overline{g}(x)$ é o vector com elementos $\int_T g_{i,\epsilon}(x,t)dt$, $s$ é o vector das variáveis $s_i$ e $\lambda$ é o vector das variáveis $\lambda_i$.

Multiplicando a equação (8.3.5b) por $S$, obtém-se o sistema primal-dual

$$\nabla f(x) + \sum_{i=1}^{m} \lambda_i \int_T \nabla_x g_{i,\epsilon}(x,t)dt = 0 \tag{8.3.6a}$$

$$-\mu e + S\Lambda e = 0 \tag{8.3.6b}$$

$$-\overline{g}(x) - s = 0 \tag{8.3.6c}$$

onde $\Lambda$ é a matriz diagonal com elementos $\lambda_i$.

Aplicando o método de Newton à resolução do sistema (8.3.6) obtêm-se as seguintes equações

$$\begin{pmatrix} H(x,\lambda) & 0 & J(x) \\ 0 & \Lambda & S \\ -J^T(x) & -I & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta s \\ \Delta\lambda \end{pmatrix} = \begin{pmatrix} -\nabla f(x) - \sum_{i=1}^{m} \lambda_i \int_T \nabla_x g_{i,\epsilon}(x,t)dt \\ \mu e - S\Lambda e \\ \overline{g}(x) + s \end{pmatrix} \tag{8.3.7}$$

onde

$$H(x,\lambda) = \nabla^2 f(x) + \sum_{i=1}^{m} \lambda_i \int_T \nabla^2_{xx} g_{i,\epsilon}(x,t)dt, \tag{8.3.8}$$

$$J(x) = \left( \int_T \nabla_x g_{1,\epsilon}(x,t)dt, \ldots, \int_T \nabla_x g_{m,\epsilon}(x,t)dt \right) \tag{8.3.9}$$

e $(\Delta x, \Delta s, \Delta\lambda)$ é a direcção Newton.

O sistema (8.3.7) pode ser reescrito numa forma mais compacta como

$$\begin{pmatrix} H & 0 & J \\ 0 & \Lambda & S \\ -J^T & -I & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta s \\ \Delta\lambda \end{pmatrix} = \begin{pmatrix} \sigma \\ \gamma \\ \rho \end{pmatrix} \tag{8.3.10}$$

onde

$$\sigma = -\nabla f - J\lambda \tag{8.3.11a}$$

$$\gamma = \mu e - S\Lambda e \tag{8.3.11b}$$

$$\rho = \bar{g} + s. \tag{8.3.11c}$$

O vector $\sigma$ mede a não admissibilidade dual e o vector $\rho$ mede a não admissibilidade primal.

Apesar de não se resolver o sistema (8.3.10) directamente, é necessário deduzir fórmulas explícitas para $\Delta x$, $\Delta s$ e $\Delta\lambda$ para a análise que será feita na Subsecção 8.3.2.

**Teorema 8.3.1** *(Semelhante ao Teorema 1 em [149].)*
*Se* $N = H + JS^{-1}\Lambda J^T$ *é não singular, o sistema* (8.3.10) *tem uma única solução, dada por*

$$\Delta x = -N^{-1}\nabla f - \mu N^{-1}JS^{-1}e - N^{-1}JS^{-1}\Lambda\rho \tag{8.3.12a}$$

$$\Delta s = J^T N^{-1}\nabla f + \mu J^T N^{-1}JS^{-1}e - (I - J^T N^{-1}JS^{-1}\Lambda)\rho \tag{8.3.12b}$$

$$\Delta\lambda = S^{-1}\Lambda\rho - S^{-1}\Lambda J^T N^{-1}\sigma + S^{-1}\Lambda J^T N^{-1}JS^{-1}\gamma - S^{-1}\Lambda J^T N^{-1}JS^{-1}\Lambda\rho + S^{-1}\gamma. \tag{8.3.12c}$$

**Prova.** Da segunda equação do sistema (8.3.10)

$$\Lambda\Delta s + S\Delta\lambda = \gamma$$

obtém-se

$$\Delta s = \Lambda^{-1}(\gamma - S\Delta\lambda). \tag{8.3.13}$$

A terceira equação do sistema (8.3.10)

$$\begin{aligned} -J^T\Delta x - I\Delta s &= \rho \\ -J^T\Delta x - \Lambda^{-1}(\gamma - S\Delta\lambda) &= \rho \end{aligned}$$

pode escrever-se na forma

$$\Delta\lambda = S^{-1}\Lambda(\rho + J^T\Delta x) + S^{-1}\gamma. \tag{8.3.14}$$

Finalmente, da primeira equação do sistema (8.3.10) e usando (8.3.14) obtém-se

$$\begin{aligned} H\Delta x + J\Delta\lambda &= \sigma \\ H\Delta x + J(S^{-1}\Lambda(\rho + J^T\Delta x) + S^{-1}\gamma) &= \sigma \\ (H + JS^{-1}\Lambda J^T)\Delta x + JS^{-1}\Lambda\rho + JS^{-1}\gamma &= \sigma \end{aligned}$$

e considerando $N = H + JS^{-1}\Lambda J^T$

$$\Delta x = N^{-1}(\sigma - JS^{-1}\gamma - JS^{-1}\Lambda\rho). \tag{8.3.15}$$

Note-se que

$$\sigma - JS^{-1}\gamma = -\nabla f - \mu JS^{-1}e. \tag{8.3.16}$$

Substituindo (8.3.16) em (8.3.15) resulta

$$\Delta x = -N^{-1}\nabla f - \mu N^{-1}JS^{-1}e - N^{-1}JS^{-1}\Lambda\rho. \tag{8.3.17}$$

Agora, usando a equação (8.3.14) em (8.3.13) obtém-se

$$\begin{aligned}\Delta s &= -\rho - J^T\Delta x \\ &= -\rho + J^T N^{-1}\nabla f + \mu J^T N^{-1} J S^{-1} e + J^T N^{-1} J S^{-1}\Lambda\rho\end{aligned}$$

de (8.3.17). Finalmente, usando a equação (8.3.17) em (8.3.14), obtém-se de imediato a fórmula (8.3.12c) para $\Delta\lambda$. ■

Dada uma aproximação inicial à solução do problema $(x_0, s_0, \lambda_0)$, o método procede iterativamente calculando sucessivas aproximações da seguinte forma

$$\begin{cases} x_{k+1} = x_k + \alpha_k \Delta x_k \\ s_{k+1} = x_k + \alpha_k \Delta s_k \\ \lambda_{k+1} = \lambda_k + \alpha_k \Delta\lambda_k \end{cases} \tag{8.3.18}$$

onde $k$ é o contador de iteração e $\alpha_k$ é o comprimento do passo determinado de forma a garantir que os vectores $s + \tau e$ e $\lambda$ sejam não negativos, bem como a convergência para um minimizante e admissibilidade.

### 8.3.2 Detalhes de implementação

#### Função Mérito

Em optimização não linear é necessário introduzir uma função mérito ou um filtro (veja-se em [33] ou [7] no contexto de pontos interiores) para garantir progresso em direcção ao minimizante local e admissibilidade. Este progresso é conseguido através da selecção de um comprimento do passo ao longo da direcção de procura, por forma a que seja obtida uma redução significativa na função mérito.

Em [26] os autores propõem uma função mérito baseada na norma $l_2$ do vector resíduo $(\sigma,\ \gamma - \mu e,\ \rho)$. Como esta função mérito pode levar o algoritmo a convergir para um maximizante local ou ponto sela [123], usaram-se duas funções de penalidade como funções mérito. A primeira é a função mérito $l_2$ que, para o problema (8.3.3), tem a seguinte forma

$$\phi(x, s; \mu, \beta) = f(x) - \mu \sum_{i=1}^{m} \log(s_i + \tau) + \frac{\beta}{2}\rho^T\rho, \tag{8.3.19}$$

onde $\rho$ é dado por (8.3.11c) e $\beta > 0$ é o parâmetro de penalidade, e a outra é a função mérito baseada na Lagrangeana aumentada

$$\mathcal{L}_A(x, s, \lambda; \mu, \beta) = f(x) - \mu \sum_{i=1}^{m} \log(s_i + \tau) + \lambda^T\rho + \frac{\beta}{2}\rho^T\rho. \tag{8.3.20}$$

Quando o problema é estritamente convexo, prova-se que, para um $\beta$ suficientemente grande, a direcção definida pelo sistema (8.3.7) é de descida para as funções mérito $l_2$ e Lagrangeana aumentada.

**Teorema 8.3.2** *Se $N$ é uma matriz definida positiva, então existe um $\beta^{\phi}_{min} > 0$, tal que a direcção de procura $(\Delta x, \Delta s)$ satisfaz*

$$\begin{pmatrix} \nabla_x \phi \\ \nabla_s \phi \end{pmatrix}^T \begin{pmatrix} \Delta x \\ \Delta s \end{pmatrix} \leq 0$$

*para todo o $\beta > \beta^{\phi}_{min}$. A igualdade é válida se e só se $(x, s)$ satisfaz (8.3.6) para algum $\lambda$.*

**Prova.** Como

$$\nabla_x \phi(x, s; \mu, \beta) = \nabla f + \beta J \rho \tag{8.3.21}$$

e

$$\nabla_s \phi(x, s; \mu, \beta) = -\mu S^{-1} e + \beta \rho, \tag{8.3.22}$$

e notando que

$$\Delta s = -\rho - J^T \Delta x, \tag{8.3.23}$$

$$\lambda = \mu S^{-1} e \tag{8.3.24}$$

tem-se que

$$\begin{pmatrix} J\rho \\ \rho \end{pmatrix}^T \begin{pmatrix} \Delta x \\ -\rho - J^T \Delta x \end{pmatrix} = \rho^T J^T \Delta x - \rho^T \rho - \rho^T J^T \Delta x = -\rho^T \rho.$$

Usando (8.3.12a), verifica-se que

$$\begin{aligned}
\begin{pmatrix} \nabla_x \phi \\ \nabla_s \phi \end{pmatrix}^T \begin{pmatrix} \Delta x \\ \Delta s \end{pmatrix} &= \begin{pmatrix} \nabla f + \beta J \rho \\ -\lambda + \beta \rho \end{pmatrix}^T \begin{pmatrix} \Delta x \\ -\rho - J^T \Delta x \end{pmatrix} \\
&= \begin{pmatrix} \nabla f \\ -\lambda \end{pmatrix}^T \begin{pmatrix} \Delta x \\ -\rho - J^T \Delta x \end{pmatrix} - \beta \rho^T \rho \\
&= \nabla f^T \Delta x - \lambda^T (-\rho - J^T \Delta x) - \beta \rho^T \rho \\
&= \nabla f^T \Delta x + \lambda^T \rho + \lambda^T J^T \Delta x - \beta \rho^T \rho \\
&= (\nabla f^T + (J\lambda)^T) \Delta x + \lambda^T \rho - \beta \rho^T \rho \\
&= (\nabla f + J\lambda)^T (-N^{-1} \nabla f - \mu N^{-1} J S^{-1} e - N^{-1} J S^{-1} \Lambda \rho) \\
&\quad + \lambda^T \rho - \beta \rho^T \rho \\
&= -(\nabla f + J\lambda)^T N^{-1} (\nabla f + J\lambda) \\
&\quad - (\nabla f + J\lambda)^T N^{-1} J S^{-1} \Lambda \rho \\
&\quad + \lambda^T \rho - \beta \rho^T \rho.
\end{aligned} \tag{8.3.25}$$

Considerem-se os seguintes casos. Se $(x, s)$ é admissível ($\rho = 0$) então ou

$$\begin{pmatrix} \nabla_x \phi \\ \nabla_s \phi \end{pmatrix}^T \begin{pmatrix} \Delta x \\ \Delta s \end{pmatrix} = -\sigma^T N^{-1} \sigma < 0$$

por $N$ ser definida positiva, ou $\sigma N^{-1} \sigma = 0$. Neste caso como $N$ é definida positiva vem que $\sigma = 0$. Esta equação e a admissibilidade assumida fazem parte das condições de optimalidade do problema barreira.

Suponha-se agora que $(x, s)$ não é admissível ($\rho \neq 0$), então escolhendo

$$\beta_{min}^{\phi} = \frac{-\sigma^T N^{-1} \sigma - \sigma^T N^{-1} J S^{-1} \Lambda \rho + \lambda^T \rho}{\rho^T \rho},$$

garante-se que $(\Delta x, \Delta s)$ é uma direcção de descida para a função mérito $l_2$, para todo o $\beta > \beta_{min}^{\phi}$. ■

**Teorema 8.3.3** *Se $N$ é uma matriz definida positiva, então existe $\beta_{min}^{\mathcal{L}_A} > 0$, tal que a direcção de procura $(\Delta x, \Delta s, \Delta \lambda)$ satisfaz*

$$\begin{pmatrix} \nabla_x \mathcal{L}_A \\ \nabla_s \mathcal{L}_A \\ \nabla_\lambda \mathcal{L}_A \end{pmatrix}^T \begin{pmatrix} \Delta x \\ \Delta s \\ \Delta \lambda \end{pmatrix} \leq 0$$

*para todo o $\beta > \beta_{min}^{\mathcal{L}_A}$. A igualdade é verificada se e só se $(x, s)$ satisfaz* (8.3.6) *para algum $\lambda$.*

**Prova.** Note-se que se tem

$$\mathcal{L}_A(x, s, \lambda; \mu, \beta) = \phi(x, s; \mu, \beta) + \lambda^T \rho$$

e

$$\nabla_x \mathcal{L}_A = \nabla f + J\lambda + \beta J \rho = \nabla_x \phi + J\lambda$$

$$\nabla_s \mathcal{L}_A = -\mu S^{-1} e + \lambda + \beta \rho = \nabla_s \phi + \lambda$$

$$\nabla_\lambda \mathcal{L}_A = \rho,$$

e consequentemente tem-se que,

$$\begin{pmatrix} \nabla_x \mathcal{L}_A \\ \nabla_s \mathcal{L}_A \\ \nabla_\lambda \mathcal{L}_A \end{pmatrix}^T \begin{pmatrix} \Delta x \\ \Delta s \\ \Delta \lambda \end{pmatrix} = \begin{pmatrix} \nabla_x \phi + J\lambda \\ \nabla_s \phi + \lambda \\ \rho \end{pmatrix}^T \begin{pmatrix} \Delta x \\ \Delta s \\ \Delta \lambda \end{pmatrix}.$$

Usando (8.3.23) e (8.3.24), tem-se que

$$\begin{aligned}\begin{pmatrix} J\lambda \\ \lambda \\ \rho \end{pmatrix}^T \begin{pmatrix} \Delta x \\ \Delta s \\ \Delta\lambda \end{pmatrix} =& \lambda^T J^T \Delta x + \lambda^T \Delta s + \rho^T \Delta\lambda \\ =& \lambda^T J^T \Delta x - \lambda^T \left(\rho + J^T \Delta x\right) + \rho^T \Delta\lambda \\ =& -\lambda^T \rho + \rho^T \Delta\lambda,\end{aligned} \tag{8.3.26}$$

onde $\Delta\lambda$ é dado por (8.3.12c).

Combinando (8.3.25) e (8.3.26) chega-se a

$$\begin{pmatrix} \nabla_x \mathcal{L}_A \\ \nabla_s \mathcal{L}_A \\ \nabla_\lambda \mathcal{L}_A \end{pmatrix}^T \begin{pmatrix} \Delta x \\ \Delta s \\ \Delta\lambda \end{pmatrix} = -\sigma^T N^{-1}\sigma - \sigma^T N^{-1} J S^{-1}\Lambda\rho - \beta\rho^T\rho - \rho^T\Delta\lambda.$$

Usando um raciocínio idêntico ao da prova anterior e escolhendo

$$\beta_{min}^{\mathcal{L}_\mathcal{A}} = \frac{-\sigma^T N^{-1}\sigma - \sigma^T N^{-1} J S^{-1}\Lambda\rho + \rho^T\Delta\lambda}{\rho^T\rho},$$

garante-se que $(\Delta x, \Delta s, \Delta\lambda)$ é uma direcção de descida para a função mérito baseada na Lagrangeana aumentada, para todo o $\beta > \beta_{min}^{\mathcal{L}_\mathcal{A}}$. ■

Apesar das fórmulas para o $\beta_{min}$, o algoritmo procede da seguinte forma para calcular $\beta$ tal que a direcção de procura seja de descida para as funções mérito.

**Algoritmo 8.3.1** *Cálculo do $\beta$.*

*Passo(a)* *Se $k = 0$ então faça-se $\beta_k = 0$.*

*Passo(b)* *Se*

$$\begin{pmatrix} \nabla_x \phi \\ \nabla_s \phi \end{pmatrix}^T \begin{pmatrix} \Delta x \\ \Delta s \end{pmatrix} < 0 \quad ou \quad \begin{pmatrix} \nabla_x \mathcal{L}_A \\ \nabla_s \mathcal{L}_A \\ \nabla_\lambda \mathcal{L}_A \end{pmatrix}^T \begin{pmatrix} \Delta x \\ \Delta s \\ \Delta\lambda \end{pmatrix} < 0$$

*então pára-se, caso contrário vai-se para o Passo*(c)

*Passo(c)* *Se $\beta_k = 0$ então faça-se $\beta_k = 0.1$ e vai-se para o Passo*(b)*, senão $\beta_k = 10\beta_k$ e vai-se para o Passo*(b).

### Cálculo do comprimento do passo

Numa estratégia de redução sucessiva do comprimento do passo (*backtraking*) com o objectivo de determinar um $\alpha_k$ que gere uma redução significativa na função mérito, a inicialização com $\alpha_k = 1$ pode violar a propriedade de que as variáveis $\omega_k = s_k + \tau e$ e $\lambda_k$ têm de ser não negativas. No contexto da discussão apresentada na Subsecção 8.3.1, isto significa que $s_{i,k} + \alpha_k \Delta s_{i,k}$ tem de ser superior a $-\tau$ e $\lambda_{i,k} + \alpha_k \Delta \lambda_{i_k}$ tem de ser positivo, para todo o $i = 1, \ldots, m$. As desigualdades

$$\lambda + \alpha \Delta \lambda \geq 0 \tag{8.3.27}$$

e

$$s + \alpha \Delta s \geq -\tau e \tag{8.3.28}$$

implicam que o valor máximo possível para $\alpha$ é

$$\min \left\{ \frac{-\lambda_i}{\Delta \lambda_i}, \frac{-\tau - s_i}{\Delta s_i} \right\}, \tag{8.3.29}$$

para todo o $i$ tal que $\Delta \lambda_i < 0$ e $\Delta s_i < 0$.

Como procedimento de salvaguarda escolhe-se

$$\alpha_{max} = \min \left\{ 1, 0.95 \min \left\{ \frac{-\lambda_i}{\Delta \lambda_i}, \frac{-\tau - s_i}{\Delta s_i} \right\} \right\}, \tag{8.3.30}$$

para evitar a proximidade das variáveis $s$ a $-\tau e$ e de $\lambda$ a zero.

A estratégia de redução sucessiva é então implementada no intervalo $(0, \alpha_{max}]$ para determinar o $\alpha_k$ que produza uma redução na função mérito.

### Aproximações iniciais

Podem ser definidas algumas heurísticas no sentido de fornecer aproximações iniciais para as variáveis de folga e duais. O método aceita qualquer aproximação inicial às variáveis $x$. Se não for fornecida uma aproximação inicial para as variáveis primais, pode usar-se o procedimento descrito na Secção 6.2 e em [151].

Uma vez que $x_0$ pode ser um ponto não admissível ou até um ponto muito próximo da fronteira da região admissível e as variáveis de folga ($s_0$) serem menores ou muito próximas de $-\tau e$, introduz-se um parâmetro $\theta > 0$ tal que $s_0$ seja pelo menos igual a $\theta$. Assim, a heurística implementada consiste em definir

$$s_{i,0} = \max \left\{ \int_T g_{i,\epsilon}(x_0, t) dt, \theta \right\}, \quad i = 1, \ldots, m \tag{8.3.31}$$

e, para as variáveis duais, usar estimativas dos mínimos quadrados, que se obtêm através do problema

$$min_\lambda \| \nabla f + J\lambda \|_2^2$$

cuja solução é

$$\lambda = -[J^T J]^{-1} J^T \nabla f. \tag{8.3.32}$$

Note-se que sempre que se actualiza o parâmetro $\tau$, no contexto da resolução de (8.2.15) como aproximação ao problema (8.1.1) através de um factor de redução $\xi_\tau$, há necessidade de recalcular as variáveis de folga para manter $s_i + \tau > 0$. A fórmula (8.3.31) é então usada com $\theta$ substituído por $\xi_\tau^k \theta$, onde $k$ é o contador de iterações.

### Cálculo do parâmetro barreira $\mu$

Até agora não foi feita qualquer referência à escolha do parâmetro barreira.

Com base em (8.3.6b), o valor para $\mu$ pode ser recuperado usando $(s_i + \tau)\lambda_i$, para qualquer $i$. Em alternativa, pode escolher-se $\mu$ como

$$\mu = \delta_\mu \frac{\lambda^T (s + \tau e)}{m}, \tag{8.3.33}$$

onde $0 \leq \delta_\mu < 1$ é usado para obter um ponto mais próximo da optimalidade que a aproximação actual.

### Ordenação e ordenação superior

Existem três caminhos diferentes para resolver o sistema (8.3.10). Uma resulta da factorização directa do sistema (8.3.10) e as outras duas estão relacionadas com as equações normais obtidas a partir de (8.3.10). Dependendo da sequência usada nos cálculos assim se obtêm a ordenação primal e a ordenação dual das equações normais.

De seguida descrevem-se as duas ordenações principais das equações normais.

### Ordenação primal

A ordenação primal é obtida a partir do sistema (8.3.10) através da resolução da primeira e segunda equações respectivamente em ordem a $\Delta x$ e $\Delta s$ e substituindo na última equação para obter $\Delta\lambda$:

$$\Delta x = H^{-1}(\sigma - J\Delta\lambda), \tag{8.3.34a}$$

$$\Delta s = \Lambda^{-1}(\gamma - S\Delta\lambda), \tag{8.3.34b}$$

e

$$(J^T H^{-1} J + \Lambda^{-1} S)\Delta\lambda = \Lambda^{-1}\gamma + \rho + J^T H^{-1}\sigma. \tag{8.3.34c}$$

Quando o problema é não convexo, $H^{-1}$ pode nem ser definida. Se uma técnica quasi-Newton apropriada for usada para aproximar $H^{-1}$, este problema numérico pode ser ultrapassado (veja-se na Subsecção 8.3.3).

### Ordenação dual

A ordenação dual é obtida através da resolução das duas últimas equações do sistema (8.3.10) em ordem a $\Delta s$ e $\Delta\lambda$ e substituindo na primeira equação para obter $\Delta x$:

$$\Delta s = -\rho - J^T \Delta x, \tag{8.3.35a}$$

$$\Delta\lambda = S^{-1}(\gamma - \Lambda\Delta s), \tag{8.3.35b}$$

e

$$(H + JS^{-1}\Lambda J^T)\Delta x = -\nabla f - JS^{-1}(\Lambda\rho + \mu e). \tag{8.3.35c}$$

Neste caso, o termo $JS^{-1}\Lambda J^T$ pode tornar a matriz $H + JS^{-1}\Lambda J^T$ definida positiva, mesmo que $H$ seja indefinida. Em geral, para problemas não lineares a ordenação dual é numericamente mais estável que a ordenação primal.

### Ordenação superior

É possível introduzir um esquema preditor corrector com o objectivo de acelerar a convergência do processo. Mehrotra [86] propôs um esquema preditor corrector para pontos interiores aplicados a problemas lineares finitos. O algoritmo consiste no cálculo de duas direcções, a preditora e a correctora, na mesma iteração, ambas baseadas numa única factorização da matriz do sistema resultante da aplicação do método de Newton às condições KKT. Em [17] os autores descrevem uma extensão ao método preditor corrector de Mehrotra para múltiplas correcções no contexto da programação linear e quadrática convexa finita e a relação desta com o método composto de Newton. Em [43, 61] os autores analisam o uso de vários passos correctores no contexto da programação linear finita. Em [123] os autores descrevem a ordenação primal, a dual e a técnica preditora correctora para o caso não linear e finito. Como esta técnica pode não produzir uma direcção de descida para a função mérito $l_2$, Shanno e Vanderbei propõem uma mudança para a direcção por defeito se o parâmetro de penalidade aumentar demasiado ou se o comprimento do passo for demasiado pequeno.

A técnica preditora correctora aqui implementada resolve primeiro o sistema Newton não perturbado

$$\begin{pmatrix} H(x,\lambda) & 0 & J(x) \\ 0 & \Lambda & S \\ -J^T(x) & -I & 0 \end{pmatrix} \begin{pmatrix} \Delta x_p \\ \Delta s_p \\ \Delta\lambda_p \end{pmatrix} = \begin{pmatrix} \sigma \\ -S\Lambda e \\ \rho \end{pmatrix} \tag{8.3.36}$$

para obter a direcção preditora $\Delta x_p$, $\Delta s_p$ e $\Delta\lambda_p$, com base na ordenação dual.

A direcção correctora de ordem $i$ ($\Delta x_c^i$, $\Delta s_c^i$, $\Delta\lambda_c^i$), $i = 1, \ldots, m_c$ é então obtida do sistema Newton perturbado

$$\begin{pmatrix} H(x,\lambda) & 0 & J(x) \\ 0 & \Lambda & S \\ -J^T(x) & -I & 0 \end{pmatrix} \begin{pmatrix} \Delta x_c^i \\ \Delta s_c^i \\ \Delta\lambda_c^i \end{pmatrix} = \begin{pmatrix} \sigma \\ -S\Lambda e + \mu e - \Delta S^{i-1}\Delta\Lambda^{i-1}e \\ \rho \end{pmatrix} \tag{8.3.37}$$

onde $\Delta S^{i-1}$, $\Delta\Lambda^{i-1}$ são matrizes diagonais respectivamente com os elementos $\Delta s_c^{i-1}$ e $\Delta\lambda_c^{i-1}$ e $\Delta S^0$, $\Delta\Lambda^0$ são as correspondentes matrizes diagonais calculadas com base respectivamente nos elementos de $\Delta s_p$ e $\Delta\lambda_p$.

O algoritmo é descrito da seguinte forma.

**Algoritmo 8.3.2** *Esquema Preditor Corrector.*

*Passo(a)* *Resolve-se o sistema* (8.3.36) *para obter a direcção preditora* $\Delta x_p$, $\Delta s_p$ *e* $\Delta\lambda_p$ *usando as seguintes fórmulas*

$$\left(H + JS^{-1}\Lambda J^T\right)\Delta x_p = -\nabla f - JS^{-1}\Lambda\rho \tag{8.3.38a}$$

$$\Delta s_p = -\left(\rho + J^T\Delta x_p\right) \tag{8.3.38b}$$

$$\Delta\lambda_p = -S^{-1}\left(\Lambda\Delta s_p + S\Lambda e\right) \tag{8.3.38c}$$

*Passo(b)* *Para* $i = 1, \ldots, m_c$ *faz-se*

*Passo(b.1)* *Calcula-se* $\mu = \frac{\left(\lambda+\alpha_{max}\Delta\lambda_c^{i-1}\right)^T\left(s+\tau e+\alpha_{max}\Delta s_c^{i-1}\right)}{m}$, *com* $\Delta x_c^0 = \Delta x_p$, $\Delta s_c^0 = \Delta s_p$, $\Delta\lambda_c^0 = \Delta\lambda_p$ *e* $\alpha_{max}$ *é o comprimento máximo do passo, dado por* (8.3.30).

*Passo(b.2)* *Resolve-se o sistema* (8.3.37) *para obter a direcção correctora* $\Delta x_c^i$, $\Delta s_c^i$ *e* $\Delta\lambda_c^i$ *usando as seguintes fórmulas*

$$\left(H + JS^{-1}\Lambda J^T\right)\Delta x_c^i = -\nabla f - JS^{-1}\left(\Lambda\rho + \mu e - \Delta S_c^{i-1}\Delta\Lambda_c^{i-1}e\right) \tag{8.3.39a}$$

$$\Delta s_c^i = -\left(\rho + J^T\Delta x_c\right) \tag{8.3.39b}$$

$$\Delta\lambda_c^i = -S^{-1}\left(\Lambda\Delta s_p + S\Lambda e - \mu e + \Delta S_c^{i-1}\Delta\Lambda_c^{i-1}e\right) \tag{8.3.39c}$$

*Passo(c)* *continue-se com* $\Delta x = \Delta x_c^{m_c}$, $\Delta s = \Delta s_c^{m_c}$ *e* $\Delta\lambda = \Delta\lambda_c^{m_c}$.

**Proposição 8.3.1** *Se* $x$ *é uma solução admissível para o problema* (8.2.15) *então a direcção correctora obtida através do sistema* (8.3.37) *pode não ser de descida para a função mérito* $l_2$ (8.3.19).

**Prova.** Do mesmo modo que na prova do Teorema 8.3.2, tem-se que

$$\begin{pmatrix}\nabla_x\phi\\ \nabla_s\phi\end{pmatrix}^T\begin{pmatrix}\Delta x_c^i\\ \Delta s_c^i\end{pmatrix} = -\sigma^T N^{-1}\sigma + \sigma^T N^{-1}\left(JS^{-1}\left(\gamma + \Lambda\rho - \Delta S_c^{i-1}\Delta\Lambda_c^{i-1}e\right)\right) + \lambda^T\rho - \beta\rho^T\rho.$$

Quando $x$ é admissível tem-se que

$$\begin{pmatrix}\nabla_x\phi\\ \nabla_s\phi\end{pmatrix}^T\begin{pmatrix}\Delta x_c^i\\ \Delta s_c^i\end{pmatrix} = -\sigma^T N^{-1}\sigma + \sigma^T N^{-1}JS^{-1}\left(\gamma - \Delta S_c^{i-1}\Delta\Lambda_c^{i-1}e\right)$$

que pode não ser negativo. ■

### Cálculo dos integrais

O cálculo dos integrais que surgem no vector $\bar{g}$ e na matriz $J$ é feito da forma já descrita na Subsecção 8.1.5.

### Técnica de cão de guarda

Como não é possível garantir que a direcção preditora correctora seja de descida para a função mérito $l_2$ pode implementar-se uma técnica de cão de guarda (*watchdog*) semelhantes às usadas em [38, 99]. Esta técnica define uma estratégia não monótona e consiste em permitir um aumento da função mérito em certas iterações antes de forçar uma redução.

A técnica de cão de guarda é descrita no seguinte algoritmo.

**Algoritmo 8.3.3** *Técnica de cão de guarda.*

*Passo(a)* *Seja* `watchmax` *o número máximo de aumentos consecutivos na função mérito.*

*Passo(b)* *Se* $k = 0$ *então seja* `watch`$=0$.

*Passo(c)* *Calcula-se* $\alpha_{max,k}$ *usando a fórmula* (8.3.30).

*Passo(d)* *Se* `watch`$<$`watchmax`

*então* *Seja* $x_{k+1} = x_k + \alpha_{max,k}\Delta x_k$, $s_{k+1} = s_k + \alpha_{max,k}\Delta s_k$ *e* $\lambda_{k+1} = \lambda_k + \alpha_{max,k}\Delta\lambda_k$.
*Se* $\phi(x_{k+1}, s_{k+1}; \mu_k, \beta_k) - \phi(x_k, s_k; \mu_k, \beta_k) < 0$

*então* ***Guardar*** $x_{k+1}$, $s_{k+1}$, $\lambda_{k+1}$, $\Delta x_k$, $\Delta s_k$, $\Delta\lambda_k$, $\alpha_{max,k}$, $\beta_k$ *e* $\phi(x_{k+1}, s_{k+1}; \mu_k, \beta_k)$.
*Fazer* `watch`$=0$ *e ir para o Passo*(g).

*senão* *Se* `watch=0`

*então* ***Guardar*** $x_k$, $s_k$, $\lambda_k$, $\Delta x_k$, $\Delta s_k$, $\Delta\lambda_k$, $\alpha_{max,k}$, $\beta_k$ *e* $\phi(x_k, s_k; \mu_k, \beta_k)$.
*Fazer* `watch`$=$`watch`$+1$. *Ir para o Passo*(g)

*senão* ***Recuperar*** $x_k$, $s_k$, $\lambda_k$, $\Delta x_k$, $\Delta s_k$, $\Delta\lambda_k$, $\alpha_{max,k}$, $\beta_k$ *e* $\phi(x_k, s_k; \mu_k, \beta_k)$. *Fazer* $\alpha_k = \frac{\alpha_{max,k}}{2}$ *ir para o Passo*(e)

*Passo(e)* *Calcular* $\beta_{k+1}$ *usando o Algoritmo 8.3.1.*

*Passo(f)* *Calcular* $\alpha_k$ *tal que* $\phi(x_{k+1}, s_{k+1}; \mu_k, \beta_{k+1}) - \phi(x_k, s_k; \mu_k, \beta_{k+1}) < 0$, *com* $x_{k+1} = x_k + \alpha_k\Delta x_k$, $s_{k+1} = s_k + \alpha_k\Delta s_k$ *e* $\lambda_{k+1} = \lambda_k + \alpha_k\Delta\lambda_k$. ***Guardar*** $x_{k+1}$, $s_{k+1}$, $\lambda_{k+1}$, $\Delta x_k$, $\Delta s_k$, $\Delta\lambda_k$, $\alpha_{max,k}$, $\beta_{k+1}$ *e* $\phi(x_{k+1}, s_{k+1}; \mu_k, \beta_{k+1})$.

*Passo(g)* *Continuar.*

**Falhas e estabilidade**

Os elementos da matriz $S^{-1}$ podem originar alguns problemas numéricos na resolução dos sistemas lineares (8.3.35) (veja-se em [160]) quando conduzidos para zero demasiado rápido. Uma maneira de ultrapassar este inconveniente consiste em definir um limite inferior para esses valores, através da heurística [8]

$$s_{i,k} = \max\left\{s_{i,k}, 10^{-8} - \tau\right\}, i = 1, \ldots, m. \tag{8.3.40}$$

Em alternativa, Benson *et al.* em [8] sugerem a média harmónica dos valores de $s_i$ como limite inferior das variáveis.

Elementos pivôs próximos de zero podem provocar falhas no algoritmo aquando da factorização Cholesky do sistema (8.3.35). É aconselhável implementar-se uma versão modificada da factorização Cholesky para resolver estes problemas. O algoritmo que foi implementado neste trabalho é descrito por Wright em [168]. O algoritmo baseia-se na substituição dos pivôs próximos de zero por quantidades suficientemente grandes antes de proceder à factorização usual de Cholesky. Esta estratégia também é usada no software PCx [24].

No trabalho de Sporre e Frosgren [128] é analisada uma questão importante que tem a ver com o facto dos multiplicadores de Lagrange poderem divergir num método primal-dual de pontos interiores. A implicação prática desta situação ainda não é conhecida mas está a ser investigada.

### 8.3.3 A fórmula BFGS

Armand *et al.* [4] introduzem e analisam um método quasi-Newton de pontos interiores admissível que usa a fórmula de actualização BFGS na resolução de problemas estritamente convexos. Num trabalho posterior, os mesmos autores em [5] relaxam a hipótese de convexidade da função objectivo ou das restrições, mas assumem a convexidade da função Lagrangeana.

Para evitar o pesado cálculo da matriz Hessiana $H(x, \lambda)$, em (8.3.8) e conservar a matriz $N$ definida positiva, foi implementada uma estratégia quasi-Newton baseada na fórmula de actualização BFGS para aproximar $H(x_{k+1}, \lambda_{k+1})$,

$$B_{k+1} = B_k - \frac{B_k v_k v_k^T B_k}{v_k^T B_k v_k} + \frac{y_k y_k^T}{v_k^T y_k}$$

onde

$$v_k = x_{k+1} - x_k, \qquad y_k = \nabla_x \mathcal{L}_\mu(x_{k+1}, \lambda_{k+1}) - \nabla_x \mathcal{L}_\mu(x_k, \lambda_{k+1}).$$

A correspondente fórmula de actualização para a inversa da Hessiana $H^{-1}$ é

$$C_{k+1} = \left(I - \frac{v_k y_k^T}{y_k^T v_k}\right) C_k \left(I - \frac{y_k v_k^T}{y_k^T v_k}\right) + \frac{v_k v_k^T}{y_k^T v_k}.$$

Sempre que a condição de curvatura $(v_k)^T y_k \geq 0$ não é verificada, deve implementar-se uma estratégia modificada (*damped*) idêntica à proposta em [99]. Se a fórmula modificada continuar a não satisfazer a condição de curvatura, para uma dada precisão ($\delta_c$), então opta-se pela não actualização das matrizes ($B_{k+1} = B_k$ ou $C_{k+1} = C_k$).

### 8.3.4 Critério de paragem

Neste método primal-dual mede-se a proximidade da aproximação em relação à solução através dos vectores que medem a não admissibilidade primal e dual. Assim, o algoritmo pára se as aproximações calculadas para as variáveis primais e duais verificarem

$$\|\rho\|_2 \leq \delta_f \tag{8.3.41}$$

e

$$\|\sigma\|_2 \leq \delta_f, \tag{8.3.42}$$

para $\delta_f$ positivo e próximo de zero.

### 8.3.5 O algoritmo implementado

Apresenta-se nesta subsecção o algoritmo completo primal-dual de pontos interiores não admissível com base na ordenação dual das equações normais, sem a opção da estratégia preditora correctora.

**Algoritmo 8.3.4** *Algoritmo primal-dual de pontos interiores.*

*Passo(a)* *Dados* $x_0$, $\epsilon$, $\tau$, $\theta$, $\delta_\mu$ *e* $\delta_f$.

*Passo(b)* *Calcule-se* $s_{i,0}$, $i = 1, \ldots, m$ *usando* (8.3.31). *Calcule-se* $\lambda^0_{i,0}$, $i = 1, \ldots, m$ *usando* (8.3.32). *Seja* $k = 0$.

*Passo(c)* *Seja* $y_{eps} = x_k$ *o último valor de* $y$ *calculado para um dado* $\epsilon$.

*Passo(d)* *Calcule-se ou actualize-se* $\mu_k$ *usando* (8.3.33).

*Passo(e)* *Testa-se o critério de paragem descrito na Subsecção 8.3.4. Se o critério de paragem é verificado então se há uma diferença significativa entre* $y_{eps}$ *e* $x_k$ *decrementa-se* $\epsilon$, $\tau$, *actualiza-se as variáveis de folga e vai-se para o Passo*(c)*; caso contrário pára-se.*

*Passo(f)* *Actualiza-se* $B_k$ *usando a fórmula BFGS. Se* $k = 0$ *então* $B_k$ *=Matriz identidade.*

*Passo(g)* *Resolve-se o sistema KKT para obter a direcção de procura* $(\Delta x_k, \Delta s_k, \Delta \lambda_k)$, *usando* (8.3.35).

*Passo(h)* *Calcula-se* $\beta$ *como descrito no Algoritmo 8.3.1.*

*Passo(i)* *Calcula-se* $\alpha_{max}$ *através de* (8.3.30).

*Passo(j)* *Calcula-se* $\alpha_k$ *usando uma estratégia de redução sucessiva tal que produza uma redução na função mérito* (8.3.19) *ou* (8.3.20).

*Passo(k)* *Calcula-se* $x_{k+1}$, $s_{k+1}$ *e* $\lambda_{k+1}$ *como descrito em* (8.3.18).

*Passo(l)* *Vai-se para o Passo*(d).

Os algoritmos que implementam a ordenação primal e a opção das direcções preditora e correctora são semelhantes ao descrito, com as correspondentes alterações (determinação da direcção de procura, actualização da inversa da Hessiana e determinação das direcções preditora e correctora).

# Capítulo 9

# Software de resolução de problemas de programação semi-infinita não linear

---

O pacote NSIPS (*Nonlinear Semi-Infinite Programming Solver*) é um software que foi desenvolvido no âmbito deste trabalho e está vocacionado para resolver problemas de PSI. O NSIPS v1.0 para o sistema operativo Linux, continha a implementação de três tipos de métodos: uma versão modificada do método de discretização, o método baseado em funções de penalidade e um método de programação quadrática sequencial. A versão do método de discretização é apresentada em três variantes e o método de penalidade em duas, resultando num total de seis algoritmos implementados. O NSIPS v2.0 incluía, além dos algoritmos da versão 1.0, o método de pontos interiores. A versão mais actual do software, NSIPS v2.1, está preparada para os sistemas operativos Linux e Windows.

---

A primeira secção apresenta algumas considerações sobre o pacote NSIPS. A passagem de opções ao NSIPS é descrita na Secção 9.2 e a Secção 9.3 apresenta as opções que levam à selecção de um dos métodos para a resolução de um problema de PSI. A Secção 9.4 mostra o modo de apresentação de resultados do NSIPS e a Secção 9.5 descreve as opções específicas de cada método. Os resultados específicos de cada método são apresentados na Secção 9.6.

Os detalhes para a utilização e instalação do NSIPS são relegados para o Apêndice C.

## 9.1 Considerações sobre o software NSIPS

O NSIPS é um código (*solver*) escrito na linguagem C que contém vários métodos para a resolução de problemas de PSI. Este *solver* deve ser usado em ligação ao pacote SIPAM-PL que contém as rotinas de interface necessárias ao cálculo das funções envolvidas no problema. A versão inicial (1.0) do NSIPS incluía os métodos de discretização, de programação quadrática sequencial e de penalidade e a versão 2.0 introduz o método de pontos

interiores. Com base na experiência adquirida ao longo deste projecto e nas solicitações entretanto surgidas, desenvolveram-se as versões do SIPAMPL e NSIPS para o sistema operativo Windows, dando origem à versão 2.0 do SIPAMPL e à versão 2.1 do NSIPS.

A versão modificada do método de discretização e o método de programação quadrática sequencial (PQS) usam o software NPSOL [40] para resolver os correspondentes subproblemas finitos. Como o NPSOL é um software comercial, não pode ser distribuído juntamente com o NSIPS. A versão em formato binário do NSIPS não inclui as subrotinas do NPSOL, donde os métodos de discretização e PQS não podem ser usados nesta situação. O software NPSOL pode ser obtido através do endereço:

```
Stanford Business Software
2680 Bayshore Parkway, Suite 304
Mountain View, CA 94043
Phone:  +1-415-962-8719
Fax:    +1-415-962-1869
```

O potencial utilizador do NSIPS deve construir a biblioteca `libopt.a` usando as instruções que acompanham o software NPSOL, uma vez que ela é usada com o *solver* NSIPS para obter a versão com todos os métodos operacionais.

Quando não se disponibiliza uma aproximação inicial à solução de um problema, o NSIPS usa o procedimento descrito na Secção 6.2 para determinar uma aproximação. Se o utilizador não puder aceder ao NPSOL e o problema não tiver uma aproximação inicial, recomenda-se a edição do ficheiro do problema a resolver (com extensão `.mod`) propondo uma aproximação inicial.

## 9.2 Passagem de opções ao *solver* NSIPS

As opções são passadas ao software NSIPS da mesma forma que são passadas as opções para qualquer *solver* ligado ao AMPL. As opções podem ser passadas

- como argumentos na linha de comandos,
- nas variáveis de ambiente da linha de comandos,
- no ficheiro modelo do problema (com extensão `.mod`) através do comando `option` do AMPL,
- num ficheiro de opções.

Os exemplos seguintes dizem respeito ao *solver* NSIPS.
Na linha de comandos, se o problema estiver no formato intermédio (formato *stub*),

```
% nsips stub [opção=valor]
```

onde *opção* é a opção que se pretende modificar, atribuindo-lhe o valor *valor*. Os parên-

tesis rectos [x] significam que a opção x é opcional e podem ser repetidos as vezes desejadas para modificar outras opções.

Podem atribuir-se valores à variável de ambiente `nsips_options` da linha de comandos

```
% nsips_options='[opção=valor]'
% export nsips_options
```

Uma das dificuldades deste processo, no sistema operativo MSDOS, tem a ver com o facto da concha (*shell*) não permitir a existência de dois sinais de `=` no mesmo comando, originando um erro de sintaxe.

O comando `option` do AMPL também pode ser usado para atribuir valores às mesmas opções

```
option nsips_options '[opção=valor]';
```

directamente na linha de comandos do AMPL ou no ficheiro modelo do problema, antes do comando `solve` do AMPL.

A variável de ambiente `OPTIONS_IN` indica o nome do ficheiro que deve ser lido pelo AMPL antes de ler o problema. A leitura deste ficheiro permite que sejam passadas opções ao AMPL usando comandos AMPL. Como exemplo, pode executar-se o seguinte comando em MSDOS

```
C:\AMPL\SOLVERS\sip\nsips>set OPTIONS_IN=.\nsipsopt.ini
```

em que o ficheiro `nsipsopt.ini` contém comandos AMPL que passam opções ao *solver*.

## 9.3 Selecção do método

O método usado por defeito é o método de penalidade descrito na Secção 8.2. Optou-se por este método uma vez que não necessita do software de resolução de problemas finitos NPSOL.

Esta opção do método pode ser modificada através do uso da opção `method` que pode tomar um dos seguintes valores:

- `disc_hett` para o método modificado de discretização de Hettich (Capítulo 6);
- `disc_halt` para o método modificado de discretização de Hettich usando pontos pseudo-aleatórios de Halton (Capítulo 6);
- `disc_reem` para o método modificado de discretização de Reemtsen (Capítulo 6);
- `penalty` para o método de penalidade (Secção 8.2);

- `penalty_m` para o método dos multiplicadores baseado nas funções de penalidade Lagrangeana aumentada e exponencial (Secção 8.2);
- `sqp` para o método de programação quadrática sequencial (Secção 8.1);
- `intp` para o método de pontos interiores (Secção 8.3).

Para seleccionar o método modificado de discretização de Hettich, a partir da variável de ambiente, as linhas de comandos devem ser

```
% nsips_options='method=disc_hett'
% export nsips_options
```

invocando depois o *solver* NSIPS.

## 9.4 Resultados apresentados pelo software NSIPS

Os problemas codificados na base de dados do SIPAMPL incluem alguns comandos que permitem visualizar a solução encontrada pelo *solver* de resolução. Pode editar-se o ficheiro modelo (ficheiro `.mod`) no sentido de se obter informação acerca dos valores da função objectivo, restrições e variáveis. A apresentação de resultados, específica de cada método, é descrita nas respectivas secções dos métodos. Cada método ao ser utilizado adiciona uma linha no ficheiro `results` do directório de trabalho corrente.

## 9.5 Opções para os métodos implementados no NSIPS

### 9.5.1 Método de discretização

As opções para o método de discretização são apresentadas na Tabela 9.1. A primeira coluna ("Opção") apresenta o nome da `opção`. A coluna "Tipo" apresenta o `valor` que a opção pode tomar: "Inteiro" é um número inteiro (exemplo:1, 10, 100); "Duplo" é um número real em dupla precisão (exemplo: 1.0, 1, 1e+1, -1e-1) e "Duplos" é uma lista de "Duplo" separada por vírgulas (exemplo: 0.1,0.2 0.3,0.4 0.1). A coluna "Omissão" indica o valor por defeito para a opção. A coluna "Descrição" apresenta uma breve descrição da opção. Sempre que a opção tenha uma correspondência com a notação usada nos algoritmos, ela é indicada.

### 9.5.2 Método de programação quadrática sequencial

As opções para o método de PQS são apresentadas na Tabela 9.2.

As colunas da Tabela 9.2 têm o mesmo significado que na Tabela 9.1.

Como o método de programação quadrática sequencial requer o cálculo numérico de integrais, as correspondentes opções são indicadas de seguida.

| Opção | Tipo | Omissão | Descrição |
|---|---|---|---|
| disc_dist | Duplo | 0.1 | Dois pontos são vizinhos se $\|x-y\|_2 <$disc_dist. |
| disc_eps | Duplo | 0.01 | $\widetilde{\epsilon}_0$. |
| disc_h | Duplos | 0.1 | $h_i$, $i = 1, \ldots, p$. |
| disc_h_mx | Inteiro | $100\times p$ | Número de pontos Halton adicionados em cada refinamento. |
| disc_ha_mx | Inteiro | $1000\times p$ | Número máximo de pontos Halton. |
| disc_inner | Inteiro | 100 | Número máximo de problemas PNL($\widetilde{T}[h^{k-1}]$) resolvidos para um dado $k$. |
| disc_k | Inteiro | 3 | Número máximo de refinamentos, $r$. |
| zero | Duplo | $10^{-6}$ | Aproximação de zero. Usado para encontrar pontos que tornam as restrições activas e para comparar dois pontos, $\widetilde{\delta}_k$ para todo o $k$. |

Tabela 9.1: Opções para o método de discretização

**Cálculo do integral**

A opção

```
pf_int=gaussian
```

selecciona a fórmula Gaussiana adaptativa (em cada subintervalo o integral é calculado numericamente por uma fórmula Gaussiana). A opção

```
pf_int=trapezoid
```

selecciona a fórmula adaptativa do trapézio (em cada subintervalo o integral é estimado pela fórmula do trapézio). A Tabela 9.3 apresenta as restantes opções para definir o cálculo do integral e as suas colunas têm o mesmo significado que na Tabela 9.1.

A fórmula adaptativa do trapézio é a usada por defeito.

### 9.5.3 Método de penalidade

A opção

```
pf_type=[p1|p2|p3]
```

selecciona a função de penalidade a usar. A opção usada por defeito é a `p3`. A Tabela 9.4 apresenta a relação entre as opções `p1`, `p2`, `p3` e as funções de penalidade.

As opções para o método de penalidade são apresentadas na Tabela 9.5. As colunas da Tabela 9.5 têm o mesmo significado que na Tabela 9.1.

São igualmente válidas as opções para o cálculo dos integrais das funções de penalidade

| Opção | Tipo | Omissão | Descrição |
|---|---|---|---|
| armijo | Duplo | $10^{-1}$ | Constante $\eta$ na condição de Armijo. |
| damped | Inteiro | 1 | 0 para a fórmula BFGS sem uso da estratégia *damped*. Outro inteiro para uso da fórmula com a estratégia *damped*. Veja-se em [99]. |
| maxiteri | Inteiro | 400 | Número máximo de iterações interiores. |
| maxitero | Inteiro | 400 | Número máximo de iterações exteriores. |
| pf_preci | Duplo | $10^{-4}$ | Tolerância para o critério de paragem para a iteração interior, $\delta_1$. |
| pf_preco | Duplo | $10^{-4}$ | Tolerância para o critério de paragem para a iteração exterior, $\delta_2$. |
| reset | Inteiro | 0 | 0 para a não reiniciação da aproximação à inversa da Hessiana. Outro inteiro para a reiniciação. |
| scale | Inteiro | 0 | 0 para o não escalonamento da direcção. Outro valor para o contrário. Se a norma da direcção não está entre $10^{-2}$ e $10^{2}$ a direcção é escalonada. |
| dual_ini | Inteiro | 1 | 0 se não se pretende a reiniciação da aproximação inicial para o problema de PSIQ. Outro valor para o contrário (usa sempre a aproximação inicial (8.1.29)). |

Tabela 9.2: Opções para o método de PQS

| Opção | Tipo | Omissão | Descrição |
|---|---|---|---|
| int_amp | Duplo | $10^{-2}$ | Amplitude mínima do intervalo, $\rho$. |
| int_error | Duplo | $10^{-8}$ | Precisão no cálculo do integral, $\epsilon_T$. |
| int_n | Inteiro | 20 | Número inicial de subintervalos na fórmula adaptativa do trapézio, $ni$, ou número de pontos na fórmula Gaussiana adaptativa. 6, 8, 16 são os números de pontos permitidos na fórmula Gaussiana adaptativa. |

Tabela 9.3: Opções no cálculo do integral

| Método | `pf_type` | Função de penalidade |
|---|---|---|
| `penalty` | `p1` | (8.2.10) |
| | `p2` | (8.2.11) |
| | `p3` | (8.2.12) |
| `penalty_m` | `p1` | (8.2.17) |
| | `p3` | (8.2.25) |

Tabela 9.4: Relação entre as opções e as funções de penalidade

já descritas na Subsecção 9.5.2.

### 9.5.4 Método de pontos interiores

São também válidas as opções já apresentadas na Subsecção 9.5.2 relativamente ao cálculo dos integrais. As opções para o método de pontos interiores são apresentadas na Tabela 9.6. As colunas da tabela têm o mesmo significado que na Tabela 9.1. "String" na coluna "Tipo" significa que a opção esperada é uma sequência de caracteres iniciada por uma letra.

## 9.6 Resultados apresentados pelos métodos implementados

### 9.6.1 Método de discretização

O método de discretização adiciona a linha

& gi & gf & med & nsub & fx \\

ao ficheiro de resultados (`results`). "gi" é o número de pontos na grelha inicial; "gf" é o número de pontos na grelha final; "med" é o número médio de pontos usados nos subproblemas finitos resolvidos (número de restrições finitas) - o número de restrições no primeiro subproblema finito resolvido não é contabilizado; "nsub" é o número de subproblemas finitos resolvidos; "fx" é o valor da função objectivo na solução encontrada. O primeiro & é

| Opção | Tipo | Omissão | Descrição |
|---|---|---|---|
| armijo | Duplo | $10^{-1}$ | Constante $\eta$ na condição de Armijo. |
| damped | Inteiro | 1 | 0 para a fórmula BFGS sem uso da estratégia *damped*. Outro inteiro para uso da fórmula com a estratégia *damped*. Veja-se em [99]. |
| maxiteri | Inteiro | 400 | Número máximo de iterações interiores. |
| maxitero | Inteiro | 400 | Número máximo de iterações exteriores. |
| mu0 | Duplo | 1 | Aproximação inicial para o parâmetro de penalidade, $\mu_0$. |
| muf | Duplo | 10 | Factor multiplicativo para o parâmetro de penalidade, $\mu_f$ |
| pf_preci | Duplo | $10^{-4}$ | Tolerância para o critério de paragem para a iteração interior, $\delta_1$. |
| pf_preco | Duplo | $10^{-4}$ | Tolerância para o critério de paragem para a iteração exterior, $\delta_2$. |
| pf_eps | Duplo | $10^{-4}$ | Valor inicial do parâmetro de suavização para a diferenciabilidade, $\epsilon_0$. |
| reset | Inteiro | 0 | 0 para a não reiniciação da aproximação à inversa da Hessiana. Outro inteiro para a reiniciação. |
| scale | Inteiro | 0 | 0 para o não escalonamento da direcção. Outro valor para o contrário. Se a norma da direcção não está entre $10^{-2}$ e $10^{2}$ a direcção é escalonada. |

Tabela 9.5: Opções para o método de penalidade

incluído por forma a que o nome do problema possa ser introduzido, no sentido de poder ser construída uma tabela de resultados em LaTeX. O *solver* que resolve o problema de PSI (executado pelo AMPL) não tem informação acerca do nome do problema e não pode fornecer o primeiro campo da tabela. Podem usar-se as seguintes instruções na linha de comandos

```
% echo -n "problema" >> ./results
% ampl problema.mod
```

para obter uma linha completa da tabela

problema & gi & gf & med & nsub & fx \\

e após incluir o início e fim da tabela, surge a linha em LaTeX

$$\| \text{problema} \mid \text{gi} \mid \text{gf} \mid \text{med} \mid \text{nsub} \mid \text{fx} \| \ .$$

As tabelas apresentadas no Capítulo 10 de experiências computacionais foram obtidas seguindo o procedimento aqui exposto.

### 9.6.2 Método de programação quadrática sequencial

O método de PQS adiciona a linha

& ie & nm & ngm & nc & nj & fx \\

ao ficheiro de resultados (`results`), em que "ie" é o número de iterações exteriores; "nm" é o número de cálculos da função mérito; "ngm" é o número de cálculos do gradiente da função mérito ; "nc" é o número de cálculos das restrições; "nj" é o número de cálculos do Jacobiano das restrições e "fx" é o valor da função objectivo na solução encontrada.

Aplicam-se os mesmos comentários da Subsecção 9.6.1 para a construção de uma tabela de resultados em LaTeX.

### 9.6.3 Método de penalidade

O método de penalidade adiciona a linha

& ti & ne & nfp & ngfp & fx & mu & eps \\

no ficheiro de resultados `results`. "ti" é o número total de iterações interiores; "ne" é o número de iterações exteriores; "nfp" é o número de cálculos da função de penalidade; "ngfp" é o número de cálculos do gradiente da função de penalidade; "fx" é o valor da função

objectivo na solução encontrada; "mu" é o valor final do parâmetro de penalidade e "eps" é o valor final de $\epsilon$, parâmetro de suavização.

Os mesmos comentários da Subsecção 9.6.1 aplicam-se para a construção da tabela de resultados em LaTeX.

### 9.6.4 Método de pontos interiores

O método de pontos interiores adiciona a linha

& iter & nlag & nmer & fx & $||\rho||_2$ & $||\sigma||_2$ & $\mu$ & $\epsilon$ \\

ao ficheiro de resultados (`results`), sendo "iter" o número de iterações; "nlag" o número de cálculos da função Lagrangeana; "nmer" o número de cálculos da função mérito; "fx" o valor da função objectivo na solução encontrada; "$||\rho||_2$" e "$||\sigma||_2$" respectivamente a não admissibilidade primal e dual e "$\epsilon$" o último $\epsilon$ usado na função aproximação (8.3.1).

Aplicam-se os mesmos comentários da Subsecção 9.6.1 para a construção de uma tabela de resultados em LaTeX.

O estudo original [158] do método de pontos interiores considerou os multiplicadores de Lagrange como quantidades negativas, pelo que essa é a correspondente implementação no NSIPS.

## 9.7 Limitações do software NSIPS

Durante o desenvolvimento do *solver* NSIPS não houve uma preocupação de construir um software robusto. Se, por exemplo, o utilizador codificar um problema de maximização e/ou as restrições forem desigualdades do tipo $\geq$, o software pode responder de uma forma inesperada.

O método de discretização é, dos métodos implementados, o de aplicação mais generalizada, uma vez que o uso do software NPSOL permite incluir restrições finitas e de limites simples. Os problemas resolvidos por este método podem ser da forma geral (1.1.1).

Os métodos baseados em funções de penalidade, PQS e pontos interiores usam as rotinas que calculam os integrais, respectivamente nas funções de penalidade, na função mérito e no cálculo da direcção e função mérito. Na versão actual do NSIPS estas rotinas apenas calculam integrais simples (a uma dimensão). Por isso, estes métodos só podem ser usados na resolução de problemas com uma variável infinita.

Para os métodos de penalidade, PQS e pontos interiores não foi ainda desenvolvido o algoritmo para as restrições finitas. Os métodos resolvem apenas problemas com restrições infinitas, ignorando as restrições finitas.

As restrições infinitas têm de ser desigualdades do tipo $\leq$. São indicados na Tabela 9.7 os problemas que foram reescritos para satisfazerem a condição de que as restrições devem ser do tipo desigualdade simples ($\leq$). A primeira coluna apresenta o nome do problema

na base de dados do SIPAMPL e a segunda o novo nome do problema, para uma resolução pelos métodos de penalidade e PQS.

Em todos os métodos, usa-se o procedimento descrito na Secção 6.2, se o problema a ser resolvido não possui uma aproximação inicial à solução.

| Opção | Tipo | Omissão | Descrição |
|---|---|---|---|
| int_merit | String | `m1` | `m1` para a função mérito $l_2$ (8.3.19). `m2` para a função mérito baseada na Lagrangeana aumentada (8.3.20). |
| damped | Inteiro | 1 | 0 para a fórmula BFGS sem uso da estratégia *damped*. Outro inteiro para uso da fórmula com a estratégia *damped*. Veja-se em 8.3.3 e [99]. |
| int_maxit | Inteiro | 1000 | Número máximo de iterações permitidas. |
| int_prec | Duplo | $10^{-3}$ | Não admissibilidade primal e dual permitida no critério de paragem, $\delta_f$. |
| int_eps | Duplo | $10^{-4}$ | Valor inicial do parâmetro de suavização da função aproximação, $\epsilon_0$ ($\tau_0 = \epsilon_0$) |
| int_epsfactor | Duplo | 0.1 | Factor de redução do parâmetro de suavização, $\xi_\tau$. |
| int_epslimit | Duplo | $10^{-8}$ | Valor mínimo para o parâmetro de suavização. |
| reset | Inteiro | 0 | 0 para que a aproximação à Hessiana não seja reiniciada. Outro valor inteiro para que $B_k$ seja reiniciada com a matriz identidade, sempre que $k$ é múltiplo de $n$. |
| scale | Inteiro | 0 | 0 para o não reescalonamento da direcção. Outro valor para o contrário. Se a norma da direcção não está entre $10^{-2}$ e $10^2$ a direcção é escalonada. |
| theta | Duplo | 1.0 | Valor usado para evitar a proximidade das variáveis de folga da fronteira, $\theta$ em (8.3.31). |
| delta | Duplo | 0.1 | Factor de redução no cálculo do parâmetro barreira, $\delta_\mu$ em (8.3.33). |
| watchmax | Inteiro | 3 | Número máximo de aumentos da função mérito consecutivos. |
| watchdog | Inteiro | 0 | 0 para não usar a estratégia de cão de guarda e outro valor em caso contrário. |
| bfgsskip | Duplo | $10^{-8}$ | A actualização BFGS da matriz Hessiana não será efectuada se a condição de curvatura for inferior ao valor dado, $\delta_c$. |
| int_ord | String | `d` | `p` para ordem primal, `d` para ordem dual e `pc` para preditor corrector. |
| int_corr | Inteiro | 1 | Número de correcções à direcção preditora, $m_c$. |

Tabela 9.6: Opções para o método de pontos interiores

| Problema | Problema na forma padrão |
|---|---|
| elke1 | elke1std |
| elke2 | elke2std |
| elke3 | elke3std |
| elke4 | elke4std |
| elke5 | elke5std |
| elke6 | elke6std |
| elke7 | elke7std |

Tabela 9.7: Problemas na forma padrão

# Capítulo 10

# Experiências computacionais

---

São apresentadas, neste capítulo, as experiências computacionais realizadas ao longo deste projecto. Os resultados obtidos com os problemas de robótica e de desenho óptimo de sinais foram já analisados com algum detalhe no Capítulo 7. Incluem-se os resultados conseguidos com os quatro tipos de métodos implementados no *solver* NSIPS, na resolução dos problemas que constam da base de dados do SIPAMPL.

---

Os resultados numéricos foram obtidos num computador com processador Pentium III a 450Mhz com 128MB de memória RAM e num sistema operativo Linux (Red Hat 5.2) com a versão limitada do AMPL (*Student Version*) número 19991027 (Linux 2.0.18).

As quatro secções seguintes apresentam os resultados obtidos com os métodos implementados no *solver* NSIPS na resolução dos problemas que os métodos se propõem resolver: a Secção 10.1 para o método de discretização, a Secção 10.2 para o método de programação quadrática sequencial, a Secção 10.3 para o método de penalidade e a Secção 10.4 para o método de pontos interiores. Na última secção apresentam-se algumas conclusões retiradas na análise dos resultados numéricos.

As tabelas completas com os resultados numéricos são apresentadas no Apêndice J.

## 10.1 Método de discretização

Em [55] Hettich apresentou alguns resultados numéricos para o algoritmo correspondente ao Algoritmo 6.3.1. O algoritmo foi aplicado a dois problemas de aproximação de Chebyshev baseados em polinómios como funções de aproximação (`hettich6` e `hettich7` na base de dados do SIPAMPL).

Os problemas de aproximação de Chebyshev são problemas de PSI lineares e os subproblemas finitos correspondentes são também lineares. Hettich usou um algoritmo Simplex do tipo primal-dual. Os resultados apresentados por Hettich ([55]) e os da versão modificada aqui apresentada estão registados na Tabela 10.1. Nestes casos foram usados os

parâmetros propostos por Hettich: $h = (h_1, h_2) = (0.1, 0.15)$ com três refinamentos, resultando em 121 pontos na grelha inicial e 32761 na grelha final. Hettich não mencionou aproximações iniciais à solução e por isso foi usado o procedimento descrito na Secção 6.2. A notação usada na Tabela 10.1 é a seguinte: "Problema" designa o nome do problema na base de dados do SIPAMPL; "d" é o grau do polinómio de aproximação (é possível editar o ficheiro que contém os problemas *hettich6* e *hettich7* e modificar o grau do polinómio, aumentando a dificuldade de resolução do problema); "nx" é o número de variáveis finitas ($n$); "MR" e "MR(H)" são as médias do número de restrições usadas nos subproblemas finitos resolvidos (o número de restrições do primeiro problema não é contabilizado). A indicação "MR" diz respeito ao resultados aqui obtidos e "MR(H)" aos obtidos por Hettich. "N" e "N(H)" indicam o número de subproblemas finitos resolvidos e "fx" é o valor da função objectivo.

| Problema | d | nx | MR | MR(H) | N | N(H) | fx |
|---|---|---|---|---|---|---|---|
| hettich6 | 2 | 7 | 26.4 | 24 | 6 | 5 | 2.80e-2 |
| hettich6 | 3 | 11 | 40.6 | 38 | 7 | 8 | 3.47e-3 |
| hettich6 | 4 | 16 | 99.3 | 88 | 7 | 12 | 6.96e-4 |
| hettich6 | 5 | 22 | 176 | 90 | 9 | 13 | 1.62e-4 |
| hettich6 | 6 | 29 | 473.3 | 140 | 7 | 15 | 3.96e-5 |
| hettich6 | 7 | 37 | 1072 | 221 | 6 | 12 | 1.00e-5 |
| hettich7 | 2 | 7 | 24 | 23 | 6 | 6 | 1.78e-1 |
| hettich7 | 3 | 11 | 36.3 | 33 | 7 | 5 | 3.66e-2 |
| hettich7 | 4 | 16 | 65.5 | 56 | 9 | 11 | 4.67e-3 |
| hettich7 | 5 | 22 | 88.2 | 79 | 11 | 11 | 7.38e-4 |
| hettich7 | 6 | 29 | 160.9 | 108 | 8 | 12 | 7.67e-5 |
| hettich7 | 7 | 37 | 384.4 | 145 | 6 | 10 | 8.79e-6 |

Tabela 10.1: Comparação com os resultados obtidos por Hettich (H)

O algoritmo de Reemtsen ([116]) foi proposto para problemas de PSI quadráticos. Apesar disso, os resultados numéricos apresentados em [116] dizem respeito apenas a problemas de aproximação de Chebyshev (codificados de `reemtsen1` até `reemtsen5` na base de dados do SIPAMPL). Uma vez que os problemas de aproximação de Chebyshev são problemas de PSI lineares, Reemtsen usou a rotina de programação quadrática DQPROG da biblioteca IMSL para resolver os subproblemas finitos e não foram indicadas aproximações iniciais às soluções. Os resultados obtidos por Reemtsen e os obtidos pela versão modificada aqui apresentada estão resumidos na Tabela 10.2. Foram usados parâmetros idênticos: $\widetilde{\delta}_i = 0$, para todo o $i$ e os refinamentos são baseados em $h^{k+1} = h^k/2$. A notação usada na Tabela 10.2 é a seguinte. "Problema" é o nome do problema na base de dados do SIPAMPL, "d/t" é um parâmetro do problema que indica o grau do polinómio de aproximação, "nx" é o número de variáveis finitas do problema (parametrizado por "d/t"), "h" é o espaçamento inicial usado para formar a grelha de pontos, "$\epsilon$" é o valor inicial de $\widetilde{\epsilon}_k$, "GI" é o número de pontos na grelha inicial, "GF" é o número de pontos na grelha final, "l" é o número

de grelhas usadas (= $r + 1$, em que $r$ é o número de refinamentos)e "MR" e "MR(R)" são médias do número de restrições usadas nos subproblemas finitos. Mais uma vez a primeira grelha de pontos não é contabilizada, quer nos resultados aqui obtidos quer nos reportados por Reemtsen. "N" e "N(R)" representam o número de subproblemas resolvidos e "fx" é o valor da função objectivo.

| Problema | d/t | nx | h | $\epsilon$ | GI | GF | l | MR | MR(R) | N | N(R) | fx |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reemtsen1 | 2 | 11 | 0.2 | 0.02 | 216 | 68921 | 3 | 48.7 | 24 | 4 | 4 | 1.52e-1 |
| reemtsen1 | 3 | 21 | 0.2 | 0.03 | 216 | 68921 | 3 | 79.7 | 40 | 7 | 7 | 3.11e-2 |
| reemtsen1 | 4 | 36 | 0.2 | 0.03 | 216 | 68921 | 3 | 129.6 | 67 | 12 | 12 | 4.88e-3 |
| reemtsen1 | 5 | 57 | 0.2 | 0.04 | 216 | 68921 | 3 | 219.3 | 102 | 10 | 11 | 7.09e-4 |
| reemtsen2 | 2 | 10 | 2/3 | 0.01 | 16 | 591361 | 8 | 20.3 | 11 | 7 | 12 | 5.84e-2 |
| reemtsen3 | 2 | 10 | 2/3 | 0.01 | 16 | 591361 | 8 | 22 | 16 | 7 | 9 | 7.35e-1 |
| reemtsen4 | 2 | 10 | 2/3 | 0.01 | 16 | 591361 | 8 | 18.5 | 9 | 9 | 6 | 1.14e-2 |
| reemtsen4 | 3 | 17 | 2/7 | 0.01 | 64 | 201601 | 8 | 132.6 | 15 | 14 | 6 | 2.75e-3 |
| reemtsen4 | 4 | 26 | 2/7 | 0.02 | 64 | 201601 | 8 | 817.4 | 36 | 20 | 13 | 7.40e-4 |
| reemtsen4 | 5 | 37 | 2/7 | 0.02 | 64 | 201601 | 8 | 24 | 73 | 5 | 13 | 2.13e-4 |
| reemtsen5 | 2 | 11 | 0.25 | 0.01 | 125 | 274625 | 4 | 137.5 | 74 | 5 | 5 | 8.89e-2 |
| reemtsen5 | 3 | 21 | 0.25 | 0.01 | 125 | 274625 | 4 | 269.6 | 59 | 15 | 12 | 4.81e-2 |

Tabela 10.2: Comparação com os resultados obtidos por Reemtsen (R)

Nas tabela J.1, J.2 e J.3 são apresentados os resultados relevantes obtidos pelos métodos modificados de discretização na resolução de todos os problemas da base de dados do SIPAMPL a que o método se aplica.

A Tabela 10.3 apresenta um resumo dos resultados obtidos com o método de discretização em termos dos valores médios e número de falhas. A versão de Hettich com pontos pseudo-aleatórios não pode ser comparada com as restantes, visto que o número de pontos nos subconjuntos intermédios é diferente.

| Versão | Média de restrições | Média de subproblemas | Falhas |
|---|---|---|---|
| Hettich | 215.74 | 5.53 | 2 |
| Reemtsen | 447.78 | 2.79 | 6 |

Tabela 10.3: Médias dos resultados para os métodos de discretização

As falhas na resolução dos problemas é devida à não convergência do software NPSOL nos subproblemas finitos.

Podemos concluir que o método de discretização na versão Hettich é mais robusto, no sentido de que consegue resolver um maior número de problemas. Por seu lado, a versão

Reemtsen resolve um menor número de subproblemas finitos, mas com um maior número médio de restrições.

A versão Hettich com pontos pseudo-aleatórios é mais indicada para a resolução de problemas em que o número de variáveis infinitas é elevado e se pretenda ter um maior controlo sobre o número de pontos iniciais e finais.

## 10.2 Método de programação quadrática sequencial

Apresentam-se, nesta secção, os resultados obtidos pela implementação do método de programação quadrática sequencial baseado na parametrização das variáveis duais, tal como foi descrito na Secção 8.1.

Para os problemas que não possuem aproximação inicial foi usado o procedimento descrito na Secção 6.2.

Os integrais (8.1.31) foram calculados através da fórmula adaptativa do trapézio com as opções definidas por defeito.

As Tabelas J.4 e J.5 mostram os resultados numéricos obtidos para os problemas seleccionados que não têm restrições finitas e têm apenas uma variável infinita $t$. O método não conseguiu resolver os problemas de robótica (de `elke1std` a `elke7std`, `elke8`, `elke9` e `elke10`).

A Tabela 10.4 apresenta os resultados médios obtidos com o método de programação quadrática sequencial. Foram contabilizados como falhas os problemas em que o número de iterações é superior a 400 e os 10 problemas de robótica.

| Versão | Iterações Exteriores | Mérito | Restrições | Jacobiano | Falhas |
|---|---|---|---|---|---|
| Sem reiniciação | 19.71 | 68.02 | 2589910.90 | 2564547.83 | 12 |
| Com reiniciação | 7.29 | 42.05 | 1106048.28 | 1092595.32 | 10 |

Tabela 10.4: Médias dos resultados para o método de programação quadrática sequencial

A falha nos problemas de robótica é devida à não convergência do método, enquanto que a falha nos restantes dois problemas é devida a se ter atingido o número máximo de iterações e a um erro numérico (avaliação da exponencial de um número elevado).

A versão com reiniciação da aproximação inicial na resolução dos problemas quadráticos é melhor em todos parâmetros aqui apresentados.

## 10.3 Método de penalidade

Da implementação dos métodos de penalidade e dos multiplicadores baseados nas funções $\phi_S^1$, $\phi_S^2$, $\phi_S^3$, $\phi_{LA}$, e $\phi_E$ descritos na Secção 8.2, foram obtidos resultados numéricos para os problemas da base de dados do SIPAMPL caracterizados por possuírem apenas uma variável infinita e restrições infinitas.

Foi usado o procedimento descrito na Secção 6.2 nos problemas da base de dados que não possuem aproximação inicial.

As Tabelas J.6, J.7 e J.8 contêm os resultados obtidos da implementação do método de penalidade com base respectivamente nas funções $\phi_S^1$, $\phi_S^2$, $\phi_S^3$ com $\mu_f = 10$.

Os resultados da implementação do método dos multiplicadores com base em $\phi_{LA}$ e $\mu_f = 10$ são apresentados na Tabela J.9. As Tabelas J.10 e J.11 contêm os resultados numéricos do método dos multiplicadores que usa a função de penalidade $\phi_E$. A primeira tabela corresponde a uma actualização do parâmetro de penalidade com $\mu_f = 1.1$ e a segunda com $\mu_f = 2$. Devido à natureza da função envolvida no termo de penalidade, a função $\phi_E$ requer uma actualização do parâmetro de penalidade mais suave do que as outras funções implementadas.

A Tabela 10.5 apresenta os resultados médios para todas as funções de penalidade. "TII méd" é a média do total de iterações interiores e "TIE méd" é a média do total de iterações exteriores. "FP méd" é a média do número de cálculos da função de penalidade e "GFP méd" é a média do número de cálculos de gradientes da função de penalidade. "Falhas" indica o número de problemas que não convergiram.

| Função de Penalidade | $\phi_S^1$ | $\phi_S^2$ | $\phi_S^3$ | $\phi_{LA}$ | $\phi_E$ | $\phi_E$ |
|---|---|---|---|---|---|---|
| $\mu_f$ | 10.0 | 10.0 | 10.0 | 10.0 | 1.1 | 2.0 |
| TII méd | 143.6 | 248.31 | 94.46 | 196.02 | 385.54 | 257.51 |
| TIE méd | 4.69 | 4.76 | 4.88 | 8.8 | 89.59 | 18.62 |
| FP méd | 536.1 | 832.57 | 462.5 | 746.41 | 2400.82 | 1093.85 |
| GFP méd | 148.09 | 252.75 | 99.28 | 203.57 | 474.91 | 275.84 |
| Falhas | 8 | 3 | 25 | 19 | 7 | 20 |

Tabela 10.5: Médias dos resultados para as diferentes funções de penalidade

Os problemas em que o algoritmo falhou foram removidos do cálculo da média e considerou-se como falha quando o $\mu$ final é superior a $10^8$, uma vez que o algoritmo parou porque $\mu$ ultrapassou o valor máximo permitido. Note-se que a falha pode ser assumida e no entanto ser encontrada uma solução aproximada. Em alguns problemas a falha é devida a um erro numérico (avaliação da exponencial de um número elevado).

Da Tabela 10.5 pode concluir-se que a função de penalidade $\phi_S^2$ é mais robusta do que as outras e a função $\phi_S^3$ é a mais eficiente, no sentido de que exige, em média, menor

| Ordenação | Dual | | Primal | |
|---|---|---|---|---|
| Função mérito | $l_2$ | Aumentada | $l_2$ | Aumentada |
| Média de iterações | 80.84 | 97.79 | 108.56 | 139.26 |
| Média da função mérito | 328.91 | 362.68 | 407.19 | 488.64 |
| Falhas (>400 iter. e NaN) | 32 | 19 | 39 | 36 |

Tabela 10.6: Valores médios para a ordenação dual e primal

número de iterações e de cálculos da função e do gradiente até convergir. As experiências mostraram que a função $\phi_E$ é muito sensível à actualização do parâmetro de penalidade $\mu$. Quanto menor for o factor $\mu_f$ maior é a robustez. Em contrapartida o processo iterativo torna-se mais lento.

## 10.4 Método de pontos interiores

Os problemas da base de dados do SIPAMPL que não incluem qualquer restrição finita e que possuem apenas uma variável infinita foram seleccionados e resolvidos pelo método de pontos interiores apresentado na Secção 8.3.

Mais uma vez usou-se o procedimento descrito na Secção 6.2 para encontrar uma aproximação inicial para os problemas que não contêm essa informação no ficheiro modelo.

Os resultados numéricos obtidos da implementação do método de pontos interiores são apresentados nas Tabelas J.12, J.13, J.14 e J.15. Estas tabelas correspondem respectivamente às versões baseadas na ordenação dual com função mérito $l_2$ e com função mérito $\mathcal{L}_A$ e na ordenação primal com função mérito $l_2$ e com $\mathcal{L}_A$.

Os valores médios do número de iterações e do número de cálculos da função mérito e as falhas são apresentados na Tabela 10.6. A ordenação primal reduz o número de sucessos para ambas as funções mérito. Embora a opção preditor corrector tenha sido testada em combinação com a técnica de cão de guarda, os resultados foram desanimadores.

As falhas que não resultam da ultrapassagem do limite de iterações, são devidas a erros numéricos (obtenção de NaN (*Not a Number*) e de divisão por zero).

## 10.5 Conclusões dos resultados numéricos

Terminada a fase experimental deste projecto e face aos resultados obtidos surge uma questão relacionada com a escolha do método mais adequado à resolução de um problema de PSI.

De acordo com o estado actual de desenvolvimento dos algoritmos propostos, se o problema possuir restrições finitas de qualquer tipo (igualdade, desigualdade ou limites simples) ou mais do que uma variável infinita, a escolha recai sobre o método de discretização. Este método, na versão de Hettich, é o que resolve um maior número de problemas com sucesso. No entanto, tanto o método de discretização como o método de programação

quadrática sequencial só se tornam funcionais para o sistema operativo Linux, se se puder recorrer ao software NPSOL para a resolução dos subproblemas finitos.

Se o problema não possuir restrições finitas de qualquer tipo e tiver apenas uma variável infinita deve usar-se em primeiro lugar o método de penalidade baseado na função $\phi_S^2$ (8.2.11), uma vez que é o método com menor número de insucessos depois do método de discretização.

A existência de uma aproximação inicial à solução, para incluir no ficheiro do problema, é outro factor importante para a selecção do método. Se o problema não tiver aproximação inicial é necessário possuir uma versão do NSIPS com NPSOL. Em alternativa, pode colocar-se uma qualquer aproximação no ficheiro do problema, sem contudo haver garantia de que seja uma boa aproximação à solução.

Uma última consideração a fazer sobre os resultados obtidos tem a ver com o facto de, para a grande maioria dos problemas, não se conhecerem as soluções exactas, pelo que não é possível tirar conclusões sobre a precisão das soluções obtidas.

# Capítulo 11

# Conclusões e trabalho futuro

---

As conclusões que podem ser retiradas do trabalho desenvolvido no âmbito deste projecto são apresentadas neste capítulo. Apresentam-se também novas direcções de trabalho que resultam principalmente de extensões e melhoramentos a introduzir nos métodos desenvolvidos para a PSI não linear e de novos desenvolvimentos a incluir nos pacotes SIPAMPL e NSIPS para aumentar as suas potencialidades.

---

## 11.1 Conclusões

O desenvolvimento do pacote de software SIPAMPL vem guarnecer a PSI de uma nova ferramenta que permite:

- a fácil codificação de problemas de PSI;
- o uso de uma base de dados com inúmeros problemas de PSI já codificados;
- o uso de uma interface para ligação de software de resolução de problemas de PSI à base de dados;
- o uso da ferramenta *select*, que permite seleccionar problemas com determinadas características da base de dados dos problemas;
- a ligação à função do MATLAB que resolve problemas de PSI;
- a codificação de problemas que usem "B-Splines", através do uso de uma biblioteca dinâmica de "B-Splines".

A codificação de problemas de PSI em AMPL permite aliviar o utilizador do cálculo de derivadas, uma vez que o AMPL possui derivação automática. Evita por isso o trabalho de dedução das fórmulas e de codificação, diminuindo assim a possibilidade de erros.

A referência apenas aos nomes dos problemas que se encontram numa base de dados, disponível ao público, é mais simples e evita erros de apresentação dos problemas em trabalhos futuros.

A interface do SIPAMPL incentiva os investigadores à construção de software para resolver problemas de PSI e proporciona a comparação com outros códigos existentes.

O *solver* NSIPS usa a interface do SIPAMPL para resolver os problema de PSI existentes na base de dados. Este software é o primeiro código numérico disponível ao público em formato binário e de código fonte. Implementa quatro métodos diferentes num total de sete versões.

Um dos métodos implementados corresponde a uma variante do método de discretização desenvolvida para problemas de PSI não linear. Embora este método, na versão actual, esteja dependente do software NPSOL para a resolução dos subproblemas finitos, foi o que resolveu uma maior percentagem de problemas.

O método de programação quadrática sequencial, que recorre a uma estratégia simples de parametrização das variáveis duais para a resolução dos problemas quadráticos, origina problemas finitos com muitas variáveis e restrições, mas no cômputo global é robusto. A resolução dos subproblemas finitos está também dependente do software NPSOL.

Com as duas versões do método de penalidade, uma baseada em funções de penalidade simples e a outra baseada em funções que envolvem o vector dos multiplicadores, uma Lagrangeana aumentada e uma função exponencial, obtiveram-se resultados satisfatórios. As experiências computacionais realizadas parecem apontar no sentido de que uma estratégia diferente de actualização do parâmetro de penalidade irá aumentar a eficácia deste tipo de métodos.

Finalmente, o método primal-dual de pontos interiores não admissível apresenta valores aceitáveis do número de iterações e do número de cálculos da função mérito, mas não consegue convergir para a solução em mais problemas do que os outros métodos.

A existência dos pacotes SIPAMPL e NSIPS permite assim a codificação e resolução de um problema de PSI de uma forma fácil e rápida.

A versão 1.0 do *solver* NSIPS esteve disponível ao público no endereço (*site*) do NEOS *server*[1]. Desde 25 de Fevereiro de 2003 encontra-se disponível a versão 2.1 do NSIPS, em uso do SIPAMPL versão 2.0. Esta implementação, com todos os métodos operacionais, no NEOS *server* é devida a Hans D. Mittelmann e permite que sejam submetidos problemas de PSI e obtidos os respectivos resultados numéricos através da internet.

## 11.2 Trabalho futuro

Uma das actuais limitações do *solver* NSIPS é a sua dependência do software NPSOL para a resolução dos subproblemas finitos. Um desenvolvimento que torna o *solver* mais acessível a qualquer utilizador é a extensão a outros pacotes de software (NAG [76], IMSL [100], LOQO [148], etc) para a resolução dos subproblemas finitos.

[1] http://www-neos.mcs.anl.gov/neos/solvers/SIO:NSIPS/

No contexto dos métodos desenvolvidos, designadamente do método de programação quadrática sequencial e dos métodos que recorrem à transcrição das restrições infinitas em restrições finitas, como o método de penalidade e o de pontos interiores, existem ideias relativamente à extensão do cálculo do integral para a integração múltipla, permitindo a resolução de problemas com mais do que uma variável infinita.

A inclusão de novos métodos no pacote NSIPS, nomeadamente de métodos de redução que utilizem técnicas de optimização global eficazes, é outra ideia que será concretizada no futuro.

Para a maior parte dos problemas não se conhecem as soluções, o que dificulta, por vezes, a comparação entre códigos de resolução de problemas. Para resolver esta limitação, pensa-se desenvolver uma técnica/algoritmo que gere aleatoriamente problemas de PSI e as correspondentes soluções.

Para a codificação dos problemas de trajectória de robôs descritos na Subsecção 7.1.2 é necessário proceder à construção de uma biblioteca de "C-Splines" (*splines* cúbicas), à imagem da biblioteca de "B-Splines".

Relativamente à base de dados de problemas de PSI do SIPAMPL, uma extensão natural considera a codificação de novos problemas, à medida que forem aparecendo na literatura.

A actual interface do MATLAB ao SIPAMPL é limitativa, no sentido de que é vocacionada para o uso da função `fseminf`. Pensa-se desenvolver uma nova interface de ligação ao MATLAB mais flexível que apoiará o potencial desenvolvimento de novos *solvers* para o MATLAB que possam usar a base de dados de problemas de PSI do SIPAMPL.

# Bibliografia


[1] M.D. Ašić e V.V. Kovačević-Vujčić. An interior semi-infinite programming method. *Journal of Optimization Theory and Applications*, 59(3):353–367, 1988.

[2] I. Adler, M.G.C. Resende, G. Veiga, e N. Karmarkar. An implementation of karmarkar's algorithm for linear programming. *Mathematical Programming*, 44:297–335, 1989.

[3] E.J. Anderson e A.S. Lewis. An extension of the simplex algorithm for semi-infinite linear programming. *Mathematical Programming*, 44:247–269, 1989.

[4] P. Armand, J.C. Gilbert, e S. Jan-Jégou. A feasible BFGS interior point algorithm for solving strongly convex minimization problems. *SIAM Journal on Optimization*, 11:199–222, 2000.

[5] P. Armand, J.C. Gilbert, e S. Jan-Jégou. A BFGS-IP algorithm for solving strongly convex optimization problems with feasibility enforced by an exact penalty approach. Technical Report 4087, INRIA - Institut National de Recherche en Informatique et en Automatique, December 2000. Optimization Online.

[6] M.S. Bazaraa, H.D. Sherali, e C.M. Shetty. *Nonlinear Programming, Theory and Algorithms*. John Wiley & Sons, Inc., segunda edição, 1993.

[7] H.Y. Benson, D.F. Shanno, e R.J. Vanderbei. Interior-point methods for nonconvex nonlinear programming: Filter methods and merit functions. Technical Report ORFE-00-06, Princeton University, 2000.

[8] H.Y. Benson, D.F. Shanno, e R.J. Vanderbei. Interior-point methods for nonconvex nonlinear programming: Jamming and comparative numerical testing. Technical Report ORFE-00-02, Princeton University, 2000.

[9] C.G. Lo Bianco e A. Piazzi. A semi-infinite optimization approach to optimal spline trajectory planning of mechanical manipulators. *Em [41]*, pp. 271–297, 2001.

[10] J.W. Blankenship e J.E. Falk. Infinitely constrained optimization problems. *Journal of Optimization Theory and Applications*, 19(2):261–281, Junho 1976.

[11] I. Bongartz, A.R. Conn, N. Gould, e Ph.L. Toint. CUTE: Constrained and Unconstrained Testing Environment. *ACM Transactions on Mathematical Software*, 21(1):123–160, 1995. ftp://130.246.9.91/pub/cute.

[12] C. Boor. *A Pratical Guide to Splines*. Springer-Verlag, 1978.

[13] D. Boukari e V. Fiacco. Survey of penalty, exact-penalty and multiplier methods from 1968 to 1993. *Optimization*, 32:301–334, 1995.

[14] A. Brooke, D. Kendrick, A. Meeraus, e Ramesh Raman. GAMS: A User's Guide. http://www.gams.com/, Dezembro 1998.

[15] R.H. Byrd, J.C. Gilbert, e J. Nocedal. A trust region method based on interior point tecniques for nonlinear programming. *Mathematical Programming*, 89:149–185, 2000.

[16] R.H. Byrd, M.R. Hribar, e J. Nocedal. An interior point algorithm for large scale nonlinear programming. *SIAM Journal on Optimization*, 9(4):877–900, 2000.

[17] T.J. Carpenter, I.J. Lustig, J.M. Mulvey, e D.F. Shanno. Higher-order predictor-corrector interior point methods with application to quadratic objectives. *SIAM Journal on Optimization*, 3(4):696–725, 1993.

[18] T. Coleman, M.A. Branch, e A. Grace. *Optimization Toolbox for Use with MATLAB*. The MathWorks Inc., 1999.

[19] A.R. Conn. Nonlinear programming, exact penalty functions and projection techniques for non-smooth functions. In P.T. Boggs, R.H. Byrd, e R.B. Schnabel, editors, *Numerical Optimization 1984*. SIAM, 1985.

[20] A.R. Conn e N.I.M. Gould. An exact penalty function for semi-infinite programming. *Mathematical Programming*, 37:19–40, 1987.

[21] A.R. Conn, N.I.M. Gould, e Ph.L. Toint. *LANCELOT: A Fortran Package for Large-Scale Nonlinear Optimization (Release A)*, volume 17. Springer-Verlag, Heidelberg, Berlin, 1992.

[22] I.D. Coope e G.A. Watson. A projected lagrangian algorithm for semi-infinite programming. *Mathematical programming*, 32(3):337–356, 1985.

[23] J.J. Craig. *Introduction to Robotics, Mechanics and Control*. Addison-Wesley, segunda edição, 1989.

[24] J. Czyzyk, S. Mehrotra, e S.J. Wright. PCx user guide. Technical Report OTC 96/01, Optimization Technology Center, Argonne National Laboratory and Nothwestern University, 1996.

[25] R. Hettich (Ed.). *Semi-Infinite Programming*. Springer-Verlag, 1979.

[26] A.S. El-Bakry, R.A. Tapia, T. Tsuchiya, e Y. Zhang. On the formulation and theory of the Newton interior-point method for nonlinear programming. *Journal of Optimization Theory and Applications*, 89(3):507–541, 1996.

[27] S.-C. Fang, C.-J. Lin, e S.-Y. Wu. On solving convex quadratic semi-infinite programming problems. *Optimization*, 21:107–125, 1994.

[28] S.-C. Fang e S.-Y. Wu. An inexact approach to solving linear semi-infinite programming problems. *Optimization*, 28:291–299, 1994.

[29] M.C. Ferris e A.B. Philpott. An interior point algorithm for semi-infinite linear programming. *Mathematical Programming*, 43:257–276, 1989.

[30] M.C. Ferris e A.B. Philpott. On affine scalling and semi-infinite programming. *Mathematical Programming*, 56:361–364, 1992.

[31] A.V. Fiacco e K.O. Kortanek (Eds.). *Semi-Infinite Programming and Applications*. Springer-Verlag, 1983.

[32] R. Fletcher, N.I.M. Gould, S. Leyffer, e P.L. Toint. Global convergence of trust-region SQP-filter algorithms for general nonlinear programming. Technical Report 99/03, Department of Mathematics, University of Dundee, 1999. http://www.numerical.rl.ac.uk/reports/reports.html.

[33] R. Fletcher e S. Leyffer. Nonlinear programming without a penalty function. Numerical analysis report, NA/171, University of Dundee, September 1997.

[34] R. Fletcher, S. Leyffer, e P.L. Toint. On the global convergence of an SLP-filter algorithm. Technical Report 98/13, Department of Mathematics, Facultés Universitaires ND de la Paix, 1998. ftp://thales.math.fundp.ac.be/pub/reports.

[35] A. Forsgren e P.E. Gill. Primal-dual interior point methods for nonconvex nonlinear programming. *SIAM Journal on Optimization*, 8(4):1132–1152, 1998.

[36] R. Fourer, D.M. Gay, e B.W. Kernighan. A modeling language for mathematical programming. *Management Science*, 36(5):519–554, 1990.

[37] D.M. Gay. Hooking your solver to AMPL. *Numerical Analysis Manuscript 93-10, AT&T Bell Laboratories, ftp://netlib.bell-labs.com/netlib/att/cs/doc/93/4-10.ps.Z*, 1993.

[38] D.M. Gay, M.L. Overton, e M.H. Wright. A primal-dual interior method for nonconvex nonlinear programming. Technical Report 97-4-08, Computing Sciences Research Center, Bell Laboratories, 1997. http://cm.bell-labs.com/cm/cs/doc/97/4-08.ps.gz.

[39] P.E. Gill, W. Murray, D.B. Ponceleón, e M.A. Saunders. Solving reduced KKT systems in barrier methods for linear and quadratic programming. Technical Report SOL 91-7, Systems Optimization Laboratory, Department of Operations Research, Stanford University, 1991.

[40] P.E. Gill, W. Murray, M.A. Saunders, e M.H. Wright. *User's Guide for NPSOL: A Fortran Package for Nonlinear Programming.* Stanford University, 1986.

[41] M. Goberna e M. López (Eds.). *Semi-Infinite Programming: Recent Advances*, volume 57 of *Nonconvex Optimization and its Applications.* Kluwer Academics Publishers, Dordrecht, The Netherlands, 2001.

[42] M.S. Gockenbach e A.J. Kearsley. Optimal signal sets for non-gaussian detectors. *SIAM Journal on Optimization*, 9(2):316–326, 1999.

[43] J. Gondzio. Multiple centrality corrections in a primal-dual method for linear programming. Technical Report 1994.20, University of Geneva, Switzerland, 1995.

[44] C. Gonzaga, E. Polak, e R. Trahan. An improved algorithm for optimization problems with functional inequality constraints. *IEEE Transactions on Automatic Control*, AC-25(1):49–54, Fevereiro 1980.

[45] S. Görner. *Ein Hybridverfahren Zur Lösung Nichtlinearer Semi-Infiniter Optimierungsprobleme.* PhD thesis, Fachbereich Mathematik der Technischen Universität Berlin, 1997.

[46] S. Görner, A. Potchinkov, e R. Reemtsen. The direct solution of nonconvex nonlinear FIR filter design problems by a SIP method. *Optimization and Engineering*, 1:123–154, 2000.

[47] T.J. Graettinger e B.H. Krogh. The acceleration radius: A global performance measure for robotic manipulators. *IEEE Journal of Robotics and Automation*, 4(1):60–69, 1988.

[48] P.R. Gribik. A central-cutting-plane algorithm for semi-infinite programming problems. *Em [25]*, pp. 66–82, 1979.

[49] J. Guddat, H.Th. Jongen F. NožIčKa, J. Still, e F. Twilt. Parametric optimization and related topics IV. In B. Brosowski, F. Deutsch, e J. Guddat, editors, *Approximation & Optimization*, volume 9. Verlag Peter Lang, 1995.

[50] M. Gugat. *Fractional Semi-Infinite Programming.* PhD thesis, Trier University, Germany, 1994.

[51] S.-A. Gustafson. A three-phase algorithm for semi-infinite programs. *Em [31]*, pp. 138–157, 1983.

[52] E. Haaren-Retagne. *A Semi-Infinite Programming Algorithm for Robot Trajectory Planning*. PhD thesis, University of Trier, 1992.

[53] J.K. Hartman. Iterative determination of parameters for an exact penalty function. *Journal of Optimization Theory and Applications*, 16(1/2):49–66, 1975.

[54] R. Hettich. A comparison of some numerical methods for semi-infinite programming. *Em [25]*, pp. 112–125, 1979.

[55] R. Hettich. An implementation of a discretization method for semi-infinite programming. *Mathematical Programming*, 34(3):354–361, 1986.

[56] R. Hettich. Some recent results in parametric semi-infinite programming. *Optimization*, 32:359–372, 1995.

[57] R. Hettich e G. Gramlich. A note on an implementation of a method for quadratic semi-infinite programming. *Mathematical Programming*, 46:249–254, 1990.

[58] R. Hettich e K.O. Kortanek. Semi-infinite programming: Theory, methods, and applications. *SIAM Review*, 35(3):380–429, 1993.

[59] H. Hu. A globally convergent method for semi-infinite linear programming. *Journal of Global Optimization*, 8:189–199, 1996.

[60] F. Jarre. Comparing two interior-point approaches for semi-infinite programs. Technical report, Universität Trier, 1999.

[61] F. Jarre e M. Wechs. Extending Mehrotra's corrector for linear programs. Technical report, Universität Würzburg, Federal Republic of Germany, 1999.

[62] L.S. Jennings, M.E. Fisher, K.L. Teo, e C.J. Goh. *MISER3 Optimal Control Software, Theory and User Manual*. Versão 2.0.

[63] L.S. Jennings e K.L. Teo. A computational algorithm for functional inequality constrained optimization problems. *Automatica*, 26(2):371–375, 1990.

[64] D. C. Jiang, K. L. Teo, e W. Y. Yan. A new computational method for the functional inequality constrained minimax optimization problem. *Computers and Mathematics with Applications*, 33(6):53–63, 1997.

[65] H.TH. Jongen e O. Stein. On generic one-parametric semi-infinite optimization. *SIAM Journal on Optimization*, pp. 1103–1137, 1997.

[66] J. Kaliski, D. Haglin, C. Roos, e T. Terlaky. Logarithmic barrier decomposition methods for semi-infinite programming. *International Transactions in Operational Research*, 4(4):285–303, 1997.

[67] K.O. Kortanek e H. No. A central cutting plane algorithm for convex semi-infinite programming problems. *SIAM Journal on Optimization*, 3(4):901–918, 1993.

[68] C.T. Lawrence. *A Computationally Efficient Feasible Sequencial Quadratic Programming Algorithm*. PhD thesis, Institute for Systems Research, University of Maryland, 1998.

[69] C.T. Lawrence e A.L. Tits. Feasible sequential quadratic programming for finely discretized problems from SIP. *Em [119]*, pp. 159–193, 1998.

[70] T. León, S. Sanmatías, e E. Vercher. A multi-local optimization algorithm. *Top*, 6(1):1–18, 1998.

[71] T. León, S. Sanmatías, e E. Vercher. On the numerical treatment of linearly constrained semi-infinite optimization problems. *European Journal of Operational Research*, 121:78–91, 2000.

[72] T. León e E. Vercher. A purification algorithm for semi-infinite programming. *European Journal of Operational Research*, 57:412–420, 1992.

[73] E. Levitin e R. Tichatschke. A branch-and-bound approach for solving a class of generalized semi-infinite programming problems. *Journal of Global Optimization*, 13:299–315, 1998.

[74] R.M. Lewis, V. Torczon, e M. W. Trosset. Why pattern search works. *OPTIMA The mathematical programming society newsletter*, (59):1–7, 1998.

[75] Y. Li e D. Wang. A semi-infinite programming model for Earliness/Tardiness production planning with simulated annealing. *Mathematical and Computer Modelling*, 26(7):35–42, 1997.

[76] The NAG Library. *Numerical Algorithms Group*. Mayfield House, 256 Banbury Rd., Oxford OX2 7DE, U.K.

[77] C.-J. Lin, S.-C. Fang, e S.-Y. Wu. An unconstrained convex programming approach to linear semi-infinite programming. *SIAM Journal on Optimization*, 8(2):443–456, 1998.

[78] C.-S. Lin, P.-R. Chang, e J. Y. S. Luh. Formulation and optimization of cubic polynomial joint trajectories for industrial robots. *IEEE Transactions on Automatic Control*, AC-28(12):1066–1074, Dezembro 1983.

[79] Y. Liu e K.L. Teo. A dual parameterization algorithm for linear quadratic semi-infinite programming problems. *Nonlinear Analysis*, 47:5635–5646, 2001.

[80] Y. Liu, K.L. Teo, e S. Ito. A dual parametrization approach to linear-quadratic semi-infinite programming problems. *Optimization Methods and Software*, 10:471–495, 1999.

[81] A. De Luca, L. Lanari, e G. Oriolo. A sensitivity approach to optimal spline robot trajectories. *Automatica*, 27(3):535–539, 1991.

[82] S.P. Marin. Optimal parametrization of curves for robot trajectory design. *IEEE Transactions on Automatic Control*, 33(2):209–214, 1988.

[83] MatWorks. *Application Program Interface Guide*. The MatWorks Inc., 1996.

[84] MatWorks. *MATLAB*. The MatWorks Inc., 2001. Versão 6.1, Release 12.1.

[85] MatWorks. *MATLAB Optimization Toolbox*. The MathWorks Inc., 2001. Em [84].

[86] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2(4):575–601, 1992.

[87] C. Mészáros. BPMPD interior point solver. http://www.sztaki.hu/ meszaros/bpmpd/.

[88] C. Mészáros. *The Efficient Implementation of Interior Point Methods for Linear Programming and their Applications*. PhD thesis, Eötvös Loránd University of Sciences, 1996.

[89] C. Mészáros. Fast Cholesky factorization for interior point methods of linear programming. *Computers & Matehmatics with Applications*, 31(4/5):49–51, 1996.

[90] C. Mészáros. Steplenghts in infeasible primal-dual interior point algorithms of convex quadratic programming. Technical Report DOC 97/7, Imperial College, 1997.

[91] C. Mészáros. The BPMPD interior point solver for convex quadratic problems. Technical Report WP 98-8, Laboratory of Operations Research and Decision Systems, Hungarian Academy of Sciences, 1998.

[92] C. Mészáros. On the property of the Cholesky factorization and its consequences in interior point methods. Technical Report WP 98-7, Laboratory of Operations Research and Decision Systems, Hungarian Academy of Sciences, 1998.

[93] M. Mitchell. *An Introdution to Genetic Algorithms*. MIT Press, Cambridge, MA, 1996.

[94] R.E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, 1966.

[95] B.J.F. MURTEIRA. *Probabilidades e Estatística*, volume I e II. McGraw-Hill Portugal.

[96] S.G. Nash e A. Sofer. *Linear and Nonlinear Programming*. McGraw-Hill International Editions, 1996.

[97] J.A. Nelder e R. Mead. A simplex method for function minimization. *Computing Journal*, 7:308–313, 1965.

[98] Y. Nesterov. Interior-point methods: An old and new approach to nonlinear programming. *Mathematical Programming*, 79:285–297, 1997.

[99] J. Nocedal e S.J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, 1999.

[100] Visual Numerics. Library IMSL. http://www.vni.com/products/imsl/.

[101] E.R. Panier e A.L. Tits. A globally convergent algorithm with adaptively refined discretization for semi-infinite optimization problems arising in engineering design. *IEEE Transactions on Automatic Control*, 34(8):903–908, 1989.

[102] R.P. Paul. *Robot Manipulators: Mathematics, Programming, and Control*. MIT Press, 1981.

[103] E. Polak. On the mathematical foundations of nondifferentiable optimization in engineering design. *SIAM Review*, 29(1):21–89, Março 1987.

[104] E. Polak, L. Qi, e D. Sun. First-order algorithms for generalized semi-infinite minmax problems. *Computational Optimization and Applications*, 13:137–161, 1999.

[105] E. Polak e A.L. Tits. A recursive quadratic programming algorithm for semi-infinite optimization problems. *Applied Mathematics and Optimization*, 8:325–349, 1982.

[106] A. Potchinkov e R. Reemtsen. FIR filter design in the complex domain by a semi-infinite programming technique. I. the method. *AEÜ*, 48(3):135–144, 1994.

[107] A. Potchinkov e R. Reemtsen. FIR filter design in the complex domain by a semi-infinite programming technique. II. examples. *AEÜ*, 48(4):200–209, 1994.

[108] A. Potchinkov e R. Reemtsen. The design of FIR filters in the complex plane by convex optimization. *Signal Processing*, 46:127–146, 1995.

[109] A. Potchinkov e R. Reemtsen. The simultaneous approximation of magnitude and phase by FIR digital filters. I: A new approach. *International Journal of Circuit Theory and Applications*, 25:167–177, 1997.

[110] A. Potchinkov. Design of optimal linear phase FIR filters by a semi-infinite programming technique. *Signal Processing*, pp. 165–180, 1997.

[111] C.J. Price. *Non-Linear Semi-Infinite Programming*. PhD thesis, University of Canterbury, New Zealand, Agosto 1992.

[112] C.J. Price e I.D. Coope. An exact penalty function algorithm for semi-infinite programmes. *BIT*, 30:723–734, 1990.

[113] C.J. Price e I.D. Coope. Numerical experiments in semi-infinite programming. *Computational Optimization and Applications*, 6:169–189, 1996.

[114] R.D.C. e I. Adler. Interior path following primal-dual algorithms. part I: Convex quadratic programming. *Mathematical Programming*, 44:43–66, 1989.

[115] R.D.C. e I. Adler. Interior path following primal-dual algorithms. part II: Convex quadratic programming. *Mathematical Programming*, 44:43–66, 1989.

[116] R. Reemtsen. Discretization methods for the solution of semi-infinite programming problems. *Journal of Optimization Theory and Applications*, 71(1):85–103, 1991.

[117] R. Reemtsen. A cutting plane method for solving minimax problems in the complex plane. *Numerical Algorithms*, 2:409–436, 1992.

[118] R. Reemtsen. Some outer approximation methods for semi-infinite optimization problems. *Journal of Computational and Applied Mathematics*, 53:87–108, 1994.

[119] R. Reemtsen e J.-J. Rückmann (Eds.). *Semi-Infinite Programming*. Kluwer Academic Publishers, 1998.

[120] R. Reemtsen e A. Potchinkov. FIR filter design in regard to frequency responce, magnitude, and phase by semi-infinite programming. *In [49]*, pp. 299–314, 1995.

[121] K. Roleff. A stable multiple exchange algorithm for linear SIP. *Em [25]*, pp. 83–96, 1979.

[122] J.-J. Rückmann e O. Stein. On convex lower level problems in generalized semi-infinite optimization. Technical Report 92, Aachen University of Technology, 2000.

[123] D.F. Shanno e R.J. Vanderbei. Interior-point methods for nonconvex nonlinear programming: Orderings and higher-order methods. *Mathematical Programming*, 87:303–316, 2000.

[124] A. Shapiro. Directional differentiability of the optimal value function in convex semi-infinite programming. *Mathematical Programming*, 70:149–157, 1995.

[125] K.G. Shin e N.D. McKay. Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Transaction on Automatic Control*, AC-30(6):531–541, Junho 1985.

[126] K.G. Shin e N.D. McKay. A dynamic programming approach to trajectory planning of robotic manipulators. *IEEE Transactions on Automatic Control*, AC-31(6):491–500, Junho 1986.

[127] K.G. Shin e N.D. McKay. Selection of near-minimum time geometric paths for robotic manipulators. *IEEE Transactions on Automatic Control*, AC-31(6):501–511, Junho 1986.

[128] G. Sporre e A. Forsgren. Relations between divergence of multipliers and convergence to infeasible points in primal-dual interior methods for nonconvex nonlinear programming. Technical Report TRITA-MAT-2002-OS7, Royal Institute of Technology, Abril 2002.

[129] M.D. Srinath, P.K. Rajasekaran, e R. Viswanathan. *Introduction to Statistical Signal Processing with Applications*. Prentice Hall, 1996.

[130] O. Stein. The feasible set in generalized semi-infinite optimization. Technical Report 90, Aachen University of Technology, 1999.

[131] O. Stein. First order optimality conditions for degenerate index sets in generalized semi-infinite optimization. Technical Report 91, Aachen University of Technology, 2000.

[132] O. Stein e G. Still. On optimality conditions for generalized semi-infinite programming problems. *Journal of Optimization Theory and Applications*, 103(2):443–458, 2000.

[133] O. Stein e G. Still. On generalized semi-infinite optimization and bilevel optimization. *European Journal of Operational Research*, Artigo em publicação, 2001.

[134] O. Stein e G. Still. Solving semi-infinite optimization problems with interior point techniques. Technical Report 96, Aachen University of Technology, 2001.

[135] G. Still. Generalized semi-infinite programming: Theory and methods. *European Journal of Operational Research*, 119:301–313, 1999.

[136] G. Still. Discretization in semi-infinite programming: The rate of convergence. *Mathematical Programming*, 91:53–69, 2001.

[137] O. Von Stryk e M. Schlemmer. Optimal control of the industrial robot manutec r3. *Em: R. Bulirsch, D. Kraft (eds.), Computational Optimal Control, International Series of Numerical Mathematics*, 115:367–382, 1994.

[138] Y. Tanaka. A trust region method for semi-infinite programming problems. *International Journal of Systems Science*, 30(2):199–204, 1999.

[139] K.L. Teo e C.J. Goh. A simple computational procedure for optimization problems with functional inequality constraints. *IEEE Transactions on Automatic Control*, AC-32(10):940–941, Outubro 1987.

[140] K.L. Teo, V. Rehbock, e L.S. Jennings. A new computational algorithm for functional inequality constrained optimization problems. *Automatica*, 29(3):789–792, 1993.

[141] A.L. Tits, T.J. Urban, S. Bakhtiari, e C.T. Lawrence. A primal-dual interior-point method for nonlinear programming with strong global and local convergence properties. Technical Report ISR TR 2001-3 R2, Department of Electrical and Computer Engineering and Institute of Systems Research, University of Maryland, 2001.

[142] M.J. Todd. Interior-point algorithms for semi-infinite programming. *Mathematical Programming*, 65:217–245, 1994.

[143] V. Torczon. Pattern search methods for nonlinear optimization. *SIAG/OPT Views and news, a forum for the SIAM activity group on optimization*, (6):7–11, 1995.

[144] V. Torczon. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7(1):1–25, 1997.

[145] R. Trahan e E. Polak. A derivative-free algorithm for a class of infinitely constrained problems. *IEEE Transactions on Automatic Control*, AC-25(1):54–61, 1980.

[146] M. Ulbrich, S. Ulbrich, e L.N. Vicente. A globally convergent primal-dual interior-point filter method for nonconvex nonlinear programming. Technical Report TR00-12, Department of Computational and Applied Mathematics, Rice University, 2000.

[147] R.J. Vanderbei. LOQO: An interior point code for quadratic programming. Technical Report SOR-94-15, Statistics and Operation Research, Princeton University, 1998.

[148] R.J. Vanderbei. LOQO user's manual - version 4.05. Technical Report No. ORFE-99-??, Princeton University, Operations Research and Financial Engineering, 2000.

[149] R.J. Vanderbei e D.F. Shanno. An interior-point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13:231–252, 1999.

[150] A.I.F. Vaz, E.M.G.P. Fernandes, e M.P.S.F. Gomes. Coding and solving semi-infinite programming problems. Em *CD-ROM do X Congresso Latino-Iberoamericano de Investigação Operacional*, 2000.

[151] A.I.F. Vaz, E.M.G.P. Fernandes, e M.P.S.F. Gomes. Discretization methods for semi-infinite programming. *Investigação Operacional*, 21(1):37–46, 2001.

[152] A.I.F. Vaz, E.M.G.P. Fernandes, e M.P.S.F. Gomes. A penalty technique for semi-infinite programming. *Proceedings of V-SGAPEIO*, pp. 235–238, Ferrol, Espanha, 2001.

[153] A.I.F. Vaz, E.M.G.P. Fernandes, e M.P.S.F. Gomes. Robot trajectory planning with semi-infinite programming. *Proceedings of the ORP3 conference (7 páginas)*, 2001. Paris.

[154] A.I.F. Vaz, E.M.G.P. Fernandes, e M.P.S.F. Gomes. A sequential quadratic method with a dual parametrization approach to semi-infinite programming, em publicação, *TOP*, 2003.

[155] A.I.F. Vaz, E.M.G.P. Fernandes, e M.P.S.F. Gomes. NSIPS v2.1: Nonlinear Semi-Infinite Programming Solver. *Technical Report ALG/EF/5-2002*, Dezembro 2002. http://www.norg.uminho.pt/aivaz/.

[156] A.I.F. Vaz, E.M.G.P. Fernandes, e M.P.S.F. Gomes. Optimal signal sets via semi-infinite programming. *Investigação Operacional*, 22(1):87–101, 2002.

[157] A.I.F. Vaz, E.M.G.P. Fernandes, e M.P.S.F. Gomes. SIPAMPL v2.0: Semi-Infinite Programming with AMPL. Technical Report ALG/EF/4-2002, Universidade do Minho, Braga, Portugal, Dezembro 2002. http://www.norg.uminho.pt/aivaz/, em revisão, ACM Transactions On Mathematical Software.

[158] A.I.F. Vaz, E.M.G.P. Fernandes, e M.P.S.F. Gomes. A Quasi-Newton Interior Point Method for Semi-Infinite Programming. *Proceedings of OMS 2002*, página 141, Dezembro 2002.

[159] Y.V. Volkov e S.K. Zavriev. A general stochastic outer approximations method. *SIAM Journal on Control and Optimization*, 35(4):1387–1421, 1997.

[160] Andreas Wachter e Lorenz T. Biegler. Failure of global convergence for a class of interior point methods for nonlinear programming. *Mathematical Programming*, 88:565–574, 2000.

[161] D. Wang e S.-C. Fang. A semi-infinite programming model for Earliness/Tardiness production planning with a genetic algorithm. *Computers and Mathematics with Applications*, 31(8):95–106, 1996.

[162] M.-H. Wang e Y.-E. Kuo. A perturbation method for solving linear semi-infinite programming problems. *Computers and Mathematic with Applications*, 37:181–198, 1999.

[163] A. Wätchter e L.T. Biegler. Global and local convergence of line search filter methods for nonlinear programming. Technical Report CADP B-01-09, Department of Chemical Engineering, Carnegie Mellon University, 2001.

[164] G.A. Watson. Numerical experiments with globally convergent methods for semi-infinite programming problems. *Em [31]*, pp. 193–205, 1983.

[165] G.-W. Weber. *Generalized Semi-Infinite Optimization and Related Topics.* PhD thesis, Darmstadt University of Technology, 1999.

[166] R.L. Wheeden e A. Zygmund. *Measure and Integral, an Introduction to Real Analysis.* Marcel Dekker, Inc., 1977.

[167] M.A. Wolfe. *Numerical Methods for Unconstrained Optimization, an Introduction.* Van Nostrand Reinhold Company, 1978.

[168] S.J. Wright. Modified Cholesky factorization in interior-point algorithms for linear programming. Technical Report ANL/MCS-P600-0596, Mathematics and Computer Science Division, Argonne National Laboratory, 1996.

[169] S.-Y. Wu e S.-C. Fang. Solving convex programs with infinitely many constraints by a relaxed cutting plane method. *Computers and Mathematics with Applications*, 38:23–33, 1999.

[170] H. Yamashita e H. Yabe. Superlinear and quadratic convergence of some primal-dual interior pont methods for constrained optimization. *Mathematical Programming*, 75:377–397, 1996.

[171] P. Zencke e R. Hettich. Directional derivatives for the value-function in semi-infinite programming. *Mathematical Programming*, 38:323–340, 1987.

[172] J.L. Zhou e A.L. Tits. An SQP algorithm for finely discretized continuous minimax problems and other minimax problems with many objective functions. *SIAM Journal on Optimization*, 6(2):461–487, 1996.

# Apêndices

# Apêndice A

# Distribuição em CDROM

---

O CDROM que acompanha esta tese contém os pacotes de software SIPAMPL e NSIPS. O uso do CDROM permite que seja divulgada toda a informação relativa ao trabalho que foi desenvolvido neste projecto, que não seria tão eficaz se se optasse apenas por um suporte em papel.

---

O CDROM contém dois directórios principais a partir do directório raiz. O directório `doc` contém os relatórios técnicos, *SIPAMPL v2.0 : Semi-Infinite Programming with AMPL* [157] e *NSIPS v2.1: Nonlinear Semi-Infinite Programming Solver* [155]. O primeiro serve de manual de utilização da ferramenta SIPAMPL e o segundo é o manual de utilizador do pacote NSIPS. A estrutura do outro directório `solvers` pode ser consultada na Figura A.1.

Os directórios têm o seguinte significado:

- `solvers` refere-se ao directório onde o ficheiro `solvers.tar` foi descompactado. É o directório onde reside a interface do AMPL;
- o directório `sip` contém a interface do SIPAMPL (onde está a biblioteca `libsip.a` (Linux) ou `sipampl.lib` (MSDOS));
- o directório `matlab` contém as rotinas de interface de ligação do MATLAB ao SIPAMPL e o M-ficheiro (*M-File*), que é um exemplo de como usar a função MATLAB `sipampl`;
- o directório `nsips` contém o software NSIPS e a biblioteca dinâmica de "B-Splines";
- o directório `s_solver` contém um exemplo de como usar a interface do SIPAMPL. Neste exemplo são usadas as rotinas da interface e os valores obtidos são impressos no terminal (monitor);
- o directório `sipmod` contém todos os problemas de PSI codificados;

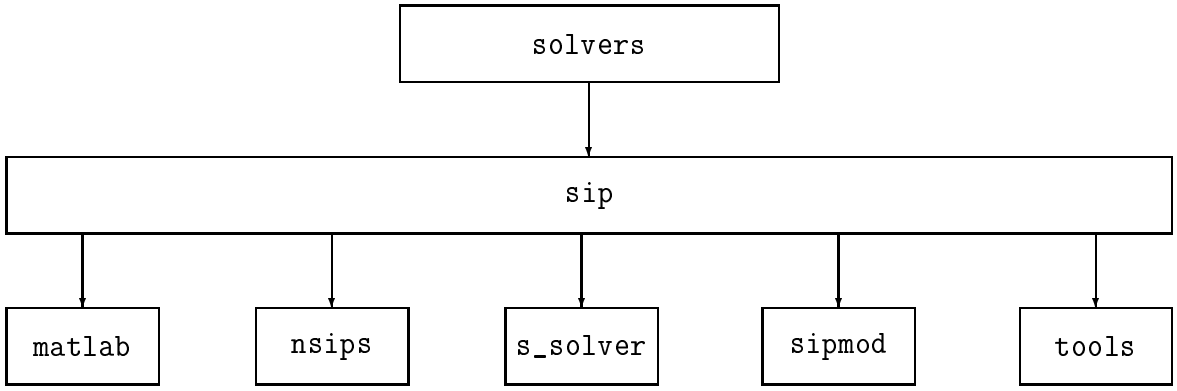


Figura A.1: Organização dos directórios do SIPAMPL

- o directório `tools` contém a ferramenta *select*;

O utilizador pode modificar a estrutura de directórios, desde que modifique os ficheiros de construção de binários (*makefiles*) para reflectir a nova estrutura.

# Apêndice B

# Instalação do software de interface do SIPAMPL

## B.1 Instalação passo a passo da interface do AMPL

Os seguintes passos devem ser tomados para ligar um software de resolução de problemas ao AMPL.

- Obtenha uma cópia do ficheiro `solvers.tar`[1].
- Descompacte e construa a biblioteca `amplsolver.a` (Linux) ou `amplsolv.lib` (MS-DOS). Esta biblioteca fornece as funções de interface do AMPL. O ficheiro de cabeçalho `asl.h` contém as definições de estruturas de dados importantes, sendo a estrutura de dados `asl` a mais usada e que contém os dados do problema.
- Construa o software de resolução e ligue-o com a biblioteca de interface do AMPL.

O AMPL comunica com o software de resolução de problemas através do ficheiro com extensão `.nl` (`stub.nl`). Na Figura B.1 apresenta-se um diagrama de interacção entre o AMPL e o software de resolução de problemas. O AMPL lê a descrição do problema - de um ficheiro ou da entrada standard (*standard input*) - e escreve o ficheiro com a informação acerca do problema (`stub.nl`). O software de resolução de problemas lê o ficheiro `stub.nl` para a memória (usando as rotinas de interface do AMPL). O problema é guardado na memória numa estrutura de dados `asl`. Esta estrutura fornece a maior parte dos dados do problema (aproximação inicial, aproximação inicial para o problema dual, número de funções objectivo, expressões das funções objectivas, número de restrições, expressões das funções das restrições, etc.). Após encontrar uma solução para o problema, o software de resolução de problemas, escreve o ficheiro `stub.sol` com a solução encontrada. O AMPL lê o ficheiro `stub.sol` e pode apresentar a solução do problema, se pedida pelo utilizador.

Alguns dados do problema não estão disponíveis no ficheiro `stub.nl` (por exemplo, os nomes das variáveis e restrições não são fornecidos). Para obter mais informação acerca

[1] http://netlib.bell-labs.com/netlib/ampl/

do problema pode ser necessário fornecer mais ficheiros de dados (por exemplo, os ficheiros `.col` e `.row` que contêm os nomes das variáveis e restrições).

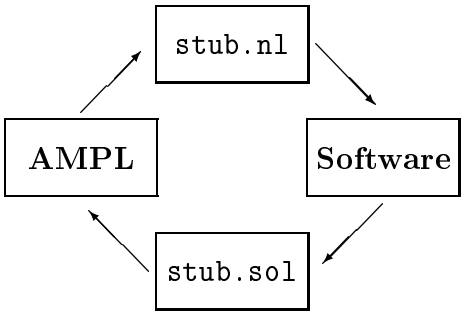


Figura B.1: Interacção entre o AMPL e o software de resolução de problemas.

## B.2 Instalação da interface com o MATLAB

A interface para o MATLAB é fornecida com o SIPAMPL (veja-se na Subsecção 5.1.3). Segue-se uma instalação passo a passo:

- torne o directório MATLAB como directório corrente (dentro do directório do SIPAMPL);
- use a `makefile` (Linux) ou `makefile.vc` (MSDOS) para construir a biblioteca executável MATLAB. A biblioteca fornece a função `sipampl` ao MATLAB.
- coloque a biblioteca executável num directório onde o MATLAB o possa encontrar (caminho de executáveis do MATLAB);
- chame o AMPL com o problema e produza os ficheiros `stub.nl`, `stub.row` e `stub.col`;
- chame o pacote de optimização do MATLAB como descrito na Subsecção 5.2.1.

# Apêndice C

# Utilização e instalação do software NSIPS

O software pode ser obtido num formato binário (sistema operativo MSDOS) e num formato de código fonte (Linguagem C). No sistema operativo Linux a tarefa de produzir o binário é simples, visto que o compilar de C (gcc) faz parte da instalação padrão. No sistema operativo Linux e para as diferentes versões do núcleo do sistema operativo (*kernel*) os binários produzidos são diferentes.

No formato binário, o NSIPS deve ser copiado para uma directoria onde o utilizador tenha permissões de execução e escrita e preferencialmente onde a linha de comandos procure por executáveis (directoria indicada na variável *PATH* de ambiente).

No formato de código fonte, o ficheiro `nsips.tgz` deve ser descompactado (`tar xvzf nsips.tgz` ou usando o software WinZip) num directório ao mesmo nível que a base de dados do SIPAMPL. A Figura A.1 apresenta uma estrutura de directórios típica para o SIPAMPL e NSIPS. O Apêndice A descreve os directórios envolvidos.

O pacote de software NSIPS fornece três ficheiros de construção (*makefile*): `makefile.nps` que compila o software NSIPS com a biblioteca `libopt.a` do software NPSOL (Linux), `makefile.std` que produz a versão do software NSIPS sem o uso do software NPSOL (Linux) e `make_std.vc` que produz a versão do software NSIPS sem o uso do software NPSOL (MSDOS). Para usar os ficheiros de construção pode proceder-se de uma das seguintes formas

- copiar o ficheiro `makexxxx.xxx` para `makefile` e depois usar o utilitário `make`, ou
- usar o utilitário `make` com a opção `-f`, i.e., `make -f makexxxx.xxx`

onde `xxxx` e `xxx` deve ser substituído pela versão pretendida.

O software NSIPS usa as rotinas de interface do SIPAMPL e do AMPL (veja-se a Figura C.1). As bibliotecas `libsip.a` (Linux) ou `sipampl.lib` (MSDOS) (rotinas da interface do SIPAMPL) e `amplsolver.a` (Linux) ou `amplsolv.lib` (MSDOS) (rotinas da interface do AMPL) são ligadas com as rotinas do software NSIPS. Estas bibliotecas são também fornecidas para o sistema operativo MSDOS.

Se o problema já se encontrar no formato intermédio *stub* (após processamento pelo AMPL) apenas é necessário escrever

`nsips` *`stub`*

na linha de comandos.

Se o problema se encontra num ficheiro de modelo (base de dados do SIPAMPL) é necessário escrever

`ampl` *`problema`*`.mod`

onde *`problema`* é o nome do problema da base de dados do SIPAMPL.

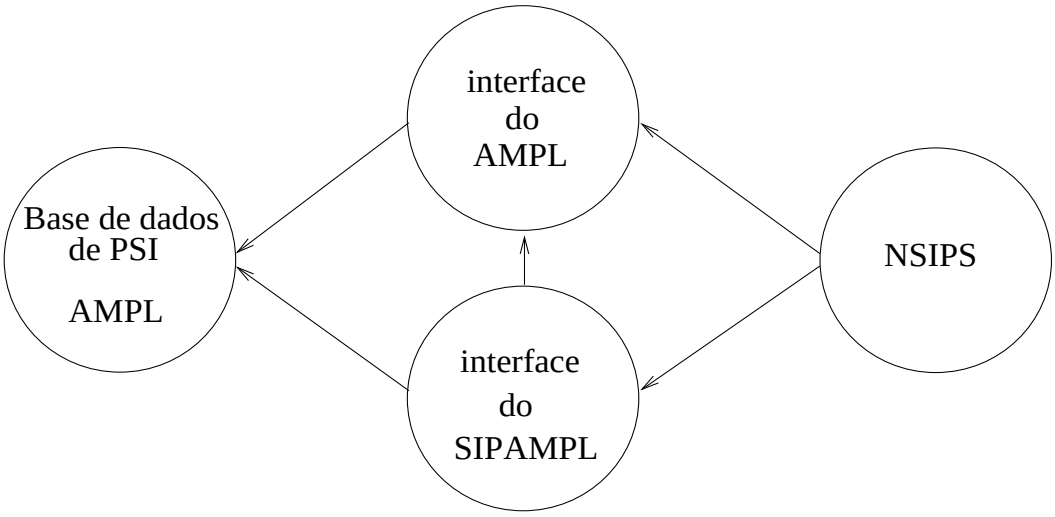


Figura C.1: Interligação entre o NSIPS e as interfaces do SIPAMPL e AMPL

# Apêndice D

# Uma sessão exemplo com a ferramenta *select*

Para demonstrar como um utilizador pode usar a ferramenta *select*, para seleccionar um conjunto de problemas da base de dados do SIPAMPL, apresenta-se de seguida uma sessão interactiva efectuada no sistema operativo Linux. A ferramenta *select* tem um comportamento idêntico para o sistema operativo MSDOS.

`[aivaz@demo tools]$` é a mensagem de espera da linha de comandos do computador. O utilizador começa por definir a variável de ambiente AMPLFUNC para carregar a biblioteca dinâmica `bspline.dll` (necessária para os problemas de robótica existentes na base de dados do SIPAMPL) e invoca a ferramenta *select* através do comando `./select`. O directório da base de dados e o local do binário do AMPL propostos são aceites como correctos. É seleccionado um problema quadrático com apenas uma variável infinita. É imposta a não existência de limites simples nas variáveis finitas. Após encontrados os problemas que satisfazem as condições impostas pelo utilizador é escrito o ficheiro `select.run` que contém uma pilha de comandos para a linha de comandos (*bash shell script*) que corre o AMPL com todos os problemas encontrados. `select.run` terá permissão de execução para o utilizador, grupo e outros (modo -rwxr-xr-x). No sistema operativo MSDOS este ficheiro terá o nome `select.bat` e será uma igualmente uma pilha de comandos, neste caso para o MSDOS.

```
[aivaz@demo tools]$ export AMPLFUNC=../nsips/bspline.dll
[aivaz@demo tools]$ ./select
Select v2.0 tool for SIPAMPL


Default database directory: ../sipmod/
New database directory [CR=Accept default]:
Using database directory: ../sipmod/

Full path for AMPL binary: ../ampl
```

```
New full path for AMPL binary [CR=Accept default]:
Using ../ampl when executing AMPL

Specified Options:
1)  Objective Type      :   All type
2)  Constraints Type    :   All type
3)  0               <= Number of finite variables      <=       +Infinity
4)  0               <= Number of infinite variables    <=       +Infinity
5)  0               <= Number of finite constraints    <=       +Infinity
6)  0               <= Number of infinite constraints  <=       +Infinity
7)  Finite variables:   Everything
8)  Infinite variables: Everything
9)  Initial guess:      :   With or without initial guess

Enter option number to change option [CR=End]:1
Processing option 1
  New objective type
  1) Linear
  2) Quadratic
  3) Polynomial
  4) Generic
  5) All type
Option:2

Specified Options:
1)  Objective Type      :   Quadratic
2)  Constraints Type    :   All type
3)  0               <= Number of finite variables      <=       +Infinity
4)  0               <= Number of infinite variables    <=       +Infinity
5)  0               <= Number of finite constraints    <=       +Infinity
6)  0               <= Number of infinite constraints  <=       +Infinity
7)  Finite variables:   Everything
8)  Infinite variables: Everything
9)  Initial guess:      :   With or without initial guess

Enter option number to change option [CR=End]:4
Processing option 4
New number of infinite variables
Lower bound [CR=keep value]:
Upper bound [CR=keep value, INF=+Infinity]:1

Specified Options:
1)  Objective Type      :   Quadratic
```

```
2)  Constraints Type    :   All type
3)  0               <= Number of finite variables      <=       +Infinity
4)  0               <= Number of infinite variables    <=               1
5)  0               <= Number of finite constraints    <=       +Infinity
6)  0               <= Number of infinite constraints  <=       +Infinity
7)  Finite variables:   Everything
8)  Infinite variables: Everything
9)  Initial guess:      :   With or without initial guess

Enter option number to change option [CR=End]:7
Processing option 7
  Finite variables simple bounds
  1) Both limits finite
  2) Lower limit finite
  3) Upper limit finite
  4) None limit finite
  5) Everything
Option:4

Specified Options:
1)  Objective Type      :   Quadratic
2)  Constraints Type    :   All type
3)  0               <= Number of finite variables      <=       +Infinity
4)  0               <= Number of infinite variables    <=               1
5)  0               <= Number of finite constraints    <=       +Infinity
6)  0               <= Number of infinite constraints  <=       +Infinity
7)  Finite variables:   None limits finite
8)  Infinite variables: Everything
9)  Initial guess:      :   With or without initial guess

Enter option number to change option [CR=End]:


  21 file(s) found with specified options

Do you want me to:
  1) Save results to file select.res
  2) Save results to a bash script select.run
  3) Save results to a M-file sip_run.m
  4) Print results to stdout
  5) Just quit
Option:2
File select.run exists
```

```
Do you want to overwrite it?[Y/N]y


  21 file(s) found with specified options

Do you want me to:
  1) Save results to file select.res
  2) Save results to a bash script select.run
  3) Save results to a M-file sip_run.m
  4) Print results to stdout
  5) Just quit
Option:5
[aivaz@demo tools]$
```

# Apêndice E

# "B-Splines"

## E.1 Fórmulas das "B-Splines"

As fórmulas usadas na biblioteca das "B-Splines" são:

### E.1.1 Funções base das "B-Splines"

$$B_{i,k,\xi}(t) = \frac{t - \xi_i}{\xi_{i+k-1} - \xi_i} B_{i,k-1,\xi}(t) + \frac{\xi_{i+k} - t}{\xi_{i+k} - \xi_{i+1}} B_{i+1,k-1,\xi}(t)$$

### E.1.2 Derivadas das funções base da "B-Spline"

Derivadas das funções base da "B-Spline" com respeito ao parâmetro $t$.

$$\frac{\partial^j B_{i,k,\xi}(t)}{\partial t^j} = (k-1) \left( \frac{-\frac{\partial^{j-1} B_{i+1,k-1,\xi}(t)}{\partial t^{j-1}}}{\xi_{i+k} - \xi_{i+1}} + \frac{\frac{\partial^{j-1} B_{i,k-1,\xi}(t)}{\partial t^{j-1}}}{\xi_{i+k-1} - \xi_i} \right)$$

com

$$\frac{\partial^0 B_{i,k,\xi}(t)}{\partial t^0} = B_{i,k,\xi}(t)$$

$$\frac{\partial^j B_{k,\xi}(t)}{\partial t^j} = \frac{\partial^j \left( \sum_{i=1}^{n} x_i B_{i,k,\xi}(t) \right)}{\partial t^j} = \sum_{i=1}^{n+j} x_i^{(j+1)} B_{i,k-j,\xi}(t)$$

com

$$x_i^{(j+1)} = \begin{cases} x_i & \text{para } \; j = 0 \\ (k-j) \frac{x_i^{(j)} - x_{i-1}^{(j)}}{\xi_{i+k-j} - \xi_i} & \text{para } \; j > 0 \end{cases}$$

### E.1.3 Primeiras derivadas da "B-Spline"

Primeiras derivadas da "B-Spline" com respeito aos coeficientes.

$$\frac{\partial\left(\frac{\partial^j B_{k,\xi}(t)}{\partial t^j}\right)}{\partial x_l} = \frac{\partial\left(\frac{\partial^j \sum_{i=1}^{n} x_i B_{i,k,\xi}(t)}{\partial t^j}\right)}{\partial x_l} = \frac{\partial^j B_{l,k,\xi}(t)}{\partial t^j}$$

### E.1.4 Segundas derivadas da "B-Spline"

As "B-Splines" são combinações lineares das funções base que dependem apenas da variável $t$, logo às segundas derivadas com respeito aos coeficientes é zero.

## E.2 Instalação da biblioteca das "B-Spline"

Se usar o procedimento descrito no Apêndice C para construir o binário do NSIPS então a biblioteca também é construída.

Segue-se a instalação passo a passo no caso de pretender usar a biblioteca de "B-Splines" isoladamente:

- Obtenha o ficheiro `solvers.tar` da internet[1].
- Obtenha o pacote de software do SIPAMPL da internet[2] ou apenas o ficheiro `bspline.c`. Na versão MSDOS são necessários também os ficheiros `bspline.def` e `bspline.lbc`.
- Descompacte o ficheiro `solvers.tar`. Extraia o ficheiro `bspline.c` do pacote de software SIPAMPL.
- Construa a biblioteca de nome `bspline.dll` executando

  *gcc -o bspline.dll -shared bspline.c* (Linux)

  ou

  *link -dll -def:bspline.def -out:bspline.dll @bspline.lbc* (MSDOS).

O ficheiro `bspline.c` inclui um ficheiro de cabeçalho (*header file*) do pacote `solvers.tar` que deve ser fornecido (mude para o directório onde `solvers.tar` foi descompactado ou use a opção (*flag*) *-I* para o compilador.

Para usar a biblioteca, atribua o caminho para a biblioteca à variável de ambiente *AMPLFUNC* (*export AMPLFUNC=bspline.dll* (Linux) ou *set AMPLFUNC=bspline.dll* (MSDOS) na linha de comandos). O AMPL lê a biblioteca `funcadd.dll` por defeito ou aceita a opção (*-i*) com o nome do ficheiro da biblioteca. Uma alternativa é renomear o ficheiro `spline.dll` como `funcadd.dll` ou chamar o AMPL com a opção *-i bspline.dll*. Use as funções da biblioteca das "B-Splines" da forma descrita na Subsecção 5.1.2.

[1] *http://netlib.bell-labs.com/netlib/ampl/*
[2] *http://www.norg.uminho.pt/aivaz/*

# Apêndice F

# Derivadas de $\mathcal{L}_l^*$

As derivadas de $\mathcal{L}_l^*$ são:

$$\begin{aligned}\nabla_{t_p}\mathcal{L}_l^*(v) = &\left(\nabla f(x_k) + \sum_{j=1}^{m} c_j\right)^T H_k^{-1} \\ &\left(\sum_{j=1}^{m}\left(\nabla_{xt}^2 g_j(x_k,t_p)h_{pj} + (w_{pj} - w_{p+1j})\nabla_x g_j(x_k,t_p)\right)\right) \\ &- \sum_{j=1}^{m}\left(\nabla_t g_j(x_k,t_p)h_{pj} + (w_{pj} - w_{p+1j})g_j(x_k,t_p)\right)\end{aligned} \tag{F.0.1a}$$

para $p = 1, \ldots, l$,

$$\begin{aligned}\nabla_{h_{pq}}\mathcal{L}_l^*(v) = &\left(\nabla f(x_k) + \sum_{j=1}^{m} c_j\right)^T H_k^{-1}\left(\nabla_x g_q(x_k,t_p)\right) \\ &- g_q(x_k,t_p)\end{aligned} \tag{F.0.1b}$$

para $p = 1, \ldots, l$ e $q = 1, \ldots, m$,

$$\begin{aligned}\nabla_{w_{pq}}\mathcal{L}_l^*(v) = &\left(\nabla f(x_k) + \sum_{j=1}^{m} c_j\right)^T H_k^{-1}\left(\int_{t_{p-1}}^{t_p} \nabla_x g_q(x_k,t)dt\right) \\ &- \int_{t_{p-1}}^{t_p} g_q(x_k,t)dt\end{aligned} \tag{F.0.1c}$$

para $p = 1, \ldots, l+1$ e $q = 1, \ldots, m$.

# Apêndice G

# Dedução da expressão para $\mathcal{L}^*(v)$

Relembre-se da equação (8.1.8) e da Lagrangeana associada ao problema (8.1.3)

$$\mathcal{L}(d,v) = \frac{1}{2}d^T H_k d + d^T \nabla f(x_k) + \sum_{j=1}^{m} \int_a^b \left(d^T \nabla_x g_j(x_k,t) + g_j(x_k,t)\right) dv_j(t),$$

omitindo as dependências das variáveis, para facilitar a apresentação, temos que

$$\begin{aligned}
\mathcal{L}^*(v) =& -\mathcal{L}(d,v) \\
=& -\frac{1}{2}\left(\nabla f + \sum_{j=1}^{m}\int_a^b \nabla_x g_j dv_j\right)^T H_k^{-1}\left(\nabla f + \sum_{j=1}^{m}\int_a^b \nabla_x g_j dv_j\right) \\
&+ \left(\nabla f + \sum_{j=1}^{m}\int_a^b \nabla_x g_j dv_j\right)^T H_k^{-1}\nabla f \\
&+ \sum_{j=1}^{m}\int_a^b \left(\left(\nabla f + \sum_{j=1}^{m}\int_a^b \nabla_x g_j dv_j\right)^T H_k^{-1}\nabla_x g_j - g_j\right) dv_j \\
=& -\frac{1}{2}\left(\nabla f + \sum_{j=1}^{m}\int_a^b \nabla_x g_j dv_j\right)^T H_k^{-1}\left(\nabla f + \sum_{j=1}^{m}\int_a^b \nabla_x g_j dv_j\right) \\
&+ \left(\nabla f + \sum_{j=1}^{m}\int_a^b \nabla_x g_j dv_j\right)^T H_k^{-1}\left(\nabla f + \sum_{j=1}^{m}\int_a^b \nabla_x g_j dv_j\right) - \sum_{j=1}^{m}\int_a^b g_j dv_j \\
=& \frac{1}{2}\left(\nabla f + \sum_{j=1}^{m}\int_a^b \nabla_x g_j dv_j\right)^T H_k^{-1}\left(\nabla f + \sum_{j=1}^{m}\int_a^b \nabla_x g_j dv_j\right) - \sum_{j=1}^{m}\int_a^b g_j dv_j,
\end{aligned}$$

como pretendido.

# Apêndice H

# Problema exemplo codificado na base de dados do SIPAMPL

O problema tomado como exemplo transcrito para esta secção foi o *elke1* (ficheiro `elke1.mod` da base de dados do SIPAMPL)

O problema codificado é então

```
#############################################################################
# Objective: Linear
# Constraints: Generic
#############################################################################
#  Robotics path planning problem
#   As described by Elke Haaren-Retagne, PhD Thesis,
#                 submitted to Trier University, "A Semi-Infinite Programming
#                                  "Algorithm for Robot Trajectory Planning",
#                 1992
#   Originaly described by Ech-cherif, Ecker, Kupferschmid and Marin
#
#   Example 1A (model 1)
#    Coded by A. Ismael F. Vaz 20/01/2000  aivaz@dps.uminho.pt
#    University of Minho, Portugal
#############################################################################
function bspline;
function dbspline;


param k;            # spline degree
param n;            # number of blend splines
# The number of knots can be changed, do not forget to change the Knots array
param nk;           # number of knots
param Knots{1..nk}; # knots
param c{1..3,1..3};
param epson;
param x0{1..n};     # starting guess
param t0;

var x{i in 1..n} := x0[i];  # x variables (coeficients in the spline)
```

```
var t := t0;                # spline parameter

# describe trajectory
var zeta = t^3*(6*t^2-15*t+10);
var zetad1 = 30*t^4-60*t^3+30*t^2; # first derivative
var zetad2 = 120*t^3-180*t^2+60*t; # second derivative
var zetad3 = 360*t^2-360*t+60;     # third derivative

var teta1 = 1.5*zeta;
var teta1d1 = 1.5*zetad1;
var teta1d2 = 1.5*zetad2;
var teta1d3 = 1.5*zetad3;

var teta2 = -0.5*sin(4.7*zeta);
var teta2d1 = -2.35*cos(4.7*zeta)*zetad1;
var teta2d2 = 11.045*sin(4.7*zeta)*(zetad1)^2-2.35*cos(4.7*zeta)*zetad2;
var teta2d3 = 51.9115*cos(4.7*zeta)*(zetad1)^3+33.135*sin(4.7*zeta)*
    zetad1*zetad2-2.35*cos(4.7*zeta)*zetad3;

var teta3 = -1.3*zeta+2.6;
var teta3d1 = -1.3*zetad1;
var teta3d2 = -1.3*zetad2;
var teta3d3 = -1.3*zetad3;

var g =    bspline(t,    k, n, {i in 1..n} x[i], {i in 1..nk} Knots[i]);
var gd1 = dbspline(t, 1, k, n, {i in 1..n} x[i], {i in 1..nk} Knots[i]);
var gd2 = dbspline(t, 2, k, n, {i in 1..n} x[i], {i in 1..nk} Knots[i]);


minimize obj:
    (1/k)*(sum {i in 1..n} (x[i]*(Knots[i+k]-Knots[i])));

subject to bound:
    0 <= t <= 0.999999;

subject to tcons1:
    g - epson >= 0;

subject to tcons2j1:
    -1 <= teta1d1/(c[1,1]*g) <= 1;

subject to tcons2j2:
    -1 <= teta2d1/(c[2,1]*g) <= 1;

subject to tcons2j3:
    -1 <= teta3d1/(c[3,1]*g) <= 1;



subject to tcons3j1:
    -1 <= (teta1d2-teta1d1*(gd1/g))/(c[1,2]*(g)^2) <= 1;
```

```
subject to tcons3j2:
    -1 <= (teta2d2-teta2d1*(gd1/g))/(c[2,2]*(g)^2) <= 1;

subject to tcons3j3:
    -1 <= (teta3d2-teta3d1*(gd1/g))/(c[3,2]*(g)^2) <= 1;



subject to tcons4j1:
    -1 <= (teta1d3-3*teta1d2*(gd1/g)+teta1d1*((3*(gd1)^2-g*gd2)/(g)^2))/
        (c[1,3]*(g)^3) <= 1;

subject to tcons4j2:
    -1 <= (teta2d3-3*teta2d2*(gd1/g)+teta2d1*((3*(gd1)^2-g*gd2)/(g)^2))/
        (c[2,3]*(g)^3) <= 1;

subject to tcons4j3:
    -1 <= (teta3d3-3*teta3d2*(gd1/g)+teta3d1*((3*(gd1)^2-g*gd2)/(g)^2))/
        (c[3,3]*(g)^3) <= 1;


data;


param k  := 4;
param n  := 9;
param nk := 13;
param epson := 1D-2;
param Knots :=
    1   0
    2   0
    3   0
    4   0
    5   0.1666667
    6   0.3333333
    7   0.5
    8   0.6666667
    9   0.8333333
    10  1
    11  1
    12  1
    13  1;

param x0 :=
    1   1.42031
    2   1.42031
    3   1.42031
    4   1.42031
    5   1.42031
    6   1.42031
```

```
    7   1.42031
    8   1.42031
    9   1.42031;


param t0 := 0;


param c :
    1   2   3 :=
1   2   8   250
2   3   18  650
3   4   50  1000;


# solve problem
# don't forget to write .col and .row files
option nsips_auxfiles rc;
option solver nsips;
solve;

###################
# Solution found #
###################
printf "Solution found\n";
display x;
display obj;

################################################
# solution of problem presented by the authors #
################################################
printf "Original Solution\n";
printf "obj=%lf\n", 1.08351;
```

# Apêndice I

# Resolução do problema exemplo com o NSIPS

Transcreve-se uma sessão interactiva com a ferramenta NSIPS. No caso do sistema operativo Linux, inicia-se a resolução começando por seleccionar o método de discretização na versão modificada de Hettich em que o passo de refinamento é de 0.01 e são feitos três refinamentos. É executado o software AMPL com o problema `elke1.mod`. Para o sistema operativo MSDOS, inicia-se a resolução através da selecção do método de penalidade, uma vez que o NPSOL não está disponível na versão MSDOS, com a função de penalidade $\phi_S^1$. É executado o software AMPL com o problema `elke1std.mod`, uma vez que o método de penalidade impõem que os problemas apenas possuam desigualdades do tipo $\leq$.

O uso da biblioteca dinâmica de "B-Splines" é obrigatório, uma vez que se tratam de problemas de robótica.

## I.1 Sistema operativo Linux

```
[aivaz@laptop nsips]$ nsips_options='method=disc_hett disc_h=0.01 disc_k=3'
[aivaz@laptop nsips]$ export nsips_options
[aivaz@laptop nsips]$ AMPLFUNC=./bspline.dll
[aivaz@laptop nsips]$ export AMPLFUNC
[aivaz@laptop nsips]$ ampl ../sipmod/elke1.mod
NSIPS: Discretization method selected Hettich version
Nx=9     Nt=1
h=(0.010000)
Refinements=3
Iter    Grid    Nnlsp   Nacvi   FullGrid
    Nonlinear used 10 iterations Obj=1.081258 Inform=1
0       100     1000    9       --
    Nonlinear used 1 iterations Obj=1.081258 Inform=0
1       200     90      9       Yes
    Nonlinear used 7 iterations Obj=1.082432 Inform=0
```

```
1       200     130     12      No
    Nonlinear used 5 iterations Obj=1.082790 Inform=0
1       200     210     13      No
    Nonlinear used 5 iterations Obj=1.082869 Inform=0
1       200     140     15      No
    Nonlinear used 3 iterations Obj=1.082891 Inform=0
1       200     170     16      No
    Nonlinear used 1 iterations Obj=1.082891 Inform=0
2       600     160     16      Yes
Some statistics
nsip_objval=68
nsip_objgrd=61
nsip_conval=0
nsip_jacval=0
nsip_pfval=0
nsip_gpfval=0
nsip_conxval=0
nsip_contval=57550
nsip_conxgrd=0
nsip_contgrd=27650
nnlpc/nnlp=150.000000
nnlp=7
End

Solution found
x [*] :=
1  0.711379
2  0.712735
3  1.05252
4  0.991565
5  1.89865
6  0.958974
7  1.0593
8  0.709541
9  0.701236
;

obj = 1.08289

Original Solution
obj=1.083510
```

O resultado escrito para o ficheiro de resultados é

```
[aivaz@laptop nsips]$ cat ./results
 & 100 & 600 & 150.0 & 7 & 1.082891\\
```

## I.2 Sistema operativo MSDOS

```
C:\AMPL\SOLVERS\sip\nsips>type nsipsopt.ini
# Option for nsips solver
# In MSDOS type "set OPTIONS_IN=.\nsipsopt.ini"

# Edit to change options to nsips
option nsips_options 'method=penalty pf_type=p1';


C:\AMPL\SOLVERS\sip\nsips>set OPTIONS_IN=.\nsipsopt.ini

C:\AMPL\SOLVERS\sip\nsips>set AMPLFUNC=.\bspline.dll

NSIPS: Penalty method selected
Penalty function 1
Nx=9     Nt=1
STARTING INNER ITERATIONS ********** miu=1.000000, epsilon=0.000100
Gradient:
0.0416666750 0.0833333250 0.1249599995 0.1583009699 0.1503697296
0.1648772074 0.1250000000 0.0833333250 0.0416666750
X: 1.4203100000 1.4203100000 1.4203100000 1.4203100000 1.4203100000
1.4203100000 1.4203100000 1.4203100000 1.4203100000
Penalty 1.420374
Gradient: 0.0416666750 0.0833333250 0.1249599995 0.1583009699 0.1503697296
0.1648772074 0.1250000000 0.0833333250 0.0416666750
Direction: -0.0416666750 -0.0833333250 -0.1249599995 -0.1583009699
-0.1503697296 -0.1648772074 -0.1250000000 -0.0833333250 -0.0416666750
phi0=1.420374 pt=0.000064
alpha=1.000000 phiomega=1.334928 pt=0.042149
Step=1.000000
X: 1.3786433250 1.3369766750 1.2953500005 1.2620090301 1.2699402704
1.2554327926 1.2953100000 1.3369766750 1.3786433250


...
Texto removido.
...

INNER STOPPING IN ITERATION ********* 1
STOPPING IN ITERATION ********* 4
Solution found
0.7186663733 0.7342663858 1.0710302461 1.0195621603 1.8378928632
1.0060370500 1.0318832037 0.7370110004 0.7155862879
Penalty function value
```

```
1.089146
Gradient of penalty:
0.0416666750 0.0833333250 0.1250000000 0.1666666750 0.1666666500
0.1666666750 0.1250000000 0.0833333250 0.0416666750
Gradient of objective:
0.0416666750 0.0833333250 0.1250000000 0.1666666750 0.1666666500
0.1666666750 0.1250000000 0.0833333250 0.0416666750
Some statistics
nsip_objval=225
nsip_objgrd=39
nsip_conval=0
nsip_jacval=0
nsip_pfval=224
nsip_gpfval=38
nsip_conxval=0
nsip_contval=284856
nsip_conxgrd=0
nsip_contgrd=3913
nnlpc/nnlp=0.000000
nnlp=0
End

Solution found
x [*] :=
1  0.718666
2  0.734266
3  1.07103
4  1.01956
5  1.83789
6  1.00604
7  1.03188
8  0.737011
9  0.715586
;

obj = 1.08915

Original Solution
obj=1.083510
```

O resultado escrito para o ficheiro de resultados é

```
C:\AMPL\SOLVERS\sip\nsips>type .\results
 & 34 & 4 & 224 & 38 & 1.089146e+00 & 1.0e+02 & 1.0e-05 \\
```

# Apêndice J

# Tabelas com resultados numéricos

Neste apêndice são apresentados os resultados numéricos para os diferentes métodos implementados. A possibilidade de passar opções aos métodos aliada à possibilidade de editar os ficheiros dos problemas e modificar alguns deles faz com que seja impossível apresentar todos os resultados numéricos possíveis. Os resultados apresentados são pois limitados aos métodos implementados, mas apenas com uma ou outra opção modificada. Em quase todos eles usaram-se as opções por omissão.

Os resultados numéricos são apresentados para todos os problemas da base de dados do SIPAMPL a que os respectivos métodos se aplicam. Foi usada a ferramenta *select* para seleccionar esses problemas. No método de discretização incluem-se todos os problemas em que o conjunto $T$ é formado por um produto cartesiano de intervalos de limites finitos. Retiraram-se os problemas `elke1std` até `elke7std`, uma vez que estes problemas são equivalentes aos `elke1` até `elke7` (veja-se na Subsecção 5.1.3). Nos restantes métodos foram seleccionados os problemas que não possuem restrições finitas de qualquer tipo e que tenham apenas uma variável infinita ($p = 1$). Usaram-se os problemas `elke1std` até `elke7std`, uma vez que estes estão na forma padrão necessária por estes métodos. A excepção é o problema `blankenship1` que possui limites simples nas variáveis finitas, mas uma vez que estas não estão activas na solução, optou-se pela sua inclusão.

## J.1 Método de discretização

Nos resultados numéricos obtidos, nas Tabelas J.1 e J.2 são feitos dois refinamentos em que $h^1 = h^0/2$, $h^2 = h^1/3$. Na versão modificada de Reemtsen foram usados $\widetilde{\epsilon}_0 = 0.01$ e $\widetilde{\delta}_k = 0$, para todo o $k$. $\widetilde{\epsilon}_k$ é actualizado segundo a seguinte fórmula

$$\begin{cases} \widetilde{\epsilon}_1 = \widetilde{\epsilon}_0/2^p & \text{para } k = 0 \\ \widetilde{\epsilon}_{k+1} = \widetilde{\epsilon}_k/3^p & \text{para } k \neq 0 \, . \end{cases}$$

A notação usada nas Tabelas J.1 e J.2 é a seguinte. "Problema" é o nome do problema na base de dados do SIPAMPL, "GI" é o número de pontos usados na grelha inicial, "GF" é o número de pontos na grelha final, "MRH"/"MRR" é a média de restrições usadas

nos problemas finitos resolvidos na versão Hettich/Reemtsen (o número de restrições no primeiro PNL resolvido não é contabilizado), "NH"/"NR" é o número de subproblemas finitos resolvidos na versão Hettich/Reemtsen e "fxH"/"fxR" é o valor da função objectivo na solução encontrada para a grelha final na versão Hettich/Reemtsen.

| Problema | GI | GF | MRH | NH | fxH |
|---|---|---|---|---|---|
| andreson1 | 441 | 14641 | 3.7 | 4 | -3.333333e-01 |
| blankenship1 | 21 | 121 | 31.0 | 3 | -5.421011e-20 |
| coopeL | 32 | 189 | 1.0 | 3 | 3.428960e-01 |
| coopeM | 32 | 189 | 2.0 | 3 | 1.000000e+00 |
| coopeN | 21 | 121 | 1.0 | 3 | -2.524491e-21 |
| elke1 | 100 | 600 | 150.0 | 7 | 1.082891e+00 |
| elke10 | 100 | 600 | 77.0 | 4 | 2.187568e+00 |
| elke2 | | | | | |
| elke3 | 100 | 600 | 116.7 | 4 | 1.523819e+00 |
| elke4 | 100 | 600 | 50.0 | 5 | 2.503993e+00 |
| elke5 | 100 | 600 | 130.0 | 4 | 2.395812e+00 |
| elke6 | 100 | 600 | 122.5 | 5 | 4.487116e+00 |
| elke7 | 100 | 600 | 130.0 | 5 | 2.908544e+00 |
| elke8 | 100 | 600 | 81.7 | 4 | 2.152577e+00 |
| elke9 | 100 | 600 | 84.0 | 4 | 1.634611e+00 |
| fang1 | 11 | 61 | 1.7 | 4 | 4.794255e-01 |
| fang2 | 11 | 61 | 14.3 | 4 | 6.931472e-01 |
| fang3 | 11 | 61 | 16.0 | 3 | 1.718282e+00 |
| ferris1 | 11 | 61 | 19.3 | 4 | 4.723011e-04 |
| ferris2 | 11 | 61 | 9.0 | 4 | -1.786933e+00 |
| gockenbach1 | 101 | 601 | 1058.7 | 4 | -4.694300e-01 |
| gockenbach10 | 101 | 601 | 2802.7 | 4 | -2.794077e-01 |
| gockenbach2 | 101 | 601 | 1088.0 | 5 | -6.070190e-01 |
| gockenbach3 | 101 | 601 | 1005.3 | 4 | -5.179585e-01 |
| gockenbach4 | 101 | 601 | 1104.0 | 5 | -7.807219e-01 |
| gockenbach5 | 101 | 601 | 269.3 | 4 | -1.993260e-01 |
| gockenbach6 | 101 | 601 | 2856.0 | 5 | -6.936058e-01 |
| gockenbach7 | 101 | 601 | 2376.0 | 4 | -8.053168e-01 |
| gockenbach8 | 101 | 601 | 2856.0 | 5 | -7.354465e-01 |
| gockenbach9 | 101 | 601 | 2802.7 | 4 | -1.106585e+00 |
| goerner1 | 11 | 61 | 8.0 | 3 | 4.285327e-03 |
| goerner2 | 8 | 46 | 10.7 | 4 | 2.639219e-03 |
| goerner3 | 11 | 61 | 20.5 | 5 | 4.290934e-05 |
| goerner4 | 121 | 3721 | 12.0 | 3 | 5.327814e-02 |
| goerner5 | 121 | 3721 | 20.0 | 4 | 2.714819e-02 |
| goerner6 | 121 | 3721 | 54.0 | 6 | 1.071870e-03 |

| Problema | GI | GF | MRH | NH | fxH |
|---|---|---|---|---|---|
| goerner7 | 121 | 3721 | 29.5 | 5 | 3.050459e-02 |
| gugat1 | 11 | 61 | 44.0 | 5 | 1.228811e-03 |
| gugat2 | 11 | 61 | 36.0 | 4 | 8.566219e-02 |
| gugat3 | 196 | 1171 | 4.0 | 3 | 6.275449e-01 |
| gugat4a | 21 | 121 | 124.0 | 3 | 2.062885e-07 |
| gugat4b | 61 | 361 | 42.7 | 4 | 6.788929e-03 |
| gugat4c | 11 | 61 | 44.0 | 6 | 3.245986e-03 |
| gugat4d | 11 | 61 | 46.0 | 5 | 7.046525e-02 |
| gugat4e | 21 | 121 | 44.0 | 5 | 1.690969e-02 |
| gugat4f | 31 | 181 | 184.0 | 3 | 2.672432e-07 |
| gugat5a | 21 | 121 | 34.7 | 4 | 8.619152e-05 |
| gugat5b | 61 | 361 | 33.3 | 4 | 3.079525e-01 |
| gugat5c | 11 | 61 | 30.0 | 5 | 8.203911e-03 |
| gugat5d | 11 | 61 | 40.0 | 5 | 2.692308e-01 |
| gugat5e | 21 | 121 | 24.0 | 3 | 5.423527e-02 |
| gugat5f | 31 | 181 | 29.3 | 4 | 1.527719e-04 |
| gugat6 | 11 | 61 | 24.0 | 3 | 2.330344e-05 |
| gugat7 | 11 | 61 | 21.3 | 4 | 1.094575e-03 |
| hettich1 | 121 | 3721 | 29.5 | 5 | 3.050459e-02 |
| hettich10 | 21 | 121 | 2.0 | 3 | 1.000000e+00 |
| hettich10c | 21 | 121 | 5.3 | 4 | 9.996875e-01 |
| hettich2 | 21 | 121 | 7.3 | 4 | 5.380857e-01 |
| hettich3 | 8 | 46 | 10.7 | 4 | 3.528515e-03 |
| hettich4 | 8 | 46 | 3.0 | 3 | 1.000000e+00 |
| hettich5 | 225 | 7225 | 8.0 | 3 | 5.356661e-01 |
| hettich6 | 176 | 5551 | 19.3 | 4 | 2.800728e-02 |
| hettich7 | 176 | 5551 | 18.0 | 4 | 1.775698e-01 |
| hettich8 | 11 | 61 | 8.0 | 3 | 5.640093e-03 |
| hettich9 | 176 | 5551 | 30.0 | 4 | 3.468477e-03 |
| kortanek1 | 80 | 480 | 1.0 | 3 | 3.220681e+00 |
| kortanek2 | 121 | 3721 | 16.0 | 3 | 6.862915e-01 |
| kortanek3 | 8 | 48 | 6.0 | 4 | 1.637604e-04 |
| kortanek4 | 63 | 377 | 94.5 | 3 | 2.712674e-05 |
| leon1 | 11 | 61 | 10.7 | 4 | 4.499769e-03 |
| leon10 | 21 | 121 | 7.3 | 4 | 5.380857e-01 |
| leon11 | 11 | 61 | 7.3 | 4 | 4.840809e-02 |
| leon12 | 63 | 377 | 2.7 | 4 | -1.000233e+00 |
| leon13 | 11 | 61 | 2.7 | 4 | 2.355556e-01 |
| leon14 | 11 | 61 | 2.7 | 4 | 6.650000e-01 |
| leon15 | 11 | 61 | 2.7 | 4 | -6.666667e-01 |
| leon16 | 11 | 61 | 3.7 | 4 | 1.726320e+00 |
| leon17 | 11 | 61 | 1.0 | 3 | -2.000000e+00 |

| Problema | GI | GF | MRH | NH | fxH |
|---|---|---|---|---|---|
| leon18 | 11 | 61 | 2.7 | 4 | -1.750000e+00 |
| leon19 | 11 | 61 | 6.3 | 4 | 7.858326e-01 |
| leon2 | 21 | 121 | 16.7 | 4 | 4.165709e-05 |
| leon3 | 11 | 61 | 13.3 | 4 | 5.081780e-04 |
| leon4 | 21 | 121 | 21.3 | 4 | 2.578180e-03 |
| leon5 | 41 | 241 | 20.7 | 4 | 1.420474e-02 |
| leon6 | 11 | 61 | 12.7 | 4 | 1.540736e-04 |
| leon7 | 11 | 61 | 12.7 | 4 | 2.088885e-03 |
| leon8 | 41 | 241 | 19.3 | 4 | 5.417359e-02 |
| leon9 | 61 | 361 | 16.7 | 4 | 1.633353e-01 |
| li1 | 2001 | 12001 | 135.1 | 15 | 1.959474e+05 |
| li2 | 2001 | 12001 | 75.7 | 10 | 2.909000e+04 |
| lin1 | 121 | 3721 | 23.8 | 6 | -1.823544e+00 |
| liu1 | 21 | 121 | 1.0 | 3 | -4.653846e+00 |
| liu2 | 16 | 95 | 2.7 | 4 | -3.363570e+00 |
| liu3 | 51 | 301 | 9.3 | 4 | 1.540940e+02 |
| matlab1 | 991 | 5941 | 10.0 | 4 | 9.445149e-02 |
| matlab2 | 100 | 3600 | 2.0 | 4 | 1.905816e-04 |
| polak1 | 441 | 14641 | 255.3 | 4 | 5.443259e+00 |
| polak2 | 441 | 14641 | 255.3 | 4 | 6.197698e+00 |
| potchinkov1 | 182 | 5738 | 2010.7 | 4 | -1.399417e-15 |
| potchinkov2 | | | | | |
| potchinkov3 | 168 | 5472 | 1888.0 | 4 | 3.381589e-17 |
| potchinkov4a | 24 | 144 | 91.0 | 203 | -2.301623e+00 |
| potchinkov4b | 24 | 144 | 20.0 | 3 | 2.966089e+01 |
| potchinkovPL | 168 | 5472 | 1888.0 | 4 | -1.000000e-04 |
| potchinkovPLR | 168 | 5472 | 1888.0 | 4 | -1.000000e+00 |
| powell1 | 11 | 61 | 2.7 | 4 | -1.000000e+00 |
| priceK | 32 | 189 | 2.7 | 4 | -3.000607e+00 |
| priceS3 | 1331 | 226981 | 6.3 | 4 | -3.674507e+00 |
| priceS4 | 256 | 194481 | 3.3 | 4 | -4.148553e+00 |
| priceS5 | 243 | 371293 | 64.3 | 4 | -4.469974e+00 |
| priceS6 | 64 | 117649 | 3.7 | 4 | -5.297322e+00 |
| priceT | 2197 | 389017 | 9.2 | 5 | 4.373904e+00 |
| priceU | 64 | 117649 | 4.3 | 4 | -3.449949e+00 |
| reemtsen1 | 1331 | 226981 | 30.7 | 4 | 1.523560e-01 |
| reemtsen2 | 441 | 14641 | 11.3 | 4 | 5.833333e-02 |
| reemtsen3 | 441 | 14641 | 19.3 | 4 | 7.346798e-01 |
| reemtsen4 | 441 | 14641 | 18.0 | 4 | 2.124692e-04 |
| reemtsen5 | 1331 | 226981 | 22.5 | 5 | 8.888537e-02 |
| still1 | 21 | 121 | 3.0 | 3 | 1.000000e+00 |
| tanaka1 | 16 | 95 | 2.0 | 3 | -1.000000e+00 |

| Problema | GI | GF | MRH | NH | fxH |
|---|---|---|---|---|---|
| teo1 | 300 | 1800 | 1.0 | 3 | 1.250721e-01 |
| teo2 | 300 | 1800 | 2.7 | 4 | 1.746271e-01 |
| watson1 | 21 | 121 | 1.0 | 3 | -2.500022e-01 |
| watson10 | 2601 | 90601 | 2.0 | 4 | 2.751856e-01 |
| watson11 | 2601 | 90601 | 6.7 | 4 | -4.386407e+00 |
| watson12 | 2601 | 90601 | 2.7 | 4 | 1.947264e+00 |
| watson13 | 2601 | 90601 | 4.8 | 5 | 1.947264e+00 |
| watson14 | 11 | 61 | 1.0 | 3 | 2.200000e+00 |
| watson2 | 11 | 61 | 16.0 | 3 | 2.618034e+00 |
| watson3 | 11 | 61 | 1.0 | 3 | 5.334687e+00 |
| watson4a | 11 | 61 | 3.7 | 4 | 6.487713e-01 |
| watson4b | 11 | 61 | 7.3 | 4 | 6.160664e-01 |
| watson4c | 11 | 61 | 10.0 | 4 | 6.156512e-01 |
| watson5 | 11 | 61 | 2.0 | 3 | 4.301106e+00 |
| watson6 | 11 | 61 | 1.0 | 3 | 9.715885e+01 |
| watson7 | 121 | 3721 | 1.0 | 3 | 1.000000e+00 |
| watson8 | 121 | 3721 | 8.3 | 4 | 2.435643e+00 |
| watson9 | 441 | 14641 | 61.0 | 3 | -1.200000e+01 |
| zhou1 | 11 | 61 | 2.7 | 4 | 1.780822e-01 |

Tabela J.1: Resultados numéricos para o método de discretização, versão Hettich

| Problema | GI | GF | MRR | NR | fxR |
|---|---|---|---|---|---|
| andreson1 | 441 | 14641 | 8.5 | 3 | -3.333333e-01 |
| blankenship1 | 21 | 121 | 0 | 1 | -5.421011e-20 |
| coopeL | 32 | 189 | 6.0 | 2 | 3.431422e-01 |
| coopeM | 32 | 189 | 0 | 1 | 1.000000e+00 |
| coopeN | 21 | 121 | 0 | 1 | -8.233816e-16 |
| elke1 | 100 | 600 | 545.0 | 3 | 1.083368e+00 |
| elke10 | 100 | 600 | 595.0 | 3 | 2.187581e+00 |
| elke2 | | | | | |
| elke3 | 100 | 600 | 540.0 | 3 | 1.524447e+00 |
| elke4 | 100 | 600 | 606.0 | 3 | 2.504255e+00 |
| elke5 | 100 | 600 | 675.0 | 3 | 2.396374e+00 |
| elke6 | 100 | 600 | 755.0 | 3 | 4.517460e+00 |
| elke7 | 100 | 600 | 455.0 | 3 | 2.910800e+00 |
| elke8 | 100 | 600 | 696.5 | 3 | 2.152595e+00 |
| elke9 | 100 | 600 | 252.0 | 3 | 1.634739e+00 |
| fang1 | 11 | 61 | 8.5 | 3 | 4.794255e-01 |

| Problema | GI | GF | MRR | NR | fxR |
|---|---|---|---|---|---|
| fang2 | 11 | 61 | 21.0 | 2 | 6.931472e-01 |
| fang3 | 11 | 61 | 0 | 1 | 1.718282e+00 |
| ferris1 | 11 | 61 | 22.0 | 3 | 4.882764e-04 |
| ferris2 | 11 | 61 | 41.0 | 3 | -1.786904e+00 |
| gockenbach1 | 101 | 601 | 344.0 | 3 | -4.693247e-01 |
| gockenbach10 | 101 | 601 | 4722.7 | 4 | -2.985144e-01 |
| gockenbach2 | 101 | 601 | 888.0 | 3 | -6.045200e-01 |
| gockenbach3 | 101 | 601 | 856.0 | 3 | -5.187170e-01 |
| gockenbach4 | 101 | 601 | 696.0 | 5 | -8.236573e-01 |
| gockenbach5 | 101 | 601 | 717.3 | 4 | -2.123597e-01 |
| gockenbach6 | 101 | 601 | 3229.3 | 7 | -7.063051e-01 |
| gockenbach7 | 101 | 601 | 4509.3 | 4 | -8.398691e-01 |
| gockenbach8 | 101 | 601 | 3816.0 | 3 | -7.233277e-01 |
| gockenbach9 | 101 | 601 | 3328.0 | 3 | -1.050211e+00 |
| goerner1 | 11 | 61 | 14.0 | 2 | 4.293612e-03 |
| goerner2 | 8 | 46 | 12.0 | 3 | 2.647564e-03 |
| goerner3 | 11 | 61 | 20.0 | 4 | 4.431199e-05 |
| goerner4 | 121 | 3721 | 16.0 | 2 | 5.332291e-02 |
| goerner5 | 121 | 3721 | 30.0 | 3 | 2.726857e-02 |
| goerner6 | 121 | 3721 | 61.6 | 6 | 1.076432e-03 |
| goerner7 | 121 | 3721 | 128.0 | 6 | 3.062215e-02 |
| gugat1 | 11 | 61 | 62.0 | 7 | 2.707722e-03 |
| gugat2 | 11 | 61 | 105.3 | 4 | 8.070486e-02 |
| gugat3 | 196 | 1171 | 0 | 1 | 8.734363e-01 |
| gugat4a | 21 | 121 | 0 | 1 | 2.035827e-07 |
| gugat4b | 61 | 361 | 76.0 | 3 | 6.797444e-03 |
| gugat4c | 11 | 61 | 42.0 | 5 | 4.805271e-03 |
| gugat4d | 11 | 61 | 58.0 | 5 | 1.769459e-01 |
| gugat4e | 21 | 121 | 62.0 | 5 | 2.496647e-02 |
| gugat4f | 31 | 181 | 0 | 1 | 2.589946e-07 |
| gugat5a | 21 | 121 | 48.0 | 3 | 8.686208e-05 |
| gugat5b | 61 | 361 | 66.0 | 3 | 3.079959e-01 |
| gugat5c | 11 | 61 | 32.0 | 4 | 8.439353e-03 |
| gugat5d | 11 | 61 | 46.0 | 5 | 4.761905e-01 |
| gugat5e | 21 | 121 | 52.0 | 2 | 5.427301e-02 |
| gugat5f | 31 | 181 | 32.0 | 2 | 1.527719e-04 |
| gugat6 | 11 | 61 | 40.0 | 2 | 2.385315e-05 |
| gugat7 | 11 | 61 | 26.0 | 3 | 1.100686e-03 |
| hettich1 | 121 | 3721 | 22.0 | 2 | 1.668835e-01 |
| hettich10 | 21 | 121 | 0 | 1 | 1.000000e+00 |
| hettich10c | 21 | 121 | 13.0 | 3 | 9.999653e-01 |
| hettich2 | 21 | 121 | 13.0 | 3 | 5.382079e-01 |

| Problema | GI | GF | MRR | NR | fxR |
|---|---|---|---|---|---|
| hettich3 | 8 | 46 | 13.0 | 3 | 3.545568e-03 |
| hettich4 | 8 | 46 | 0 | 1 | 1.000000e+00 |
| hettich5 | 225 | 7225 | 284.0 | 2 | 5.356722e-01 |
| hettich6 | 176 | 5551 | 49.0 | 3 | 2.806174e-02 |
| hettich7 | 176 | 5551 | 61.0 | 3 | 1.776467e-01 |
| hettich8 | 11 | 61 | 18.0 | 2 | 5.643814e-03 |
| hettich9 | 176 | 5551 | 45.0 | 3 | 3.474291e-03 |
| kortanek1 | 80 | 480 | 10.0 | 2 | 3.221147e+00 |
| kortanek2 | 121 | 3721 | 0 | 1 | 6.862915e-01 |
| kortanek3 | 8 | 48 | 7.0 | 4 | 2.129324e-04 |
| kortanek4 | 63 | 377 | 0 | 1 | 2.712674e-05 |
| leon1 | 11 | 61 | 12.0 | 3 | 4.501227e-03 |
| leon10 | 21 | 121 | 13.0 | 3 | 5.382079e-01 |
| leon11 | 11 | 61 | 10.0 | 2 | 4.840809e-02 |
| leon12 | 63 | 377 | 8.5 | 3 | -1.000026e+00 |
| leon13 | 11 | 61 | 3.5 | 3 | 2.360494e-01 |
| leon14 | 11 | 61 | 4.5 | 3 | 6.666667e-01 |
| leon15 | 11 | 61 | 9.0 | 3 | -6.666667e-01 |
| leon16 | 11 | 61 | 19.0 | 2 | 1.726320e+00 |
| leon17 | 11 | 61 | 0 | 1 | -2.000000e+00 |
| leon18 | 11 | 61 | 4.0 | 3 | -1.750000e+00 |
| leon19 | 11 | 61 | 41.0 | 3 | 7.858362e-01 |
| leon2 | 21 | 121 | 18.0 | 2 | 4.165709e-05 |
| leon3 | 11 | 61 | 18.0 | 3 | 5.211889e-04 |
| leon4 | 21 | 121 | 26.0 | 3 | 2.601229e-03 |
| leon5 | 41 | 241 | 32.0 | 3 | 1.425230e-02 |
| leon6 | 11 | 61 | 16.0 | 3 | 1.552315e-04 |
| leon7 | 11 | 61 | 15.0 | 3 | 2.096595e-03 |
| leon8 | 41 | 241 | 28.0 | 3 | 5.422114e-02 |
| leon9 | 61 | 361 | 34.0 | 3 | 1.633801e-01 |
| li1 | 2001 | 12001 | 6667.7 | 4 | 2.089091e+05 |
| li2 | 2001 | 12001 | 4342.0 | 3 | 4.202574e+04 |
| lin1 | 121 | 3721 | 29.0 | 3 | -1.822954e+00 |
| liu1 | 21 | 121 | 0 | 1 | -4.653846e+00 |
| liu2 | 16 | 95 | 14.5 | 3 | -3.363568e+00 |
| liu3 | 51 | 301 | 131.0 | 3 | 1.541135e+02 |
| matlab1 | 991 | 5941 | 26.0 | 3 | 9.445487e-02 |
| matlab2 | 100 | 3600 | 3.0 | 2 | 1.905816e-04 |
| polak1 | 441 | 14641 | 1762.0 | 3 | 5.443723e+00 |
| polak2 | 441 | 14641 | 1762.0 | 3 | 6.198087e+00 |
| potchinkov1 | | | | | |
| potchinkov2 | | | | | |

| Problema | GI | GF | MRR | NR | fxR |
|---|---|---|---|---|---|
| potchinkov3 | | | | | |
| potchinkov4a | | | | | |
| potchinkov4b | 24 | 144 | 0 | 1 | 1.353041e-01 |
| potchinkovPL | | | | | |
| potchinkovPLR | 168 | 5472 | 12192.0 | 3 | -9.977536e-01 |
| powell1 | 11 | 61 | 3.0 | 2 | -1.000000e+00 |
| priceK | 32 | 189 | 14.5 | 3 | -3.000052e+00 |
| priceS3 | 1331 | 226981 | 21.5 | 3 | -3.674412e+00 |
| priceS4 | 256 | 194481 | 319.5 | 3 | -4.087524e+00 |
| priceS5 | 243 | 371293 | 155.0 | 3 | -4.452495e+00 |
| priceS6 | 64 | 117649 | 36.5 | 3 | -5.145461e+00 |
| priceT | 2197 | 389017 | 30.0 | 3 | 4.376236e+00 |
| priceU | 64 | 117649 | 38.2 | 6 | -3.571090e+00 |
| reemtsen1 | 1331 | 226981 | 77.0 | 3 | 1.524880e-01 |
| reemtsen2 | 441 | 14641 | 35.3 | 4 | 5.834113e-02 |
| reemtsen3 | 441 | 14641 | 62.0 | 3 | 7.354245e-01 |
| reemtsen4 | 441 | 14641 | 24.0 | 3 | 2.130556e-04 |
| reemtsen5 | 1331 | 226981 | 53.5 | 5 | 8.893950e-02 |
| still1 | 21 | 121 | 0 | 1 | 1.000000e+00 |
| tanaka1 | 16 | 95 | 0 | 1 | -1.000000e+00 |
| teo1 | 300 | 1800 | 0 | 1 | 1.250721e-01 |
| teo2 | 300 | 1800 | 5.0 | 2 | 1.746271e-01 |
| watson1 | 21 | 121 | 0 | 1 | -2.500022e-01 |
| watson10 | 2601 | 90601 | 25.5 | 3 | 2.752565e-01 |
| watson11 | 2601 | 90601 | 1096.0 | 3 | -4.386134e+00 |
| watson12 | 2601 | 90601 | 134.5 | 3 | 1.950909e+00 |
| watson13 | 2601 | 90601 | 139.0 | 3 | 1.949589e+00 |
| watson14 | 11 | 61 | 0 | 1 | 2.200000e+00 |
| watson2 | 11 | 61 | 0 | 1 | 2.618034e+00 |
| watson3 | 11 | 61 | 0 | 1 | 5.334687e+00 |
| watson4a | 11 | 61 | 7.5 | 3 | 6.490421e-01 |
| watson4b | 11 | 61 | 41.0 | 3 | 6.160831e-01 |
| watson4c | 11 | 61 | 41.0 | 3 | 6.156530e-01 |
| watson5 | 11 | 61 | 21.0 | 2 | 4.301117e+00 |
| watson6 | 11 | 61 | 0 | 1 | 9.715885e+01 |
| watson7 | 121 | 3721 | 0 | 1 | 1.000000e+00 |
| watson8 | 121 | 3721 | 33.0 | 3 | 2.435643e+00 |
| watson9 | 441 | 14641 | 0 | 1 | -1.200000e+01 |
| zhou1 | 11 | 61 | 3.5 | 3 | 1.783866e-01 |

Tabela J.2: Resultados numéricos para o método de discretização, versão Reemtsen

São apresentados na Tabela J.3 os resultados numéricos para o método de discretização na versão Hettich com pontos pseudo-aleatórios. A notação usada é: "Problema" é o nome do problema na base de dados, "GI" número de pontos de Halton no conjunto inicial, "GF" é o número de pontos de Halton no conjunto final, "MR" é o número de restrições médias usadas nos subproblemas, "N" é o número de subproblemas resolvidos e "fx" é o valor da função objectivo na solução encontrada.

| Problema | GI | GF | MR | N | fx |
|---|---|---|---|---|---|
| andreson1 | 200 | 2000 | 23.5 | 12 | -3.333333e-01 |
| blankenship1 | 100 | 1000 | 511.1 | 10 | 2.019484e-28 |
| coopeL | 100 | 1000 | 11.3 | 13 | 3.431453e-01 |
| coopeM | 100 | 1000 | 13.2 | 10 | 1.000000e+00 |
| coopeN | 100 | 1000 | 27.2 | 10 | -4.665074e-09 |
| elke1 | 100 | 1000 | 410.8 | 27 | 1.083145e+00 |
| elke10 | 100 | 1000 | 219.8 | 21 | 2.187584e+00 |
| elke2 | | | | | |
| elke3 | 100 | 1000 | 668.9 | 36 | 1.696988e+00 |
| elke4 | | | | | |
| elke5 | 100 | 1000 | 266.5 | 18 | 2.396256e+00 |
| elke6 | | | | | |
| elke7 | 100 | 1000 | 389.6 | 29 | 3.408710e+00 |
| elke8 | 100 | 1000 | 179.5 | 18 | 2.152592e+00 |
| elke9 | 100 | 1000 | 199.9 | 19 | 1.634721e+00 |
| fang1 | 100 | 1000 | 13.3 | 13 | 4.794255e-01 |
| fang2 | 100 | 1000 | 506.0 | 10 | 6.931472e-01 |
| fang3 | 100 | 1000 | 511.1 | 10 | 1.718282e+00 |
| ferris1 | 100 | 1000 | 81.2 | 18 | 4.773755e-04 |
| ferris2 | 100 | 1000 | 70.3 | 15 | -1.786900e+00 |
| gockenbach1 | 100 | 1000 | 969.6 | 16 | -5.398206e-01 |
| gockenbach10 | 100 | 1000 | 1194.3 | 25 | -2.769733e-01 |
| gockenbach2 | 100 | 1000 | 995.2 | 16 | -5.750700e-01 |
| gockenbach3 | 100 | 1000 | 985.6 | 16 | -5.083645e-01 |
| gockenbach4 | 100 | 1000 | 484.5 | 17 | -7.538303e-01 |
| gockenbach5 | 100 | 1000 | 1170.5 | 17 | -2.027431e-01 |
| gockenbach6 | 100 | 1000 | 1672.8 | 30 | -6.907565e-01 |
| gockenbach7 | 100 | 1000 | 1872.2 | 35 | -7.880436e-01 |
| gockenbach8 | 100 | 1000 | 1486.5 | 22 | -7.686988e-01 |
| gockenbach9 | 100 | 1000 | 1302.8 | 21 | -1.199237e+00 |
| goerner1 | 100 | 1000 | 42.9 | 18 | 4.239240e-03 |
| goerner2 | 100 | 1000 | 54.8 | 18 | 2.632503e-03 |

| Problema | GI | GF | MR | N | fx |
|---|---|---|---|---|---|
| goerner3 | 100 | 1000 | 163.8 | 17 | 4.392261e-05 |
| goerner4 | 200 | 2000 | 53.6 | 18 | 4.943628e-02 |
| goerner5 | 200 | 2000 | 59.2 | 16 | 2.587510e-02 |
| goerner6 | 200 | 2000 | 120.0 | 26 | 9.429062e-04 |
| goerner7 | 200 | 2000 | 76.9 | 22 | 2.888078e-02 |
| gugat1 | 100 | 1000 | 161.6 | 23 | 2.833840e-03 |
| gugat2 | 100 | 1000 | 268.0 | 23 | 8.052256e-02 |
| gugat3 | 100 | 1000 | 23.4 | 11 | 7.109648e-01 |
| gugat4a | 100 | 1000 | 1815.6 | 11 | 2.010239e-07 |
| gugat4b | 100 | 1000 | 125.9 | 20 | 6.624897e-03 |
| gugat4c | 100 | 1000 | 109.1 | 20 | 5.028253e-03 |
| gugat4d | 100 | 1000 | 168.6 | 26 | 9.093010e-01 |
| gugat4e | 100 | 1000 | 292.4 | 31 | 4.548613e-02 |
| gugat4f | 100 | 1000 | 1842.4 | 11 | 2.584890e-07 |
| gugat5a | 100 | 1000 | 249.1 | 19 | 8.495515e-05 |
| gugat5b | 100 | 1000 | 123.0 | 24 | 3.028299e-01 |
| gugat5c | 100 | 1000 | 101.1 | 20 | 8.468996e-03 |
| gugat5d | 100 | 1000 | 105.6 | 24 | 9.093300e-01 |
| gugat5e | 100 | 1000 | 91.6 | 20 | 5.402965e-02 |
| gugat5f | 100 | 1000 | 208.0 | 19 | 1.494680e-04 |
| gugat6 | 100 | 1000 | 383.3 | 12 | 2.356818e-05 |
| gugat7 | 100 | 1000 | 123.5 | 18 | 1.089678e-03 |
| hettich1 | 200 | 2000 | 76.6 | 23 | 2.888078e-02 |
| hettich10 | 100 | 1000 | 23.1 | 17 | 9.999994e-01 |
| hettich10c | 100 | 1000 | 1098.9 | 810 | -9.999900e+19 |
| hettich2 | 100 | 1000 | 36.6 | 20 | 5.373077e-01 |
| hettich3 | 100 | 1000 | 47.1 | 19 | 3.520491e-03 |
| hettich4 | 100 | 1000 | 23.0 | 11 | 1.000000e+00 |
| hettich5 | 200 | 2000 | 63.3 | 18 | 5.295236e-01 |
| hettich6 | 200 | 2000 | 56.5 | 21 | 2.691857e-02 |
| hettich7 | 200 | 2000 | 55.6 | 20 | 1.599404e-01 |
| hettich8 | 100 | 1000 | 56.4 | 18 | 5.634597e-03 |
| hettich9 | 200 | 2000 | 76.8 | 19 | 3.259643e-03 |
| kortanek1 | 100 | 1000 | 11.8 | 13 | 3.221152e+00 |
| kortanek2 | 200 | 2000 | 26.2 | 11 | 6.847201e-01 |
| kortanek3 | 100 | 1000 | 35.7 | 18 | 2.115774e-04 |
| kortanek4 | 100 | 1000 | 460.6 | 11 | 2.712674e-05 |
| leon1 | 100 | 1000 | 42.9 | 18 | 4.459487e-03 |
| leon10 | 100 | 1000 | 38.0 | 19 | 5.373077e-01 |
| leon11 | 100 | 1000 | 30.9 | 18 | 4.788273e-02 |
| leon12 | 100 | 1000 | 11.9 | 17 | -1.000038e+00 |
| leon13 | 100 | 1000 | 12.2 | 17 | 2.360648e-01 |

| Problema | GI | GF | MR | N | fx |
|---|---|---|---|---|---|
| leon14 | 100 | 1000 | 11.9 | 17 | 6.666646e-01 |
| leon15 | 100 | 1000 | 12.2 | 15 | -6.666667e-01 |
| leon16 | 100 | 1000 | 15.4 | 18 | 1.733774e+00 |
| leon17 | 100 | 1000 | 12.1 | 10 | -2.000000e+00 |
| leon18 | 100 | 1000 | 12.9 | 11 | -1.750000e+00 |
| leon19 | 100 | 1000 | 28.1 | 11 | 7.858376e-01 |
| leon2 | | | | | |
| leon3 | 100 | 1000 | 80.7 | 18 | 5.184306e-04 |
| leon4 | 100 | 1000 | 52.2 | 12 | 2.602621e-03 |
| leon5 | 100 | 1000 | 56.2 | 18 | 1.425498e-02 |
| leon6 | 100 | 1000 | 109.5 | 17 | 1.525712e-04 |
| leon7 | 100 | 1000 | 79.8 | 17 | 2.061492e-03 |
| leon8 | 100 | 1000 | 54.7 | 19 | 5.357741e-02 |
| leon9 | 100 | 1000 | 60.3 | 21 | 1.622843e-01 |
| li1 | 100 | 1000 | 28.2 | 23 | 1.740418e+05 |
| li2 | 100 | 1000 | 20.6 | 21 | 3.164839e+04 |
| lin1 | 200 | 2000 | 24.8 | 17 | -1.824563e+00 |
| liu1 | 100 | 1000 | 10.8 | 18 | -4.656786e+00 |
| liu2 | 100 | 1000 | 10.3 | 16 | -3.363507e+00 |
| liu3 | 100 | 1000 | 23.4 | 18 | 1.495741e+02 |
| matlab1 | 100 | 1000 | 33.2 | 22 | 9.444497e-02 |
| matlab2 | 200 | 2000 | 19.6 | 13 | 1.536154e-04 |
| polak1 | 200 | 2000 | 48.2 | 13 | 5.443778e+00 |
| polak2 | 200 | 2000 | 45.6 | 15 | 6.198127e+00 |
| potchinkov1 | | | | | |
| potchinkov2 | | | | | |
| potchinkov3 | 200 | 2000 | 752.3 | 25 | 1.084140e-02 |
| potchinkov4a | | | | | |
| potchinkov4b | 100 | 1000 | 32.2 | 11 | 7.801844e-02 |
| potchinkovPL | | | | | |
| potchinkovPLR | 200 | 2000 | 3212.2 | 19 | -9.797381e-01 |
| powell1 | 100 | 1000 | 11.9 | 17 | -1.000038e+00 |
| priceK | 100 | 1000 | 11.9 | 16 | -3.000001e+00 |
| priceS3 | 300 | 3000 | 25.2 | 15 | -3.679074e+00 |
| priceS4 | 400 | 4000 | 36.3 | 15 | -4.063623e+00 |
| priceS5 | 500 | 5000 | 49.0 | 12 | -4.476345e+00 |
| priceS6 | 600 | 6000 | 54.7 | 13 | -5.327011e+00 |
| priceT | 300 | 3000 | 26.9 | 15 | 4.397542e+00 |
| priceU | 600 | 6000 | 51.1 | 15 | -4.061943e+00 |
| priceU | 600 | 6000 | 51.1 | 15 | -4.061943e+00 |
| reemtsen1 | 300 | 3000 | 100.6 | 24 | 1.263707e-01 |
| reemtsen2 | 200 | 2000 | 103.5 | 22 | 5.687940e-02 |

| Problema | GI | GF | MR | N | fx |
|---|---|---|---|---|---|
| reemtsen3 | 200 | 2000 | 92.7 | 20 | 7.032254e-01 |
| reemtsen4 | 200 | 2000 | 373.4 | 38 | 1.831106e-04 |
| reemtsen5 | 300 | 3000 | 93.4 | 26 | 8.381576e-02 |
| still1 | 100 | 1000 | 12.7 | 16 | 9.999994e-01 |
| tanaka1 | 100 | 1000 | 14.1 | 10 | -1.000000e+00 |
| teo1 | 100 | 1000 | 11.1 | 10 | 1.250721e-01 |
| teo2 | 100 | 1000 | 10.6 | 12 | 1.746026e-01 |
| watson1 | 100 | 1000 | 12.1 | 10 | -2.500022e-01 |
| watson10 | 200 | 2000 | 23.2 | 10 | 2.733173e-01 |
| watson11 | 200 | 2000 | 21.6 | 17 | -4.393431e+00 |
| watson12 | 200 | 2000 | 20.7 | 13 | 1.893685e+00 |
| watson13 | 200 | 2000 | 20.7 | 13 | 1.884985e+00 |
| watson14 | 100 | 1000 | 11.1 | 17 | 2.195864e+00 |
| watson2 | 100 | 1000 | 511.1 | 10 | 2.618034e+00 |
| watson3 | 100 | 1000 | 11.1 | 17 | 5.328279e+00 |
| watson4a | 100 | 1000 | 26.4 | 18 | 6.481903e-01 |
| watson4b | 100 | 1000 | 39.2 | 19 | 6.160649e-01 |
| watson4c | 100 | 1000 | 74.3 | 18 | 6.156517e-01 |
| watson5 | 100 | 1000 | 15.1 | 17 | 4.300221e+00 |
| watson6 | 100 | 1000 | 12.1 | 10 | 9.715885e+01 |
| watson7 | 200 | 2000 | 23.2 | 10 | 1.000000e+00 |
| watson8 | 200 | 2000 | 41.9 | 23 | 2.385848e+00 |
| watson9 | 200 | 2000 | 32.1 | 20 | -1.200001e+01 |
| zhou1 | 100 | 1000 | 11.9 | 17 | 1.783935e-01 |

Tabela J.3: Resultados numéricos para o método de discretização, versão Hettich com pontos pseudo-aleatórios

## J.2 Método de programação quadrática sequencial

O problema (8.1.27) foi resolvido com o pacote de software NPSOL. As fórmulas para as derivadas da função $\mathcal{L}_l^*$ com respeito às variáveis $t$, $h$, e $w$ estão descritas no Apêndice F.

Na Tabela J.4 a solução do problema de PSIQ anterior é usada como aproximação inicial para o problema de PSIQ seguinte (sem reiniciação da aproximação inicial). Na Tabela J.5 a solução inicial para o problema de PSIQ é sempre calculada com base em (8.1.29).

Foram usadas as opções por omissão para o algoritmo.

Nas duas tabelas, "Problema" refere-se ao nome do problema na base de dados SIPAM-PL, "IE" é o número de iterações exteriores (número de problemas de PSIQ resolvidos),

"NM" representa o número de cálculos da função mérito, "NR" é o número de restrições calculadas, "NJ" é o número de Jacobianos calculados e "FX" é o valor da função objectivo na solução encontrada. Note-se que a avaliação do integral da restrição (e gradiente da restrição) originam um número de cálculos da restrição que não é fixo. Os valores nas colunas "NR" e "NJ" referem-se ao total de chamadas às rotinas de interface do SIPAMPL.

| Problema | IE | NM | NC | NJ | FX |
|---|---|---|---|---|---|
| blankenship1 | 401 | 10244 | 1033673 | 327512 | -1.617528e-06 |
| coopeL | 8 | 47 | 79223 | 73132 | 3.418237e-01 |
| coopeN | 7 | 55 | 38779 | 31762 | -9.159415e-05 |
| ferris1 | 12 | 81 | 897722 | 875111 | 5.266086e-02 |
| ferris2 | 5 | 25 | 194478 | 187209 | -1.787311e+00 |
| goerner1 | 10 | 49 | 804816 | 794274 | 6.769590e-02 |
| goerner2 | 10 | 43 | 636520 | 625415 | 8.750914e-03 |
| goerner3 | 6 | 29 | 733148 | 722627 | 1.301205e-03 |
| gugat1 | 4 | 20 | 759288 | 750486 | 1.351459e-02 |
| gugat2 | 2 | 35 | 2359000 | 2331653 | -9.106489e-01 |
| gugat4a | 1 | 0 | 104288 | 104444 | 1.101723e-29 |
| gugat4b | 3 | 43 | 4960218 | 4922180 | 1.768198e-02 |
| gugat4c | 1 | 3 | 182106 | 180428 | 1.651342e-02 |
| gugat4d | 1 | 25 | 1048332 | 1035144 | 2.583631e-09 |
| gugat4e | 2 | 27 | 2201596 | 2181596 | 2.298675e-02 |
| gugat4f | 1 | 0 | 124940 | 125096 | 4.444564e-24 |
| gugat5a | 375 | 762 | 7117506 | 6757628 | 6.563510e+01 |
| gugat5b | 6 | 34 | 1336222 | 1316143 | 3.165008e-01 |
| gugat5c | 2 | 24 | 124406 | 115650 | -1.247740e-04 |
| gugat5d | 1 | 2 | 2741390 | 2740013 | 6.333853e-02 |
| gugat5e | 2 | 4 | 764160 | 759136 | -3.682024e-02 |
| gugat5f | 6 | 30 | 284904 | 270335 | 2.099774e-04 |
| gugat6 | 1 | 0 | 52324 | 52384 | 1.347200e-05 |
| gugat7 | 5 | 26 | 75912 | 65807 | 2.701267e-04 |
| hettich10 | 6 | 67 | 69460 | 55056 | 9.984267e-01 |
| hettich10c | 11 | 69 | 359866 | 343548 | 1.000436e+00 |
| hettich2 | 8 | 40 | 462776 | 450027 | 5.376187e-01 |
| hettich3 | 2 | 5 | 74558 | 73195 | 5.867161e-05 |
| hettich4 | 3 | 23 | 159712 | 153651 | 1.672780e-01 |
| hettich8 | 4 | 29 | 48242 | 42853 | 3.579080e-03 |
| leon1 | 14 | 62 | 291652 | 274605 | 2.513939e-02 |
| leon10 | 15 | 50 | 292880 | 274630 | 5.391943e-01 |
| leon11 | 10 | 66 | 1212612 | 1194668 | 4.921379e-02 |
| leon12 | 15 | 68 | 176048 | 153344 | -1.001034e+00 |
| leon13 | 8 | 62 | 30554 | 22865 | 2.334567e-01 |
| leon14 | 11 | 80 | 50410 | 39077 | 6.635158e-01 |

| Problema | IE | NM | NC | NJ | FX |
|---|---|---|---|---|---|
| leon15 | 7 | 32 | 121961 | 118258 | -6.669716e-01 |
| leon16 | 8 | 59 | 169770 | 162792 | 1.855109e+00 |
| leon18 | 7 | 38 | 50459 | 45715 | -1.751574e+00 |
| leon19 | 7 | 30 | 95598 | 89255 | 7.921413e-01 |
| leon2 | 9 | 44 | 721086 | 708650 | 2.611987e-04 |
| leon3 | 7 | 34 | 256556 | 248985 | -1.270186e-03 |
| leon4 | 11 | 55 | 1038218 | 1022828 | 3.895892e-03 |
| leon5 | 15 | 87 | 10335986 | 10267092 | 2.457125e-01 |
| leon6 | 9 | 45 | 501036 | 487487 | 8.547468e-03 |
| leon7 | 8 | 42 | 181028 | 169428 | 1.378229e-02 |
| leon8 | 4 | 39 | 4733746 | 4697619 | 1.206286e-01 |
| leon9 | 6 | 57 | 8674996 | 8613071 | 2.183743e-01 |
| liu1 | 1 | 0 | 29688 | 29685 | -4.653846e+00 |
| liu2 | 19 | 90 | 256581 | 232672 | -3.373690e+00 |
| liu3 | 9 | 110 | 9729746 | 9676614 | 1.537981e+02 |
| matlab1 | 7 | 44 | 93391742 | 93256259 | 2.940148e+00 |
| powell1 | 401 | 800 | 565481 | 314890 | -1.199200e+03 |
| priceK | 14 | 90 | 519950 | 489382 | -3.003357e+00 |
| still1 | 2 | 18 | 33000 | 25967 | 7.998385e-01 |
| watson1 | 10 | 59 | 48866 | 39888 | -2.887967e-01 |
| watson14 | 27 | 104 | 34963 | 19651 | 2.190568e+00 |
| watson2 | 19 | 76 | 74290 | 63836 | 1.941849e-01 |
| watson3 | 30 | 162 | 304183 | 273158 | 5.331010e+00 |
| watson4a | 5 | 28 | 41699 | 35364 | 6.190993e-01 |
| watson4b | 8 | 39 | 290736 | 285190 | 6.366405e-01 |
| watson4c | 4 | 32 | 45681 | 41888 | 6.142454e-01 |
| watson5 | 6 | 47 | 54676 | 48197 | 4.298711e+00 |
| watson6 | | | | | |
| zhou1 | 4 | 39 | 12617 | 7520 | 1.370410e+00 |

Tabela J.4: Resultados numéricos sem reiniciação da aproximação inicial

| Problema | IE | NM | NC | NJ | FX |
|---|---|---|---|---|---|
| blankenship1 | 9 | 46 | 18370 | 12475 | -6.432359e-04 |
| coopeL | 7 | 18 | 164457 | 160550 | 0.000000e+00 |
| coopeN | 11 | 106 | 196554 | 182603 | 2.008104e-04 |
| ferris1 | 6 | 45 | 487488 | 474063 | 3.307131e-02 |
| ferris2 | 5 | 29 | 162872 | 155045 | -1.791432e+00 |
| goerner1 | 10 | 63 | 659692 | 638632 | 1.061885e-01 |
| goerner2 | 10 | 46 | 523116 | 512053 | 1.511158e-02 |
| goerner3 | 11 | 52 | 909580 | 891860 | 4.120990e-03 |
| gugat1 | 3 | 31 | 580022 | 568459 | 9.559516e-03 |
| gugat2 | 2 | 4 | 765988 | 762520 | 1.707727e+03 |
| gugat4a | 6 | 49 | 966066 | 943939 | 1.893978e-03 |
| gugat4b | 4 | 35 | 2338866 | 2323484 | 2.441676e-01 |
| gugat4c | 1 | 18 | 144672 | 138750 | 3.799838e-10 |
| gugat4d | 2 | 8 | 1158292 | 1153661 | 3.881189e-01 |
| gugat4e | 1 | 23 | 725770 | 710612 | 5.647261e-10 |
| gugat4f | 1 | 0 | 127878 | 128034 | 1.827898e-24 |
| gugat5a | 1 | 0 | 19980 | 19968 | 4.763963e-05 |
| gugat5b | 4 | 33 | 4225656 | 4185875 | 2.040105e-01 |
| gugat5c | 1 | 2 | 55198 | 54101 | 4.895108e-05 |
| gugat5d | 1 | 27 | 445532 | 434154 | 1.250000e-01 |
| gugat5e | 2 | 4 | 424198 | 419172 | -3.682024e-02 |
| gugat5f | 8 | 48 | 5352970 | 5318879 | 8.860540e-03 |
| gugat6 | 1 | 0 | 53524 | 53584 | 1.347200e-05 |
| gugat7 | 4 | 31 | 607734 | 597277 | 4.934073e-04 |
| hettich10 | 5 | 34 | 109042 | 101988 | 9.981165e-01 |
| hettich10c | 11 | 63 | 233926 | 218439 | 9.992412e-01 |
| hettich2 | 4 | 39 | 486832 | 477634 | 5.462190e-01 |
| hettich3 | 5 | 35 | 230046 | 222409 | 2.100701e-03 |
| hettich4 | 3 | 23 | 67172 | 61099 | 1.672780e-01 |
| hettich8 | 3 | 26 | 46102 | 40466 | 1.167680e-03 |
| leon1 | 7 | 49 | 377700 | 366736 | 6.667777e-02 |
| leon10 | 21 | 89 | 1902412 | 1861465 | 1.211598e+00 |
| leon11 | 9 | 54 | 686330 | 669428 | 1.539880e-02 |
| leon12 | 9 | 44 | 231683 | 224723 | -1.001309e+00 |
| leon13 | 5 | 34 | 42192 | 36092 | 2.111357e-01 |
| leon14 | 7 | 38 | 76836 | 72737 | 6.644717e-01 |
| leon15 | 16 | 69 | 161254 | 150767 | -6.662225e-01 |
| leon16 | 10 | 49 | 181836 | 175437 | 1.869855e+00 |
| leon18 | 10 | 52 | 100540 | 93064 | -1.750329e+00 |
| leon19 | 8 | 37 | 91656 | 85733 | 7.981162e-01 |
| leon2 | 8 | 41 | 944284 | 930491 | 1.276432e-02 |

| Problema | IE | NM | NC | NJ | FX |
|---|---|---|---|---|---|
| leon3 | 6 | 33 | 243996 | 236099 | 1.308901e-02 |
| leon4 | 8 | 46 | 628282 | 616635 | 4.457146e-03 |
| leon5 | 4 | 31 | 3502756 | 3464149 | 1.513474e-02 |
| leon6 | 7 | 36 | 459748 | 452057 | 9.234195e-03 |
| leon7 | 6 | 34 | 222014 | 213683 | 1.540178e-02 |
| leon8 | 4 | 39 | 2816538 | 2780411 | 1.206286e-01 |
| leon9 | 6 | 57 | 5422354 | 5360429 | 2.183743e-01 |
| liu1 | 1 | 0 | 29706 | 29703 | -4.653846e+00 |
| liu2 | 11 | 52 | 234102 | 225001 | -3.367475e+00 |
| liu3 | 5 | 52 | 6579422 | 6551999 | 1.539197e+02 |
| matlab1 | 6 | 37 | 22716178 | 22652882 | 0.000000e+00 |
| powell1 | 19 | 76 | 185512 | 172835 | -1.007096e+00 |
| priceK | 6 | 37 | 92096 | 88678 | -3.000738e+00 |
| still1 | 2 | 18 | 33606 | 26575 | 7.998385e-01 |
| watson1 | 10 | 69 | 53579 | 44115 | -2.886135e-01 |
| watson14 | 26 | 90 | 46506 | 33730 | 2.199133e+00 |
| watson2 | 14 | 52 | 45701 | 39637 | 3.791673e-01 |
| watson3 | 18 | 108 | 165086 | 142087 | 5.334306e+00 |
| watson4a | 6 | 35 | 62660 | 56302 | 6.409683e-01 |
| watson4b | 9 | 54 | 205013 | 197067 | 6.442224e-01 |
| watson4c | 5 | 33 | 62482 | 58340 | 6.158162e-01 |
| watson5 | 10 | 96 | 584572 | 571823 | 4.298538e+00 |
| watson6 | 20 | 89 | 235384 | 204772 | 1.146141e+02 |
| zhou1 | 13 | 65 | 181527 | 171229 | 1.766077e-01 |

Tabela J.5: Resultados numéricos com reiniciação da aproximação inicial

## J.3 Método de penalidade

Para obter os resultados numéricos foram usados os seguintes valores: na fórmula adaptativa do trapézio considerou-se $ni = 20$, $\rho = 10^{-3}$, $\epsilon_T = 10^{-8}$; as tolerâncias para a paragem das iterações interiores foi $\delta_1 = 10^{-4}$ e para as iterações exteriores foi $\delta_2 = 10^{-4}$. O parâmetro de penalidade $\mu$ e a precisão da aproximação $\epsilon$ são respectivamente aumentadas e diminuídas por um factor de 10, sempre que se considerou ser necessário uma actualização. Os valores iniciais para estes parâmetros foram respectivamente 1 e $10^{-4}$. $\mu$ é incrementado por um factor de 1.1 ou 2 para a função exponencial $\phi_E$. O algoritmo termina se o parâmetro de penalidade exceder $10^8$ ou se $\epsilon$ for inferior a $10^{-8}$. $\tau$ é mantido igual a $\epsilon$ durante todo o procedimento.

Os resultados numéricos para os problemas são apresentados nas Tabelas J.6, J.7, J.8, J.9, J.10 e J.11. "Problema" é o nome do problema na base de dados do SIPAMPL. "TI" é o número total de iterações interiores e "IE" é o número de iterações exteriores (número

de funções de penalidade $\phi$ minimizadas). "NP" é o número de avaliações da função de penalidade e "NGP" é o número de gradientes calculados da função de penalidade. "FX" é o valor da função objectivo na solução encontrada. "$\mu$" e "$\epsilon$" são os valores finais dos parâmetros $\mu$ e $\epsilon$.

| Problema | TI | IE | NP | NGP | FX | $\mu$ | $\epsilon$ |
|---|---|---|---|---|---|---|---|
| blankenship1 | 26 | 6 | 212 | 32 | 1.079356e-05 | 1.0e+01 | 1.0e-09 |
| coopeL | 27 | 4 | 153 | 31 | 3.430775e-01 | 1.0e+02 | 1.0e-05 |
| coopeN | 33 | 7 | 240 | 40 | -1.241784e-07 | 1.0e+02 | 1.0e-09 |
| elke10 | 24 | 3 | 180 | 27 | 3.013692e+00 | 1.0e+01 | 1.0e-05 |
| elke1std | 34 | 4 | 224 | 38 | 1.089146e+00 | 1.0e+02 | 1.0e-05 |
| elke2std | 84 | 5 | 389 | 89 | 2.557289e+00 | 1.0e+03 | 1.0e-05 |
| elke3std | 54 | 7 | 329 | 61 | 1.524999e+00 | 1.0e+05 | 1.0e-05 |
| elke4std | 91 | 5 | 409 | 96 | 2.504717e+00 | 1.0e+03 | 1.0e-05 |
| elke5std | 67 | 4 | 281 | 71 | 2.453787e+00 | 1.0e+02 | 1.0e-05 |
| elke6std | 94 | 6 | 424 | 100 | 4.526954e+00 | 1.0e+03 | 1.0e-06 |
| elke7std | 74 | 5 | 348 | 79 | 2.917988e+00 | 1.0e+03 | 1.0e-05 |
| elke8 | 43 | 3 | 229 | 46 | 2.153125e+00 | 1.0e+01 | 1.0e-05 |
| elke9 | 26 | 3 | 171 | 29 | 1.874673e+00 | 1.0e+01 | 1.0e-05 |
| ferris1 | 814 | 4 | 2594 | 816 | 2.074924e-03 | 1.0e+02 | 1.0e-05 |
| ferris2 | 214 | 4 | 737 | 218 | -1.782561e+00 | 1.0e+02 | 1.0e-05 |
| goerner1 | 52 | 4 | 292 | 56 | 4.297923e-03 | 1.0e+02 | 1.0e-05 |
| goerner2 | 338 | 4 | 1137 | 342 | 2.656481e-03 | 1.0e+02 | 1.0e-05 |
| goerner3 | 44 | 4 | 216 | 48 | 4.502669e-03 | 1.0e+02 | 1.0e-05 |
| gugat1 | 12 | 5 | 106 | 17 | 1.894149e-02 | 1.0e+03 | 1.0e-05 |
| gugat2 | 9 | 9 | 108 | 18 | 1.624898e+01 | 1.0e+09 | 1.0e-04 |
| gugat4a | 3 | 3 | 57 | 6 | 5.182333e-05 | 1.0e+01 | 1.0e-05 |
| gugat4b | 4 | 4 | 58 | 8 | 2.268720e-02 | 1.0e+02 | 1.0e-05 |
| gugat4c | 6 | 4 | 61 | 10 | 1.472256e-02 | 1.0e+02 | 1.0e-05 |
| gugat4d | 14 | 4 | 74 | 18 | 8.717525e-01 | 1.0e+02 | 1.0e-05 |
| gugat4e | 287 | 4 | 971 | 291 | 6.324722e-02 | 1.0e+02 | 1.0e-05 |
| gugat4f | 8 | 7 | 185 | 15 | 6.921816e-04 | 1.0e+05 | 1.0e-05 |
| gugat5a | 277 | 6 | 1051 | 283 | 3.843820e-03 | 1.0e+04 | 1.0e-05 |
| gugat5b | 48 | 5 | 354 | 53 | 3.131977e-01 | 1.0e+03 | 1.0e-05 |
| gugat5c | 5 | 5 | 66 | 10 | 2.985734e-02 | 1.0e+03 | 1.0e-05 |
| gugat5d | 32 | 4 | 159 | 36 | 7.374376e-01 | 1.0e+02 | 1.0e-05 |
| gugat5e | 8 | 5 | 70 | 13 | 9.255783e-02 | 1.0e+03 | 1.0e-05 |
| gugat5f | 5 | 5 | 103 | 10 | 3.443981e-04 | 1.0e+03 | 1.0e-05 |
| gugat6 | 4 | 4 | 74 | 8 | 7.150625e-05 | 1.0e+02 | 1.0e-05 |
| gugat7 | 55 | 5 | 296 | 60 | 1.239319e-03 | 1.0e+03 | 1.0e-05 |
| hettich10 | 25 | 4 | 131 | 29 | 9.999594e-01 | 1.0e+02 | 1.0e-05 |
| hettich10c | 33 | 4 | 160 | 37 | 9.999662e-01 | 1.0e+02 | 1.0e-05 |

| Problema | TI | IE | NP | NGP | FX | $\mu$ | $\epsilon$ |
|---|---|---|---|---|---|---|---|
| hettich2 | 73 | 5 | 368 | 78 | 5.397169e-01 | 1.0e+03 | 1.0e-05 |
| hettich3 | 9 | 9 | 293 | 18 | 3.385714e-03 | 1.0e+09 | 1.0e-04 |
| hettich4 | 42 | 5 | 266 | 47 | 1.066433e+00 | 1.0e+03 | 1.0e-05 |
| hettich8 | 29 | 5 | 237 | 34 | 5.669032e-03 | 1.0e+02 | 1.0e-06 |
| leon1 | 52 | 4 | 285 | 56 | 4.463282e-03 | 1.0e+02 | 1.0e-05 |
| leon10 | 71 | 5 | 358 | 76 | 5.382721e-01 | 1.0e+03 | 1.0e-05 |
| leon11 | 62 | 6 | 413 | 68 | 4.845152e-02 | 1.0e+03 | 1.0e-06 |
| leon12 | 25 | 4 | 132 | 29 | -1.000008e+00 | 1.0e+02 | 1.0e-05 |
| leon13 | 466 | 4 | 1439 | 469 | 2.360362e-01 | 1.0e+02 | 1.0e-05 |
| leon14 | 832 | 6 | 2582 | 836 | 6.664719e-01 | 1.0e+03 | 1.0e-06 |
| leon15 | 26 | 4 | 131 | 30 | -6.666882e-01 | 1.0e+02 | 1.0e-05 |
| leon16 | 42 | 5 | 236 | 47 | 1.859152e+00 | 1.0e+03 | 1.0e-05 |
| leon18 | 454 | 5 | 1426 | 458 | -1.749994e+00 | 1.0e+03 | 1.0e-05 |
| leon19 | 394 | 5 | 1271 | 399 | 7.859067e-01 | 1.0e+03 | 1.0e-05 |
| leon2 | 38 | 5 | 285 | 43 | 1.591045e-04 | 1.0e+03 | 1.0e-05 |
| leon3 | 455 | 5 | 1493 | 459 | 9.739253e-04 | 1.0e+03 | 1.0e-05 |
| leon4 | 57 | 4 | 342 | 61 | 2.803503e-03 | 1.0e+02 | 1.0e-05 |
| leon5 | 36 | 9 | 468 | 45 | 1.409685e-02 | 1.0e+09 | 1.0e-04 |
| leon6 | 47 | 5 | 268 | 52 | 4.214630e-04 | 1.0e+02 | 1.0e-06 |
| leon7 | 805 | 5 | 2500 | 808 | 2.368049e-03 | 1.0e+03 | 1.0e-05 |
| leon8 | 87 | 9 | 566 | 96 | 6.431139e-02 | 1.0e+09 | 1.0e-04 |
| leon9 | 1172 | 12 | 4012 | 1182 | 1.531191e-01 | 1.0e+09 | 1.0e-07 |
| liu1 | 28 | 6 | 186 | 34 | -4.653827e+00 | 1.0e+03 | 1.0e-06 |
| liu2 | 30 | 5 | 188 | 35 | -3.363651e+00 | 1.0e+03 | 1.0e-05 |
| liu3 | 92 | 9 | 672 | 101 | 1.494336e+02 | 1.0e+09 | 1.0e-04 |
| matlab1 | 11 | 3 | 104 | 14 | 0.000000e+00 | 1.0e+00 | 1.0e-06 |
| powell1 | 558 | 5 | 1741 | 562 | -1.000005e+00 | 1.0e+03 | 1.0e-05 |
| priceK | 37 | 5 | 216 | 42 | -2.999992e+00 | 1.0e+03 | 1.0e-05 |
| still1 | 540 | 6 | 1822 | 545 | 1.000018e+00 | 1.0e+03 | 1.0e-06 |
| watson1 | 40 | 6 | 162 | 46 | -2.577511e-01 | 1.0e+04 | 1.0e-05 |
| watson14 | | | | | | | |
| watson2 | 27 | 4 | 158 | 31 | 1.944574e-01 | 1.0e+02 | 1.0e-05 |
| watson3 | 43 | 5 | 218 | 48 | 5.334686e+00 | 1.0e+03 | 1.0e-05 |
| watson4a | 63 | 5 | 301 | 68 | 6.491699e-01 | 1.0e+03 | 1.0e-05 |
| watson4b | 417 | 4 | 1349 | 420 | 6.166097e-01 | 1.0e+02 | 1.0e-05 |
| watson4c | 252 | 4 | 828 | 256 | 6.157630e-01 | 1.0e+02 | 1.0e-05 |
| watson5 | 51 | 5 | 259 | 56 | 4.301155e+00 | 1.0e+03 | 1.0e-05 |
| watson6 | | | | | | | |
| zhou1 | 589 | 4 | 1814 | 592 | 1.783713e-01 | 1.0e+02 | 1.0e-05 |

Tabela J.6: Resultados numéricos para a função de penalidade $\phi_S^1$

| Problema | TI | IE | NP | NGP | FX | $\mu$ |
|---|---|---|---|---|---|---|
| blankenship1 | 32 | 4 | 155 | 36 | -5.000000e-04 | 1.0e+03 |
| coopeL | 44 | 5 | 209 | 49 | 3.398985e-01 | 1.0e+04 |
| coopeN | 39 | 5 | 177 | 44 | -4.425000e-04 | 1.0e+04 |
| elke10 | 67 | 5 | 362 | 72 | 2.209662e+00 | 1.0e+04 |
| elke1std | 69 | 5 | 331 | 74 | 1.084093e+00 | 1.0e+04 |
| elke2std | 134 | 5 | 482 | 139 | 2.563349e+00 | 1.0e+04 |
| elke3std | 63 | 5 | 285 | 68 | 1.523708e+00 | 1.0e+04 |
| elke4std | 125 | 6 | 497 | 131 | 2.508644e+00 | 1.0e+05 |
| elke5std | 123 | 5 | 469 | 128 | 2.391329e+00 | 1.0e+04 |
| elke6std | 296 | 6 | 945 | 302 | 4.518200e+00 | 1.0e+05 |
| elke7std | 91 | 5 | 358 | 96 | 2.903613e+00 | 1.0e+04 |
| elke8 | 44 | 3 | 224 | 47 | 2.156404e+00 | 1.0e+02 |
| elke9 | 47 | 5 | 360 | 52 | 1.650023e+00 | 1.0e+04 |
| ferris1 | 1662 | 5 | 5103 | 1663 | 1.566432e-03 | 1.0e+04 |
| ferris2 | 1638 | 5 | 4999 | 1639 | -1.784107e+00 | 1.0e+04 |
| goerner1 | 300 | 5 | 1020 | 305 | 3.519880e-03 | 1.0e+04 |
| goerner2 | 772 | 5 | 2445 | 776 | 1.865854e-03 | 1.0e+04 |
| goerner3 | 219 | 5 | 792 | 224 | 1.822101e-03 | 1.0e+04 |
| gugat1 | 20 | 4 | 111 | 24 | 1.197674e-02 | 1.0e+03 |
| gugat2 | 12 | 9 | 226 | 21 | 1.005223e+02 | 1.0e+09 |
| gugat4a | 30 | 4 | 193 | 34 | -2.701690e-04 | 1.0e+03 |
| gugat4b | 632 | 5 | 1976 | 636 | 3.619715e-02 | 1.0e+04 |
| gugat4c | 8 | 3 | 45 | 11 | 2.030744e-02 | 1.0e+02 |
| gugat4d | 8 | 4 | 38 | 12 | 1.333799e+00 | 1.0e+03 |
| gugat4e | 185 | 5 | 616 | 190 | 6.074862e-02 | 1.0e+04 |
| gugat4f | 5 | 2 | 24 | 7 | 8.645833e-02 | 1.0e+01 |
| gugat5a | 4 | 2 | 20 | 6 | 2.562500e-01 | 1.0e+01 |
| gugat5b | 87 | 5 | 398 | 92 | 3.044815e-01 | 1.0e+04 |
| gugat5c | 30 | 5 | 152 | 35 | 1.786837e-02 | 1.0e+04 |
| gugat5d | 7 | 4 | 35 | 11 | 1.147986e+00 | 1.0e+03 |
| gugat5e | 13 | 5 | 79 | 18 | 8.547533e-02 | 1.0e+04 |
| gugat5f | 32 | 4 | 205 | 36 | -2.551582e-04 | 1.0e+03 |
| gugat6 | 15 | 4 | 96 | 19 | 2.909994e-03 | 1.0e+03 |
| gugat7 | 161 | 5 | 590 | 166 | 7.510325e-04 | 1.0e+04 |
| hettich10 | 31 | 5 | 156 | 36 | 9.982500e-01 | 1.0e+04 |
| hettich10c | 32 | 5 | 160 | 37 | 9.982500e-01 | 1.0e+04 |
| hettich2 | 121 | 5 | 457 | 126 | 5.362636e-01 | 1.0e+04 |
| hettich3 | 18 | 9 | 248 | 27 | 3.950577e-04 | 1.0e+09 |
| hettich4 | 58 | 6 | 290 | 64 | 9.957333e-01 | 1.0e+05 |
| hettich8 | 77 | 5 | 342 | 82 | 5.253954e-03 | 1.0e+04 |

| Problema | TI | IE | NP | NGP | FX | $\mu$ |
|---|---|---|---|---|---|---|
| leon1 | 363 | 5 | 1196 | 368 | 3.707298e-03 | 1.0e+04 |
| leon10 | 123 | 5 | 467 | 128 | 5.362636e-01 | 1.0e+04 |
| leon11 | 62 | 5 | 329 | 67 | 4.231747e-02 | 1.0e+04 |
| leon12 | 38 | 5 | 164 | 43 | -1.001405e+00 | 1.0e+04 |
| leon13 | 49 | 5 | 210 | 54 | 2.344011e-01 | 1.0e+04 |
| leon14 | 59 | 5 | 234 | 64 | 6.555655e-01 | 1.0e+04 |
| leon15 | 68 | 5 | 280 | 73 | -6.678418e-01 | 1.0e+04 |
| leon16 | 57 | 5 | 221 | 62 | 1.850006e+00 | 1.0e+04 |
| leon18 | 822 | 5 | 2518 | 825 | -1.753640e+00 | 1.0e+04 |
| leon19 | 1671 | 5 | 5102 | 1673 | 7.854734e-01 | 1.0e+04 |
| leon2 | 157 | 4 | 587 | 161 | -2.543861e-04 | 1.0e+03 |
| leon3 | 1336 | 4 | 4090 | 1338 | -4.934283e-04 | 1.0e+03 |
| leon4 | 283 | 4 | 951 | 287 | 1.911282e-03 | 1.0e+03 |
| leon5 | 64 | 4 | 390 | 68 | 1.915378e-02 | 1.0e+03 |
| leon6 | 582 | 4 | 1841 | 586 | -5.007928e-04 | 1.0e+03 |
| leon7 | 1237 | 5 | 3790 | 1239 | 2.412374e-03 | 1.0e+04 |
| leon8 | 367 | 5 | 1274 | 372 | 5.591292e-02 | 1.0e+04 |
| leon9 | 17 | 9 | 342 | 26 | 2.343289e-01 | 1.0e+09 |
| liu1 | 38 | 5 | 171 | 43 | -4.655794e+00 | 1.0e+04 |
| liu2 | 37 | 5 | 159 | 42 | -3.369120e+00 | 1.0e+04 |
| liu3 | 150 | 7 | 803 | 157 | 1.544838e+02 | 1.0e+06 |
| matlab1 | 13 | 1 | 97 | 14 | 5.205369e-02 | 1.0e+00 |
| powell1 | 35 | 5 | 151 | 40 | -1.004681e+00 | 1.0e+04 |
| priceK | 30 | 5 | 140 | 35 | -3.004476e+00 | 1.0e+04 |
| still1 | 52 | 6 | 254 | 58 | 9.966000e-01 | 1.0e+05 |
| watson1 | 45 | 6 | 171 | 51 | -2.898116e-01 | 1.0e+05 |
| watson14 | 592 | 6 | 1811 | 597 | 2.195116e+00 | 1.0e+05 |
| watson2 | 38 | 5 | 178 | 43 | 1.937176e-01 | 1.0e+04 |
| watson3 | 44 | 5 | 200 | 49 | 5.327237e+00 | 1.0e+04 |
| watson4a | 266 | 5 | 885 | 271 | 6.464482e-01 | 1.0e+04 |
| watson4b | 1166 | 5 | 3604 | 1169 | 6.160950e-01 | 1.0e+04 |
| watson4c | 573 | 5 | 1820 | 578 | 6.155342e-01 | 1.0e+04 |
| watson5 | 46 | 5 | 191 | 51 | 4.291581e+00 | 1.0e+04 |
| watson6 | 53 | 6 | 290 | 59 | 9.711019e+01 | 1.0e+05 |
| zhou1 | 41 | 5 | 180 | 46 | 1.769535e-01 | 1.0e+04 |

Tabela J.7: Resultados numéricos para a função de penalidade $\phi_S^2$

| Problema | TI | IE | NP | NGP | FX | $\mu$ | $\epsilon$ |
|---|---|---|---|---|---|---|---|
| blankenship1 | 24 | 9 | 319 | 33 | 3.607918e-05 | 1.0e+09 | 1.0e-04 |
| coopeL | 38 | 4 | 232 | 42 | 3.430722e-01 | 1.0e+02 | 1.0e-05 |
| coopeN | 31 | 9 | 380 | 40 | -6.177195e-05 | 1.0e+09 | 1.0e-04 |
| elke10 | 18 | 9 | 3061 | 27 | 1.312486e+02 | 1.0e+09 | 1.0e-04 |
| elke1std | 29 | 4 | 201 | 33 | 1.099645e+00 | 1.0e+02 | 1.0e-05 |
| elke2std | 46 | 4 | 233 | 50 | 2.574505e+00 | 1.0e+02 | 1.0e-05 |
| elke3std | 41 | 9 | 409 | 50 | 1.522144e+00 | 1.0e+09 | 1.0e-04 |
| elke4std | 67 | 4 | 319 | 71 | 2.544811e+00 | 1.0e+02 | 1.0e-05 |
| elke5std | 37 | 4 | 188 | 41 | 2.405766e+00 | 1.0e+02 | 1.0e-05 |
| elke6std | 101 | 6 | 428 | 107 | 4.562032e+00 | 1.0e+03 | 1.0e-06 |
| elke7std | 57 | 5 | 291 | 62 | 2.919170e+00 | 1.0e+03 | 1.0e-05 |
| elke8 | 38 | 9 | 2257 | 47 | 1.486950e+05 | 1.0e+09 | 1.0e-04 |
| elke9 | 182 | 5 | 2028 | 187 | 2.687295e+01 | 1.0e+03 | 1.0e-05 |
| ferris1 | 301 | 4 | 1073 | 305 | 3.556623e-03 | 1.0e+02 | 1.0e-05 |
| ferris2 | 302 | 4 | 1022 | 306 | -1.782217e+00 | 1.0e+02 | 1.0e-05 |
| goerner1 | 39 | 9 | 441 | 48 | 3.979672e-03 | 1.0e+09 | 1.0e-04 |
| goerner2 | 36 | 6 | 293 | 42 | 4.326237e-03 | 1.0e+03 | 1.0e-06 |
| goerner3 | | | | | | | |
| gugat1 | 15 | 5 | 107 | 20 | 1.618919e-02 | 1.0e+03 | 1.0e-05 |
| gugat2 | 4 | 4 | 30 | 8 | 5.813481e-01 | 1.0e+02 | 1.0e-05 |
| gugat4a | 14 | 10 | 281 | 24 | 7.897301e-03 | 1.0e+08 | 1.0e-05 |
| gugat4b | 61 | 9 | 382 | 70 | 2.712558e-01 | 1.0e+09 | 1.0e-04 |
| gugat4c | 4 | 4 | 47 | 8 | 2.628685e-02 | 1.0e+02 | 1.0e-05 |
| gugat4d | 20 | 4 | 110 | 24 | 1.129155e+00 | 1.0e+02 | 1.0e-05 |
| gugat4e | 136 | 5 | 576 | 141 | 6.002777e-02 | 1.0e+03 | 1.0e-05 |
| gugat4f | 16 | 9 | 372 | 25 | 1.280562e-02 | 1.0e+09 | 1.0e-04 |
| gugat5a | 201 | 5 | 1662 | 206 | 7.231532e-01 | 1.0e+03 | 1.0e-05 |
| gugat5b | 42 | 9 | 415 | 51 | 3.136972e-01 | 1.0e+09 | 1.0e-04 |
| gugat5c | 9 | 5 | 86 | 14 | 1.698311e-02 | 1.0e+03 | 1.0e-05 |
| gugat5d | 39 | 4 | 172 | 43 | 6.375011e-01 | 1.0e+02 | 1.0e-05 |
| gugat5e | 8 | 5 | 70 | 13 | 9.150440e-02 | 1.0e+03 | 1.0e-05 |
| gugat5f | 6 | 5 | 121 | 11 | 7.422211e-04 | 1.0e+03 | 1.0e-05 |
| gugat6 | 9 | 9 | 259 | 18 | 9.657300e-06 | 1.0e+09 | 1.0e-04 |
| gugat7 | 22 | 9 | 343 | 31 | 2.000411e-03 | 1.0e+09 | 1.0e-04 |
| hettich10 | 25 | 4 | 154 | 29 | 9.999725e-01 | 1.0e+02 | 1.0e-05 |
| hettich10c | 32 | 4 | 224 | 36 | 1.000005e+00 | 1.0e+02 | 1.0e-05 |
| hettich2 | 47 | 9 | 451 | 56 | 5.384851e-01 | 1.0e+09 | 1.0e-04 |
| hettich3 | 9 | 9 | 291 | 18 | 3.397089e-03 | 1.0e+09 | 1.0e-04 |
| hettich4 | 37 | 5 | 288 | 42 | 1.072242e+00 | 1.0e+03 | 1.0e-05 |
| hettich8 | 31 | 5 | 275 | 36 | 5.744441e-03 | 1.0e+02 | 1.0e-06 |
| leon1 | 45 | 9 | 438 | 54 | 4.672810e-03 | 1.0e+09 | 1.0e-04 |

| Problema | TI | IE | NP | NGP | FX | $\mu$ | $\epsilon$ |
|---|---|---|---|---|---|---|---|
| leon10 | 48 | 4 | 302 | 52 | 5.385027e-01 | 1.0e+02 | 1.0e-05 |
| leon11 | 53 | 9 | 544 | 62 | 4.716599e-02 | 1.0e+09 | 1.0e-04 |
| leon12 | 30 | 9 | 340 | 39 | -1.000051e+00 | 1.0e+09 | 1.0e-04 |
| leon13 | 39 | 6 | 322 | 45 | 2.361974e-01 | 1.0e+03 | 1.0e-06 |
| leon14 | 34 | 6 | 283 | 40 | 6.669250e-01 | 1.0e+03 | 1.0e-06 |
| leon15 | 33 | 4 | 211 | 37 | -6.666108e-01 | 1.0e+02 | 1.0e-05 |
| leon16 | 50 | 6 | 361 | 56 | 1.859153e+00 | 1.0e+03 | 1.0e-06 |
| leon18 | 678 | 5 | 2106 | 682 | -1.749706e+00 | 1.0e+03 | 1.0e-05 |
| leon19 | 128 | 4 | 522 | 132 | 7.884399e-01 | 1.0e+02 | 1.0e-05 |
| leon2 | 29 | 9 | 443 | 38 | 2.883327e-04 | 1.0e+09 | 1.0e-04 |
| leon3 | 402 | 5 | 1374 | 407 | 1.252551e-03 | 1.0e+02 | 1.0e-06 |
| leon4 | 35 | 9 | 473 | 44 | 3.488999e-03 | 1.0e+09 | 1.0e-04 |
| leon5 | 57 | 9 | 674 | 66 | 1.413844e-02 | 1.0e+09 | 1.0e-04 |
| leon6 | 49 | 5 | 357 | 54 | 4.780569e-04 | 1.0e+02 | 1.0e-06 |
| leon7 | 420 | 5 | 1459 | 424 | 1.093084e-02 | 1.0e+03 | 1.0e-05 |
| leon8 | 97 | 9 | 590 | 106 | 6.339948e-02 | 1.0e+09 | 1.0e-04 |
| leon9 | 26 | 4 | 366 | 30 | 7.030676e-01 | 1.0e+02 | 1.0e-05 |
| liu1 | 26 | 5 | 206 | 31 | -4.653803e+00 | 1.0e+03 | 1.0e-05 |
| liu2 | 28 | 9 | 304 | 37 | -3.361528e+00 | 1.0e+09 | 1.0e-04 |
| liu3 | 108 | 9 | 814 | 117 | 1.534736e+02 | 1.0e+09 | 1.0e-04 |
| matlab1 | 14 | 3 | 111 | 17 | 0.000000e+00 | 1.0e+00 | 1.0e-06 |
| powell1 | 36 | 5 | 230 | 41 | -1.000005e+00 | 1.0e+03 | 1.0e-05 |
| priceK | 26 | 5 | 194 | 31 | -3.000006e+00 | 1.0e+03 | 1.0e-05 |
| still1 | 41 | 5 | 309 | 46 | 1.000175e+00 | 1.0e+03 | 1.0e-05 |
| watson1 | 40 | 6 | 243 | 46 | -2.577493e-01 | 1.0e+04 | 1.0e-05 |
| watson14 | 512 | 8 | 1719 | 519 | 2.204591e+00 | 1.0e+04 | 1.0e-07 |
| watson2 | 24 | 4 | 166 | 28 | 1.944584e-01 | 1.0e+02 | 1.0e-05 |
| watson3 | 41 | 5 | 276 | 46 | 5.334767e+00 | 1.0e+03 | 1.0e-05 |
| watson4a | 46 | 9 | 427 | 55 | 6.502023e-01 | 1.0e+09 | 1.0e-04 |
| watson4b | 144 | 4 | 551 | 148 | 6.179816e-01 | 1.0e+02 | 1.0e-05 |
| watson4c | 25 | 5 | 248 | 30 | 6.198671e-01 | 1.0e+03 | 1.0e-05 |
| watson5 | 34 | 5 | 230 | 39 | 4.302436e+00 | 1.0e+03 | 1.0e-05 |
| watson6 | 76 | 6 | 468 | 82 | 9.716106e+01 | 1.0e+04 | 1.0e-05 |
| zhou1 | 40 | 9 | 367 | 49 | 1.783181e-01 | 1.0e+09 | 1.0e-04 |

Tabela J.8: Resultados numéricos para a função de penalidade $\phi_S^3$

| Problema | TI | IE | NP | NGP | FX | $\mu$ | $\epsilon$ |
|---|---|---|---|---|---|---|---|
| blankenship1 | 22 | 5 | 158 | 27 | 1.603559e-08 | 1.0e+01 | 1.0e-07 |
| coopeL | 53 | 9 | 306 | 62 | 3.431224e-01 | 1.0e+06 | 1.0e-06 |
| coopeN | 45 | 10 | 266 | 55 | -1.241026e-06 | 1.0e+05 | 1.0e-09 |
| elke10 | 38 | 9 | 385 | 47 | 2.259288e+00 | 1.0e+09 | 1.0e-04 |
| elke1std | 87 | 11 | 528 | 98 | 1.083747e+00 | 1.0e+07 | 1.0e-07 |
| elke2std | 117 | 11 | 534 | 128 | 2.556055e+00 | 1.0e+07 | 1.0e-07 |
| elke3std | 70 | 6 | 334 | 76 | 1.523681e+00 | 1.0e+04 | 1.0e-05 |
| elke4std | 131 | 10 | 609 | 141 | 2.492215e+00 | 1.0e+09 | 1.0e-05 |
| elke5std | 198 | 11 | 767 | 209 | 2.404472e+00 | 1.0e+08 | 1.0e-06 |
| elke6std | 311 | 9 | 1040 | 320 | 4.531870e+00 | 1.0e+06 | 1.0e-06 |
| elke7std | 108 | 11 | 514 | 119 | 2.905227e+00 | 1.0e+09 | 1.0e-06 |
| elke8 | 41 | 8 | 298 | 49 | 2.153319e+00 | 1.0e+05 | 1.0e-06 |
| elke9 | 24 | 10 | 294 | 34 | 1.753995e+00 | 1.0e+09 | 1.0e-05 |
| ferris1 | 540 | 11 | 1862 | 550 | 2.096273e-03 | 1.0e+09 | 1.0e-06 |
| ferris2 | 1526 | 11 | 4806 | 1535 | -1.783468e+00 | 1.0e+07 | 1.0e-07 |
| goerner1 | 54 | 11 | 407 | 65 | 3.959922e-03 | 1.0e+09 | 1.0e-06 |
| goerner2 | 434 | 10 | 1546 | 444 | 6.369646e-03 | 1.0e+07 | 1.0e-06 |
| goerner3 | 95 | 9 | 440 | 104 | 9.837083e-04 | 1.0e+09 | 1.0e-04 |
| gugat1 | 8 | 7 | 89 | 15 | 1.214835e-02 | 1.0e+05 | 1.0e-05 |
| gugat2 | 7 | 7 | 59 | 14 | 3.691191e-01 | 1.0e+05 | 1.0e-05 |
| gugat4a | 2 | 2 | 36 | 4 | -4.417865e-09 | 1.0e+00 | 1.0e-05 |
| gugat4b | 63 | 9 | 392 | 72 | 5.722462e-03 | 1.0e+09 | 1.0e-04 |
| gugat4c | 10 | 9 | 119 | 19 | 1.263816e-02 | 1.0e+07 | 1.0e-05 |
| gugat4d | 19 | 6 | 112 | 25 | 5.770075e-01 | 1.0e+04 | 1.0e-05 |
| gugat4e | 377 | 12 | 1357 | 389 | 5.232385e-02 | 1.0e+08 | 1.0e-07 |
| gugat4f | 10 | 10 | 266 | 20 | 3.059348e-05 | 1.0e+09 | 1.0e-05 |
| gugat5a | 97 | 5 | 543 | 102 | 1.848377e-01 | 1.0e+03 | 1.0e-05 |
| gugat5b | 8 | 7 | 83 | 15 | 3.419458e-01 | 1.0e+05 | 1.0e-05 |
| gugat5c | 7 | 7 | 85 | 14 | 2.749656e-02 | 1.0e+05 | 1.0e-05 |
| gugat5d | 7 | 6 | 48 | 13 | 7.695452e-01 | 1.0e+04 | 1.0e-05 |
| gugat5e | 8 | 8 | 90 | 16 | 8.999488e-02 | 1.0e+06 | 1.0e-05 |
| gugat5f | 2 | 2 | 36 | 4 | 1.300741e-04 | 1.0e+00 | 1.0e-05 |
| gugat6 | 2 | 2 | 35 | 4 | 3.283219e-06 | 1.0e+00 | 1.0e-05 |
| gugat7 | 103 | 11 | 521 | 114 | 1.243658e-03 | 1.0e+07 | 1.0e-07 |
| hettich10 | 41 | 11 | 232 | 52 | 9.999299e-01 | 1.0e+07 | 1.0e-07 |
| hettich10c | 77 | 11 | 379 | 88 | 9.999368e-01 | 1.0e+07 | 1.0e-07 |
| hettich2 | 108 | 9 | 541 | 117 | 5.379449e-01 | 1.0e+06 | 1.0e-06 |
| hettich3 | 10 | 10 | 223 | 20 | 3.239686e-03 | 1.0e+09 | 1.0e-05 |
| hettich4 | 91 | 10 | 500 | 101 | 1.000001e+00 | 1.0e+07 | 1.0e-06 |
| hettich8 | 45 | 10 | 378 | 55 | 5.577070e-03 | 1.0e+07 | 1.0e-06 |
| leon1 | 64 | 13 | 524 | 77 | 4.482859e-03 | 1.0e+08 | 1.0e-08 |

| Problema | TI | IE | NP | NGP | FX | $\mu$ | $\epsilon$ |
|---|---|---|---|---|---|---|---|
| leon10 | 113 | 9 | 566 | 122 | 5.379578e-01 | 1.0e+06 | 1.0e-06 |
| leon11 | 111 | 11 | 710 | 122 | 4.737253e-02 | 1.0e+09 | 1.0e-06 |
| leon12 | 60 | 9 | 282 | 69 | -1.000012e+00 | 1.0e+06 | 1.0e-06 |
| leon13 | 89 | 9 | 439 | 98 | 2.360082e-01 | 1.0e+06 | 1.0e-06 |
| leon14 | 115 | 9 | 492 | 124 | 6.662506e-01 | 1.0e+06 | 1.0e-06 |
| leon15 | 64 | 9 | 288 | 73 | -6.666707e-01 | 1.0e+06 | 1.0e-06 |
| leon16 | 93 | 10 | 440 | 103 | 1.859083e+00 | 1.0e+07 | 1.0e-06 |
| leon18 | 567 | 9 | 1820 | 575 | -1.750001e+00 | 1.0e+06 | 1.0e-06 |
| leon19 | 1418 | 12 | 4497 | 1427 | 7.871741e-01 | 1.0e+08 | 1.0e-07 |
| leon2 | 27 | 2 | 149 | 29 | 1.762059e-05 | 1.0e+00 | 1.0e-05 |
| leon3 | 414 | 11 | 1430 | 424 | 9.335106e-04 | 1.0e+09 | 1.0e-06 |
| leon4 | 75 | 9 | 557 | 84 | 2.734029e-03 | 1.0e+06 | 1.0e-06 |
| leon5 | 35 | 9 | 390 | 44 | 1.212924e-02 | 1.0e+09 | 1.0e-04 |
| leon6 | 50 | 9 | 299 | 59 | 8.388449e-05 | 1.0e+07 | 1.0e-05 |
| leon7 | 1077 | 11 | 3458 | 1086 | 2.104617e-03 | 1.0e+09 | 1.0e-06 |
| leon8 | 91 | 10 | 547 | 101 | 5.772095e-02 | 1.0e+09 | 1.0e-05 |
| leon9 | 13 | 9 | 316 | 22 | 9.320219e-01 | 1.0e+06 | 1.0e-06 |
| liu1 | 52 | 12 | 303 | 64 | -4.653773e+00 | 1.0e+08 | 1.0e-07 |
| liu2 | 59 | 9 | 315 | 68 | -3.363750e+00 | 1.0e+06 | 1.0e-06 |
| liu3 | 145 | 10 | 878 | 155 | 1.528981e+02 | 1.0e+09 | 1.0e-05 |
| matlab1 | 22 | 6 | 334 | 28 | 0.000000e+00 | 1.0e+03 | 1.0e-06 |
| powell1 | 68 | 9 | 283 | 77 | -1.000192e+00 | 1.0e+06 | 1.0e-06 |
| priceK | 60 | 9 | 309 | 69 | -3.000093e+00 | 1.0e+06 | 1.0e-06 |
| still1 | 88 | 10 | 476 | 98 | 1.013698e+00 | 1.0e+07 | 1.0e-06 |
| watson1 | 71 | 11 | 311 | 82 | -2.769099e-01 | 1.0e+09 | 1.0e-06 |
| watson14 | 594 | 13 | 1984 | 606 | 2.200000e+00 | 1.0e+08 | 1.0e-08 |
| watson2 | 53 | 10 | 294 | 63 | 1.944740e-01 | 1.0e+06 | 1.0e-07 |
| watson3 | 69 | 10 | 382 | 79 | 5.334783e+00 | 1.0e+07 | 1.0e-06 |
| watson4a | 186 | 11 | 765 | 197 | 6.483266e-01 | 1.0e+09 | 1.0e-06 |
| watson4b | 1829 | 12 | 5746 | 1837 | 6.165812e-01 | 1.0e+08 | 1.0e-07 |
| watson4c | 1227 | 11 | 3911 | 1235 | 6.157654e-01 | 1.0e+07 | 1.0e-07 |
| watson5 | 82 | 12 | 500 | 94 | 4.300837e+00 | 1.0e+08 | 1.0e-07 |
| watson6 | 69 | 11 | 449 | 80 | 9.714333e+01 | 1.0e+09 | 1.0e-06 |
| zhou1 | 103 | 9 | 467 | 112 | 1.784178e-01 | 1.0e+06 | 1.0e-06 |

Tabela J.9: Resultados numéricos para a função de penalidade $\phi_{LA}$

| Problema | TI | IE | NP | NGP | FX | $\mu$ | $\epsilon$ |
|---|---|---|---|---|---|---|---|
| blankenship1 | 21 | 5 | 149 | 26 | 6.479107e-07 | 1.1e+00 | 1.0e-07 |
| coopeL | 277 | 116 | 2215 | 393 | 3.430458e-01 | 4.3e+04 | 1.0e-07 |
| coopeN | 355 | 112 | 2283 | 467 | -9.924213e-06 | 2.7e+04 | 1.0e-09 |
| elke10 | 45 | 6 | 2362 | 51 | 3.844845e+00 | 1.5e+00 | 1.0e-05 |
| elke1std | 309 | 119 | 3198 | 428 | 1.082560e+00 | 5.2e+04 | 1.0e-08 |
| elke2std | 623 | 97 | 3233 | 720 | 2.555807e+00 | 7.1e+03 | 1.0e-07 |
| elke3std | 328 | 87 | 2630 | 415 | 1.524053e+00 | 2.7e+03 | 1.0e-07 |
| elke4std | 647 | 119 | 3908 | 766 | 2.504990e+00 | 5.2e+04 | 1.0e-08 |
| elke5std | 637 | 80 | 3144 | 717 | 2.389430e+00 | 1.4e+03 | 1.0e-07 |
| elke6std | 1028 | 105 | 4323 | 1133 | 4.508794e+00 | 1.4e+04 | 1.0e-08 |
| elke7std | 428 | 95 | 2922 | 523 | 2.910606e+00 | 5.8e+03 | 1.0e-07 |
| elke8 | 192 | 96 | 2535 | 288 | 2.152371e+00 | 7.1e+03 | 1.0e-06 |
| elke9 | 60 | 3 | 374 | 63 | 1.049311e+01 | 1.1e+00 | 1.0e-05 |
| ferris1 | 932 | 129 | 4435 | 1059 | 2.592997e-03 | 1.4e+05 | 1.0e-08 |
| ferris2 | 2057 | 98 | 7260 | 2152 | -1.783316e+00 | 8.6e+03 | 1.0e-06 |
| goerner1 | 158 | 110 | 1961 | 268 | 4.504599e-03 | 2.4e+04 | 1.0e-07 |
| goerner2 | 406 | 103 | 2560 | 509 | 2.533245e-03 | 1.4e+04 | 1.0e-06 |
| goerner3 | | | | | | | |
| gugat1 | 109 | 105 | 1261 | 214 | 1.751769e-02 | 1.7e+04 | 1.0e-06 |
| gugat2 | 2 | 2 | 8 | 4 | 0 | 1.0e+00 | 1.0e-05 |
| gugat4a | 2 | 2 | 36 | 4 | 8.504620e-07 | 1.0e+00 | 1.0e-05 |
| gugat4b | 72 | 72 | 834 | 144 | 1.924941e-02 | 7.9e+02 | 1.0e-05 |
| gugat4c | 72 | 72 | 746 | 144 | 2.532162e-02 | 7.9e+02 | 1.0e-05 |
| gugat4d | 8 | 5 | 75 | 13 | 2.061205e+00 | 1.3e+00 | 1.0e-05 |
| gugat4e | 473 | 79 | 2693 | 552 | 4.706544e-02 | 1.3e+03 | 1.0e-07 |
| gugat4f | 98 | 98 | 1473 | 196 | 2.240762e-03 | 9.4e+03 | 1.0e-05 |
| gugat5a | 3 | 3 | 52 | 6 | 5.812754e-05 | 1.1e+00 | 1.0e-05 |
| gugat5b | 276 | 119 | 3274 | 395 | 3.096000e-01 | 5.8e+04 | 1.0e-07 |
| gugat5c | 118 | 107 | 1396 | 225 | 1.577413e-02 | 2.0e+04 | 1.0e-06 |
| gugat5d | 30 | 30 | 201 | 60 | 7.429128e-01 | 1.4e+01 | 1.0e-05 |
| gugat5e | 78 | 78 | 758 | 156 | 8.609787e-02 | 1.4e+03 | 1.0e-05 |
| gugat5f | 4 | 4 | 85 | 8 | 1.072150e-04 | 1.2e+00 | 1.0e-05 |
| gugat6 | 2 | 2 | 35 | 4 | 3.276489e-06 | 1.0e+00 | 1.0e-05 |
| gugat7 | 202 | 127 | 2217 | 329 | 1.154064e-03 | 1.1e+05 | 1.0e-08 |
| hettich10 | 236 | 110 | 1811 | 346 | 9.999126e-01 | 2.4e+04 | 1.0e-07 |
| hettich10c | 285 | 114 | 2073 | 399 | 9.999156e-01 | 3.6e+04 | 1.0e-07 |
| hettich2 | 535 | 123 | 3798 | 658 | 5.381963e-01 | 7.7e+04 | 1.0e-08 |
| hettich3 | 154 | 154 | 19611 | 308 | 0 | 2.0e+06 | 1.0e-05 |
| hettich4 | 341 | 84 | 2321 | 425 | 9.999768e-01 | 1.9e+03 | 1.0e-08 |
| hettich8 | 144 | 123 | 1951 | 267 | 5.698045e-03 | 8.4e+04 | 1.0e-07 |
| leon1 | 174 | 115 | 1919 | 289 | 4.480063e-03 | 3.9e+04 | 1.0e-07 |

| Problema | TI | IE | NP | NGP | FX | $\mu$ | $\epsilon$ |
|---|---|---|---|---|---|---|---|
| leon10 | 621 | 121 | 3956 | 742 | 5.381805e-01 | 7.0e+04 | 1.0e-07 |
| leon11 | 557 | 168 | 17978 | 725 | 0 | 6.8e+06 | 1.0e-06 |
| leon12 | 253 | 117 | 1961 | 370 | -1.000055e+00 | 4.8e+04 | 1.0e-07 |
| leon13 | 586 | 117 | 3243 | 703 | 2.359919e-01 | 4.3e+04 | 1.0e-08 |
| leon14 | 577 | 109 | 3149 | 686 | 6.662494e-01 | 2.2e+04 | 1.0e-07 |
| leon15 | 296 | 116 | 2010 | 412 | -6.667208e-01 | 4.3e+04 | 1.0e-07 |
| leon16 | 354 | 108 | 2404 | 462 | 1.858847e+00 | 2.0e+04 | 1.0e-07 |
| leon18 | 1584 | 45 | 5132 | 1627 | -1.751100e+00 | 6.0e+01 | 1.0e-05 |
| leon19 | 1533 | 108 | 5819 | 1638 | 7.860410e-01 | 2.2e+04 | 1.0e-06 |
| leon2 | 28 | 2 | 157 | 30 | 2.163985e-05 | 1.0e+00 | 1.0e-05 |
| leon3 | 581 | 120 | 3375 | 700 | 5.135557e-04 | 7.0e+04 | 1.0e-06 |
| leon4 | 163 | 115 | 2265 | 278 | 2.976824e-03 | 3.9e+04 | 1.0e-07 |
| leon5 | 113 | 70 | 1738 | 183 | 1.408370e-02 | 6.5e+02 | 1.0e-05 |
| leon6 | 163 | 125 | 2283 | 288 | 1.976771e-04 | 1.0e+05 | 1.0e-07 |
| leon7 | 1070 | 101 | 4370 | 1169 | 1.871097e-03 | 1.3e+04 | 1.0e-05 |
| leon8 | 188 | 86 | 2032 | 274 | 6.224114e-02 | 2.7e+03 | 1.0e-06 |
| leon9 | 275 | 119 | 16550 | 394 | 0 | 6.3e+04 | 1.0e-06 |
| liu1 | 232 | 108 | 1972 | 340 | -4.653893e+00 | 2.0e+04 | 1.0e-07 |
| liu2 | 353 | 102 | 2217 | 455 | -3.363746e+00 | 1.1e+04 | 1.0e-07 |
| liu3 | 337 | 96 | 4154 | 433 | 1.552691e+02 | 7.1e+03 | 1.0e-06 |
| matlab1 | 13 | 3 | 101 | 16 | 0.000000e+00 | 1.0e+00 | 1.0e-06 |
| powell1 | 282 | 108 | 2057 | 390 | -1.000206e+00 | 2.0e+04 | 1.0e-07 |
| priceK | 258 | 117 | 2075 | 375 | -3.000106e+00 | 4.8e+04 | 1.0e-07 |
| still1 | 366 | 103 | 2691 | 469 | 9.999246e-01 | 1.4e+04 | 1.0e-06 |
| watson1 | 422 | 125 | 2046 | 547 | -2.617853e-01 | 9.3e+04 | 1.0e-09 |
| watson14 | | | | | | | |
| watson2 | 327 | 114 | 2119 | 441 | 1.944138e-01 | 3.6e+04 | 1.0e-07 |
| watson3 | 294 | 104 | 2542 | 398 | 5.334677e+00 | 1.4e+04 | 1.0e-07 |
| watson4a | 940 | 120 | 4496 | 1060 | 6.490393e-01 | 6.3e+04 | 1.0e-07 |
| watson4b | 648 | 116 | 3539 | 764 | 6.167197e-01 | 4.3e+04 | 1.0e-07 |
| watson4c | 940 | 125 | 4553 | 1063 | 6.157989e-01 | 1.0e+05 | 1.0e-07 |
| watson5 | 427 | 122 | 3199 | 549 | 4.300778e+00 | 7.7e+04 | 1.0e-07 |
| watson6 | | | | | | | |
| zhou1 | 503 | 117 | 3100 | 620 | 1.783219e-01 | 4.3e+04 | 1.0e-08 |

Tabela J.10: Resultados numéricos para a função de penalidade $\phi_E$ com $\mu_f = 1.1$

| Problema | TI | IE | NP | NGP | FX | $\mu$ | $\epsilon$ |
|---|---|---|---|---|---|---|---|
| blankenship1 | 21 | 5 | 149 | 26 | 6.479190e-07 | 2.0e+00 | 1.0e-07 |
| coopeL | 95 | 22 | 597 | 117 | 3.430426e-01 | 2.6e+05 | 1.0e-07 |
| coopeN | 83 | 23 | 503 | 106 | -6.775513e-06 | 2.6e+05 | 1.0e-09 |
| elke10 | 35 | 7 | 6290 | 42 | NaN | 3.2e+01 | 1.0e-05 |
| elke1std | 107 | 23 | 790 | 130 | 1.085166e+00 | 2.6e+05 | 1.0e-08 |
| elke3std | | | | | | | |
| elke2std | 254 | 26 | 1243 | 280 | 2.555690e+00 | 2.1e+06 | 1.0e-08 |
| elke4std | 236 | 20 | 1058 | 256 | 2.512311e+00 | 3.3e+04 | 1.0e-08 |
| elke5std | 232 | 20 | 1072 | 252 | 2.421656e+00 | 6.6e+04 | 1.0e-07 |
| elke7std | | | | | | | |
| elke6std | 366 | 19 | 1329 | 385 | 4.524740e+00 | 3.3e+04 | 1.0e-07 |
| elke8 | 110 | 21 | 935 | 131 | 2.152434e+00 | 2.6e+05 | 1.0e-06 |
| elke9 | 60 | 3 | 374 | 63 | 1.047936e+01 | 2.0e+00 | 1.0e-05 |
| ferris1 | 428 | 25 | 3460 | 452 | NaN | 8.4e+06 | 1.0e-05 |
| ferris2 | 2368 | 21 | 7435 | 2386 | -1.784681e+00 | 2.6e+05 | 1.0e-06 |
| goerner1 | 70 | 27 | 2608 | 97 | 3.043214e-03 | 1.3e+08 | 1.0e-04 |
| goerner2 | 489 | 27 | 3940 | 516 | NaN | 3.4e+07 | 1.0e-05 |
| goerner3 | | | | | | | |
| gugat1 | 12 | 5 | 2480 | 17 | NaN | 8.0e+00 | 1.0e-05 |
| gugat2 | 2 | 2 | 8 | 4 | NaN | 1.0e+00 | 1.0e-05 |
| gugat4a | 3 | 3 | 52 | 6 | 4.181507e-05 | 2.0e+00 | 1.0e-05 |
| gugat4b | 154 | 23 | 2871 | 177 | NaN | 2.1e+06 | 1.0e-05 |
| gugat4c | 19 | 18 | 217 | 37 | 1.199640e-02 | 6.6e+04 | 1.0e-05 |
| gugat4d | 39 | 10 | 192 | 49 | 7.529886e-01 | 2.6e+02 | 1.0e-05 |
| gugat4e | 130 | 22 | 693 | 152 | 4.979380e-02 | 2.6e+05 | 1.0e-07 |
| gugat4f | 18 | 18 | 415 | 36 | 1.939779e-04 | 6.6e+04 | 1.0e-05 |
| gugat5a | 2 | 2 | 36 | 4 | 5.137838e-05 | 1.0e+00 | 1.0e-05 |
| gugat5b | 137 | 21 | 1070 | 158 | 3.084493e-01 | 1.3e+05 | 1.0e-07 |
| gugat5c | 28 | 19 | 280 | 47 | 1.598847e-02 | 6.6e+04 | 1.0e-06 |
| gugat5d | 190 | 21 | 2927 | 211 | NaN | 5.2e+05 | 1.0e-05 |
| gugat5e | 16 | 16 | 169 | 32 | 8.736369e-02 | 1.6e+04 | 1.0e-05 |
| gugat5f | 3 | 3 | 51 | 6 | 1.423091e-04 | 2.0e+00 | 1.0e-05 |
| gugat6 | 2 | 2 | 35 | 4 | 3.276490e-06 | 1.0e+00 | 1.0e-05 |
| gugat7 | 122 | 23 | 709 | 145 | 1.100740e-03 | 5.2e+05 | 1.0e-07 |
| hettich10 | 64 | 22 | 398 | 86 | 9.999181e-01 | 2.6e+05 | 1.0e-07 |
| hettich10c | 91 | 22 | 530 | 113 | 9.999121e-01 | 2.6e+05 | 1.0e-07 |
| hettich2 | 207 | 30 | 1694 | 237 | 5.380420e-01 | 1.3e+08 | 1.0e-07 |
| hettich3 | 26 | 26 | 2119 | 52 | NaN | 1.7e+07 | 1.0e-05 |
| hettich4 | 123 | 19 | 667 | 142 | 9.999807e-01 | 1.6e+04 | 1.0e-08 |
| hettich8 | 56 | 24 | 632 | 80 | 5.695735e-03 | 1.0e+06 | 1.0e-07 |
| leon1 | 90 | 24 | 806 | 114 | 4.578112e-03 | 5.2e+05 | 1.0e-08 |

| Problema | TI | IE | NP | NGP | FX | $\mu$ | $\epsilon$ |
|---|---|---|---|---|---|---|---|
| leon10 | 233 | 30 | 1848 | 263 | 5.380234e-01 | 1.3e+08 | 1.0e-07 |
| leon11 | 168 | 29 | 3329 | 197 | 4.708528e-02 | 1.3e+08 | 1.0e-06 |
| leon12 | 69 | 22 | 433 | 91 | -1.000056e+00 | 2.6e+05 | 1.0e-07 |
| leon13 | 169 | 23 | 796 | 192 | 2.360078e-01 | 2.6e+05 | 1.0e-08 |
| leon14 | 180 | 23 | 889 | 203 | 6.664130e-01 | 2.6e+05 | 1.0e-08 |
| leon15 | 94 | 23 | 533 | 117 | -6.666776e-01 | 5.2e+05 | 1.0e-07 |
| leon16 | 147 | 23 | 725 | 170 | 1.858914e+00 | 5.2e+05 | 1.0e-07 |
| leon18 | 847 | 13 | 2611 | 858 | -1.750272e+00 | 1.0e+03 | 1.0e-06 |
| leon19 | 1931 | 26 | 6230 | 1954 | 7.864351e-01 | 2.1e+06 | 1.0e-08 |
| leon2 | 28 | 2 | 157 | 30 | 2.163985e-05 | 1.0e+00 | 1.0e-05 |
| leon3 | 508 | 25 | 1939 | 532 | 6.901619e-04 | 2.1e+06 | 1.0e-07 |
| leon4 | 83 | 29 | 1475 | 112 | 3.020168e-03 | 1.3e+08 | 1.0e-06 |
| leon5 | 47 | 22 | 2449 | 69 | NaN | 1.0e+06 | 1.0e-05 |
| leon6 | 60 | 22 | 511 | 82 | 9.366022e-05 | 1.0e+06 | 1.0e-05 |
| leon7 | 968 | 24 | 3268 | 990 | 2.615165e-03 | 1.0e+06 | 1.0e-07 |
| leon8 | 122 | 25 | 2792 | 147 | NaN | 8.4e+06 | 1.0e-05 |
| leon9 | 178 | 22 | 2996 | 200 | NaN | 5.2e+05 | 1.0e-06 |
| liu1 | 84 | 22 | 486 | 106 | -4.653850e+00 | 2.6e+05 | 1.0e-07 |
| liu2 | 108 | 21 | 593 | 129 | -3.363816e+00 | 1.3e+05 | 1.0e-07 |
| liu3 | 156 | 26 | 3153 | 182 | NaN | 1.7e+07 | 1.0e-05 |
| matlab1 | 13 | 3 | 101 | 16 | 0.000000e+00 | 1.0e+00 | 1.0e-06 |
| powell1 | 109 | 22 | 562 | 131 | -1.000201e+00 | 2.6e+05 | 1.0e-07 |
| priceK | 80 | 22 | 491 | 102 | -3.000112e+00 | 2.6e+05 | 1.0e-07 |
| still1 | 152 | 24 | 905 | 176 | 9.999723e-01 | 5.2e+05 | 1.0e-08 |
| watson1 | 138 | 25 | 627 | 163 | -2.568208e-01 | 1.0e+06 | 1.0e-09 |
| watson14 | | | | | | | |
| watson2 | 93 | 22 | 517 | 115 | 1.944146e-01 | 2.6e+05 | 1.0e-07 |
| watson3 | 134 | 22 | 707 | 156 | 5.334684e+00 | 2.6e+05 | 1.0e-07 |
| watson4a | 327 | 24 | 1402 | 351 | 6.490589e-01 | 5.2e+05 | 1.0e-08 |
| watson4b | 1715 | 26 | 5667 | 1738 | 6.166178e-01 | 4.2e+06 | 1.0e-07 |
| watson4c | 838 | 24 | 2891 | 860 | 6.157830e-01 | 1.0e+06 | 1.0e-07 |
| watson5 | 136 | 25 | 891 | 161 | 4.301396e+00 | 1.0e+06 | 1.0e-08 |
| watson6 | | | | | | | |
| zhou1 | 170 | 24 | 866 | 194 | 1.783655e-01 | 5.2e+05 | 1.0e-08 |

Tabela J.11: Resultados numéricos para a função de penalidade $\phi_E$ com $\mu_f = 2$

## J.4 Método de pontos interiores

A Tabela J.12 apresenta os resultados numéricos para a função mérito $l_2$ e a Tabela J.13 apresenta os resultados numéricos para a função mérito Lagrangeana aumentada, ambas com ordenação dual. As tabelas J.14 e J.15 apresentam os resultados numéricos para a ordenação primal, respectivamente com as funções mérito $l_2$ e Lagrangeana aumentada. Em todas as tabelas: "Problema" é o nome do problema na base de dados do SIPAMPL, "iter" é o número de iterações, "nmer" é o número de cálculos da função mérito, "fx" é o valor da função objectivo na solução encontrada, "$||\rho||_2$" é a não admissibilidade primal e "$||\sigma||_2$" é a não admissibilidade dual, "$\mu$" é o último $\mu$ usado e "$\epsilon$" é o último $\epsilon$ usado.

Foram usadas as opções por omissão para o algoritmo, a excepção de `int_eps=1e-2` e `int_epslimit=1e-4`.

| Problema | iter | nmer | fx | $\|\|\rho\|\|_2$ | $\|\|\sigma\|\|_2$ | $\mu$ | $\epsilon$ |
|---|---|---|---|---|---|---|---|
| blankenship1 | 66 | 336 | 5.019180e-01 | 1.0e-04 | 3.9e-05 | 5.5e+00 | 1.0e-03 |
| coopeL | 34 | 160 | 1.999545e+00 | 1.0e-04 | 1.3e-05 | 4.2e+01 | 1.0e-03 |
| coopeN | 44 | 136 | 3.810186e-01 | 2.0e-06 | 3.3e-04 | 5.0e-01 | 1.0e-03 |
| elke10 | 401 | 2649 | 2.200346e+00 | 4.8e-05 | 1.0e-01 | 3.6e-11 | 1.0e-02 |
| elke1std | 401 | 1308 | 1.895443e+02 | 2.5e-04 | 1.1e+04 | 3.8e+02 | 1.0e-02 |
| elke2std | 401 | 455 | 5.097468e+00 | 2.3e-04 | 1.4e+02 | 3.8e+02 | 1.0e-02 |
| elke3std | 401 | 736 | 1.858301e+01 | 2.1e-04 | 3.8e+02 | 4.0e+04 | 1.0e-02 |
| elke4std | 401 | 876 | 5.776004e+00 | 2.3e-04 | 1.3e-01 | 2.6e+00 | 1.0e-02 |
| elke5std | 401 | 786 | 3.809850e+00 | 2.3e-04 | 3.8e-01 | 1.7e+00 | 1.0e-02 |
| elke6std | 401 | 722 | 1.496566e+01 | 2.2e-04 | 4.9e+00 | 9.1e+01 | 1.0e-02 |
| elke7std | 401 | 1595 | 5.354616e+00 | 3.3e-04 | 3.4e-01 | 1.6e-13 | 1.0e-02 |
| elke8 | 401 | 1532 | 2.160138e+00 | 5.1e-05 | 2.9e-01 | 1.9e-03 | 1.0e-02 |
| elke9 | 401 | 2076 | 2.194054e+00 | 2.2e+04 | 1.7e+05 | 2.4e-12 | 1.0e-03 |
| ferris1 | 276 | 986 | 2.512743e-01 | 2.8e-06 | 6.9e-04 | 1.4e-01 | 1.0e-03 |
| ferris2 | 65 | 254 | -1.536274e+00 | 2.0e-06 | 8.8e-04 | 2.7e-01 | 1.0e-03 |
| goerner1 | 401 | 965 | 5.432383e+00 | 1.9e-05 | 3.6e-03 | 1.1e+01 | 1.0e-02 |
| goerner2 | 62 | 197 | 1.877071e-01 | 1.4e-04 | 8.3e-04 | 1.0e+00 | 1.0e-03 |
| goerner3 | 4 | 6 | NaN | NaN | NaN | NaN | 1.0e-03 |
| gugat1 | 53 | 165 | 3.404459e-01 | 2.0e-04 | 1.7e-05 | 1.3e+00 | 1.0e-03 |
| gugat2 | 78 | 378 | 2.629821e+01 | 2.0e-04 | 6.8e-05 | 2.6e+02 | 1.0e-03 |
| gugat4a | 42 | 222 | 5.148651e-01 | 2.0e-04 | 5.9e-05 | 2.4e+00 | 1.0e-03 |
| gugat4b | 72 | 228 | 1.500066e+00 | 2.9e-06 | 8.7e-04 | 4.1e-01 | 1.0e-03 |
| gugat4c | 260 | 1012 | 1.295970e-02 | 6.5e-06 | 4.2e-04 | 1.2e-02 | 1.0e-05 |
| gugat4d | 401 | 2707 | 7.087924e-01 | 1.9e-03 | 9.9e-01 | 2.4e-14 | 1.0e-03 |
| gugat4e | 54 | 297 | 5.017053e-01 | 2.0e-04 | 6.7e-04 | 1.4e+00 | 1.0e-03 |
| gugat4f | 401 | 499 | 1.991755e+00 | 1.2e-04 | 9.6e-03 | 6.9e+00 | 1.0e-02 |
| gugat5a | 401 | 881 | 4.832969e-02 | 3.0e-05 | 2.2e-02 | 2.2e-03 | 1.0e-03 |
| gugat5b | 401 | 623 | 4.825172e+01 | 3.2e-05 | 1.3e-03 | 4.0e+01 | 1.0e-02 |
| gugat5c | 49 | 268 | 2.956084e-01 | 2.0e-05 | 1.7e-04 | 9.6e+00 | 1.0e-04 |
| gugat5d | 56 | 427 | 9.642006e-01 | 2.1e-07 | 5.0e-04 | 3.5e-06 | 1.0e-03 |

| Problema | iter | nmer | fx | $\|\|\rho\|\|_2$ | $\|\|\sigma\|\|_2$ | $\mu$ | $\epsilon$ |
|---|---|---|---|---|---|---|---|
| gugat5e | 73 | 554 | 5.008056e-01 | 2.1e-05 | 2.7e-04 | 2.5e-01 | 1.0e-03 |
| gugat5f | 53 | 167 | 7.499832e-01 | 2.1e-05 | 5.7e-04 | 3.7e-01 | 1.0e-03 |
| gugat6 | 32 | 98 | 3.397611e-01 | 2.0e-05 | 7.9e-06 | 1.3e+01 | 1.0e-04 |
| gugat7 | 32 | 90 | 3.477914e-01 | 2.0e-05 | 9.0e-05 | 1.4e+01 | 1.0e-04 |
| hettich10 | 401 | 499 | 1.289924e+00 | 2.1e-06 | 5.5e+00 | 1.5e+00 | 1.0e-03 |
| hettich10c | 40 | 127 | 1.289963e+00 | 1.4e-04 | 2.5e-08 | 3.9e+01 | 1.0e-03 |
| hettich2 | 50 | 166 | 7.830054e-01 | 1.4e-04 | 8.8e-04 | 1.9e+00 | 1.0e-03 |
| hettich3 | 324 | 1078 | 1.874087e-01 | 2.8e-06 | 1.2e-04 | 1.0e-01 | 1.0e-03 |
| hettich4 | 40 | 150 | 9.995821e-01 | 2.2e-11 | 6.5e-04 | 1.9e-07 | 1.0e-05 |
| hettich8 | 39 | 114 | 2.509253e-01 | 1.4e-04 | 3.6e-04 | 1.4e+00 | 1.0e-03 |
| leon1 | 72 | 677 | 2.499440e-01 | 1.4e-04 | 5.4e-04 | 1.4e+00 | 1.0e-03 |
| leon10 | 177 | 923 | 7.829833e-01 | 1.4e-04 | 7.2e-04 | 1.9e+00 | 1.0e-03 |
| leon11 | 175 | 572 | 5.031648e-01 | 2.8e-06 | 2.0e-05 | 2.7e-01 | 1.0e-03 |
| leon12 | 401 | 479 | -1.163466e-01 | 3.2e-04 | 5.3e-01 | 9.9e-01 | 1.0e-02 |
| leon13 | 114 | 824 | 2.360888e-01 | 2.3e-06 | 1.7e-05 | 3.0e-05 | 1.0e-05 |
| leon14 | 85 | 235 | 1.182596e+00 | 2.0e-06 | 1.9e-04 | 7.1e-01 | 1.0e-03 |
| leon15 | 401 | 1025 | -6.571396e-01 | 2.3e-05 | 2.5e-02 | 2.7e-03 | 1.0e-03 |
| leon16 | 32 | 126 | 1.986721e+00 | 1.0e-04 | 8.3e-04 | 2.5e+00 | 1.0e-03 |
| leon18 | 49 | 138 | -1.764857e+00 | 1.9e-04 | 2.4e-06 | 1.6e-03 | 1.0e-03 |
| leon19 | 52 | 159 | 1.035388e+00 | 1.0e-04 | 5.3e-04 | 2.7e+00 | 1.0e-03 |
| leon2 | 124 | 433 | 4.999546e-01 | 2.8e-06 | 8.3e-04 | 2.7e-01 | 1.0e-03 |
| leon3 | 99 | 322 | 2.501498e-01 | 2.8e-06 | 2.1e-04 | 1.4e-01 | 1.0e-03 |
| leon4 | 64 | 216 | 4.999664e-01 | 1.6e-05 | 4.5e-04 | 5.0e-01 | 1.0e-03 |
| leon5 | 401 | 1610 | 5.438738e+01 | 6.6e-04 | 1.0e+00 | 1.2e-16 | 1.0e-02 |
| leon6 | 38 | 135 | 2.499225e-01 | 1.4e-04 | 2.0e-04 | 1.4e+00 | 1.0e-03 |
| leon7 | 401 | 1951 | 5.752130e-01 | 1.7e-04 | 9.4e-01 | 1.2e-10 | 1.0e-03 |
| leon8 | 401 | 1501 | 9.671140e-01 | 1.6e-03 | 9.9e-01 | 3.9e-14 | 1.0e-02 |
| leon9 | 401 | 1401 | 3.076949e-01 | 1.1e-03 | 3.9e+01 | 1.5e-11 | 1.0e-02 |
| liu1 | 123 | 370 | -4.654982e+00 | 1.5e-05 | 2.6e-04 | 1.2e-04 | 1.0e-04 |
| liu2 | 401 | 500 | -2.519175e+00 | 2.0e-04 | 6.0e-02 | 2.4e-01 | 1.0e-02 |
| liu3 | 401 | 1070 | 1.598071e+02 | 2.8e-04 | 1.5e-02 | 1.4e+00 | 1.0e-02 |
| matlab1 | 401 | 1037 | 2.805521e+01 | 9.7e-04 | 4.1e+00 | 5.1e+00 | 1.0e-02 |
| powell1 | 41 | 121 | -9.768454e-01 | 1.1e-05 | 4.5e-06 | 8.1e-02 | 1.0e-03 |
| priceK | 28 | 91 | -1.465937e+00 | 1.1e-05 | 6.1e-04 | 5.6e+00 | 1.0e-03 |
| still1 | 401 | 1410 | 9.447308e-01 | 5.2e-03 | 2.4e+01 | 1.3e-05 | 1.0e-03 |
| watson1 | 401 | 476 | -2.539101e-01 | 3.5e-04 | 2.1e-01 | 3.9e+00 | 1.0e-02 |
| watson14 | 89 | 395 | 2.219764e+00 | 2.0e-06 | 1.0e-06 | 8.6e-02 | 1.0e-03 |
| watson2 | 401 | 1148 | 2.435838e+00 | 4.9e-07 | 2.4e+00 | 7.3e-07 | 1.0e-03 |
| watson3 | 62 | 190 | 5.332102e+00 | 1.9e-05 | 2.0e-04 | 2.9e-04 | 1.0e-04 |
| watson4a | 401 | 1256 | 6.527233e-01 | 1.3e-05 | 3.3e-01 | 1.3e-12 | 1.0e-03 |
| watson4b | 66 | 205 | 8.655301e-01 | 2.0e-06 | 6.7e-04 | 2.7e-01 | 1.0e-03 |
| watson4c | 50 | 278 | 8.655419e-01 | 1.0e-04 | 2.4e-04 | 2.7e+00 | 1.0e-03 |
| watson5 | 401 | 1110 | 5.027390e+00 | 2.1e-04 | 8.5e-03 | 1.4e-01 | 1.0e-02 |
| watson6 | 401 | 1230 | 1.363690e+02 | 2.0e+00 | 1.5e+02 | 8.3e-28 | 1.0e-02 |

| Problema | iter | nmer | fx | $\|\|\rho\|\|_2$ | $\|\|\sigma\|\|_2$ | $\mu$ | $\epsilon$ |
|---|---|---|---|---|---|---|---|
| zhou1 | 42 | 123 | 3.582207e-01 | 1.0e-04 | 6.2e-04 | 2.3e+00 | 1.0e-03 |

Tabela J.12: Resultados numéricos para a função mérito $l_2$ e ordenação dual

| Problema | iter | nmer | fx | $\|\|\rho\|\|_2$ | $\|\|\sigma\|\|_2$ | $\mu$ | $\epsilon$ |
|---|---|---|---|---|---|---|---|
| blankenship1 | 50 | 148 | 4.999724e-03 | 1.7e-10 | 2.6e-04 | 1.1e-07 | 1.0e-05 |
| coopeL | 47 | 144 | 3.433274e-01 | 5.7e-11 | 1.7e-04 | 7.8e-10 | 1.0e-05 |
| coopeN | 36 | 96 | 3.812033e-01 | 1.0e-04 | 4.3e-04 | 5.0e+00 | 1.0e-03 |
| elke10 | 401 | 2509 | 6.170781e+01 | 4.0e+05 | 2.0e+09 | 2.6e+05 | 1.0e-03 |
| elke1std | 401 | 2069 | 1.090713e+00 | 1.2e-05 | 2.1e-01 | 2.4e-14 | 1.0e-03 |
| elke2std | 401 | 4604 | 1.466359e+01 | 1.0e-02 | 4.2e-01 | 1.8e-06 | 1.0e-03 |
| elke3std | 401 | 4018 | 7.396301e+01 | 4.5e-02 | 3.6e-01 | 3.5e-07 | 1.0e-02 |
| elke4std | 401 | 2383 | 5.326445e+01 | 1.6e-04 | 3.6e-01 | 1.7e-07 | 1.0e-03 |
| elke5std | 401 | 1517 | 2.120420e+01 | 2.4e-06 | 1.7e+01 | 7.1e-03 | 1.0e-03 |
| elke6std | 401 | 2180 | 2.323130e+01 | 2.1e+10 | 3.9e+13 | 8.3e-04 | 1.0e-03 |
| elke7std | 401 | 2774 | 9.684906e+00 | 3.9e+00 | 3.8e-01 | 1.1e-09 | 1.0e-02 |
| elke8 | 401 | 3005 | 3.274265e+02 | 1.8e-08 | 3.5e-01 | 1.2e+00 | 1.0e-04 |
| elke9 | 401 | 2930 | 1.901740e+01 | 1.2e-02 | 4.8e+00 | 1.1e-11 | 1.0e-03 |
| ferris1 | 224 | 841 | 2.539537e-03 | 1.6e-12 | 5.2e-04 | 8.9e-11 | 1.0e-05 |
| ferris2 | 325 | 1253 | -1.783497e+00 | 5.5e-13 | 9.7e-04 | 1.9e-12 | 1.0e-05 |
| goerner1 | 81 | 270 | 4.835923e-03 | 2.4e-13 | 7.4e-04 | 5.5e-13 | 1.0e-05 |
| goerner2 | 130 | 419 | 1.886592e-02 | 1.4e-04 | 6.3e-05 | 9.4e-01 | 1.0e-04 |
| goerner3 | 4 | 6 | NaN | NaN | NaN | NaN | 1.0e-03 |
| gugat1 | 350 | 2110 | 1.325057e-02 | 2.1e-06 | 8.8e-04 | 4.6e-05 | 1.0e-05 |
| gugat2 | 174 | 886 | 2.188760e+01 | 2.0e-04 | 3.7e-04 | 2.6e+02 | 1.0e-03 |
| gugat4a | 137 | 478 | 5.009468e-02 | 2.8e-07 | 5.1e-04 | 1.4e-02 | 1.0e-04 |
| gugat4b | 68 | 274 | 1.647362e+00 | 4.8e-05 | 4.1e-04 | 1.2e+02 | 1.0e-05 |
| gugat4c | 134 | 465 | 1.353470e-02 | 3.8e-10 | 8.3e-04 | 9.4e-14 | 1.0e-05 |
| gugat4d | 112 | 550 | 9.674698e-01 | 2.0e-04 | 6.3e-04 | 4.1e+04 | 1.0e-04 |
| gugat4e | 401 | 2174 | 4.737247e-02 | 1.5e-04 | 3.6e+00 | 3.5e-09 | 1.0e-04 |
| gugat4f | 56 | 166 | 7.499745e-02 | 2.0e-04 | 5.5e-05 | 1.9e+00 | 1.0e-04 |
| gugat5a | 53 | 160 | 5.000696e-03 | 1.8e-11 | 9.6e-04 | 2.3e-08 | 1.0e-05 |
| gugat5b | 24 | 63 | 1.672767e+00 | 4.8e-05 | 6.3e-04 | 1.2e+02 | 1.0e-05 |
| gugat5c | 401 | 2103 | 1.327177e-02 | 2.3e-06 | 2.1e-02 | 4.2e-05 | 1.0e-04 |
| gugat5d | 161 | 571 | 9.959455e-01 | 2.0e-04 | 7.0e-04 | 4.5e+00 | 1.0e-04 |
| gugat5e | 197 | 623 | 5.994241e-02 | 1.6e-09 | 4.3e-04 | 2.4e-13 | 1.0e-05 |
| gugat5f | 180 | 639 | 7.701709e-03 | 5.8e-11 | 7.6e-04 | 3.0e-10 | 1.0e-05 |
| gugat6 | 27 | 78 | 2.791129e-01 | 5.2e-07 | 6.9e-04 | 1.1e+00 | 1.0e-05 |
| gugat7 | 95 | 325 | 2.715103e-03 | 2.5e-13 | 2.2e-04 | 6.0e-13 | 1.0e-05 |
| hettich10 | 57 | 208 | 1.000033e+00 | 1.6e-11 | 8.1e-06 | 1.9e-10 | 1.0e-05 |
| hettich10c | 58 | 198 | 1.000033e+00 | 1.8e-13 | 5.9e-06 | 9.5e-10 | 1.0e-05 |

| Problema | iter | nmer | fx | $\|\|\rho\|\|_2$ | $\|\|\sigma\|\|_2$ | $\mu$ | $\epsilon$ |
|---|---|---|---|---|---|---|---|
| hettich2 | 53 | 173 | 5.439317e-01 | 1.5e-04 | 1.1e-04 | 5.7e-01 | 1.0e-04 |
| hettich3 | 131 | 550 | 3.911570e-03 | 2.2e-14 | 8.4e-04 | 6.8e-12 | 1.0e-05 |
| hettich4 | 25 | 66 | 9.995804e-01 | 9.6e-14 | 2.6e-05 | 2.4e-08 | 1.0e-05 |
| hettich8 | 71 | 238 | 7.154622e-03 | 1.1e-11 | 5.6e-04 | 1.1e-12 | 1.0e-05 |
| leon1 | 67 | 230 | 5.018197e-03 | 2.7e-12 | 1.9e-04 | 5.3e-13 | 1.0e-05 |
| leon10 | 100 | 357 | 5.380129e-01 | 1.4e-04 | 9.1e-04 | 1.4e-03 | 1.0e-05 |
| leon11 | 126 | 478 | 4.840022e-02 | 2.5e-15 | 7.1e-05 | 6.0e-12 | 1.0e-05 |
| leon12 | 31 | 86 | -6.091416e-01 | 1.2e-05 | 2.3e-05 | 1.0e+00 | 1.0e-03 |
| leon13 | 35 | 100 | 2.361105e-01 | 3.6e-11 | 2.0e-04 | 1.5e-08 | 1.0e-05 |
| leon14 | 60 | 178 | 6.667646e-01 | 2.2e-11 | 4.2e-04 | 3.7e-08 | 1.0e-05 |
| leon15 | 77 | 392 | -6.664801e-01 | 1.0e-04 | 1.5e-04 | 4.0e-03 | 1.0e-05 |
| leon16 | 40 | 127 | 1.857584e+00 | 1.5e-05 | 2.7e-05 | 3.5e-04 | 1.0e-04 |
| leon18 | 54 | 147 | -1.751575e+00 | 1.1e-04 | 2.2e-08 | 1.6e-03 | 1.0e-04 |
| leon19 | 136 | 467 | 7.879396e-01 | 7.4e-11 | 5.0e-04 | 2.5e-12 | 1.0e-05 |
| leon2 | 94 | 331 | 4.999501e-02 | 1.6e-06 | 5.0e-05 | 5.0e-02 | 1.0e-04 |
| leon3 | 110 | 375 | 2.539490e-03 | 5.5e-14 | 3.0e-04 | 2.3e-09 | 1.0e-05 |
| leon4 | 168 | 734 | 5.498604e-03 | 5.5e-13 | 5.3e-04 | 2.3e-11 | 1.0e-05 |
| leon5 | 401 | 2243 | 1.521845e-02 | 5.4e-06 | 8.5e-01 | 3.7e-05 | 1.0e-04 |
| leon6 | 67 | 210 | 2.502747e-03 | 1.8e-12 | 1.4e-04 | 6.8e-10 | 1.0e-05 |
| leon7 | 102 | 327 | 2.513515e-02 | 1.4e-04 | 5.1e-04 | 1.3e+00 | 1.0e-04 |
| leon8 | 401 | 1570 | 5.508702e-02 | 1.3e-14 | 1.7e-02 | 8.5e-13 | 1.0e-04 |
| leon9 | 401 | 1301 | 1.635532e+00 | 3.4e-03 | 9.9e+02 | 2.0e+02 | 1.0e-03 |
| liu1 | 42 | 131 | -4.654982e+00 | 1.1e-04 | 2.6e-04 | 1.4e-03 | 1.0e-04 |
| liu2 | 36 | 102 | -2.505259e+00 | 1.1e-05 | 1.1e-04 | 2.4e+00 | 1.0e-03 |
| liu3 | 401 | 3313 | 1.538693e+02 | 7.4e-06 | 4.3e+01 | 2.7e-03 | 1.0e-04 |
| matlab1 | 401 | 2185 | 2.111817e-03 | 6.1e-02 | 9.7e-02 | 3.7e-08 | 1.0e-02 |
| powell1 | 50 | 151 | -1.000190e+00 | 1.1e-09 | 4.0e-04 | 1.0e-07 | 1.0e-04 |
| priceK | 29 | 83 | -1.174761e+00 | 1.0e-04 | 1.6e-06 | 4.1e+01 | 1.0e-03 |
| still1 | 100 | 355 | 1.000050e+00 | 2.5e-12 | 2.4e-04 | 4.7e-09 | 1.0e-05 |
| watson1 | 59 | 159 | -2.787140e-01 | 1.4e-05 | 7.4e-04 | 6.6e-04 | 1.0e-04 |
| watson14 | 43 | 124 | 2.197067e+00 | 1.0e-04 | 3.8e-15 | 7.0e-03 | 1.0e-05 |
| watson2 | 45 | 149 | 2.430537e+00 | 3.5e-11 | 7.4e-04 | 2.1e-08 | 1.0e-05 |
| watson3 | 51 | 157 | 5.334417e+00 | 2.2e-14 | 9.9e-05 | 2.7e-09 | 1.0e-05 |
| watson4a | 263 | 842 | 6.491129e-01 | 3.8e-12 | 9.2e-04 | 2.3e-13 | 1.0e-05 |
| watson4b | 78 | 246 | 6.183869e-01 | 6.0e-12 | 3.9e-04 | 2.3e-12 | 1.0e-05 |
| watson4c | 193 | 723 | 6.181483e-01 | 5.4e-12 | 7.8e-04 | 2.5e-12 | 1.0e-05 |
| watson5 | 101 | 364 | 4.301433e+00 | 1.2e-12 | 2.4e-04 | 1.5e-09 | 1.0e-05 |
| watson6 | 401 | 1227 | 9.488534e+01 | 1.1e-01 | 1.5e+01 | 8.7e-13 | 1.0e-02 |
| zhou1 | 63 | 193 | 1.784856e-01 | 1.1e-10 | 6.0e-04 | 3.7e-08 | 1.0e-05 |

Tabela J.13: Resultados numéricos para a função mérito baseada na Lagrangeana aumentada e ordenação dual

| Problem | iter | nmer | fx | $\|\|\rho\|\|_2$ | $\|\|\sigma\|\|_2$ | $\mu$ | $\epsilon$ |
|---|---|---|---|---|---|---|---|
| blankenship1 | 36 | 111 | 5.000839e-01 | 1.0e-04 | 2.4e-05 | 5.5e+00 | 1.0e-03 |
| coopeL | 35 | 151 | 1.999485e+00 | 1.0e-04 | 6.3e-04 | 4.2e+01 | 1.0e-03 |
| coopeN | 42 | 122 | 3.811448e-01 | 2.0e-06 | 6.2e-04 | 5.0e-01 | 1.0e-03 |
| elke10 | 401 | 1632 | 2.832798e+00 | 3.0e-03 | 3.2e-01 | 3.4e-14 | 1.0e-02 |
| elke1std | 401 | 1121 | 1.674320e+28 | 2.1e-04 | 1.8e-01 | 4.8e+54 | 1.0e-02 |
| elke2std | 401 | 924 | 1.438004e+05 | 2.1e-04 | 1.1e-01 | 6.7e+12 | 1.0e-02 |
| elke3std | 253 | 1534 | NaN | NaN | NaN | NaN | 1.0e-03 |
| elke4std | 401 | 499 | 5.847752e+00 | 2.3e-04 | 1.3e-01 | 2.6e+00 | 1.0e-02 |
| elke5std | 401 | 418 | 2.843560e+00 | 9.9e-03 | 6.4e-02 | 2.3e-01 | 1.0e-02 |
| elke6std | 401 | 1069 | 1.508812e+09 | 2.1e-04 | 3.6e-01 | 6.4e-06 | 1.0e-02 |
| elke7std | 401 | 1029 | 1.102368e+06 | 3.8e+00 | 3.9e-01 | 7.9e+15 | 1.0e-02 |
| elke8 | 401 | 2043 | 1.226246e+02 | 7.1e-06 | 3.5e+00 | 1.6e+07 | 1.0e-02 |
| elke9 | 401 | 1430 | 3.297468e+00 | 1.0e-05 | 3.6e-01 | 3.8e-12 | 1.0e-02 |
| ferris1 | 401 | 1208 | 8.942993e+03 | 2.0e-06 | 7.6e-01 | 4.0e-12 | 1.0e-02 |
| ferris2 | 401 | 1352 | -1.761790e+00 | 3.3e-05 | 7.4e-03 | 2.1e-03 | 1.0e-03 |
| goerner1 | 401 | 1358 | 6.709092e+04 | 5.6e-13 | 7.1e-01 | 1.7e-11 | 1.0e-02 |
| goerner2 | 74 | 228 | 2.744073e-01 | 1.4e-05 | 6.3e-04 | 2.2e+01 | 1.0e-04 |
| goerner3 | 4 | 6 | NaN | NaN | NaN | NaN | 1.0e-03 |
| gugat1 | 145 | 601 | 1.821847e+01 | 2.0e-05 | 2.7e-04 | 6.6e+02 | 1.0e-03 |
| gugat2 | 41 | 144 | 3.335296e-01 | 2.1e-05 | 4.9e-06 | 1.3e-01 | 1.0e-03 |
| gugat4b | 200 | 637 | 1.500105e+00 | 3.1e-06 | 9.2e-04 | 4.1e-01 | 1.0e-03 |
| gugat4c | 401 | 1272 | 2.532579e-01 | 7.1e-12 | 3.9e-03 | 6.5e-11 | 1.0e-02 |
| gugat4d | 401 | 1527 | 2.507936e+00 | 1.3e-07 | 3.0e-01 | 4.1e-12 | 1.0e-02 |
| gugat4e | 186 | 571 | 5.008322e-01 | 2.0e-04 | 9.6e-04 | 1.4e+00 | 1.0e-03 |
| gugat4f | 213 | 838 | 3.586114e+01 | 2.0e-05 | 2.3e-04 | 8.6e+02 | 1.0e-03 |
| gugat5a | 152 | 653 | 3.883604e+01 | 2.0e-05 | 6.8e-04 | 3.9e+03 | 1.0e-03 |
| gugat5b | 36 | 96 | 1.567262e+02 | 2.0e-05 | 3.1e-04 | 8.2e+03 | 1.0e-03 |
| gugat5c | 67 | 197 | 3.359218e-01 | 2.0e-05 | 3.7e-04 | 1.2e+01 | 1.0e-04 |
| gugat5d | 154 | 549 | 9.459567e-01 | 1.7e-04 | 3.8e-04 | 5.7e-04 | 1.0e-03 |
| gugat5e | 401 | 1343 | 6.729910e-02 | 3.7e-11 | 6.2e-03 | 8.6e-14 | 1.0e-04 |
| gugat5f | 134 | 664 | 6.865588e+01 | 2.0e-05 | 9.4e-05 | 3.2e+03 | 1.0e-03 |
| gugat6 | 401 | 1239 | 5.099849e+00 | 8.6e-05 | 1.0e+00 | 7.2e-285 | 1.0e-02 |
| gugat7 | | | | | | | |
| gugat4a | 217 | 690 | 4.847140e+01 | 2.0e-05 | 1.5e-09 | 1.3e+05 | 1.0e-04 |
| hettich10 | 401 | 1096 | 1.317929e+04 | 1.1e-08 | 3.3e+02 | 5.7e+09 | 1.0e-02 |
| hettich10c | 142 | 449 | 1.234883e+02 | 1.4e-05 | 1.1e-07 | 3.1e+04 | 1.0e-03 |
| hettich2 | 330 | 996 | 5.439311e-01 | 1.3e-05 | 4.4e-05 | 6.2e-03 | 1.0e-04 |
| hettich3 | 401 | 1387 | 8.844294e-02 | 5.4e-03 | 9.8e-01 | 1.7e-11 | 1.0e-02 |
| hettich4 | 73 | 318 | 9.995820e-01 | 2.3e-11 | 6.7e-04 | 1.9e-07 | 1.0e-05 |
| hettich8 | 39 | 114 | 2.509253e-01 | 1.4e-04 | 3.6e-04 | 1.4e+00 | 1.0e-03 |
| leon1 | 401 | 691 | 3.125863e+00 | 2.5e-05 | 2.3e-02 | 2.2e+00 | 1.0e-02 |
| leon10 | 380 | 1393 | 5.437256e-01 | 1.3e-05 | 1.9e-04 | 5.9e-03 | 1.0e-04 |
| leon11 | 212 | 888 | 5.032405e-01 | 2.8e-06 | 1.9e-04 | 2.7e-01 | 1.0e-03 |
| leon12 | 401 | 478 | -1.217591e-01 | 3.2e-04 | 1.9e-01 | 1.3e+00 | 1.0e-02 |

| Problema | iter | nmer | fx | $\|\|\rho\|\|_2$ | $\|\|\sigma\|\|_2$ | $\mu$ | $\epsilon$ |
|---|---|---|---|---|---|---|---|
| leon13 | 153 | 675 | 2.360901e-01 | 2.3e-06 | 4.9e-05 | 3.0e-05 | 1.0e-05 |
| leon14 | 91 | 250 | 1.182640e+00 | 2.0e-06 | 2.5e-04 | 7.1e-01 | 1.0e-03 |
| leon15 | 42 | 224 | -4.409487e-01 | 1.0e-04 | 7.6e-05 | 2.7e+00 | 1.0e-03 |
| leon16 | 27 | 87 | 1.986698e+00 | 1.0e-04 | 8.2e-04 | 2.5e+00 | 1.0e-03 |
| leon18 | 401 | 1217 | -1.761015e+00 | 4.0e-11 | 2.0e-03 | 7.2e-12 | 1.0e-02 |
| leon19 | 46 | 139 | 1.035514e+00 | 1.0e-04 | 9.8e-04 | 2.7e+00 | 1.0e-03 |
| leon2 | 401 | 1393 | 9.903630e+00 | 4.5e-06 | 8.8e-01 | 1.9e-12 | 1.0e-02 |
| leon3 | 401 | 1229 | 4.873820e+02 | 1.4e-04 | 1.0e+00 | 2.7e-11 | 1.0e-02 |
| leon4 | 148 | 862 | 5.001280e-01 | 2.8e-06 | 9.8e-04 | 2.7e-01 | 1.0e-03 |
| leon5 | 401 | 1328 | 5.221671e-02 | 1.4e-02 | 1.5e+01 | 3.8e-10 | 1.0e-02 |
| leon6 | 70 | 592 | 2.499265e-01 | 1.4e-04 | 7.4e-04 | 1.4e+00 | 1.0e-03 |
| leon7 | 401 | 1201 | 3.730727e+02 | 4.2e-09 | 8.2e-01 | 3.7e-12 | 1.0e-02 |
| leon8 | 401 | 1303 | 3.044629e+04 | 8.3e-04 | 1.0e+00 | 1.6e-13 | 1.0e-02 |
| leon9 | 401 | 1270 | 6.162072e+02 | 3.6e-05 | 9.3e-01 | 1.8e-12 | 1.0e-02 |
| liu1 | 77 | 229 | -4.654997e+00 | 1.6e-05 | 6.9e-04 | 1.2e-04 | 1.0e-04 |
| liu2 | 401 | 500 | -2.519175e+00 | 2.0e-04 | 6.0e-02 | 2.4e-01 | 1.0e-02 |
| liu3 | 401 | 1484 | 1.598070e+02 | 2.8e-04 | 1.1e-01 | 1.4e+00 | 1.0e-02 |
| matlab1 | 401 | 1668 | 2.341856e+01 | 1.0e-03 | 3.4e+00 | 6.1e+00 | 1.0e-02 |
| powell1 | 42 | 124 | -9.766819e-01 | 1.1e-05 | 3.3e-13 | 8.1e-02 | 1.0e-03 |
| priceK | 34 | 115 | -1.473531e+00 | 1.1e-05 | 7.7e-04 | 5.6e+00 | 1.0e-03 |
| still1 | 401 | 1286 | 9.931855e-01 | 8.5e-05 | 5.8e-01 | 6.7e-13 | 1.0e-03 |
| watson1 | 401 | 476 | -2.539101e-01 | 3.5e-04 | 2.1e-01 | 3.9e+00 | 1.0e-02 |
| watson14 | 82 | 369 | 2.217541e+00 | 2.1e-06 | 5.9e-04 | 7.6e-02 | 1.0e-03 |
| watson2 | 401 | 1172 | 2.650607e+00 | 3.1e-06 | 3.6e-01 | 3.3e-04 | 1.0e-02 |
| watson3 | 62 | 187 | 5.332101e+00 | 1.9e-05 | 2.4e-04 | 2.9e-04 | 1.0e-04 |
| watson4a | 401 | 1281 | 6.545106e-01 | 2.8e-06 | 6.7e-01 | 3.5e-12 | 1.0e-03 |
| watson4b | 44 | 143 | 8.655735e-01 | 1.0e-04 | 7.4e-04 | 2.7e+00 | 1.0e-03 |
| watson4c | 43 | 139 | 8.655385e-01 | 1.0e-04 | 3.7e-04 | 2.7e+00 | 1.0e-03 |
| watson5 | 401 | 1071 | 4.898543e+00 | 2.7e-06 | 2.6e-01 | 1.3e-03 | 1.0e-02 |
| watson6 | 401 | 1378 | 9.714003e+01 | 2.1e-05 | 2.4e-02 | 2.4e-03 | 1.0e-03 |
| zhou1 | 39 | 114 | 3.577507e-01 | 1.0e-04 | 5.1e-04 | 2.3e+00 | 1.0e-03 |

Tabela J.14: Resultados numéricos para a função mérito $l_2$ e ordenação primal

| Problem | iter | nmer | fx | $\|\|\rho\|\|_2$ | $\|\|\sigma\|\|_2$ | $\mu$ | $\epsilon$ |
|---|---|---|---|---|---|---|---|
| blankenship1 | 58 | 174 | 4.999468e-03 | 6.6e-12 | 1.1e-04 | 9.5e-11 | 1.0e-05 |
| coopeL | 45 | 140 | 3.433276e-01 | 1.3e-11 | 3.4e-04 | 1.1e-08 | 1.0e-05 |
| coopeN | 56 | 170 | 9.801729e-04 | 3.9e-11 | 5.6e-04 | 1.2e-07 | 1.0e-05 |
| elke10 | 401 | 2240 | 2.187920e+00 | 2.3e-06 | 3.3e-01 | 5.3e-12 | 1.0e-03 |
| elke1std | 401 | 1276 | 2.788942e+05 | 4.7e-14 | 3.6e-01 | 1.4e-182 | 1.0e-02 |
| elke2std | 401 | 1231 | 1.436819e+05 | 8.2e-13 | 3.6e-01 | 1.3e-296 | 1.0e-02 |

| Problema | iter | nmer | fx | $\|\|\rho\|\|_2$ | $\|\|\sigma\|\|_2$ | $\mu$ | $\epsilon$ |
|---|---|---|---|---|---|---|---|
| elke3std | 401 | 1240 | 7.562609e+00 | 8.8e-05 | 4.6e-01 | 5.4e-13 | 1.0e-02 |
| elke4std | 401 | 1356 | 2.050708e+01 | 5.5e+04 | 3.6e-01 | 8.3e-35 | 1.0e-03 |
| elke5std | 401 | 418 | 2.843560e+00 | 9.9e-03 | 6.4e-02 | 2.3e-01 | 1.0e-02 |
| elke6std | 401 | 1231 | 9.200421e+04 | 2.9e-13 | 3.6e-01 | 5.1e-275 | 1.0e-02 |
| elke7std | 401 | 936 | 2.631946e+13 | 4.5e+11 | 3.2e-01 | 4.6e+09 | 1.0e-02 |
| elke8 | 401 | 1244 | 2.738667e+04 | 2.5e-14 | 3.6e-01 | 3.5e-156 | 1.0e-03 |
| elke9 | 401 | 2018 | 1.560632e+05 | 4.5e-13 | 3.6e-01 | 8.2e-29 | 1.0e-03 |
| ferris1 | 401 | 1365 | 4.593631e-03 | 5.7e-11 | 8.4e-03 | 5.9e-13 | 1.0e-04 |
| ferris2 | 401 | 1200 | -1.760926e+00 | 5.0e-12 | 1.8e-03 | 2.4e-11 | 1.0e-03 |
| goerner1 | 102 | 309 | 2.499377e-01 | 1.4e-04 | 9.0e-04 | 1.4e+00 | 1.0e-03 |
| goerner2 | 308 | 1001 | 3.145066e-03 | 7.6e-11 | 9.8e-04 | 4.9e-13 | 1.0e-05 |
| goerner3 | 4 | 6 | NaN | NaN | NaN | NaN | 1.0e-03 |
| gugat1 | 240 | 803 | 1.403480e-02 | 3.5e-12 | 9.5e-04 | 9.7e-14 | 1.0e-05 |
| gugat2 | 33 | 92 | 3.831804e-01 | 1.0e-08 | 4.8e-04 | 3.0e-07 | 1.0e-03 |
| gugat4a | 165 | 551 | 5.000878e-03 | 1.6e-14 | 9.8e-04 | 1.3e-12 | 1.0e-05 |
| gugat4b | 401 | 1234 | 1.806634e+03 | 1.5e-13 | 1.0e+00 | 1.7e-104 | 1.0e-03 |
| gugat4c | 401 | 1232 | 1.262439e-02 | 1.6e-14 | 3.0e-03 | 8.1e-14 | 1.0e-04 |
| gugat4d | 401 | 1330 | 6.037148e+08 | 4.5e+111 | 4.3e+05 | 8.8e+115 | 1.0e-03 |
| gugat4e | 401 | 1303 | 6.958836e-02 | 2.9e-05 | 3.9e-03 | 1.2e-01 | 1.0e-03 |
| gugat4f | 401 | 2264 | 5.947687e+08 | 2.6e+02 | 1.7e+03 | 3.3e-09 | 1.0e-04 |
| gugat5a | 401 | 1198 | 5.019595e-02 | 9.9e-11 | 1.9e-03 | 1.3e-11 | 1.0e-03 |
| gugat5b | 401 | 1221 | 1.526308e+00 | 2.5e-08 | 1.1e-03 | 3.9e-10 | 1.0e-02 |
| gugat5c | 401 | 1245 | 2.410068e-01 | 1.3e+100 | 2.3e-01 | 8.4e+99 | 1.0e-03 |
| gugat5d | 401 | 1228 | 2.396281e+00 | 9.9e+121 | 5.5e+02 | 1.4e+125 | 1.0e-03 |
| gugat5e | 401 | 1199 | 9.004917e-02 | 2.7e-10 | 6.7e-03 | 1.3e-10 | 1.0e-03 |
| gugat5f | 401 | 1218 | 4.626361e-01 | 9.0e-04 | 2.3e-01 | 4.2e-10 | 1.0e-02 |
| gugat6 | | | | | | | |
| gugat7 | 67 | 203 | 2.721520e-03 | 1.0e-10 | 9.9e-04 | 6.0e-13 | 1.0e-05 |
| hettich10 | 401 | 907 | -4.486681e-01 | 4.5e+00 | 3.4e-11 | 1.1e-02 | 1.0e-02 |
| hettich10c | 401 | 1211 | 1.994537e+03 | 4.7e+94 | 1.0e+05 | 1.7e+92 | 1.0e-04 |
| hettich2 | 79 | 253 | 5.439318e-01 | 1.5e-04 | 3.0e-04 | 5.7e-01 | 1.0e-04 |
| hettich3 | 147 | 585 | 3.909985e-03 | 1.4e-14 | 2.6e-04 | 3.7e-13 | 1.0e-05 |
| hettich4 | 28 | 80 | 9.995804e-01 | 1.6e-13 | 3.3e-05 | 2.4e-08 | 1.0e-05 |
| hettich8 | 136 | 456 | 7.154637e-03 | 1.4e-14 | 9.9e-04 | 1.1e-12 | 1.0e-05 |
| leon1 | 109 | 349 | 5.018199e-03 | 2.7e-14 | 8.6e-05 | 1.8e-11 | 1.0e-05 |
| leon10 | 103 | 364 | 5.381677e-01 | 1.4e-04 | 4.4e-04 | 1.1e-03 | 1.0e-05 |
| leon11 | 249 | 982 | 4.840022e-02 | 3.8e-13 | 3.6e-04 | 1.6e-11 | 1.0e-05 |
| leon12 | 31 | 86 | -6.091416e-01 | 1.2e-05 | 2.3e-05 | 1.0e+00 | 1.0e-03 |
| leon13 | 36 | 103 | 2.361105e-01 | 3.4e-11 | 2.2e-04 | 1.4e-08 | 1.0e-05 |
| leon14 | 64 | 185 | 6.667644e-01 | 4.3e-12 | 8.9e-05 | 1.4e-08 | 1.0e-05 |
| leon15 | 88 | 478 | -6.664801e-01 | 1.0e-04 | 1.3e-04 | 4.0e-03 | 1.0e-05 |
| leon16 | 45 | 139 | 1.857584e+00 | 1.0e-04 | 2.3e-05 | 3.2e-03 | 1.0e-04 |
| leon18 | 401 | 1207 | -1.761269e+00 | 4.3e-11 | 1.9e-03 | 7.4e-12 | 1.0e-02 |
| leon19 | 394 | 1196 | 7.882031e-01 | 6.9e-13 | 1.0e-03 | 2.2e-12 | 1.0e-05 |

| Problema | iter | nmer | fx | $\|\|\rho\|\|_2$ | $\|\|\sigma\|\|_2$ | $\mu$ | $\epsilon$ |
|---|---|---|---|---|---|---|---|
| leon2 | 259 | 998 | 4.999630e-03 | 6.7e-12 | 3.0e-04 | 3.3e-10 | 1.0e-05 |
| leon3 | 401 | 1202 | 7.343257e+01 | 2.6e-11 | 7.1e-01 | 1.5e-11 | 1.0e-02 |
| leon4 | 346 | 1521 | 5.498603e-03 | 4.8e-12 | 6.0e-04 | 8.3e-11 | 1.0e-05 |
| leon5 | 401 | 2050 | 1.538577e-02 | 2.0e-06 | 9.4e-01 | 1.6e-05 | 1.0e-04 |
| leon6 | 115 | 365 | 2.502752e-03 | 1.4e-11 | 2.2e-04 | 1.2e-12 | 1.0e-05 |
| leon7 | 401 | 1217 | 2.738988e-02 | 1.6e-12 | 1.3e-03 | 1.1e-11 | 1.0e-03 |
| leon8 | 401 | 1254 | 1.002418e+00 | 1.2e-11 | 2.7e-03 | 5.0e-10 | 1.0e-02 |
| leon9 | 401 | 1548 | 1.066535e+00 | 2.3e-07 | 1.0e+00 | 8.0e-05 | 1.0e-03 |
| liu1 | 36 | 120 | -4.653991e+00 | 2.0e-14 | 5.4e-04 | 4.1e-09 | 1.0e-05 |
| liu2 | 36 | 102 | -2.505259e+00 | 1.1e-05 | 1.1e-04 | 2.4e+00 | 1.0e-03 |
| liu3 | 401 | 2886 | 1.540193e+02 | 2.1e-05 | 4.5e+01 | 4.8e-12 | 1.0e-04 |
| matlab1 | 401 | 1804 | 8.248210e+02 | 2.5e-04 | 2.7e+01 | 2.5e+01 | 1.0e-02 |
| powell1 | 50 | 151 | -1.000190e+00 | 9.2e-11 | 2.0e-04 | 3.2e-08 | 1.0e-04 |
| priceK | 38 | 115 | -1.174759e+00 | 2.0e-06 | 4.5e-05 | 4.1e+00 | 1.0e-03 |
| still1 | 67 | 227 | 1.000050e+00 | 1.5e-13 | 1.2e-04 | 6.6e-10 | 1.0e-05 |
| watson1 | 59 | 159 | -2.787140e-01 | 1.4e-05 | 7.4e-04 | 6.6e-04 | 1.0e-04 |
| watson14 | 43 | 131 | 2.199312e+00 | 5.0e-14 | 3.8e-05 | 1.5e-09 | 1.0e-05 |
| watson2 | 45 | 151 | 2.430538e+00 | 4.5e-10 | 4.5e-04 | 8.7e-08 | 1.0e-05 |
| watson3 | 51 | 168 | 5.334417e+00 | 2.3e-14 | 9.9e-05 | 2.7e-09 | 1.0e-05 |
| watson4a | 124 | 387 | 6.491129e-01 | 2.0e-11 | 4.3e-04 | 2.3e-13 | 1.0e-05 |
| watson4b | 159 | 598 | 6.184051e-01 | 2.6e-12 | 7.5e-04 | 2.2e-12 | 1.0e-05 |
| watson4c | 68 | 213 | 6.183075e-01 | 6.7e-12 | 8.6e-04 | 2.3e-12 | 1.0e-05 |
| watson5 | 102 | 360 | 4.301433e+00 | 4.4e-12 | 8.4e-04 | 1.1e-10 | 1.0e-05 |
| watson6 | 401 | 2044 | 2.505217e+02 | 8.1e-01 | 5.4e+02 | 3.1e-59 | 1.0e-02 |
| zhou1 | 47 | 133 | 1.856317e-01 | 1.0e-04 | 6.1e-04 | 1.2e+00 | 1.0e-04 |

Tabela J.15: Resultados numéricos para a função mérito baseada na Lagrangeana aumentada e ordenação primal

# Índice

## W



## Z