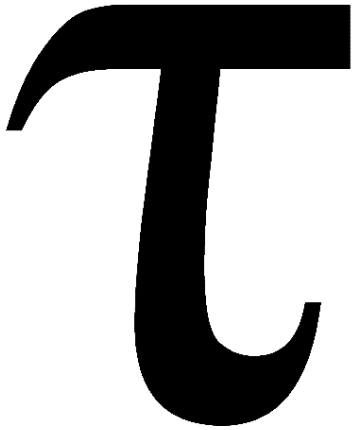

Estabilização numérica e implementação do método tau de Lanczos

Marcelo da Silva Trindade
Doutoramento em Matemática Aplicada
Departamento de Matemática
2017

Orientador
Dr. Paulo Beleza Vasconcelos
Professor da Faculdade de Economia (FEP – UP) e membro do Centro de Matemática da Universidade do Porto (CMUP)

Co-orientador
Dr. José Manuel Matos
Professor do Instituto Superior de Engenharia do Porto (ISEP – IPP) e membro do Centro de Matemática da Universidade do Porto (CMUP)

Para Marina Lanes de Almeida, quem combateu o bom combate, guardou e transmitiu a fé da melhor forma possível: com o exemplo. É desejo profundo do meu coração que estejas a contemplar o rosto do Senhor, na companhia eterna da Virgem Santa Maria, de São José seu esposo, dos bem-aventurados apóstolos e mártires, e de todos os santos e anjos.

Agradecimentos

Agradeço primeiramente a Deus por nos sustentar a cada instante em Sua infinita bondade, e Lhe peço que os frutos deste trabalho sejam de alguma forma úteis para que me faça instrumento de Sua vontade.

Obrigado Professor Paulo e Professor José por todos os ensinamentos, dedicação e amizade, sob vossas supervisões o trabalho desenvolveu-se de maneira muito agradável e produtiva. Não poderia ter melhores orientadores.

Obrigado minha esposa Andréa por encarar esta jornada ao meu lado, e por me dar neste período o melhor presente que eu poderia receber, a Maria Flor. Suas companhias foram, e são, fundamentais. Vocês são a razão de todo o esforço.

Obrigado a todos os colegas e amigos frutos do doutoramento. A descontração, incentivo mútuo e amizade são factores indispensáveis para que todos alcancem seus objetivos.

Obrigado a toda minha família que do Brasil me apoiaram, e pacientemente me esperaram para o dia do retorno, amo todos vocês.

Reconhecer o apoio daquelas pessoas que se importaram pra que esta meta fosse cumprida é minha obrigação, e o faço com muito carinho, o meu sincero muito obrigado a cada um de vós!

Resumo

Este trabalho tem como objetivo principal a análise do método tau e a sua implementação em rotinas de cálculo numérico. O método tau pertence à classe de métodos designados por espetrais, cujo objetivo consiste em resolver problemas diferenciais aproximando a solução por um polinómio. A fim de obter uma solução com boas propriedades de aproximação, nestes métodos, o resíduo, resultante de aproximar a solução por um polinómio, é projetado numa base de polinómios ortogonais e são impostas condições de forma a garantir um resíduo pequeno, numa norma adequada. A implementação eficiente e robusta do método, conseguida com esta Tese, resulta da conjugação do trabalho em três áreas: Análise Numérica, Álgebra Linear Numérica e Programação.

O trabalho de Análise Numérica apresentado aprofunda a análise do método tau e dos métodos numéricos auxiliares, incluindo a obtenção de formulações alternativas para as operações com polinómios, a representação algébrica de operadores diferenciais lineares e com coeficientes polinomiais e a generalização da representação a outras classes de operadores. As propriedades dos polinómios ortogonais são utilizadas para substituir procedimentos numéricos por fórmulas exatas, ou por procedimentos numéricos mais estáveis, permitindo reduzir a propagação de erros. A aproximação de funções de matrizes é introduzida para aproximar a representação algébrica de operadores integro-diferenciais com coeficientes não polinomiais. A aproximação parcelar, a análise do erro e a análise de métodos iterativos de implementação do método tau, permitem alcançar resultados com a precisão disponível nas ferramentas de cálculo.

A Álgebra Linear Numérica inclui a redefinição da representação matricial de operações, a exploração de propriedades de esparsidade de matrizes e a construção de algoritmos iterativos nas operações com matrizes e vetores. Para obter soluções numéricas com elevada precisão, introduz-se no método a utilização de precondicionadores e de técnicas de pivotagem, juntamente com a utilização de métodos diretos e de métodos iterativos de resolução dos sistemas algébricos lineares associados ao método tau. São também disponibilizadas versões por blocos de alguns destes algoritmos.

O trabalho de Programação consiste na implementação de funções e rotinas `MATLAB` capazes de interpretar problemas integro-diferenciais, construir a sua representação algébrica numa base de polinómios ortogonais, construir o problema linear algébrico associado, resolvê-lo e apresentar a solução, de forma numérica, simbólica e gráfica. A classe de problemas tratados inclui equações e sistemas de equações integro-diferenciais, lineares ou não lineares, com coeficientes polinomiais ou coeficientes representados por funções elementares não polinomiais ou funções compostas de funções elementares e

polinómios, com condições iniciais, fronteira ou outras. As rotinas de cálculo permitem calcular a solução por métodos diretos ou iterativos, e com versão parcelar. É possível calcular uma solução, ou uma sucessão de soluções, apresentar os resíduos e obter estimativas dos erros. Os problemas, juntamente com os parâmetros e as opções de cálculo, podem ser fornecidos ao sistema por diversos processos, adaptados a diversos níveis de especialização do utilizador, incluindo uma interface gráfica de utilizador.

Os resultados da Tese permitem levar o método tau mais longe, mais rápido e de forma mais fácil. A implementação na forma de uma caixa de ferramentas, permite resolver problemas integro-diferenciais de uma forma mais intuitiva. A introdução de rotinas estáveis e de cálculo eficiente permite obter resultados com polinómios de graus mais elevado e mais precisos do que alguma vez publicado na literatura do método.

Abstract

This work analysis the tau method and its numerical implementation. The tau method belongs to the class of spectral methods and is intended to numerically compute polynomial approximate solutions of differential problems. In order to obtain a solution with good approximation properties, in these methods, the residual, resulting from the approximation of the solution by a polynomial, is projected on an orthogonal polynomial basis and conditions are imposed to ensure a small residual, in a suitable norm. The efficient and robust implementation of the method, obtained with this thesis, results from the conjugation of the work in three areas: Numerical Analysis, Numerical Linear Algebra and Programming.

The Numerical Analysis to be presented analyzes, in deep, the tau method and auxiliary numerical methods, including the development of alternative formulations for polynomial operations, the algebraic representation of linear and polynomial differential operators, and the extension of the representation of other classes of operators. The properties of the orthogonal polynomials are used to replace numerical procedures by exact formulas, or by more stable numerical procedures, reducing the propagation of errors. The approximation of matrix functions is introduced to approximate the algebraic representation of integral-differential operators with non-polynomial coefficients. The piecewise approximation, the analysis of the error and the analysis of iterative methods to implement the tau method, allow for numerical results within machine precision.

Numerical Linear Algebra includes the redefinition of the matrix representation of operations, the exploration of the sparsity of some of the large dimensional matrices involved and the construction of iterative algorithms in operations with matrices and vectors. To obtain numerical solutions with high precision, the use of preconditioners and pivoting techniques are introduced along with the use of direct methods and iterative methods for solving linear algebraic systems associated with the tau method. Block versions of some of these algorithms are also developed.

The programming work consists on the implementation of MATLAB functions capable of interpreting integral-differential problems, constructing their algebraic representation on a basis of orthogonal polynomials, building the associated algebraic linear problem, solving it and showing the solution, numerically and graphically. The class of problems dealt with includes systems of integral equations, linear or non-linear, with polynomial coefficients or coefficients represented by non-polynomial elementary functions or composite functions of elementary and polynomial functions, with initial, boundary, or other conditions. The routines developed allow the computation of the solution by direct or iterative methods, and with piecewise versions for large domains.

It is possible to compute a solution, or a succession of solutions, to present the residuals and obtain estimates for the errors. The problems, together with the parameters and options, can be provided to the system by various processes, adapted to various levels of user specialization. A graphical user interface is also provided.

The results of this thesis allow taking the tau method further, making it faster to compute and easier to tackle. The implementation in the form of a toolbox, allows solving integral-differential problems in a more intuitive way. The introduction of stable and efficient routines allows results to be obtained with higher degree polynomials and with more precision than ever published in the method's literature.

Résumé

Cette thèse analyse la méthode tau et sa mise en œuvre numérique. C'est une méthode appartenant à la famille des méthodes spectrales et son objectif est de calculer une approximation polynomiale de la solution d'un problème différentiel. Afin d'avoir de bonnes propriétés sur le plan de l'approximation de fonctions, la méthode projette le résidu de la solution numérique polynomiale sur un sous-espace engendré par une base orthogonale de polynômes et on introduit des conditions conduisant à réduire ce résidu projeté, mesuré avec une norme particulière. L'efficience et la robustesse de l'implementation proposée dans cette thèse découlent de l'imbrication de résultats provenant de trois domaines: l'analyse numérique, l'algèbre linéaire numérique et la programmation d'algorithmes.

L'analyse numérique approfondie à trait à la méthode tau et des méthodes auxiliaires comprenant des formulations alternatives des opérations polynomiales, la représentation algébrique d'opérateurs linéaires et différentiels polynomiaux, ainsi que de l'extension de la représentation d'autres classes d'opérateurs. Les propriétés des polynômes orthogonaux sont utilisées pour substituer des calculs exacts, ou des procédures d'approximation stables, à la place d'approximations numériques classiques, réduisant ainsi la propagation des erreurs. L'approximation de fonctions de matrices est introduite afin d'approcher la représentation algébrique d'opérateurs intégro-différentiels à coefficients non polynomiaux. Des approximations par morceaux, l'analyse d'erreur et celle d'algorithmes itératifs pour la mise en œuvre de la méthode tau permettent l'obtention de résultats possédant la précision machine.

Les résultats du domaine de l'algèbre linéaire numérique apportent une redéfinition de la représentation matricielle des opérations, l'exploration du caractère creux des matrices de grande taille et la construction d'algorithmes itératifs pour les opérations concernant vecteurs et matrices. Pour améliorer la précision, on a recours à des techniques de préconditionnement et de pivotage utilisées en combinaison avec des méthodes directes ou itératives lors de la résolution de systèmes d'équations algébriques linéaires inhérents à la méthode tau. Le travail inclut aussi les versions par blocs correspondantes.

Le domaine de la programmation d'algorithmes contribue à cette thèse au travers des fonctions de la bibliothèque MATLAB qui sont capables d'interpréter des problèmes intégro-différentiels, par le biais de la construction de leur représentation dans une base de polynômes orthogonaux. Ceci permet la création du problème algébrique linéaire associé, l'obtention de sa solution et l'exhibition de celle-ci sous formes numérique et graphique. Parmi les problèmes traités dans cette thèse, on reconnaît des systèmes d'équations intégrales linéaires et non linéaires avec des coefficients polynomiaux ou représentés par

des fonctions non polynomiales mais élémentaires ou par la composition des fonctions précitées, avec des conditions aux bords ou d'autres types de contraintes. Les programmes informatiques de calcul numérique développés dans ce travail permettent l'obtention de la solution numérique du problème par des méthodes directes ou itératives incluant des versions par morceaux mieux adaptées aux problèmes posés sur des ensembles beaucoup trop vastes. On est enfin en mesure de calculer une solution numérique, ou une suite de telles solutions, de présenter leur résidu et d'avoir des estimations de l'erreur. Le problème à résoudre, les paramètres et les options choisis peuvent être communiqués au système par divers procédés adaptés aux différents niveaux selon la spécialité de l'usager. À cela s'ajoute une interface graphique.

Les résultats de cette thèse rendent la méthode tau plus accessible, plus performante, plus générale et plus rapide. Sa présentation sous la forme d'une "boîte à outils" permet la résolution des problèmes intégro-différentiels d'une façon plus intuitive. L'introduction de routines stables et efficientes permet d'obtenir de bons résultats dans le cas de polynômes de plus haut degré et avec une meilleure précision que ce qui a été publié jusqu'à nos jours sur ce sujet.

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

$\mathcal{X}$	Base das potências de x, $\mathcal{X} = [1,\ x,\ x^2,\ ...]$.
$\mathcal{Z}$	Base de polinómios ortogonais, $\mathcal{Z} = [Z_0,\ Z_1,\ Z_2,\ ...]$.
$\mathcal{T}$	Base dos polinómios de Chebyshev (1^{a} espécie), $\mathcal{T} = [T_0,\ T_1,\ T_2,\ ...]$.
$\mathcal{U}$	Base dos polinómios de Chebyshev (2^{a} espécie), $\mathcal{U} = [U_0,\ U_1,\ U_2,\ ...]$.
$\mathcal{P}$	Base dos polinómios de Legendre, $\mathcal{P} = [P_0,\ P_1,\ P_2,\ ...]$.
$\mathcal{H}$	Base dos polinómios de Hermite, $\mathcal{H} = [H_0,\ H_1,\ H_2,\ ...]$.
$\mathcal{L}$	Base dos polinómios de Laguerre, $\mathcal{L} = [L_0,\ L_1,\ L_2,\ ...]$.
$\mathfrak{D}$	Operador diferencial de coeficientes polinomiais, $\mathfrak{D} = \sum_{i=0}^{\nu} p_i \frac{d^i}{dx^i}$.
$\mathfrak{S}$	Operador integral de coeficientes polinomiais, $\mathfrak{S} = \sum_{i=0}^{\gamma} p_i \int dx^i$.
$\mathfrak{S}_V$	Operador de Volterra de coeficientes polinomiais, $\mathfrak{S}_V = \sum_{i=0}^{\gamma_V} p_i \int_a^x K_i(x,t)y(t)dt$.
$\mathfrak{S}_F$	Operador de Fredholm de coeficientes polinomiais, $\mathfrak{S}_F = \sum_{i=0}^{\gamma_F} p_i \int_a^b K_i(x,t)y(t)dt$.
$\mathfrak{E}$	Operador integro-diferencial de coeficientes polinomiais, $\mathfrak{E} = \mathfrak{D} + \mathfrak{S} + \mathfrak{S}_V + \mathfrak{S}_F$.
C	Matriz resultante das condições de contorno, $\mathsf{C} = [c_{ij}]_{\nu\times\infty}$.
D	Matriz resultante do operador $\mathfrak{D}$, $\mathsf{D} = [d_{ij}]_{\infty\times\infty}$.
S	Matriz resultante do operador $\mathfrak{S}$, $\mathsf{S} = [s_{ij}]_{\infty\times\infty}$.
$\mathsf{S}^{(F)}$	Matriz resultante do operador $\mathfrak{S}_V$, $\mathsf{S}^{(F)} = [s_{ij}^{(F)}]_{\infty\times\infty}$.
$\mathsf{S}^{(V)}$	Matriz resultante do operador $\mathfrak{S}_F$, $\mathsf{S}^{(V)} = [s_{ij}^{(V)}]_{\infty\times\infty}$.
E	Matriz resultante do operador $\mathfrak{E}$, $\mathsf{E} = [e_{ij}]_{\infty\times\infty}$.
T	Matriz dos coeficientes do sistema $\mathsf{Ta} = \mathsf{b}$, $\mathsf{T} = [\mathsf{C};\ \mathsf{E}] = [t_{ij}]_{\infty\times\infty}$.
a	Vetor da solução $y_n = \mathcal{Z}\mathsf{a}$, obtido do sistema $\mathsf{Ta} = \mathsf{b}$, $\mathsf{a} = [a_0,\ a_1,\ \ldots]^T$.
b	Vetor dos termos independentes do sistema $\mathsf{Ta} = \mathsf{b}$, $\mathsf{b} = [b_0,\ b_1,\ \ldots]^T$.
$\mathsf{V}_{\mathcal{Z}}$	Matriz de mudança de base $\mathcal{X}\mathsf{V}_{\mathcal{Z}} = \mathcal{Z}$, $\mathsf{V}_{\mathcal{Z}} = [v_{ij}]_{\infty\times\infty}$.
$\mathsf{M}_{\mathcal{Z}}$	Matriz de incremento de grau polinomial: $\mathcal{Z}\mathsf{M}_{\mathcal{Z}}^k\mathsf{a}_{\mathcal{Z}} = x^k\mathcal{Z}\mathsf{a}_{\mathcal{Z}}$, $\mathsf{M}_{\mathcal{Z}} = [\mu_{ij}]_{\infty\times\infty}$.
$\mathsf{N}_{\mathcal{Z}}$	Matriz de diferenciação: $\mathcal{Z}\mathsf{N}_{\mathcal{Z}}^k\mathsf{a}_{\mathcal{Z}} = \frac{d^k}{dx^k}\mathcal{Z}\mathsf{a}_{\mathcal{Z}}$, $\mathsf{N}_{\mathcal{Z}} = [\eta_{ij}]_{\infty\times\infty}$.
$\mathsf{O}_{\mathcal{Z}}$	Matriz de primitivação: $\mathcal{Z}\mathsf{O}_{\mathcal{Z}}^k\mathsf{a}_{\mathcal{Z}} = \int \mathcal{Z}\mathsf{a}_{\mathcal{Z}}dx^k$, $\mathsf{O}_{\mathcal{Z}} = [\theta_{ij}]_{\infty\times\infty}$.

Sumário

Capítulo 1

Introdução

O método tau, introduzido por Lanczos (1938), é um método espetral originalmente desenvolvido para encontrar um polinómio que aproxime a solução de uma equação diferencial ordinária linear de coeficientes polinomiais. No sentido do método tau, um polinómio y_n de grau $n-1$ para aproximar a solução y de um problema diferencial obtém-se impondo que y_n resolva exatamente um problema perturbado, com a adição de um polinómio τ_n na equação diferencial. Visando minimizar o erro, τ_n é projetado numa base de polinómios ortogonais e são anulados os coeficientes dos termos de menor ordem. Este processo conduz a um sistema de equações algébricas lineares de dimensão infinita. A aproximação y_n é calculada truncando o sistema na ordem n, no caso de uma matriz de coeficientes não singular. Esta abordagem matricial do método tau foi iniciada com a formulação operacional introduzida por Ortiz and Samara (1981).

Muitos estudos têm sido realizados sobre a extensão do método às aplicações para aproximar, casuisticamente, a solução de problemas diferenciais lineares e não lineares, em derivadas parciais, integro-diferenciais lineares e não lineares, derivadas fracionárias entre outros. No entanto, em geral, nestes trabalhos o método tau é aplicado na resolução de problemas específicos e não como procedimento genérico. É usual encontrar na literatura aplicações dependentes do tipo de problema e de uma base (ortogonal) selecionada.

Se estamos interessados em aproximar a solução de um problema diferencial suave numa geometria simples e com grande precisão, então os métodos espetrais são uma ferramenta apropriada. Em comparação aos métodos mais usuais, como diferenças finitas ou elementos finitos, os métodos espetrais permitem alcançar uma ordem de precisão elevada (Trefethen, 2000). O método espetral tau permite obter sucessões de aproximantes com taxa de convergência espetral e erros com distribuição quase uniforme no domínio de aproximação. Outra vantagem do método prende-se com a forma indistinta como condições iniciais ou fronteira são tratadas. Por outro lado, estes métodos apresentam algumas desvantagens como a aplicabilidade apenas a problemas não singulares e suaves, em geometrias simples. Também o mau condicionamento do problema algébrico associado a aproximações com grau elevado pode prejudicar as propriedades da solução numérica.

As importantes propriedades espetrais são uma motivação para aprimorarmos o método tau com novas formulações matemáticas. O desenvolvimento de formulações

matemáticas alternativas estáveis e de implementações eficientes permitirá a disseminação na utilização do método tau.

Nesta Tese são apresentados contributos para o desenvolvimento de formulações matemáticas numericamente estáveis e são incorporadas contribuições dispersas de outros autores. Estas novas abordagens são implementadas numa biblioteca de rotinas matemáticas que generalizam a aplicação de método tau a problemas não lineares integro-diferenciais.

Contributos da Tese

- Adaptação da formulação operacional do método tau para a notação convencional da álgebra linear (Capítulo 4).

- Avaliação de polinómios em bases de polinómios ortogonais. Dado um polinómio escrito como combinação linear de uma sequência de polinómios ortogonais, a sua avaliação passa por uma mudança de base, que em geral introduz erro numérico. Utilizamos uma recorrência que evita a mudança de base e avalia, com elevada precisão numérica, o polinómio diretamente na base aplicada (Secção 5.1).

- Obtenção das matrizes de mudança de base por recorrência. A forma convencional de operar uma transformação de semelhança recorre à inversão (possivelmente pela resolução de um sistema de equações lineares) da matriz de mudança de base V, o que gera erros numéricos à medida que a dimensão destas matrizes cresce. Estes erros são mitigados ao utilizar uma formulação recursiva alternativa para a obtenção de V^{-1} (Secção 5.2).

- Cálculo exato das matrizes que representam os operadores integro-diferenciais. As matrizes que são usadas para a transformação de um problema integro-diferencial para um problema algébrico são usualmente transformadas da base das potências para a base ortogonal com a operação $\mathsf{V}^{-1}\mathsf{M}\mathsf{V}$. Evitamos estas transformações de semelhança obtendo tais matrizes recursivamente e diretamente na base utilizada (Secção 5.3).

- Cálculo recursivo do produto, ou potenciação, de matrizes associadas às bases de polinómios ortogonais. Usualmente no método tau, os coeficientes polinomiais são transformados por operações que usam potências de matrizes. Encontramos formas mais rápidas de obter estas operações, em termos de custo computacional, quando confrontadas com as operações convencionais. Para além disto, esta formulação melhora a qualidade da aproximação (Secção 5.4).

- Utilização dos coeficientes de linearização do produto de polinómios em problemas não lineares. Na resolução de problemas não lineares aparecem produtos de polinómios expressos em bases ortogonais, em geral calculados através de uma operação de convolução. Esta operação introduz instabilidade numérica. Em alternativa utilizamos os coeficientes de linearização do produto de polinómios, o que proporciona resultados com maior precisão (Secção 5.5).

- Esquema iterativo baseado em complementos de Schur com estimativa do erro. Este esquema permite aproximar problemas diferenciais e integro-diferenciais iterativamente, com grau polinomial crescente, até que seja atingida uma determinada precisão (Capítulo 6).

- Automatização do método tau para problemas que envolvem coeficientes não polinomiais. Esta abordagem é conseguida por três vias alternativas: utilização de (i) funções de matrizes, (ii) interpolação ortogonal e (iii) aplicação prévia do método tau (Capítulo 7).

- Desenvolvimento e implementação da `Tau Toolbox`, um conjunto de rotinas em `MATLAB` para o cálculo de soluções aproximadas de problemas integro-diferenciais pelo método tau, incorporando todos os contributos supra mencionados, assim como agregando as abordagens no estado da arte referidas na literatura. A `Tau Toolbox` implementa o método tau através de algoritmos estáveis e portanto oferece soluções aproximadas com elevada precisão. A `Tau Toolbox` é capaz de aproximar problemas diferenciais e integro-diferenciais, lineares e não lineares, com condições iniciais, de fronteiras ou outras e operando com as bases mais usuais de polinómios ortogonais clássicos (Capítulo 8).

Estrutura da Tese

Esta Tese está organizada em três partes. A Parte I expõe as ideias e conceitos básicos subjacentes ao método tau, como a teoria dos polinómios ortogonais e generalidades sobre métodos espetrais; ainda nesta Parte (no Capítulo 4) são sintetizados vários dos estudos já desenvolvidos sobre o método tau. Na Parte II são apresentados os contributos de estabilidade para o método tau, como a obtenção de fórmulas de recorrência para: (i) avaliar valores de funções em bases ortogonais, (ii) mudar de base, (iii) calcular o produto de polinómios em bases ortogonais, (iv) calcular os coeficientes de linearização do produto de polinómios, (v) evitar inversões de matrizes e (vi) obter a representação matricial de operadores integro-diferenciais com coeficientes não polinomiais. A Parte III foca-se no desenvolvimento e apresentação da `Tau Toolbox`, uma biblioteca de rotinas em `MATLAB` para a aproximação, via método tau, da solução de problemas diferenciais e integro-diferenciais, lineares e não lineares. São apresentadas as especificações de algumas das funções desenvolvidas, onde são explicados os argumentos de entrada e de saída; são ilustrados vários problemas diferenciais e integro-diferenciais, lineares e não lineares e o processo de obtenção de respetiva solução aproximada via `Tau Toolbox`. Esta biblioteca de funções está disponível para uso pela comunidade científica e industrial de forma livre através do endereço www.fc.up.pt/tautoolbox.

A notação usada nesta Tese segue as orientações seguintes: Letras minúsculas sem serifa a, b, c, ... representam vetores coluna, e portanto os vetores linha aparecem na forma a^T, b^T, c^T, ... Letras minúsculas f, g, h, ... representam as funções de uma variável, salvo as letras i, j, k, ℓ, m, n e r que são reservadas para índices e as letras a, b, c, e d que são reservadas para representar os elementos de matrizes e vetores. Funções de duas ou mais variáveis são denotadas por letras maiúsculas A, B, C, ...,

que também denotam polinómios ortogonais e para isso reservamos somente as letras Z, P, T, U, H, L e B. Letras maiúsculas sem serifa A, B, C, ... são obrigatoriamente matrizes. Letras maiúsculas caligráficas $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, ... designam sequências de polinómios. Letras maiúsculas góticas $\mathfrak{A}$, $\mathfrak{B}$, $\mathfrak{C}$, ... designam operadores. As letras em "blackboard" $\mathbb{P}$, $\mathbb{N}$, $\mathbb{Z}$, ... representam conjuntos ou espaços. O alfabeto grego α, β, γ, ... representa exclusivamente escalares. Sequências numéricas aparecerão na forma $i = \alpha(\beta)\gamma$, onde neste caso, o índice i inicia em α, progride aritmeticamente com razão β até γ. Processos iterativos na variável v têm cada iteração k denotada por $v^{(k)}$. Blocos matriciais de uma matriz A são denotados por $\mathsf{A}^{[l_i:l_f,c_i:c_f]}$, onde l_i e l_f são, respetivamente, a linha inicial e a linha final do bloco de A; da mesma forma, c_i e c_f são, respetivamente, a coluna inicial e a coluna final.

Resultados da Tese

A presente Tese originou apresentação de resultados à comunidade científica internacional, na forma de artigos, capítulo de livro, apresentações em conferências internacionais e elaboração de software. A seguir apresentamos estes resultados.

Publicações

- M. S. Trindade, J. Matos, and P. B. Vasconcelos, *Towards a Lanczos' τ-method toolkit for differential problems*, Mathematics in Computer Science, Springer, 2016, 313-329, 10(3).

  Abstract: The aim of this work is to build a numerical software library based on the τ-method to solve differential problems using MATLAB. The τ-method can be very effective in the solution of certain type of these problems, and therefore, the existence of a numerical library for its dissemination is of major importance. Furthermore, the method has been used for the solution of particular problems but has not yet been explored as a general technique. Focus will be on stability issues, namely those issued from the solution of algebraic linear systems required for the process. Additionally, preconditioners for the solution with the τ-method will be tackled, with emphasizes on incomplete LU factorizations and (block) Jacobi preconditioners. We also propose an iterative approach, build upon an LU factorization over a moderate initial size, generating better approximations and providing a priori error estimate at each iteration. Numerical results enlightening the efficiency of the proposed methods will be presented.

- M. S. Trindade, P. B. Vasconcelos and J. Matos, *Dealing with non-polynomial coefficients within tau method*, Mathematics in Computer Science, Springer (submited).

  Abstract: The tau method is a spectral method originally proposed by Lanczos for the solution of linear differential problems with polynomial coefficients. In this contribution we present three approches to deal with differential equations with non-polinomial functional coefficients: (i) making use of function of matrices, (ii) applying orthogonal interpolation and (iii) solving auxiliary differential problems by tau method itself. These approaches are part of the Tau Toolbox efforts for deploying a numerical library for the solution of integro-differential problems. Numerical experiments illustrate the use of all these polynomial approximations in the context of the tau method.

- P. B. Vasconcelos, J. Matos, and M. S. Trindade, *Spectral Lanczos' tau method for systems of nonlinear integro-differential equations*, Integral Methods in Science and Engineering, Springer, 2017 (to appear).

  Abstract: In this paper an extension of the spectral Lanczos' tau method to systems of non linear integro-differential equations is proposed. This extension includes (i) linearization coefficients of orthogonal polynomials products issued from nonlinear terms and (ii) recursive relations to implement matrix inversion whenever a polynomial change of basis is required and (iii) orthogonal polynomial evaluations directly on the orthogonal basis. All these improvements ensure numerical stability and accuracy in the approximate solution. Exposed in detail, this novel approach is able to significantly outperform numerical approximations with other methods as well as different tau implementations. Numerical results on a set of problems illustrate the impact of the mathematical techniques introduced.

Apresentações em conferências internacionais

- 2014 - *A Lanczos' tau method software library for the solution of differential equations* - 2nd International Conference on Numerical and Symbolic Computation (**Universidade do Algarve, Faro - Portugal**).
- 2015 - *An iterative implementation of the tau method based on Schur complements* - New Directions in Numerical Computation (**Universidade de Oxford, Oxford - Inglaterra**).
- 2016 - *Spectral tau method for systems of nonlinear integro-differential equations* - 14th International Conference on Integral Methods in Science and Engineering (**Universidade de Padova, Padova - Itália**).
- 2017 - *Dealing with non-polynomial coefficients within tau method* - 3nd International Conference on Numerical and Symbolic Computation (**Universidade do Minho, Guimarães - Portugal**).

Software

- `Tau Toolbox` - Biblioteca de rotinas em `MATLAB` para a aproximação, via método tau, de problemas diferenciais e integro-diferenciais, lineares e não lineares.

Prêmio

- Wolfram Research Award - Atribuído pela Wolfram para melhor artigo, no Symcomp'2017.

Parte I

Preliminares e revisão bibliográfica

Capítulo 2

Polinómios ortogonais

Sumário

As sequências de polinómios ortogonais associadas aos nomes Hermite, Laguerre, Bessel e Jacobi (incluindo os casos especiais posteriormente nomeados como Chebyshev, Legendre e Gegenbauer) são amplamente as mais estudadas e aplicadas e constituem coletivamente as famílias de polinómios ortogonais clássicas. Neste capítulo fortemente baseado em Szego (1939), Abramowitz and Stegun (1972) e Chihara (1978), são apresentadas algumas definições e teoremas relacionados com a teoria dos polinómios ortogonais.

2.1 Preliminares

Considerando o espaço $\mathbb{P}_n$ de todas as combinações lineares dos polinómios de grau $r \leq n$, cujos elementos podem ser escritos como $Z_r = \sum_{i=0}^{r} a_{ri}x^i,\ a_{rr} \neq 0$, começamos por apresentar as duas definições referentes a um subconjunto de polinómios $\mathcal{Z} = [Z_0,\ Z_1,\ \ldots] \in \mathbb{P}_n$ que estabelecem uma relação de ortogonalidade.

Definição 2.1. *Seja $\mathcal{Z} = [Z_0,\ Z_1,\ \ldots]$ uma sequência de polinómios e $w \geq 0$ uma função peso definida no intervalo $[a,\ b],\ a < b \in \mathbb{R}$. Diz-se que $\mathcal{Z}$ constitui uma sequência de polinómios ortogonais se Z_k tem grau exatamente k e se*

$$\left\langle Z_p, Z_q \right\rangle_w = \int_a^b Z_p Z_q w dx = h_p \delta_{pq}, \quad \forall p,\ q,$$

com $h_p = ||Z_p||^2 > 0$ e δ_{pq} a função delta de Kronecker.

Os valores h_p da norma do polinómio Z_p associada à função peso w dependem de uma condição de normalização. Dependendo dos autores, esta condição pode consistir

em impor $h_p = 1, \forall p \geq 0$, e neste caso falamos nos polinómios ortonormados; ou impor que o coeficiente principal de Z_n é unitário, e neste caso os polinómios dizem-se mónicos. Outros autores normalizam os polinómios fixando outros parâmetros das propriedades de ortogonalização. Nesta Tese seguimos a normalização dos polinómios ortogonais apresentada em Abramowitz and Stegun (1972).

Uma sequência de polinómios ortogonais é necessariamente linearmente independente (Szego, 1939). A Definição seguinte fornece os coeficientes da representação de uma função arbitrária como série formal em termos dos elementos de $\mathcal{Z}$.

Definição 2.2. *Seja $\mathcal{Z} = [Z_0,\ Z_1,\ \ldots]$ uma sequência de polinómios ortogonais. A uma função real arbitrária y, corresponde a série formal de Fourier $y \sim \sum_{i=0}^{\infty} a_i Z_i$, onde*

$$a_n = \frac{1}{h_n}\left\langle y, Z_n \right\rangle_w = \frac{1}{h_n}\int_a^b y Z_n w dx,\ n \geq 0$$

são designados por coeficientes de Fourier de y com respeito à sequência $\mathcal{Z}$.

É apresentada no Teorema seguinte, uma importante propriedade das somas parciais das séries de Fourier, que constitui a propriedade de minimização do erro segundo a norma associada ao produto interno, geral aos polinómios ortogonais.

Teorema 2.1. *(Szego, 1939) Seja $\mathcal{Z} = [Z_0,\ Z_1,\ \ldots]$ uma sequência de polinómios ortogonais e $y_n = \sum_{i=0}^{n} a_i Z_i$ a soma parcial da série de Fourier y, então*

$$\int_a^b (y - y_n)^2 w dx \leq \int_a^b (y - p_n)^2 w dx,\ \forall p_n \in \mathbb{P}_n.$$

O resultado deste Teorema significa que o polinómio de Fourier y_n satisfaz a propriedade de minimização

$$||y - y_n||^2 \leq ||y - p_n||^2,$$

para qualquer polinómio p_n de grau menor ou igual a n.

Teorema 2.2. *Seja $\mathcal{Z} = [Z_0,\ Z_1,\ \ldots]$ uma sequência de polinómios ortogonais como estabelecida na Definição 2.1, e seja Q_k um polinómio de grau k. Nestas condições, é verdadeira a condição: se $n > k$, então Z_n é ortogonal a Q_k com respeito a w.*

Demonstração. Ver Szego (1939). □

O Teorema seguinte, garantindo que os polinómios ortogonais maximizam o número de zeros no intervalo de ortogonalidade, dá sentido à versão operacional do método tau, como apresentado no Capítulo 4. Ao impor que o resíduo, resultante da aplicação do método tau, é ortogonal aos primeiros elementos de uma sequência de polinómios ortogonais, procura-se que o resíduo no problema diferencial tenha o número máximo de zeros possível no intervalo de aproximação.

Teorema 2.3. *Seja $\mathcal{Z} = [Z_0,\ Z_1,\ \ldots]$ uma sequência de polinómios ortogonais como estabelecida na Definição 2.1. Z_n possui n raízes reais simples em $[a,\ b]$.*

Demonstração. Ver Szego (1939). □

O Teorema seguinte estabelece uma base quer para o cálculo dos aproximantes, quer para toda a metodologia de implementação do método tau utilizada nesta Tese. No Capítulo 5 são apresentados contributos para a estabilidade no método tau, desenvolvidos no âmbito desta Tese, e que são decisivos na qualidade dos resultados numéricos obtidos na implementação da `Tau Toolbox`.

Teorema 2.4. *Seja* $\mathcal{Z} = [Z_0,\ Z_1,\ \ldots]$ *uma sequência de polinómios ortogonais, então existe uma relação de recorrência a três termos associada*

$$\begin{cases} xZ_n = \alpha_n Z_{n+1} + \beta_n Z_n + \gamma_n Z_{n-1}, \quad n \geq 0 \\ Z_0 = 1, \quad Z_{-1} = 0 \end{cases}, \tag{2.1}$$

onde α_n, β_n *e* γ_n *são constantes reais dependentes apenas de* n.

Demonstração. Ver Chihara (1978). □

A Definição seguinte define os polinómios ortogonais clássicos que são os utilizados no âmbito desta Tese devido às suas propriedades.

Definição 2.3. *Seja* $\mathcal{Z} = [Z_0,\ Z_1,\ \ldots]$ *uma sequência de polinómios ortogonais e* $\mathcal{Q} = [Q_0,\ Q_1,\ \ldots]$, *com*

$$Q_n = \frac{d}{dx} Z_{n+1}.$$

Dizemos que $\mathcal{Z}$ *é uma sequência de polinómios ortogonais clássicos se e somente se* $\mathcal{Q}$ *também for uma sequência de polinómios ortogonais (Hahn, 1935; Suetin, 2001).*

As boas propriedades de aproximação espetral estão associadas aos polinómios ortogonais clássicos: (i) do Teorema 2.3 resulta que os zeros estão distribuídos no intervalo de ortogonalidade, e isto é vantajoso por tornar nulo o erro da aproximação nestes pontos e (ii) da Definição 2.3, resulta a vantagem no uso dos polinómios ortogonais clássicos no sentido de que em aproximação de problemas diferenciais, o item (i), é válido também para as sucessivas derivadas dos polinómios.

A Secção a seguir ocupa-se de apresentar as famílias de polinómios ortogonais utilizas nesta Tese.

2.2 Polinómios ortogonais clássicos

Os polinómios ortogonais clássicos satisfazem algumas importantes relações que obedecem à mesma forma geral, sendo as mais relevantes delas:

(i) são solução de uma equação diferencial da forma: $g_2 \frac{d^2}{dx^2} Z_n + g_1 \frac{d}{dx} Z_n + \rho_n Z_n = 0$, onde g_2 e g_1 são polinómios independentes de n e ρ_n é uma constante dependente somente de n;

(ii) possuem uma fórmula de Rodrigues associada: $Z_n = \frac{1}{\rho_n w}\frac{d^n}{dx^n}\Big(wg^n\Big)$, onde g é um polinómio em x independente de n e ρ_n é uma constante;

(iii) possuem uma função geradora: $G(x,t) = \sum_{n=0}^{\infty} a_n Z_n t^n$, onde a_n é uma sequência de constantes e

(iv) relacionam-se uns com os outros através de uma relação de recorrência a três termos: $xZ_n = \alpha_n Z_{n+1} + \beta_n Z_n + \gamma_n Z_{n-1},\ n \geq 0,\ Z_0 = 1,\ Z_{-1} = 0.$

O objetivo desta secção é apresentar resumidamente as famílias de polinómios ortogonais clássicos utilizadas: Jacobi (inclui Chebyshev e Legendre), Hermite, Laguerre e Bessel, visando principalmente conhecer as formas ((ii) e (iv)) de calcular a matriz dos coeficientes $\mathsf{V}_{\mathcal{Z}}$, que satisfaz $\mathcal{X}\mathsf{V}_{\mathcal{Z}} = \mathcal{Z}$, com $\mathcal{X} = [1,\ x,\ x^2,\ \ldots]$.

Definição 2.4. *A sequência de polinómios* $\mathcal{J} = [J_0^{(\alpha,\beta)},\ J_1^{(\alpha,\beta)},\ \ldots]$ *definida por*

$$J_n^{(\alpha,\beta)} = \frac{1}{(-1)^n 2^n n!(1-x)^\alpha(1+x)^\beta}\frac{d^n}{dx^n}\Big((1-x)^{n+\alpha}(1+x)^{n+\beta}\Big) \tag{2.2}$$

designa-se por polinómios de Jacobi associados aos parâmetros α *e* β *(Abramowitz and Stegun, 1972).*

Ao aplicar a formula de Leibniz (ver Chihara (1978)) em (2.2) para a n-ésima derivada do produto, obtemos a forma explícita dos polinómios de Jacobi:

$$J_n^{(\alpha,\beta)} = \frac{1}{2^n}\sum_{k=0}^{n}\binom{n-\alpha}{n-k}\binom{n+\beta}{k}(x-1)^k(x+1)^{n-k}.$$

Em Chihara (1978) e Szego (1939) encontramos a relação de recorrência a três termos para os polinómios de Jacobi na forma

$$F(n,\alpha,\beta)J_n^{(\alpha,\beta)} = G(n,\alpha,\beta,x)J_{n-1}^{(\alpha,\beta)} - H(n,\alpha,\beta)J_{n-2}^{(\alpha,\beta)} \tag{2.3}$$

onde

$$F(n,\alpha,\beta) = 2n(n+\alpha+\beta)(2n+\alpha+\beta-2),$$
$$G(n,\alpha,\beta,x) = (2n+\alpha+\beta-1)\Big((2n+\alpha+\beta)(2n+\alpha+\beta-2)x + \alpha^2 - \beta^2\Big)$$

e

$$H(n,\alpha,\beta) = 2(n+\alpha-1)(n+\beta-1)(2n+\alpha+\beta),$$

e portanto, com algumas manipulações em (2.3), chegamos aos coeficientes α_n, β_n e γ_n da relação de recorrência

$$xJ_n^{(\alpha,\beta)} = \alpha_n J_{n+1}^{(\alpha,\beta)} + \beta_n J_n^{(\alpha,\beta)} + \gamma_n J_{n-1}^{(\alpha,\beta)},$$

onde

$$\alpha_n = \frac{2(n+1)(n+\alpha+\beta+1)}{(2n+\alpha+\beta+1)(2n+\alpha+\beta+2)}$$
$$\beta_n = -\frac{\alpha^2-\beta^2}{(2n+\alpha+\beta+2)(2n+\alpha+\beta)},$$

e

$$\gamma_n = \frac{2(n+\alpha)(n+\beta)}{(2n+\alpha+\beta+1)(2n+\alpha+\beta)}.$$

Considerando o caso especial dos polinómios da família de Jacobi $J_n^{(\alpha,\beta)}$ em que $\alpha = \beta = -\frac{1}{2}$ e em que $\alpha = \beta = \frac{1}{2}$, temos os polinómios de Chebyshev:

Definição 2.5. *As sequências de polinómios* $\mathcal{T} = [T_0,\ T_1,\ \ldots]$, *e* $\mathcal{U} = [U_0,\ U_1,\ \ldots]$, *definidas por*

$$T_n = \frac{(-1)^n\sqrt{(1-x^2)\pi}}{2^n\Gamma\left(n+\frac{1}{2}\right)}\frac{d^n}{dx^n}(1-x^2)^{n-\frac{1}{2}} \tag{2.4}$$

e

$$U_n = \frac{(-1)^n(n+1)\sqrt{\pi}}{2^{n+1}\Gamma\left(n+\frac{3}{2}\right)\sqrt{(1-x^2)}}\frac{d^n}{dx^n}(1-x^2)^{n+\frac{1}{2}}. \tag{2.5}$$

designam-se, respetivamente, por polinómios de Chebyshev de 1ª espécie e polinómios de Chebyshev de 2ª espécie, e as equações (2.4) e (2.5) são, respetivamente, as fórmulas de Rodrigues para a obtenção dos mesmos (Abramowitz and Stegun, 1972).

Considerando o caso especial dos polinómios da família de Jacobi $J_n^{(\alpha,\beta)}$ em que $\alpha = \beta = 0$ obtemos os polinómios de Legendre:

Definição 2.6. *A sequência de polinómios* $\mathcal{P} = [P_0,\ P_1,\ \ldots]$ *definida por*

$$P_n = \frac{(-1)^n}{2^n n!}\frac{d^n}{dx^n}(1-x^2)^n. \tag{2.6}$$

designa-se por polinómios de Legendre, e (2.6) é a fórmula de Rodrigues para a sua obtenção (Abramowitz and Stegun, 1972).

Definição 2.7. *A sequência de polinómios* $\mathcal{H} = [H_0,\ H_1,\ \ldots]$ *definida por*

$$H_n = (-1)^n e^{x^2}\frac{d^n}{dx^n}\frac{1}{e^{x^2}}. \tag{2.7}$$

designa-se por polinómios de Hermite, e a equação (2.7) é a fórmula de Rodrigues para a sua obtenção (Abramowitz and Stegun, 1972).

Definição 2.8. *A sequência de polinómios* $\mathcal{L} = [L_0,\ L_1,\ \ldots]$ *definida por*

$$L_n = \frac{e^x}{n!}\frac{d^n}{dx^n}\frac{x^n}{e^x}. \tag{2.8}$$

designa-se por polinómios de Laguerre, e a equação (2.8) é a fórmula de Rodrigues para a obtenção dos mesmos (Abramowitz and Stegun, 1972).

Definição 2.9. *A sequência de polinómios* $\mathcal{Y} = [Y_0,\ Y_1,\ \ldots]$ *definida por*

$$Y_n = \frac{e^{\frac{2}{x}}}{2^n}\frac{d^n}{dx^n}\frac{x^{2n}}{e^{\frac{2}{x}}}. \tag{2.9}$$

designa-se por polinómios de Bessel, e a equação (2.9) é a fórmula de Rodrigues para a sua obtenção.

As fórmulas de Rodrigues acima apresentadas são úteis para se calcular isoladamente o j-ésimo polinómio de uma determinada família. Na necessidade de construir uma sequência de polinómios $\mathcal{Z} = [Z_0,\ Z_1,\ \ldots]$, é mais vantajoso aplicar a relação de recorrência, por evitar o cálculo analítico de muitas derivadas, fundamental para efeitos computacionais. Uma forma de obter a relação de recorrência associada a uma família de polinómios ortogonais é a partir de sua função geradora. Para ilustrar mostraremos este processo para a família de Legendre, sendo as demais análogas.

Dada a função geradora dos polinómios de Legendre

$$G(x,t) = \frac{1}{\sqrt{1-2xt+t^2}} = \sum_{n=0}^{\infty} P_n t^n, \quad 0 \leq t < 1, \quad -1 \leq x \leq 1, \tag{2.10}$$

começamos por derivar a equação (2.10) em relação a t, obtendo

$$\frac{\partial}{\partial t} G(x,t) = \frac{x-t}{(1-2xt+t^2)\sqrt{1-2xt+t^2}}, \tag{2.11}$$

e, multiplicando ambos os lados de (2.11) por $(1-2xt+t^2)$ e substituindo (2.10) na mesma, obtemos

$$(1-2xt+t^2)\frac{\partial}{\partial t} G(x,t) = (x-t)G(x,t),$$

portanto, facilmente percebemos que

$$(1-2xt+t^2)\sum_{n=1}^{\infty} nP_n t^{n-1} = (x-t)\sum_{n=0}^{\infty} P_n t^n,$$

donde por distributividade

$$\sum_{n=1}^{\infty} nP_n t^{n-1} - 2\sum_{n=1}^{\infty} nxP_n t^n + \sum_{n=1}^{\infty} nP_n t^{n+1} = \sum_{n=0}^{\infty} xP_n t^n - \sum_{n=0}^{\infty} P_n t^{n+1};$$

reindexando a variável t

$$\sum_{n=0}^{\infty}(n+1)P_{n+1}t^n - 2\sum_{n=1}^{\infty} nxP_n t^n + \sum_{n=2}^{\infty}(n-1)P_{n-1}t^n = \sum_{n=0}^{\infty} xP_n t^n - \sum_{n=1}^{\infty} P_{n-1}t^n,$$

e reagrupando coeficientes, encontramos

$$(n+1)P_{n+1} - (2n+1)xP_n + nP_{n-1} = 0, \quad \forall n \geq 1.$$

Assim, a relação de recorrência a três termos para os polinómios de Legendre é dada por

$$\begin{cases} xP_n = \dfrac{n+1}{2n+1}P_{n+1} + \dfrac{n}{2n+1}P_{n-1}, & n \geq 1 \\ P_0 = 1,\ P_1 = x \end{cases}.$$

A Tabela 2.1 contém os coeficientes α_n, β_n e γ_n da relação de recorrência a três termos para as famílias de polinómios ortogonais utilizadas. Também são apresentados os elementos iniciais e o intervalo de ortogonalidade destas famílias.

$\mathcal{Z}$	α_n	β_n	γ_n	Z_1	ortogonalidade
$\mathcal{T}$	$\frac{1}{2}$	0	$\frac{1}{2}$	$Z_1 = x$	$[-1,\ 1]$
$\mathcal{U}$	$\frac{1}{2}$	0	$\frac{1}{2}$	$Z_1 = 2x$	$[-1,\ 1]$
$\mathcal{P}$	$\frac{n+1}{2n+1}$	0	$\frac{n}{2n+1}$	$Z_1 = x$	$[-1,\ 1]$
$\mathcal{H}$	$\frac{1}{2}$	0	n	$Z_1 = 2x$	$[-\infty,\ +\infty]$
$\mathcal{L}$	-1	$2n+1$	n^2	$Z_1 = 1 - x$	$[0,\ +\infty]$
$\mathcal{Y}$	$\frac{1}{2n+1}$	0	$-\frac{1}{2n+1}$	$Z_1 = 1 + x$	Círculo unitário

Tabela 2.1: Coeficientes da relação de recorrência a três termos para as famílias de polinómios ortogonais utilizadas (em todos os casos $Z_0 = 1$).

As matrizes abaixo ilustram os coeficientes de $\mathsf{V}_{\mathcal{Z}},\ \mathcal{Z} = \mathcal{T},\ \mathcal{U},\ \mathcal{P},\ \mathcal{H}\ , \mathcal{L},\ \mathcal{Y}$.

$$\mathsf{V}_{\mathcal{T}} = \begin{bmatrix} 1 & 0 & -1 & 0 & 1 & 0 & \dots \\ & 1 & 0 & -3 & 0 & 5 & \dots \\ & & 2 & 0 & -8 & 0 & \dots \\ & & & 4 & 0 & -20 & \dots \\ & & & & 8 & 0 & \dots \\ & & & & & 16 & \dots \\ & & & & & & \ddots \end{bmatrix}, \qquad \mathsf{V}_{\mathcal{U}} = \begin{bmatrix} 1 & 0 & -1 & 0 & 1 & 0 & \dots \\ & 2 & 0 & -4 & 0 & 6 & \dots \\ & & 4 & 0 & -12 & 0 & \dots \\ & & & 8 & 0 & -32 & \dots \\ & & & & 16 & 0 & \dots \\ & & & & & 32 & \dots \\ & & & & & & \ddots \end{bmatrix},$$

$$\mathsf{V}_{\mathcal{P}} = \begin{bmatrix} 1 & 0 & -\frac{1}{2} & 0 & \frac{3}{8} & 0 & \dots \\ & 1 & 0 & -\frac{3}{2} & 0 & \frac{15}{8} & \dots \\ & & \frac{3}{2} & 0 & -\frac{15}{4} & 0 & \dots \\ & & & \frac{5}{2} & 0 & -\frac{35}{4} & \dots \\ & & & & \frac{35}{8} & 0 & \dots \\ & & & & & \frac{63}{8} & \dots \\ & & & & & & \ddots \end{bmatrix}, \qquad \mathsf{V}_{\mathcal{H}} = \begin{bmatrix} 1 & 0 & -2 & 0 & 12 & 0 & \dots \\ & 2 & 0 & -12 & 0 & 120 & \dots \\ & & 4 & 0 & -48 & 0 & \dots \\ & & & 8 & 0 & -160 & \dots \\ & & & & 16 & 0 & \dots \\ & & & & & 32 & \dots \\ & & & & & & \ddots \end{bmatrix},$$

$$\mathsf{V}_{\mathcal{L}} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & \dots \\ & -1 & -2 & -3 & -4 & -5 & \dots \\ & & \frac{1}{2} & \frac{3}{2} & 3 & 5 & \dots \\ & & & -\frac{1}{6} & -\frac{2}{3} & -\frac{5}{3} & \dots \\ & & & & \frac{1}{24} & \frac{5}{24} & \dots \\ & & & & & -\frac{1}{120} & \dots \\ & & & & & & \ddots \end{bmatrix}, \qquad \mathsf{V}_{\mathcal{Y}} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & \dots \\ & 1 & 3 & 6 & 10 & 15 & \dots \\ & & 3 & 15 & 45 & 105 & \dots \\ & & & 15 & 105 & 420 & \dots \\ & & & & 105 & 945 & \dots \\ & & & & & 945 & \dots \\ & & & & & & \ddots \end{bmatrix}.$$

2.3 Mudança de intervalo de ortogonalidade

Em algumas aplicações, inclusive no método tau, pode ser necessário usar polinómios ortogonais no intervalo $[0,\ 1]$ ou em algum outro intervalo $[a,\ b]$, assim há a

necessidade de realizarmos uma mudança de intervalo de ortogonalidade. Em relação ao intervalo $[-1,\ 1]$ por exemplo, a mudança da ortogonalidade para $[0,\ 1]$ é obtida com a substituição $x = 2x^* - 1$. Um problema que se pode colocar é como obter os coeficientes dos polinómios na variável x^* a partir dos coeficientes do polinómio em x. Esta mudança de coeficientes pode obter-se seguindo um esquema recursivo (Abramowitz and Stegun, 1972). Se

$$Z_n = \sum_{m=0}^{n} a_m x^m, \qquad Z_n^* = Z_n\Big|_{2x-1} = \sum_{m=0}^{n} a_m^* x^m,$$

então os coeficientes a_m^* são dados recursivamente a partir dos coeficientes a_m através das relações:

$$a_m^{(j)} = 2a_m^{(j-1)} - a_{m+1}^{(j)}, \quad m = n-1(-1)j, \quad j = 0(1)n$$

$$a_m^{(-1)} = \frac{a_m}{2}, \quad m = 0(1)n$$

$$a_n^{(j)} = 2^j a_n, \quad j = 0(1)n; \quad a_m^* = a_m^{(m)}, \quad m = 0(1)n$$

Generalizando é possível definir polinómios ortogonais em qualquer intervalo finito $[a,\ b]$ com $b > a$ a partir de uma família de polinómios ortogonais definida num intervalo finito. Em particular, é possível definir os polinómios das famílias de Chebyshev e Legendre, bem como de outras de Jacobi, ao utilizar a transformação

$$Z_n = \sum_{m=0}^{n} a_m x^m, \qquad Z_n^* = Z_n\Big|_{\frac{2x-(a+b)}{b-a}} = \sum_{m=0}^{n} a_m^* x^m. \tag{2.12}$$

Formulação 1

A transformação (2.12) faz com que cada Z_n tenha a variável x substituída por uma expressão que conduz, para cada potência k, à formulação

$$\left(\frac{2x-(a+b)}{b-a}\right)^k = \sum_{i=0}^{k} \frac{k!}{i!(k-i)!}\left(\frac{2x}{b-a}\right)^{k-i}\left(-\frac{a+b}{b-a}\right)^i.$$

Esta abordagem é útil se a matriz $\mathsf{V}_{\mathcal{Z}}$ já foi construída para a base $\mathcal{Z}$.

Formulação 2

Caso a matriz $\mathsf{V}_{\mathcal{Z}}$ ainda esteja por ser construída, podemos então desde já aplicar a transformação (2.12) à relação de recorrência, e então gerar os polinómios ortogonais em $[a,\ b]$ diretamente, ou seja

$$Z_{n+1}\Big|_{x=\frac{2x-(a+b)}{b-a}} = \frac{\left(\frac{2x-(a+b)}{b-a} - \beta_n\right)Z_n\Big|_{\frac{2x-(a+b)}{b-a}} - \gamma_n Z_{n-1}\Big|_{\frac{2x-(a+b)}{b-a}}}{\alpha_n},$$

e uma vez considerando $x = \dfrac{2x^* - (a+b)}{b-a}$, apenas quando argumento, temos

$$Z_{n+1}^* = \left(\frac{2x^*-(a+b)}{b-a} - \beta_n\right)Z_n^*\alpha_n^{-1} - \gamma_n Z_{n-1}^*\alpha_n^{-1},$$

Algoritmo 1: Mudança de intervalo de ortogonalidade.

Dados: valores a, b e matriz $\mathsf{V}_{\mathcal{Z}}$ ortogonal no domínio $[-1,\ 1]$.
Resultado: matriz $\mathsf{V}^*_{\mathcal{Z}}$ ortogonal no domínio $[a,\ b]$.
início
- $t_1 \leftarrow 2/(b-a)$
- $t_2 \leftarrow -(a+b)/(b-a)$
- $n \leftarrow$ tamanho de $\mathsf{V}_{\mathcal{Z}}$
- $\mathsf{P} \leftarrow$ triângulo de pascal no formato de matriz triangular superior
- **para** $r = 1 : n$ **faça**
  - $\mathsf{t} \leftarrow [\underbrace{0,\ 0,\ \ldots, 0}_{n \text{ vezes}}]^T$
  - **para** $n = 1 : r-1$ **faça**
    - $\mathsf{c} \leftarrow [\underbrace{0,\ 0,\ \ldots,\ 0}_{n+1 \text{ vezes}}]^T$
    - **se** $\mathsf{V}_{\mathcal{Z}n+1,r} \neq 0$ **então**
      - **para** $i = 0 : n$ **faça**
        - $\mathsf{c}_{i+1} \leftarrow \mathsf{p}_{n+1,i+1} t_1^{n-1} t_2^i$
      - $\mathsf{t}_{1:n+1} \leftarrow \mathsf{t}_{1:n+1} + \mathsf{V}_{\mathcal{Z}n+1,r}\mathsf{c}_{n+1:-1:1}$
  - $\mathsf{V}^*_{\mathcal{Z}1:n+1,r} = \mathsf{t}$
- $\mathsf{V}^*_{\mathcal{Z}1,1:n} = \mathsf{V}^*_{\mathcal{Z}1,1:n} + \mathsf{V}_{\mathcal{Z}1,1:n}$
- Mostra $\mathsf{V}^*_{\mathcal{Z}}$

ou então, organizando na forma $x^*Z_n = \alpha_n^* Z_{n+1}^* + \beta_n^* Z_n^* + \gamma_n^* Z_{n-1}^*$, onde os novos coeficientes da relação a três termos são dados, respetivamente, por

$$\alpha_n^* = \frac{(b-a)\alpha_n}{2} \quad \beta_n^* = \frac{(b-a)\beta_n + a + b}{2} \quad \text{e} \quad \gamma_n^* = \frac{(b-a)\gamma_n}{2}.$$

Esta relação permite calcular por recorrência as colunas da matriz $\mathsf{V}^*_{\mathcal{Z}}$, relativa à projeção dos polinómios ortogonais $\mathcal{Z}^*$ definidos em $[a,\ b]$, a partir dos coeficientes da relação de recorrência dos polinómios associados à mesma família e definidos no seu intervalo padrão.

2.4 Conclusões e considerações

Neste Capítulo foram apresentados alguns resultados sobre a teoria dos polinómios ortogonais. A fórmula de Rodrigues e a recorrência a três termos para as bases de polinómios ortogonais clássicos de Jacobi (Chebyshev e Legendre), Hermite, Laguerre e Bessel foram sugeridas como formas de geração dos elementos destas famílias. Duas formas de mudança de intervalo de ortogonalidade foram propostas para os polinómios ortogonais no intervalo $[-1,\ 1]$.

Capítulo 3

Métodos numéricos espetrais

Sumário

Há essencialmente duas classes de métodos numéricos aplicáveis à obtenção de uma aproximação da solução de equações diferenciais e integrais. A primeira delas é a que trata de métodos numéricos discretos, ou seja, dada uma equação diferencial e suas condições de contorno, a ideia dessa classe de métodos consiste em discretizar o domínio da solução em pontos, muitas vezes chamados nós, e então em cada um destes obter um valor aproximado da solução. Nessa classe de métodos, na qual se enquadram métodos conhecidos como Euler progressivo e regressivo, Runge-Kutta, diferenças finitas entre outros, os resultados aproximados à solução são vetores, ou seja, valores numéricos que pontualmente aproximam a solução. A outra classe, que é foco desta Tese, trata dos métodos numéricos espetrais, que por sua vez abordam a aproximação da solução do problema fazendo uso de funções base, isto é, a aproximação calculada é agora uma função, e não um vetor pontual. Para este capítulo são usadas as referências Gottlieb and Orszag (1977); Canuto et al. (2010); Trefethen (2000, 2013); Boyd (2001), onde são abordados de forma mais aprofundada os elementos necessários à compreensão de métodos espetrais.

As funções para aproximar uma solução exata em um método espetral são geralmente chamadas de funções teste, e constituem uma combinação linear de funções base previamente escolhidas de forma a satisfazer a equação diferencial e/ou integral e, eventualmente, as condições de contorno. A ideia na base desta classe de métodos é minimizar a função resíduo em relação a uma norma. Neste sentido, métodos espetrais são considerados casos especiais do método dos resíduos ponderados. Por outro lado, por estarem baseados em impor que o resíduo satisfaça uma condição de ortogonalidade em relação a uma função peso, também podem ser considerados como casos especiais dos métodos de Galerkin-Petrov (Finlayson and Scriven, 1966; Zienkiewicz and Cheung, 1967). Para ilustrar o funcionamento dos métodos espetrais, apresentamos a seguir o método da colocação e o método de Galerkin.

3.1 Método da Colocação

O método da colocação é um método que serve para a obtenção da solução numérica de equações diferenciais ordinárias, equações diferenciais parciais e equações integro-diferenciais. A ideia principal deste método está em escolher um conjunto de soluções candidatas e um número de pontos no domínio, os chamados pontos de colocação, e com estes elementos aproximar os coeficientes das funções candidatas que anulam o resíduo nos pontos de colocação. i.e., a solução satisfaz o problema nos pontos de colocação. Seja

$$\mathfrak{F}y = f, \quad x \in]a,\ b[\tag{3.1}$$

uma equação diferencial ordinária com $\mathfrak{F}$ sendo o operador diferencial definido por

$$\mathfrak{F} = \sum_{i=0}^{m} f_i \frac{d^i}{dx^i},$$

onde m denota a ordem da equação e f_i são os coeficientes polinomiais. Para o problema (3.1) consideramos ainda as condições de contorno

$$g_j(y) = \sigma_j, \quad j = 1(1)m,$$

que podem ser de Neumann, Dirichlet ou mistas.

Consideramos uma aproximação y_n na forma

$$y_n = v + \sum_{k=0}^{n} a_k u_k = v + \mathsf{ua}, \tag{3.2}$$

onde $\mathsf{a} = [a_0,\ \ldots,\ a_n]^T$ são coeficientes reais a determinar, v e u_k são funções que satisfazem as condições de contorno e $\mathsf{u} = [u_0,\ \ldots,\ u_n]$.

Para determinar os valores a_k aplicamos o operador diferencial $\mathfrak{F}$ a y_n e em seguida impomos condições de forma a ter um resíduo próximo de zero, ou seja

$$\begin{aligned} R(x, \mathsf{a}) &= \mathfrak{F}y_n - f \\ &= \mathfrak{F}\left(v + \sum_{k=1}^{n} a_k u_k\right) - f \\ &\approx 0. \end{aligned}$$

Uma forma de procurar alcançar esse objetivo consiste em escolher n pontos $\in [a,\ b]$, chamados pontos de colocação $(x_p),\ p = 1(1)n$, e, fazendo uso da função delta Dirac

$$w_p = \delta(x - x_p),$$

aplicada da forma

$$\int_a^b w_p R dx = R(x_p) = 0,$$

impor a nulidade do resíduo nos n pontos de colocação.

Como (neste caso ilustrativo) o número de coeficientes coincide com o número de pontos de colocação x_p obtemos o sistema quadrado

$$\begin{bmatrix} \mathfrak{F}u_1(x_1) & \mathfrak{F}u_2(x_1) & \dots & \mathfrak{F}u_n(x_1) \\ \mathfrak{F}u_1(x_2) & \mathfrak{F}u_2(x_2) & \dots & \mathfrak{F}u_n(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \mathfrak{F}u_1(x_n) & \mathfrak{F}u_2(x_n) & \dots & \mathfrak{F}u_n(x_n) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} f(x_1) - \mathfrak{F}v(x_1) \\ f(x_2) - \mathfrak{F}v(x_2) \\ \vdots \\ f(x_n) - \mathfrak{F}v(x_n) \end{bmatrix} \tag{3.3}$$

para encontrar as constantes a_k, $k = 1(1)n$, onde os coeficientes da matriz $\mathsf{A} = [a_{i,j}]_{n\times n}$ e do vetor dos termos independente $\mathsf{b} = [b_i]_{n\times 1}$ são dados respetivamente por

$$a_{i,j} = \int_a^b \delta(x - x_i)\mathfrak{F}u_j dx = \mathfrak{F}u_j(x_i), \quad i, j = 1(1)n$$

e

$$b_i = \int_a^b \delta(x - x_i)\Big(f - \mathfrak{F}v\Big)dx = f(x_i) - \mathfrak{F}v(x_i), \quad i = 1(1)n.$$

Exemplo 3.1. Resolver o problema de valor de fronteira

$$\begin{cases} \dfrac{d^2}{dx^2}y - y = 0, \quad x \in]0,\ 1[\\ y(0) = 0,\ y(1) = 1 \end{cases}.$$

Solução. Começamos por escolher a quantidade de pontos de colocação dentro do domínio $[0,\ 1]$; neste exemplo escolhemos $n = 5$, e as funções v e u_k tais que $v + u_k$ satisfaçam as condições iniciais do problema

$$v = x \quad \text{e} \quad u_k = x^k(x - 1), \quad k = 1(1)5;$$

uma vez que

$$y_5 = v + \sum_{k=1}^{5} a_k u_k = x + \sum_{k=1}^{5} a_k x^k(x - 1), \tag{3.4}$$

então

$$\begin{cases} y_5(0) = v(0) + u_k(0) = 0 \\ y_5(1) = v(1) + u_k(1) = 1 \end{cases}, \quad k = 1(1)5.$$

Como

$$\frac{d^2}{dx^2}u_k = (k^2 + k)x^{k-1} - (k^2 - k)x^{k-2},$$

temos, para $k = 1(1)5$, que

$$\mathfrak{F}u_k = \frac{d^2}{dx^2}u_k - u_k = (k^2 + k)x^{k-1} - (k^2 - k)x^{k-2} - x^k(x - 1), \tag{3.5}$$

e uma vez que

$$\frac{d^2}{dx^2}v = 0,$$

temos que

$$\mathfrak{F}v = \frac{d^2}{dx^2}v - v = -x. \tag{3.6}$$

Alocamos ao domínio $[0,\ 1]$ os 5 pontos de colocação, neste caso igualmente espaçados

$$\mathsf{x} = \left[\frac{1}{6},\ \frac{1}{3},\ \frac{1}{2},\ \frac{2}{3},\ \frac{5}{6}\right], \tag{3.7}$$

e portanto, com (3.5) e (3.6) nos pontos (3.7), o sistema (3.3) para este problema é escrito (arredondando-se a 4 casas decimais) como

$$\begin{bmatrix} 2,1389 & -0,9769 & -0,6628 & -0,2401 & -0,0693 \\ 2,2222 & 0,0741 & -0,6420 & -0,5844 & -0,3676 \\ 2,2500 & 1,1250 & 0,0625 & -0,4688 & -0,6094 \\ 2,2222 & 2,1481 & 1,4321 & 0,6584 & 0,0439 \\ 2,1389 & 3,1157 & 3,4298 & 3,3211 & 2,9605 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} 0,1667 \\ 0,3333 \\ 0,5000 \\ 0,6667 \\ 0,8333 \end{bmatrix},$$

cuja solução é

$$\mathsf{a} = [0,1491,\ 0,1490,\ 0,0075,\ 0,0068,\ 0,0006]^T. \tag{3.8}$$

Assim, substituindo (3.8) em (3.4) temos uma aproximação y_5 com erro absoluto $\max|y_5 - y| = 2,1458 \cdot 10^{-7}$ para a solução analítica

$$y = \frac{e^x - e^{-x}}{e - e^{-1}}.$$

O Código 3.1 calcula e representa graficamente as soluções de colocação, utilizando este processo, com números de pontos $n = 5, 10, 15$ e 20.

Código 3.1. Método da colocação para o Exemplo 3.1.

```
% Número de pontos de colocação: 5, 10, 15 e 20.
for n = 5:5:20
    clear a

    % Domínio e passo.
    J = [0, 1]; passo = (J(2)-J(1))/(n+1);
    x = passo:passo:J(2) - passo;

    % Reserva de memória.
    A = zeros(n);

    % Construindo A e b do sistema Ax=b.
    x = x';
    A(:, 1) = 2-x.*(x-1);
    for k = 1:n-1
        A(:, k+1) = A(:, k).*x+2*x.^(k-1).*((k+1)*x-k);
    end
    b = x;

    % Obtendo a solução.
    a = A\b;
    x = J(1):0.001:J(2);

    % Calculando a solução aproximada.
    y = x;
    for k = 1:n
```

```
        y = y+a(k)*x.^k.*(x-1);
    end

    % Resultados.
    ya = (exp(x)-exp(-x))/(exp(1)-exp(-1));
    semilogy(x, abs(y-ya), 'Linewidth', 1.6); hold on

end
xlabel('$x$','Interpreter','Latex')
ylabel('$|y(x)-y_n(x)|$','Interpreter','Latex')
l = legend('$n=5$','$n=10$','$n=15$','$n=20$');
set(l,'Interpreter','Latex')
```

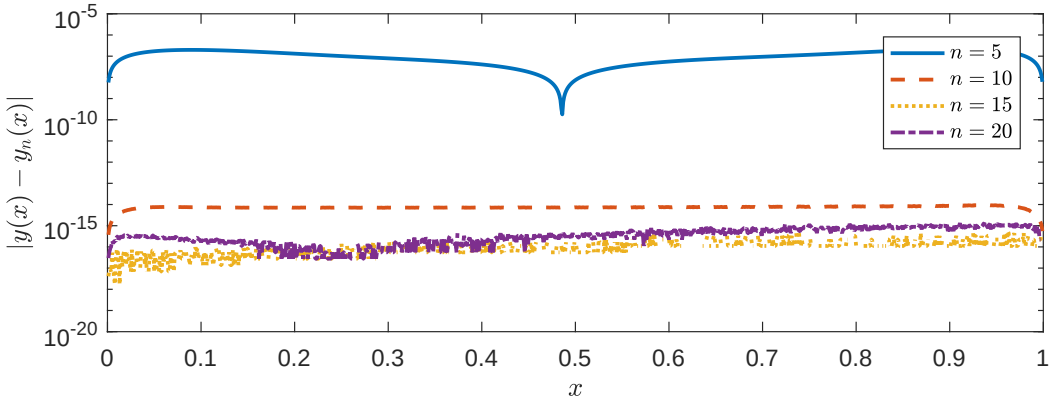

Figura 3.1: Erro na aproximação do Exemplo 3.1 pelo método da colocação, com $n = 5,\ 10,\ 15$ e 20.

Conforme o resultado apresentado na Figura 3.1, é atingida a precisão da máquina para $n = 15$ e $n = 20$. Conforme aumentamos n, mantemo-nos ainda próximos à precisão da máquina para este mesmo exemplo, todavia a matriz A já começa a ter um elevado número de condição.

3.2 Método de Galerkin

De forma similar ao método da colocação, e partir da equação de aproximação utilizada, o método de Galerkin consiste em usar os próprios $u_k,\ k = 1(1)n$ (referidos em (3.2)) em vez da função delta de Dirac. Assim, a matriz para o método de Galerkin é montada a partir de

$$a_{i,j} = \int_a^b u_i \mathfrak{F} u_j dx, \quad i, j = 1(1)n$$

e de

$$b_i = \int_a^b u_i \Big(f - \mathfrak{F} v \Big) dx, \quad i = 1(1)n,$$

ou seja

$$\int_a^b \begin{bmatrix} u_1\mathfrak{F}u_1 & u_1\mathfrak{F}u_2 & \dots & u_1\mathfrak{F}u_n \\ u_2\mathfrak{F}u_1 & u_2\mathfrak{F}u_2 & \dots & u_2\mathfrak{F}u_n \\ \vdots & \vdots & \ddots & \vdots \\ u_n\mathfrak{F}u_1 & u_n\mathfrak{F}u_2 & \dots & u_n\mathfrak{F}u_n \end{bmatrix} dx \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \int_a^b \begin{bmatrix} u_1\Big(f-\mathfrak{F}v\Big) \\ u_2\Big(f-\mathfrak{F}v\Big) \\ \vdots \\ u_n\Big(f-\mathfrak{F}v\Big) \end{bmatrix} dx$$

Novamente para o problema do Exemplo 3.1 usamos como função de aproximação (3.4), logo

$$a_{i,j} = \int_a^b x^i(x-1)\Big(2jx^{j-1} - x^j(x-1) + jx^{j-2}(j-1)(x-1)\Big)dx, \quad i,j = 1(1)n \tag{3.9}$$

e

$$b_i = \int_a^b x^{i+1}(x-1)dx, \quad i = 1(1)n. \tag{3.10}$$

Usando integração numérica para (3.9) e (3.10), obtemos o sistema

$$\begin{bmatrix} -0,3667 & -0,1833 & -0,1095 & -0,0726 & -0,0516 \\ -0,1833 & -0,1429 & -0,1060 & -0,0802 & -0,0623 \\ -0,1095 & -0,1060 & -0,0897 & -0,0742 & -0,0615 \\ -0,0726 & -0,0802 & -0,0742 & -0,0655 & -0,0571 \\ -0,0516 & -0,0623 & -0,0615 & -0,0571 & -0,0517 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} -0,0833 \\ -0,0500 \\ -0,0333 \\ -0,0238 \\ -0,0179 \end{bmatrix},$$

cuja solução é

$$\mathsf{a} = [0,1491,\ 0,1491,\ 0,0074,\ 0,0068,\ 0,0006]^T. \tag{3.11}$$

Substituindo (3.11) em (3.4) temos $\max|y_5 - y| = 3,4755 \cdot 10^{-8}$.

O Código 3.2 calcula e representa as soluções obtidas por este processo, com polinómios de graus $n = 5, 6, 8, 10$.

Código 3.2. Método de Galerkin para o Exemplo 3.1.

```
% Número de pontos de colocação: 4, 6, 8 e 10.
for n = 4:2:10
    clear a
    J = [0, 1]; a = J(1); b = J(2); A = zeros(n); B = zeros(n, 1);

    % Construindo A e b do sistema Ax=b.
    for i = 1:n
        for j = 1:n
            % Integração numérica para para A(i, j).
            A(i, j) = integral(@(x) x.^i.*(x-1).*(2*j*x.^(j-1) ...
                              -x.^j.*(x-1)+j*x.^(j-2).*(j-1).*(x-1)), a, b);
        end

        % Integração numérica para b(i).
        B(i) = integral(@(x) x.^(i+1).*(x-1), a, b);
```

```
    end

    % Resolve o sistema.
    a = A\B;
    x = J(1):0.001:J(2);

    % Calculando a solução aproximada.
    y = x;
    for k = 1:n
        y = y+a(k)*x.^k.*(x-1);
    end

    % Resultados.
    ya = (exp(x)-exp(-x))/(exp(1)-exp(-1));
    figure(2); semilogy(x, abs(y-ya), 'Linewidth', 1.6); hold on
end
xlabel('$x$', 'Interpreter', 'Latex')
ylabel('$|y(x)-y_n(x)|$', 'Interpreter', 'Latex')
l = legend('$n=4$','$n=6$','$n=8$','$n=10$');
set(l,'Interpreter','Latex')
```

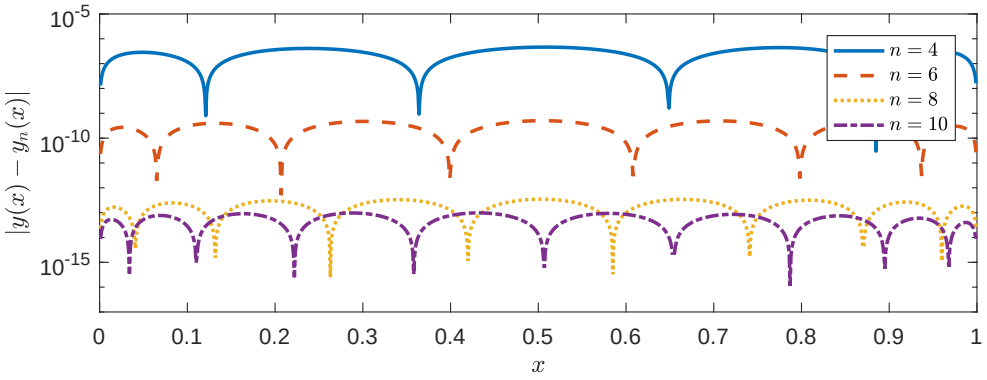

Figura 3.2: Erro na aproximação do Exemplo 3.1 pelo método de Galerkin, com $n = 4, 6, 8$ e 10.

Conforme o resultado apresentado na Figura 3.2, é atingida a precisão da máquina para $n = 10$. Mesmo sendo possível obter os integrais de forma exata para este exemplo, usamos a integração numérica, por se tratar de uma forma viável quando as funções base não são polinomiais, e portanto podem dificultar a integração.

3.3 Conclusões e considerações

Seja y a solução exata de um problema e y_n a sua aproximação espetral de ordem n. É característico nos métodos espetrais, o erro $y - y_n$ apresentar um comportamento oscilatório em torno de zero. O número de oscilações tende a aumentar com o grau n da aproximação, significando que o polinómio aproximante interpola a solução exata

num número crescente de pontos. Tornar n demasiado grande, todavia, implica em resolver sistemas de equações lineares algébricas também de mesma dimensão. Este problema assume particular importância uma vez que, tal como referido anteriormente, as matrizes associadas a estes problemas tendem a ser mal condicionadas, com n crescente. O Apêndice A apresenta métodos para a solução destes sistemas, que serão também os métodos de solução dos sistemas associados ao método espetral tau de Lanczos, e que por sua vez é apresentado no Capítulo a seguir.

Capítulo 4

O método tau

Sumário

O método tau é um método numérico para aproximar, através de polinómios, a solução de uma equação ou sistema de equações diferenciais, integrais ou integro-diferenciais. É um método que pertencente à classe dos métodos espetrais, pelo que a sua eficiência resulta das excelentes qualidades da convergência espetral. O método tem tido aplicação em equações diferenciais ordinárias lineares e não lineares, por exemplo os trabalhos Ortiz (1978); Ortiz and Samara (1981); Crisci and Russo (1983) e Liu and Pan (1999), e também tem vindo a ser aplicado em equações em derivadas parciais, como Namasivayam and Ortiz (1981); Ortiz and Dinh (1987) e Matos et al. (2004), bem como em equações integro-diferenciais de Fredholm e/ou Volterra AliAbadi and Shahmorad (2002); Hosseini and Shahmorad (2003a,b); Mahmound et al. (2005); Rahimi et al. (2010); Saeedi et al. (2013) e Mokhtary and Ghoreishi (2014). Em geral os trabalhos

desenvolvidos com o método tau são específicos para a resolução de um determinado problema.

O método tem como base a resolução de um problema com uma perturbação no segundo membro da equação diferencial, que por sua vez resulta em truncar o sistema algébrico linear infinito que traduz relações entre os coeficientes da solução polinomial a construir. Tanto a solução aproximada como a perturbação resultante no segundo membro, constituem uma combinação linear de polinómios. Dedicaremos este capítulo exclusivamente para abordar o método tau de Lanczos, e algumas motivações já são pontualmente apresentadas: (i) evitar o cálculo de integrais definidos, como na formulação de Galerkin; (ii) evitar a imposição de se ter que escolher funções de base que satisfaçam as condições de contorno do problema dado e (iii) automatizar um método de forma independente das especificidades do problema.

4.1 Formulação original de Lanczos

Em Lanczos (1938) Corneluis Lanczos propõe o método tau, e volta a abordá-lo com maior detalhe em Lanczos (1956) onde mostra alguns resultados, exemplos e estimativas de erro para casos elementares. Trata-se de uma ideia conceptualmente simples, porém de difícil implementação (da Silva, 1990): apresentar uma função y como uma combinação linear de polinómios, ortogonais ou não, e projetar o resíduo nesta base, de forma a evitar o cálculo de integrais definidos como nos métodos de colocação e de Galerkin (ilustrados no capítulo anterior).

Seja y uma função que satisfaz a equação

$$\begin{cases} \mathfrak{D}y = f, \quad x \in]a,\ b[\\ g_j(y) = \sigma_j, \quad j = 1(1)\nu \end{cases}, \tag{4.1}$$

onde

$$\mathfrak{D} = \sum_{r=0}^{\nu} p_r \frac{d^r}{dx^r} \tag{4.2}$$

é um operador diferencial linear de ordem ν e com coeficientes polinomiais, f é um polinómio algébrico (ou uma aproximação polinomial algébrica) de grau λ e σ_j são as ν condições de contorno obtidas com as funcionais lineares g_j. Nestas condições, Lanczos propôs perturbar o segundo membro do problema (4.1) com a inserção de uma combinação linear de polinómios de Chebyshev, resultando o problema perturbado, ou problema tau associado

$$\begin{cases} \mathfrak{D}y_n = f + \tau, \quad x \in]a,\ b[\\ g_j(y_n) = \sigma_j, \quad j = 1(1)\nu \end{cases},$$

onde

$$\tau = \sum_{i=n-\nu+1}^{n+h} \tau_i T_i \tag{4.3}$$

é o polinómio perturbador na base de Chebyshev de primeira espécie em $[a,\ b]$ e h é a altura do operador, definida por

$$h = \sup_{n \in \mathbb{N}} \left(\text{grau}(\mathfrak{D}T_n) - n\right). \tag{4.4}$$

Os coeficientes $\tau_i,\ i = n - \nu + 1(1)n + h$ da combinação linear perturbadora são chamados de parâmetros tau, dando nome à formulação. A forma (4.3) para a perturbação é justificada pela tentativa de se obter uma perturbação pequena no sentido da norma uniforme e porque o erro $e_n = y - y_n$ é solução do problema diferencial

$$\begin{cases} \mathfrak{D}e_n = -\tau, & x \in]a,\ b[\\ g_j(e_n) = 0, & j = 1(1)\nu \end{cases} .$$

Embora estejam evitados, nesta formulação original, os cálculos de integrais definidos, ainda persiste o problema da imposição da escolha de uma função que satisfaça as condições de contorno, o que se agrava principalmente em termos computacionais, pois não é de fácil automatização. Portanto, em vez de fixar a perturbação τ e só depois resolver de forma exata o problema tau, segundo a primeira formulação de Lanczos (1938), o procedimento passa por fixar uma base $\mathcal{Z}_n$ de $\mathcal{Z}$, e então representar y_n nesta base, na forma

$$y_n = \sum_{i=0}^{n} a_i Z_i,$$

determinando os coeficientes $\mathsf{a} = [a_0,\ \ldots,\ a_n]$ de tal forma que y_n satisfaça simultaneamente as condições g_i e $\mathfrak{D}y_n$ coincida com $\mathfrak{D}y$ o mais longe possível, isto é, nos coeficientes até ao grau $n - \nu$ (da Silva, 1990; Matos, 1994).

Exemplo 4.1. Seja o problema diferencial

$$\begin{cases} \dfrac{d}{dx}y - y = 0,\ x \in]0,\ 1[\\ y(0) = 1 \end{cases} ,$$

cuja solução exata é $\exp(x)$.

Solução. Considerando a base dos polinómios de Chebyshev de primeira espécie, ortogonais no intervalo $[0,\ 1]$, encontramos

$$\begin{aligned} T_0 &= 1 \\ T_1 &= 2x - 1 \\ T_2 &= 8x^2 - 8x + 1 \\ T_3 &= 32x^3 - 48x^2 + 18x - 1 \\ T_4 &= 128x^4 - 256x^3 + 160x^2 - 32x + 1 \\ T_5 &= 512x^5 - 1280x^4 + 1120x^3 - 400x^2 + 50x - 1 \\ &\vdots \end{aligned}$$

de onde podemos escrever suas derivadas como

$$\begin{aligned} \frac{d}{dx}T_0 &= 0 \\ \frac{d}{dx}T_1 &= 2T_0 \\ \frac{d}{dx}T_2 &= 8T_1 \end{aligned}$$

$$\begin{aligned}\frac{d}{dx}T_3 &= 12T_2 + 6T_0\\ \frac{d}{dx}T_4 &= 16T_3 + 16T_1\\ \frac{d}{dx}T_5 &= 20T_4 + 20T_2 + 10T_0\\ &\vdots\end{aligned}$$

Seja $y_5 = \sum_{i=0}^{5} a_i T_i$ a aproximação de y. Aplicando o método tau ao Exemplo 4.1, que tem $\nu = 1$ (de (4.2)), e $h = 0$ (de (4.4)), temos o problema perturbado

$$\begin{cases}\dfrac{d}{dx}y_5 - y_5 = \tau_5 T_5,\ x \in]0,\ 1[\\ y(0) = 1\end{cases},$$

e como $\dfrac{d}{dx}y_5 = \sum_{i=0}^{5} a_i \sum_{j=0}^{i-1} c_j T_j = \sum_{i=0}^{5} a_i \dfrac{d}{dx}T_i$, encontramos

$$\begin{aligned}\mathfrak{D}y_5 &= (2a_1 + 6a_3 + 10a_5 - a_0)T_0\\ &+ (8a_2 + 16a_4 - a_1)T_1\\ &+ (12a_3 + 20a_5 - a_2)T_2\\ &+ (16a_4 - a_3)T_3\\ &+ (20a_5 - a_4)T_4\\ &+ -a_5 T_5.\end{aligned}$$

Ao comparar os coeficientes, encontramos 6 equações lineares algébricas, nas 7 incógnitas $a_0,\ \ldots,\ a_5,\ \tau_5$ referentes ao operador, que são homogéneas. Temos ainda uma equação oriunda da condição inicial: $\sum_{i=0}^{n} a_i T_i(0) = 1$, fornecendo $a_0 - a_1 + a_2 - a_3 + a_4 - a_5 = 1$. O sistema final, cuja solução é o vetor $\mathsf{a}_{\mathcal{T}} = [a_0,\ \ldots,\ a_5]^T$ é

$$\begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1\\ -1 & 2 & 0 & 6 & 0 & 10\\ & -1 & 8 & 0 & 16 & 0\\ & & -1 & 12 & 0 & 20\\ & & & -1 & 16 & 0\\ & & & & -1 & 20\\ & & & & & -1 \end{bmatrix} \begin{bmatrix} a_0\\ a_1\\ a_2\\ a_3\\ a_4\\ a_5 \end{bmatrix} = \begin{bmatrix} 1\\ 0\\ 0\\ 0\\ 0\\ 0\\ \tau_5 \end{bmatrix}.$$

Truncando e resolvendo o sistema em suas primeiras 6 equações, temos a aproximação na base de Chebyshev de primeira espécie

$$y_5 = \frac{647}{369}T_0 + \frac{2609}{3068}T_1 + \frac{305}{2899}T_2 + \frac{49}{5618}T_3 + \frac{20}{36689}T_4 + \frac{1}{36689}T_5,$$

que na base $\mathcal{X}$ é

$$y_5 = 1 + \frac{36690}{36689}x + \frac{374}{749}x^2 + \frac{565}{3322}x^3 + \frac{98}{2809}x^4 + \frac{79}{5661}x^5.$$

A Figura 4.1 apresenta a diferença $|y_5 - p_5|$, onde p_5 é o polinómio de Taylor de grau 5 para $\exp(x)$, o resíduo $|-\tau_5 T_5|$ e o erro $|y_5 - \exp(x)|$.

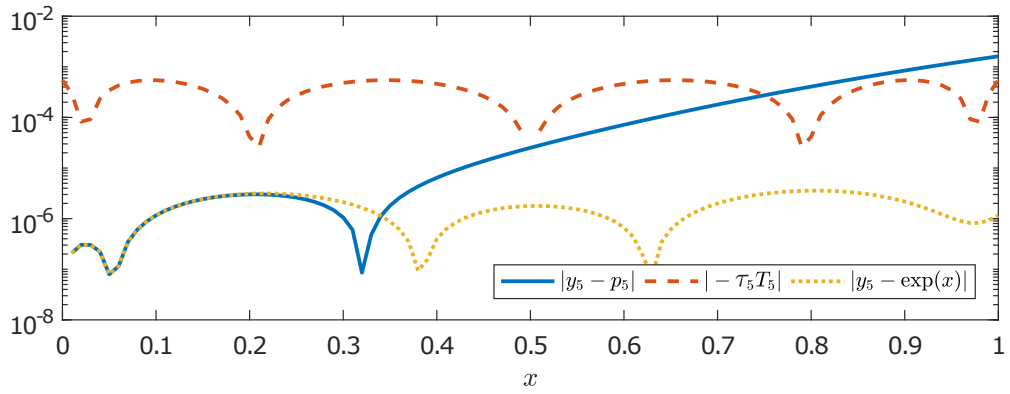

Figura 4.1: Resultados para o Exemplo 4.1.

Da Figura 4.1 percebemos que as três mensurações são diferentes, sendo que o erro apresenta valores menores no domínio em relação à expansão em série de Taylor e ao resíduo. Quando aumentamos a ordem n de aproximação, as mensurações de comparação com a expansão em série de Taylor e de erro aproximam-se da precisão da máquina, enquanto o resíduo $\tau_n T_n$ aproxima-se de zero.

Exemplo 4.2. Seja o problema diferencial de coeficientes polinomiais

$$\begin{cases} x^3 \dfrac{d^2}{dx^2} y + x \dfrac{d}{dx} y - y = 60x^5 + 21x^4 - 6x^3 - 4x^2 + 12, \quad x \in]-1,\ 1[\\ y(-1) = -19,\ y(1) = -3 \end{cases},$$

cuja solução exata é o polinómio $y = 5x^4 + x^3 - 4x^2 + 7x - 12$.

Solução. Aplicando o método tau com a base de Chebyshev neste problema, que tem $\nu = 2$ e $h = 1$, temos o problema perturbado

$$\begin{cases} x^3 \dfrac{d^2}{dx^2} y_7 + x \dfrac{d}{dx} y_7 - y_7 = 60x^5 + 21x^4 - 6x^3 - 4x^2 + 12 + \tau_6 T_6 + \tau_7 T_7 + \tau_8 T_8 \\ y_7(-1) = -19,\ y_7(1) = -3 \end{cases},$$

Neste problema, diferentemente do anterior, temos os coeficientes polinomiais, que exigem alguma manipulação analítica adicional, mas da mesma forma, aplicamos o operador $\mathfrak{D}$ em $y_7 = \sum_{i=0}^{7} a_i T_i$, evidenciando os T_i, $i = 0(1)7$. Escrevemos então o polinómio do lado direito da equação diferencial na base $\mathcal{T}$, donde comparando os

coeficientes encontramos 9 equações lineares, que juntamente com as duas equações oriundas das condições de fronteira, geram o sistema

$$\begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & 0 & 2 & 9 & 4 & 55 & 6 & 161 \\ 0 & 0 & 3 & 6 & 48 & 10 & 192 & 14 \\ & 0 & 1 & 12 & 8 & 90 & 12 & 294 \\ & & 1 & 2 & 26 & 10 & 144 & 14 \\ & & & 3 & 3 & 45 & 12 & 210 \\ & & & & 6 & 4 & 69 & 14 \\ & & & & & 10 & 5 & 98 \\ & & & & & & 15 & 6 \\ & & & & & & & 21 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} = \begin{bmatrix} -19 \\ -3 \\ \frac{143}{8} \\ 33 \\ \frac{17}{2} \\ \frac{69}{4} \\ \frac{21}{8} \\ \frac{15}{4} \\ \tau_6 \\ \tau_7 \\ \tau_8 \end{bmatrix}. \tag{4.5}$$

Resolvendo o sistema considerando as suas primeiras 8 equações, obtemos

$$y_7 = -\frac{97}{8}T_0 + \frac{31}{4}T_1 + \frac{1}{2}T_2 + \frac{1}{4}T_3 + \frac{5}{8}T_4 + \sum_{i=5}^{7} a_i T_i,$$

que corresponde à solução exata escrita na base $\mathcal{T}$, exceto dos coeficientes a_i, $i = 5, 6, 7$, que são da ordem de $1 \cdot 10^{-15}$. Considerando as últimas 3 equações, é possível calcular os coeficientes τ_i, $i = 6, 7, 8$, do resíduo, que por sua vez é apresentado na Figura 4.2.

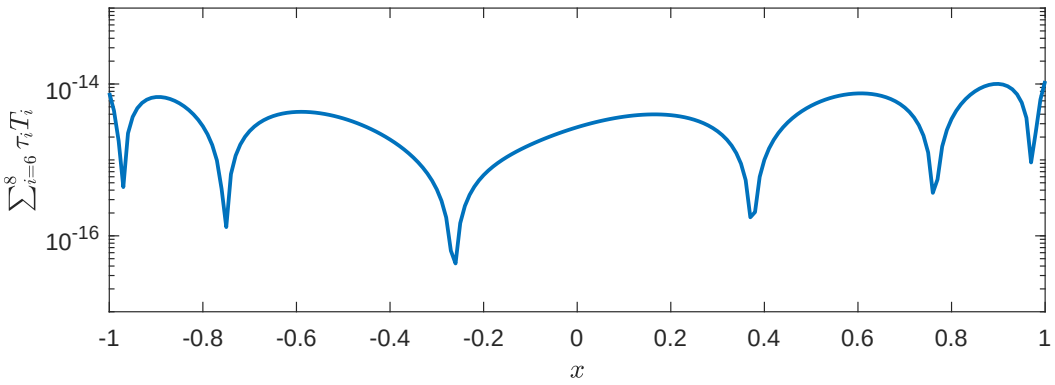

Figura 4.2: Resíduo tau para o Exemplo 4.1.

A complexidade do operador é proporcional à complexidade da manipulação algébrica para se comparar os coeficientes da base em ambos os lados do problema, processo este necessário para montar o sistema de equações lineares algébricas associado. A Secção seguinte, trata de contornar esta dificuldade, onde a formulação original de Lanczos dá lugar à formulação operacional de Ortiz: uma forma equivalente à primeira, mas de fácil manipulação algébrica.

4.2 Formulação operacional de Ortiz

Ortiz and Samara (1981) introduziram uma formulação operacional para aplicar o método tau de Lanczos alicerçada num conjunto de operações matriciais que traduzem a ação do operador diferencial $\mathfrak{D}$, definido em (4.2), em termos de uma combinação linear de polinómios, juntamente com as condições de contorno. Trata-se de uma forma computacionalmente prática de aplicar o método tau. Nesta secção mostraremos como representar matricialmente os operadores que transformam polinómios em polinómios, o emprego destas representações matriciais para operadores integro-diferenciais com coeficientes polinomiais e os resultados de mudança de base.

4.2.1 Representação matricial de operadores

Seja $\mathcal{X} = [x^0,\ x^1,\ x^2,\ \ldots]$ a base das potências, e $y_n = \sum_{i=0}^{n} a_i x^i = \mathcal{X}\mathsf{a}$ um polinómio escrito como combinação linear dos elementos desta base, com $\mathsf{a} = [a_0,\ \ldots,\ a_n,\ 0,\ 0,\ \ldots]^T$, então as matrizes M, N e O que representam, respetivamente, a ação sobre os coeficientes de y_n do operador que incrementa o grau em y_n, a ação sobre os coeficientes de y_n do operador que deriva y_n e a ação sobre os coeficientes de y_n do operador que primitiva y_n, satisfazem, respetivamente a

$$\mathcal{X}\mathsf{M}\mathsf{a} = xy, \quad \mathcal{X}\mathsf{N}\mathsf{a} = \frac{d}{dx}y \quad \text{e} \quad \mathcal{X}\mathsf{O}\mathsf{a} = \int^x y dx,$$

e apresentam a forma

$$\mathsf{M} = \begin{bmatrix} 0 & & & & \\ 1 & 0 & & & \\ & 1 & 0 & & \\ & & 1 & 0 & \\ & & & \ddots & \ddots \end{bmatrix}, \quad \mathsf{N} = \begin{bmatrix} 0 & 1 & & & \\ & 0 & 2 & & \\ & & 0 & 3 & \\ & & & \ddots & \ddots \end{bmatrix} \quad \text{e} \quad \mathsf{O} = \begin{bmatrix} 0 & & & & \\ 1 & 0 & & & \\ & \frac{1}{2} & 0 & & \\ & & \frac{1}{3} & 0 & \\ & & & \ddots & \ddots \end{bmatrix}. \tag{4.6}$$

As potências destas matrizes traduzem operadores de ordem superior, conforme apresentado nas proposições que se seguem.

Proposição 4.1. *Sejam $\mathcal{X}$ a base das potências, p e q dois vetores de coeficientes dos polinómios, p e q, representados como combinação linear dos elementos de $\mathcal{X}$ e M a matriz apresentada em (4.6) que representa a ação do operador de incremento de grau sobre p e q, assim*

$$\mathcal{X}x^r = \mathcal{X}\mathsf{M}^r, \quad r \in \mathbb{N}, \tag{4.7}$$

$$\mathcal{X}p = \mathcal{X}p(\mathsf{M}), \quad p(\mathsf{M}) = \sum_{i=0}^{n} p_i \mathsf{M}^i \tag{4.8}$$

e

$$pq = \mathcal{X}p(\mathsf{M})\mathsf{q} = \mathcal{X}q(\mathsf{M})\mathsf{p}. \tag{4.9}$$

Demonstração. O resultado (4.7) é imediato, a partir da definição de M e por indução sobre r.

Considerando o caso da multiplicação por um polinómio, e sendo

$$p = \sum_{i=0}^{n} p_i x^i = \mathcal{X}\mathsf{p}, \quad \mathsf{p} = [p_0,\ p_1,\ \ldots,\ p_n,\ 0,\ 0,\ \ldots]^T,$$

o resultado (4.8) é consequência de (4.7).

Finalmente, considerando o caso (4.9) do produto pq, onde p é de ordem m e q de ordem n, temos que

$$\begin{aligned} pq &= p\sum_{i=0}^{n} q_i x^i \\ &= \sum_{i=0}^{n} q_i p x^i \\ &= \mathcal{X}p(\mathsf{M})\mathsf{q} = \mathcal{X}q(\mathsf{M})\mathsf{p} \end{aligned}$$

□

Proposição 4.2. *Seja $\mathcal{X}$ a base das potências, a o vetor dos coeficientes de um polinómio representado como combinação linear dos elementos de $\mathcal{X}$ e N a matriz apresentada em (4.6) que representa a ação do operador de diferenciação sobre a, assim*

$$\frac{d^r}{dx^r}\mathcal{X}\mathsf{a} = \mathcal{X}\mathsf{N}^r\mathsf{a}, \quad r \in \mathbb{N} \tag{4.10}$$

representa a derivação de ordem superior em termos operacionais.

Demonstração. Uma vez que a derivada de ordem r de x^n é dada por

$$\frac{d^r}{dx^r}x^n = \frac{d^{r-1}}{dx^{r-1}}nx^{n-1} = \frac{d^{r-2}}{dx^{r-2}}(n-1)nx^{n-2} = \ldots = \frac{n!}{(n-r)!}x^{n-r},$$

então temos, para $r \geq 0$, que

$$\frac{d^r}{dx^r}x^n = \begin{cases} 0, & \text{se} \quad n = 0(1)r-1 \\ \dfrac{n!}{(n-r)!}x^{n-r}, & \text{se} \quad n \geq r \end{cases},$$

e assim a potência r de N que satisfaz

$$\frac{d^r}{dx^r}\mathcal{X} = \mathcal{X}\mathsf{N}^r$$

será da forma

$$\mathsf{N}^r = \begin{bmatrix} 0 & \cdots & 0 & \frac{r!}{0!} & & \\ 0 & \cdots & 0 & 0 & \frac{(r+1)!}{1!} & \\ \vdots & & \vdots & 0 & 0 & \frac{(r+2)!}{2!} & \\ & & & & & \ddots & \ddots \end{bmatrix}.$$

□

Proposição 4.3. *Seja $\mathcal{X}$ a base das potências, a o vetor dos coeficientes de um polinómio representado como combinação linear dos elementos de $\mathcal{X}$ e O a matriz apresentada em (4.6) que representa a ação do operador de integração sobre a, assim*

$$\underbrace{\int^x \cdots \int^x}_{r \ \ vezes} \mathcal{X}\mathsf{a}dx^r = \mathcal{X}\mathsf{O}^r\mathsf{a}, \quad r \in \mathbb{N} \tag{4.11}$$

representa a integração de ordem superior em termos operacionais.

Demonstração. Procedendo de forma similar ao caso da derivação, temos que a r-ésima primitiva de x^n é dada por

$$\begin{aligned}
\underbrace{\int^x \cdots \int^x}_{r \text{ vezes}} x^n dx^r &= \underbrace{\int^x \cdots \int^x}_{r-1 \text{ vezes}} \frac{x^{n+1}}{n+1} dx^{r-1} \\
&= \underbrace{\int^x \cdots \int^x}_{r-2 \text{ vezes}} \frac{x^{n+2}}{(n+2)(n+1)} dx^{r-2} \\
&= \frac{n!}{(n+r)!} x^{n+r},
\end{aligned}$$

e portanto, a potência r de O que satisfaz

$$\underbrace{\int^x \cdots \int^x}_{r \text{ vezes}} \mathcal{X} dx^r = \mathcal{X}\mathsf{O}^r$$

terá a forma

$$\mathsf{O}^r = \begin{bmatrix}
0 & 0 & \cdots & & \\
\vdots & \vdots & & & \\
0 & 0 & \cdots & & \\
\frac{0!}{r!} & 0 & & & \\
 & \frac{1!}{(r+1)!} & 0 & & \\
 & & \frac{2!}{(r+2)!} & 0 & \\
 & & & \ddots & \ddots
\end{bmatrix}.$$

□

Propostas as matrizes que representam as ações dos operadores que transformam polinómios em polinómios, temos o alicerce para traduzir operadores integro-diferenciais em termos matriciais, conforme descrição seguinte.

4.2.2 Operador integro-diferencial com coeficientes polinomiais

Com os resultados da secção anterior é possível representar matricialmente a ação de um operador integro-diferencial linear com coeficientes polinomiais sobre o vetor dos coeficientes de uma série.

Teorema 4.1. *Seja* $y_n = \sum_{i=0}^{n} a_i x^i = \mathcal{X}\mathsf{a}, \quad \mathsf{a} = [a_0,\ \ldots,\ a_n,\ 0,\ 0,\ \ldots]^T$ *um polinómio na base das potências, de grau* n*, e seja*

$$\mathfrak{D} = \sum_{k=0}^{\nu} p_k \frac{d^k}{dx^k}$$

onde

$$p_k = \sum_{i=0}^{\partial_k} p_{ki} x^i = \mathcal{X}\mathsf{p_k}, \quad \mathsf{p_k} = [p_{k0},\ p_{k1},\ \ldots,\ p_{k\partial_k},\ 0,\ 0,\ \ldots]^T,$$

um operador diferencial linear ordinário de ordem ν *e com coeficientes polinomiais, então, fazer* $\mathfrak{D}$ *operar sobre* y_n *produz o resultado*

$$\mathfrak{D}y_n = \mathcal{X}\mathsf{D}\mathsf{a}, \quad \mathsf{D} = \sum_{k=0}^{\nu} p_k(\mathsf{M})\mathsf{N}^k. \tag{4.12}$$

Demonstração. Como $y_n = \mathcal{X}\mathsf{a}$ e considerando $k, s \in \mathbb{N}$, de (4.10) temos que

$$\frac{d^k}{dx^k} y_n = \mathcal{X}\mathsf{N}^k\mathsf{a}, \tag{4.13}$$

e de (4.7) e (4.10) temos que

$$x^s \frac{d^k}{dx^k}\mathcal{X} = \mathcal{X}\mathsf{M}^s\mathsf{N}^k. \tag{4.14}$$

Assim, considerando o resultado (4.10) simultaneamente com (4.13) e (4.14) é verdade que

$$p\frac{d^k}{dx^k} y_n = \mathcal{X}p(\mathsf{M})\mathsf{N}^k\mathsf{a}, \tag{4.15}$$

e portando (4.12) é combinação linear de (4.15). □

Teorema 4.2. *Seja* $y_n = \sum_{i=0}^{n} a_i x^i = \mathcal{X}\mathsf{a}, \quad \mathsf{a} = [a_0,\ \ldots,\ a_n,\ 0,\ 0,\ \ldots]^T$ *um polinómio na base das potências, de grau* n*, e seja*

$$\mathfrak{S} = \sum_{k=0}^{\gamma} p_k \underbrace{\int^x \ldots \int^x}_{k\ vezes} \cdot\, dx^k$$

onde

$$p_k = \sum_{i=0}^{\partial_k} p_{ki} x^i = \mathcal{X}\mathsf{p_k}, \quad \mathsf{p_k} = [p_{k0},\ p_{k1},\ \ldots,\ p_{k\partial_k},\ 0,\ 0,\ \ldots]^T,$$

um operador integral linear ordinário de ordem γ *e com coeficientes polinomiais, então, fazer* $\mathfrak{S}$ *operar sobre* y_n *produz o resultado*

$$\mathfrak{S}y_n = \mathcal{X}\mathsf{S}\mathsf{a}, \quad \mathsf{S} = \sum_{k=0}^{\gamma} p_k(\mathsf{M})\mathsf{O}^k.$$

Demonstração. Análoga à do Teorema 4.1, fazendo uso da matriz O. □

Um operador integro-diferencial linear de coeficientes polinomiais pode, com a finalidade de unificar e simplificar a notação, ser expresso como

$$\mathfrak{E} = \mathfrak{D} + \mathfrak{S} = \sum_{k=-\gamma}^{\nu} p_k \frac{d^k}{dx^k}, \tag{4.16}$$

onde $\dfrac{d^{-k}}{dx^{-k}}$ denota a k-ésima primitiva.

4.2.3 Mudança para bases de polinómios ortogonais por transformações de semelhança

A mudança da representação polinomial, da base de potências para uma base de polinómios ortogonais, pode fazer-se por transformação de semelhança nas matrizes de representação dos operadores.

Teorema 4.3. *Seja $\mathcal{Z} = [Z_0,\ Z_1,\ \ldots]$ uma base de polinómios ortogonais, e $\mathsf{V}_{\mathcal{Z}}$ a matriz que define $\mathcal{Z} = \mathcal{X}\mathsf{V}_{\mathcal{Z}}$. As matrizes que traduzem as operações de multiplicação $x^k\mathcal{Z} = \mathcal{Z}\mathsf{M}^k_{\mathcal{Z}}$, derivação $\frac{d^k}{dx^k}\mathcal{Z} = \mathcal{Z}\mathsf{N}^k_{\mathcal{Z}}$ e primitivação $\underbrace{\int^x \ldots \int^x}_{k\ vezes} \mathcal{Z}dx^k = \mathcal{Z}\mathsf{O}^k_{\mathcal{Z}}$, de polinómios na base $\mathcal{Z}$ são, respetivamente*

$$\mathsf{M}_{\mathcal{Z}} = \mathsf{V}_{\mathcal{Z}}^{-1}\mathsf{M}\mathsf{V}_{\mathcal{Z}}, \quad \mathsf{N}_{\mathcal{Z}} = \mathsf{V}_{\mathcal{Z}}^{-1}\mathsf{N}\mathsf{V}_{\mathcal{Z}} \quad e \quad \mathsf{O}_{\mathcal{Z}} = \mathsf{V}_{\mathcal{Z}}^{-1}\mathsf{O}\mathsf{V}_{\mathcal{Z}}.$$

Demonstração. Segue de (4.7) e da definição de $\mathcal{Z}$ que

$$\begin{aligned} x^k\mathcal{Z} &= x^k\mathcal{X}\mathsf{V}_{\mathcal{Z}} \\ &= \mathcal{X}\mathsf{M}^k\mathsf{V}_{\mathcal{Z}} \\ &= \mathcal{Z}\mathsf{V}_{\mathcal{Z}}^{-1}\mathsf{M}^k\mathsf{V}_{\mathcal{Z}} \\ &= \mathcal{Z}\left(\mathsf{V}_{\mathcal{Z}}^{-1}\mathsf{M}\mathsf{V}_{\mathcal{Z}}\right)^k = \mathcal{Z}\mathsf{M}^k_{\mathcal{Z}}, \end{aligned}$$

similarmente de (4.10) segue que

$$\begin{aligned} \frac{d^k}{dx^k}\mathcal{Z} &= \frac{d^k}{dx^k}\mathcal{X}\mathsf{V}_{\mathcal{Z}} \\ &= \mathcal{X}\mathsf{N}^k\mathsf{V}_{\mathcal{Z}} \\ &= \mathcal{Z}\mathsf{V}_{\mathcal{Z}}^{-1}\mathsf{N}^k\mathsf{V}_{\mathcal{Z}} \\ &= \mathcal{Z}\left(\mathsf{V}_{\mathcal{Z}}^{-1}\mathsf{N}\mathsf{V}_{\mathcal{Z}}\right)^k = \mathcal{Z}\mathsf{N}^k_{\mathcal{Z}}, \end{aligned}$$

e de (4.11) que

$$\begin{aligned} \underbrace{\int^x \ldots \int^x}_{k\ \text{vezes}} \mathcal{Z}dx^k &= \underbrace{\int^x \ldots \int^x}_{k\ \text{vezes}} \mathcal{X}\mathsf{V}_{\mathcal{Z}}dx^k \\ &= \mathcal{X}\mathsf{O}^k\mathsf{V}_{\mathcal{Z}} \end{aligned}$$

$$\begin{aligned} &= \mathcal{Z}\mathsf{V}_{\mathcal{Z}}^{-1}\mathsf{O}^k\mathsf{V}_{\mathcal{Z}} \\ &= \mathcal{Z}\left(\mathsf{V}_{\mathcal{Z}}^{-1}\mathsf{O}\mathsf{V}_{\mathcal{Z}}\right)^k = \mathcal{Z}\mathsf{O}_{\mathcal{Z}}^k. \end{aligned}$$

□

Com os resultados do Teorema anterior, podemos representar o operador integro-diferencial (4.16) na base $\mathcal{Z}$, conforme apresentado no Teorema seguinte.

Teorema 4.4. *Seja $\mathcal{Z} = [Z_0,\ Z_1,\ \ldots]$ uma base de polinómios ortogonais, e $\mathsf{V}_{\mathcal{Z}}$ a matriz satisfazendo $\mathcal{Z} = \mathcal{X}\mathsf{V}_{\mathcal{Z}}$, então temos que*

$$\mathfrak{E}_{\mathcal{Z}} = \mathcal{Z}\left(\sum_{i=1}^{\gamma} p_i(\mathsf{M}_{\mathcal{Z}})\mathsf{O}_{\mathcal{Z}}^i + \sum_{i=0}^{\nu} p_i(\mathsf{M}_{\mathcal{Z}})\mathsf{N}_{\mathcal{Z}}^i\right), \tag{4.17}$$

Demonstração. Temos que

$$\begin{aligned} x^s\frac{d^k}{dx^k}\mathcal{Z} &= x^s\frac{d^k}{dx^k}\mathcal{X}\mathsf{V}_{\mathcal{Z}} \\ &= \mathcal{X}\mathsf{M}^s\mathsf{N}^k\mathsf{V}_{\mathcal{Z}} \\ &= \mathcal{Z}\mathsf{V}_{\mathcal{Z}}^{-1}\mathsf{M}^s\mathsf{N}^k\mathsf{V}_{\mathcal{Z}} \\ &= \mathcal{Z}\mathsf{V}_{\mathcal{Z}}^{-1}\mathsf{M}^s\mathsf{V}_{\mathcal{Z}}\mathsf{V}_{\mathcal{Z}}^{-1}\mathsf{N}^k\mathsf{V}_{\mathcal{Z}} = \mathcal{Z}\mathsf{M}_{\mathcal{Z}}^s\mathsf{N}_{\mathcal{Z}}^k, \end{aligned} \tag{4.18}$$

e que

$$\begin{aligned} x^s\underbrace{\int^x\ldots\int^x}_{k\text{ vezes}}\mathcal{Z}dx^k &= x^s\underbrace{\int^x\ldots\int^x}_{k\text{ vezes}}\mathcal{X}\mathsf{V}_{\mathcal{Z}}dx^k \\ &= \mathcal{X}\mathsf{M}^s\mathsf{O}^k\mathsf{V}_{\mathcal{Z}} \\ &= \mathcal{Z}\mathsf{V}_{\mathcal{Z}}^{-1}\mathsf{M}^s\mathsf{O}^k\mathsf{V}_{\mathcal{Z}} \\ &= \mathcal{Z}\mathsf{V}_{\mathcal{Z}}^{-1}\mathsf{M}^s\mathsf{V}_{\mathcal{Z}}\mathsf{V}_{\mathcal{Z}}^{-1}\mathsf{O}^k\mathsf{V}_{\mathcal{Z}} = \mathcal{Z}\mathsf{M}_{\mathcal{Z}}^s\mathsf{O}_{\mathcal{Z}}^k, \end{aligned} \tag{4.19}$$

e portanto (4.17) é obtida da combinação linear de (4.18) e (4.19). □

4.3 Método tau para equações diferenciais ordinárias

Seja $\mathcal{Z} = [Z_0,\ Z_1,\ \ldots]$ uma base de polinómios ortogonais e seja $y_n = \sum_{i=0}^{n} a_i Z_i = \mathcal{Z}\mathsf{a}_{\mathcal{Z}}$ um polinómio na base $\mathcal{Z}$, de coeficientes $\mathsf{a}_{\mathcal{Z}} = [a_0,\ \ldots,\ a_n,\ 0,\ 0,\ \ldots]^T$. Com estas definições faremos uso dos resultados da Subsecção 4.2.1 para mostrar que a resolução do problema perturbado

$$\begin{cases} \mathfrak{D}y_n = f + \displaystyle\sum_{i=n-\nu+1}^{n+h}\tau_i Z_i, \quad x \in]a,\ b[\\ g_j(y_n) = \sigma_j, \quad j = 1(1)\nu \end{cases}, \tag{4.20}$$

com

$$\mathfrak{D} = \sum_{r=0}^{\nu} p_r\frac{d^r}{dx^r},$$

pode fazer-se recorrendo à resolução de um sistema de equações lineares algébricas, seguindo a formulação operacional do método tau de Ortiz and Samara (1981). Começamos por transcrever o problema para o formato matricial, definindo as matrizes $\mathsf{C}_{\mathcal{Z}}$ e $\mathsf{D}_{\mathcal{Z}}$, respetivamente, como

$$\mathsf{C}_{\mathcal{Z}} = [c_{ij}]_{\nu\times\infty}, \quad c_{ij} = g_i(Z_j), \quad i = 1(1)\nu, \quad j \geq 0$$

e

$$\mathsf{D}_{\mathcal{Z}} = \sum_{r=0}^{\nu} p_r(\mathsf{M}_{\mathcal{Z}})\mathsf{N}_{\mathcal{Z}}^{\mathsf{r}}, \qquad p_r(\mathsf{M}_{\mathcal{Z}}) = \sum_{k=0}^{n_r} p_{r,k}\mathsf{M}_{\mathcal{Z}}^{\mathsf{k}}, \quad \text{se} \quad p_r = \sum_{k=0}^{n_r} p_{r,k}x^k.$$

Pelo Teorema 4.1, o problema (4.1) a ser resolvido é convertido no sistema duplamente infinito

$$\begin{cases} \mathsf{C}_{\mathcal{Z}}\mathsf{a}_{\mathcal{Z}} = \mathsf{s} \\ \mathsf{D}_{\mathcal{Z}}\mathsf{a}_{\mathcal{Z}} = \mathsf{f}_{\mathcal{Z}} \end{cases},$$

onde $\mathsf{s} = [\sigma_1, \ \ldots, \ \sigma_\nu]^T$ e $\mathsf{f}_{\mathcal{Z}} = [f_0, \ \ldots, \ f_\lambda, \ 0, \ 0, \ \ldots]^T$ tal que $f_{\mathcal{Z}} = \sum_{i=0}^{\lambda} f_i Z_i$. Concatenando vetores e matrizes, podemos dizer que os coeficientes $\mathsf{a}_{\mathcal{Z}}$ da solução, são solução do sistema

$$\mathsf{T}_{\mathcal{Z}}\mathsf{a}_{\mathcal{Z}} = \mathsf{b}_{\mathcal{Z}},$$

onde

$$\mathsf{T}_{\mathcal{Z}} = \begin{bmatrix} \mathsf{C}_{\mathcal{Z}} \\ \mathsf{D}_{\mathcal{Z}} \end{bmatrix}, \quad \text{e} \quad \mathsf{b}_{\mathcal{Z}} = \begin{bmatrix} \mathsf{s} \\ \mathsf{f}_{\mathcal{Z}} \end{bmatrix}.$$

Ao truncarmos as matrizes infinitas $\mathsf{T}_{\mathcal{Z}}$, $\mathsf{a}_{\mathcal{Z}}$ e $\mathsf{b}_{\mathcal{Z}}$ na dimensão n, no caso de resultar uma matriz regular, então temos uma aproximação polinomial de ordem $n-1$ para a solução y. O formato característico da matriz $\mathsf{T}_{\mathcal{Z}}$ nessa formulação é geralmente como mostrado na Figura 4.3.

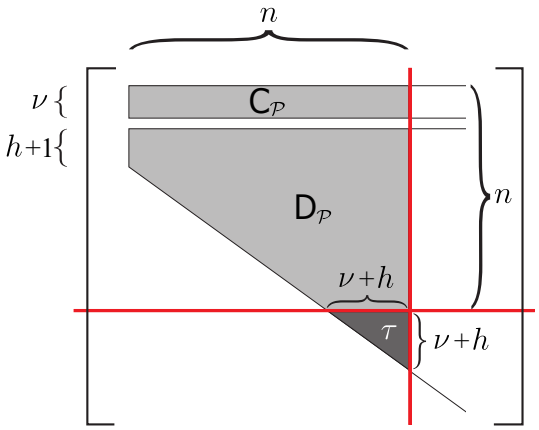

Figura 4.3: Formato da matriz $\mathsf{T}_{\mathcal{Z}}$ truncada na ordem n.

Quanto ao valor de n, da dimensão da matriz truncada, deve ser tal que o sistema truncado contenha todos os elementos do operador $\mathfrak{D}$ e todos os elementos de $\mathsf{f}_{\mathcal{Z}}$, e portanto deve satisfazer

$$n \geq \max(\nu + h, \nu + \lambda),$$

onde $h = \sup_{n\in\mathbb{N}} (\text{grau}(\mathfrak{D}Z_n) - n)$.

4.4 Método tau para sistemas de equações diferenciais ordinárias

Pretendemos aqui mostrar a forma de se aplicar o método tau para sistemas de equações diferenciais, e para isso começamos por considerar o problema de resolver um sistema de equações diferenciais ordinárias lineares não homogéneas e com coeficientes polinomiais

$$\begin{cases} \mathfrak{D}\mathsf{y} = \mathsf{f}, \quad x \in]a,\ b[\\ \mathsf{g}_k(\mathsf{y}) = \sigma_k, \qquad k = 1(1)\nu \end{cases}, \tag{4.21}$$

onde

$$\mathsf{y} = [y_1,\ y_2,\ \ldots,\ y_m]^T,$$

$$\mathsf{f} = \left[f_1 = \sum_{k=0}^{n} f_{1,k}x^k,\ f_2 = \sum_{k=0}^{n} f_{2,k}x^k,\ \ldots,\ f_m = \sum_{k=0}^{n} f_{m,k}x^k \right]^T,$$

e

$$\mathsf{g}_k(\mathsf{y}) = \sum_{i=1}^{m} g_{k,i}(y_i)$$

são, respetivamente, o vetor y m-dimensional, cujas componentes são as funções a aproximar, o vetor f m-dimensional, cujas componentes são polinómios ou aproximações polinomiais e g_k as funcionais multi-lineares nas componentes de

$$\mathsf{y}, \quad \frac{d}{dx}\mathsf{y}, \quad \ldots, \quad \frac{d^{\nu-1}}{dx^{\nu-1}}\mathsf{y},$$

que traduzem as condições de contorno do problema. A restrição de que os polinómios f_i têm todos o mesmo grau n é apenas aparente, bastando acrescentar coeficientes $f_{i,k} = 0$ em número suficiente quando assim não for.

Finalmente, $\mathfrak{D}$ é o operador diferencial, agora no formato matricial

$$\mathfrak{D} = [\mathfrak{D}_{ij}]_{i,j=1(1)m}, \quad \mathfrak{D}_{ij} = \sum_{r=0}^{\nu_{ij}} p_r^{ij} \frac{d^r}{dx^r},$$

e que impõe ao problema (4.21) as condições diferenciais com coeficientes polinomiais

$$p_r^{ij} = \sum_{k=0}^{\partial_{ij}} p_{rk}^{ij} x^k = \mathcal{X}\mathsf{p}_r^{ij}.$$

Com esta notação, temos que

$$\begin{aligned} \mathfrak{D}\mathsf{y} = \mathsf{f} \quad &\Leftrightarrow \quad \sum_{i=1}^{m} \mathfrak{D}_{ij}(y_i) = \sum_{k=0}^{n} f_{i,k}x^k, \quad j = 1(1)m \\ &\Leftrightarrow \quad \sum_{i=1}^{m} \sum_{r=0}^{\nu_{ij}} p_r^{ij} \frac{d^r}{dx^r} y_i = \sum_{k=0}^{n} f_{i,k}x^k, \quad j = 1(1)m. \end{aligned}$$

Com isso, a ordem do sistema (4.21) é dada por

$$\nu = \sum_{i=1}^{m} \nu_i, \quad \nu_i = \max_{1\leq j\leq n} \nu_{ij}.$$

Definido o tipo de problema a abordar, passamos a associar-lhe o método tau, e para isso perturbamos o sistema (4.21) com m combinações lineares de grau $n+h$ de polinómios ortogonais $\mathcal{Z} = [Z_0,\ Z_1,\ \ldots]$

$$\tau_n = [\tau_{n1},\ \tau_{n2},\ \ldots,\ \tau_{nm}]^T, \quad \tau_{ni} = \sum_{k=n-\nu+1}^{n+h} \tau_{ik} Z_k$$

de forma a garantir a existência dos polinómios $\mathsf{y}_n = [y_{n1},\ y_{n2},\ \ldots,\ y_{nm}]$ que solucionam exatamente o problema agora perturbado

$$\begin{cases} \mathfrak{D}\mathsf{y}_n = \mathsf{f} + \tau_n, & x \in]a,\ b[\\ \mathsf{g}_k(\mathsf{y}_n) = \sigma_k, & k = 1(1)\nu \end{cases}, \tag{4.22}$$

e portanto, esperamos que $\mathsf{y}_n \approx \mathsf{y}$.

Considerando cada uma das aproximações na base $\mathcal{Z} = [Z_0,\ Z_1,\ \ldots]$, assim

$$y_{ni} = \sum_{j=0}^{n} a_{nij} Z_j = \mathcal{Z}\mathsf{a}_{ni}, \quad i = 1(1)m,$$

uma vez que são lineares as funcionais g_k, podemos escrever que

$$\begin{aligned} \mathsf{g}_k(\mathsf{y}_n) &= \sum_{i=1}^{m} g_{ki}(y_{ni}) \\ &= \sum_{i=1}^{m}\sum_{j=0}^{n} a_{nij} g_{ki}(Z_j), \quad k = 1(1)\nu, \end{aligned}$$

Desejamos, a partir daqui, montar um sistema de equações lineares algébricas que traduza simultaneamente as condições de contorno e a ação do operador $\mathfrak{D}$ no problema (4.22). Para tal efeito, truncando a base $\mathcal{Z}$ na dimensão n, o que implica em obter um polinómio de ordem n para cada aproximante y_{ni} e portanto com n coeficientes, temos

$$\mathsf{a} = [\mathsf{a}_{n1}^T,\ \mathsf{a}_{n2}^T,\ \ldots,\ \mathsf{a}_{nm}^T]^T,$$

donde o sistema linear a obter

$$\mathsf{T}\mathsf{a} = \mathsf{b} \tag{4.23}$$

terá $n \times m$ equações e incógnitas. A matriz T, será tal que receberá os blocos

$$\mathsf{C}_i = \begin{bmatrix} g_{1i}(Z_0) & g_{1i}(Z_1) & \ldots & g_{1i}(Z_n) \\ g_{2i}(Z_0) & g_{2i}(Z_1) & \ldots & g_{2i}(Z_n) \\ \vdots & \vdots & \ddots & \vdots \\ g_{\nu i}(Z_0) & g_{\nu i}(Z_1) & \ldots & g_{\nu i}(Z_n) \end{bmatrix}, \quad i = 1(1)m$$

que representam em suas ν linhas a aplicação das funcionais (4.23), e que portanto traduzem as condições de contorno, e relacionam-se com os componentes de a da forma

$$\sum_{i=1}^{m} \mathsf{C}_i \mathsf{a}_{ni} = \mathsf{s}, \quad \mathsf{s} = [\sigma_1,\ \sigma_2,\ \ldots,\ \sigma_\nu]^T,$$

e os blocos

$$\mathsf{D}_{ij} = \sum_{r=0}^{\nu_{ij}} p_r^{ij}(\mathsf{M})\mathsf{N}^r, \quad i,j = 1(1)m$$

que são as matrizes que representam, na base $\mathcal{Z}$, a ação do operador $\mathfrak{D}_{ij}$ sobre os polinómios Z_k, $k = 0(1)n$. Assim $\mathsf{D}_{ij}\mathsf{a}_{ni}$ irá representar, nas componentes de a_{ni}, tal ação de $\mathfrak{D}_{ij}$ nos coeficientes de cada aproximação y_{ni} e desta forma, as equações

$$\sum_{i=1}^{m} \mathfrak{D}_{ij}(y_{ni}) = f_j + \tau_{nj}, \quad j = 1(1)m$$

tomam a forma

$$\sum_{i=1}^{m} \mathsf{D}_{ij}\mathsf{a}_{ni} = \mathsf{f}_j, \quad j = 1(1)m,$$

onde f_j é o vetor que representa f_j na base $\mathcal{Z}$, e portanto satisfaz

$$\sum_{k=0}^{n_j} f_{jk} Z_k = \mathcal{Z}\mathsf{f}_j =, \quad \mathsf{f}_j = [\underbrace{f_{j0},\ f_{j1},\ \ldots,\ f_{jn_j},\ 0,\ \ldots,\ 0}_{n-\nu_j}]^T$$

com $n_j \leq n$, $j = 1(1)m$.

Conforme formulação, supra mencionada, T terá a forma mostrada na Figura 4.4

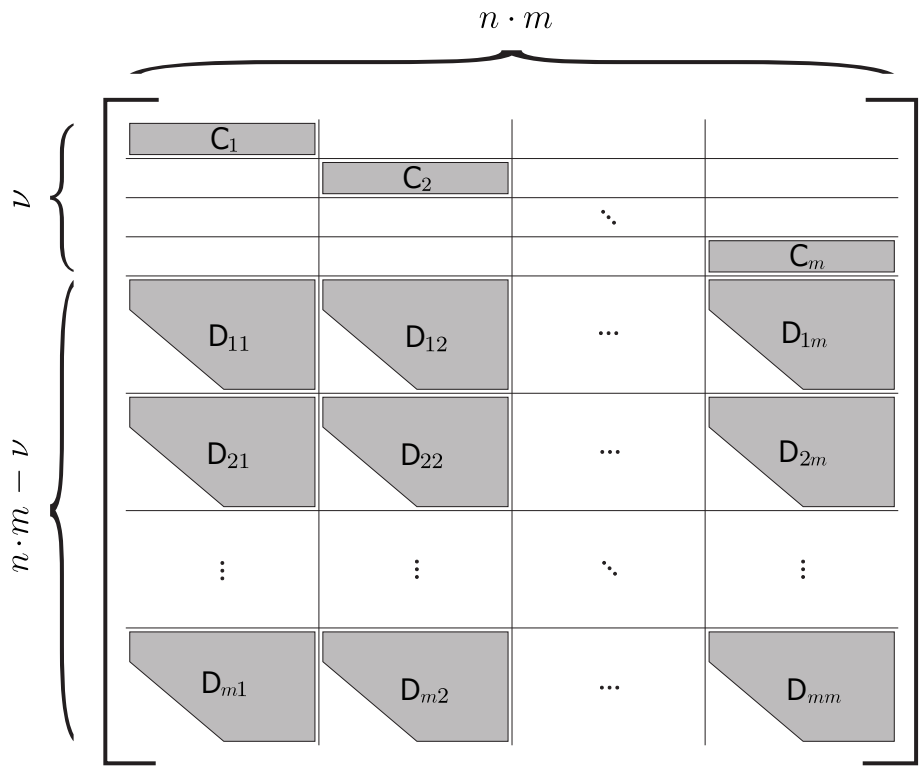

Figura 4.4: Formato da matriz T na aplicação do método tau para um sistema de equações diferencias ordinárias.

Finalmente, b em (4.23) é o vetor $\mathsf{b} = [\mathsf{s}^T,\ \mathsf{f}_1^T,\ \mathsf{f}_2^T,\ \ldots \mathsf{f}_m^T]^T$.

Afim de satisfazer compatibilidade nas dimensões de (4.23), cada um dos blocos D_{ij} é truncado nas suas n primeiras colunas, e nas suas $n - \nu_j$ linhas. Esta opção não constitui propriamente uma limitação do método, mas antes uma simplificação na notação. Existindo alguma razão para isso, é possível implementar o método com graus distintos em cada uma das aproximações para as diferentes componentes da solução vetorial.

4.5 Método tau parcelar

O método tau em sua versão parcelar, como sugerido por Ortiz (1975), consiste em decompor o domínio $[a,\ b]$ em subdomínios, e obter para cada um destes uma aproximação parcelar de y. Embora seja aplicável para problemas diferenciais com condições iniciais e com condições de fronteira, tem sua formulação de forma ligeiramente diferente, de forma a depender intrinsecamente de quais são estas condições. Comecemos por caracterizar os tipos de condições de contorno, e a seguir tratar de cada caso em sua versão parcelar separadamente.

Para o caso mais geral das aplicações do método tau que abordamos até agora,

as condições de contorno de um problema diferencial podem descrever-se pela ação de funcionais lineares da forma

$$g_j(y) = \sum_{r=0}^{\nu-1} \left(a_{jr} \left(\frac{d^r}{dx^r} y \right) \Big|_{x=a} + b_{jr} \left(\frac{d^r}{dx^r} y \right) \Big|_{x=b} \right), \quad j = 1(1)\nu, \tag{4.24}$$

onde a_{jr} são coeficientes reais que quando não nulos fornecem condições no início do intervalo, e b_{jr} são coeficientes reais que quando não nulos fornecem condições no final do intervalo.

Consideremos o problema de aproximar a solução de

$$\begin{cases} \mathfrak{D}y = f, & x \in]a,\ b[\\ g_j(y) = \sigma_j & j = 1(1)\nu \end{cases}, \tag{4.25}$$

onde $\mathfrak{D}$ é o operador diferencial de ordem ν com coeficientes polinomiais conforme definido para o problema (4.20), f é um polinómio ou aproximação polinomial de uma função definida no intervalo $]a,\ b[$ e g_j são as ν funcionais associadas às condições do problema, e portanto se reduzem a algum caso particular de (4.24).

Sejam $[x_0 = a,\ x_1] \cup [x_1,\ x_2] \cup \ldots \cup [x_{p-1},\ x_p = b]$ $p \geq 2$ sub-intervalos que particionam o intervalo $[a,\ b]$. Dizemos que o vetor de polinómios

$$y_n = [y_{n1},\ y_{n2},\ \ldots,\ y_{np}],$$

com y_{ni} definido para $x \in [x_{i-1},\ x_i]$, $i = 1(1)p$, é uma aproximação polinomial parcelar de grau $n-1$ da solução y_n do problema (4.25) se cada uma das componentes y_{ni} é uma aproximação de y no intervalo $[x_{i-1},\ x_i]$. Desta forma, $y_{ni},\ i = 1(1)p$ é a solução exata do problema perturbado (Onumanyi and Ortiz, 1984)

$$\begin{cases} \mathfrak{D}y_{ni} = f + \tau_{ni}, & x \in]x_{i-1},\ x_i[, \quad i = 1(1)p \\ \sum_{r=0}^{\nu-1} \left(a_{jr} \left(\frac{d^r}{dx^r} y_{n1} \right) \Big|_{x=a} + b_{jr} \left(\frac{d^r}{dx^r} y_{np} \right) \Big|_{x=b} \right) = \sigma_j, & j = 1(1)\nu \\ \left(\frac{d^r}{dx^r} y_{ni} \right) \Big|_{x=x_{i-1}} = \left(\frac{d^r}{dx^r} y_{n,i-1} \right) \Big|_{x=x_{i-1}}, & \begin{cases} i = 2(1)p \\ r = 0(1)\nu - 1 \end{cases} \end{cases}, \tag{4.26}$$

As condições (4.24) em (4.26) garantem que a aproximação y_n satisfaz as ν condições de (4.25), e as últimas equações em (4.26) tratam-se de condições impostas para garantir a regularidade da aproximação y_n nos $p-1$ pontos $x_1,\ x_2\ \ldots\ x_{p-1}$ na fronteira dos sub-intervalos que definem a partição. Com esta formulação, obtemos um sistema de p equações diferenciais de ordem ν com $p \times \nu$ condições fronteira.

4.5.1 Problemas de valor inicial

A forma mais intuitiva de aplicar o método tau na versão parcelar para problemas de valor inicial é aplicá-lo p vezes, onde as condições iniciais $\sigma_j,\ j = 1(1)\nu$, para aproximação em um passo seguinte, consistem exatamente na imagem do polinómio aproximante do passo anterior.

Proposição 4.4. *Obtemos uma aproximação tau parcelar da solução do problema* (4.26)*, resolvendo os p problemas*

$$\begin{cases} \mathfrak{D}y_{n1} = f + \tau_{n1}, & x \in]x_0,\ x_1[\\ g_j(y_{n1}) = \sigma_j, & j = 1(1)\nu \end{cases},$$

$$\begin{cases} \mathfrak{D}y_{ni} = f + \tau_{ni}, & x \in]x_{i-1},\ x_i[\\ g_j(y_{ni}) = \displaystyle\sum_{r=0}^{\nu-1} a_{jr} \left(\frac{d^r}{dx^r} y_{n,i-1} \right) \Bigg|_{x=x_{i-1}}, & j = 1(1)\nu \end{cases}, \quad i = 2(1)p,$$

onde $g_j(y_{ni})$ introduz em cada ponto x_{i-1} as condições iniciais impostas a x_0.

Na verificação de que a base de Legendre produz melhores aproximações no extremo direito do intervalo, quando comparada com outras bases, Ortiz (1975) apresentou a seguinte formulação parcelar: em cada um dos sub-intervalos $[x_{i-1},\ x_i]$, $i = 1(1)p$ construir uma aproximação

$$y_{ni}^{(\mathcal{T})} = \sum_{i=0}^{n-1} a_i T_i$$

obtida com a base de Chebyshev de primeira espécie e outra aproximação

$$y_{ni}^{(\mathcal{P})} = \sum_{i=0}^{n-1} a_i P_i$$

obtida com a base de Legendre, e assim obter simultaneamente uma aproximação para $y,\ x \in]x_{i-1},\ x_i[$ e uma para $y(x_i)$.

Proposição 4.5. *Obtemos uma aproximação tau parcelar da solução do problema* (4.26)*, resolvendo os p problemas*

$$\begin{cases} \mathfrak{D}y_{n1}^{(\mathcal{T})} = f + \tau_{n1}, & x \in]x_0,\ x_1[\\ g_j(y_{n1}^{(\mathcal{T})}) = \sigma_j, & j = 1(1)\nu \end{cases},$$

$$\begin{cases} \mathfrak{D}y_{ni}^{(\mathcal{T})} = f + \tau_{ni}, & x \in]x_{i-1},\ x_i[\\ \displaystyle\sum_{r=0}^{\nu-1} a_{jr} \left(\frac{d^r}{dx^r} y_{n,i}^{(\mathcal{T})} \right) \Bigg|_{x=x_{i-1}} = \sigma_j^{(i)}, & j = 1(1)\nu \end{cases}, \quad i = 2(1)p,$$

com

$$\sigma_j^{(i)} = \sum_{r=0}^{\nu-1} a_{jr} \left(\frac{d^r}{dx^r} y_{n,i-1}^{(\mathcal{P})} \right) \Bigg|_{x=x_{i-1}}.$$

Esta abordagem faz com que a aproximação seja descontínua nos $p-1$ pontos da partição, o que pode ser evitado se na construção de $y_{ni}^{(\mathcal{T})}$ o problema de valor inicial for substituído por um problema de valor de fronteira. Para isso, basta acrescentar as condições $y_{ni}^{(\mathcal{T})}(x_i) = y_{ni}^{(\mathcal{P})}(x_i)$. Aproximantes parcelares com um grau superior de regularidade nos pontos interiores da partição obtém-se ao impor tais condições sobre

$$\frac{d^r}{dx^r} y_{ni}^{(\mathcal{T})} \quad \text{e} \quad \frac{d^r}{dx^r} y_{ni}^{(\mathcal{P})}, \quad r = 1(1)\nu.$$

Nesta formulação resolvem-se $2p$ problemas, e as equações auxiliares para $y^{(\mathcal{P})}$ possuem a mesma forma, mas com as matrizes referentes à base $\mathcal{P}$ ($\mathsf{M}_{\mathcal{P}}$ e $\mathsf{N}_{\mathcal{P}}$).

4.5.2 Problemas de valor de fronteira

A versão parcelar do método tau para problemas de valor de fronteira não é tão intuitiva como no caso dos problemas de valor inicial, requerendo alguma formulação adicional sobre as condições de regularidade nas fronteiras de cada partição $[x_{i-1},\ x_i]$, $i = 1(1)p$. Visando obter esta formulação, consideremos duas matrizes cujos elementos são os coeficientes de (4.24)

$$\mathsf{A} = [a_{rj}]_{\nu\times\nu} \quad \text{e} \quad \mathsf{B} = [b_{kj}]_{\nu\times\nu}, \quad \begin{cases} j = 1(1)\nu \\ k = 0(1)\nu - 1 \end{cases},$$

e fixemos uma base $\mathcal{Z}^{(i)} = [Z_0^{(i)},\ Z_1^{(i)},\ \ldots]$ ortogonal em $[x_{i-1},\ x_i]$, $i = 1(1)p$, denotando por

$$\mathsf{R}^{(i)}(x) = [r_{kj}^{(i)}]_{\nu\times(n+1)}, \quad r_{kj}^{(i)} = \frac{d^k}{dx^k} Z_j^{(i)}, \quad \begin{cases} j = 0(1)n \\ k = 0(1)\nu - 1 \end{cases}$$

a matriz que possui as imagem dos elementos de $\mathcal{Z}^{(i)}$ e das suas derivadas em x.

Consideremos agora que em cada uma das parcelas $[x_{i-1},\ x_i]$ $i = 1(1)p$ da partição de $[a,\ b]$ existe um polinómio que aproxima y nesse segmento, sendo uma combinação linear daquela base $\mathcal{Z}^{(i)}$, ou seja

$$y_{ni} = \sum_{k=0}^{n} a_{nk}^{(i)} Z_{nk}^{(i)} = \mathcal{Z}^{(i)} \mathsf{a}_{ni}, \quad i = 1(1)p, \tag{4.27}$$

onde

$$\mathsf{a}_{ni} = [a_{n0}^{(i)},\ a_{n1}^{(i)},\ \ldots,\ a_{nn}^{(i)}]^T. \tag{4.28}$$

Se cada um destes polinómios forem agrupados em um único vetor, temos

$$\mathsf{a} = [\mathsf{a}_{n1}^T,\ \mathsf{a}_{n2}^T,\ \ldots,\ \mathsf{a}_{np}^T]^T. \tag{4.29}$$

A condição (4.24) pode então ser traduzida para a forma matricial

$$\mathsf{A}\mathsf{R}^{(1)}(x_0)\mathsf{a}_{n1} + \mathsf{B}\mathsf{R}^{(p)}(x_p)\mathsf{a}_{np} = \mathsf{s}, \tag{4.30}$$

onde $\mathsf{s} = [\sigma_1,\ \sigma_2,\ \ldots,\ \sigma_\nu]^T$, e as condições de regularidade

$$\left(\frac{d^r}{dx^r} y_{ni}\right)\Big|_{x=x_{i-1}} = \left(\frac{d^r}{dx^r} y_{n,i-1}\right)\Big|_{x=x_{i-1}}, \quad \begin{cases} i = 2(1)p \\ r = 0(1)\nu - 1 \end{cases}$$

traduzidas para a forma matricial

$$\mathsf{R}^{(i-1)}(x_{i-1})\mathsf{a}_{n,i-1} - \mathsf{R}^{(i)}(x_{i-1})\mathsf{a}_{ni} = 0, \quad i = 2(1)p. \tag{4.31}$$

Seja $\mathsf{D}_{\mathcal{Z}^{(i)}}$ a matriz que traduz a ação do operador $\mathfrak{D}$ em termos matriciais nos elementos da base $\mathcal{Z}^{(i)}$, e $\mathsf{f}_i = [f_{i0},\ f_{i1},\ \ldots,\ f_{im},\ 0,\ 0,\ \ldots]^T$ o vetor que tem nas suas

primeiras $m+1$ componentes os coeficientes da representação do polinómio f na base $\mathcal{Z}^{(i)}$. Nestas condições cada um dos vetores em (4.28) deve satisfazer a equação

$$\mathsf{D}_{\mathcal{Z}^{(i)}}\mathsf{a}_{ni} = \mathsf{f}_i,$$

referentes aos coeficientes dos polinómios que aproximam y no intervalo $[x_{i-1},\ x_i]$ e portanto associados à perturbação

$$\tau_{ni} = \sum_{j=n-\nu+1}^{\nu+h} \tau_{ij} Z^{(i)}_{n-\nu+j}, \quad \tau_{ij} = \mathsf{e}_j^T \mathsf{D}_{\mathcal{Z}^{(i)}}\mathsf{a}_{ni}.$$

Proposição 4.6. *Seja*

$$\mathsf{T} = \begin{bmatrix} \mathsf{D}_{\mathcal{Z}^{(1)}}\mathsf{A}\mathsf{R}^{(1)}_{x_0} & & & & & \mathsf{B}\mathsf{R}^{(1)}_{x_p} \\ \mathsf{R}^{(1)}_{x_1} & -\mathsf{D}_{\mathcal{Z}^{(2)}}\mathsf{R}^{(2)}_{x_1} & & & & \\ & \mathsf{R}^{(2)}_{x_2} & -\mathsf{D}_{\mathcal{Z}^{(3)}}\mathsf{R}^{(3)}_{x_2} & & & \\ & & & \ddots & & \\ & & & & \mathsf{R}^{(p-1)}_{x_{p-1}} & -\mathsf{D}_{\mathcal{Z}^{(p)}}\mathsf{R}^{(p)}_{x_{p-1}} \end{bmatrix}$$

a matriz construída por blocos, com $\mathsf{R}^{(i)}_{x_j} = \mathsf{R}^{(i)}(x_j)$ *e*

$$\mathsf{b} = [\mathsf{b}_1^T,\ \mathsf{b}_2^T,\ \ldots,\ \mathsf{b}_p^T]^T, \quad \mathsf{b}_i = \begin{cases} \mathsf{s} + \mathsf{M}^\nu \mathsf{f}_i, & i = 1 \\ \mathsf{M}^\nu \mathsf{f}_i, & i = 2(1)p \end{cases}.$$

Se T *é não singular, então a solução tau parcelar* y_{ni} *para o problema* (4.25) *na forma* (4.27) *pode obter-se na forma do vetor* a (4.29)*, resolvendo o sistema* $\mathsf{T}\mathsf{a} = \mathsf{b}$.

As matrizes resultantes de (4.30) e(4.31) e que integram os blocos de T devem estar truncadas às suas primeiras $n+1$ colunas e primeiras $n+1-\nu$ linhas. Da mesma forma os vetores b_i devem estar truncados às suas primeiras $n+1-\nu$ linhas para a compatibilidade do sistema.

4.6 Método tau para equações envolvendo integrais

Com a mesma formulação operacional do método tau, é possível generalizar a sua aplicação a problemas integro-diferenciais. Tratando-se de primitivas simples, as equações integrais traduzem-se em termos operacionais com o uso da matriz O, apresentada em 4.6, de forma análoga ao proposto às equações diferenciais. Já os integrais envolvendo um núcleo e limites de integração, como os de Fredholm e Volterra, traduzem-se em termos operacionais com o uso desta mesma matriz O mas requer alguma formulação adicional, conforme apresentado a seguir.

4.6.1 Equações integro-diferenciais elementares

Seja $\mathcal{Z} = [Z_0,\ Z_1,\ \ldots]$ uma base de polinómios ortogonais e seja $y_n = \sum_{i=0}^{n} a_i Z_i = \mathcal{Z}\mathsf{a}_{\mathcal{Z}}$ um polinómio na base $\mathcal{Z}$, de coeficientes $\mathsf{a}_{\mathcal{Z}} = [a_0,\ \ldots,\ a_n,\ 0,\ 0,\ \ldots]^T$. Seja ainda o problema-integro diferencial

$$\begin{cases} \mathfrak{D}y + \mathfrak{S}y = f, \quad x \in]a,\ b[\\ g_j(y) = \sigma_j, \quad j = 1(1)\nu \end{cases}, \tag{4.32}$$

com

$$\mathfrak{D} = \sum_{k=0}^{\nu} p_k^{(\mathfrak{D})} \frac{d^k}{dx^k} \quad \text{e} \quad \mathfrak{S} = \sum_{k=1}^{\gamma} p_k^{(\mathfrak{S})} \underbrace{\int^x \ldots \int^x}_{k \text{ vezes}},$$

onde $p_k^{(\mathfrak{D})},\ k = 0(1)\nu$ e $p_k^{(\mathfrak{S})},\ k = 1(1)\gamma$ são polinómios ou aproximações polinomiais. Perturbando o problema (4.32), temos o problema tau

$$\begin{cases} \mathfrak{D}y_n + \mathfrak{S}y_n = f + \sum_{i=n-\nu+1}^{n+h} \tau_i Z_i, \quad x \in]a,\ b[\\ g_j(y_n) = \sigma_j, \quad j = 1(1)\nu \end{cases},$$

onde através dos Teoremas 4.1 e 4.2 $\mathfrak{D}$ e $\mathfrak{S}$ traduzem-se em sua formulação operacional, respetivamente, como

$$\mathfrak{D}y_n = \mathcal{Z}\mathsf{D}_{\mathcal{Z}}\mathsf{a}_{\mathcal{Z}}, \quad \mathsf{D}_{\mathcal{Z}} = \sum_{k=0}^{\nu} p_k^{(\mathfrak{D})}(\mathsf{M}_{\mathcal{Z}})\mathsf{N}_{\mathcal{Z}}^{\mathsf{k}}$$

e

$$\mathfrak{S}y_n = \mathcal{Z}\mathsf{S}_{\mathcal{Z}}\mathsf{a}_{\mathcal{Z}}, \quad \mathsf{S}_{\mathcal{Z}} = \sum_{k=0}^{\gamma} p_k^{(\mathfrak{S})}(\mathsf{M}_{\mathcal{Z}})\mathsf{O}_{\mathcal{Z}}^{\mathsf{k}}.$$

Considerando as ν condições de contorno, avaliadas como $\mathsf{C}_{\mathcal{Z}} = [c_{ij}]_{\nu\times\infty},\ c_{ij} = g_i(Z_j),\ i = 1(1)\nu,\ j \geq 0$, temos o sistema duplamente infinito

$$\begin{cases} \mathsf{C}_{\mathcal{Z}}\mathsf{a}_{\mathcal{Z}} = \mathsf{s} \\ (\mathsf{D}_{\mathcal{Z}} + \mathsf{S}_{\mathcal{Z}})\mathsf{a}_{\mathcal{Z}} = \mathsf{f}_{\mathcal{Z}} \end{cases},$$

onde $\mathsf{s} = [\sigma_1,\ \ldots,\ \sigma_\nu]^T$ e $\mathsf{f}_{\mathcal{Z}} = [f_0,\ \ldots,\ f_\lambda,\ 0,\ 0,\ \ldots]^T$ tal que $f_{\mathcal{Z}} = \sum_{i=0}^{\lambda} f_i Z_i$. Concatenando vetores e matrizes, os coeficientes $\mathsf{a}_{\mathcal{Z}}$ da solução, são solução do sistema $\mathsf{T}_{\mathcal{Z}}\mathsf{a}_{\mathcal{Z}} = \mathsf{b}_{\mathcal{Z}}$, onde

$$\mathsf{T}_{\mathcal{Z}} = \begin{bmatrix} \mathsf{C}_{\mathcal{Z}} \\ \mathsf{D}_{\mathcal{Z}} + \mathsf{S}_{\mathcal{Z}} \end{bmatrix}, \quad \text{e} \quad \mathsf{b}_{\mathcal{Z}} = \begin{bmatrix} \mathsf{s} \\ \mathsf{f}_{\mathcal{Z}} \end{bmatrix}.$$

Ao truncarmos e solucionarmos o sistema resultante na dimensão n, temos uma aproximação polinomial de ordem $n-1$ para a solução y.

4.6.2 Equações integro-diferenciais de Fredholm-Volterra

A classe de equações cuja solução pretendemos aproximar aqui, são conhecidas como equações íntegro-diferenciais de Fredholm-Volterra, por envolverem na mesma equação os termos que definem estes operadores integrais e portanto o problema a resolver é da forma

$$\begin{cases} \mathfrak{D}y + \mathfrak{S}_F y + \mathfrak{S}_V y = f, \quad x \in]a,\ b[\\ g_i(y) = \sigma_i, \quad i = 1(1)\nu \end{cases}, \tag{4.33}$$

onde $\mathfrak{D}$ é o operador diferencial definido em (4.3),

$$\mathfrak{S}_F y = \sum_{i=1}^{\gamma_F} p_i \int_a^b K_i(x,t)y(t)dt,$$

$$\mathfrak{S}_V y = \sum_{i=1}^{\gamma_V} q_i \int_a^x K_i(x,t)y(t)dt,$$

K_i são polinómios de duas variáveis ou aproximações polinomiais, assim como f, p_i e q_i são polinómios na variável x ou então aproximações polinomiais, portanto

$$K_*(x,t) = \sum_{i=0}^{n_x}\sum_{j=0}^{n_t} k_{i,j}^{(*)} x^i t^j = \mathcal{X}\mathsf{K}_*\mathcal{X}^T(t),$$

onde $\mathcal{X}(t) = [1,\ t,\ t^2,\ \ldots]$ e tomando $n_x = n_t = N$

$$\mathsf{K}_* = \begin{bmatrix} k_{00}^{(*)} & k_{01}^{(*)} & \ldots & k_{0N}^{(*)} \\ k_{10}^{(*)} & k_{11}^{(*)} & \ldots & k_{1N}^{(*)} \\ \vdots & \vdots & \ddots & \vdots \\ k_{N0}^{(*)} & k_{N1}^{(*)} & \ldots & k_{NN}^{(*)} \end{bmatrix}.$$

Se existe pelo menos um índice i tal que $p_i \neq 0,\ i = 1(1)\gamma_F$ e se existe pelo menos um índice i tal que $q_i \neq 0,\ i = 1(1)\gamma_V$, então a equação (4.33) é um problema integro-diferencial de Fredholm-Volterra, para $p_i = 0,\ i = 1(1)\gamma_F$ integro-diferencial de Volterra e para $q_i = 0,\ i = 1(1)\gamma_V$ integro-diferencial de Fredholm.

Definido o problema, passamos a abordar o método tau para esta classe de equações. Os lemas e teoremas seguintes mostram como obter, respetivamente, a representação matricial na base das potências ($\mathcal{X}$) dos termos de Fredholm e de Volterra (AliAbadi and Shahmorad, 2002; Hosseini and Shahmorad, 2003b; Saeedi et al., 2013; Rahimi et al., 2010).

Teorema 4.5. *Para um operador integral*

$$\int^x K(x,t)y(t)dt,$$

onde $\int^x$ *significa avaliar o integral em* x, $K(x,t) = \sum_{i=0}^{n_x}\sum_{j=0}^{n_t} k_{ij}x^i t^j$ *e* $y = \sum_{i=0}^{\infty} a_i x^i = \mathcal{X}\mathsf{a}$, *temos que*

$$\int^x K(x,t)y(t)dt \quad = \quad \mathcal{X}\mathsf{S}^{(I)}\mathsf{a}$$

$$= \mathcal{Z}\mathsf{S}_{\mathcal{Z}}^{(I)}\mathsf{a}_{\mathcal{Z}},$$

onde $\mathsf{S}_{\mathcal{Z}}^{(I)} = \mathsf{V}_{\mathcal{Z}}^{-1}\mathsf{S}^{(I)}\mathsf{V}_{\mathcal{Z}}$ *e*

$$\mathsf{S}^{(I)} = \sum_{i=0}^{n_x}\sum_{j=0}^{n_t} k_{ij}\mathsf{M}^i\mathsf{O}\mathsf{M}^j.$$

Demonstração. Imediata, por linearidade do operador e, uma vez que

$$\int^x x^i t^j y(t)dt = \mathcal{X}\mathsf{M}^i\mathsf{O}\mathsf{M}^j\mathsf{a}.$$

□

Lema 4.1. *Sendo* $y = \mathcal{X}\mathsf{a}$ *então temos que*

$$\int_a^x x^i t^j y(t)dt = \mathcal{X}(\mathsf{M}^i\mathsf{O}\mathsf{M}^j - \mathsf{e}_{i+1}\mathsf{z})\mathsf{a},$$

onde e_{i+1} *é a* $(i+1)$*-ésima coluna da matriz identidade e* $\mathsf{z} = \mathcal{X}(a)\mathsf{O}\mathsf{M}^j$.

Demonstração. A partir do Teorema 4.5, escrevemos

$$\begin{aligned}\int_a^x x^i t^j y(t)dt &= \left(\mathsf{M}^i\mathcal{X}\Big|_a^x \mathsf{O}\mathsf{M}^j\right)\mathsf{a} \\ &= \mathcal{X}\mathsf{M}^i\mathsf{O}\mathsf{M}^j\mathsf{a} - \mathsf{M}^i\mathcal{X}(a)\mathsf{O}\mathsf{M}^j\mathsf{a}\end{aligned}$$

e uma vez que estando na base das potências $\mathsf{M}^i\mathsf{e}_1 = \mathsf{e}_{i+1}$, escrevemos

$$\begin{aligned}\mathsf{M}^i\mathcal{X}(a)\mathsf{O}\mathsf{M}^j\mathsf{a} &= \mathsf{M}^i\mathcal{X}\mathsf{e}_1\mathcal{X}(a)\mathsf{O}\mathsf{M}^j\mathsf{a} \\ &= \mathsf{M}^i\mathcal{X}\mathsf{e}_1\mathsf{z}\mathsf{a} \\ &= \mathcal{X}\mathsf{M}^i\mathsf{e}_1\mathsf{z}\mathsf{a} \\ &= \mathcal{X}\mathsf{e}_{i+1}\mathsf{z}\mathsf{a}.\end{aligned}$$

□

Lema 4.2. *Sendo* $y = \mathcal{X}\mathsf{a}$ *então temos que*

$$\int_a^b x^i t^j y(t)dt = \mathcal{X}\mathsf{e}_{i+1}\left(\mathsf{z}_{ij}(b) - \mathsf{z}_{ij}(a)\right)\mathsf{a},$$

onde $\mathsf{z}_{ij}(c) = \mathcal{X}(c)\mathsf{O}\mathsf{M}^j$, *correspondente ao termo* $x^i t^j$ *do núcleo* K.

Demonstração. Similar à demonstração do Lema 4.1. □

Com os dois lemas acima apresentados, temos a estrutura para transformar os operadores integrais de Fredholm e de Volterra em termos matriciais, e portanto podemos apresentar o resultado a seguir:

Teorema 4.6. *Considerando* $K_*(x,t) = \sum_{i=0}^{n_x}\sum_{j=0}^{n_t} k_{ij}^{(*)} x^i t^j$, *então*

$$\int_a^x K_1(x,t)y(t)dt = \mathcal{X}\mathsf{S}^{(V)}\mathsf{a} \tag{4.34}$$

e

$$\int_a^b K_2(x,t)y(t)dt = \mathcal{X}\mathsf{S}^{(F)}\mathsf{a}, \tag{4.35}$$

onde

$$\mathsf{S}^{(V)} = \sum_{i=0}^{n_x}\sum_{j=0}^{n_t} k_{ij}^{(1)}\left(\mathsf{M}^i\mathsf{O}\mathsf{M}^j - \mathsf{e}_{i+1}\mathsf{z}_{ij}(a)\right) \tag{4.36}$$

e

$$\mathsf{S}^{(F)} = \sum_{i=0}^{n_x}\sum_{j=0}^{n_t} k_{ij}^{(2)}\mathsf{e}_{i+1}\Big(\mathsf{z}_{ij}(b) - \mathsf{z}_{ij}(a)\Big). \tag{4.37}$$

Demonstração. Verificar que estes resultados são, respetivamente combinações lineares dos Lemas 4.1 e 4.2. □

Corolário 4.1. *Se nas expressões* (4.34) *e* (4.35) *acrescentarmos, respetivamente, os coeficientes polinomiais* p *e* q, *então as matrizes* (4.36) *e* (4.37) *são substituídas, respetivamente, por*

$$\mathsf{S}^{(V)} = p(\mathsf{M})\sum_{i=0}^{n_x}\sum_{j=0}^{n_t} k_{ij}^{(1)}\Big(\mathsf{M}^i\mathsf{O}\mathsf{M}^j - \mathsf{e}_{i+1}\mathsf{z}_{ij}(a)\Big)$$

e

$$\mathsf{S}^{(F)} = q(\mathsf{M})\sum_{i=0}^{n_x}\sum_{j=0}^{n_t} k_{ij}^{(2)}\mathsf{e}_{i+1}\Big(\mathsf{z}_{ij}(b) - \mathsf{z}_{ij}(a)\Big).$$

Obs. De igual modo são válidas transformações de semelhanças traduzindo $\mathcal{X}\mathsf{S}^{(V)}\mathsf{a}$ na base canónica em $\mathcal{Z}\mathsf{S}_{\mathcal{Z}}^{(V)}\mathsf{a}_{\mathcal{Z}}$ na base $\mathcal{Z}$ e $\mathcal{X}\mathsf{S}^{(F)}\mathsf{a}$ em $\mathcal{Z}\mathsf{S}_{\mathcal{Z}}^{(F)}\mathsf{a}_{\mathcal{Z}}$.

Com a formulação acima apresentada, à equação (4.33) passamos a associar-lhe o problema tau, o qual representado na forma matricial conduz ao sistema linear $\mathsf{T}_{\mathcal{Z}}\mathsf{a}_{\mathcal{Z}} = \mathsf{b}_{\mathcal{Z}}$ truncado na ordem n, onde

$$\mathsf{T}_{\mathcal{Z}} = \begin{bmatrix} \mathsf{C}_{\mathcal{Z}} \\ \mathsf{D}_{\mathcal{Z}} + \mathsf{S}_{\mathcal{Z}}^{(V)} + \mathsf{S}_{\mathcal{Z}}^{(F)} \end{bmatrix}, \quad \mathsf{C}_{\mathcal{Z}} = \begin{bmatrix} g_1(Z_o) & \dots & g_1(Z_n) \\ \vdots & \ddots & \vdots \\ g_\nu(Z_o) & \dots & g_\nu(Z_n) \end{bmatrix},$$

$$\mathsf{a}_{\mathcal{Z}} = [a_0,\ a_1,\ \dots,\ a_n]^T, \quad \mathsf{b}_{\mathcal{Z}} = [\sigma_1,\ \dots,\ \sigma_\nu,\ \mathsf{f}_{\mathcal{Z}}^T,\ 0,\ 0,\ \dots,\ 0]^T.$$

4.7 O método tau para problemas não lineares

Se, ao invés de ser linear, a equação incluir termos não lineares que envolvem a solução desconhecida y e/ou as suas derivadas, podemos sempre descrever o problema como

$$F\left(y, \frac{d}{dx}y, \dots, \frac{d^\nu}{dx^\nu}y\right) = 0 \tag{4.38}$$

onde F é um operador não linear agindo sobre um apropriado espaço de funções suaves e contínuas.

Seja y uma solução de (4.38). Se F tem todas as suas primeiras derivadas parciais contínuas em uma vizinhança Ω de

$$\omega = \left(y, \frac{d}{dx}y, \ldots, \frac{d^\nu}{dx^\nu}y \right)$$

e se

$$\omega^{(0)} = \left(y^{(0)}, \frac{d}{dx}y^{(0)}, \ldots, \frac{d^\nu}{dx^\nu}y^{(0)} \right) \in \Omega$$

é uma aproximação de ω, então, admitindo que todas as derivadas parciais de F são contínuas em Ω, podemos definir um operador linear T representado pelo polinómio de Taylor de ordem um centrado em $\omega^{(0)}$

$$T\left(y, \frac{d}{dx}y, \ldots, \frac{d^\nu}{dx^\nu}y \right) = F(\omega^{(0)}) + \sum_{i=0}^{\nu} \left. \frac{\partial F}{\partial \frac{d^i}{dx^i}y} \right|_{\omega^{(0)}} \left(\frac{d^i}{dx^i}y - \frac{d^i}{dx^i}y^{(0)} \right).$$

Assim como no método de Newton para problemas algébricos, podemos substituir F por T em (4.38) e então resolver a equação aproximada

$$\sum_{i=0}^{\nu} \left. \frac{\partial F}{\partial \frac{d^i}{dx^i}y} \right|_{\omega^{(0)}} \frac{d^i}{dx^i}y = -F(\omega^{(0)}) + \sum_{i=0}^{\nu} \left. \frac{\partial F}{\partial \frac{d^i}{dx^i}y} \right|_{\omega^{(0)}} \frac{d^i}{dx^i}y^{(0)} \tag{4.39}$$

Aplicando o método tau para a equação diferencial linear (4.39) e tomando

$$w^{(1)} = \left(y^{(1)}, \frac{d}{dx}y^{(1)}, \ldots, \frac{d^\nu}{dx^\nu}y^{(1)} \right)$$

como solução, se $\omega^{(1)} \in \Omega$ podemos repetir o esquema, obtendo um processo iterativo, resolvendo para

$$\omega^{(k)} = \left(y^{(k)}, \frac{d}{dx}y^{(k)}, \ldots, \frac{d^\nu}{dx^\nu}y^{(k)} \right)$$

a equação diferencial linear

$$\sum_{i=0}^{\nu} \left. \frac{\partial F}{\partial \frac{d^i}{dx^i}y} \right|_{\omega^{(k-1)}} \frac{d^i}{dx^i}y^{(k)} = -F(\omega^{(k-1)}) + \sum_{i=0}^{\nu} \left. \frac{\partial F}{\partial \frac{d^i}{dx^i}y} \right|_{\omega^{(k-1)}} \frac{d^i}{dx^i}y^{(k-1)}, \ k = 1, 2, \ldots$$

Ilustraremos o processo de linearização supramencionado com os dois exemplos a seguir.

Exemplo 4.3. Seja a equação diferencial não linear homogénea

$$\begin{cases} \dfrac{d}{dx}y - 2\cos(y) = 0, \ x \in]0,1[\\ y(0) = 0 \end{cases},$$

que possui solução exata $y = 2\arctan(\tanh(x))$.

Solução. Aplicando o processo de linearização de Newton, encontramos uma recorrência baseada nas equações diferenciais lineares

$$\frac{d}{dx}y^{(k)} + 2\sin(y^{(k-1)})y^{(k)} = 2\cos(y^{(k-1)}) + 2y^{(k-1)}\sin(y^{(k-1)}),\ k = 1, 2, \ldots$$

Para $y^{(0)}$ podemos tomar a função constante $y = 0$, satisfazendo $y(0) = 0$. Com este ponto de partida, chegamos ao primeiro problema diferencial $\frac{d}{dx}y^{(1)} = 2$ com $y^{(1)}(0) = 0$ resultando em $y^{(1)} = 2x$ e, no segundo problema diferencial

$$\frac{d}{dx}y^{(2)} + 2\sin(2x)y^{(2)} = 2\cos(2x) + 4x\sin(2x)$$

com $y^{(2)}(0) = 0$, no qual a solução já pode ser aproximada pelo método tau.

Exemplo 4.4. Seja a equação diferencial não linear homogénea

$$\begin{cases} \dfrac{d^2}{dx^2}yy + \left(\dfrac{x}{\lambda} - 1\right)\left(\dfrac{d}{dx}y\right)^2 = 0,\ x \in]0,\ 1[\\ y(0) = -1,\ y(1) = (1-\lambda)/(1+\lambda) \end{cases},$$

onde para um parâmetro $\lambda \in \mathbb{R}\backslash[-1,\ 0]$ tem solução exata $y = (x-\lambda)/(x+\lambda),\ x \in [0,\ 1]$.

Solução. O processo de linearização produz

$$y^{(k-1)}\frac{d^2}{dx^2}y^{(k)} + 2\frac{d}{dx}y^{(k-1)}\left(\frac{x}{\lambda} - 1\right)\frac{d}{dx}y^{(k)} + \frac{d^2}{dx^2}y^{(k-1)}y^{(k)} = \\ = \frac{d^2}{dx^2}y^{(k-1)}y^{(k-1)} + \left(\frac{x}{\lambda} - 1\right)\left(\frac{d}{dx}y^{(k-1)}\right)^2,\ k = 1, 2, \ldots$$

Para $y^{(0)}$ podemos tomar a reta que interpola $y(0) = -1$ e $y(1) = (1-\lambda)/(1+\lambda)$, portanto $y^{(0)} = \frac{2}{1+\lambda}x - 1$.

Obs. Nesta formulação, há um aspecto de implementação importante: consideremos um termo não linear e sua expansão em série de Taylor truncada na primeira ordem

$$y_1(t)y_2(t) \approx y_1^{(k)}y_2^{(k+1)} + y_2^{(k)}y_1^{(k+1)} - y_1^{(k)}y_2^{(k)},$$

ao utilizarmos esta aproximação, os termos $y_1^{(k)}y_2^{(k+1)} + y_2^{(k)}y_1^{(k+1)}$ definem o operador do problema a resolver e são calculados como $y_1^{(k)}(\mathsf{M}_{\mathcal{Z}})y_2^{(k+1)} + y_2^{(k)}(\mathsf{M}_{\mathcal{Z}})y_1^{(k+1)}$, enquanto $y_1^{(k)}y_2^{(k)}$ define o segundo membro da equação e trata-se do produto de dois polinómios.

4.8 Conclusões e considerações

Neste capítulo apresentamos algumas das principais formulações do método tau encontradas na literatura. Resultados mostram que, com a formulação usual e para valores elevados do grau n de aproximação, os erros introduzidos na transcrição do problema para o formato matricial propagam-se de forma significativa e repercutem-se diretamente na qualidade da aproximação. Em alguns casos, esta propagação dos erros pode destruir as propriedades de convergência do método. Aqui encerramos a Parte I da Tese. Na Parte que segue, apresentamos os contributos desenvolvidos e aplicados para a estabilização e extensão do método tau, visando garantir melhor qualidade de aproximação mesmo para n elevados.

Parte II

Contributos para a estabilidade do método tau

Capítulo 5

Exploração da recursividade

Sumário

No Capítulo 4 mostramos algumas das formulações para o método tau encontradas na literatura, e visto que as suas implementações em sistemas de computação, com precisão finita, podem ser mais ou menos estáveis, dedicamos este capítulo a desenvolver alguns contributos de estabilidade relacionados, em sua maioria, à recursividade dos polinómios ortogonais. Para evidenciar a precisão das implementações com os contributos propostos, trazemos algumas comparações destas com as formas usuais da literatura acerca do método tau.

5.1 Avaliação em bases ortogonais de polinómios

Seja $\mathcal{Z} = [Z_0,\ Z_1,\ \ldots]$ uma sequência de polinómios ortogonais, $\mathsf{V}_{\mathcal{Z}}$ a matriz que projeta $\mathcal{Z}$ em $\mathcal{X}$ e $y_n = \sum_{i=0}^{n} a_{\mathcal{Z},i} Z_i$ uma determinada aproximação para uma função y, de coeficientes $\mathsf{a}_{\mathcal{Z}} = [a_{\mathcal{Z},0},\ a_{\mathcal{Z},1},\ \ldots,\ a_{\mathcal{Z},n}]$, uma forma de avaliar $y_n(x)$ é passar $\mathsf{a}_{\mathcal{Z}}$ para a base das potências, ou seja,

$$\mathsf{a} = \mathsf{V}_{\mathcal{Z}} \mathsf{a}_{\mathcal{Z}} \tag{5.1}$$

e então usar a função nativa do MATLAB `polyval` em a.

Esta abordagem mostra-se instável à medida que o grau n cresce, e assim, uma alternativa é evitar a mudança de base avaliando y_n na própria base de polinómios ortogonais $\mathcal{Z}$. Com este objetivo, definimos:

Definição 5.1. *Sejam* p *e* q *dois vetores quaisquer de igual dimensão e cujas componentes são escalares reais. Define-se como* $\odot$ *a operação que fornece o produto*

$$\mathsf{p} \odot \mathsf{q} = [p_0,\ p_1,\ \ldots] \odot [q_0,\ q_1,\ \ldots] = [p_0 q_0,\ p_1 q_1,\ \ldots].$$

Com esta operação e com o resultado da proposição seguinte, é possível calcular as imagens dos polinómios Z_i nos elementos de um vetor $\mathsf{x} = [x_0,\ x_1,\ \ldots]^T$, utilizando uma relação de recorrência.

Proposição 5.1. *Seja* $\mathcal{Z}^* = [Z_0^*,\ Z_1^*,\ \ldots]$ *uma sequência de polinómios ortogonais no intervalo* $[a,\ b]$. *Sejam ainda* $\mathsf{x} = [x_0,\ x_1,\ \ldots,\ x_k]^T$ *um vetor de escalares e* $y(\mathsf{x}) = \sum_{i=0}^{n} a_{\mathcal{Z}^*,i} Z_i^*(\mathsf{x}) = \mathsf{y}$ *uma combinação linear de* $\mathcal{Z}^*$ *em elementos de* x. *Nestas condições, avalia-se* y *diretamente na base ortogonal por*

$$\begin{cases} Z_i^*(\mathsf{x}) = \mathsf{p}_i = \dfrac{(c_1\mathsf{x} + (c_2 - \beta_{i-1})r) \odot \mathsf{p}_{i-1} - \gamma_{i-1}\mathsf{p}_{i-2}}{\alpha_{i-1}}, \quad i = 2(1)n \\ Z_1^*(\mathsf{x}) = \mathsf{p}_1 = \dfrac{(c_1\mathsf{x} + c_2\mathsf{r}) - \beta_0\mathsf{r}}{\alpha_0} \\ Z_0^*(\mathsf{x}) = \mathsf{p}_0 = \mathsf{r} \end{cases},$$

onde $c_1 = \dfrac{2}{b-a}$, $c_2 = -\dfrac{a+b}{b-a}$, $\mathsf{r} = [\overbrace{1,\ 1,\ \ldots,\ 1}^{k+1\ vezes}]^T$ *e* $\alpha_i,\ \beta_i,\ \gamma_i$ *são os coeficientes da relação de recorrência característica dos polinómios* Z_i *ortogonais no intervalo* $[-1,\ 1]$.

Demonstração. Conforme (2.12), a mudança de intervalo de ortogonalidade dá-se fazendo

$$x = \left(\frac{2x^* - (a+b)}{b-a}\right), \tag{5.2}$$

de onde tiramos que c_1 é o coeficiente de x^* e c_2 o termo independente em (5.2). Partindo da relação de recorrência (2.1) transladada por (5.2) temos

$$\begin{cases} (c_1 x^* + c_2) Z_i = \alpha_i Z_{i+1} + \beta_i Z_i + \gamma_i Z_{i-1}, \quad i \geq 0 \\ Z_0 = 1, \quad Z_{-1} = 0 \end{cases}, \tag{5.3}$$

observamos diretamente que $Z_0(\mathsf{x})$ sempre será um vetor de 1's para qualquer x. Quanto a $Z_1(\mathsf{x})$, observamos que

$$Z_1 = \frac{1}{\alpha_0} x - \frac{\beta_0}{\alpha_0},$$

de onde segue que $Z_1(\mathsf{x}) = \dfrac{c_1\mathsf{x} + (c_2 - \beta_0)\mathsf{r}}{\alpha_0}$. E finalmente voltando à relação de recorrência (5.3), e explicitando Z_{n+1} temos

$$Z_{i+1} = \frac{(c_1 x^* + c_2 - \beta_i) Z_i - \gamma_i Z_{i-1}}{\alpha_i}, \quad i \geq 0,$$

de onde encontramos exatamente

$$Z_i(\mathsf{x}) = \frac{(c_1\mathsf{x} + (c_2 - \beta_{i-1})\mathsf{r}) \odot \mathsf{p}_{i-1} - \gamma_{i-1}\mathsf{p}_{i-2}}{\alpha_{i-1}},$$

que são os demais polinómios para $i = 2(1)n$. Para terminar a demonstração, basta verificar que $y = \sum_{i=0}^{n} a_{Z,i} p_i$. □

Utilizando o resultado desta proposição foi implementada a função **orthoval**, capaz de avaliar as imagens $y = \sum_{i=0}^{n} a_{\mathcal{Z},i} Z_i$ a partir dos seus coeficientes $\mathtt{a}_{\mathcal{Z}} = [a_{\mathcal{Z},0}, \ \ldots, \ a_{\mathcal{Z},n}]^T$ numa base ortogonal $\mathcal{Z}$. Desta forma, evitando a mudança para a base das potências, conseguimos eliminar a propagação de erros referentes à mudança de base (5.1). Os exemplos a seguir ilustram a vantagem da Proposição 5.1.

Exemplo 5.1. Como ilustração, vamos aproximar a solução do problema diferencial

$$\begin{cases} \dfrac{d^2}{dx^2} y + y = 0, \quad x \in]0, \ 10[\\ y(0) = 0, \quad y'(0) = 1 \end{cases},$$

que tem como solução exata a função $y = \sin(x)$.

Solução. Aproximamos a solução deste problema utilizando o método tau com polinómios de Chebyshev de primeira e de segunda espécie e com polinómios de Legendre, ambos truncados na dimensão 100. As soluções tau foram avaliadas com as funções `polyval` e **orthoval** e calculamos a diferença entre estes valores e a solução exata. Os resultados são mostrados na Figura 5.1 e 5.2.

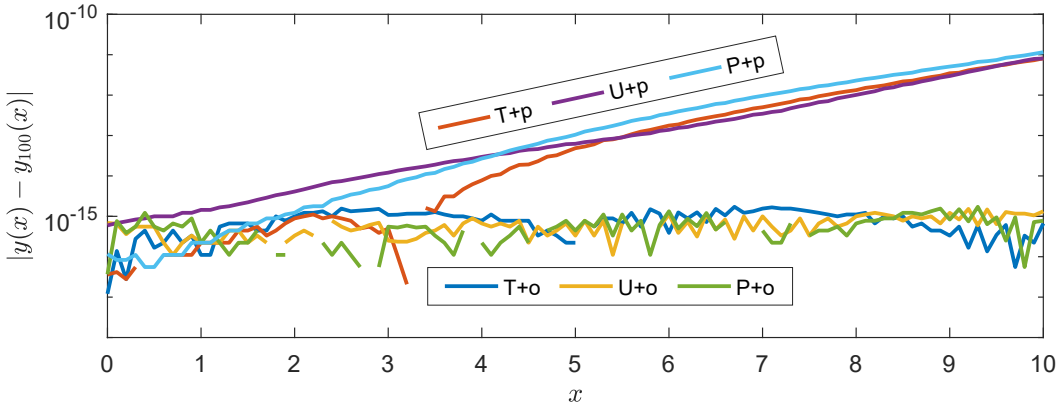

Figura 5.1: Erros $|y(x) - y_{100}(x)|$, $x \in [0, \ 10]$ na solução do problema do Exemplo 5.1, com polinómios de Chebyshev de primeira espécie (T) e de segunda espécie (U) e com polinómios de Legendre (P), utilizando a função `MATLAB polyval` (p) e a função **orthoval** (o).

Experiência semelhante foi realizada, com as mesmas bases e o mesmo grau dos polinómios, com a implementação do método tau na versão parcelar, apresentada na Secção 4.5. Foram utilizados 4 passos na divisão do intervalo $[0, \ 10]$ em 4 parcelas de igual amplitude. Os resultados apresentam-se na Figura 5.2.

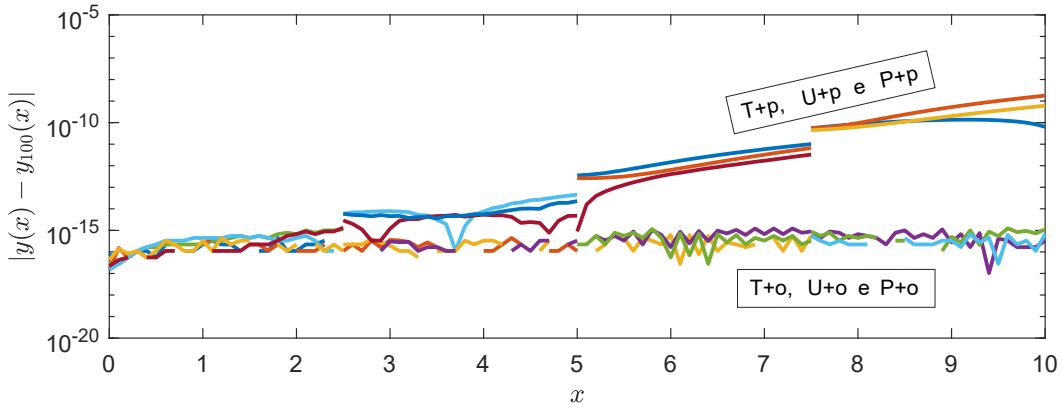

Figura 5.2: Erros $|y(x) - y_{100}(x)|$, $x \in [0,\ 10]$ na solução do problema do Exemplo 5.1, utilizando a versão parcelar do método tau, com polinómios de Chebyshev de primeira espécie (T) e de segunda espécie (U) e com polinómios de Legendre (P), utilizando a função `MATLAB` `polyval` (p) e a função **`orthoval`** (o).

Fica evidente a perda de precisão no emprego da função `polyval`, por requerer a transformação de base. Tanto na versão tau executada no domínio inteiro (Figura 5.1) como na versão parcelar (Figura 5.2) a qualidade da avaliação da imagem da aproximação permaneceu próxima da precisão da máquina para o exemplo em questão, quando aplicado **`orthoval`**. Esta proposição é inspirada no algoritmo de Clenshaw (1955), que propôs avaliar a imagem de funções aproximadas por determinadas séries truncadas através de relações de recorrência. Uma vez que conhecemos as relações de recorrência para os polinómios ortogonais, estas fórmulas foram implementadas de forma a avaliar a imagem dos polinómios e suas derivadas em escalares, em vetores e em matrizes.

Foi implementado um conjunto de rotinas baseadas nestas fórmulas e que permitem calcular $Z_i(\mathsf{x})$, para cada elemento Z_i de uma base de polinómios ortogonais, onde x é um vetor. Além disso, estas rotinas permitem calcular, de forma estável, $Z_i(\mathsf{M})$, onde M é uma matriz. Esta funcionalidade é relevante para o cálculo de elevada precisão dos elementos das matrizes T, de representação dos operadores integro-diferenciais e assume particular importância na aplicação a problemas não lineares. Retomando o Exemplo 4.4 do Capítulo anterior, analisando o problema linearizado, verificamos a existência de coeficientes polinomiais, $y^{(k-1)}$ e $\dfrac{d}{dx}y^{(k-1)}$, cuja representação na matriz T consiste das matrizes $y^{(k-1)}(\mathsf{M})$ e $\dfrac{d}{dx}y^{(k-1))}(\mathsf{M})$. Uma vez que os polinómios $y^{(k-1)}$ estão representados numa base de polinómios ortogonais, utilizamos a rotina **`orthovalM`**, com a versão matricial da Proposição 5.1, para o cálculo de elevada precisão dos elementos destas matrizes. No caso do exemplo 4.3, a existência do coeficiente não polinomial $\sin(y^{(k-1)})$, traduzindo-se por $\sin(\mathsf{M})$ e exigindo uma aproximação polinomial prévia, como veremos no Capítulo 7, torna o processo de estabilização do cálculo ainda mais relevante.

5.2 Cálculo recursivo das matrizes de mudanças de base

As bases de polinómios ortogonais que utilizaremos na presente Tese serão manipuladas em grande parte das vezes no formato matricial, uma vez que estamos focados também na implementação computacional eficiente dos métodos. Para formalizarmos a sugerida notação, consideramos:

Proposição 5.2. *Seja* $\mathcal{Z} = [Z_0,\ Z_1,\ \ldots]$ *uma sequência de polinómios ortogonais que satisfazem a relação de recorrência* (2.1)*, e seja* $\mathcal{X} = [1,\ x^1,\ x^2,\ \ldots]$ *a base das potências, então verifica-se a igualdade*

$$\mathcal{X}\mathsf{V}_{\mathcal{Z}} = \mathcal{Z},\ \mathsf{V}_{\mathcal{Z}} = \begin{bmatrix} v_{00} & v_{01} & v_{02} & \ldots \\ & v_{11} & v_{12} & \cdots \\ & & v_{22} & \cdots \\ & & & \ddots \end{bmatrix},$$

com os coeficientes v_{ij} *determinados por*

$$\mathsf{v}_{j+1} = \frac{\hat{\mathsf{v}}_j - \beta_j \mathsf{v}_j - \gamma_j \mathsf{v}_{j-1}}{\alpha_j}, \quad j \geq 0,$$

com $\mathsf{v}_{-1} = [0,\ 0,\ \ldots]^T$, $\mathsf{v}_1 = [1,\ 0,\ 0,\ \ldots]^T$, $\mathsf{v}_j = \mathsf{V}_{\mathcal{Z}}\mathsf{e}_j$ *e* $\hat{\mathsf{v}}_j = [0;\ \mathsf{v}_j]$.

Demonstração. Se $\mathcal{X}\mathsf{V}_{\mathcal{Z}} = \mathcal{Z}$, então temos que

$$\mathcal{X}\mathsf{V}_{\mathcal{Z}}\mathsf{e}_{j+1} = \mathcal{Z}\mathsf{e}_{j+1} = Z_{j+1} \tag{5.4}$$

e que

$$xZ_j = \sum_{i=0}^{j} v_{ij} x^{i+1} = \sum_{i=0}^{j+1} v_{i-1,j} x^i = \mathcal{X}[0;\ \mathsf{V}_{\mathcal{Z}}\mathsf{e}_j]. \tag{5.5}$$

Da relação (2.1) obtemos

$$\sum_{i=0}^{j+1} v_{i,j+1} x^i = Z_{j+1} = \frac{(x - \beta_j Z_j) - \gamma_j Z_{j-1}}{\alpha_j}. \tag{5.6}$$

e finalmente substituindo (5.4) e (5.5) em (5.6) encontramos

$$\mathcal{Z}\mathsf{e}_{j+1} = \frac{\mathcal{X}\left([0;\ \mathsf{V}_{\mathcal{Z}}\mathsf{e}_j] - \beta_j \mathsf{V}_{\mathcal{Z}}\mathsf{e}_j - \gamma_j \mathsf{V}_{\mathcal{Z}}\mathsf{e}_{j-1}\right)}{\alpha_j}.$$

A prova se completa considerando que v_{-1} e v_0 são os vetores que contém, respetivamente, os coeficientes de Z_{-1} e Z_0. □

Esta Proposição dá origem na `Tau Toolbox` às funções que geram as matrizes $\mathsf{V}_{\mathcal{Z}}$ para as famílias de polinómios ortogonais clássicas: **`polchebyshevT`**, **`polchebyshevU`**, **`pollegendreP`**, **`polhermiteH`**, **`pollaguerreL`** e **`polbesselY`**.

Conforme mostrado na equação (5.1), a igualdade $\mathsf{a} = \mathsf{V}_{\mathcal{Z}}\mathsf{a}_{\mathcal{Z}}$ leva um vetor de coeficientes da base $\mathcal{Z}$ para a base $\mathcal{X}$, e desta forma é evidente que uma das maneiras de se obter a operação contrária, isto é, levar um vetor de coeficientes na base $\mathcal{X}$ para a base $\mathcal{Z}$, se faz da forma

$$\mathsf{a}_{\mathcal{Z}} = \mathsf{V}_{\mathcal{Z}}^{-1}\mathsf{a}.$$

Todavia, a inversão explícita nunca é recomendada em termos de cálculos computacionais, e para a evitar propõe-se uma forma de obter a inversa de $\mathsf{V}_{\mathcal{Z}}$ por recorrência, conforme:

Proposição 5.3. *Seja $\mathsf{V}_{\mathcal{Z}}$ a matriz que traduz a base ortogonal $\mathcal{Z}$ em série de potências. Uma forma de encontrar $\mathsf{W}_{\mathcal{Z}} = \mathsf{V}_{\mathcal{Z}}^{-1}$ é através da recorrência*

$$\begin{cases} \mathsf{w}_{j+1} = \mathsf{M}_{\mathcal{Z}}\mathsf{w}_j, \quad j \geq 1 \\ \mathsf{w}_1 = \mathsf{e}_1 \end{cases}, \tag{5.7}$$

onde $\mathsf{M}_{\mathcal{Z}}$ é a matriz tal que $x\mathcal{Z} = \mathcal{Z}\mathsf{M}_{\mathcal{Z}}$, w_j é a j-ésima coluna de $\mathsf{W}_{\mathcal{Z}}$ e e_1 a primeira coluna da matriz identidade.

Demonstração. Por definição $\mathcal{Z} = \mathcal{X}\mathsf{V}_{\mathcal{Z}}$ e portanto $\mathcal{X} = \mathcal{Z}\mathsf{W}_{\mathcal{Z}}$, e então cada um dos elementos de $\mathcal{X}$ é encontrado na forma

$$x_j = \sum_{i\geq 0} Z_i w_{i+1,j} = \mathcal{Z}\mathsf{w}_j, \quad j \geq 1,$$

e portanto

$$x_{j+1} = x\mathcal{Z}\mathsf{w}_j, \quad j \geq 1.$$

Como $x\mathcal{Z} = \mathcal{Z}\mathsf{M}_{\mathcal{Z}}$ por definição, segue que

$$\mathcal{Z}\mathsf{w}_{j+1} = \mathcal{Z}\mathsf{M}_{\mathcal{Z}}\mathsf{w}_j \Rightarrow \mathsf{w}_{j+1} = \mathsf{M}_{\mathcal{Z}}\mathsf{w}_j, \quad j \geq 1.$$

□

Esta proposição dá origem à função **pow2orthmatrix**, que tem como parâmetro de entrada a matriz $\mathsf{M}_{\mathcal{Z}}$ truncada e como parâmetro de saída a matriz $\mathsf{W}_{\mathcal{Z}}$, truncada na mesma dimensão. São mostrados na Tabela 5.1 os resultados das normas $||\mathsf{V}_{\mathcal{Z}}\mathsf{V}_{\mathcal{Z}}^{-1} - \mathsf{I}||_2$ e $||\mathsf{V}_{\mathcal{Z}}\mathsf{W}_{\mathcal{Z}} - \mathsf{I}||_2$, ou seja, aplicando a inversão convencional `inv` no primeiro caso e a Proposição 5.3 no outro. Para $\mathcal{Z}$ foram usadas as bases de Chebyshev de primeira e segunda espécie e a base de Legendre.

n	$\underbrace{\|\mathsf{V}\mathsf{V}^{-1}-\mathsf{I}\|_2}_{ChebyshevT}$	$\underbrace{\|\mathsf{V}\mathsf{W}-\mathsf{I}\|_2}_{ChebyshevT}$	$\underbrace{\|\mathsf{V}\mathsf{V}^{-1}-\mathsf{I}\|_2}_{ChebyshevU}$	$\underbrace{\|\mathsf{V}\mathsf{W}-\mathsf{I}\|_2}_{ChebyshevU}$	$\underbrace{\|\mathsf{V}\mathsf{V}^{-1}-\mathsf{I}\|_2}_{Legendre}$	$\underbrace{\|\mathsf{V}\mathsf{W}-\mathsf{I}\|_2}_{Legendre}$
10	0	0	0	0	$4,60\cdot10^{-16}$	$8,95\cdot10^{-16}$
20	0	0	0	0	$1,18\cdot10^{-13}$	$3,43\cdot10^{-14}$
30	0	0	0	0	$1,82\cdot10^{-11}$	$3,83\cdot10^{-13}$
40	0	0	0	0	$2,64\cdot10^{-9}$	$1,07\cdot10^{-11}$
50	$1,03\cdot10^{-12}$	0	0	0	$1,55\cdot10^{-7}$	$4,92\cdot10^{-10}$
60	$2,37\cdot10^{-6}$	$2,06\cdot10^{-9}$	$1,01\cdot10^{-6}$	$1,48\cdot10^{-9}$	$4,27\cdot10^{-5}$	$6,39\cdot10^{-9}$
70	$3,08\cdot10^{-3}$	$2,03\cdot10^{-7}$	$1,56\cdot10^{-3}$	$1,05\cdot10^{-7}$	$1,07\cdot10^{-2}$	$2,62\cdot10^{-7}$
80	$5,96$	$3,73\cdot10^{-6}$	$1,56$	$2,58\cdot10^{-6}$	$7,98$	$5,59\cdot10^{-6}$
90	$3,81\cdot10^{2}$	$1,53\cdot10^{-4}$	$9,15\cdot10^{2}$	$8,34\cdot10^{-5}$	$1,81\cdot10^{3}$	$3,35\cdot10^{-4}$
100	$1,95\cdot10^{5}$	$5,10\cdot10^{-3}$	$3,25\cdot10^{5}$	$4,01\cdot10^{-3}$	$1,36\cdot10^{5}$	$1,04\cdot10^{-2}$

Tabela 5.1: Normas $||\mathsf{V}_{\mathcal{Z}}\mathsf{V}_{\mathcal{Z}}^{-1}-\mathsf{I}||_2$ e $||\mathsf{V}_{\mathcal{Z}}\mathsf{W}_{\mathcal{Z}}-\mathsf{I}||_2$ (`inv` vs **`pow2orthmatrix`**) para as bases de Chebyshev de primeira e segunda espécie e de Legendre, usando a inversão convencional (V^{-1}) e por recorrência (W).

Podemos perceber, da Tabela 5.1, que obter $\mathsf{W}_{\mathcal{Z}}$ pela recorrência (5.7) é um procedimento mais estável do que obter diretamente $\mathsf{V}_{\mathcal{Z}}^{-1}$. A norma $||\cdot||_\infty$ também produz resultados semelhantes. O impacto desta estabilidade pode ser visto em um caso concreto de aproximação utilizando o método tau, conforme o Exemplo a seguir.

Exemplo 5.2. Seja o problema diferencial

$$\begin{cases} \dfrac{d^3}{dx^3}y + x^{70}y = x^{76} - 2x^{73} + 10x^{71} + 120x^3 - 12 \\ y(0) = 0,\ y'(-1) = -2,\ y''(1) = 18 \end{cases},$$

cuja solução exata é o polinómio $x^6 - 2x^3 + 10x$.

Este problema foi escolhido por conter um polinómio de ordem relativamente elevada no segundo membro, o qual será transformado para a base ortogonal com o uso da matriz de mudança de base. A Figura 5.3 apresenta o erro para a aproximação (com $n = 80$) da solução do Exemplo 5.2, quando usada a inversão formal (V^{-1}) e quando utilizada a recorrência (W).

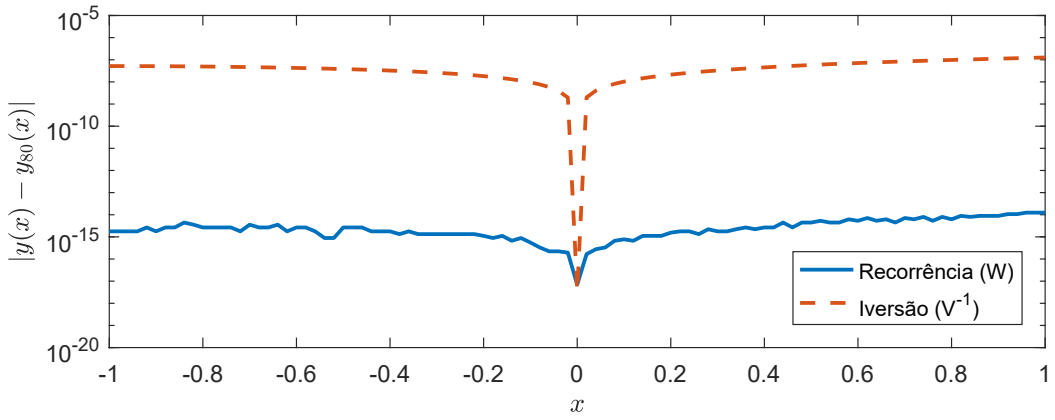

Figura 5.3: Erro para a aproximação (com $n = 80$ e base de Legendre) da solução do Exemplo 5.2.

Analisando a Figura 5.3 percebemos que, para o problema dado, a recorrência permitiu manter uma precisão de 10^{-15} para o erro, enquanto a inversão convencional originou o erro próximo a 10^{-7}, para o mesmo $n = 80$. Desta forma, mostramos com a aproximação da solução de um problema diferencial a vantagem da Proposição 5.2.

5.3 Transformações de semelhança

As matrizes $\mathsf{M}_{\mathcal{Z}} = [\mu_{ij}]_{\infty\times\infty}$, $\mathsf{N}_{\mathcal{Z}} = [\eta_{ij}]_{\infty\times\infty}$ e $\mathsf{O}_{\mathcal{Z}} = [\theta_{ij}]_{\infty\times\infty}$ introduzidas na Secção 4.2, referem-se à ação, respetivamente, do operador que incrementa grau, do operador de derivação e do operador de integração, sobre os coeficientes a_i de uma determinada série

$$y = \sum_{i=0}^{\infty} a_i Z_i = \mathcal{Z}\mathsf{a}_{\mathcal{Z}}, \quad \mathsf{a}_{\mathcal{Z}} = [a_0,\ a_1,\ \ldots]^T,$$

e portanto são tais que

$$\mathsf{M}_{\mathcal{Z}}^k \mathsf{a}_{\mathcal{Z}} = x^k y_n, \quad \mathsf{N}_{\mathcal{Z}}^k \mathsf{a}_{\mathcal{Z}} = \frac{d^k}{dx^k} y_n \quad \text{e} \quad \mathsf{O}_{\mathcal{Z}}^k \mathsf{a}_{\mathcal{Z}} = \int^x \ldots \int^x y_n dx^k.$$

Estas matrizes são usualmente obtidas através da transformação de semelhança de $\mathcal{X}$ para $\mathcal{Z}$, conforme o Teorema 4.3, implicando portanto na necessidade da inversão de $\mathsf{V}_{\mathcal{Z}}$. Outra forma seria obter individualmente cada elemento destas matrizes, utilizando as fórmulas dos coeficientes de Fourier dadas pela Definição 2.2. No caso das matrizes $\mathsf{M}_{\mathcal{Z}}$, $\mathsf{N}_{\mathcal{Z}}$ e $\mathsf{O}_{\mathcal{Z}}$

$$\mu_{i,j} = \frac{\langle xZ_i, Z_j\rangle_w}{\sqrt{\langle Z_i, Z_i\rangle_w}}, \quad \eta_{i,j} = \frac{\left\langle \frac{d}{dx}Z_i, Z_j\right\rangle_w}{\sqrt{\langle Z_i, Z_i\rangle_w}}, \quad \text{e} \quad \theta_{i,j} = \frac{\left\langle \int^x Z_i dx, Z_j\right\rangle_w}{\sqrt{\langle Z_i, Z_i\rangle_w}},$$

todavia estas não são as formas computacionalmente mais fáceis e estáveis de se obter tais resultados. Com o propósito de evitar as transformações de semelhança e

consequentemente a inversão de $\mathsf{V}_{\mathcal{Z}}$, utilizamos a formulação para obter $\mathsf{M}_{\mathcal{Z}}$, $\mathsf{N}_{\mathcal{Z}}$ e $\mathsf{O}_{\mathcal{Z}}$ diretamente na base ortogonal $\mathcal{Z}$ proposta por Matos et al. (2017).

Proposição 5.4. *Seja $\mathcal{Z} = [Z_0,\ Z_1,\ \ldots]$ uma base de polinómios ortogonais, satisfazendo a relação de recorrência a três termos (2.1), então $\mathsf{M}_{\mathcal{Z}}$ é a matriz tridiagonal*

$$\mathsf{M}_{\mathcal{Z}} = \begin{bmatrix} \mu_{00} & \mu_{01} & & & \\ \mu_{10} & \mu_{11} & \mu_{12} & & \\ & \mu_{21} & \mu_{22} & \mu_{23} & \\ & & \ddots & \ddots & \ddots \end{bmatrix} \quad com \begin{cases} \mu_{j,j} = \beta_j \\ \mu_{j+1,j} = \alpha_j \\ \mu_{j-1,j} = \gamma_j \end{cases} . \tag{5.8}$$

Demonstração. Por definição da matriz M temos que $x\mathcal{Z} = \mathcal{Z}\mathsf{M}_{\mathcal{Z}}$, logo

$$x\mathcal{Z}\mathsf{e}_{j+1} = \mathcal{Z}\mathsf{M}_{\mathcal{Z}}\mathsf{e}_{j+1}, \quad j \geq 0$$

de que resulta

$$\begin{aligned} \mathcal{Z}\mathsf{M}_{\mathcal{Z}}\mathsf{e}_{j+1} &= xZ_j \\ &= \alpha_j Z_{j+1} + \beta_j Z_j + \gamma_j Z_{j-1} \\ &= \mathcal{Z}(\alpha_j \mathsf{e}_{j+2} + \beta_j \mathsf{e}_{j+1} + \gamma_j \mathsf{e}_j), \quad j \geq 0. \end{aligned}$$

A demonstração termina observando que

$$\mu_{ik} = \mathsf{e}_{i+1}^T \mathsf{M}_{\mathcal{Z}} \mathsf{e}_{j+1} = \begin{cases} \alpha_j, \text{ se } i = j+1 \\ \beta_j, \text{ se } i = j \\ \gamma_j, \text{ se } i = j-1 \end{cases} ,$$

e $\mu_{ij} = 0$ se $|j - i| > 1$ □

Proposição 5.5. *Seja $\mathcal{Z} = [Z_0,\ Z_1,\ \ldots]$ uma base de polinómios ortogonais, satisfazendo a relação de recorrência a três termos (2.1), então a matriz $\mathsf{N}_{\mathcal{Z}}$ pode ser escrita como*

$$\mathsf{N}_{\mathcal{Z}} = \begin{bmatrix} 0 & \eta_{01} & \eta_{02} & \eta_{03} & \ldots \\ & 0 & \eta_{12} & \eta_{13} & \ldots \\ & & 0 & \eta_{23} & \ldots \\ & & & 0 & \ldots \\ & & & & \ddots \end{bmatrix}, \tag{5.9}$$

onde, para $j \geq 1$, os valores η_{ij} podem calcular-se por

$$\begin{cases} \eta_{i,j+1} = \dfrac{\alpha_{i-1}\eta_{i-1,j} + (\beta_i - \beta_j)\eta_{i,j} + \gamma_{i+1}\eta_{i+1,j} - \gamma_j \eta_{i,j-1}}{\alpha_j}, \quad i = 0(1)j-1 \\ \eta_{j,j+1} = \dfrac{\alpha_{j-1}\eta_{j-1,j} + 1}{\alpha_j} \end{cases} . \tag{5.10}$$

Demonstração. A partir da definição de $\mathsf{N}_{\mathcal{Z}}$ tem-se que

$$\frac{d}{dx} Z_j = \sum_{i=0}^{j-1} \eta_{i,j} Z_i, \quad j \geq 0, \tag{5.11}$$

e também que

$$x\sum_{i=0}^{j-1}\eta_{i,j}Z_i=\sum_{i=0}^{j-1}\eta_{i,j}xZ_i. \tag{5.12}$$

Derivando ambos os lados da relação de recorrência (2.1) vem

$$\begin{cases} x\dfrac{d}{dx}Z_j+Z_j=\alpha_j\dfrac{d}{dx}Z_{j+1}+\beta_j\dfrac{d}{dx}Z_j+\gamma_j\dfrac{d}{dx}Z_{j-1}, \quad j\geq 0 \\ \dfrac{d}{dx}Z_{-1}=0, \quad \dfrac{d}{dx}Z_0=0 \end{cases}, \tag{5.13}$$

substituindo (5.11) em (5.13) e organizando os índices para conservar Z_i obtém-se

$$\alpha_j\sum_{i=0}^{j}\eta_{i,j+1}Z_i=Z_j+x\sum_{i=0}^{j-1}\eta_{i,j}Z_i-\beta_j\sum_{i=0}^{j-1}\eta_{i,j}Z_i-\gamma_j\sum_{i=0}^{j-2}\eta_{i,j-1}Z_i; \tag{5.14}$$

substituindo (2.1) em (5.12) obtém-se

$$x\sum_{i=0}^{j-1}\eta_{i,j}Z_i=\sum_{i=0}^{j-1}\eta_{i,j}\Big(\alpha_iZ_{i+1}+\beta_iZ_i+\gamma_iZ_{i-1}\Big),$$

e portanto

$$x\sum_{i=0}^{j-1}\eta_{i,j}Z_i=\sum_{i=1}^{j}\alpha_{i-1}\eta_{i-1,j}Z_i+\sum_{i=0}^{j-1}\beta_i\eta_{i,j}Z_i+\sum_{i=0}^{j-2}\gamma_{i+1}\eta_{i+1,j}Z_i. \tag{5.15}$$

Finalmente, ao substituir (5.15) em (5.14), obtém-se

$$\alpha_j\eta_{i,j+1}=\alpha_{i-1}\eta_{i-1,j}+(\beta_i-\beta_j)\eta_{i,j}+\gamma_{i+1}\eta_{i+1,j}-\gamma_j\eta_{i,j-1}, \quad i=0(1)j-1,$$

considerando $\eta_{i,j}=0$ sempre que $i\geq j$ ou $i<0$; e assim $\mathsf{N}_{\mathcal{Z}}$ em (5.10) é obtida. □

Quanto à matriz de integração $\mathsf{O}_{\mathcal{Z}}$, pode ser obtida à custa dos elementos η_{ij} da matriz $\mathsf{N}_{\mathcal{Z}}$.

Proposição 5.6. *Seja $\mathcal{Z}=[Z_0,\ Z_1,\ \ldots]$ uma sequência de polinómios ortogonais satisfazendo a relação de recorrência a três termos (2.1) e $\mathsf{N}_{\mathcal{Z}}$ a matriz tal que $\mathcal{Z}\mathsf{N}_{\mathcal{Z}}=\frac{d}{dx}\mathcal{Z}$, então a matriz $\mathsf{O}_{\mathcal{Z}}=[\theta_{i,j}]$ tem a forma*

$$\mathsf{O}_{\mathcal{Z}}=\begin{bmatrix} 0 & 0 & 0 & 0 & \ldots \\ \theta_{10} & \theta_{11} & \theta_{12} & \theta_{13} & \ldots \\ & \theta_{21} & \theta_{22} & \theta_{23} & \ldots \\ & & \theta_{32} & \theta_{33} & \ldots \\ & & & & \ddots \end{bmatrix},$$

onde, para cada $j\geq 0$

$$\begin{cases} \theta_{j+1,j}=\dfrac{\alpha_j}{j+1} \\ \theta_{i+1,j}=-\dfrac{\alpha_i}{i+1}\displaystyle\sum_{k=i+2}^{j+1}\eta_{i,k}\theta_{k,j}, \quad i=j-1(-1)0 \end{cases}$$

Demonstração. Uma vez que a primitiva $\int^x Z_i dx$ é um polinómio de grau $i+1$ definida com uma constante arbitrária, temos que

$$\int^x Z_j dx = \sum_{k=1}^{j+1} \theta_{k,j} Z_k, \quad j \geq 0. \tag{5.16}$$

Derivando ambos os membros de (5.16) encontramos

$$Z_j = \sum_{k=1}^{j+1} \theta_{k,j} \frac{d}{dx} Z_k, \tag{5.17}$$

e substituindo (5.11) em (5.17) escrevemos

$$\begin{aligned} Z_j &= \sum_{k=1}^{j+1} \theta_{k,j} \sum_{i=0}^{k-1} \eta_{i,k} Z_i \\ &= \sum_{i=0}^{j} \left(\sum_{k=i+1}^{j+1} \eta_{i,k} \theta_{k,j} \right) Z_i, \end{aligned}$$

e finalmente identificando os termos semelhantes temos

$$\begin{cases} \eta_{j,j+1} \theta_{j+1,j} = 1, & i = j \\ \sum_{k=i+1}^{j+1} \eta_{i,k} \theta_{k,j} = 0, & i = 0(1)j-1 \end{cases}. \tag{5.18}$$

Uma vez que $\eta_{j,j+1} = \dfrac{j+1}{\alpha_j}$, $j \geq 0$ (ver Matos et al. (2017)), de (5.18) temos

$$\theta_{j+1,j} = \frac{\alpha_j}{j+1}$$

e

$$\theta_{i+1,j} = -\frac{\alpha_i}{i+1} \sum_{k=i+2}^{j+1} \eta_{i,k} \theta_{k,j}, \quad i = j-1(-1)0,$$

para $j \geq 0$. □

Aplicação a alguns polinómios ortogonais clássicos

Embora diretamente com (5.10) e (5.8) seja possível obter os elementos destas matrizes por recorrência, nestes casos é possível obter fórmulas explícitas para os elementos das matrizes (Matos et al., 2017). Apresentamos a seguir as fórmulas explícitas para a obtenção das matrizes $\mathsf{N}_{\mathcal{Z}}$ e $\mathsf{O}_{\mathcal{Z}}$ para as bases de Chebyshev ($\mathcal{T}$), Legendre ($\mathcal{P}$), Laguerre ($\mathcal{L}$), Hermite ($\mathcal{H}$) e Bessel ($\mathcal{Y}$).

$\mathcal{Z}$	η_{ij}	i,j
$\mathcal{T}$	$\eta_{ij} = (\frac{1}{2})^{\delta_{i1}}(1-(-1)^{i+j})(j-1)$	$j>i,\ i\geq 1$
$\mathcal{U}$	$\eta_{ij} = (1-(-1)^{i+j})i$	$j>i,\ i\geq 1$
$\mathcal{P}$	$\eta_{ij} = \frac{1}{2}(1-(-1)^{i+j})(2i-1)$	$j>i,\ i\geq 1$
$\mathcal{L}$	$\eta_{ij} = -1$	$j>i,\ i\geq 1$
$\mathcal{H}$	$\eta_{ij} = 2i$	$j=i+1,\ i\geq 1$
$\mathcal{Y}$	$\eta_{ij} = \frac{2i-1}{2}(-1)^{i+j}(i(i-1)-j(j-1))$	$j>i,\ i\geq 1$

Tabela 5.2: Fórmulas explícitas para a obtenção da matriz $\mathsf{N}_{\mathcal{Z}}$ para as bases de Chebyshev ($\mathcal{T}$ e $\mathcal{U}$), Legendre ($\mathcal{P}$), Laguerre ($\mathcal{L}$), Hermite ($\mathcal{H}$) e Bessel ($\mathcal{Y}$).

$\mathcal{Z}$	θ_{ij}	i,j
$\mathcal{T}$	$\theta_{21}=1,\ \theta_{23}=-\frac{1}{2},\ \theta_{j,j-1}=-\theta_{j,j+1}=\frac{1}{2(j-1)}$	$j\geq 3$
$\mathcal{U}$	$\theta_{21}=\frac{1}{2},\ \theta_{32}=\frac{1}{4},\ \theta_{j+1,j}=-\theta_{j-1,j}=\frac{1}{2j}$	$j\geq 3$
$\mathcal{P}$	$\theta_{j,j-1}=\frac{1}{2j-3},\ \theta_{i,j+1}=\frac{-1}{2j+1}$	$j\geq 2$
$\mathcal{L}$	$\theta_{jj}=-\theta_{j+1,j}=1$	$j\geq 1$
$\mathcal{H}$	$\theta_{j+1,j}=\frac{1}{2j}$	$j\geq 1$
$\mathcal{Y}$	$\theta_{jj}=\frac{1}{j(j-1)},\ \theta_{j-1,j}=\frac{j}{2j-1},\ \theta_{j+1,j}=\frac{j-1}{2j-1}$	$j\geq 3$

Tabela 5.3: Fórmulas explícitas para a obtenção da matriz $\mathsf{O}_{\mathcal{Z}}$ para as bases de Chebyshev ($\mathcal{T}$ e $\mathcal{U}$), Legendre ($\mathcal{P}$), Laguerre ($\mathcal{L}$), Hermite ($\mathcal{H}$) e Bessel ($\mathcal{Y}$).

Com os parâmetros da relação de recorrência a três termos e as fórmulas apresentadas nas Tabelas 5.2 e 5.3, obtemos as seguintes matrizes:

$$\underbrace{\begin{bmatrix} 0 & \frac{1}{2} & & & & \\ 1 & 0 & \frac{1}{2} & & & \\ & \frac{1}{2} & 0 & \frac{1}{2} & & \\ & & \frac{1}{2} & 0 & \frac{1}{2} & \\ & & & \frac{1}{2} & 0 & \ddots \\ & & & & \ddots & \ddots \end{bmatrix}}_{\mathsf{M}_{\mathcal{T}}},\quad \underbrace{\begin{bmatrix} 0 & 1 & 0 & 3 & 0 & 5 & \dots \\ & 0 & 4 & 0 & 8 & 0 & \dots \\ & & 0 & 6 & 0 & 10 & \dots \\ & & & 0 & 8 & 0 & \dots \\ & & & & 0 & 10 & \dots \\ & & & & & 0 & \dots \\ & & & & & & \ddots \end{bmatrix}}_{\mathsf{N}_{\mathcal{T}}},\quad \underbrace{\begin{bmatrix} 0 & 0 & & & & \\ 1 & 0 & -\frac{1}{2} & & & \\ & \frac{1}{4} & 0 & -\frac{1}{4} & & \\ & & \frac{1}{6} & 0 & -\frac{1}{6} & \\ & & & \frac{1}{8} & 0 & \ddots \\ & & & & \ddots & \ddots \end{bmatrix}}_{\mathsf{O}_{\mathcal{T}}},$$

$$\underbrace{\begin{bmatrix} 0 & \frac{1}{2} & & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & & \\ & \frac{1}{2} & 0 & \frac{1}{2} & & \\ & & \frac{1}{2} & 0 & \frac{1}{2} & \\ & & & \frac{1}{2} & 0 & \ddots \\ & & & & \ddots & \ddots \end{bmatrix}}_{\mathsf{M}_{\mathcal{U}}},\quad \underbrace{\begin{bmatrix} 0 & 2 & 0 & 2 & 0 & 2 & \dots \\ & 0 & 4 & 0 & 4 & 0 & \dots \\ & & 0 & 6 & 0 & 6 & \dots \\ & & & 0 & 8 & 0 & \dots \\ & & & & 0 & 10 & \dots \\ & & & & & 0 & \dots \\ & & & & & & \ddots \end{bmatrix}}_{\mathsf{N}_{\mathcal{U}}},\quad \underbrace{\begin{bmatrix} 0 & 0 & & & & \\ \frac{1}{2} & 0 & -\frac{1}{6} & & & \\ & \frac{1}{4} & 0 & -\frac{1}{8} & & \\ & & \frac{1}{6} & 0 & -\frac{1}{10} & \\ & & & \frac{1}{8} & 0 & \ddots \\ & & & & \ddots & \ddots \end{bmatrix}}_{\mathsf{O}_{\mathcal{U}}},$$

$$\underbrace{\begin{bmatrix} 0 & \frac{1}{3} & & & & \\ 1 & 0 & \frac{2}{5} & & & \\ & \frac{2}{3} & 0 & \frac{3}{7} & & \\ & & \frac{3}{5} & 0 & \frac{4}{9} & \\ & & & \frac{4}{7} & 0 & \ddots \\ & & & & \ddots & \ddots \end{bmatrix}}_{\mathsf{M}_{\mathcal{P}}}, \underbrace{\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & \dots \\ & 0 & 3 & 0 & 3 & 0 & \dots \\ & & 0 & 5 & 0 & 5 & \dots \\ & & & 0 & 7 & 0 & \dots \\ & & & & 0 & 9 & \dots \\ & & & & & 0 & \dots \\ & & & & & & \ddots \end{bmatrix}}_{\mathsf{N}_{\mathcal{P}}}, \underbrace{\begin{bmatrix} 0 & 0 & & & & \\ 1 & 0 & -\frac{1}{5} & & & \\ & \frac{1}{3} & 0 & -\frac{1}{7} & & \\ & & \frac{1}{5} & 0 & -\frac{1}{9} & \\ & & & \frac{1}{7} & 0 & \ddots \\ & & & & \ddots & \ddots \end{bmatrix}}_{\mathsf{O}_{\mathcal{P}}},$$

$$\underbrace{\begin{bmatrix} 1 & -1 & & & \\ -1 & 3 & -2 & & \\ & -2 & 5 & -3 & \\ & & -3 & 7 & \ddots \\ & & & \ddots & \ddots \end{bmatrix}}_{\mathsf{M}_{\mathcal{L}}}, \underbrace{\begin{bmatrix} 0 & -1 & -1 & -1 & \dots \\ & 0 & -1 & -1 & \dots \\ & & 0 & -1 & \dots \\ & & & 0 & \dots \\ & & & & \ddots \end{bmatrix}}_{\mathsf{N}_{\mathcal{L}}}, \underbrace{\begin{bmatrix} 0 & & & & \\ -1 & 1 & & & \\ & -1 & 1 & & \\ & & -1 & 1 & \\ & & & \ddots & \ddots \end{bmatrix}}_{\mathsf{O}_{\mathcal{L}}},$$

$$\underbrace{\begin{bmatrix} 0 & 1 & & & \\ \frac{1}{2} & 0 & 2 & & \\ & \frac{1}{2} & 0 & 3 & \\ & & \frac{1}{2} & 0 & \ddots \\ & & & \ddots & \ddots \end{bmatrix}}_{\mathsf{M}_{\mathcal{H}}}, \underbrace{\begin{bmatrix} 0 & 2 & & & \\ & 0 & 4 & & \\ & & 0 & 6 & \\ & & & 0 & \ddots \\ & & & & \ddots \end{bmatrix}}_{\mathsf{N}_{\mathcal{H}}}, \underbrace{\begin{bmatrix} 0 & & & & \\ \frac{1}{2} & 0 & & & \\ & \frac{1}{4} & 0 & & \\ & & \frac{1}{6} & 0 & \\ & & & \ddots & \ddots \end{bmatrix}}_{\mathsf{O}_{\mathcal{H}}},$$

$$\underbrace{\begin{bmatrix} 0 & -\frac{1}{3} & & & \\ 1 & 0 & -\frac{1}{5} & & \\ & \frac{1}{3} & 0 & -\frac{1}{7} & \\ & & \frac{1}{5} & 0 & \ddots \\ & & & \ddots & \ddots \end{bmatrix}}_{\mathsf{M}_{\mathcal{Y}}}, \underbrace{\begin{bmatrix} 0 & 1 & -3 & 6 & \dots \\ & 0 & 6 & -15 & \dots \\ & & 0 & -15 & \dots \\ & & & 0 & \dots \\ & & & & \ddots \end{bmatrix}}_{\mathsf{N}_{\mathcal{Y}}}, \underbrace{\begin{bmatrix} 0 & 0 & & & \\ \frac{1}{3} & \frac{1}{6} & \frac{4}{7} & & \\ & \frac{2}{5} & \frac{1}{12} & \frac{5}{9} & \\ & & \frac{3}{7} & \frac{1}{20} & \ddots \\ & & & \ddots & \ddots \end{bmatrix}}_{\mathsf{O}_{\mathcal{Y}}}.$$

Para a `Tau Toolbox`, as Proposições 5.4, 5.5 e 5.6 dão origem, respetivamente, às funções **`matrixM`**, **`matrixN`** e **`matrixO`**. E ainda, uma vez que as potências das matrizes $\mathsf{M}_{\mathcal{Z}}$, $\mathsf{N}_{\mathcal{Z}}$ e $\mathsf{O}_{\mathcal{Z}}$ traduzem operadores de ordem superior, respetivamente, ao incremento de grau em polinómios, derivação, e integração, conforme introduzido na Subsecção 4.2.1, no sentido da `Tau Toolbox` estas potências foram implementadas, respetivamente, como **`mpower`**, **`diff`** e **`int`**. A seguir comparamos o efeito dessas implementações.

Testes numéricos

Para comparar o efeito da utilização destas fórmulas com o efeito das transformações de semelhança habitualmente referidas na literatura, mostramos na Figura 5.4 os padrões de esparsidade das matrizes na base de Chebyshev de primeira espécie de dimensão 80, obtidas aplicando as transformações de semelhança e aplicando as fórmulas explícitas. Percebemos visualmente que o ruído resultante da primeira formulação é inexistente na segunda. Resultados similares acontecem para as demais bases apresentadas.

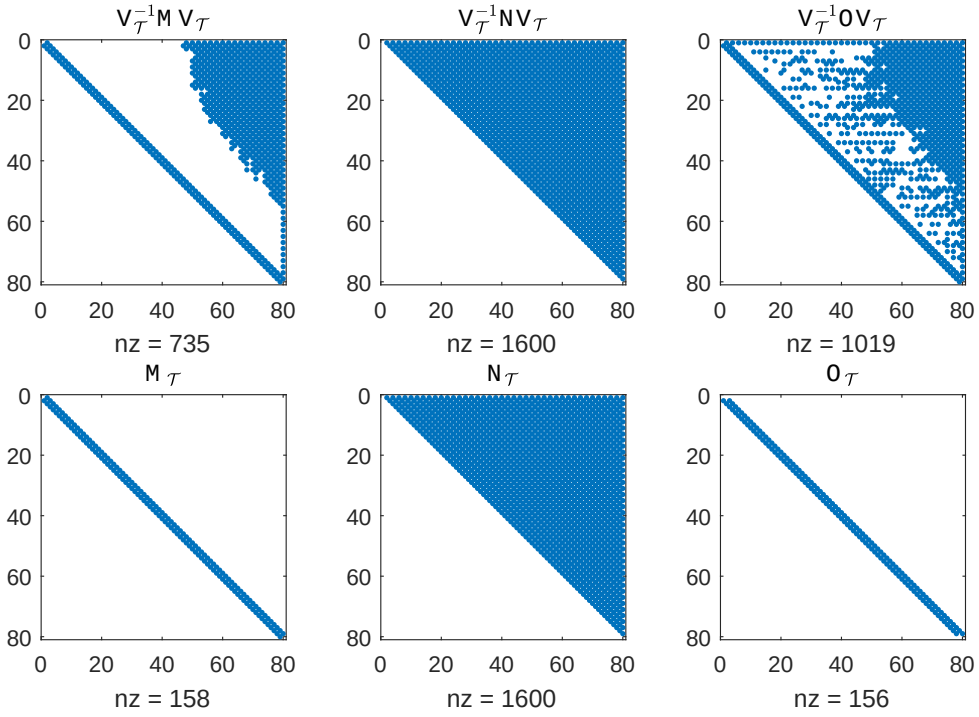

Figura 5.4: Comparação entre as matrizes $\mathsf{M}_{\mathcal{T}}$, $\mathsf{N}_{\mathcal{T}}$ e $\mathsf{O}_{\mathcal{T}}$, quando obtidas por transformações de semelhança e com as fórmulas explícitas.

Na Figura 5.4, percebemos visualmente o resíduo gerado nas matrizes $\mathsf{M}_{\mathcal{T}}$ e $\mathsf{O}_{\mathcal{T}}$ (também presente em $\mathsf{N}_{\mathcal{T}}$) quando obtidas por transformações de semelhança, que por sua vez é completamente evitado ao aplicar as fórmulas explícitas. Esta nova abordagem mostra que $\mathsf{M}_{\mathcal{Z}}$ e $\mathsf{O}_{\mathcal{Z}}$ são matrizes tridiagonais, e portanto tiramos partido em termos computacionais com suas implementações em formato esparso, visado não ocupar memória e ganhar eficiência, principalmente em problemas de grandes dimensões.

O impacto das Proposições 5.4, 5.5 e 5.6 sobre uma aproximação tau na resolução de problemas integro-diferenciais é testado com o Exemplo seguinte.

Exemplo 5.3. Consideremos o exemplo de resolver o problema integro-diferencial

$$\begin{cases} \exp(x)\dfrac{d^2}{dx^2}y + \cos(x)\dfrac{d}{dx}y + \sin(x)y + \\ \qquad\qquad + \displaystyle\int_{-1}^{1} \exp\Big((x+1)t\Big)y(t)dt = f, \quad x \in]-1,\ 1[\quad , \\ y(1) + y(-1) = \mathrm{e} + \dfrac{1}{\mathrm{e}}, \quad y(1) + y(-1) - y'(-1) = \mathrm{e} \end{cases}$$

com

$$f = \Big(\cos(x) + \sin(x) + \exp(x)\Big)\exp(x) + \frac{2\sinh(x+2)}{x+2}$$

e cuja solução exata é $y = \exp(x)$.

Usamos para este problema as bases de Chebyshev de primeira e segunda espécie e de Legendre, com graus $n = 2^i$ e $n = 2^i + 1$, $i = 2(1)8$, aplicado a formulação usual de transformações de semelhança para a obtenção das matrizes $\mathsf{M}_{\mathcal{Z}}$, $\mathsf{N}_{\mathcal{Z}}$ e $\mathsf{O}_{\mathcal{Z}}$. Na Tabela 5.4 podemos observar que a qualidade da solução fica comprometida já no grau 64, sendo que a partir de determinados graus a aproximação deixa de fazer sentido. Este facto aparece associado ao crescimento do número de condição das matrizes, com o aumento da respetiva dimensão. Enquanto que usando as fórmulas explicitas propostas, a precisão da máquina é mantida para o erro

$$\varepsilon_{\mathcal{Z}} = \max |y_n - y|$$

e o número de condição daquelas matrizes mantém-se consideravelmente menor do que no caso anterior (Ver Tabela 5.5).

n	$\varepsilon_{\mathcal{T}}$	$\varepsilon_{\mathcal{U}}$	$\varepsilon_{\mathcal{P}}$	cond($\mathsf{T}_{\mathcal{T}}$)	cond($\mathsf{T}_{\mathcal{U}}$)	cond($\mathsf{T}_{\mathcal{P}}$)
4	$0,12$	$0,55$	$0,06$	$4,85 \cdot 10^{1}$	$4,15 \cdot 10^{1}$	$2,87 \cdot 10^{1}$
5	$0,01$	$0,08$	$0,01$	$1,26 \cdot 10^{2}$	$1,03 \cdot 10^{2}$	$7,43 \cdot 10^{1}$
8	$2,68 \cdot 10^{-5}$	$0,01$	$1,02 \cdot 10^{-5}$	$1,11 \cdot 10^{3}$	$6,44 \cdot 10^{2}$	$5,62 \cdot 10^{2}$
9	$2,19 \cdot 10^{-6}$	$2,49 \cdot 10^{-5}$	$1,05 \cdot 10^{-6}$	$1,85 \cdot 10^{3}$	$1,03 \cdot 10^{3}$	$9,18 \cdot 10^{2}$
64	$2,44 \cdot 10^{-13}$	$3,64 \cdot 10^{-14}$	$4,16 \cdot 10^{-11}$	$1,25 \cdot 10^{16}$	$1,34 \cdot 10^{14}$	$1,03 \cdot 10^{14}$
65	$3,03 \cdot 10^{-13}$	$7,36 \cdot 10^{-14}$	$2,82 \cdot 10^{-12}$	$1,11 \cdot 10^{17}$	$3,20 \cdot 10^{15}$	$4,22 \cdot 10^{14}$
128	$2,49 \cdot 10^{-9}$	$1,11 \cdot 10^{-9}$	NaN	$3,14 \cdot 10^{65}$	$4,71 \cdot 10^{63}$	NaN
129	$2,68 \cdot 10^{-11}$	$2,61 \cdot 10^{-11}$	NaN	$1,87 \cdot 10^{66}$	$4,90 \cdot 10^{63}$	NaN
256	NaN	NaN	NaN	NaN	NaN	NaN
257	NaN	NaN	NaN	NaN	NaN	NaN

Tabela 5.4: Erros $\varepsilon_{\mathcal{Z}} = \max |y_n - y|$ e número de condição das matrizes $\mathsf{T}_{\mathcal{Z}}$, $\mathcal{Z} = \mathcal{T}, \mathcal{U}, \mathcal{P}$, para o Exemplo 5.3, quando calculadas com transformações de semelhança.

n	$\varepsilon_{\mathcal{T}}$	$\varepsilon_{\mathcal{U}}$	$\varepsilon_{\mathcal{P}}$	cond($\mathsf{T}_{\mathcal{T}}$)	cond($\mathsf{T}_{\mathcal{U}}$)	cond($\mathsf{T}_{\mathcal{P}}$)
4	$0,12$	$0,55$	$0,06$	$4,85 \cdot 10^{1}$	$4,15 \cdot 10^{1}$	$2,87 \cdot 10^{1}$
5	$0,01$	$0,08$	$0,01$	$1,26 \cdot 10^{2}$	$1,03 \cdot 10^{2}$	$7,43 \cdot 10^{1}$
8	$2,68 \cdot 10^{-5}$	$0,01$	$1,02 \cdot 10^{-5}$	$1,11 \cdot 10^{3}$	$6,44 \cdot 10^{2}$	$5,62 \cdot 10^{2}$
9	$2,19 \cdot 10^{-6}$	$2,49 \cdot 10^{-5}$	$1,05 \cdot 10^{-6}$	$1,85 \cdot 10^{3}$	$1,03 \cdot 10^{3}$	$9,18 \cdot 10^{2}$
64	$2,66 \cdot 10^{-15}$	$2,66 \cdot 10^{-15}$	$2,22 \cdot 10^{-15}$	$6,12 \cdot 10^{6}$	$2,38 \cdot 10^{6}$	$2,49 \cdot 10^{6}$
65	$1,77 \cdot 10^{-15}$	$2,72 \cdot 10^{-15}$	$1,77 \cdot 10^{-15}$	$6,52 \cdot 10^{6}$	$2,53 \cdot 10^{6}$	$2,65 \cdot 10^{6}$
128	$3,77 \cdot 10^{-15}$	$4,44 \cdot 10^{-15}$	$2,88 \cdot 10^{-15}$	$9,98 \cdot 10^{7}$	$3,76 \cdot 10^{7}$	$3,98 \cdot 10^{7}$
129	$1,55 \cdot 10^{-15}$	$3,77 \cdot 10^{-15}$	$5,32 \cdot 10^{-15}$	$1,02 \cdot 10^{8}$	$3,87 \cdot 10^{7}$	$4,11 \cdot 10^{7}$
256	$8,04 \cdot 10^{-15}$	$5,32 \cdot 10^{-15}$	$5,77 \cdot 10^{-15}$	$1,61 \cdot 10^{9}$	$5,97 \cdot 10^{8}$	$6,38 \cdot 10^{8}$
257	$4,88 \cdot 10^{-15}$	$5,77 \cdot 10^{-15}$	$5,32 \cdot 10^{-15}$	$1,63 \cdot 10^{9}$	$6,06 \cdot 10^{8}$	$6,48 \cdot 10^{8}$

Tabela 5.5: Erros $\varepsilon_{\mathcal{Z}} = \max |y_n - y|$ e número de condição das matrizes $\mathsf{T}_{\mathcal{Z}}$, $\mathcal{Z} = \mathcal{T}, \mathcal{U}, \mathcal{P}$, para o Exemplo 5.3, quando calculadas com fórmulas explícitas.

5.4 Cálculo recursivo do produto em bases ortogonais

Como visto nos Teoremas 4.1 e 4.3, o efeito, no vetor dos coeficientes da solução, de um determinado coeficiente p_r, em uma base $\mathcal{Z}$, é traduzido algebricamente pela matriz

$$p_r(\mathsf{M}_{\mathcal{Z}}) = \sum_{k=0}^{n_r} p_{r,k}\mathsf{M}_{\mathcal{Z}}^k, \quad \text{se} \quad p_r = \sum_{k=0}^{n_r} p_{r,k}x^k,$$

e isto obriga a calcular as potências da matriz $\mathsf{M}_{\mathcal{Z}}$. Mas uma vez que estas matrizes têm dimensão infinita, uma potência k: $\mathsf{M}_{\mathcal{Z}}^k = [\mu_{ij}^{(k)}]_{\infty\times\infty}$, seria definida como um encadeamento de produtos de séries da forma

$$\mu_{r,c}^k = \sum_{i_{k-1}}^{\infty}\left(\sum_{i_{k-2}}^{\infty}\left(\cdots\left(\sum_{i_2}^{\infty}\left(\sum_{i_1}^{\infty}\mu_{r,i_1}\mu_{i_1,i_2}\right)\mu_{i_2,i_3}\right)\cdots\right)\mu_{i_{k-2},i_{k-1}}\right)\mu_{i_{k-1},c},$$

isto é, seguindo a definição formal de multiplicação de matrizes.

Segue que ao truncar a matriz na ordem n aparecem erros na matriz final, oriundos daqueles elementos que não foram considerados. Uma das formas de evitar este erro é considerar a altura do operador para calcular estas potências, isto é, calculam-se as potências considerando a dimensão mínima necessária, na qual após truncada os erros ficam de fora. É uma maneira que resolve o problema mas envolve o cálculo em dimensões maiores que as aplicadas.

Consideremos um primeiro exemplo que consiste em calcular a norma da diferença entre a matriz $\mathsf{M}_{\mathcal{Z}}^{k\,**}$, que é truncada na dimensão desejada e posteriormente calculada a potência (chamaremos esta forma de computar de Formal**), e a matriz $\mathsf{M}_{\mathcal{Z}}^{k\,*}$, que é primeiramente calculada a potência na matriz aumentada, com dimensão $n+k$, e somente depois truncada (chamaremos esta forma de computar de Formal*). Os resultados são mostrados na Figura 5.5

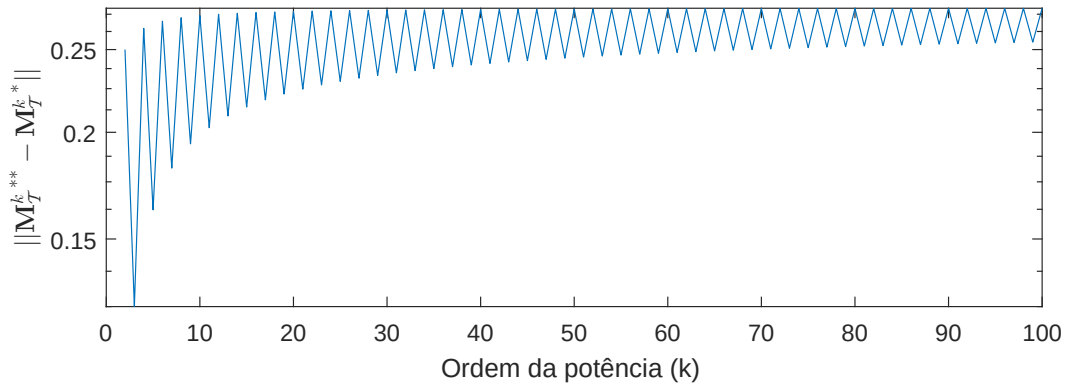

Figura 5.5: Norma-2 da diferença entre as potências das matrizes $\mathsf{M}_{\mathcal{T}}$ calculadas considerando e não considerando a ordem de truncamento.

Relembremos que, por definição, a potência k da matriz $\mathsf{M}_{\mathcal{Z}}$ traduz a ação do operador que incrementa k unidades o grau do polinómio $y_n = \sum_{i=0}^{n} a_i Z_i = \mathcal{Z}\mathsf{a}_{\mathcal{Z}}$, isto é, tal que $\mathcal{Z}\mathsf{M}_{\mathcal{Z}}^k\mathsf{a}_{\mathcal{Z}} = x^k y_n$.

A proposição seguinte apresenta uma forma alternativa de calcular as potências $\mathsf{M}_{\mathcal{Z}}^k$, $k \geq 0$. Tirando partido da estrutura tridiagonal das matrizes $\mathsf{M}_{\mathcal{Z}}$ e do facto de os seus elementos constituírem os coeficientes da relação de recorrência característica dos polinómios $\mathcal{Z}$, construímos uma relação de recorrência para os elementos não nulos das matrizes $\mathsf{M}_{\mathcal{Z}}^k$.

Proposição 5.7. *Sejam α_i, β_i e γ_i, $i \geq 0$, os coeficientes da relação de recorrência a três termos de uma base de polinómios ortogonais $\mathcal{Z}$, e seja $\mathsf{M}_{\mathcal{Z}}$ a matriz tal que $x\mathcal{Z} = \mathcal{Z}\mathsf{M}_{\mathcal{Z}}$, definida na Subsecção 4.2.3, então $\mathsf{M}_{\mathcal{Z}}^k = [\mu_{i,j}^{(k)}]_{i,j\geq 0}$, $k \geq 1$, é a matriz banda com os elementos não nulos dados por*

$$\mu_{i,j}^{(k)} = \mu_{i-1,j}^{(k-1)}\alpha_{i-1} + \mu_{i,j}^{(k-1)}\beta_i + \mu_{i+1,j}^{(k-1)}\gamma_{i+1}, \quad i = j - k(1)j + k, \tag{5.19}$$

considerando nulos os elementos com um índice negativo.

Demonstração. A demonstração pode fazer-se por indução sobre k. Considerando que $\mathsf{M}_{\mathcal{Z}}$ é a matriz tridiagonal dada por (5.8), então

$$\begin{aligned} \mu_{ij}^{(2)} &= \mathsf{e}_{i+1}^T\mathsf{M}_{\mathcal{Z}}^2\mathsf{e}_{j+1} = (\mathsf{e}_{i+1}^T\mathsf{M}_{\mathcal{Z}})(\mathsf{M}_{\mathcal{Z}}\mathsf{e}_{j+1}) \\ &= \left(\alpha_{i-1}\mathsf{e}_i^T + \beta_i\mathsf{e}_{i+1}^T + \gamma_{i+1}\mathsf{e}_{i+2}^T\right)\left(\alpha_j\mathsf{e}_{j+2} + \beta_j\mathsf{e}_{j+1} + \gamma_j\mathsf{e}_j\right),\ i,j \geq 0 \end{aligned}$$

logo são nulos os elementos tais que $i + 2 < j$ ou $i > j + 2$. Para os restantes,

$$\mu_{ij}^{(2)} = \alpha_{i-1}\mu_{i-1,j}^{(1)} + \beta_i\mu_{i,j}^{(1)} + \gamma_{i+1}\mu_{i+1,j}^{(1)}$$

o que termina a demonstração do resultado para $k = 2$. Admitindo que (5.19) é válida, pretendemos mostrar a mesma propriedade para a matriz $\mathsf{M}_{\mathcal{Z}}^{k+1}$. Neste caso, por hipótese

$$\begin{aligned} \mu_{ij}^{(k+1)} &= (\mathsf{e}_{i+1}^T\mathsf{M}_{\mathcal{Z}})(\mathsf{M}_{\mathcal{Z}}^k\mathsf{e}_{j+1}) \\ &= \left(\alpha_{i-1}\mathsf{e}_i^T + \beta_i\mathsf{e}_{i+1}^T + \gamma_{i+1}\mathsf{e}_{i+2}^T\right)\left(\sum_{r=j-k}^{j+k}\mu_{rj}^{(k)}\mathsf{e}_{r+1}\right),\ i,j \geq 0, \end{aligned}$$

logo são nulos os elementos tais que $i + 2 < j - k + 1$ ou $i > j + k + 1$ e os restantes

$$\mu_{i,j}^{(k+1)} = \alpha_{i-1}\mu_{i-1,j}^{(k)} + \beta_i\mu_{i,j}^{(k)} + \gamma_{i+1}\mu_{i+1,j}^{(k)},\ i = j - k - 1(1)j + k + 1$$

□

Para testar a eficácia da Proposição 5.7, em termos de custos computacionais, realizamos o seguinte teste: Consideramos a base de Chebyshev de primeira espécie nas dimensões 2^n, $n = 5(1)10$, e computamos a média dos tempos em 10 execuções dos cálculos de $\mathsf{M}_{\mathcal{T}}^k$, $k = 2(1)100$, nas versões formal

$$\mathsf{M}_{\mathcal{T}}^{k\,*} = \prod_{i=1}^{k}\mathsf{M}_{\mathcal{T}}$$

e recorrência

$$\mathsf{M}_{\mathcal{T}}^{k} = \mu_{i-1,j}^{(k-1)}\alpha_{i-1} + \mu_{i,j}^{(k-1)}\beta_i + \mu_{i+1,j}^{(k-1)}\gamma_{i+1}, \quad i = j-k-1(1)j+k+1.$$

Os resultados são apresentados na Figura 5.6.

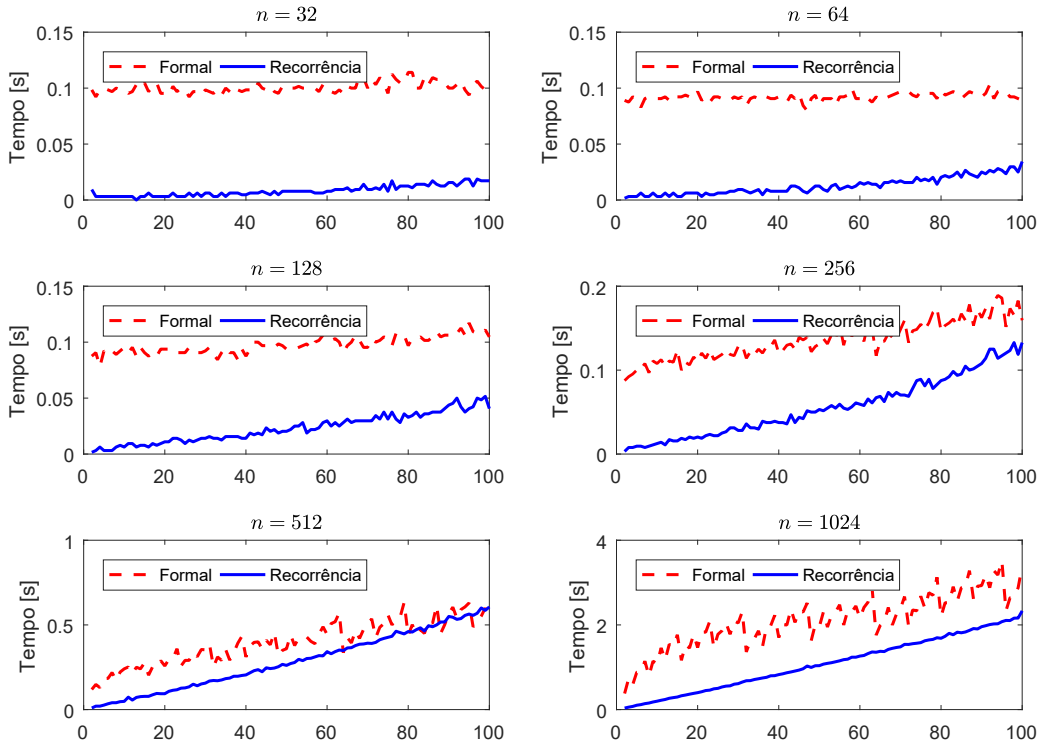

Figura 5.6: Tempo de computação para as potências das matrizes $\mathsf{M}_{\mathcal{T}}$, formal e por recorrência.

A partir dos resultados apresentados na Figura 5.6 concluímos que os tempos de computação das potências de $\mathsf{M}_{\mathcal{T}}$ por recorrência são em geral menores que os tempos quando estas potências são calculadas de forma usual. Os valores nos eixos das abscissas referem-se à ordem da potência k. Estes resultados concordam com o facto de o produto de duas matrizes quadradas de dimensão n envolver n^3 produtos dos seus elementos. Enquanto que o cálculo dos n^2 elementos de $\mathsf{M}_{\mathcal{T}}^2$ pela recorrência apenas necessitar de $3n^2$ produtos. Acresce que o cálculo exacto da matriz $\mathsf{M}_{\mathcal{T}}^k$ obriga a operar com a matriz truncada à dimensão $(n+k)$, pelo que são necessários $(k-1)(n+k)^3$ produtos elementares. Em contraste com os $3(k-1)n^2$ produtos elementares envolvidos no cálculo por recorrência.

Interessa-nos saber se há diferenças, no que diz respeito à precisão no cálculo dos resultados finais, entre estas formas de calcular as potências, e para isso realizamos o seguinte teste: Computamos a norma-2 da diferença entre as potências $\mathsf{M}_{\mathcal{Z}}^k$ (recorrência)

e $\mathsf{M}_{\mathcal{Z}}^{k\,*}$ (formal), para as bases de Chebyshev de primeira e segunda espécie e Legendre, para dimensão $n = 2^i,\ i = 5(1)10$ e $k = 2(1)100$.

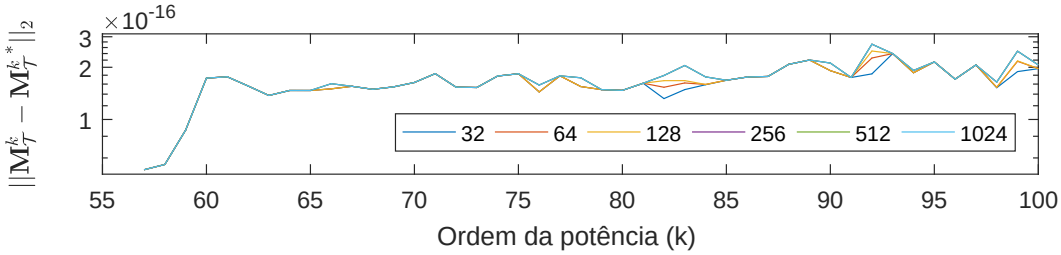

Figura 5.7: Norma-2 da diferença entre as potências $\mathsf{M}_{\mathcal{T}}^k,\ k = 56(1)100$, com dimensão $n = 2^i,\ i = 5(1)10$, calculadas formalmente e por recorrência.

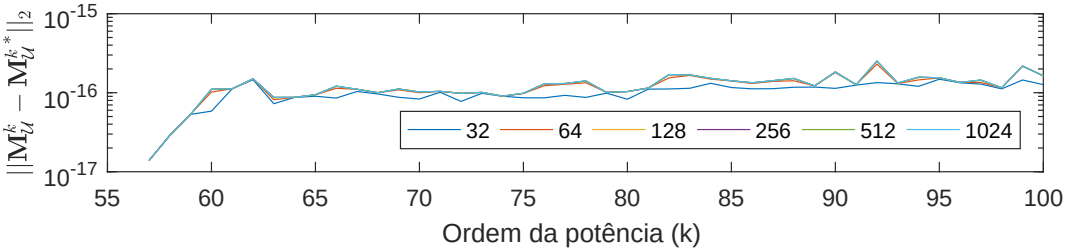

Figura 5.8: Norma-2 da diferença entre as potências $\mathsf{M}_{\mathcal{U}}^k,\ k = 56(1)100$, com dimensão $n = 2^i,\ i = 5(1)10$, calculadas formalmente e por recorrência.

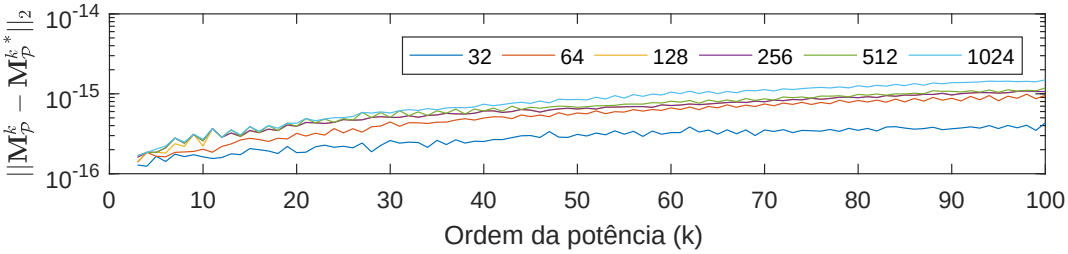

Figura 5.9: Norma-2 da diferença entre as potências $\mathsf{M}_{\mathcal{P}}^k,\ k = 56(1)100$, com dimensão $n = 2^i,\ i = 5(1)10$, calculadas formalmente e por recorrência.

As Figuras 5.7, 5.8 e 5.9 mostram que a diferença $||\mathsf{M}_{\mathcal{Z}}^k - \mathsf{M}_{\mathcal{Z}}^{k\,*}||_2$ é da ordem da precisão da máquina, resultado este interessante pelo facto da recorrência requerer menos esforço computacional para obter um resultado semelhante àquele calculado de forma usual. Os valores k omitidos nestes gráficos devem-se ao facto da diferença ser nula nestas ordens de potência.

Com o intuito último de testar a eficácia da Proposição 5.7, agora em termos de qualidade na aproximação resultante da aplicação do método tau a um problema

diferencial, mostramos um exemplo propositalmente criado para conter um coeficiente polinomial de ordem elevada (x^{100}).

Exemplo 5.4. Seja o problema diferencial linear ordinário de coeficientes polinomiais

$$\begin{cases} (-2x^4 + x^{10} + 50x^{100})\dfrac{d}{dx}y + y = 3(-2x^4 + x^{10} + 50x^{100})x^2 + x^3 \\ y(0) = 0 \end{cases},$$

de solução exata $y = x^3$.

Em termos de aproximação pelo método tau, temos $\mathsf{D}_{\mathcal{Z}} = (-2\mathsf{M}_{\mathcal{Z}}^4 + \mathsf{M}_{\mathcal{Z}}^{10} + 50\mathsf{M}_{\mathcal{Z}}^{100})\mathsf{N}_{\mathcal{Z}} + \mathsf{I}$, $\mathsf{C}_{\mathcal{Z}} = [Z_0(0),\ Z_1(0),\ \ldots]$, $\mathsf{b} = [0,\ \mathsf{f}_{\mathcal{Z}},\ 0,\ \ldots]^T$ e $\mathsf{f}_{\mathcal{Z}}$ é o vetor dos coeficientes $f_i,\ i = 0, 1, \ldots$ tais que descrevem f na base $\mathcal{Z}$. Nestes termos, aproximamos a solução do Exemplo 5.4 para $\mathcal{Z} = \mathcal{T}^*$: os polinómios de Chebyshev de primeira espécie no intervalo $[0,\ 1]$, e o resultado é apresentado na Figura 5.10.

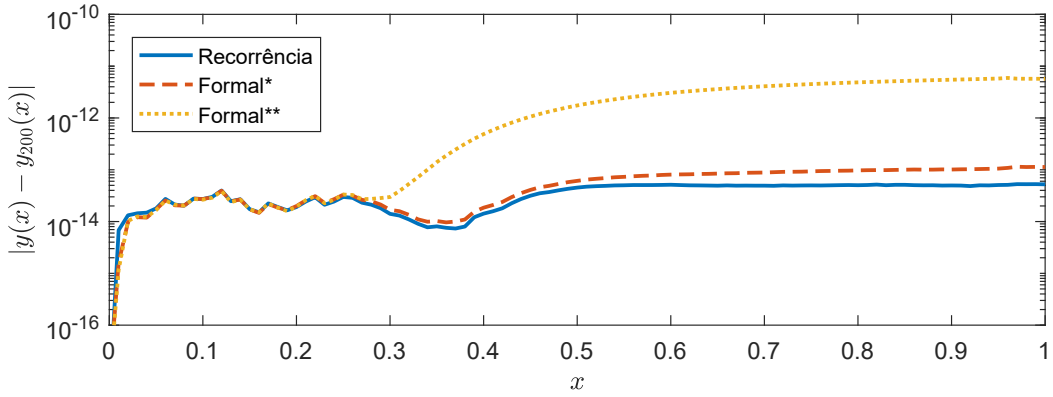

Figura 5.10: Erros para o Exemplo 5.4, considerando as potências de $\mathsf{M}_{\mathcal{T}}$ pela Proposição 5.7 (Recorrência), formal com aumento e truncamento (Formal*) e formal sem aumento (Formal**).

A Proposição 5.7 aparenta ser, face aos resultados obtidos, ligeiramente melhor em termos de precisão da aproximação, frente à forma usual de computar as potências de $\mathsf{M}_{\mathcal{Z}}$. Cabe ainda relevar que a esparsidade de $\mathsf{M}_{\mathcal{Z}}$ pode contribuir significativamente para a redução dos tempos dos cálculos de suas potências, mas ainda assim, conforme o resultado apresentado na Figura 5.10, a Proposição 5.7 mostra-se mais eficiente. Uma última comparação pode ser feita tomando a representação da solução exata na base de Chebyshev de primeira espécie ortogonal no intervalo $[0,\ 1]$: $x^3 = \frac{5}{16}T_0 + \frac{15}{32}T_1 + \frac{3}{16}T_2 + \frac{1}{32}T_3 = \mathcal{T}\mathsf{a}_{\mathcal{T}}$. Donde encontramos as normas $||\mathsf{a}_{1000} - \mathsf{a}_{\mathcal{T}}|| = 5,4 \cdot 10^{-13}$, $||\mathsf{a}_{1000}^{*} - \mathsf{a}_{\mathcal{T}}|| = 1,01 \cdot 10^{-12}$ e $||\mathsf{a}_{1000}^{**} - \mathsf{a}_{\mathcal{T}}|| = 2,14 \cdot 10^{-10}$, que mostram-se semelhantes aos resultados da Figura 5.10. Outras bases, como Chebyshev de segunda espécie e Legendre, também mostram mesmo comportamento para a aproximação deste exemplo.

Para a `Tau Toolbox`, a Proposição 5.7 foi implementada como **`mpower`**.

5.5 Coeficientes de linearização do produto de polinómios

Como visto na Secção 4.7, o processo de linearização faz com que apareçam produtos de polinómios escritos em alguma base $\mathcal{Z}$, isto é, $\mathsf{p}_{\mathcal{Z}}\mathsf{q}_{\mathcal{Z}}$. Para calcular este produto, a forma habitual é fazer a mudança de base de $\mathcal{Z}$ para $\mathcal{X}$, aplicar a convolução e finalmente o produto é mudado novamente, agora para $\mathcal{Z}$:

$$\mathsf{p}_{\mathcal{Z}}\mathsf{q}_{\mathcal{Z}} = \mathsf{V}_{\mathcal{Z}}^{-1}\text{conv}(\mathsf{V}_{\mathcal{Z}}\mathsf{p}_{\mathcal{Z}}, \mathsf{V}_{\mathcal{Z}}\mathsf{q}_{\mathcal{Z}}),$$

que é uma forma instável à medida que a dimensão das matrizes de mudança de base $\mathsf{V}_{\mathcal{Z}}$ e $\mathsf{V}_{\mathcal{Z}}^{-1}$ cresce. Para ultrapassar esta instabilidade introduzimos a utilização de coeficientes de linearização do produto de polinómios.

Coeficientes de linearização

Como visto, a linearização de um problema integro diferencial resulta numa sucessão de problemas lineares envolvendo o produto de v polinómios, e uma vez que estamos aplicando o método tau com uma base ortogonal, o produto é da forma

$$p_1 \times \ldots \times p_v = \prod_{j=1}^{v} \sum_{i=0}^{n_j} p_{ji} Z_i,$$

onde p_{ji} são os coeficientes do polinómio $p_j = \mathcal{Z}\mathsf{p}_j$. Algumas maneiras de se calcular estes produtos são: (i) avaliar cada um dos fatores p_j num conjunto discreto de valores x_i, $i = 0, 1, \ldots k$, calcular os produtos dos valores $p_j(x)$ em cada ponto x_i e interpolar o resultado com coeficientes em $\mathcal{Z}$, obtendo uma aproximação; (ii) passar cada p_j para a base das potências, aplicar a convolução e voltar para a base ortogonal e (iii) obter diretamente os coeficientes do produto já na base ortogonal, evitando qualquer mudança de base. Sendo este último processo mais exato e estável, é o que iremos utilizar.

Definição 5.2. *Seja $\mathcal{Z} = [Z_0, Z_1, \ldots]$ uma sequência de polinómios ortogonais, designam-se por coeficientes de linearização os valores $l_{i,j,k}$, tais que*

$$Z_i Z_j = \sum_{k=0}^{i+j} l_{i,j,k} Z_k.$$

Uma vez que Z constitui uma base do espaço dos polinómios de qualquer grau, a existência e unicidade destes coeficientes de linearização constitui um resultado elementar. No entanto, o desenvolvimento de fórmulas de cálculo para estes coeficientes constitui uma tarefa mais exigente. Ver, por exemplo, Askey (1975).

Proposição 5.8. *Seja $\mathcal{Z} = [Z_0, Z_1, \ldots]$ uma sequência de polinómios ortogonais obedecendo a uma relação de recorrência com coeficientes α_n, β_n e γ_n, $n \geq 0$, então os coeficientes de linearização $l_{i,j,k}$ definidos em 5.2 satisfazem a seguinte relação*

$$l_{i,j+1,k} = \alpha_j^{-1} \left[\alpha_i l_{i+1,j,k} - \gamma_j l_{i,j-1,k} + (\beta_i - \beta_j) l_{i,j,k} + \gamma_i l_{i-1,j,k}\right],$$

com os valores iniciais

$$\begin{cases} l_{i,0,k} = \delta_{ik} \\ l_{i,1,i-1} = \gamma_i \\ l_{i,1,i} = \beta_i \\ l_{i,1,i+1} = \alpha_i \\ l_{i,1,j} = 0 \quad \textit{sempre que} \quad |j-i| > 1 \end{cases}$$

para $k = 0(1)i + j + 1$.

Demonstração. Ver Gavina (2016) pg. 82. □

O resultado desta proposição juntamente com a proposição seguinte permite obter os coeficientes da representação do produto de dois polinómios na base ortogonal $\mathcal{Z}$.

Proposição 5.9. *Sejam* $p = \sum_{i=0}^{n} a_i Z_i$ *e* $q = \sum_{i=0}^{m} b_i Z_i$ *dois polinómios representados na base de polinómios ortogonais* $\mathcal{Z}$*. Nestas condições, o produto* pq *é dado por*

$$pq = \sum_{i=0}^{n} \left(a_i b_i Z_i^2 + \sum_{j=1}^{n-i} (a_i b_{i+j} + a_{i+j} b_i) Z_i Z_{i+j} \right).$$

Obs. Sem perda de generalidade, supomos que $n = m$, mesmo que não sejam inicialmente iguais tornamos $n = \max\{n, m\}$ e acrescentamos coeficientes nulos ao polinómio de menor grau.

Demonstração.

$$\begin{aligned} pq &= \left(\sum_{i=0}^{n} a_i Z_i \right) \left(\sum_{i=0}^{n} b_i Z_i \right) \\ &= \sum_{i=0}^{n} \left(a_i b_i Z_i^2 + \sum_{j=0}^{i-1} (a_i b_j + a_j b_i) Z_i Z_j \right) \\ &= \sum_{i=0}^{n} \left(a_i b_i Z_i^2 + \sum_{j=1}^{n-i} (a_i b_{i+j} + a_{i+j} b_i) Z_i Z_{i+j} \right). \end{aligned}$$

□

A proposição seguinte concretiza os resultados anteriores para o caso do produto de polinómios representados na base de Chebyshev.

Proposição 5.10. *Sejam* $p = \sum_{i=0}^{n} a_i T_i$ *e* $q = \sum_{i=0}^{n} b_i T_i$ *dois polinómios representados na*

base de Chebyshev, assim, os coeficientes do produto pq, são os coeficientes da expansão $pq = \frac{1}{2}\sum_{k=0}^{2n} c_k T_k$, *com os valores* c_k *dados por*

$$c_k = \begin{cases} 2a_0b_0 + \sum_{i=1}^{n} a_i b_i, & k = 0 \\ c'_k + \sum_{i=0}^{n-k}(a_i b_{k+i} + a_{k+i}b_i) + \sum_{i=0}^{\lfloor\frac{k-1}{2}\rfloor}(a_i b_{k-i} + a_{k-i}b_i), & k = 1,\ldots,n \\ c'_k = \sum_{i=k-n}^{\lfloor\frac{k-1}{2}\rfloor}(a_i b_{k-i} + a_{k-i}b_i), & k = n+1,\ldots,2n-1 \\ a_n b_n, & k = 2n \end{cases},$$

onde $c'_k = a_{k/2}b_{k/2}$ *se* k *é par e zero caso contrário.*

Demonstração. Ver Gavina et al. (2016). □

Na `Tau Toolbox`, os coeficientes de linearização do produto de polinómios em bases ortogonais foram implementados nas funções **`chebypolyprod`**, **`chebypolypow`**, **`legpolyprod`** e **`legpolypow`**, para as bases de Chebyshev e Legendre, e na função **`orthopolyprod`** para o caso geral. Ilustramos as vantagens da utilização destes resultados, em termos de estabilidade numérica, com a resolução de um problema diferencial não linear.

Exemplo 5.5. Seja o problema diferencial não linear

$$\begin{cases} \frac{d}{dx}y - y^2 = 0, & x \in]0\ 1[\\ y(0) = \frac{1}{2} \end{cases},$$

cuja solução exata é $y = (2-x)^{-1}$.

Solução. A aproximação de Taylor de grau 1 para o termo não linear, em torno de y_0, é $y^2 \approx 2y_0y - y_0^2$, e portanto reescrevemos o problema (5.5), iterativamente por Newton, como

$$\begin{cases} \frac{d}{dx}y_n^{(k+1)} - 2y^{(k)}y_n^{(k+1)} = -{y^{(k)}}^2, & x \in]0,\ 1[, \quad k \geq 0 \\ y^{(k+1)}(0) = \frac{1}{2} \end{cases},$$

com $y^{(0)}(0) = \mathcal{Z}_n \mathsf{y}^{(0)}$, onde $\mathsf{y}^{(0)} = [\frac{1}{2}Z_0(0)^{-1}, \underbrace{0 \quad \ldots \quad 0}_{n-1 \text{ vezes}}]^T$, e assim o operador é definido como

$$\mathsf{H}_{\mathcal{Z}}^{(k)} = \mathsf{N}_{\mathcal{Z}} - 2\mathsf{Y}^{(k)}(\mathsf{M}_{\mathcal{Z}}), \quad \mathsf{Y}^{(k)}(\mathsf{M}_{\mathcal{Z}}) = \sum_{i=0}^{n-1} y_{n,i}^{(k)} Z_i(\mathsf{M}_{\mathcal{Z}}).$$

São apresentados na Figura 5.11 os erros da aproximação com $n = 25$, quando usada a convolução e quando usados os coeficientes de linearização para calcular os coeficientes dos polinómios ${y^{(k)}}^2$.

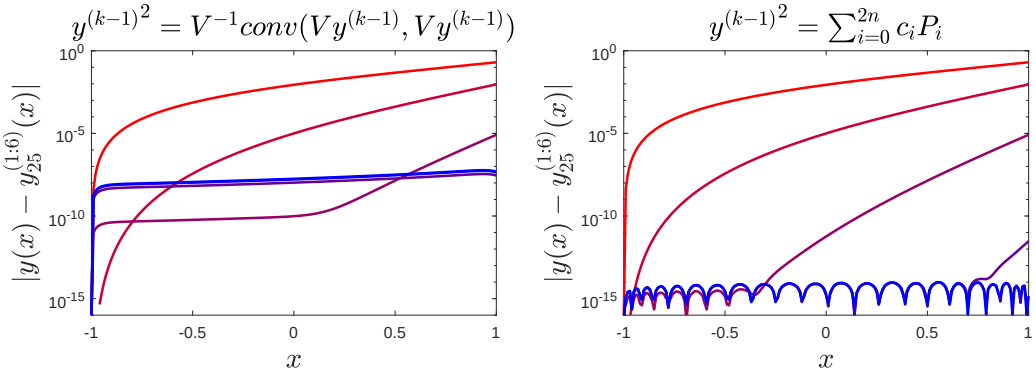

Figura 5.11: Resultados obtidos com a operação de convolução e com a utilização de coeficientes de linearização, com $k = 1(1)6$ iterações, utilizando polinómios de Chebyshev, com grau $n = 25$ no Exemplo 5.5.

Conforme a Figura 5.11 ilustra, observamos que os erros introduzidos pela aplicação da convolução condicionaram um erro absoluto máximo nos aproximantes tau da ordem de $1 \cdot 10^{-8}$, mesmo que o processo itere para $k > 6$. A aplicação dos coeficientes de linearização evidenciam o grande impacto sobre a precisão da aproximação, que para $k \geq 6$ mantém-se na precisão da máquina.

A sensibilidade da solução ao cálculo dos elementos dos sistemas lineares associados ao método tau, aparentemente crescente com o grau das aproximações a calcular, aparece associada ao facto de as matrizes envolvidas apresentarem número de condição elevado. Com estes coeficientes de linearização resolvemos esta instabilidade numérica ao corrigir quer o cálculo dos termos independentes, como ilustrado no exemplo com os coeficientes de $y^{(k)^2}$, mas também no cálculo das matrizes, quando estas envolvem o cálculo de $p(\mathsf{M}_{\mathcal{Z}})$ e p é um polinómio resultante do produto de dois polinómios representados na base $\mathcal{Z}$. Esta última situação revela também a necessidade de ter calculados com grande precisão os elementos das matrizes $\mathsf{M}_{\mathcal{Z}}$, introduzidos na Secção 5.3.

5.6 Conclusões e considerações

Nas Secções anteriores tratamos do problema de calcular com grande precisão os elementos dos sistemas algébricos associados ao método tau. Ao fim de cada Secção já adiantamos os nomes das funções desenvolvidas em MATLAB no âmbito da criação da Tau Toolbox, que por sua vez é apresentada na Parte III desta Tese. Experimentos numéricos neste Capítulo ilustraram o positivo impacto na qualidade da solução de problemas integro-diferenciais lineares e não lineares. No Capítulo seguinte tratamos de uma forma de implementar a resolução dos sistemas lineares algébricos associados à formulação operacional do método tau, baseada em complementos de Schur com estimativa do erro.

Capítulo 6

Esquema iterativo com complementos de Schur

Sumário

Nos problemas abordados até aqui, fixa-se um grau $n-1$ ao polinómio y_n que aproxima a solução y de um problema integro-diferencial, e obtemos o resíduo que resulta desta aproximação. No caso de o resíduo, ou o erro, não ser satisfatório, tornávamos a aplicar o método com um novo grau n e repetimos o processo até obter a precisão requerida. Apresentamos neste Capítulo uma forma de automatizar este processo, evitando a escolha prévia do grau n, mas sim pré estabelecendo a precisão desejada.

6.1 Estimativa do erro

Voltemos à problemática de encontrar, no sentido do método tau, uma solução para a equação

$$\begin{cases} \mathfrak{D}y_n = f + \tau_n, \quad x \in]a,\ b[\\ g_j(y_n) = \sigma_j, \quad j = 1(1)\nu \end{cases}, \quad \mathfrak{D} = \sum_{r=0}^{\nu} p_r \frac{d^r}{dx^r}. \tag{6.1}$$

Algumas medidas frequentemente utilizadas para avaliar a qualidade de uma solução, podem utilizar-se como critério de paragem num processo iterativo, como

$$\max|y_{n+1} - y_n|, \ \int_a^b |y_{n+1} - y_n| w dx \quad \text{ou} \quad ||a_{n+1} - a_n||_\infty.$$

Nesta Secção, em vez de adotar critérios de paragem baseados nestas diferenças, adotamos uma estratégia baseada em estimativas do erro absoluto. Seja

$$e_n = y - y_n,$$

o erro com que a solução aproximada y_n representa a solução exata y. Uma vez que $\mathfrak{D}$ é um operador linear e g_j são funcionais lineares, então o erro e_n é solução exata do problema diferencial

$$\begin{cases} \mathfrak{D}e_n = \tau_n, & x \in]a,\ b[\\ g_j(e_n) = 0, & j = 1(1)\nu \end{cases} . \tag{6.2}$$

Da mesma forma que para o problema original, podemos admitir que a solução de (6.2) admite um desenvolvimento em série

$$e_n = \mathcal{Z}_n \mathsf{c}_n = \sum_{i \geq 0} c_{n,i} Z_i,$$

cujos coeficientes podem aproximar-se resolvendo (6.2) pelo próprio método tau, donde obtemos o problema perturbado com τ_m, $m > n$

$$\begin{cases} \mathfrak{D}e_{n,m} = \tau_n + \tau_m, & x \in]a,\ b[\\ g_j(e_{n,m}) = 0, & j = 1(1)\nu \end{cases} ,$$

onde

$$\tau_m = \sum_{i=m+1-\nu}^{m+h} \tau_i Z_i, \tag{6.3}$$

tomando $m = n + \nu + h$ permite considerar os $\nu + h$ coeficientes de τ_n.

O Exemplo a seguir ilustra a diferença entre o erro estimado e o erro real.

Exemplo 6.1. Voltemos a usar Exemplo 4.2 do Capítulo 4. Com o bloco das últimas 3 equações do sistema (4.5) calculamos os coeficientes τ_6, τ_7 e τ_8 e obtemos o sistema, já truncado

$$\begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & 0 & 2 & 9 & 4 & 55 & 6 & 161 \\ 0 & 0 & 3 & 6 & 48 & 10 & 192 & 14 \\ & 0 & 1 & 12 & 8 & 90 & 12 & 294 \\ & & 1 & 2 & 26 & 10 & 144 & 14 \\ & & & 3 & 3 & 45 & 12 & 210 \\ & & & & 6 & 4 & 69 & 14 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \tau_6 \\ \tau_7 \\ \tau_8 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

cuja solução fornece os coeficientes $\mathsf{c} = [c_0,\ \ldots,\ c_7]$ do polinómio para o erro estimado. Para este exemplo, obtemos $\max |e_{n,m} - e| = 9,93 \cdot 10^{-15}$.

A aproximação para o erro $e_{n,m} \approx e_n$ dá-nos uma estimativa que pode ser usada como critério de paragem em um esquema iterativo, como proposto na Secção seguinte.

6.2 Descrição do método

Propomos aplicar um esquema iterativo para aproximar a solução de (6.1), onde o sistema (3.3) é resolvido por partes, incrementando o valor de n em cada

iteração e utilizando complementos de Schur (uma versão preliminar, menos sofisticada é apresentada em Trindade et al. (2016)). Com esta formulação é possível obter aproximações consecutivas para a solução y e para o erro, usando a mesma parte da matriz T. Devemos primeiramente obter a matriz T do problema (6.1) com dimensão grande o suficiente para aplicar o esquema iterativo. Para representar o proposto esquema iterativo, vamos começar por definir a matriz T em blocos em cada iteração k. Nas fórmulas seguintes $(*)$ está relacionado com cada iteração, $[*]$ está relacionado com as dimensões (de até) e n_o representa a ordem inicial da aproximação. Relativamente à matriz T, definimos os seguintes blocos

$$\mathsf{T}_{11}^{(k)} = \mathsf{T}^{[1:n_o+(\nu+h)(k-1),1:n_o+(\nu+h)(k-1)]},$$

$$\mathsf{T}_{12}^{(k)} = \mathsf{T}^{[1:n_o+(\nu+h)(k-1),n_o+1+(\nu+h)(k-1):n_o+(\nu+h)k]},$$
$$\mathsf{T}_{21}^{(k)} = \mathsf{T}^{[n_o+1+(\nu+h)(k-1):n_o+(\nu+h)k,1:n_o+(\nu+h)(k-1)]},$$
$$\mathsf{T}_{22}^{(k)} = \mathsf{T}^{[n_o+1+(\nu+h)(k-1):n_o+(\nu+h)k,n_o+1+(\nu+h)(k-1):n_o+(\nu+h)k]},$$

e, definamos

$$\mathsf{b}_1^{(k)} = \mathsf{b}^{[1:n_o+(\nu+h)(k-1)]}$$

e

$$\mathsf{b}_2^{(k)} = \mathsf{b}^{[n_o+1+(\nu+h)(k-1):n_o+(\nu+h)k]}$$

os vetores relacionados com a respetiva parte de b.

Definição 6.1. *Sejam duas matrizes* A *e* B*, de tal forma que o número de colunas de* A *seja igual ao número de linhas de* B*. Define-se por* $\oslash$ *o operador tal que* $\mathsf{C} = \mathsf{A} \oslash \mathsf{B}$ *onde* $\mathsf{AC} = \mathsf{B}$*.*

Considerando um pequeno $n_o \geq \nu + h$ e $k = 1$ podemos obter $\mathsf{T}_{11}^{(1)} = \mathsf{L}^{(1)}\mathsf{U}^{(1)}$ onde $\mathsf{L}^{(1)}\mathsf{U}^{(1)}$ é a decomposição LU de $\mathsf{T}_{11}^{(1)}$. Uma primeira aproximação tau, $y_n^{(1)} = \mathcal{Z}\mathsf{a}^{(1)}$ de grau $n_o - 1$ pode então facilmente ser obtida calculando

$$\mathsf{a}^{(1)} = \mathsf{U}^{(1)} \oslash \left(\mathsf{L}^{(1)} \oslash \mathsf{b}_1^{(1)}\right).$$

A partir de agora, cada iteração $k \geq 1$ irá resolver dois sistemas de equações lineares algébricas com uma mesma matriz, de dimensão $n(k) = n_o + (\nu + h)(k-1)$, onde a solução do primeiro sistema está relacionada com a estimativa do erro $\mathsf{c}^{(k)}$ e a solução do segundo sistema trata-se da próxima aproximação $\mathsf{a}^{(k+1)}$. Assim, ao considerar os seguintes blocos

$$\begin{bmatrix} \mathsf{T}_{11}^{(k)} & \mathsf{T}_{12}^{(k)} \\ \mathsf{T}_{21}^{(k)} & \mathsf{T}_{22}^{(k)} \end{bmatrix} \begin{bmatrix} \mathsf{c}_1^{(k)} & \mathsf{a}_1^{(k+1)} \\ \mathsf{c}_2^{(k)} & \mathsf{a}_2^{(k+1)} \end{bmatrix} = \begin{bmatrix} \mathsf{d}_1^{(k)} & \mathsf{b}_1^{(k)} \\ \mathsf{d}_2^{(k)} & \mathsf{b}_2^{(k)} \end{bmatrix},$$

onde

$$\mathsf{d}_2^{(k)} = \tau_n^{(k)} = \mathsf{T}_{21}^{(k)}\mathsf{a}^{(k)}$$

é o resíduo (6.3) da aproximação anterior $\mathsf{a}^{(k)}$ (obtido por $\mathsf{T}_{11}^{(k)}$) e por (6.2)

$$\mathsf{d}_1^{(k)} = [\overbrace{0,\ 0,\ \ldots,\ 0}^{n_o+(\nu+h)(k-1)}]^T,$$

resolvemos (6.1) aplicando os complementos de Schur

$$\begin{bmatrix} \mathsf{T}_{11}^{(k)} & \mathsf{T}_{12}^{(k)} \\ 0 & \mathsf{T}_{22}^{(k)} - \mathsf{T}_{21}^{(k)}\mathsf{T}_{11}^{(k)^{-1}}\mathsf{T}_{12}^{(k)} \end{bmatrix} \begin{bmatrix} \mathsf{c}_1^{(k)} \\ \mathsf{c}_2^{(k)} \end{bmatrix} = \begin{bmatrix} \mathsf{d}_1^{(k)} \\ \mathsf{d}_2^{(k)} - \mathsf{T}_{21}^{(k)}\mathsf{T}_{11}^{(k)^{-1}}\mathsf{d}_1^{(k)} \end{bmatrix}, \tag{6.4}$$

e então o vetor c_2 será dado por

$$\mathsf{c}_2^{(k)} = \left(\mathsf{T}_{22}^{(k)} - \mathsf{T}_{21}^{(k)}\mathsf{T}_{11}^{(k)^{-1}}\mathsf{T}_{12}^{(k)}\right) \oslash \left(\mathsf{d}_2^{(k)} - \mathsf{T}_{21}^{(k)}\mathsf{T}_{11}^{(k)^{-1}}\mathsf{d}_1^{(k)}\right),$$

mas, uma vez que $\mathsf{T}_{11}^{(k)}$ foi decomposto por fatorização LU, temos

$$\mathsf{c}_2^{(k)} = \left(\mathsf{T}_{22}^{(k)} - \mathsf{T}_{21}^{(k)}\mathsf{U}^{(k)} \oslash \mathsf{L}^{(k)} \oslash \mathsf{T}_{12}^{(k)}\right) \oslash \left(\mathsf{d}_2^{(k)} - \mathsf{T}_{21}^{(k)}\mathsf{U}^{(k)} \oslash \mathsf{L}^{(k)} \oslash \mathsf{d}_1^{(k)}\right),$$

e então

$$\mathsf{c}_1^{(k)} = \mathsf{U}^{(k)} \oslash \left(\mathsf{L}^{(k)} \oslash \left(\mathsf{d}_1^{(k)} - \mathsf{T}_{12}^{(k)}\mathsf{c}_2^{(k)}\right)\right).$$

Se a estimativa do erro

$$e_n^{(k)} = \begin{bmatrix} \mathsf{c}_1^{(k)} \\ \mathsf{c}_2^{(k)} \end{bmatrix} \mathcal{Z}_n^{(k)} = \sum_{i=0}^{n_o+(\nu+h)(k-1)} \mathsf{c}_{n,i}^{(k)} Z_i$$

não atingiu a precisão pretendida, procedemos da mesma forma para obter $\mathsf{a}^{(k+1)}$ usando novamente os complementos de Schur

$$\begin{bmatrix} \mathsf{T}_{11}^{(k)} & \mathsf{T}_{12}^{(k)} \\ 0 & \mathsf{T}_{22}^{(k)} - \mathsf{T}_{21}^{(k)}\mathsf{T}_{11}^{(k)^{-1}}\mathsf{T}_{12}^{(k)} \end{bmatrix} \begin{bmatrix} \mathsf{a}_1^{(k+1)} \\ \mathsf{a}_2^{(k+1)} \end{bmatrix} = \begin{bmatrix} \mathsf{b}_1^{(k)} \\ \mathsf{b}_2^{(k)} - \mathsf{T}_{21}^{(k)}\mathsf{T}_{11}^{(k)^{-1}}\mathsf{b}_1^{(k)} \end{bmatrix}, \tag{6.5}$$

e então os vetores $\mathsf{a}_2^{(k+1)}$ e $\mathsf{a}_1^{(k+1)}$ serão dados respetivamente por

$$\mathsf{a}_2^{(k+1)} = \left(\mathsf{T}_{22}^{(k)} - \mathsf{T}_{21}^{(k)}\mathsf{U}^{(k)} \oslash \mathsf{L}^{(k)} \oslash \mathsf{T}_{12}^{(k)}\right) \oslash \left(\mathsf{b}_2^{(k)} - \mathsf{T}_{21}^{(k)}\mathsf{U}^{(k)} \oslash \mathsf{L}^{(k)} \oslash \mathsf{b}_1^{(k)}\right)$$

e

$$\mathsf{a}_1^{(k+1)} = \mathsf{U}^{(k)} \oslash \left(\mathsf{L}^{(k)} \oslash \left(\mathsf{b}_1^{(k)} - \mathsf{T}_{12}^{(k)}\mathsf{a}_2^{(k+1)}\right)\right),$$

a que corresponde

$$y_n^{(k+1)} = \begin{bmatrix} \mathsf{a}_1^{(k+1)} \\ \mathsf{a}_2^{(k+1)} \end{bmatrix}^T \mathcal{Z}_n^{(k)} = \sum_{i=0}^{n_o+(\nu+h)(k-1)} \mathsf{a}_{n,i}^{(k+1)} Z_i,$$

a solução de grau $n_o + (\nu + h)(k - 1)$ de (6.1).

Iterando esta formulação (ver Figura 6.1), estabelecemos como critério de paragem o polinómio $e_n^{(k)}$ conduzir à precisão desejada. O Algoritmo 2 instrui como aplicar o proposto esquema iterativo.

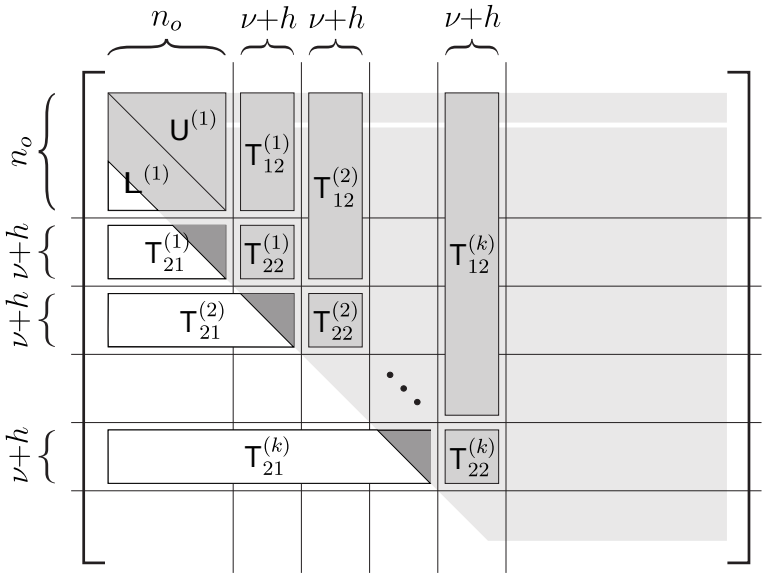

Figura 6.1: Blocos do esquema iterativo baseado em complementos de Schur.

Para incrementar a fatorização LU, como ilustrado na Figura 6.2, podemos aplicar

$$\mathsf{U}^{[:,i]} = \mathsf{L}^{[1:i-1,1:i-1]} \oslash \mathsf{T}^{[1:i-1,i]}, \tag{6.6}$$

$$\mathsf{L}^{[i,:]} = \mathsf{U}^{[1:i-1,1:i-1]^T} \oslash \mathsf{T}^{[i,1:i-1]^T}, \quad \mathsf{L}^{[i,i]} = 1,$$

e

$$\mathsf{U}^{[i,i]} = \mathsf{T}^{[i,i]} - \mathsf{L}^{[i,1:i-1]}\mathsf{U}^{[1:i-1,i]}, \tag{6.7}$$

para $i = n_0 + (\nu + h)(k-1) + 1 : n_0 + (\nu + h)k$.

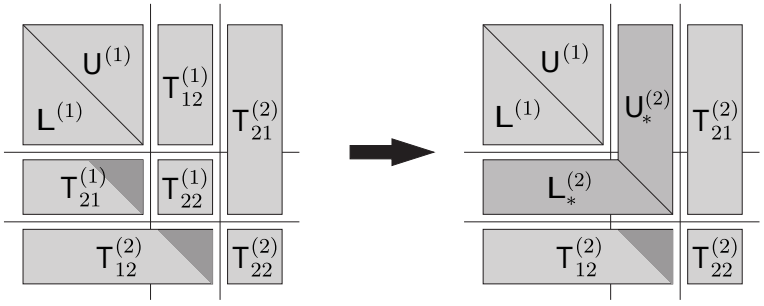

Figura 6.2: Esquema de incrementação de fatorização LU.

Algoritmo 2: Método tau com Complementos de Schur e estimativa do erro.

Dados: matriz T, vetor b, tamanho inicial n_o e precisão desejada ϵ.
Resultado: vetor a (coeficientes na base $\mathcal{Z}$).
início
- $stop \leftarrow \epsilon + 1$
- $\nu \leftarrow \text{ordem}(\mathfrak{D})$
- $h \leftarrow \sup_{n\geq 0}\{\text{grau}(\mathfrak{D}Z_n) - n\}$
- $m \leftarrow \nu + h$
- $l_{\mathsf{T}} \leftarrow$ tamanho de T
- $k = 1$
- Resolve $\mathsf{L}^{(k)}\mathsf{U}^{(k)}\mathsf{a}^{(k)} = \mathsf{b}_1^{(k)}$
- **repita**
  - Calcular $\mathsf{T}_{11}^{(k)}$, $\mathsf{T}_{12}^{(k)}$, $\mathsf{T}_{21}^{(k)}$ e $\mathsf{T}_{22}^{(k)}$ de T
  - $\tau_m^{(k)} \leftarrow \mathsf{T}_{21}^{(k)}\mathsf{a}^{(k)}$
  - Calcular $\mathsf{d}_1^{(k)}$ e $\mathsf{d}_2^{(k)}$
  - Obter $\mathsf{c}^{(k)}$ com (6.4)
  - $e \leftarrow \mathcal{Z}\mathsf{c}^{(k)}$
  - $stop \leftarrow \max|e|$
  - **se** $stop > \epsilon$ **então**
    - Calcular $\mathsf{b}_1^{(k)}$ e $\mathsf{b}_2^{(k)}$
    - Obter $\mathsf{a}^{(k+1)}$ com (6.5)
    - Recompletar LU com (6.6)-(6.7) para obter $\mathsf{L}^{(k+1)}$ e $\mathsf{U}^{(k+1)}$
  - $k \leftarrow k + 1$
- **até** $stop < \epsilon$ **ou** $l_{\mathsf{T}} \geq n_o + mk$
- Mostra a

A fatorização obtida com o Algoritmo 2 não reproduz a fatorização LU de toda a matriz T. Em particular, a relação dos pivôs para a pivotagem parcial é feita dentro de cada bloco. A estabilidade, na prática, da fatorização LU pode ser comprometida. No entanto, dadas as características das matrizes T no método tau, em geral esta estratégia mais simples de fatorização tem-se revelado suficientemente estável na resolução dos problemas. Uma abordagem de pivotagem incremental para atualizar a fatorização LU está no entanto a ser implementada. Os exemplos a seguir ilustram a aplicação do processo.

6.3 Exemplos

Exemplo 6.2. Consideremos a equação diferencial de coeficientes polinomiais

$$\begin{cases} x\dfrac{d^2}{dx^2}y + 2\dfrac{d}{dx}y - (6x + 4x^3)y = 0, \quad x \in]0,\ 1[\\ y(0) = 1, \quad y'(0) = 0 \end{cases},$$

cuja solução exata é $y = \exp(x^2)$.

Aplicando o Algoritmo 2 ao Exemplo 6.2 atingimos a precisão da máquina para $n = 20$, tanto para o erro absoluto $|y - y_n^{(k)}|$, $k = 0(1)4$, $n = 5 + 5k$, como para a estimativa obtida pelos complementos de Schur, conforme apresentado na Figura 6.3.

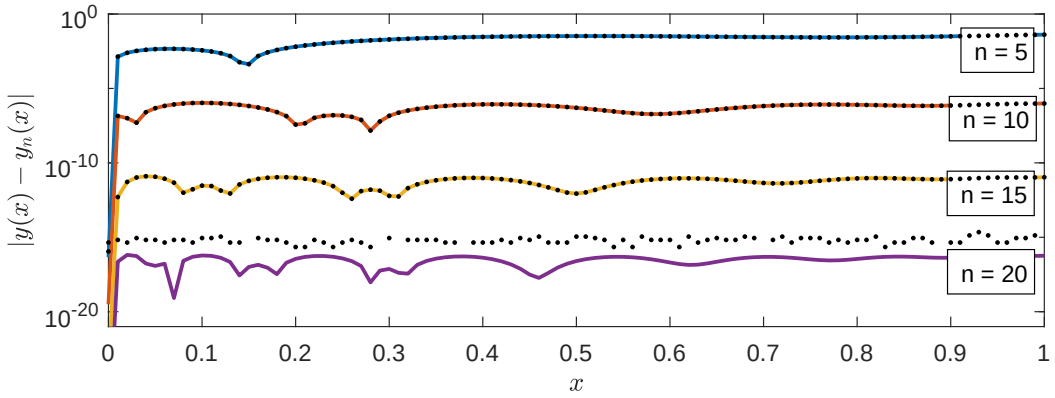

Figura 6.3: Erro absoluto $|y - y_n^{(k)}|$, $k = 0(1)4$, $n = 5+5k$, (linha pontilhada) e estimativa do erro $e_n^{(k)}$ (linha contínua) para o Exemplo 6.2.

Conforme podemos observar na Figura 6.3, as estimativas do erro seguem com elevada precisão os erros calculados a partir da solução exata. A exceção, observada com $n = 20$ ocorre apenas quando ambos os erros, estimado e observado, atingem a ordem de precisão da máquina.

Exemplo 6.3. Consideremos o problema diferencial

$$\begin{cases} (1 - 2\alpha x + \alpha^2)^2 \dfrac{d^2}{dx^2} y - \alpha(1 - 2\alpha x + \alpha^2) \dfrac{d}{dx} y - 2\alpha^2 y = 0, \quad x \in]-1,\ 1[\\ y(-1) = \dfrac{1}{1+\alpha}, \quad y(1) = \dfrac{1}{1-\alpha} \end{cases},$$

que tem como solução exata $y = \dfrac{1}{\sqrt{1 - 2\alpha x + \alpha^2}}$.

Este problema é interessante por ser conhecida a expressão exata dos coeficientes de uma expansão na sequência de Legendre: $y = \sum_{k \geq 0} \alpha^k P_k$,. Propomos utilizar a base de Legendre em nosso resultado e comparar o erro também nos coeficientes. Este mesmo problema foi apresentado em Trindade et al. (2016), onde, naquela formulação, o erro foi estimado a partir de apenas um termo da perturbação τ. Aqui, usamos todos os termos de τ_n, e portanto obtivemos melhores resultados como apresentado na Figura 6.4.

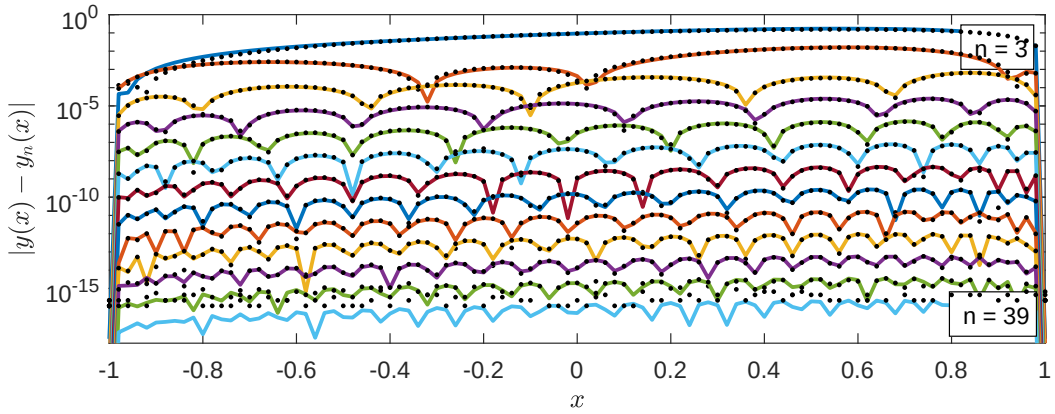

Figura 6.4: Erro absoluto $|y - y_n^{(k)}|$, $k = 0(1)12$, $n = 3 + 3k$, (linha pontilhada) e estimativa do erro $e_n^{(k)}$ (linha contínua) para o Exemplo 6.3 com $\alpha = 0,4$.

Finalmente, a Figura 6.5 apresenta o erro nos coeficientes exatos da expansão quando comparados com os aproximados pelo esquema proposto. Temos novamente a convergência obtida conforme para n crescente.

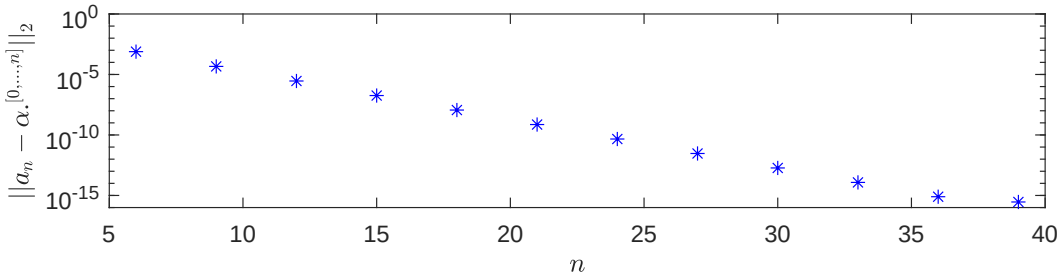

Figura 6.5: Norma quadrática do vetor erro nos coeficientes, $||a - a_n||_2$, com $n = 6(3)39$ para o Exemplo 6.3.

6.4 Conclusões e considerações

A grande vantagem em obter a solução aproximada de um problema fazendo uso do esquema iterativo com complementos de Schur proposto, está no facto de a aproximação ser calculada com a precisão requerida; o grau polinomial é adaptativo durante a resolução do sistema e portanto não acarreta custos adicionais. Este processo será de grande utilidade na `Tau Toolbox`, para a automatização da aproximação de coeficientes não polinomiais, conforme apresentado na Secção 7.3.

Capítulo 7

Aproximação de coeficientes não polinomiais

Sumário

Ao usarmos o método tau estamos condicionados, na sua formulação clássica, a que os coeficientes da função incógnita e das suas derivadas na equação diferencial sejam polinomiais, e portanto, nesta formulação, caso estes sejam funções não polinomiais devem ser primeiramente aproximados por polinómios. As alternativas que propomos para este problema são: (i) utilizar uma interpolação em polinómios ortogonais para aproximar os coeficientes da equação e em seguida utilizar a avaliação ortogonal apresentada na Secção 5.1, (ii) aproximar funções de matrizes, utilizando polinómios de Taylor, e (iii) usar o próprio método tau para aproximar estes coeficientes, nos casos em que os mesmos podem ser expressos como solução de um novo problema diferencial. Introduziremos as três abordagens propostas e em seguida apresentaremos resultados comparativos.

7.1 Aproximação de coeficientes não polinomiais por interpolação ortogonal

Numa interpolação polinomial, de Lagrange por exemplo, mesmo que aumentemos o número de pontos a precisão da aproximação nem sempre melhora. A principal questão que deve estar direcionada a este problema é se polinómios p_n que interpolam uma função contínua f em $n+1$ pontos são tais que

$$\lim_{n\to\infty} ||f - p_n|| = 0.$$

No caso particular de interpolar uma expressão do tipo $y = (1 + 16x^2)^{-1}$ Carl Runge observou que $\lim_{n\to\infty} ||f - p_n|| = \infty$ e a convergência acontece apenas em parte do intervalo, e este mau comportamento é devido aos valores dos pontos de interpolação, que por terem abcissas igualmente espaçadas tendem a fazer com que o polinómio interpolador apresente fortes oscilações perto das extremidades do intervalo (Gil et al., 2007).

Os resultados seguintes justificam a utilização de polinómios para aproximar funções contínuas e estabelecem a melhor escolha para uma interpolação polinomial.

Teorema 7.1. *(Teorema de aproximação de Weierstrass) Assumindo que f é uma função contínua no intervalo fechado $[a, b]$ e dado qualquer $\epsilon > 0$, existe um polinómio p_n com algum grau n tal que*

$$|f - p_n| < \epsilon, \quad para \quad a \leq x \leq b.$$

Demonstração. Ver Jeffreys and Jeffreys (1988) □

Teorema 7.2. *Os pontos ótimos para as fórmulas de quadratura gaussiana são precisamente os zeros dos polinómios ortogonais para o mesmo intervalo e função peso.*

Demonstração. Ver Gil et al. (2007). □

O Algoritmo de Golub and Welsch (1969) é um método para calcular os pontos e pesos para as fórmulas da quadratura de Gauss, e consiste em encontrar valores próprios e vetores próprios, como segue: Seja $\mathcal{Z} = [Z_0,\ Z_1,\ \ldots]$ uma sequência de polinómios ortogonais satisfazendo uma relação de recorrência de três termos de parâmetros α_i, β_i e γ_i

$$xZ_i = \alpha_i Z_{i+1} + \beta_i Z_i + \gamma_i Z_{i-1}, \quad i \geq 0, \quad Z_{-1} = 0, \quad Z_0 = 1,$$

e seja x_j um dos pontos da regra de quadratura, tal que $Z_{n+1}(x_j) = 0$, então podemos escrever $n + 1$ equações

$$\begin{cases} \alpha_0 Z_1(x_j) + \beta_0 Z_0(x_j) = x_j Z_0(x_j) \\ \alpha_1 Z_2(x_j) + \beta_1 Z_1(x_j) + \gamma_1 Z_0(x_j) = x_j Z_1(x_j) \\ \ldots \\ \beta_n Z_n(x_j) + \gamma_n Z_{n-1}(x_j) = x_j Z_n(x_j) \end{cases},$$

ou no formato matricial

$$\begin{bmatrix} \beta_0 & \alpha_0 & & & & \\ \gamma_1 & \beta_1 & \alpha_1 & & & \\ & \gamma_2 & \beta_2 & \alpha_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \gamma_{n-1} & \beta_{n-1} & \alpha_{n-1} \\ & & & & \gamma_n & \beta_n \end{bmatrix} \begin{bmatrix} Z_0(x_j) \\ Z_1(x_j) \\ Z_2(x_j) \\ \vdots \\ Z_{n-1}(x_j) \\ Z_n(x_j) \end{bmatrix} = x_j \begin{bmatrix} Z_0(x_j) \\ Z_1(x_j) \\ Z_2(x_j) \\ \vdots \\ Z_{n-1}(x_j) \\ Z_n(x_j) \end{bmatrix},$$

ou simplesmente $\mathsf{J}_{\mathcal{Z}}\mathsf{p}_{\mathcal{Z}}(x_j) = x_j\mathsf{p}_{\mathcal{Z}}(x_j)$, onde a matriz de Jacobi $\mathsf{J}_{\mathcal{Z}}$ contém os coeficientes da relação de recorrência para $\mathcal{Z}$ e $\mathsf{p}_{\mathcal{Z}}(x_j)$ é o vetor imagem de $\mathcal{Z}$ para x_j, donde segue que $Z_{n+1}(x_j) = 0$ se e somente se x_j é um valor próprio de $\mathsf{J}_{\mathcal{Z}}$, que é o mesmo que

dizer que os $n+1$ zeros de $Z_{n+1},\ x_0,\ x_1,\ \ldots,\ x_n$ podem ser obtidos calculando $\lambda = [x_0,\ \ldots,\ x_n],\ \det(\mathsf{J}_{\mathcal{Z}} - \lambda \mathsf{I}) = 0$ (Gil et al., 2007).

Existem métodos numéricos bastante eficientes para calcular os valores próprios e vetores próprios de matrizes simétricas tridiagonais. Nomeadamente o método QR implícito, o método da bisseção com iteração inversa, o método divide-and-conquer e o recente MRRR (Demmel, 1997). Em facto, por uma transformação de semelhança de diagonal, $\mathsf{J}_{\mathcal{Z}}$ pode ser reduzida a uma matriz simétrica tridiagonal

$$\begin{bmatrix} \beta_0 & \rho_0 & & & & \\ \rho_0 & \beta_1 & \rho_1 & & & \\ & \rho_1 & \beta_2 & \rho_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \rho_{n-2} & \beta_{n-1} & \rho_{n-1} \\ & & & & \rho_{n-1} & \beta_n \end{bmatrix},$$

com $\rho_i = \sqrt{\alpha_i \gamma_{i+1}},\ i = 0, \ldots, n-1$. Importante mencionar que $\rho_i \in \mathbb{R}$ uma vez que $\alpha_i \gamma_{i+1} = \left(\frac{k_i}{k_{i+1}}\right)^2 \frac{h_{i+1}}{h_i} > 0$, onde $k_i \in \mathbb{R}$ é o coeficiente principal de Z_i e $h_i = \|Z_i\|^2$.

Os pontos de Chebyshev podem ser explicitamente calculados, uma vez que são projeções no eixo x dos pontos igualmente espaçados no círculo (Trefethen, 2000), e portanto $n+1$ pontos no intervalo $[a,\ b]$ formam n ângulos constantes e iguais $\theta = 180/n$, conforme a figura 7.1. Assim, se $T_n,\ n \geq 0$ é um polinómio de Chebyshev, então pode ser escrito na forma

$$T_n = \cos\left(n \cos^{-1}(x)\right),$$

e portanto os pontos podem ser encontrados explicitamente por (já trasladados para o intervalo $[a,\ b]$)

$$x_i = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{(2i-1)\pi}{2n}\right), \quad i = 0(1)n.$$

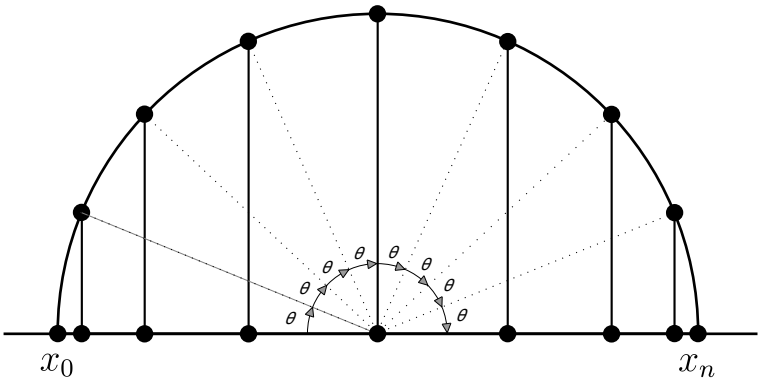

Figura 7.1: Representação dos pontos de Chebyshev.

Consideremos o caso de aproximar a expressão $y = \dfrac{1}{1+16x^2}$ em $n+1$ pontos igualmente espaçados e para $n+1$ pontos de Chebyshev, por um polinómio interpolador

$y_n = \sum_{i=0}^{n} a_i x^i$. Quando aumentamos o grau n o erro aumenta exponencialmente no caso de pontos igualmente espaçados, e encontramos o conhecido fenómeno de Runge (Figura 7.2 esq.), enquanto no caso da utilização dos pontos de Chebyshev o erro decai exponencialmente (Trefethen, 2000), (Figura 7.2 dir.).

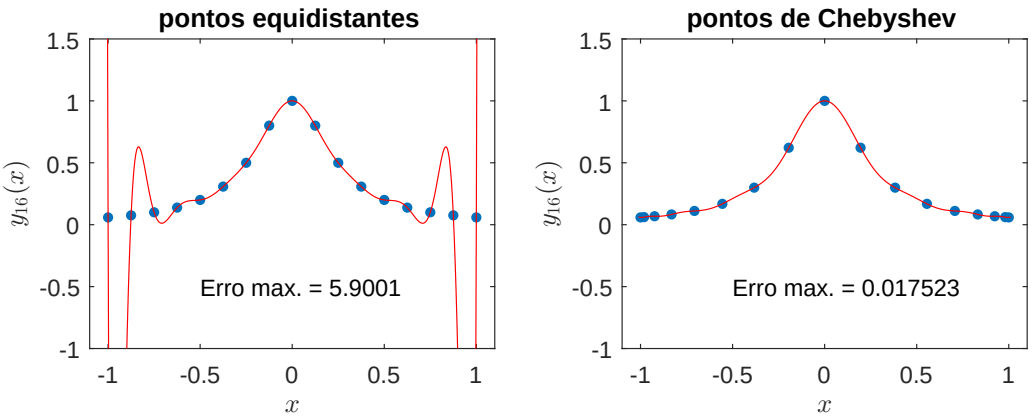

Figura 7.2: Interpolação para $y = \dfrac{1}{1+16x^2}$ com pontos igualmente espaçados e com os pontos de Chebyshev ($n = 16$).

Propomos a seguir as interpolações em base ortogonal para uma e duas variáveis, onde são usados os zeros dos polinómios ortogonais como abcissas. A interpolação em uma variável será útil nas aplicações do método tau tanto para o termo independente de um problema diferencial/integral como para tratar os coeficientes quando forem funcionais. Já a interpolação em duas variáveis será útil nas aproximações dos núcleos dos termos integrais de Fredholm e Volterra.

Funções de uma variável

Seja f uma função contínua no intervalo $[a,\ b]$ e $\mathcal{Z} = [Z_0,\ Z_1,\ \ldots]$ uma sequência de polinómios ortogonais no mesmo intervalo, assim, uma forma de calcular os coeficientes a_i do interpolador de f, consiste em impor que nos pontos da quadratura de Gauss $\mathsf{x} = [x_0,\ x_1,\ \ldots,\ x_n]$, esta igualdade seja exata, portanto, resolvendo o sistema de $n+1$ equações lineares

$$\sum_{i=0}^{n} a_i Z_i(x_j) = f(x_j),\ \ j = 0(1)n,$$

ou no formato matricial $\mathsf{Pa} = \mathsf{f}$, onde

$$\mathsf{P} = \begin{bmatrix} Z_0(x_0) & Z_1(x_0) & \ldots & Z_n(x_0) \\ Z_0(x_1) & Z_1(x_1) & \ldots & Z_n(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ Z_0(x_n) & Z_1(x_n) & \ldots & Z_n(x_n) \end{bmatrix}, \quad \mathsf{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}, \quad \text{e} \quad \mathsf{f} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}.$$

Uma vez obtido o polinómio $p_n \approx f$, e sendo este um coeficiente não polinomial na equação diferencial a resolver pelo método tau, então é tratado como

$$p_n(\mathsf{M}_{\mathcal{Z}}) = \sum_{i=0}^{n} a_i Z_i(\mathsf{M}_{\mathcal{Z}}).$$

Funções de duas variáveis

Seja $K(x,t)$ uma função contínua no domínio retangular $[a,\ b] \times [t_a,\ t_b]$, e sejam $\mathcal{Z}_x = [Z_{x0},\ Z_{x1},\ \ldots]$ e $\mathcal{Z}_t = [Z_{t0}(t),\ Z_{t1}(t),\ \ldots]$ sequências de polinómios ortogonais, respetivamente nos intervalos $[a,\ b]$ e $[t_a,\ t_b]$. Uma forma de calcular coeficientes a_{ij} do interpolador de K, consiste em impor que nos pontos (x_i, t_j), sendo x_i e t_j abcissas de quadratura de Gauss, esta igualdade seja exata, e desta forma os a_{ij} podem ser encontrados ao resolver o sistema de $(n+1)^2$ equações lineares

$$\sum_{i=0}^{n}\sum_{j=0}^{n} a_{ij} Z_{xi}(x_r) Z_{tj}(t_s) = K(x_r, t_s), \quad r,s = 0(1)n,$$

ou no formato matricial $\mathsf{Pa} = \mathsf{f}$, onde, denotando o produto $Z_{xi}(x_k)Z_{tj}(t_l)$ por p_{ij}^{kl} temos que

$$\mathsf{P} = \begin{bmatrix}
p_{0,0}^{0,0} & \cdots & p_{0,n}^{0,0} & p_{1,0}^{0,0} & \cdots & p_{1,n}^{0,0} & & p_{n,0}^{0,0} & \cdots & p_{n,n}^{0,0} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\
p_{0,0}^{0,n} & \cdots & p_{0,n}^{0,n} & p_{1,0}^{0,n} & \cdots & p_{1,n}^{0,n} & & p_{n,0}^{0,n} & \cdots & p_{n,n}^{0,n} \\
p_{0,0}^{1,0} & \cdots & p_{0,n}^{1,0} & p_{1,0}^{1,0} & \cdots & p_{1,n}^{1,0} & & p_{n,0}^{1,0} & \cdots & p_{n,n}^{1,0} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\
p_{0,0}^{1,n} & \cdots & p_{0,n}^{1,n} & p_{1,0}^{1,n} & \cdots & p_{1,n}^{1,n} & & p_{n,0}^{1,n} & \cdots & p_{n,n}^{1,n} \\
 & \vdots & & & \vdots & & \ddots & & \vdots & \\
p_{0,0}^{n,0} & \cdots & p_{0,n}^{n,0} & p_{1,0}^{n,0} & \cdots & p_{1,n}^{n,0} & & p_{n,0}^{n,0} & \cdots & p_{n,n}^{n,0} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\
p_{0,0}^{n,n} & \cdots & p_{0,n}^{n,n} & p_{1,0}^{n,n} & \cdots & p_{1,n}^{n,n} & & p_{n,0}^{n,n} & \cdots & p_{n,n}^{n,n}
\end{bmatrix},$$

$$\mathsf{f} = \Big[K(x_0,t_0),\ \ldots,\ K(x_0,t_n),\ K(x_1,t_0),\ \ldots,\ K(x_1,t_n),\ \ldots,\ K(x_n,t_0),\ \ldots,\ K(x_n,t_n)\Big]^T,$$

e

$$\mathsf{a} = \Big[a_{00},\ \ldots,\ a_{0n},\ a_{10},\ \ldots,\ a_{1n},\ \ldots,\ a_{n0},\ \ldots,\ a_{nn}\Big]^T. \tag{7.1}$$

Aplicação aos integrais de Fredholm e de Volterra

A interpolação polinomial bidimensional, introduzida na secção anterior, é útil no contexto do método tau, para generalizar a sua aplicação às equações integro-diferenciais com termos de Fredholm ou com termos de Volterra. No caso de o problema a resolver incluir algum destes termos, com um núcleo envolvendo uma função K não polinomial, esta função pode aproximar-se por um polinómio interpolador.

Assim, aproximando o núcleo $K(x,t)$ de um termo integral, e mudando o formato do vetor a em (7.1) para uma matriz

$$\mathsf{A}_{\mathcal{Z}} = [a_{i,j}]_{n+1\times n+1}$$

temos

$$K(x,t) \approx \sum_{i=0}^{n_x}\sum_{j=0}^{n_t} a_{ij} Z_{xi}(x) Z_{tj}(t) = \mathcal{Z}_x(x)\mathsf{A}_{\mathcal{Z}}\mathcal{Z}_t^T(t). \tag{7.2}$$

Esta expressão para o núcleo $K(x,t)$ pode projetar-se na base das potências, permitindo assim aplicar o Teorema (4.6), bastando para isso utilizar as matrizes de projecção nas duas variáveis, com a operação matricial $\mathsf{K} = \mathsf{V}_{\mathcal{Z}_x}^T \mathsf{K}_{\mathcal{Z}}^T \mathsf{V}_{\mathcal{Z}_t}$. No entanto esta operação é instável pois $\mathsf{V}_{\mathcal{Z}}$ contém elementos de ordens de grandeza muito distintas, com elementos que serão menosprezados nas operações aritméticas de precisão finitas, quando operados com elementos de ordem de grandeza muito superior. Desta forma, para aproveitar a precisão dos elementos de $\mathsf{A}_{\mathcal{Z}}$ em (7.2), e evitar esta operação matricial de mudança de base, a Proposição seguinte apresenta uma forma de aproximar os temos de integrais de Fredholm e Volterra.

Proposição 7.1. *Se* $K_*(x,t) \approx \sum_{i=0}^{n_x}\sum_{j=0}^{n_t} k_{ij}^{(*)} Z_i(x) Z_j(t)$, *então* (4.36) *e* (4.37) *do Teorema* (4.6) *se reescrevem, respetivamente, na forma*

$$\mathsf{S}^{(V)} = \sum_{i=0}^{n_x}\sum_{j=0}^{n_t} k_{ij}^{(1)} \Big(Z_i(\mathsf{M}_{\mathcal{Z}}) - \mathsf{e}_{i+1}\mathcal{Z}(a)\Big)\mathsf{O}_{\mathcal{Z}} Z_j(\mathsf{M}_{\mathcal{Z}}) \tag{7.3}$$

e

$$\mathsf{S}^{(F)} = \sum_{i=0}^{n_x}\sum_{j=0}^{n_t} k_{ij}^{(2)} \mathsf{e}_{i+1}\Big(\mathcal{Z}(b) - \mathcal{Z}(a)\Big)\mathsf{O}_{\mathcal{Z}} Z_j(\mathsf{M}_{\mathcal{Z}}). \tag{7.4}$$

Demonstração. Basta notar que, se na base das potências temos a representação matricial do operador integral como

$$\int^x x^i t^j y(t)dt = \mathcal{X}\mathsf{M}^i\mathsf{O}\mathsf{M}^j\mathsf{a}$$

então pelo Teorema 4.3 de mudança de base por transformação de semelhança podemos escrever

$$\begin{aligned}\int^x x^i j^j y(t)dt &= \mathcal{X}(\mathsf{V}_{\mathcal{Z}}\mathsf{M}_{\mathcal{Z}}\mathsf{V}_{\mathcal{Z}}^{-1})^i(\mathsf{V}_{\mathcal{Z}}\mathsf{O}_{\mathcal{Z}}\mathsf{V}_{\mathcal{Z}}^{-1})(\mathsf{V}_{\mathcal{Z}}\mathsf{M}_{\mathcal{Z}}\mathsf{V}_{\mathcal{Z}}^{-1})^j\mathsf{V}_{\mathcal{Z}}\mathsf{a}_{\mathcal{Z}}\\ &= \mathcal{X}\mathsf{V}_{\mathcal{Z}}\mathsf{M}_{\mathcal{Z}}^i\mathsf{V}_{\mathcal{Z}}^{-1}\mathsf{V}_{\mathcal{Z}}\mathsf{O}_{\mathcal{Z}}\mathsf{V}_{\mathcal{Z}}^{-1}\mathsf{V}_{\mathcal{Z}}\mathsf{M}_{\mathcal{Z}}^j\mathsf{V}_{\mathcal{Z}}^{-1}\mathsf{V}_{\mathcal{Z}}\mathsf{a}_{\mathcal{Z}}\\ &= \mathcal{Z}\mathsf{M}_{\mathcal{Z}}^i\mathsf{O}_{\mathcal{Z}}\mathsf{M}_{\mathcal{Z}}^j\mathsf{a}_{\mathcal{Z}},\end{aligned} \tag{7.6}$$

e então

$$\int^x Z_i(x)Z_j(t)y(t)dt = \mathcal{Z}Z_i(\mathsf{M}_{\mathcal{Z}})\mathsf{O}_{\mathcal{Z}}Z_j(\mathsf{M}_{\mathcal{Z}})\mathsf{a}_{\mathcal{Z}}. \tag{7.7}$$

□

Na equação (7.6) temos a tradução daquela integral em uma base ortogonal $\mathcal{Z}$, mas ainda com K em $\mathcal{X}$. Já em (7.7) toda a expressão se apresenta na base ortogonal $\mathcal{Z}$.

Exemplo 7.1. Consideremos como exemplo aproximar a solução da equação integro-diferencial de Volterra

$$\begin{cases} \dfrac{d}{dx}y + y - \displaystyle\int_0^x x(1+2x)\exp(t(x-t)y(t))dt = 2x+1 \\ y(0) = 1 \end{cases},$$

por um polinómio de Chebyshev de primeira espécie y_n.

São apresentados na Figura 7.3 os resíduos $F[y_n] - f$, onde F e f são respetivamente, o operador e o termo independente do Exemplo 7.1. Foi aplicada a Proposição 7.1 (resíduo em linha pontilhada) e a formulação convencional da base das potências (resíduo em linha contínua) para calcular o núcleo da integral de Volterra.

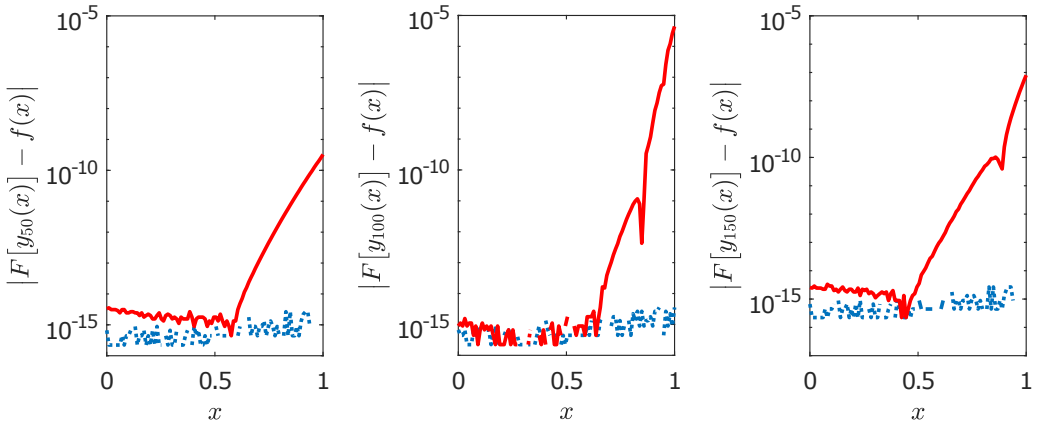

Figura 7.3: Resíduo para o Exemplo 7.1 com a Proposição 7.1 e com a formulação usual, para $n = 50,\ 100,\ 150$.

Ao aplicar o método tau e fazer F operar sobre a aproximação y_n, percebemos que utilizar a formulação apresentada na Proposição 7.1 faz com que o resíduo da solução esteja na precisão da máquina. O mesmo não ocorre na formulação usual, quando operamos as mudanças de base com as matrizes de projeção.

A Figura 7.4 ilustra a diferença entre as imagens de duas aproximações consecutivas, $|y_n - y_{n-1}|$, para $n = 50,\ 100,\ 150$, quando se opera com as matrizes de mudança de base. Percebemos que há uma diferença nestes resultados e que esta diferença oscila em torno de zero com amplitudes que, sendo decrescentes com n, aparentam decrescer muito lentamente ou até mesmo crescer. O mesmo não ocorre quando a Proposição é aplicada, e a diferença entre duas aproximações consecutivas é nula para $n = 50,\ 100,\ 150$.

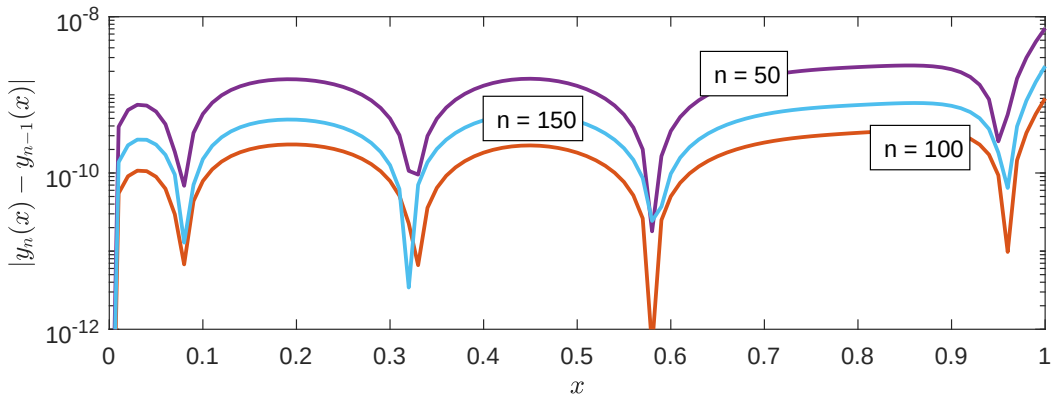

Figura 7.4: Diferença nas imagens de duas aproximações consecutivas para o Exemplo 7.1, com $n = 50,\ 100,\ 150$, resultante da aplicação do método tau e operando com as matrizes de mudança de bases.

Para a `Tau Toolbox`, as matrizes que traduzem os coeficientes não polinomiais aproximados por interpolação em base ortogonal foram implementadas como **`sini`**, **`sinhi`**, **`cosi`**, **`coshi`**, **`expi`** e **`logi`**.

7.2 Aproximação de coeficientes não polinomiais por funções de matrizes

A teoria de funções de matrizes, ocupa-se de funções $f : \mathsf{A} \to \mathsf{B},\ \mathsf{A}, \mathsf{B} \in \mathbb{C}^{n\times n}$, mas não no sentido usual $f(\mathsf{A}) = f(a_{ij}),\quad i, j = 1(1)n$, ou seja, avaliando elemento a elemento, mas sim como a expansão em série de Taylor destes argumentos matriciais. A definição de $f(\mathsf{A})$ pode vir apresentada como (i) a forma canónica de Jordan, (ii) interpolação polinomial ou (iii) teorema integral de Cauchy, que são definições que produzem funções matriciais primárias. Estas três formas são equivalentes (ver Higham (2008), Teorema 1.12). Considerando a abordagem (i): qualquer matriz $\mathsf{A} \in \mathbb{C}^{n\times n}$ pode ser expressa na forma canónica de Jordan

$$\mathsf{Z}^{-1}\mathsf{A}\mathsf{Z} = \mathsf{J} = \operatorname{diag}(\mathsf{J}_1,\ \ldots,\ \mathsf{J}_\mathsf{p}),\quad \mathsf{J}_\mathsf{k} = \mathsf{J}_\mathsf{k}(\lambda_\mathsf{k}) = \begin{bmatrix} \lambda_k & 1 & & \\ & \lambda_k & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_k \end{bmatrix} \in \mathbb{C}^{\mathsf{m_k}\times\mathsf{m_k}}, \tag{7.8}$$

onde Z é uma matriz não singular e $\sum_{i=1}^{p} m_i = n$. A matriz de Jordan J é única a menos da ordenação dos blocos $\mathsf{J_i}$, mas a matriz de transformação Z não é única.

Denotemos por $\lambda_1,\ \ldots,\ \lambda_s$ os distintos valores próprios de A e seja n_i a ordem do maior bloco de Jordan no qual λ_i aparece, e o qual é chamado de índice de λ_i.

Definição 7.1. *A função f é dita estar definida no espetro de A se existem os valores*

$$\frac{d^j}{dx^j} f(\lambda_i),\quad j = 0(1)n_i - 1,\quad i = 1(1)s,$$

e estes são chamados de valores da função f sobre o espetro de A.

A seguinte definição de $f(\mathsf{A})$ requer somente os valores de f sobre o espetro de A (Higham, 2008).

Definição 7.2. *Seja f definida no espetro de $\mathsf{A} \in \mathbb{C}^{n\times n}$ e tenha A uma representação conforme (7.8), então a função matricial via a forma canónica de Jordan é dada por*

$$f(\mathsf{A}) := \mathsf{Z} f(\mathsf{J}) \mathsf{Z}^{-1} = \mathsf{Z}\, diag\left[f(\mathsf{J}_k)\right] \mathsf{Z}^{-1},$$

onde

$$f(\mathsf{J}_k) := \begin{bmatrix} f(\lambda_k) & f'(\lambda_k) & \dots & \dfrac{f^{(m_k-1)}(\lambda_k)}{(m_k-1)!} \\ & f(\lambda_K) & \ddots & \vdots \\ & & \ddots & f'(\lambda_k) \\ & & & f(\lambda_k) \end{bmatrix}.$$

Em Higham (2008) são encontrados alguns resultados que garantem a validade de uma série de Taylor de uma função com matriz como argumento. O resultado seguinte trata da Convergência da série de Taylor de matriz.

Teorema 7.3. *Seja f uma função que tenha expansão em série de Taylor*

$$f = \sum_{k=0}^{\infty} a_k (x-\alpha)^k, \quad a_k = \frac{f^{(k)}(\alpha)}{k!} \tag{7.9}$$

com raio de convergência r. Se $\mathsf{A} \in \mathbb{C}^{n\times n}$, então $f(\mathsf{A})$ é dada por

$$f(\mathsf{A}) = \sum_{k=0}^{\infty} a_k (\mathsf{A} - \alpha \mathsf{I})^k$$

se e somente se cada um dos distintos valores próprios $\lambda_1, \dots, \lambda_s$ de A satisfaz uma das condições: (a) $|\lambda_i - \alpha| < r$ ou (b) $|\lambda_i - \alpha| = r$ e a série para $f^{(n_i-1)}(\lambda)$ (onde n_i é o índice de λ_i) é convergente no ponto $\lambda = \lambda_i, \quad i = 1(1)s$.

Demonstração. A Definição 7.2 é suficiente para provar que este teorema para um bloco de Jordan, $\mathsf{A} = \mathsf{J}(\lambda) = \lambda \mathsf{I} + \mathsf{N} \in \mathbb{C}^{n\times n}$, onde N é estritamente triangular superior. Seja $f_m = \sum_{k=0}^{m} a_k (x-\alpha)^k$. Temos que

$$\begin{aligned} f_m(\mathsf{J}(\lambda)) &= \sum_{k=0}^{m} a_k \left[(\lambda-\alpha)\mathsf{I} + \mathsf{N}\right]^k \\ &= \sum_{k=0}^{m} a_k \sum_{i=0}^{k} \binom{k}{i} (\lambda-\alpha)^{k-i} \mathsf{N}^i \\ &= \sum_{i=0}^{m} \mathsf{N}^i \sum_{k=i}^{m} a_k \binom{k}{i} (\lambda-\alpha)^{k-i} \\ &= \sum_{i=0}^{m} \frac{\mathsf{N}^i}{i!} \sum_{k=i}^{m} a_k k(k-1)\dots(k-i+1)(\lambda-\alpha)^{k-i} \end{aligned}$$

$$= \sum_{i=0}^{m} \frac{\mathsf{N}^i}{i!} f_m^{(i)}(\lambda) = \sum_{i=0}^{\min(m,n-1)} \frac{\mathsf{N}^i}{i!} f_m^{(i)}(\lambda).$$

Evidentemente, $\lim_{m\to\infty} f_m(\mathsf{J}(\lambda))$ existe se e somente se $\lim_{m\to\infty} f_m^{(i)}(\lambda)$ existe para $i = 1(1)n-1$, que portanto é essencialmente o caso apresentado (b), uma vez que a série para o termo diferenciado de f, termo a termo, $n_i - 1$ vezes converge para λ, em seguida, o mesmo acontece com a série diferenciada j vezes para $j = 0(1)n_i - 1$. O caso (a) segue do resultado padrão na análise complexa que um termo da série de potência diferenciada termo por termo converge dentro do raio de convergência da série original. □

Algumas das mais usuais séries de Taylor para funções matriciais são

$$\sin(\mathsf{M}_{\mathcal{Z}}) = \sum_{i=0}^{\infty} \frac{(-1)^i \mathsf{M}_{\mathcal{Z}}^{2i+1}}{(2i+1)!}, \quad \sinh(\mathsf{M}_{\mathcal{Z}}) = \sum_{i=0}^{\infty} \frac{\mathsf{M}_{\mathcal{Z}}^{2i+1}}{(2i+1)!},$$

$$\cos(\mathsf{M}_{\mathcal{Z}}) = \sum_{i=0}^{\infty} \frac{(-1)^i \mathsf{M}_{\mathcal{Z}}^{2i}}{(2i)!}, \quad \cosh(\mathsf{M}_{\mathcal{Z}}) = \sum_{i=0}^{\infty} \frac{\mathsf{M}_{\mathcal{Z}}^{2i}}{(2i)!},$$

$$\exp(\mathsf{M}_{\mathcal{Z}}) = \sum_{i=0}^{\infty} \frac{\mathsf{M}_{\mathcal{Z}}^{i}}{i!} \quad \text{e} \quad \log(\mathsf{M}_{\mathcal{Z}}) = \sum_{i=0}^{\infty} \frac{(-1)^i (\mathsf{M}_{\mathcal{Z}} - \mathbf{I})^{i+1}}{i+1}.$$

Estas séries servirão para aproximar os coeficientes não polinomiais nas aplicações do método tau, onde estaremos sujeitos a dois tipos de erros: o de truncamento destas séries e o de arredondamento na aritmética de ponto flutuante. Os erros de truncamento são delimitados no seguinte resultado (Higham, 2008):

Teorema 7.4. *Supondo que f tenha uma expansão em séries de Taylor como (7.9) com raio de convergência r, assim, se $\mathsf{A} \in \mathbb{C}^{n\times n}$ com $\rho(\mathsf{A} - \alpha\mathsf{I}) < r$, então para qualquer norma matricial, o limite do erro no truncamento da série de Taylor é*

$$\left\| f(\mathsf{A}) - \sum_{k=0}^{s-1} a_k (\mathsf{A} - \alpha\mathsf{I}) \right\| \leq \frac{1}{s!} \max_{0\leq t\leq 1} \left\| (\mathsf{A} - \alpha\mathsf{I})^s f^{(s)}\Big(\alpha I + t(\mathsf{A} - \alpha\mathsf{I})\Big) \right\|.$$

Demonstração. Ver Mathias (1993) Corolário 2. □

Para a `Tau Toolbox`, as matrizes que traduzem os coeficientes não polinomiais aproximados com funções de matrizes foram implementadas como **`sinm`**, **`sinhm`**, **`cosm`**, **`coshm`**, **`expm`** e **`logm`**.

7.3 Aproximação de coeficientes não polinomiais pelo método tau

Dependendo da funcional que é coeficiente em uma equação, esta pode ser aproximada usando previamente o próprio método tau, e portanto expressando-a como uma combinação linear de polinómios em uma determinada base. Se esta aproximação

for boa o suficiente, i.e. com a precisão da máquina, podemos usa-la na equação para aproximar tal coeficiente não polinomial. Como ilustração, temos as funcionais y, onde y é da forma $\sin(x)$, $\cos(x)$, $\sinh(x)$, $\cosh(x)$, $\exp(x)$ ou $\ln(x)$, podem ser escritas como sendo a solução de um problema diferencial e portanto aproximadas pelo método tau:

$$\sin(x) \quad \text{é solução de} \quad \begin{cases} \dfrac{d^2}{dx^2}y + y = 0, & x \in]a,\ b[\\ y(a) = \sin(a), & y'(b) = \cos(b) \end{cases},$$

$$\sinh(x) \quad \text{é solução de} \quad \begin{cases} \dfrac{d^2}{dx^2}y - y = 0, & x \in]a,\ b[\\ y(a) = \sinh(a), & y'(b) = \cosh(b) \end{cases},$$

$$\cos(x) \quad \text{é solução de} \quad \begin{cases} \dfrac{d^2}{dx^2}y + y = 0, & x \in]a,\ b[\\ y(a) = \cos(a), & y'(b) = -\sin(b) \end{cases},$$

$$\cosh(x) \quad \text{é solução de} \quad \begin{cases} \dfrac{d^2}{dx^2}y - y = 0, & x \in]a,\ b[\\ y(a) = \cosh(a), & y'(b) = \sinh(b) \end{cases},$$

$$\exp(x) \quad \text{é solução de} \quad \begin{cases} \dfrac{d}{dx}y - y = 0, & x \in]a,\ b[\\ y(a) = \exp(a) \end{cases}$$

e

$$\log(x) \quad \text{é solução de} \quad \begin{cases} x^2\dfrac{d}{dx}y = x, & x \in]a,\ b[\in \mathbb{R}_+^* \\ y(a) = \ln(a) \end{cases}.$$

São mostrados na Tabela 7.1 alguns resultados que expressam estas funcionais em termos de um problema diferencial, e os erros para as bases de Chebyshev e Legendre para as aproximações de ordem 10, 20 e 30. Quanto aos intervalos $[a,\ b]$, são todos iguais a $[1,\ 5]$, por simples padronização e exemplificação.

y	$\max \lvert y - y_{10} \rvert$	$\max \lvert y - y_{20} \rvert$	$\max \lvert y - y_{30} \rvert$
$\sin(x)$	$6,5 \cdot 10^{-7}(\mathcal{T})$	$1,6 \cdot 10^{-15}(\mathcal{T})$	$1,8 \cdot 10^{-15}(\mathcal{T})$
	$3,1 \cdot 10^{-7}(\mathcal{P})$	$1,9 \cdot 10^{-15}(\mathcal{P})$	$1,3 \cdot 10^{-15}(\mathcal{P})$
$\sinh(x)$	$3,7 \cdot 10^{-5}(\mathcal{T})$	$4,2 \cdot 10^{-14}(\mathcal{T})$	$2,8 \cdot 10^{-14}(\mathcal{T})$
	$2,1 \cdot 10^{-5}(\mathcal{P})$	$2,8 \cdot 10^{-14}(\mathcal{P})$	$2,8 \cdot 10^{-14}(\mathcal{P})$
$\cos(x)$	$5,0 \cdot 10^{-6}(\mathcal{T})$	$9,9 \cdot 10^{-16}(\mathcal{T})$	$7,7 \cdot 10^{-16}(\mathcal{T})$
	$1,8 \cdot 10^{-6}(\mathcal{P})$	$1,4 \cdot 10^{-15}(\mathcal{P})$	$1,2 \cdot 10^{-15}(\mathcal{P})$
$\cosh(x)$	$3,7 \cdot 10^{-5}(\mathcal{T})$	$2,8 \cdot 10^{-14}(\mathcal{T})$	$2,8 \cdot 10^{-14}(\mathcal{T})$
	$2,1 \cdot 10^{-5}(\mathcal{P})$	$5,6 \cdot 10^{-14}(\mathcal{P})$	$5,6 \cdot 10^{-14}(\mathcal{P})$
$\exp(x)$	$1,8 \cdot 10^{-4}(\mathcal{T})$	$5,6 \cdot 10^{-14}(\mathcal{T})$	$5,6 \cdot 10^{-14}(\mathcal{T})$
	$1,7 \cdot 10^{-5}(\mathcal{P})$	$8,5 \cdot 10^{-14}(\mathcal{P})$	$8,5 \cdot 10^{-14}(\mathcal{P})$
$\ln(x)$	$5,7 \cdot 10^{-5}(\mathcal{T})$	$1,3 \cdot 10^{-9}(\mathcal{T})$	$5,7 \cdot 10^{-14}(\mathcal{T})$
	$3,4 \cdot 10^{-5}(\mathcal{P})$	$1,0 \cdot 10^{-9}(\mathcal{P})$	$4,6 \cdot 10^{-14}(\mathcal{P})$

Tabela 7.1: Erro máximo da aproximação tau para equações diferenciais de soluções conhecidas, para as bases de Chebyshev e Legendre.

Uma vez que estamos interessados em aproximar estes coeficientes não polinomiais com a precisão da máquina, independentemente do grau, fazemos uso do esquema iterativo baseado em complementos de Schur, conforme mostrado na Secção 6. As funções criadas para esta abordagem são **sint**, **sinht**, **cost**, **cosht**, **expt** e **logt**, e que também sobrescrevem as funções nativas do MATLAB `sin`, `sinh`, `cos`, `cosh`, `exp` e `log`.

7.4 Comparação das abordagens: exemplos

Mostramos nas Tabelas 7.2 - 7.7, os resultados das normas das diferenças entre as funções que usam interpolação em base ortogonal, método tau e funções de matrizes para aproximar alguns coeficientes não polinomiais. Os resultados foram obtidos para três intervalos: [1, 5], [1, 10] e [1, 15].

Norma	[1, 5]	[1, 10]	[1, 15]
$\lvert\lvert \mathbf{sini}(\mathsf{M}_{\mathcal{T}}) - \mathbf{sinm}(\mathsf{M}_{\mathcal{T}}) \rvert\rvert$	$1,21 \cdot 10^{-14}$	$3,04 \cdot 10^{-14}$	$4,36 \cdot 10^{-14}$
$\lvert\lvert \mathbf{sint}(\mathsf{M}_{\mathcal{T}}) - \mathbf{sinm}(\mathsf{M}_{\mathcal{T}}) \rvert\rvert$	$1,20 \cdot 10^{-14}$	$3,12 \cdot 10^{-14}$	$3,24 \cdot 10^{-14}$
$\lvert\lvert \mathbf{sint}(\mathsf{M}_{\mathcal{T}}) - \mathbf{sini}(\mathsf{M}_{\mathcal{T}}) \rvert\rvert$	$7,30 \cdot 10^{-16}$	$3,33 \cdot 10^{-15}$	$1,36 \cdot 10^{-14}$
$\lvert\lvert \mathbf{sini}(\mathsf{M}_{\mathcal{P}}) - \mathbf{sinm}(\mathsf{M}_{\mathcal{P}}) \rvert\rvert$	$1,42 \cdot 10^{-14}$	$4,62 \cdot 10^{-14}$	$3,53 \cdot 10^{-14}$
$\lvert\lvert \mathbf{sint}(\mathsf{M}_{\mathcal{P}}) - \mathbf{sinm}(\mathsf{M}_{\mathcal{P}}) \rvert\rvert$	$1,38 \cdot 10^{-14}$	$4,60 \cdot 10^{-14}$	$3,90 \cdot 10^{-14}$
$\lvert\lvert \mathbf{sint}(\mathsf{M}_{\mathcal{P}}) - \mathbf{sini}(\mathsf{M}_{\mathcal{P}}) \rvert\rvert$	$1,24 \cdot 10^{-15}$	$2,88 \cdot 10^{-15}$	$2,96 \cdot 10^{-14}$

Tabela 7.2: Comparativos entre as normas das funções que usam interpolação ortogonal, método tau e funções de matrizes, para aproximar o seno como coeficiente não polinomial.

Norma	$[1 \quad 5]$	$[1 \quad 10]$	$[1 \quad 15]$
$\|\|\texttt{sinhi}(\mathsf{M}_{\mathcal{T}}) - \texttt{sinhm}(\mathsf{M}_{\mathcal{T}})\|\|$	$1,25 \cdot 10^{-12}$	$3,48 \cdot 10^{-10}$	$1,01 \cdot 10^{-7}$
$\|\|\texttt{sinht}(\mathsf{M}_{\mathcal{T}}) - \texttt{sinhm}(\mathsf{M}_{\mathcal{T}})\|\|$	$1,25 \cdot 10^{-12}$	$3,39 \cdot 10^{-10}$	$1,00 \cdot 10^{-7}$
$\|\|\texttt{sinht}(\mathsf{M}_{\mathcal{T}}) - \texttt{sinhi}(\mathsf{M}_{\mathcal{T}})\|\|$	$3,06 \cdot 10^{-14}$	$1,48 \cdot 10^{-11}$	$2,44 \cdot 10^{-9}$
$\|\|\texttt{sinhi}(\mathsf{M}_{\mathcal{P}}) - \texttt{sinhm}(\mathsf{M}_{\mathcal{P}})\|\|$	$2,00 \cdot 10^{-12}$	$5,11 \cdot 10^{-10}$	$6,00 \cdot 10^{-8}$
$\|\|\texttt{sinht}(\mathsf{M}_{\mathcal{P}}) - \texttt{sinhm}(\mathsf{M}_{\mathcal{P}})\|\|$	$2,03 \cdot 10^{-12}$	$5,11 \cdot 10^{-10}$	$6,15 \cdot 10^{-8}$
$\|\|\texttt{sinht}(\mathsf{M}_{\mathcal{P}}) - \texttt{sinhi}(\mathsf{M}_{\mathcal{P}})\|\|$	$6,35 \cdot 10^{-14}$	$2,13 \cdot 10^{-11}$	$4,43 \cdot 10^{-9}$

Tabela 7.3: Comparativos entre as normas das funções que usam interpolação ortogonal, método tau e funções de matrizes, para aproximar o seno hiperbólico como coeficiente não polinomial.

Norma	$[1, \ 5]$	$[1, \ 10]$	$[1, \ 15]$
$\|\|\texttt{cosi}(\mathsf{M}_{\mathcal{T}}) - \texttt{cosm}(\mathsf{M}_{\mathcal{T}})\|\|$	$1,60 \cdot 10^{-14}$	$1,83 \cdot 10^{-14}$	$5,36 \cdot 10^{-14}$
$\|\|\texttt{cost}(\mathsf{M}_{\mathcal{T}}) - \texttt{cosm}(\mathsf{M}_{\mathcal{T}})\|\|$	$1,57 \cdot 10^{-14}$	$2,39 \cdot 10^{-14}$	$5,34 \cdot 10^{-14}$
$\|\|\texttt{cost}(\mathsf{M}_{\mathcal{T}}) - \texttt{cosi}(\mathsf{M}_{\mathcal{T}})\|\|$	$7,55 \cdot 10^{-16}$	$1,03 \cdot 10^{-14}$	$3,36 \cdot 10^{-15}$
$\|\|\texttt{cosi}(\mathsf{M}_{\mathcal{P}}) - \texttt{cosm}(\mathsf{M}_{\mathcal{P}})\|\|$	$2,52 \cdot 10^{-14}$	$2,50 \cdot 10^{-14}$	$4,97 \cdot 10^{-14}$
$\|\|\texttt{cost}(\mathsf{M}_{\mathcal{P}}) - \texttt{cosm}(\mathsf{M}_{\mathcal{P}})\|\|$	$2,51 \cdot 10^{-14}$	$2,53 \cdot 10^{-14}$	$4,32 \cdot 10^{-14}$
$\|\|\texttt{cost}(\mathsf{M}_{\mathcal{P}}) - \texttt{cosi}(\mathsf{M}_{\mathcal{P}})\|\|$	$1,18 \cdot 10^{-15}$	$2,72 \cdot 10^{-15}$	$1,28 \cdot 10^{-14}$

Tabela 7.4: Comparativos entre as normas das funções que usam interpolação ortogonal, método tau e funções de matrizes, para aproximar o cosseno como coeficiente não polinomial.

Norma	$[1, \ 5]$	$[1, \ 10]$	$[1, \ 15]$
$\|\|\texttt{coshi}(\mathsf{M}_{\mathcal{T}}) - \texttt{coshm}(\mathsf{M}_{\mathcal{T}})\|\|$	$1,28 \cdot 10^{-12}$	$3,46 \cdot 10^{-10}$	$1,01 \cdot 10^{-7}$
$\|\|\texttt{cosht}(\mathsf{M}_{\mathcal{T}}) - \texttt{coshm}(\mathsf{M}_{\mathcal{T}})\|\|$	$1,27 \cdot 10^{-12}$	$3,42 \cdot 10^{-10}$	$1,01 \cdot 10^{-7}$
$\|\|\texttt{cosht}(\mathsf{M}_{\mathcal{T}}) - \texttt{coshi}(\mathsf{M}_{\mathcal{T}})\|\|$	$3,33 \cdot 10^{-14}$	$1,08 \cdot 10^{-11}$	$2,28 \cdot 10^{-9}$
$\|\|\texttt{coshi}(\mathsf{M}_{\mathcal{P}}) - \texttt{coshm}(\mathsf{M}_{\mathcal{P}})\|\|$	$2,02 \cdot 10^{-12}$	$5,12 \cdot 10^{-10}$	$5,99 \cdot 10^{-8}$
$\|\|\texttt{cosht}(\mathsf{M}_{\mathcal{P}}) - \texttt{coshm}(\mathsf{M}_{\mathcal{P}})\|\|$	$2,04 \cdot 10^{-12}$	$5,16 \cdot 10^{-10}$	$6,01 \cdot 10^{-8}$
$\|\|\texttt{cosht}(\mathsf{M}_{\mathcal{P}}) - \texttt{coshi}(\mathsf{M}_{\mathcal{P}})\|\|$	$5,68 \cdot 10^{-14}$	$2,11 \cdot 10^{-11}$	$5,05 \cdot 10^{-9}$

Tabela 7.5: Comparativos entre as normas das funções que usam interpolação ortogonal, método tau e funções de matrizes, para aproximar o cosseno hiperbólico como coeficiente não polinomial.

Norma	$[1,\ 5]$	$[1,\ 10]$	$[1,\ 15]$
$\|\|\texttt{expi}(\mathsf{M}_{\mathcal{T}}) - \texttt{expm}(\mathsf{M}_{\mathcal{T}})\|\|$	$7,21 \cdot 10^{-13}$	$1,38 \cdot 10^{-10}$	$3,14 \cdot 10^{-8}$
$\|\|\texttt{expt}(\mathsf{M}_{\mathcal{T}}) - \texttt{expm}(\mathsf{M}_{\mathcal{T}})\|\|$	$6,94 \cdot 10^{-13}$	$1,40 \cdot 10^{-10}$	$3,15 \cdot 10^{-8}$
$\|\|\texttt{expt}(\mathsf{M}_{\mathcal{T}}) - \texttt{expi}(\mathsf{M}_{\mathcal{T}})\|\|$	$8,85 \cdot 10^{-14}$	$2,66 \cdot 10^{-11}$	$4,98 \cdot 10^{-9}$
$\|\|\texttt{expi}(\mathsf{M}_{\mathcal{P}}) - \texttt{expm}(\mathsf{M}_{\mathcal{P}})\|\|$	$5,92 \cdot 10^{-13}$	$1,73 \cdot 10^{-10}$	$1,95 \cdot 10^{-8}$
$\|\|\texttt{expt}(\mathsf{M}_{\mathcal{P}}) - \texttt{expm}(\mathsf{M}_{\mathcal{P}})\|\|$	$6,18 \cdot 10^{-13}$	$1,64 \cdot 10^{-10}$	$1,41 \cdot 10^{-8}$
$\|\|\texttt{expt}(\mathsf{M}_{\mathcal{P}}) - \texttt{expi}(\mathsf{M}_{\mathcal{P}})\|\|$	$1,37 \cdot 10^{-13}$	$4,00 \cdot 10^{-11}$	$9,80 \cdot 10^{-9}$

Tabela 7.6: Comparativos entre as normas das funções que usam interpolação ortogonal, método tau e funções de matrizes, para aproximar a exponencial como coeficiente não polinomial.

Norma	$[1,\ 5]$	$[1,\ 10]$	$[1,\ 15]$
$\|\|\texttt{logi}(\mathsf{M}_{\mathcal{T}}) - \texttt{logm}(\mathsf{M}_{\mathcal{T}})\|\|$	$9,58 \cdot 10^{-15}$	$1,25 \cdot 10^{-14}$	$1,59 \cdot 10^{-14}$
$\|\|\texttt{logt}(\mathsf{M}_{\mathcal{T}}) - \texttt{logm}(\mathsf{M}_{\mathcal{T}})\|\|$	$9,98 \cdot 10^{-15}$	$1,20 \cdot 10^{-14}$	$1,46 \cdot 10^{-14}$
$\|\|\texttt{logt}(\mathsf{M}_{\mathcal{T}}) - \texttt{logi}(\mathsf{M}_{\mathcal{T}})\|\|$	$9,06 \cdot 10^{-16}$	$1,95 \cdot 10^{-15}$	$5,83 \cdot 10^{-15}$
$\|\|\texttt{logi}(\mathsf{M}_{\mathcal{P}}) - \texttt{logm}(\mathsf{M}_{\mathcal{P}})\|\|$	$1,21 \cdot 10^{-14}$	$1,47 \cdot 10^{-14}$	$1,69 \cdot 10^{-14}$
$\|\|\texttt{logt}(\mathsf{M}_{\mathcal{P}}) - \texttt{logm}(\mathsf{M}_{\mathcal{P}})\|\|$	$1,25 \cdot 10^{-14}$	$1,51 \cdot 10^{-14}$	$1,64 \cdot 10^{-14}$
$\|\|\texttt{logt}(\mathsf{M}_{\mathcal{P}}) - \texttt{logi}(\mathsf{M}_{\mathcal{P}})\|\|$	$1,32 \cdot 10^{-15}$	$3,38 \cdot 10^{-15}$	$3,09 \cdot 10^{-15}$

Tabela 7.7: Comparativos entre as normas das funções que usam interpolação ortogonal, método tau e funções de matrizes, para aproximar o logaritmo como coeficiente não polinomial.

Das Tabelas 7.2 - 7.7, percebemos que as aproximações com o método tau e com interpolação estão mais próximas entre si do que com a utilização das funções de matrizes, em todas as funções implementadas. Podemos ainda averiguar qual das três aproximações produz melhor resultado em termos de aplicações em um problema concreto, conforme o Exemplo a seguir.

Exemplo 7.2. Consideremos a equação diferencial de 10ª ordem com coeficientes não polinomiais

$$\begin{cases} \dfrac{d^{10}}{dx^{10}}y + \cosh(x)\dfrac{d^8}{dx^8}y + x^2\dfrac{d^6}{dx^6}y + x^4\dfrac{d^4}{dx^4}y + \cos(x)\dfrac{d^2}{dx^2}y + x^2y = 0 \quad x \in]-1,1[\\ y(-1) = y(1) = 0, \quad y'(-1) = y'(1) = 1, \quad y^{(k)}(\pm 1) = 0, \quad 2 \leq k \leq 4 \end{cases}.$$

Este problema foi apresentado em Olver and Townsend (2013) e nos é interessante por ter os coeficientes não polinomiais $\cosh(x)$ e $\cos(x)$. Em Trindade et al. (2016) os coeficientes não polinomiais foram aproximados pela *Chebfun* (Trefethen, 1994, 2013). Aqui, utilizamos as três formulações apresentadas anteriormente para aproximar $\cosh(x)$ e $\cos(x)$. A Figura 7.5 apresenta o resíduo, para $n = 50$, nos três casos.

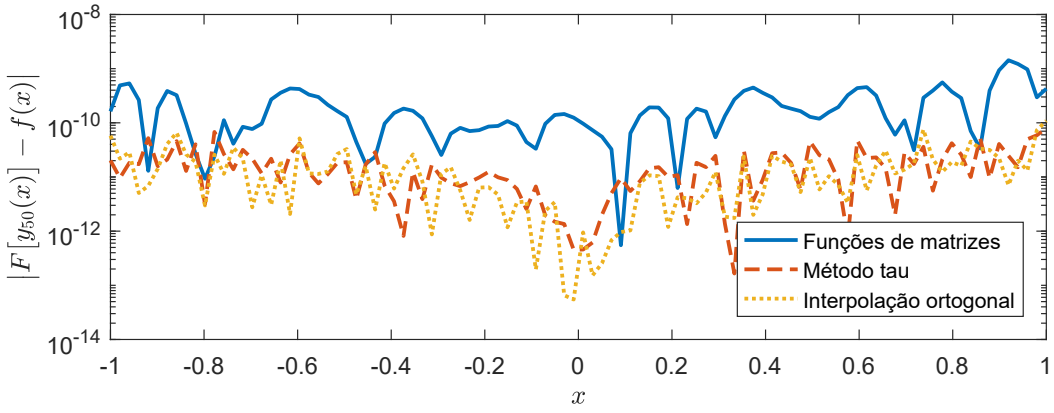

Figura 7.5: Resíduo para o Exemplo 7.2, com $n = 50$.

Conforme o resultado apresentado na Figura 7.5, o método tau e a interpolação em base ortogonal mostram-se ligeiramente mais precisos em termos do resíduo $|F[y_{50}(x)] - f(x)|$ do que quando comparados com as funções de matrizes.

7.5 Conclusões e considerações

Três tipos de aproximações para os coeficientes não polinomiais em problemas integro-deferenciais foram propostas no contexto do método tau: junto com o uso de interpolação polinomial em base ortogonal e funções de matrizes, o método tau foi explorado também para este fim. A precisão alcançada por essas três aproximações é capaz de alcançar o erro na precisão da máquina. Este recurso estende a aplicabilidade do método tau e é um aprimoramento crucial para propor uma ferramenta numérica para resolver problemas integro-diferenciais. Além disso, é acoplado o mecanismo automático baseado em complementos de Schur para fornecer essas aproximações dentro do método tau, como um problema auxiliar, prescrevendo a precisão necessária.

Com este Capítulo encerramos a Parte II da Tese. Na Parte seguinte apresentamos os resultados relacionados com as implementações dos contributos de estabilidade aqui propostos, num conjunto de exemplos da aproximação da solução de diversos problemas integro-diferenciais lineares e não lineares.

Parte III

Tau Toolbox: uma biblioteca de software matemático para o método tau

Capítulo 8

`Tau Toolbox` - Versão 1.0

Sumário

A `Tau Toolbox` é a face visível de um projeto que ambiciona (i) agregar todas as contribuições existentes na literatura sobre o método tau, (ii) potenciar o uso do método através do desenvolvimento de algoritmos mais estáveis e (iii) oferecer implementações eficientes. A biblioteca, nesta versão 1.0, é capaz de fornecer aproximações polinomiais no sentido tau à solução de vários problemas integro-diferenciais: lineares, não lineares, com condições iniciais e/ou fronteira. Contrariamente a outras bibliotecas que trabalham com uma base ortogonal, em geral de Chebyshev, a `Tau Toolbox` disponibiliza a possibilidade de utilização de várias bases ortogonais de polinómios: destacando-se os clássicos Chebyshev (I e II), Legendre, Hermite, Laguerre e Bessel. As mais de 250 funções disponibilizadas estão documentadas e catalogadas de acordo com o tipo de funcionalidade e abrangência, neste último caso distinguidas entre funções básicas, intermédias ou para utilizadores avançados,

Com esta biblioteca de software matemático, não especialistas podem ter acesso fácil a um método espetral e usufruir das suas propriedades para facilmente resolver os seus problemas. Por sua vez, especialistas podem analisar novos problemas explorando o vasto leque de rotinas disponibilizadas para desenvolver novas extensões de aplicação.

8.1 Aspetos da implementação da **Tau Toolbox**

A Tau Toolbox é construída usando a Programação orientada a objetos (OOP), que é uma abordagem de programação formal, para desenvolver aplicações complexas de computação técnica, que combina dados e ações associadas (métodos) em estruturas lógicas (objetos). Esta abordagem melhora a capacidade de gerir a complexidade, particularmente importante no desenvolvimento e na manutenção de grandes aplicações e estruturas de dados de software.

8.1.1 Técnicas para otimizar a performance

A Tau Toolbox é construída tendo em consideração boas práticas de programação MATLAB . As rotinas são estruturadas de forma a (i) usar funções em vez de scripts; (ii) criar funções locais (funções dentro de funções, e que portanto não constituem um arquivo específico); (iii) usar programação modular repartindo cálculos mais complexos em rotinas pequenas que executem apenas determinadas operações. De seguida ilustram-se algumas das técnicas implementadas para potenciar a performance dos algoritmos.

Pré-alocação

Pré-alocar uma variável consiste em determiná-la previamente, em termos de propriedades e elementos neutros, antes de iniciar a imposição dos seus valores em ciclos. Sem pré-alocação, quando são utilizados ciclos, como `for` ou `while`, para calcular os elementos de matrizes, incrementalmente aumentam o tamanho da estrutura de dados, prejudicando o desempenho e o uso de memória. Como um breve exemplo, consideremos as instruções abaixo, executadas com e sem pré-alocar a matriz A:

```
n = [10^2, 10^3, 10^4];                                    % Valores de n.
tempo = [0, 0, 0];                                       % Valores de tempo.
for exe = 1:3                                  % Para todos os valores de n.
    clear t A i j                                          % Limpa valores.
    t = cputime;                                           % Tempo da CPU.
    % A = 0;                                             % Sem pré-alocação.
    A = zeros(n(exe));                                   % Com pré-alocação.
    for i = 2:n(exe)
        for j = 2:n(exe)
            A(i, j) = A(i-1, j-1) + 1;
        end
    end
    tempo(exe) = cputime - t;                                        % Tempo.
end
```

A Tabela 8.1 apresenta os tempos obtidos com e sem pré-alocação prévia de memória.

n	**Tempo sem pré-alocação**	**Tempo com pré-alocação**
10^2	$0s$	$0s$
10^3	$0,65625s$	$0,078125s$
10^4	$718,234375s$	$7,375s$

Tabela 8.1: Tempos de computação com uso ou não de pré-alocação prévia de memória.

Este exemplo trata de um simples cálculo por recorrência dos elementos de uma matriz A de dimensão $n \times n$, $n = [10^2,\ 10^3,\ 10^4]$. Como resultado, percebemos que para dimensão 10^2 o tempo é impercetivel em ambos os casos, já para as outras dimensões, ao usar pré-alocação os tempos representam 11.9% e 1.02% dos tempos sem pré-alocação, respetivamente para $n = 10^3$ e $n = 10^4$. Esse exemplo ilustra a eficiência da pré-alocação de variáveis vetoriais de grande dimensão nos cálculos por recorrência, frequentes nas aplicações do método tau.

Vetorização

Ainda utilizando o mesmo exemplo abordado acima, consideremos estas mesmas instruções com pré-alocação, mas alteremos desta vez os dois ciclos internos `for`, ou seja,

```
for i = 2:n(exe)
    for j = 2:n(exe)
        A(i, j) = A(i-1, j-1) + 1;
    end
end
```

eliminando-os e escrevendo simplesmente no formato vetorizado

```
A(2:n(exe), 2:n(exe)) = A(1:n(exe)-1, 1:n(exe)-1) + 1;
```

e comparemos os resultados. Os tempos agora são de $0s$, $0,03125s$ e $1,859375s$, respetivamente para $n = 10^2$, $n = 10^3$ e $n = 10^4$. Estes resultados mostram que, para o exemplo em questão, a versão vetorizada representa aproximadamente 4.76% e 25.21% dos tempos da versão não vetorizada com pré-alocação, respetivamente para $n = 10^3$ e $n = 10^4$.

8.1.2 Classes e objetos na **Tau Toolbox**

A `Tau Toolbox` é implementada utilizando classes que descrevem objetos com propriedades e métodos que sobre ele atuam, e a vantagem em adotar este tipo de programação está no facto de permitir sobrescrever as funções nativas do `MATLAB` para que estas se comportem como as definimos. Por exemplo, conforme introduzido no Capítulo 4, ao encontrarmos numa equação diferencial o termo

$$\frac{d^k}{dx^k}y,$$

sabemos que na transcrição deste termo para o formato matricial, o mesmo irá traduzir-se como N^k, e assim sobrescrevemos a função nativa **diff** para que esta nos retorne o pretendido. Caso os argumentos de entrada não sejam um objeto previamente definido, então a função continua a comportar-se como inicialmente estabelecida. Para além disto, sabemos que em termos analíticos é garantida a comutatividade

$$(x^2+x)\times\frac{d}{dx}y=\frac{d}{dx}y\times(x^2+x),$$

todavia, em termos da formulação operacional do método tau, por se tratar de operações com matrizes, teremos

$$(\mathsf{M}^2+\mathsf{M})\mathsf{N}\neq\mathsf{N}(\mathsf{M}^2+\mathsf{M}).$$

Este problema é contornado com o uso de OOP, de forma que o usuário tanto pode escrever `(x^2+x)*diff(y)` como **diff**`(y)*(x^2+x)` que o processamento respeitará

$$(\mathsf{M}^2+\mathsf{M})\mathsf{N},$$

preservando a propriedade de comutatividade para a operação de multiplicação **mtimes**, que também é sobrescrita justamente para atender a este tipo de requisitos.

Para ilustrar a importância e o impacto da programação orientada a objetos na `Tau Toolbox`, consideremos aproximar a solução da equação integro-diferencial não linear

$$\cos(x)\frac{d^2}{dx^2}y+(x^2+x)\frac{d}{dx}y+y^2-x^4\int_0^x y(t)dt=\sin(x),$$

que pode ser linearizada para

$$\cos(x)\frac{d^2}{dx^2}y^{(k)}+(x^2+x)\frac{d}{dx}y^{(k)}+2y^{(k-1)}y^{(k)}-x^4\int_0^x y^{(k)}(t)dt=\sin(x)+y^{(k-1)^2}. \quad (8.1)$$

A criação das classes:

- **itau**, para o tratamento da variável independente como coeficiente no problema integro-diferencial: $\cos(x)$, (x^2+x) e $-x^4$;
- **dtau**, para o tratamento da variável dependente no problema integro-diferencial: $\frac{d^2}{dx^2}y^{(k)}$, e $y^{(k)}$;
- **ctau**, para o tratamento dos coeficientes da aproximação tau no problema integro-diferencial: $y^{(k-1)}$; e
- **rtau**, para o tratamento da variável independente quando pertencente ao membro direito do problema integro diferencial: $\sin(x)$,

proporcionam tratar o operador da equação (8.1) como

$$\underbrace{\overbrace{\cos(x)}^{\texttt{itau}}\underbrace{\frac{d^2}{dx^2}y^{(k)}}_{\texttt{dtau}}}_{\underbrace{\sum_{i=0}^{\lambda} a_i Z_i(\mathsf{M}_{\mathcal{Z}})\mathsf{N}_{\mathcal{Z}}^2}_{\texttt{dtau}}} + \underbrace{\overbrace{(x^2+x)}^{\texttt{itau}}\underbrace{\frac{d}{dx}y^{(k)}}_{\texttt{dtau}}}_{\underbrace{(\mathsf{M}_{\mathcal{Z}}^2+\mathsf{M}_{\mathcal{Z}})\mathsf{N}_{\mathcal{Z}}}_{\texttt{dtau}}} + \underbrace{\overbrace{2y^{(k-1)}}^{\texttt{ctau}}\underbrace{y^{(k)}}_{\texttt{dtau}}}_{\underbrace{2\sum_{i=0}^{\partial y} y_i Z_i(\mathsf{M}_{\mathcal{Z}})}_{\texttt{dtau}}} - \underbrace{\overbrace{x^4}^{\texttt{itau}}\underbrace{\int_0^x K(x,t)y^{(k)}(t)dt}_{\texttt{dtau}}}_{\underbrace{\mathsf{A}}_{\texttt{dtau}}}}_{\texttt{dtau}}, \tag{8.2}$$

onde $\mathsf{A} = \mathsf{M}_{\mathcal{Z}}^4 \sum_{i=0}^{n_x}\sum_{j=0}^{n_t} k_{ij}\Big(Z_i(\mathsf{M}_{\mathcal{Z}}) - \mathsf{e}_{i+1}\mathcal{Z}(a)\Big)\mathsf{O}_{\mathcal{Z}} Z_j(\mathsf{M}_{\mathcal{Z}})$, e o lado direito como

$$\underbrace{\underbrace{\sin(x)}_{\texttt{rtau}} + \underbrace{y^{(k-1)^2}}_{\texttt{ctau}}}_{\texttt{ctau}}. \tag{8.3}$$

Desta forma, na `Tau Toolbox`, operações envolvendo dois objetos dão origem a um dos dois objetos, alterando apenas suas propriedades. O objetivo é reduzir o operador de um problema integro-diferencial em um objeto **`dtau`** e o lado direito do mesmo problema em um objeto **`ctau`**. A complexidade da formulação (8.2)-(8.3) é simplificada ao utilizador final devido à modulação da implementação da `Tau Toolbox`: Em (8.2), os a_i são automaticamente aproximados pelo método tau usando o esquema iterativo com complementos de Schur proposto no Capítulo 6, além disso são internamente aplicadas as Proposições 5.1, 5.4, 5.5, 5.6, 5.7 e 7.1. Em (8.3), novamente o método tau com complementos de Schur é usado para aproximar o termo $\sin(x)$, e $y^{(k-1)^2}$ é automaticamente calculado usando a Proposição 5.9 dos coeficientes de linearização.

Para atender as necessidades da implementação do método tau na `Tau Toolbox`, as classes criadas sobrescrevem as funções nativas do `MATLAB`: **`plus`**, **`uplus`**, **`minus`**, **`uminus`**, **`mtimes`**, **`mpower`**, **`diff`**, **`int`**, **`sin`**, **`sinh`**, **`cos`**, **`cosh`**, **`exp`**, **`log`** e **`sqrt`**. A Secção seguinte explora com maior riqueza de detalhes a utilização das rotinas.

8.2 Exemplos de utilização da `Tau Toolbox`

A função `[x, y] = tau(basis, domain, n)` cria dois objetos: `x` como uma variável independente tau e `y` como uma variável dependente tau. Estes objetos carregam as propriedades: base (e.g. ChebyshevT, ChebyshevU, LegendreP, HermiteH, LaguerreL ou BesselY), domínio, ordem da aproximação e matriz associada à operação.

O seguinte exemplo cria os objetos para trabalhar com uma aproximação polinomial de Chebyshev de primeira espécie de ordem 4 no domínio $[-1,\ 1]$:

```
>> [x, y] = tau('ChebyshevT', [-1 1], 5)

    x =                              y =

    itau with properties:            dtau with properties:

       basis: 'ChebyshevT'              basis: 'ChebyshevT'
      domain: [-1 1]                   domain: [-1 1]
           n: 5                             n: 5
         mat: [5x5 double]                mat: [5x5 double]
```

Por defeito, o mesmo resultado é obtido apenas com `[x, y] = tau`.

Ao escrevermos $x^2 - 4x^5$, e uma vez que x é um objeto **itau**, esta expressão será interpretada como a matriz $\mathsf{M}_{\mathcal{T}}^2 - 4\mathsf{M}_{\mathcal{T}}^5$, que tem dimensão 5×5, e onde $\mathcal{T}$ é a base de Chebyshev de primeira espécie e $\mathsf{M}_{\mathcal{T}}$ é a matriz definida em (5.9) para esta base:

```
>> x^2-4*x^5
   ans.mat =

   0.5000   -1.2500    0.2500   -0.6250         0
  -2.5000    0.7500   -1.8750    0.2500   -0.6250
   0.5000   -1.8750    0.5000   -1.2500    0.2500
  -1.2500    0.2500   -1.2500    0.5000   -0.6250
        0   -0.6250    0.2500   -0.6250    0.2500
```

Além disto, $(x^2 - 4x^5)\dfrac{d^3}{dx^3}y$ é interpretado como $(\mathsf{M}_{\mathcal{T}}^2 - 4\mathsf{M}_{\mathcal{T}}^5)\mathsf{N}_{\mathcal{T}}^3$:

```
>> (x^2-4*x^5)*diff(y,3)
    ans.mat =

    0     0     0    12  -240
    0     0     0   -60   144
    0     0     0    12  -360
    0     0     0   -30    48
    0     0     0     0  -144
```

Com estes objetos torna-se mais fácil a tradução de um problema diferencial para a formulação operacional do método tau. Os exemplos seguintes ilustram o cálculo da solução aproximada de um conjunto de problemas lineares e não lineares, usando a `Tau Toolbox`.

8.2.1 Equações diferenciais ordinárias

Nesta Subsecção, apresentaremos um conjunto de problemas diferenciais lineares e não lineares, alguns de mais fácil solução, outros mais complexos. Apresentamos ainda alguns problemas *stiff*. Em todos eles, temos a intenção de explorar e apresentar as funcionalidades da `Tau Toolbox` .

8.2.1.1 Problemas lineares

Exemplo 8.1. Consideremos o problema diferencial linear

$$\begin{cases} \dfrac{d^2}{dx^2}y + y = 0, \quad x \in]0,\ 5[\\ y(0) = 0, \quad y'(0) = 1 \end{cases},$$

cuja solução exata é $y = \sin(x)$.

Solução. Para aproximar a solução deste problema usamos apenas duas funções `Tau Toolbox`:

```
% Create tau objects.
[x, y] = tau('ChebyshevT', [0 5], 20);

% Compute the ode approximate solution.
a = tausolver(x, y, 'diff(y,2)+y=0', {'y(0)=0';'y''(0)=1'});
```

A aproximação encontrada é um polinómio de grau 19 projetado na base de Chebyshev de primeira espécie (ortogonal em $[0,\ 5]$). O vetor a contém os 20 coeficientes da aproximação tau (Figura 8.1).

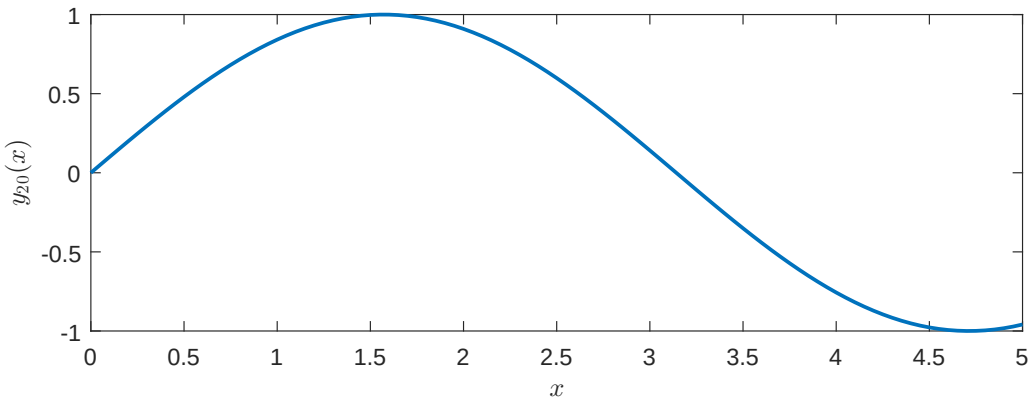

Figura 8.1: Solução aproximada $y_n = \sum_{i=0}^{n-1} a_{n,i}T_i = \mathcal{T}_n\mathsf{a}_n$, $n = 20$.

Mudar a base para projetar a aproximação é simples. Por exemplo, para o caso de Legendre basta apenas invocar `[x, y] = tau('LegendreP', [0 5], 20);`. A Figura 8.2 mostra a distância $|y_{20} - y_{19}|$ com as bases de Legendre e de Chebyshev de primeira espécie.

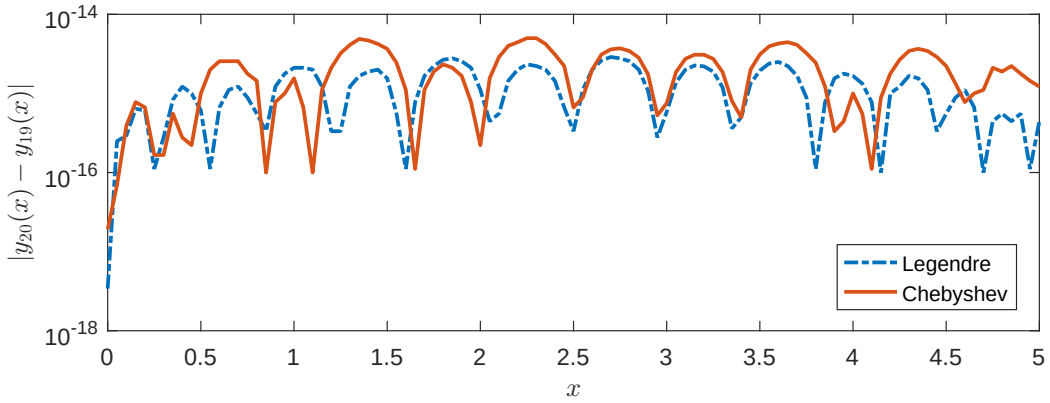

Figura 8.2: $|y_{20} - y_{19}|$ com as bases de Chebyshev de primeira espécie e de Legendre.

Num nível mais detalhado, o usuário pode parametrizar a função **tausolver** especificando certos parâmetros. Por exemplo, se a solução exata é conhecida (neste caso $y = \sin(x)$), isto pode ser informado com o parâmetro `'exact_solution'`, e a função retornará um gráfico do erro absoluto $|y - y_{20}|$. A Figura 8.3 mostra este erro e o formato da matriz T_{20} (por ativação do parâmetro `'spy'`).

```
% Create tau objects.
[x, y] = tau('ChebyshevT', [0 5], 20);

% Compute the ode approximate solution.
a = tausolver(x, y, 'diff(y,2)+y=0', {'y(0)=0';'y''(0)=1'}, ...
                                     'exact_solution', 'sin(x)', 'spy', 1);
```

O problema diferencial foi traduzido para o sistema de equações lineares algébricas $\mathsf{T}_n\mathsf{a}_n = \mathsf{b}_n$, onde

$$\mathsf{T}_n = \begin{bmatrix} \mathsf{C}_\mathcal{T}(1:\nu, 1:n) \\ \mathsf{D}_\mathcal{T}(1:n-\nu, 1:n) \end{bmatrix}, \quad \mathsf{b}_n = [0,\, 1,\, \underbrace{0,\, \ldots,\, 0}_{n-\nu}]^T,$$

com $\mathsf{D}_\mathcal{T} = \mathsf{N}_\mathcal{T}^2 + \mathsf{I}$, $\mathsf{C}_\mathcal{T} = [c_{i,j}]$, $c_{1,j} = T_j(0)$, $c_{2,j} = T_j'(0)$, $j = 0(1)n-1$, $n = 20$ e $\nu = 2$.

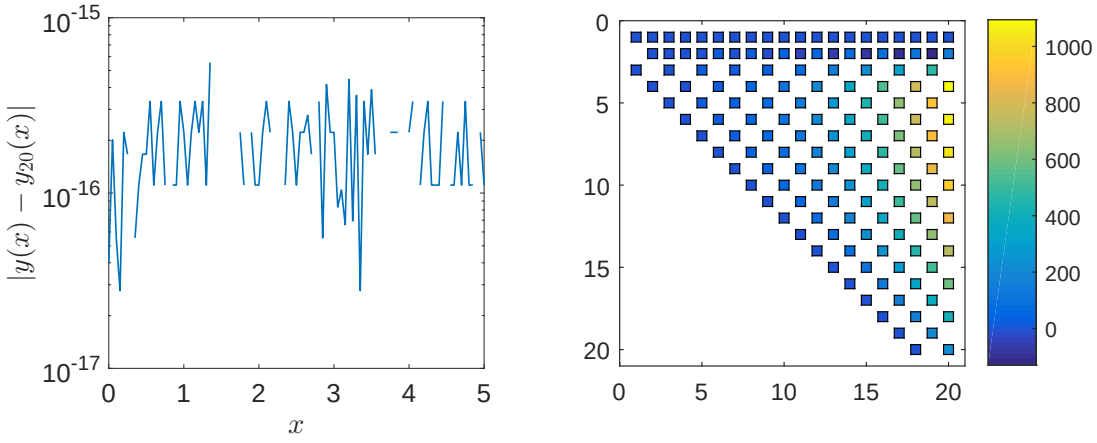

Figura 8.3: Erro e spy com a base de Chebyshev de primeira espécie e $n = 20$.

Exemplo 8.2. Seja o problema diferencial

$$\begin{cases} (x^2+1)\dfrac{d^3}{dx^3}y - (x^2+3x)\dfrac{d^2}{dx^2}y + 5x\dfrac{d}{dx}y - 5y = 60x^2 - 10 \quad x \in]-1,\ 1[\\ y(-1) = 4, \quad y'(1) = 2, \quad y''(0) = 0 \end{cases},$$

cuja solução exata é polinomial, $y = x^5 - 3x + 2$.

Solução. A solução do problema pode ser aproximada com `Tau Toolbox` como:

```
% Create tau objects.
[x, y] = tau('ChebyshevT', [-1 1], 6);

% Compute the ode approximate solution.
a = tausolver(x, y, ...
          ['(x^2+1)*diff(y,3)-(x^2+3*x)*diff(y,2)+5*x*diff(y)-5*y', ...
          ' = 60*x^2-10'], ...
          {'y(-1)=4';'y''(1)=2';'y''''(0)=0'}, ...
          'spy', 1, 'exact_solution', 'x^5-3*x+2');
```

O vetor solução a obtido, com $n = 6$, foi $\mathsf{a} = [2,\ -2.375,\ 0,\ 0.3125,\ 0,\ 0.0625]$, o qual corresponde à forma de ChebyshevT de $y = 2T_0 - \frac{19}{8}T_1 + \frac{5}{16}T_3 + \frac{1}{16}T_5 = 2 - 3x + x^5$, que é justamente a solução exata. Isto ilustra uma propriedade de método tau, de que com um n suficientemente grande (pelo menos igual ao grau da solução), a solução exata é sempre recuperada caso esta seja um polinómio.

A implementação da função **`tausolver`** é estável pois a solução exata é recuperada mesmo considerando valores de n superiores ao grau da solução (polinomial) exata. A Figura 8.4 (esquerda) mostra os erros absolutos $|\mathsf{a} - \mathsf{a}_n|$ observados nos coeficientes obtidos com a função **`tausolver`** referentes à aproximação a_n para vários valores de n. Observamos que para $n = 8,\ 64,\ 512,\ 4096$ o erro absoluto entre os coeficientes aproximados e os exatos tende a ser cada vez menor. Na Figura 8.4 (direita) é mostrado o spy de T_{4096}, do qual percebemos a estabilidade da implementação, pois a parte triangular inferior é nula, não recebendo perturbações decorrentes de operações em virgula flutuante.

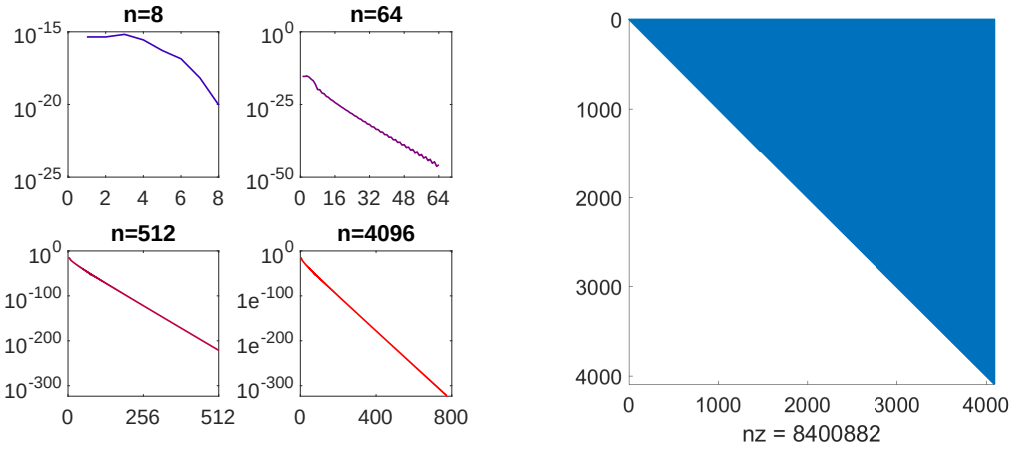

Figura 8.4: Erro nos coeficientes ($|\mathsf{a} - \mathsf{a}_n|$, $n = 8,\ 64,\ 512,\ 4096$) e spy para o Exemplo 8.2.

Exemplo 8.3. Este problema serve para ilustrar a aplicação da `Tau Toolbox` para sistemas de equações diferenciais. A equação diferencial anterior pode ser traduzida por um sistema de três equações diferenciais de primeira ordem, como

$$\begin{cases} y_2 - \dfrac{d}{dx}y_1 = 0 \\ y_3 - \dfrac{d}{dx}y_2 = 0 \\ (x^2+1)\dfrac{d}{dx}y_3 - (x^2+3x)y_3 + 5xy_2 - 5y_1 = 60x^2 - 10 \\ y_1(-1) = 4, \quad y_2(1) = 2, \quad y_3(0) = 0 \end{cases}.$$

Solução. A função **`tausolver`** aproxima também sistemas de equações diferenciais:

```
% Create tau objects.
[x, y] = tau('ChebyshevT', [-1 1], 6);

% Compute the approximate solution of the system of ode's.
a = tausolver(x, y, {'y2-1*diff(y1)=0'; ...
              'y3-1*diff(y2)=0'; ...
              ['(x^2+1)*diff(y3)-(x^2+3*x)*diff(y2)+5*x*y2-5*y1=', ...
              '60*x^2-10']}, ...
              {'y1(-1)=4';'y2(1)=2';'y3(0)=0'}, ...
              'exact_solution', {'x^5-3*x+2';'5*x^4-3';'20*x^3'}, 'spy', 1);
```

A forma de $\mathsf{T}_\mathcal{T}$ é apresentada na Figura 8.5 (direita) e a solução é mostrada na Figura 8.5 (esquerda).

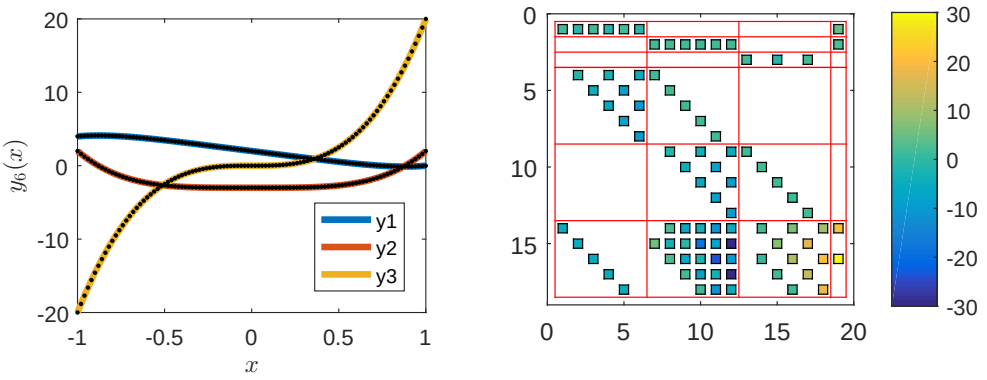

Figura 8.5: Solução e spy para o Exemplo 8.3.

O Exemplo 8.3 foi traduzido em um sistema de equações lineares algébricas $[\mathsf{C}_\mathcal{T};\ \mathsf{D}_\mathcal{T}]\mathsf{a}_\mathcal{T} = \mathsf{b}_\mathcal{T}$, onde

$$\mathsf{D}_\mathcal{T} = \begin{bmatrix} -\mathsf{N}_\mathcal{T} & \mathsf{I} & \mathbf{0} \\ \mathbf{0} & -\mathsf{N}_\mathcal{T} & \mathsf{I} \\ -5\mathsf{I} & 5\mathsf{M}_\mathcal{T}\mathsf{I} - (\mathsf{M}_\mathcal{T}^2 + 3\mathsf{M}_\mathcal{T})\mathsf{N}_\mathcal{T} & (\mathsf{M}_\mathcal{T}^2 + \mathsf{I})\mathsf{N}_\mathcal{T} \end{bmatrix}$$

traduz o operador diferencial,

$$\mathsf{C}_\mathcal{T} = \begin{cases} c_{1,i} = T_i(-1) \\ c_{2,n+i} = T_i(1) \\ c_{3,2n+i} = T_i(0) \end{cases}, \quad i = 0(1)n-1$$

traduz as condições de contorno e

$$\mathsf{b}_\mathcal{T} = [\underbrace{4,\ 2,\ 0}_{\nu=3},\ \underbrace{0,\ \ldots,\ 0}_{2(n-1)},\ \mathsf{f}_\mathcal{T}]^T.$$

A solução do sistema retorna

$$y \approx \sum_{i=0}^{n-1} a_{\mathcal{T}i} T_i, \quad y' \approx \sum_{i=0}^{n-1} a_{\mathcal{T}i+n} T_i \quad \text{e} \quad y'' \approx \sum_{i=0}^{n-1} a_{\mathcal{T}i+2n} T_i.$$

Apresentados estes exemplos iniciais, passamos agora a abordar problemas cujas soluções sejam mais difíceis de aproximar, nomeadamente, com pontos não suaves no domínio, conforme os exemplos seguintes.

Exemplo 8.4. Seja o problema diferencial

$$\begin{cases} \lambda \dfrac{d^2}{dx^2} y + 2x \dfrac{d}{dx} y = 0,\ x \in]-1,\ 1[,\ \lambda \in \mathbb{R}^+ \\ y(-1) = \operatorname{erf}\left(-\dfrac{1}{\sqrt{\lambda}}\right),\ y(1) = \operatorname{erf}\left(\dfrac{1}{\sqrt{\lambda}}\right) \end{cases},$$

cuja solução exata é

$$y = \operatorname{erf}\left(\frac{x}{\sqrt{\lambda}}\right), \ \operatorname{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-w^2} dw.$$

Embora a função erf esteja definida para $z \in \mathbb{C}$, quando restringida a $x \in \mathbb{R}$ temos que esta é uma boa aproximante da função sinal, para valores de λ suficientemente pequenos, ou seja

$$\lim_{\lambda \to 0^+} \operatorname{erf}\left(\frac{x}{\sqrt{\lambda}}\right) = \operatorname{sign}(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{cases} . \tag{8.4}$$

Este exemplo é interessante para avaliar o comportamento da aproximação tau na vizinhança de $x = 0$, para diferentes graus de polinómios e valores λ. Para tal, o Código 8.1 apresenta a utilização da `Tau Toolbox` para aproximar o problema.

Código MATLAB 8.1. `Tau Toolbox` para o Exemplo 8.4

```
% Create tau objects.
[x, y] = tau('LegendreP', [-1 1], 51);

% Specify parameter, problem, conditions and exact solution.
lambda = 1e-2;
ode = [num2str(lambda),'*diff(y, 2)+2*x*diff(y)=0'];
solution = ['erf(x/sqrt(',num2str(lambda),'))'];
conditions = {['y(-1)=erf(-1/sqrt(',num2str(lambda),'))']; ...
              ['y(1)=erf(1/sqrt(',num2str(lambda),'))']};

% Solve the problem.
a = tausolver(x, y, ode, conditions, 'exact_solution', solution, ...
    'pieces', 2);
```

Variando a ordem n (de aproximação polinomial) no Código 8.1 conseguimos obter os resultados apresentados nas Figuras 8.6 (aproximações e zoom das aproximações na zona crítica) e 8.7 (erro para vários valores de n).

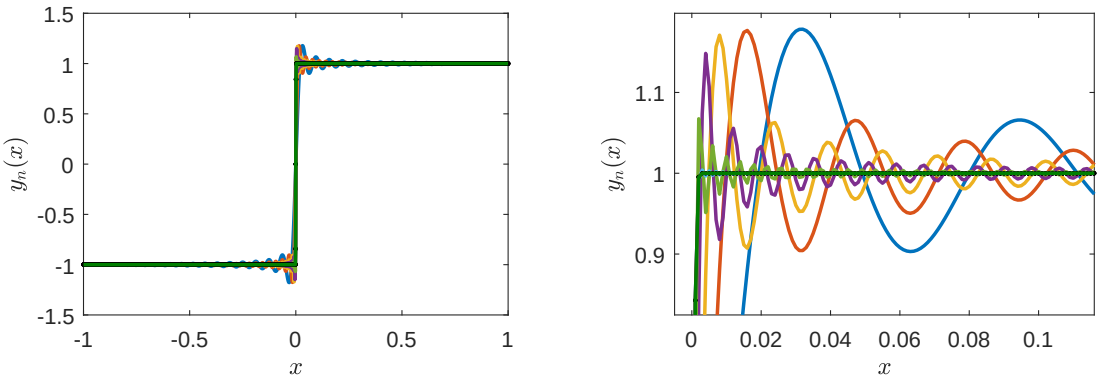

Figura 8.6: Aproximação pela `Tau Toolbox` para o exemplo 8.4, com $n = 100 \cdot 2^k$, $k = 0(1)7$ e $\lambda = 10^{-6}$.

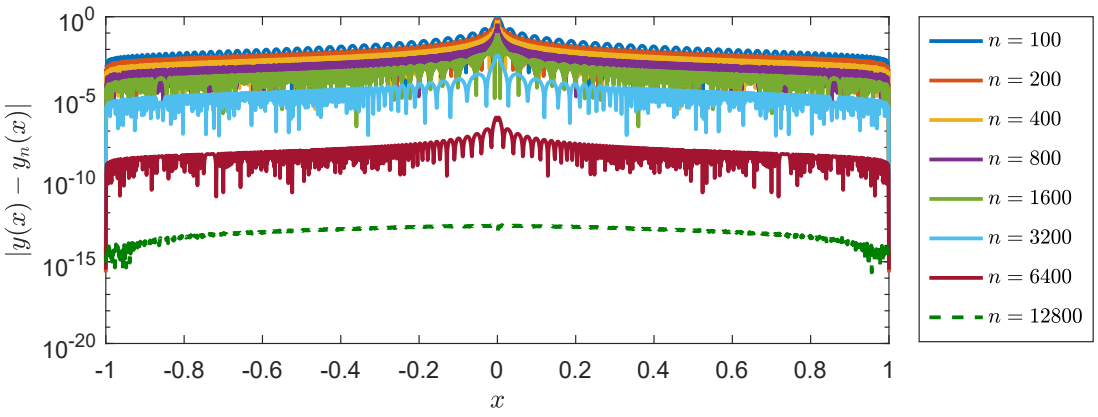

Figura 8.7: Erro para o Exemplo 8.4, com $n = 100 \cdot 2^k$, $k = 0(1)7$ e $\lambda = 10^{-6}$.

Na Figura 8.6 (esquerda) evidencia-se o fenómeno de Gibbs para valores de x próximos de zero. Aumentando o grau n da aproximação polinomial o fenómeno é amenizado, conforme o detalhe no zoom (Figura 8.6 (direita)) e erros na Figura 8.7. Para polinónios de Legendre de grau 12799, $n = 12800$ e $\lambda = 10^{-6}$, aproximamos y à precisão da máquina. Importa relembrar que a solução para este problema é $y = \text{erf}\left(\frac{x}{\sqrt{\lambda}}\right)$ e não $y = \text{sign}(x)$, sendo somente quando o limite (8.4) for satisfeito. Desta forma, é natural que o fenómeno de Gibbs se mantenha presente para valores cada vez menores de λ.

A função $\text{sign}(x)$ possui uma conhecida expansão na série dos polinómios de Legendre bem como nos de Chebyshev de primeira espécie, a saber

$$\begin{aligned} \text{sign}(x) &= \sum_{k=0}^{\infty} l_k P_k(x) = \sum_{k=0}^{\infty} \frac{(-1)^k \Gamma(\frac{1}{2}+k)\left(2k+\frac{3}{2}\right)}{\Gamma(k)(k+1)!} P_{2k-1}(x) \\ &= \sum_{k=0}^{\infty} c_k T_k(x) = \frac{4}{\pi} \sum_{k=0}^{\infty} \frac{(-1)^{k+1}}{2k-1} T_{2k-1}(x). \end{aligned}$$

Os polinómios de Legendre e Chebyshev de grau ímpar são funções ímpares, e os polinómios de Legendre e Chebyshev de grau par, são funções pares, assim é natural que as expansões acima sejam combinações lineares apenas dos respetivos polinómios de ordem ímpar.

Os coeficientes do vetor a_n, obtidos com a `Tau Toolbox` para a base de Legendre, assim como para a base de Chebyshev, que se referem aos polinómios pares, também são absolutamente todos nulos, mesmo para valores (relativamente) elevados do grau polinomial, como $n = 12800$. Isto evidencia a estabilidade da implementação da `Tau Toolbox`, uma vez que não é gerado ruido nestes coeficientes.

Comparamos ainda, para vários valores de λ, as aproximações da `Tau Toolbox` (y_n) com as aproximações da Chebfun V5.6.0-2017 (f_n), conforme a Tabela 8.2.

n	λ	$\max \lvert y - f_n \rvert$	$\max \lvert y - y_n \rvert$
110	$1 \cdot 10^{-2}$	$1,056 \cdot 10^{-13}$	$1,410 \cdot 10^{-14}$
326	$1 \cdot 10^{-3}$	$1,043 \cdot 10^{-13}$	$5,406 \cdot 10^{-14}$
950	$1 \cdot 10^{-4}$	$2,226 \cdot 10^{-12}$	$2,300 \cdot 10^{-12}$
2848	$1 \cdot 10^{-5}$	$2,343 \cdot 10^{-11}$	$2,363 \cdot 10^{-11}$
4096	$1 \cdot 10^{-6}$	$1,147 \cdot 10^{-3}$	$9,237 \cdot 10^{-4}$
4096	$1 \cdot 10^{-7}$	$4,992 \cdot 10^{-1}$	$1,869 \cdot 10^{-1}$
4096	$1 \cdot 10^{-8}$	$7,2934$	$5,879 \cdot 10^{-1}$
4096	$1 \cdot 10^{-12}$	$4330,7$	$7,417 \cdot 10^{-1}$
4096	$1 \cdot 10^{-16}$	4097	$7,417 \cdot 10^{-1}$

Tabela 8.2: Comparação entre `Tau Toolbox` (y_n) e Chebfun V5.6.0-2017 (f_n) para o Exemplo 8.4.

A Figura 8.8 apresenta o erro em todo o domínio $[-1,\ 1]$ para o Exemplo 8.4 para os dados referentes à última linha da Tabela 8.2.

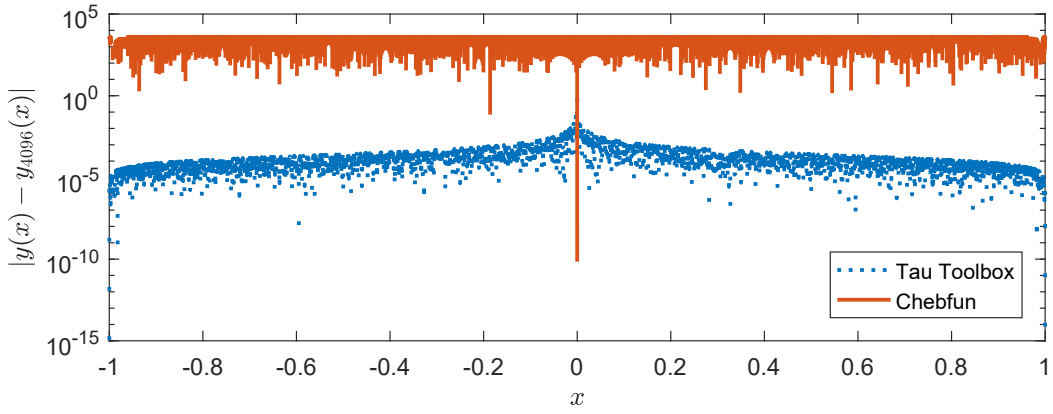

Figura 8.8: Erro para o Exemplo 8.4, com aproximações pela `Tau Toolbox` e Chebfun V5.6.0-2017, com $n = 4096$ e $\lambda = 10^{-16}$.

Conforme apresentado na Figura 8.8, percebemos que a aproximação da solução do Exemplo 8.4 pela `Tau Toolbox`, para $n = 4096$ e $\lambda = 10^{-16}$, aproxima-se razoavelmente da solução, excepto pela presença do fenómeno de Gibbs, mantendo um erro máximo na ordem de 10^{-1} na vizinhança de $x = 0$ e de 10^{-5} na maior parte do domínio.

Ainda para este Exemplo, podemos considerar o método tau parcelar e obter a solução aproximada em p partes do domínio, conforme formulado na Secção 4.5. A Figura 8.9 ilustra o experimento.

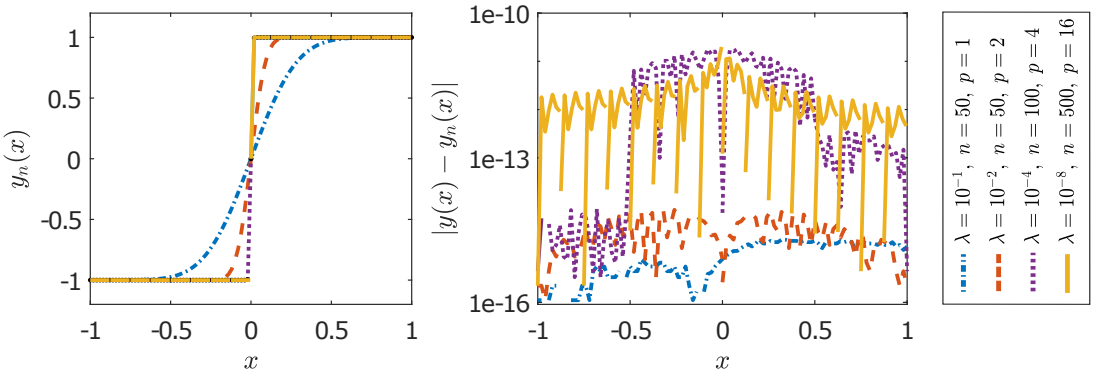

Figura 8.9: Solução aproximada e erros para o Exemplo 8.4, com diferentes valores de n, p e λ.

Conforme apresentado na Figura 8.9, mesmo para o caso com $\lambda = 10^{-8}$, se considerarmos $n = 500$ e $p = 16$ atingimos uma precisão da aproximação da solução inferior a 10^{-10}, o que não ocorre quando não decompomos o intervalo.

Exemplo 8.5. (Olver and Townsend, 2013) Seja o problema diferencial

$$\begin{cases} \lambda \dfrac{d^2}{dx^2} y - 2x \left(\cos(x) - \dfrac{8}{10} \right) \dfrac{d}{dx} y + \left(\cos(x) - \dfrac{8}{10} \right) y = 0 \\ y(-1) = y(1) = 1 \end{cases} .$$

Este problema é interessante por ser também um problema *stiff*, para valores do parâmetro λ próximos de zero, ter condições de fronteira e ter coeficientes não polinomiais. O Código a seguir apresenta a solução via `Tau Toolbox`.

Código MATLAB 8.2. `Tau Toolbox` para o Exemplo 8.5.

```
% Create tau objects.
[x, y] = tau('LegendreP', [-1 1], 100);

% Specify ode and conditions.
ode = '1e-7*diff(y, 2)-2*x*(cos(x)-0.8)*diff(y)+(cos(x)-0.8)*y=0';
conditions = {'y(-1)=1';'y(1)=1'};

% Solve the problem.
a = tausolver(x, y, ode, conditions , 'pieces', 50, 'resid', 0);
```

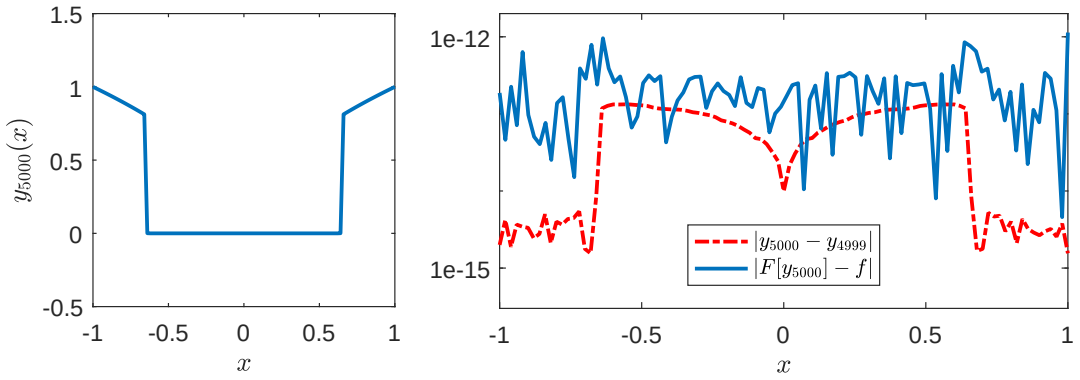

Figura 8.10: Aproximação, diferença de aproximações consecutivas e resíduo para o Exemplo 8.5 pela `Tau Toolbox` com $n = 5000$ e $\lambda = 10^{-6}$.

Conforme a Figura 8.10, tanto o erro quanto o resíduo são da ordem de $1 \cdot 10^{-13}$ para uma aproximação de ordem 5000 e com $\lambda = 10^{-6}$. Podemos, para este mesmo exemplo, aplicar a versão parcelar do método tau para problemas de valor de fronteira (Subsecção 4.5.2), com a intenção de: diminuir o grau de aproximação e manter a mesma precisão, ou, manter o grau e melhorar a precisão. Para forçar ainda mais a rigidez (*stiffness*) do problema, consideremos tornar $\lambda = 10^{-7}$ e aplicar a versão parcelar do método tau na `Tau Toolbox` apenas acrescentando um parâmetro:

```
a = tausolver(x, y, ode, conditions, 'pieces', 100)
```

Com 100 repartições do domínio encontramos a mesma precisão de 10^{-13}, para grau $n = 200$ e $\lambda = 10^{-7}$. A Figura 8.11 mostra os resultados.

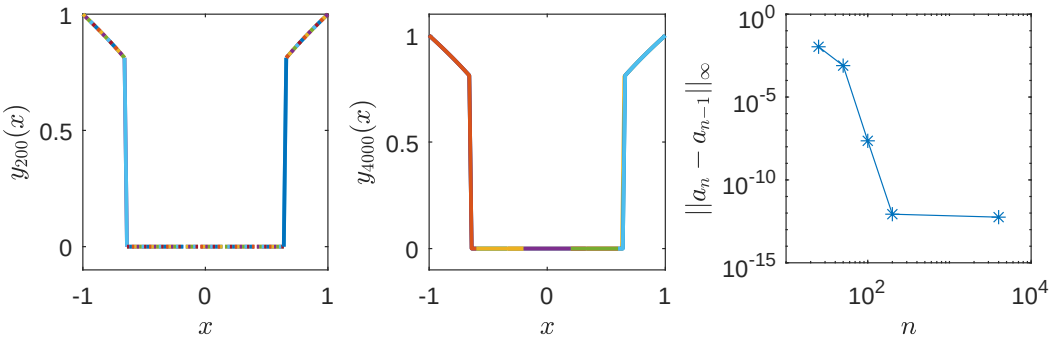

Figura 8.11: Aproximação na versão parcelar (100 e 5 divisões) e norma para aproximações consecutivas para o Exemplo 8.5 pela `Tau Toolbox` com $n = 25, 50, 100, 200, 4000$ e $\lambda = 10^{-7}$.

A grande vantagem em aplicar a versão parcelar está no tempo de computação do resultado, que é muito mais eficiente se comparada com a versão em intervalo inteiro. Mesmo considerando que a versão parcelar resolveu um sistema de 20000 equações (100

partes com grau 200), é consideravelmente mais veloz que o sistema de 5000 equações, e também produz resultados mais precisos. A característica está no facto de que na versão parcelar a matriz do sistema é construída por bocados, e ganha esparsidade (Figura 8.12).

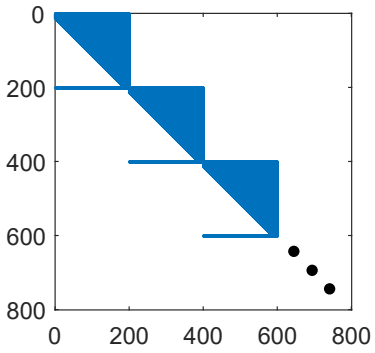

Figura 8.12: Estrutura de esparsidade da matriz T para o Exemplo 8.5 com $n = 200$. Primeiros 3 blocos $n \times n$.

8.2.1.2 Problemas não lineares

Exemplo 8.6. Seja o problema diferencial não linear

$$\begin{cases} \dfrac{d}{dx}y + \cos\left(x^2 y\right) = \cos(x) + \cos\left(x^2 \sin(x)\right), & x \in]0,\ 2[\\ y(0) = 0 \end{cases},$$

com a solução exata $y = \sin(x)$.

Solução. A expansão em série de Taylor para o termo não linear, em torno de y_0, é $\cos(x^2 y) \approx \cos(x^2 y_0) - x^2 \sin(x^2 y_0)(y - y_0)$, e portanto reescrevemos o Exemplo 8.6, iterativamente por Newton, como

$$\begin{cases} \dfrac{d}{dx}y_n^{(k+1)} - x^2 \sin\left(x^2 y_n^{(k)}\right) y_n^{(k+1)} = f, & x \in]0,\ 2[, \quad k \geq 0 \\ y(0) = 0, \quad y^{(0)}(0) = \mathcal{Z}_n \mathsf{y}^{(0)} \end{cases},$$

onde

$$f = \cos(x) + \cos\left(x^2 \sin(x)\right) - \cos\left(x^2 y_n^{(k)}\right) - x^2 y_n^{(k)} \sin\left(x^2 y_n^{(k)}\right)$$

e $\mathsf{y}^{(0)} = [\underbrace{0,\ \ldots,\ 0}_{n \text{ vezes}}]^T$. O operador é definido como

$$\mathsf{D}_{\mathcal{Z}}^{(k)} = \mathsf{N}_{\mathcal{Z}} - \mathsf{M}_{\mathcal{Z}}^2 \mathsf{Z}^{(k)} \mathsf{I}, \quad \mathsf{Z}^{(k)} = \sum_{i=0}^{n-1} c_i^{(k)} Z_i(\mathsf{M}_{\mathcal{Z}}),$$

onde $\mathsf{c}^{(k)} = [c_0^{(k)},\ \ldots,\ c_{n-1}^{(k)}]$ são os coeficientes de uma aproximação polinomial da função $\sin(x^2 y_n^{(k)})$, que podem calcular-se ou por uma interpolação polinomial ortogonal a esta

função, ou utilizando o método tau a um problema auxiliar. Para o cálculo eficiente dos coeficientes de $x^2 y_n^{(k)}$ utilizamos os coeficientes de linearização para o produto.

A Tau Toolbox oferece a função **tauodenewton**, que aproxima a solução de problemas não lineares, isto é, apenas informando o problema linearizado, conforme apresentado no Código 8.3.

Código MATLAB 8.3. Tau Toolbox para o Exemplo 8.6.

```
% Create tau objects.
[x, y] = tau('ChebyshevT', [0 2], 20);

% Specify problem and conditions.
ode = ['diff(y)-x^2*sin(x^2*yo)*y = ' ,...
       'cos(x)+cos(x^2*sin(x))-cos(x^2*yo)-x^2*yo*sin(x^2*yo)'];
conditions = 'y(0)=0';

% Approximate solution.
a = tauodenewton(x, y, ode, conditions, 'iter', 1, 'tol', eps);
```

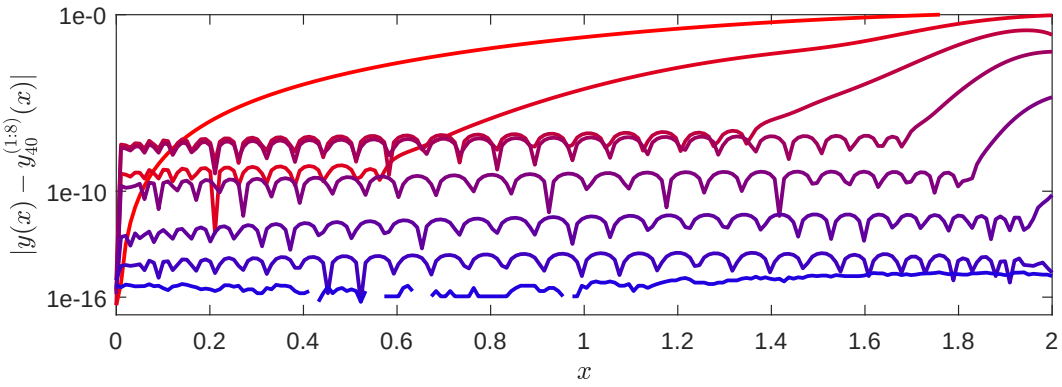

Figura 8.13: Erro para o Exemplo 8.6 com 8 iterações.

A Figura 8.13 apresenta 8 iterações para o problema proposto, que convergem para a precisão da máquina. Importante realçar que os coeficientes de linearização contribuem fortemente para esta precisão.

Exemplo 8.7. (Corless et al., 1996) Seja o problema diferencial

$$\begin{cases} \dfrac{d}{dx}y + y^2(y-1) = 0, \ x \in \left]0, \ \dfrac{2}{\delta}\right[\\ y(0) = \delta \end{cases}, \ \delta > 0,$$

cuja solução exata é $y = \left(W\left(\left(\frac{1}{\delta} - 1\right) e^{\frac{1}{\delta} - 1 - x}\right) + 1\right)^{-1}$, onde $W(z)$ é a função W de Lambert solução de $W(z)e^{W(z)} = z$.

Este problema é interessante por ser não linear e ter uma solução de difícil aproximação à medida que o valor do parâmetro *delta* diminui. A Figura 8.14 apresenta

uma comparação da aproximação obtida com a Tau Toolbox e com as funções MATLAB ode45 e ode23s, para $\delta = \frac{1}{100}$.

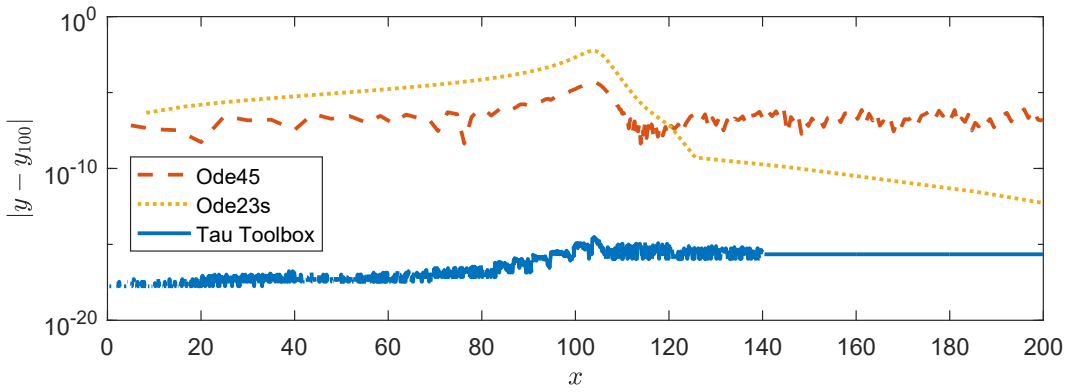

Figura 8.14: Erro para o Exemplo 8.7, com a Tau Toolbox, ode45 e ode23s e $\delta = \frac{1}{100}$.

Conforme apresentado na Figura 8.14, as funções não atingem a mesma precisão da Tau Toolbox, mesmo quando lhe impomos uma tolerância exageradamente baixa: 10^{-15}. Utilizamos, com a Tau Toolbox para este exemplo, a base de Chebyshev de primeira espécie no intervalo $[0,\ 200]$, e obtivemos a precisão da máquina. O Código a seguir apresenta a função **tauodenewton** para o problema, com a base de Legendre, $\delta = \frac{1}{700}$ e utilizando a versão parcelar com $p = 100$. Os respetivos resultados são apresentados na Figura 8.15.

Código MATLAB 8.4. Tau Toolbox para o Exemplo 8.4.

```
% Create tau objects.
[x, y] = tau('LegendreP', [0, 1400], 100);

% Specify problem, conditions and exact solution.
ode        = 'diff(y) + 3*yo^2*y - 2*yo*y = 2*yo^3 - yo^2';
conditions = 'y(0) = 1/700';
solution   = '1/(lambertw(0,((1/700)^-1-1)*exp(((1/700)^-1-1)-x))+1)';

% Compute the approximate solution.
a = tauodenewton(x, y, ode, conditions, ................% Required inputs.
    'exact_solution', solution, ................% Exact solution is known.
    'apsol', 1, ..............................% Shows approximate solution.
    'resid', 1, ..........................................% Shows residuals.
    'iter', 1, ...............................% Shows newton iterations.
    'pieces', 100); ..................................% Number of pieces.
```

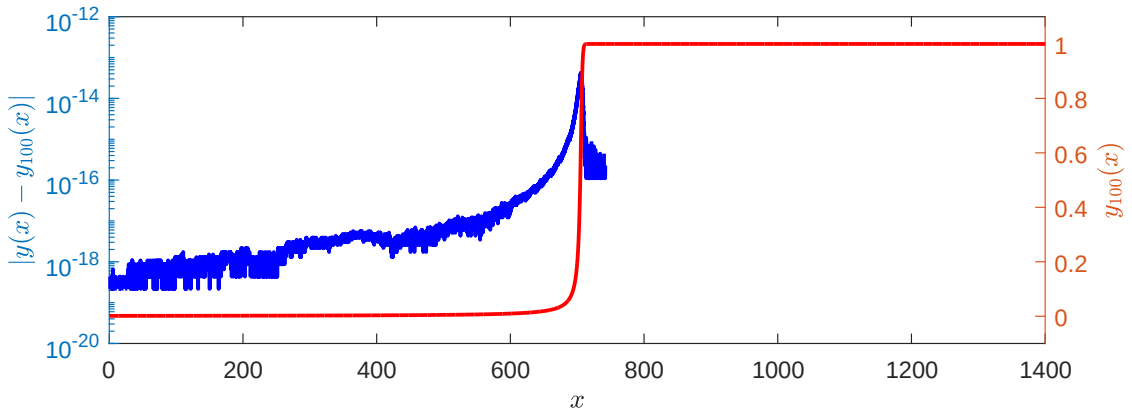

Figura 8.15: Erro (azul) e solução aproximada (vermelho) para o Exemplo 8.7 com a `Tau Toolbox` com $\delta = \frac{1}{700}$.

Conforme a Figura 8.15, percebemos estabilidade da `Tau Toolbox` para este problema *stiff*, mesmo quando diminuímos o parâmetro para $\delta = \frac{1}{700}$, a aproximação da solução foi recuperada próxima a precisão da máquina.

Exemplo 8.8. Consideremos o sistema de equações diferenciais de Lorenz

$$\begin{cases} \dfrac{d}{dt}y_1(t) - \sigma\left(y_2(t) - y_1(t)\right) = 0 \\ \dfrac{d}{dt}y_2(t) - \rho y_1(t) + y_1(t)y_3(t) + y_2(t) = 0 \\ \dfrac{d}{dt}y_3(t) - y_1(t)y_2(t) + \beta y_3(t) = 0 \\ y_1(t_0) = y_{10}, \quad y_2(t_0) = y_{20}, \quad y_3(t_0) = y_{30} \end{cases},$$

com o número de Prandtl $\sigma = 10$, número de Rayleigh $\rho = 28$ e parâmetro $\beta = \frac{8}{3}$.

Solução. Ao expandir em série de Taylor e truncar na primeira ordem os termos não lineares em (8.8) obtém-se

$$\begin{cases} \dfrac{d}{dt}y_1^{(k+1)} - \sigma\left(y_2^{(k+1)} - y_1^{(k+1)}\right) = 0 \\ \dfrac{d}{dt}y_2^{(k+1)} - \rho y_1^{(k+1)} + y_1^{(k)}y_3^{(k+1)} + y_3^{(k)}y_1^{(k+1)} + y_2^{(k+1)} = y_1^{(k)}y_3^{(k)} \\ \dfrac{d}{dt}y_3^{(k+1)} - y_1^{(k)}y_2^{(k+1)} - y_2^{(k)}y_1^{(k+1)} + \beta y_3^{(k+1)} = -y_1^{(k)}y_2^{(k)} \\ y_1^{(1)}(t_0) = y_{10}, \quad y_2^{(1)}(t_0) = y_{20}, \quad y_3^{(1)}(t_0) = y_{30} \end{cases}.$$

O sistema a ser solucionado, em cada iteração k, será $\mathsf{T}_{\mathcal{P}}\mathsf{a}_{\mathcal{P}} = \mathsf{b}_{\mathcal{P}}$, onde

$$\mathsf{T}_{\mathcal{P}} = \begin{bmatrix} P_0(t_0) & \dots & P_{n-1}(t_0) & \mathbf{0} & & & \mathbf{0} & & \\ \mathbf{0} & & & P_0(t_0) & \dots & P_{n-1}(t_0) & \mathbf{0} & & \\ \mathbf{0} & & & \mathbf{0} & & & P_0(t_0) & \dots & P_{n-1}(t_0) \\ \mathsf{N}_{\mathcal{P}} + \sigma\mathsf{I} & & & -\sigma\mathsf{I} & & & \mathbf{0} & & \\ -\rho\mathsf{I} + y_3^{(k)}(\mathsf{M}_{\mathcal{P}}) & & & \mathsf{N}_{\mathcal{P}} + \mathsf{I} & & & y_1^{(k)}(\mathsf{M}_{\mathcal{P}}) & & \\ -y_2^{(k)}(\mathsf{M}_{\mathcal{P}}) & & & -y_1^{(k)}(\mathsf{M}_{\mathcal{P}}) & & & \mathsf{N}_{\mathcal{P}} + \beta\mathsf{I} & & \end{bmatrix},$$

$$\mathsf{b}_{\mathcal{P}} = \left[y_{10},\ y_{20},\ y_{30},\ 0,\ \ldots,\ 0,\ y_1^{(k)}y_3^{(k)},\ -y_1^{(k)}y_2^{(k)}\right]^T$$

e $\mathsf{a}_{\mathcal{P}}$ é o vetor de coeficientes a determinar.

O Código 8.5 (**nonlinear_ode_lorenz**) é uma aplicação do método tau na versão parcelar, uma vez que para este tipo de problemas temos um largo domínio temporal. Utilizamos a base de Legendre por fornecer melhores resultados no fim de cada intervalo, o que pode melhorar a precisão da solução aproximada.

Código MATLAB 8.5. Tau Toolbox para o Exemplo 8.8.

```
% Solving the Lorenz's system of nonlinear differential equations.
% y1' - sigma(y2 - y1) = 0
% y2' - rho*y1 + y1*y3 + y2 = 0
% y3' - y1*y2 + beta*y3 = 0

% Parameters, domain, step and initial conditions.
rho = 28;  beta = 8/3;  sigma = 10;
domain = [0 100];  step = 0.5;  x1 = -8;  x2 = -7;  x3 = 29;

% Domain partition.
I = [(0:step:domain(2)-step)', (step:step:domain(2))'];

% For all steps.
for i = 1:length(I)

    % If not the first step, initial conditions must be updated.
    if i == 1, else, x1 = Y1(end); x2 = Y2(end); x3 = Y3(end); end

    % Create tau objects.
    [x, y] = tau('LegendreP', I(i, :), 50);

    % M matrix and points to evaluate the approximate solution.
    M = matrixM(x); points = linspace(x, 50);

    % Initial approximations (satisfying the initial conditions).
    y1 = zeros(y.n, 1); y2 = y1; y3 = y1;
    y1(1) = x1; y2(1) = x2; y3(1) = x3;

    % Blocks independent on the iteration.
    T11 = diff(y) + sigma*y;
    T12 = -sigma*y;
    T13 = zeros(y.n);
    T22 = diff(y) + y;
    T33 = diff(y) + beta*y;

    % Memory allocation for T matrix and b vector.
    T = zeros(3*y.n); b = zeros(3*y.n, 1);

    % Write the conditions.
    T(1, 1:y.n) = orthoval(x, x.domain(1), 'differorder', 0);
    T(2, y.n+1:2*y.n) = orthoval(x, x.domain(1), 'differorder', 0);
    T(3, 2*y.n+1:3*y.n) = orthoval(x, x.domain(1), 'differorder', 0);
    b(1) = x1;  b(2) = x2;  b(3) = x3;

    % First approximation.
```

```
    yprev = [y1; y1; y3];  ynext = [y1; y1; y3] + 1;

    % Loop to obtain the aproximation.
    while max(abs(ynext - yprev)) > 1e-12

        % Blocks dependent on the iteration.
        T21 = -rho*y + orthoval(x, M, 'coef', y3)*y;
        T23 = orthoval(x, M, 'coef', y1)*y;
        T31 = -orthoval(x, M, 'coef', y2)*y;
        T32 = -orthoval(x, M, 'coef', y1)*y;

        % Compute each right hand side.
        rhs2 = legpolyprod(y1, y3, x.n)';
        rhs3 = -legpolyprod(y1, y2, x.n)';

        % Truncate the blocks.
        T(4:y.n+2, 1:y.n) = T11.mat(1:end-1, :);
        T(4:y.n+2, y.n+1:2*y.n) = T12.mat(1:end-1, :);
        T(4:y.n+2, 2*y.n+1:3*y.n) = T13(1:end-1, :);
        T(y.n+3:2*y.n+1, 1:y.n) = T21.mat(1:end-1, :);
        T(y.n+3:2*y.n+1, y.n+1:2*y.n) = T22.mat(1:end-1, :);
        T(y.n+3:2*y.n+1, 2*y.n+1:3*y.n) = T23.mat(1:end-1, :);
        T(2*y.n+2:3*y.n, 1:y.n) = T31.mat(1:end-1, :);
        T(2*y.n+2:3*y.n, y.n+1:2*y.n) = T32.mat(1:end-1, :);
        T(2*y.n+2:3*y.n, 2*y.n+1:3*y.n) = T33.mat(1:end-1, :);
        b(y.n+3:2*y.n+1) = rhs2(1:end-1); b(2*y.n+2:3*y.n) = rhs3(1:end-1);

        % Solve the linear system.
        a = T\b;  a = reshape(a, y.n, 3);
        y1 = a(:, 1);  y2 = a(:, 2);  y3 = a(:, 3);

        % Update the solution.
        yprev = ynext;  ynext = [y1; y2; y3];

    end

    % Orthogonal evaluation.
    Y1 = orthoval(x, points, 'coef', y1);
    Y2 = orthoval(x, points, 'coef', y2);
    Y3 = orthoval(x, points, 'coef', y3);

    % Plot the results.
    % figure(1)
    subplot(1,2,1)
    hold on, box on, grid on, view([-5, 5, 5]), v = zeros(length(Y2), 1);
    xlim([-20 30]); ylim([-30 30]); zlim([0 50]);
    color = [.75, .75, .75];
    plot3(Y1, Y2, Y3);
    plot3(v+30, Y2, Y3, 'color', color);
    plot3(Y1, v-30, Y3, 'color', color);
    plot3(Y1, Y2, v, 'color', color);
    refreshdata, drawnow

    % figure(2)
    subplot(1,2,2)
    hold on, box on;
```

```
    plot(points, Y1, 'b'), plot(points, Y2, 'g'), plot(points, Y3, 'r');

end
figure(1)
xlabel('$y_1(t)$', 'Interpreter', 'Latex');
ylabel('$y_2(t)$', 'Interpreter', 'Latex');
zlabel('$y_3(t)$', 'Interpreter', 'Latex');
```

A execução do Código 8.5, fornece a solução tau aproximada no plano de fase (atrator de Lorenz na Figura 8.16 (direita)). O critério de paragem garantiu uma aproximação de $1 \cdot 10^{-12}$ para o erro entre duas aproximações consecutivas em cada passo. À esquerda na Figura 8.16, vemos a diferença entre a aproximação tau e a aproximação `ode45`.

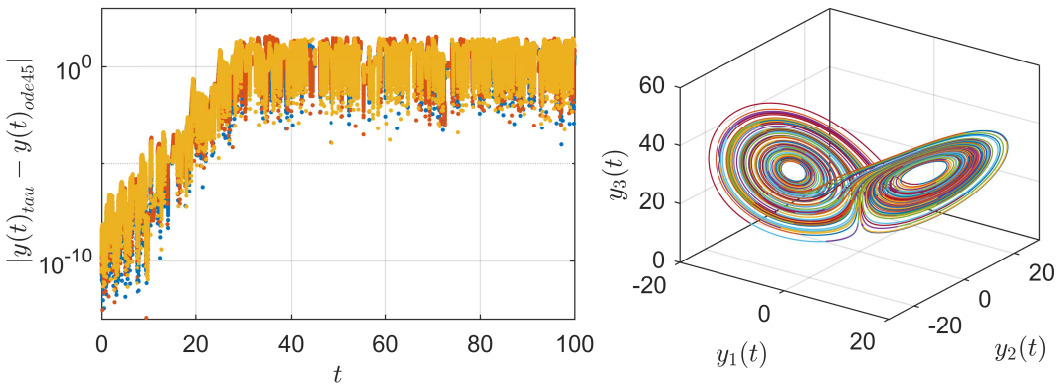

Figura 8.16: Diferença entre aproximação com `Tau Toolbox` e aproximação por `ode45` para o atrator de Lorenz.

Esta diferença (apresentada na Figura 8.16) mostra que ambas as aproximações começam muito próximas, mas à medida que t evolui se afastam até serem completamente diferentes, o que sugere que uma das duas ou ambas aproximações são ruins. Para certificar a precisão da `Tau Toolbox`, consideremos aumentar o intervalo para $[0, 10^3]$, e aplicar a versão parcelar do método tau com grau 100. E para verificar a precisão da solução, consideremos avaliar o resíduo obtido: $|F[y_{100}(t)] - f(t)|$, onde y_{100} é a solução aproximada do problema linearizado. O resultado é apresentado na Figura 8.17.

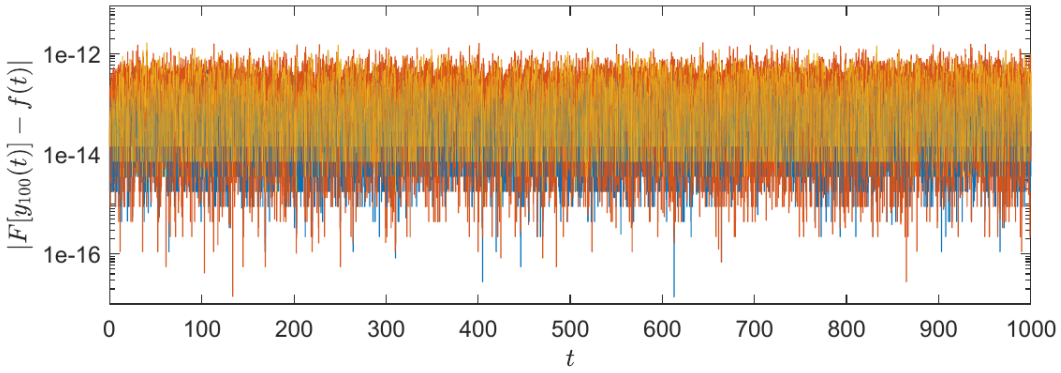

Figura 8.17: Resíduo $|F[y_{100}(t)] - f(t)|$ para o Exemplo 8.8, com $n = 100$ no intervalo $[0,\ 10^3]$.

Apresentamos o Código 8.5 para ilustrar as rotinas invocadas por um usuário avançado com um certo domínio prévio do método tau, todavia, este mesmo problema pode ser processado com a função **tausysnewton**, capaz de aproximar pelo método tau sistemas de equações diferenciais linearizadas. O Código 8.6 ilustra a simplificação frente ao Código 8.5.

Código MATLAB 8.6. Tau Toolbox para o Exemplo 8.8

```
% Create tau objects.
[x, y] = tau('LegendreP', [0 30], 10);

% Specify problem, conditions and exact solution.
ode = {'diff(y1) -  10*y2  +  10*y1                =  0'; ...
       'diff(y2) -  28*y1  + yo3*y1 + yo1*y3 + y2 =  yo1*yo3'; ...
       'diff(y3) + 8/3*y3  - yo2*y1 - yo1*y2      = -yo1*yo2'};

conditions = {'y1(0)=-8';'y2(0)=-7';'y3(0)=16'};

% Compute the approximate solution.
a = tausysnewton(x, y, ode, conditions, ................% Required inputs.
                 'pieces', 90, ........................% Number of pieces.
                 'iter', 1, ......................% Show newton iterations.
                 'phase', 1, .........................% Plot ode phase plan.
                 'step', 0.01); ................% Step for results ploting.
```

Outros problemas não lineares atribuídos aos sistemas dinâmicos periódicos e/ou caóticos (como por exemplo os que conduzem aos atractores de Chen-Lee, Halvorsen, Rossler e TSUCS2 - Figura 8.18) podem ser encontrados na página da Tau Toolbox.

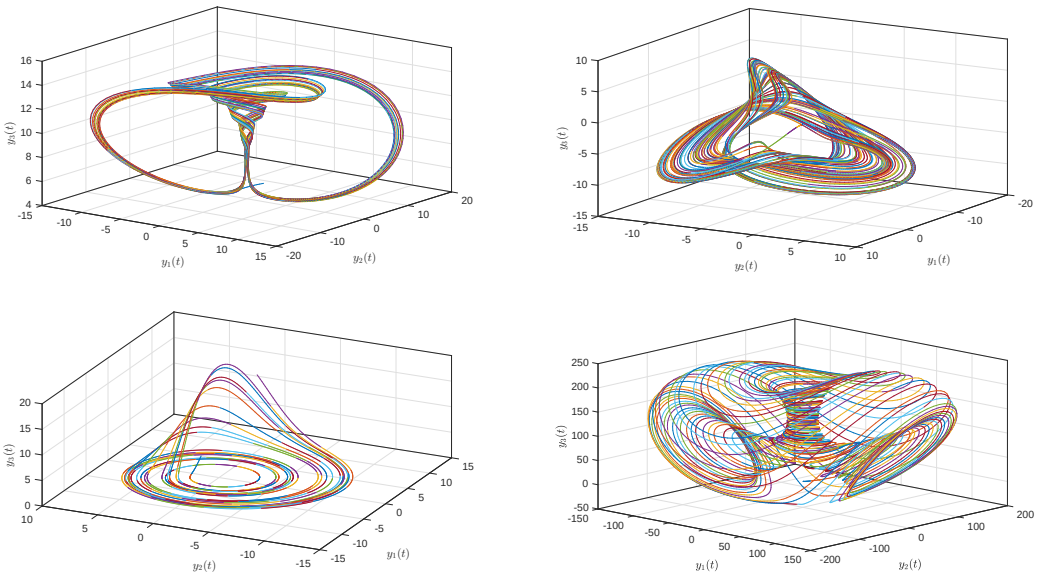

Figura 8.18: Atratores relacionados com sistemas dinâmicos, de cima para baixo e da esquerda para a direita, Chen-Lee, Halvorsen, Rossler e TSUCS2.

A Tau Toolbox mostra-se estável no sentido de que os atratores associados aos sistemas dinâmicos são recuperados com boa precisão mesmo para domínios temporais extremamente longos. Estes resultados mantêm boa precisão devido ao uso dos contributos de estabilidade (apresentados no Capítulo 5) aplicados juntamente com a versão parcelar do método tau.

8.2.2 Equações integro-diferenciais

8.2.2.1 Problemas lineares

Os dois problemas seguintes envolvem sistemas de equações integrais de Volterra, e os resultados da aproximação tau para tais problemas serão comparados com os resultados fornecidos em Yang et al. (2013) que usam o método "*Modified Reproducing Kernel Method*" e com os resultados em Rabbani et al. (2007) que usam o método "*Expansion Method*".

Exemplo 8.9. Seja o sistema de equações integrais de Volterra

$$\begin{cases} y_1 - \displaystyle\int_0^x (\sin(x-t)-1)\, y_1(t)dt - \int_0^x (1 - t\cos(x))\, y_2(t)dt = g_1 \\ y_2 - \displaystyle\int_0^x y_1(t)dt - \int_0^x (x-t) y_2(t)dt = g_2 \end{cases}$$

onde $g_1 = -\frac{1}{2}(x-2)\sin(x) - x\cos(x)^2 + (\sin(x)+2)\cos(x) - 1$ e $g_2 = -x + \sin(x)$. A solução exata é $y_1 = \cos(x)$ e $y_2 = \sin(x)$.

Código MATLAB 8.7. Tau Toolbox para o Exemplo 8.9.

```
% Create tau objects.
[x, y] = tau('ChebyshevT', [0 1], 15);

% Solve the problem.
a = tausolver( ...
    x, ..........................................% Independent tau variable.
    y, ..........................................% Dependent tau variable.
    {['y1-volt(y1,''sin(x-t)-1'')-volt(y2,''1-t*cos(x)'')', ...
    '=-0.5*(x-2)*sin(x)-x*cos(x)^2+(sin(x)+2)*cos(x)-1']; ...
    'y2-volt(y1)-volt(y2,''x-t'')=-x+sin(x)'}, .........% Problem to solve.
    {'no'}, ....................................................% Conditions.
    'exact_solution', {'cos(x)';'sin(x)'}); ..............% Exact solution.
```

São mostrados na Figura 8.19 os erros com as aproximações tau, e com as aproximações dadas Yang et al. (2013) e Rabbani et al. (2007).

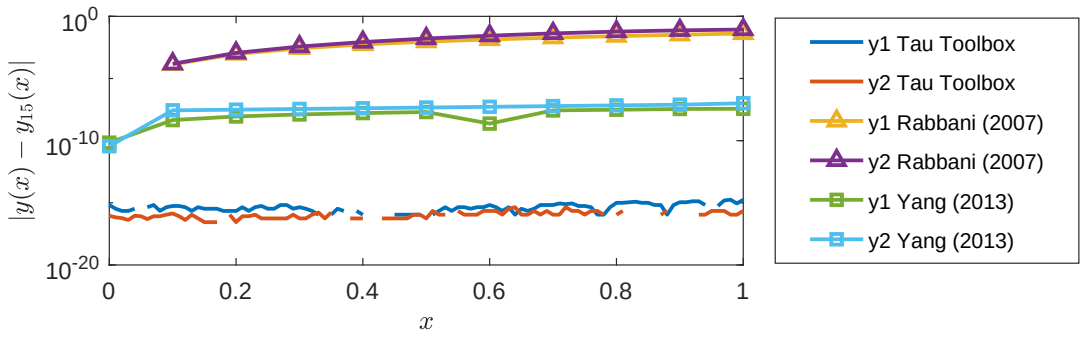

Figura 8.19: Erros para o Exemplo 8.9, com `Tau Toolbox`, Yang et al. (2013) e Rabbani et al. (2007).

Exemplo 8.10. Seja o sistema de equações integrais de Volterra

$$\begin{cases} y_1 - \int_0^x (x-t)^3 y_1(t)dt - \int_0^x (x-t)^2 y_2(t)dt = g_1 \\ y_2 - \int_0^x (x-t)^4 y_1(t)dt - \int_0^x (x-t)^3 y_2(t)dt = g_2 \end{cases}$$

onde $g_1 = -\frac{1}{3}x^4 - \frac{1}{3}x^3 + x^2 + 1$ e $g_2 = -\frac{1}{420}x^7 - \frac{1}{4}x^5 - \frac{1}{4}x^4 - x^3 + x + 1$, e cuja a solução exata é $y_1 = 1 + x^2$ e $y_2 = 1 + x - x^3$.

Código MATLAB 8.8. `Tau Toolbox` para o Exemplo 8.10.

```
% Create tau objects.
[x, y] = tau('LegendreP', [0 1], 8);

% Solve the problem.
a = tausolver( ...
    x, ..........................................% Independent tau variable.
    y, ..........................................% Dependent tau variable.
    {['y1-volt(y1,''(x-t)^3'')-volt(y2,''(x-t)^2'')=', ...
    '-1/3*x^4-1/3*x^3+x^2+1']; ...
```

```
    ['y2-volt(y1,''(x-t)^4'')-volt(y2,''(x-t)^3'')=', ...
    '-1/420*x^7-1/4*x^5-1/4*x^4-x^3+x+1']}, ............% Problem to solve.
    {'no'}, ..................................................% Conditions.
    'exact_solution', {'1+x^2';'1+x-x^3'}); ..............% Exact solution.
```

Aplicando o mesmo processo que no exemplo anterior verificamos o mesmo comportamento de resultado (Figura 8.20).

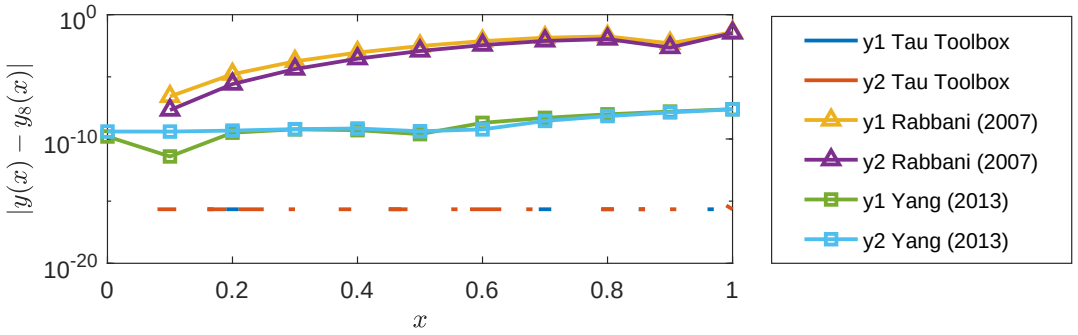

Figura 8.20: Erros para o Exemplo 8.10, com Tau Toolbox, Yang et al. (2013) e Rabbani et al. (2007).

Os problemas dos Exemplos 8.9 e 8.10 são exclusivamente integrais, pelo que a falta de termos diferenciais ou mesmo de condições de contorno não gerou instabilidade sobre a nova implementação do método tau, e as respetivas soluções aproximadas atingiram a precisão da máquina em ambos os casos.

Exemplo 8.11. Seja a equação integro-diferencial de Fredholm

$$\begin{cases} \mathrm{e}^x \dfrac{d^2}{dx^2} y + \cos(x) \dfrac{d}{dx} y + \sin(x) y + \displaystyle\int_{-1}^{1} \mathrm{e}^{(x+1)t} y(t) dt = \\ \qquad\qquad = (\cos(x) + \sin(x) + \mathrm{e}^x)\mathrm{e}^x + \dfrac{2\sinh(x+2)}{x+2} \\ y(1) + y(-1) = \mathrm{e} + \mathrm{e}^{-1}, \quad y(1) + y(-1) - y'(-1) = \mathrm{e} \end{cases},$$

cuja solução exata é $y = \exp(x)$.

Código MATLAB 8.9. Tau Toolbox para o Exemplo 8.11.

```
% Create tau objects.
[x, y] = tau('ChebyshevT', [-1 1], 20);

% Solve the problem.
a = tausolver( ...
    x, ..........................................% Independent tau variable.
    y, ..........................................% Dependent tau variable.
    {['expm(x)*diff(y,2)+cosm(x)*diff(y)+sinm(x)*y+', ...
    'fred(y,''exp((x+1)*t)'')=(cos(x)+sin(x)+', ...
    'exp(x))*exp(x)+2*sinh(x+2)/(x+2)']}, ..............% Problem to solve.
    {'y(1)+y(-1)=3.086161269630488'; ...
    'y(1)+y(-1)-y''(-1)=2.718281828459046'}, ................% Conditions.
    'exact_solution', {'exp(x)'}); ......................% Exact solution.
```

A solução calculada pela Tau Toolbox é mais próxima da solução exata do que a apresentada em AliAbadi and Shahmorad (2002), usando o mesmo grau polinomial $n = 20$ (Figura 8.21).

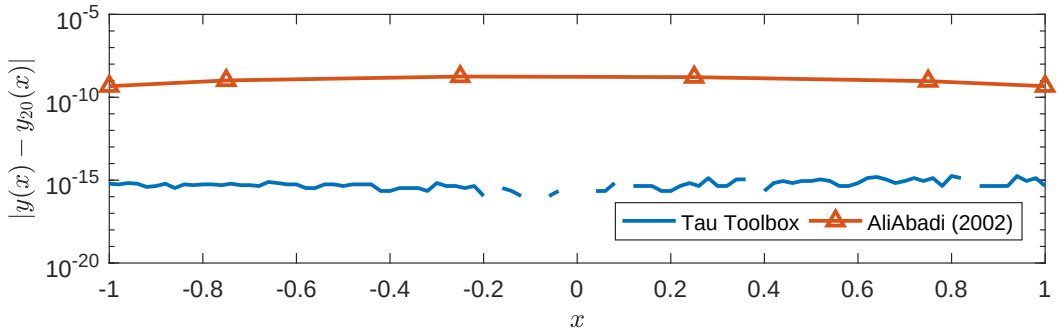

Figura 8.21: Erro para o Exemplo 8.11 com Tau Toolbox e AliAbadi and Shahmorad (2002).

Novamente neste problema atingimos a precisão da máquina (Figura 8.21), sendo que agora estamos a comparar a nova implementação do método tau com outra.

Exemplo 8.12. Seja o sistema de equações integro-diferenciais de Fredholm-Volterra

$$\begin{cases} \dfrac{d}{dx}y + xz + \displaystyle\int_0^1 \exp(x+t)y(t)dt - (x+x^2)\int_0^x \sin(x-t)z(t)dt = g_1 \\ x^2y - 3\dfrac{d}{dx}z - x\displaystyle\int_0^x (2x+\sin(t))y(t)dt - \int_0^1 (x-y)z(t)dt = g_2 \\ y(0) + 2z(0) = 1, \quad y(1) - 3z(1) \approx 5,87316 \end{cases},$$

onde g_1 e g_2 são tais que a solução exata é $y = \sin(x) + \exp(x)$ e $z = \cos(x) - \exp(x)$.

Queremos ilustrar com o Código a seguir, a utilização das rotinas da Tau Toolbox em nível avançado.

Código MATLAB 8.10. Tau Toolbox para o Exemplo 8.12.

```
% Create tau objects.
[x, y] = tau('ChebyshevT', [0 1], 20);

% Build matrix T blocks.
T11 = diff(y) + fred(y, 'exp(x+t)');
T12 = x*y -(x+x^2)*volt(y, 'sin(x-t)');
T21 = x^2*y - x*volt(y, '2*x+sin(t)');
T22 = -3*diff(y)-fred(y, 'x-t');

% Write the conditions.
T = zeros(2*x.n);
T(1, 1:x.n) = orthoval(x, 0, 'differorder', 0);
T(1, x.n+1:2*x.n) = 2*orthoval(x, 0, 'differorder', 0);
T(2, 1:x.n) = orthoval(x, 1, 'differorder', 0);
```

```
T(2, x.n+1:2*x.n) = -3*orthoval(x, 1, 'difforder', 0);

% Blocks truncation.
T(3:x.n+1, 1:x.n) = T11.mat(1:end-1, :);
T(3:x.n+1, x.n+1:2*x.n) = T12.mat(1:end-1, :);
T(x.n+2:2*x.n, 1:x.n) = T21.mat(1:end-1, :);
T(x.n+2:2*x.n, x.n+1:2*x.n) = T22.mat(1:end-1, :);

% Write b vector.
b = zeros(2*x.n, 1);
b(1) = 1; b(2) = sin(1)+exp(1)-3*(cos(1)-exp(1));
g1 = interporth(x, ['cos(x)+exp(x)+x*(cos(x)-exp(x))-', ...
    '(x^2+x)*(cos(x)/2-exp(x)/2+sin(x)/2+(x*sin(x))/2)+', ...
    '(exp(1)*exp(x)*(exp(1)-cos(1)+sin(1)))/2']);
b(3:x.n+1) = g1(1:end-1);
g2 = interporth(x, ['cos(1)+sin(1)+3*exp(x)+', ...
    '3*sin(x)-x*(x/2-2*x*(cos(x)-1)+2*x*(exp(x)-1)-', ...
    '(cos(x)*sin(x))/2-(exp(x)*(cos(x)-sin(x)))/2+1/2)-', ...
    'x*sin(1)+x^2*(exp(x)+sin(x))+x*(exp(1)-1)-2']);
b(x.n+2:2*x.n) = g2(1:end-1);

% Solve the linear system and show the results.
a = T\b;
points = x.domain(1):0.01:x.domain(2);
y = orthoval(x, points, 'coef', a(1:x.n));
z = orthoval(x, points, 'coef', a(x.n+1:2*x.n));

semilogy(points, abs(y-(sin(points)+exp(points))), 'LineWidth', 1.6); ...
   hold on
semilogy(points, abs(z-(cos(points)-exp(points))), 'LineWidth', 1.6)
xlabel('$x$', 'Interpreter', 'Latex')
ylabel('$|y(x)-y_{20}(x)|$', 'Interpreter', 'Latex')
```

A execução do Código 8.10 aproxima a solução do problema do Exemplo 8.12 na precisão da máquina.

8.2.2.2 Problemas não lineares

Exemplo 8.13. Seja a equação integro-diferencial não linear de Fredholm (Dehghan and Salehi, 2012)

$$\begin{cases} \dfrac{d}{dx}y + y - \displaystyle\int_0^1 y(t)^2 dt = 0.5(e^{-2} - 1) \\ y(0) = 1 \end{cases},$$

cuja solução exata é $y = \exp(-x)$.

Solução. Ao expandir em série de Taylor e truncar na primeira ordem o termo não linear em (8.13) obtemos

$$y^2(t) \approx 2y^{(k)}y^{(k+1)} - \left(y^{(k)}\right)^2,$$

e portanto, fazendo uso da variável auxiliar $y_2 = y_1^2$ ($y_1 = y$) o problema (8.13) passa ser escrito na forma

$$\begin{cases} \dfrac{d}{dx}y_1^{(k+1)} + y_1^{(k+1)} - \displaystyle\int_0^1 y_2^{(k+1)}dt = 0.5(e^{-2}-1) \\ \dfrac{d}{dx}y_2^{(k+1)} - 2\left(y_1^{(k)}\dfrac{d}{dx}y_1^{(k+1)} + \dfrac{d}{dx}y_1^{(k)}y_1^{(k+1)}\right) = -2y_1^{(k)}\dfrac{d}{dx}y_1^{(k)} \\ y_1^{(1)}(0) = 1, \quad y_2^{(1)}(0) = 1 \end{cases},$$

e a matriz do sistema (na base de Chebyshev) $\mathsf{T}_\mathcal{T}\mathsf{a}_\mathcal{T} = \mathsf{b}_\mathcal{T}$ será

$$\mathsf{T}_\mathcal{T} = \begin{bmatrix} T_0(0) & \dots & T_{n-1}(0) & \mathbf{0} \\ \mathbf{0} & & & T_0(0) \quad \dots \quad T_{n-1}(0) \\ \mathsf{N}_\mathcal{T} + \mathsf{I} & & & -\mathsf{S}^{(F)} \\ -2\left(y_1^{(k)}(\mathsf{M}_\mathcal{T})\mathsf{N}_\mathcal{T} + \dfrac{d}{dx}y_1^{(k)}(\mathsf{M}_\mathcal{T})\right) & & & \mathsf{N}_\mathcal{T} \end{bmatrix},$$

onde $\mathsf{S}^{(F)}$ é obtido com a equação (7.4), e fazendo uso da função **fred**. O vetor independente será

$$\mathsf{b}_\mathcal{T} = \left[1,\ 1,\ 0.5(e^{-2}-1),\ 0,\ \dots,\ 0,\ -2y_1^{(k)}\frac{d}{dx}y_1^{(k)}\right]^T.$$

O erro apresentado em Dehghan and Salehi (2012) é $||\mathsf{e}||_\infty = \max|y_n(j) - y_{exact}(j)|$, e portanto faremos uso da mesma mensuração para comparar os resultados, conforme a Tabela 8.3.

n	$\|\|\mathsf{e}\|\|_\infty$ Dehghan and Salehi (2012)	$\|\|\mathsf{e}\|\|_\infty$ Tau Toolbox	Tempo CPU Dehghan and Salehi (2012)	Tempo CPU Tau Toolbox
5	$9,63 \cdot 10^{-4}$	$1,58 \cdot 10^{-4}$	0,42	0,03
9	$1,28 \cdot 10^{-4}$	$1,28 \cdot 10^{-9}$	0,58	0,03
17	$2,87 \cdot 10^{-5}$	$7,77 \cdot 10^{-16}$	0,73	0,04
33	$5,61 \cdot 10^{-6}$	$4,44 \cdot 10^{-16}$	1,27	0,07
65	$2,39 \cdot 10^{-6}$	$4,44 \cdot 10^{-16}$	1,54	0,37
129	$1,28 \cdot 10^{-6}$	$4,44 \cdot 10^{-16}$	2,15	2,35

Tabela 8.3: Comparação entre Tau Toolbox e Dehghan and Salehi (2012).

Código MATLAB 8.11. Tau Toolbox para o Exemplo 8.13.

```
% Solving a nonlinear Fredholm integro-differential equation:
% y' + y - fred(y^2) = 1/2(e^(-2)-1)
% y(0) = 1
% Exact solution: y(x) = exp(-x)

t = cputime;
% Create tau objects.
```

```
[x, y] = tau('ChebyshevT', [0 1], 129);

% Vector with initial approximations (satisfing the conditions).
y1 = zeros(y.n, 1); y1(1) = 1;
y2 = y1;

% Matrix N and M in the Chebyshev basis.
N = matrixN(y);  M = matrixM(x);

% Compute right hand side 1.
rhs1 = interporth(x, '0.5*(exp(-2)-1)');

% Compute blocks independent of the iteration.
T11 = diff(y)+y;  T12 = fred(y);  T22 = diff(y);

% Memory allocation for T matrix ans b vector.
T = zeros(2*y.n);  b = zeros(2*y.n, 1);

% Write the conditions.
T(1, 1:y.n)       = orthoval(x, 0, 'differorder', 0); b(1) = 1; % y1(0) = 1
T(2, y.n+1:2*y.n) = orthoval(x, 0, 'differorder', 0); b(2) = 1; % y2(0) = 1

% Blocks truncation.
T(3: y.n+2-1, 1:y.n) = T11.mat(1:end-1, :);
T(3: y.n+2-1, y.n+1:2*y.n) = -T12.mat(1:end-1, :);
T(y.n+2: 2*y.n+2-2, y.n+1:2*y.n) = T22.mat(1:end-1, :);

% Allocate the right hand side 1.
b(3: y.n+2-1) = rhs1(1:end-1);

% Points for ploting.
points = linspace(x, 200);

% Loop for iterations.
for k = 1:6

% Derivate y1 using N matrix.
y1p = N*y1;

% Compute right hand side 2 and allocate space.
rhs2 = -2*chebypolyprod(y1, y1p, y.n)';
b(y.n+2: 2*y.n+2-2) = rhs2(1:end-1);

% Block truncation depending on iteration and allocate space.
T21 = -2*(orthoval(x, M, 'coef', y1p)*y + ...
      orthoval(x, M, 'coef', y1)*diff(y));
T(y.n+2: 2*y.n+2-2, 1:y.n) = T21.mat(1:end-1, :);

% Solve the linear system.
a = T\b;  a = reshape(a, y.n, 2);
y1 = a(:, 1);  y2 = a(:, 2);

semilogy(points, abs(orthoval(x, points, 'coef', y1) - (exp(-points))));
hold on
end
```

```
xlabel('$x$', 'Interpreter', 'latex')
ylabel('$|\exp(-x)-y^{(k)}(x)|$', 'Interpreter', 'latex')
```

Exemplo 8.14. Seja o sistema de equações integro-diferenciais não lineares de Volterra (Abbasbandy and Taati, 2009)

$$\begin{cases} \dfrac{d}{dx}y_1 + \dfrac{1}{2}\left(\dfrac{d}{dx}y_2\right)^2 - \displaystyle\int_0^x (x-t)y_2(t) + y_2(t)y_1(t)dt = 1 \\ \dfrac{d}{dx}y_2 - \displaystyle\int_0^x (x-t)y_1(t) - y_2^2(t) + y_1^2(t)dt = 2x \\ y_1(0) = 0, \quad y_2(0) = 1 \end{cases},$$

cuja solução exata é $y_1 = \sinh(x)$ e $y_2 = \cosh(x)$.

Solução. Ao expandir em série de Taylor e truncar na primeira ordem os termos não lineares, obtemos

$$\left(\frac{d}{dx}y_2^{(k+1)}\right)^2 \approx 2\frac{d}{dx}y_2^{(k)}\frac{d}{dx}y_2^{(k+1)} - \left(\frac{d}{dx}y_2^{(k)}\right)^2, \tag{8.5}$$

$$y_2^{(k+1)}y_1^{(k+1)} \approx y_1^{(k)}y_2^{(k+1)} + y_2^{(k)}y_1^{(k+1)} - y_1^{(k)}y_2^{(k)}, \tag{8.6}$$

e

$$\left(y_*^{(k+1)}\right)^2 \approx 2y_*^{(k)}y_*^{(k+1)} - \left(y_*^{(k)}\right)^2. \tag{8.7}$$

A aproximação (8.5) será usada tal qual, enquanto as aproximações (8.6) e (8.7) serão substituídas por novas variáveis auxiliares ($y_3 = y_1^2$, $y_4 = y_2^2$ e $y_5 = y_1y_2$), uma vez que estão dentro dos termos integrais. Desta forma o problema linearizado passa a ter a forma

$$\begin{cases} \dfrac{d}{dx}y_1^{(k+1)} + \dfrac{d}{dx}y_2^{(k)}\dfrac{d}{dx}y_2^{(k+1)} - \displaystyle\int_0^x y_2^{(k+1)}dt - \int_0^x y_5^{(k+1)}dt = 1 + 0.5\left(\dfrac{d}{dx}y_2^{(k)}\right)^2 \\ \dfrac{d}{dx}y_2^{(k+1)} - \displaystyle\int_0^x (x-t)y_1^{(k+1)}dt - \int_0^x y_3^{(k+1)} - y_4^{(k+1)}dt = 2x \\ \dfrac{d}{dx}y_3^{(k+1)} - 2\left(\dfrac{d}{dx}y_1^{(k)}y_1^{(k+1)} + y_1^{(k)}\dfrac{d}{dx}y_1^{(k+1)}\right) = -2y_1^{(k)}\dfrac{d}{dx}y_1^{(k)} \\ \dfrac{d}{dx}y_4^{(k+1)} - 2\left(\dfrac{d}{dx}y_2^{(k)}y_2^{(k+1)} + y_2^{(k)}\dfrac{d}{dx}y_2^{(k+1)}\right) = -2y_2^{(k)}\dfrac{d}{dx}y_2^{(k)} \\ \dfrac{d}{dx}y_5^{(k+1)} - y_1^{(k)}\dfrac{d}{dx}y_2^{(k+1)} - \dfrac{d}{dx}y_1^{(k)}y_2^{(k+1)} - \dfrac{d}{dx}y_2^{(k)}y_1^{(k+1)} - y_2^{(k)}\dfrac{d}{dx}y_1^{(k+1)} = \\ \qquad\qquad = -y_1^{(k)}\dfrac{d}{dx}y_2^{(k)} - \dfrac{d}{dx}y_1^{(k)}y_2^{(k)} \\ y_1^{(1)}(0) = 0, \quad y_2^{(1)}(0) = 1, \quad y_3^{(1)}(0) = 0, \quad y_4^{(1)}(0) = 1, \quad y_5^{(1)}(0) = 0 \end{cases},$$

e a matriz do sistema (na base de Chebyshev) $\mathsf{T}_\mathcal{T}\mathsf{a}_\mathcal{T} = \mathsf{b}_\mathcal{T}$ será

$$\mathsf{T}_{\mathcal{T}} = \begin{bmatrix} \mathsf{v} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathsf{v} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathsf{v} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathsf{v} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathsf{v} \\ \mathsf{N}_{\mathcal{T}} & \frac{d}{dx}y_2^{(k)}(\mathsf{M}_{\mathcal{T}})\mathsf{N}_{\mathcal{T}} - \mathsf{O}_{\mathcal{T}} & \mathbf{0} & \mathbf{0} & -\mathsf{O}_{\mathcal{T}} \\ -\mathsf{S}^{(V)} & \mathsf{N}_{\mathcal{T}} & -\mathsf{O}_{\mathcal{T}} & \mathsf{O}_{\mathcal{T}} & \mathbf{0} \\ -2\left(\frac{d}{dx}y_1^{(k)}(\mathsf{M}_{\mathcal{T}}) + y_1^{(k)}(\mathsf{M}_{\mathcal{T}})\mathsf{N}_{\mathcal{T}}\right) & \mathbf{0} & \mathsf{N}_{\mathcal{T}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -2\left(\frac{d}{dx}y_2^{(k)}(\mathsf{M}_{\mathcal{T}}) + y_2^{(k)}(\mathsf{M}_{\mathcal{T}})\mathsf{N}_{\mathcal{T}}\right) & \mathbf{0} & \mathsf{N}_{\mathcal{T}} & \mathbf{0} \\ -\frac{d}{dx}y_2^{(k)}(\mathsf{M}_{\mathcal{T}}) - y_2^{(k)}(\mathsf{M}_{\mathcal{T}})\mathsf{N}_{\mathcal{T}} & -y_1^{(k)}(\mathsf{M}_{\mathcal{T}})\mathsf{N}_{\mathcal{T}} - \frac{d}{dx}y_1^{(k)}(\mathsf{M}_{\mathcal{T}}) & \mathbf{0} & \mathbf{0} & \mathsf{N}_{\mathcal{T}} \end{bmatrix},$$

onde $\mathsf{S}^{(V)}$ é obtido conforme a equação (7.3), com uso da função **volt** e $\mathsf{v} = [T_0(0) \quad \dots \quad T_{n-1}(0)]$. O vetor independente será

$$\mathsf{b}_{\mathcal{T}} = \Bigg[0,\ 1,\ 0,\ 1,\ 0,\ 1 + 0.5\left(\frac{d}{dx}y_2^{(k)}\right)^2,\ 1,\ 1,\ 0,\ \dots,\ 0,$$

$$-2y_1^{(k)}\frac{d}{dx}y_1^{(k)},\ -2y_2^{(k)}\frac{d}{dx}y_2^{(k)},\ y_1^{(k)}\frac{d}{dx}y_2^{(k)} - \frac{d}{dx}y_1^{(k)}y_2^{(k)}\Bigg]^T.$$

Código MATLAB 8.12. `Tau Toolbox` para o Exemplo 8.14.

```
% Solving a system of nonlinear Volterra integro-differential equations
% y1' + 0.5y2'^2 -  volt((x-t)y2 + y1y2)  = 1
% y2' - volt((x-t)y1 - y2^2 + y1^2) = 2x
% y1(0) = 0, y2(0) = 1
% Exact solution: y1(x) = sinh(x) and y2(x) = cosh(x)

% Create tau objects.
[x, y] = tau('ChebyshevT', [0 1], 50);

% Vector with initial approximations (satisfing the conditions).
y1 = zeros(y.n, 1);  y2 = zeros(y.n, 1);  y2(1) = 1;  y3 = zeros(y.n, 1);
y4 = zeros(y.n, 1);  y4(1) = 1;  y5 = zeros(y.n, 1);

% Matrix N and M in the Chebyshev basis.
N = matrixN(y);  M = matrixM(x);

% Points for ploting.
points = linspace(x, 200); %#ok<NASGU>

% Looping for iterations.
for k = 1:5

% Derivate y1 and y2 using N matrix.
y1p = N*y1; y2p = N*y2;

% Compute right hand sides.
rhs1 = interporth(x, '1') + 0.5*chebypolyprod(y2p, y2p, y.n)';
rhs2 = interporth(x, '2*x');
```

```
rhs3 = -2*chebypolyprod(y1, y1p, y.n)';
rhs4 = -2*chebypolyprod(y2, y2p, y.n)';
rhs5 = -chebypolyprod(y1, y2p, y.n)'-chebypolyprod(y1p, y2, y.n)';

% Create the blocks for the T matrix.
T11 = diff(y);
T12 = orthoval(x, M, 'coef', y2p)*diff(y) - volt(y, 'x-t');
T13 = zeros(y.n);
T14 = zeros(y.n);
T15 = volt(y);

T21 = volt(y, 'x-t');
T22 = diff(y);
T23 = volt(y);
T24 = volt(y);
T25 = zeros(y.n);

T31 = -2*(orthoval(x, M, 'coef', y1p)*y + ...
      orthoval(x, M, 'coef', y1)*diff(y));
T32 = zeros(y.n);
T33 = diff(y);
T34 = zeros(y.n);
T35 = zeros(y.n);

T41 = zeros(y.n);
T42 = -2*(orthoval(x, M, 'coef', y2p)*y + ...
      orthoval(x, M, 'coef', y2)*diff(y));
T43 = zeros(y.n);
T44 = diff(y);
T45 = zeros(y.n);

T51 = -orthoval(x, M, 'coef', y2p)*y - orthoval(x, M, 'coef', y2)*diff(y);
T52 = -orthoval(x, M, 'coef', y1p)*y - orthoval(x, M, 'coef', y1)*diff(y);
T53 = zeros(y.n);
T54 = zeros(y.n);
T55 = diff(y);

% Memory allocation for T matrix and b vector.
T = zeros(5*y.n);  b = zeros(5*y.n, 1);

% Write the conditions.
T(1, 1:y.n)         = orthoval(x, 0, 'differorder', 0); b(1) = 0; % y1(0) ...
    = 0
T(2, y.n+1:2*y.n)   = orthoval(x, 0, 'differorder', 0); b(2) = 1; % y2(0) ...
    = 1
T(3, 2*y.n+1:3*y.n) = orthoval(x, 0, 'differorder', 0); b(3) = 0; % y3(0) ...
    = 0
T(4, 3*y.n+1:4*y.n) = orthoval(x, 0, 'differorder', 0); b(4) = 1; % y4(0) ...
    = 1
T(5, 4*y.n+1:5*y.n) = orthoval(x, 0, 'differorder', 0); b(5) = 0; % y5(0) ...
    = 0

% Truncating the blocks.
T(6: y.n+5-1, 1:y.n) = T11.mat(1:end-1, :);
T(6: y.n+5-1, y.n+1:2*y.n) = T12.mat(1:end-1, :);
T(6: y.n+5-1, 2*y.n+1:3*y.n) = T13(1:end-1, :);
```

```
T(6: y.n+5-1, 3*y.n+1:4*y.n) = T14(1:end-1, :);
T(6: y.n+5-1, 4*y.n+1:5*y.n) = -T15.mat(1:end-1, :);

T(y.n+5: 2*y.n+5-2, 1:y.n) = -T21.mat(1:end-1, :);
T(y.n+5: 2*y.n+5-2, y.n+1:2*y.n) = T22.mat(1:end-1, :);
T(y.n+5: 2*y.n+5-2, 2*y.n+1:3*y.n) = -T23.mat(1:end-1, :);
T(y.n+5: 2*y.n+5-2, 3*y.n+1:4*y.n) = T24.mat(1:end-1, :);
T(y.n+5: 2*y.n+5-2, 4*y.n+1:5*y.n) = T25(1:end-1, :);

T(2*y.n+5-1: 3*y.n+5-3, 1:y.n) = T31.mat(1:end-1, :);
T(2*y.n+5-1: 3*y.n+5-3, y.n+1:2*y.n) = T32(1:end-1, :);
T(2*y.n+5-1: 3*y.n+5-3, 2*y.n+1:3*y.n) = T33.mat(1:end-1, :);
T(2*y.n+5-1: 3*y.n+5-3, 3*y.n+1:4*y.n) = T34(1:end-1, :);
T(2*y.n+5-1: 3*y.n+5-3, 4*y.n+1:5*y.n) = T35(1:end-1, :);

T(3*y.n+5-2: 4*y.n+5-4, 1:y.n) = T41(1:end-1, :);
T(3*y.n+5-2: 4*y.n+5-4, y.n+1:2*y.n) = T42.mat(1:end-1, :);
T(3*y.n+5-2: 4*y.n+5-4, 2*y.n+1:3*y.n) = T43(1:end-1, :);
T(3*y.n+5-2: 4*y.n+5-4, 3*y.n+1:4*y.n) = T44.mat(1:end-1, :);
T(3*y.n+5-2: 4*y.n+5-4, 4*y.n+1:5*y.n) = T45(1:end-1, :);

T(4*y.n+5-3: 5*y.n, 1:y.n) = T51.mat(1:end-1, :);
T(4*y.n+5-3: 5*y.n, y.n+1:2*y.n) = T52.mat(1:end-1, :);
T(4*y.n+5-3: 5*y.n, 2*y.n+1:3*y.n) = T53(1:end-1, :);
T(4*y.n+5-3: 5*y.n, 3*y.n+1:4*y.n) = T54(1:end-1, :);
T(4*y.n+5-3: 5*y.n, 4*y.n+1:5*y.n) = T55.mat(1:end-1, :);

b(0*y.n+5+1: 1*y.n+4) = rhs1(1:end-1);
b(1*y.n+5-0: 2*y.n+3) = rhs2(1:end-1);
b(2*y.n+5-1: 3*y.n+2) = rhs3(1:end-1);
b(3*y.n+5-2: 4*y.n+1) = rhs4(1:end-1);
b(4*y.n+5-3: 5*y.n) = rhs5(1:end-1);

% Solve the linear system.
a = T\b;  a = reshape(a, y.n, 5);
y1 = a(:, 1);  y2 = a(:, 2);  y3 = a(:, 3);  y4 = a(:, 4);  y5 = a(:, 5);

%figure(1)
%subplot(1, 2, 1)
%semilogy(points, abs(orthoval(x, points, 'coef', y1) - (sinh(points))));
%hold on
%subplot(1, 2, 2)
%semilogy(points, abs(orthoval(x, points, 'coef', y2) - (cosh(points))));
%hold on

end
```

É mostrada na Figura 8.22 o erro entre a solução exata e a aproximada pela `Tau Toolbox`, conforme executada pelo Código 8.12.

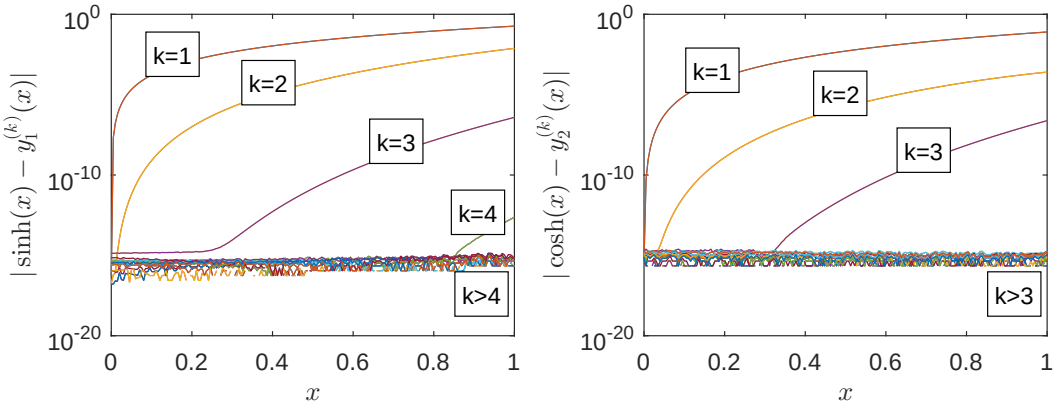

Figura 8.22: Erro entre a solução exata e aproximada para o Exemplo 8.14.

Resulta evidente deste exemplo a estabilidade do método tau na implementação em `Tau Toolbox`, uma vez que mesmo para $k > 4$ (propositalmente executadas) e grau 19, o método atinge a precisão da máquina sem degradação da qualidade de solução.

8.3 Conclusões e considerações

Apresentados diversos problemas integro-diferenciais, de coeficientes polinomiais e não polinomiais, lineares e não lineares. A `Tau Toolbox` mostra-se estável e eficiente no sentido de que as implementações são baseadas nos contributos abordados na Parte II desta Tese, e fornecem geralmente resultados com erros na precisão da máquina ou muito próximos. A listagem completa das funções da `Tau Toolbox`, é apresentada no Apêndice B.

Capítulo 9

Conclusão e trabalho futuro

Desenvolvemos alguns importantes contributos de estabilidade para o método tau de Lanczos aplicado a sistemas de equações diferenciais e integro-diferenciais, lineares e não lineares, sendo estes (i) a avaliação em bases ortogonais de polinómios, (ii) o cálculo recursivo da matriz de mudança de base, (iii) o cálculo recursivo do produto em bases ortogonais, (iv) a utilização dos coeficientes de linearização para problemas não lineares, (v) a obtenção das matrizes que representam nos coeficientes de polinómios as operações do produto por um monómio, de derivação e de integração diretamente em bases ortogonais, (vi) o esquema iterativo baseado em complementos de Schur e (vii) a aproximação de coeficientes não polinomiais por interpolação ortogonal, por função de matrizes e por recurso ao próprio método tau.

Desenvolvemos a `Tau Toolbox` (www.fc.up.pt/tautoolbox), um conjunto de rotinas escritas em `MATLAB`, que contém todos os contributos de estabilidade supra mencionados, e portanto aptas a aproximar pelo método tau diversos tipos de problemas. Com as rotinas implementadas na `Tau Toolbox`, automatizamos o processo que permite generalizar o método tau a problemas integro-diferenciais não lineares e com coeficientes não polinomiais. A implementação recursiva do método, com grau incremental, permite utilizar o método com seleção automática do grau em função de um limite para o erro pré-estabelecido. A opção por uma aproximação global ou pela aproximação polinomial por bocados também foi automatizada, permitindo resolver problemas em longos intervalos de tempo, nomeadamente sistemas dinâmicos não lineares. A representação de resultados na forma gráfica, simbólica e numérica foi implementada, sendo possível visualizar graficamente as soluções aproximadas, os erros, as diferenças entre soluções de grau consecutivo e a estrutura de esparsidade das matrizes associadas aos problemas.

Confrontando os resultados obtidos na presente Tese com alguns problemas da literatura encontramos sempre resultados melhores para o tipos de problemas que abordamos, e portanto acreditamos na relevância da teoria aqui desenvolvida e na biblioteca de rotinas criada.

Como trabalho futuro, pretendemos desenvolver técnicas e núcleos computacionais para o método tau (i) para equações diferenciais em derivadas parciais, fracionárias e com atraso; (ii) com bases de polinómios em variável complexa; (iii) na versão parcelar com passo adaptativo, como proposto por Rodrigues and Matos (2012) e (iv) com versões de programação paralela de alguns dos núcleos computacionais.

Apêndice A

Sistemas de equações lineares algébricas e precondicionamento

Sumário

Este Apêndice tem o objetivo de apresentar mais detalhadamente como obter a solução dos sistemas de equações lineares algébricas associados ao uso do método tau, uma vez que as matrizes utilizadas na formulação operacional do método herdam o mau condicionamento (ver A.1.2) das matrizes $\mathsf{N}_{\mathcal{P}}$, $\mathsf{M}_{\mathcal{Z}}$ e $\mathsf{O}_{\mathcal{Z}}$. Como será visto nas secções seguintes, melhores resultados são obtidos quando o sistema é substituído por outro, com igual conjunto de solução, mas com matriz dos coeficientes com um número de condição

menor. A esta técnica chama-se precondicionamento. Serão explorados aqui os tópicos necessários ao domínio do assunto.

Saad (2003) mostra detalhadamente diversos métodos de solução de sistemas de equações lineares algébricas, e portanto, será uma das principais referências deste Apêndice. Os métodos aqui propostos podem ser classificados em três tipos, a saber: diretos (eliminação de Gauss e fatorização LU); iterativos básicos (Jacobi, Gauss-Seidel e SOR); e os iterativos baseados em subespaços de Krylov (GMRES, CG, BiCG e BiCGstab).

A.1 Preliminares

Usando a notação matricial, um sistema linear de equações algébricas pode ser descrito pela equação matricial $\mathsf{Ax} = \mathsf{b}$, onde A é uma matriz de coeficientes de tamanho $m \times n$, b é o vetor de termos independentes de tamanho $m \times 1$ e x, $n \times 1$, é o vetor das incógnitas que soluciona a equação, ou seja

$$\mathsf{Ax} = \mathsf{b} \equiv \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}.$$

Sistemas de equações lineares podem ter uma solução única, não ter solução ou ter infinitas soluções, o que está diretamente relacionado com a singularidade da matriz A (quando $m = n$), de tal forma que só terá uma solução única se a matriz A for não singular. Muitos sistemas de equações lineares algébricas de dimensões não elevadas e com número de condição pequeno podem ser facilmente resolvidos com o uso da matriz inversa, $\mathsf{x} = \mathsf{A}^{-1}\mathsf{b}$, todavia o processo de calcular a inversa, sobretudo para sistemas de grandes dimensões, é um processo a evitar pois é numericamente instável. Também não tem vantagem em relação aos custos: o cálculo de $\mathsf{x} = \mathsf{A}^{-1}\mathsf{b}$ requer $2n^3$ operações em vírgula flutuante, com A^{-1} calculado por eliminação de Gauss com pivotagem parcial, e a aplicação do mesmo algoritmo diretamente a $\mathsf{Ax} = \mathsf{b}$ requer um custo computacional 3 vezes menor: $\frac{2n^3}{3}$ operações em vírgula flutuante.

A.1.1 Norma matricial

Para poder mensurar os erros e a sensibilidade que envolvem a solução de um sistema linear, deve estar definida a noção de "tamanho" de vetores e matrizes. O conceito escalar de magnitude, módulo, ou valor absoluto pode ser generalizado para o conceito de norma para vetores e matrizes (Heath, 2002). Uma norma-p para um vetor $\mathsf{x} = [x_1, \ldots, x_n]^T$, com $p > 0$ é definida por

$$||\mathsf{x}||_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p},$$

e alguns casos especiais são

$$||\mathsf{x}||_1 = \sum_{i=1}^{n} |x_i|, \quad ||\mathsf{x}||_2 = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{1/2} \quad \text{e} \quad ||\mathsf{x}||_\infty = \max_{1 \leq i \leq n} |x_i|,$$

respetivamente para a norma-1, norma-2 (correspondente a usual distância no espaço euclidiano) e a norma do infinito.

A norma matricial subordinada à norma-p define-se como sendo a função $|| \cdot || : \mathbb{R}^{n \times n} \to \mathbb{R}$ tal que

$$||\mathsf{A}||_p = \max_{\mathsf{x} \neq 0} \frac{||\mathsf{Ax}||_p}{||\mathsf{x}||_p}.$$

A norma matricial subordinada à norma de um vetor mede o máximo alongamento que esta faz em um vetor qualquer. As normas matriciais norma-1, norma-2, norma-∞ e a norma de Frobenius são dadas respetivamente por

$$||\mathsf{A}||_1 = \max_j \sum_{i=1}^{n} |a_{ij}|, \qquad ||\mathsf{A}||_2 = \max_{\mathsf{x} \neq 0} \frac{||\mathsf{Ax}||_2}{||\mathsf{x}||_2},$$

$$||\mathsf{A}||_\infty = \max_i \sum_{j=1}^{n} |a_{ij}|, \quad \text{e} \quad ||\mathsf{A}||_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}.$$

A.1.2 Condicionamento de um sistema linear

O número de condição de uma matriz mede quão próxima esta está de ser singular, através do rácio entre o maior prolongamento e o menor encurtamento que a matriz é capaz de fazer sobre um vetor não nulo. Grandes valores do número de condição refletem problemas mal condicionados. Um problema é tanto melhor condicionado quanto o número de condição é próximo de 1. O número de condição é uma característica intrínseca à própria matriz, e nada tem a ver com o conceito fundamental de estabilidade (de um método numérico).

O número de condição de uma matriz quadrada, respeitando a uma determinada norma-p, é o número real

$$\text{cond}(\mathsf{A}) = ||\mathsf{A}|| \cdot ||\mathsf{A}^{-1}|| = \left(\max_{\mathsf{x} \neq 0} \frac{||\mathsf{Ax}||}{||\mathsf{x}||} \right) \cdot \left(\min_{\mathsf{x} \neq 0} \frac{||\mathsf{Ax}||}{||\mathsf{x}||} \right)^{-1}.$$

Algumas propriedades do número de condição de uma matriz são:

1. Para qualquer matriz A, $\text{cond}(\mathsf{A}) \geq 1$.
2. Para a matriz identidade, $\text{cond}(\mathsf{I}) = 1$.
3. Para qualquer matriz de permutação P, $\text{cond}(\mathsf{P}) = 1$.
4. Para qualquer matriz A e um escalar não nulo γ, $\text{cond}(\gamma\mathsf{A}) = \text{cond}(\mathsf{A})$.

5. Para qualquer matriz diagonal $\mathsf{D} = \text{diag}(d_i)$, $\text{cond}(\mathsf{D}) = \dfrac{\max |d_i|}{\min |d_i|}$.

Com o intuito de mostrar os efeitos sobre o resultado fornecido ao solucionar um sistema provido de uma matriz A mal condicionada, criamos o seguinte problema: Definimos um sistema $\mathsf{Ax} = \mathsf{b}$ de tal forma que a solução seja $\mathsf{x} = [1,\ 1,\ \ldots,\ 1]^T$ e a matriz A define-se como sendo muito próxima a singular, e para isso criamos suas linhas "quase" como combinações lineares uma das outras, salvo por um acréscimo de uma pequena parcela $r_{i,j}$, para de facto não ser singular.

Código A.1. Exemplo de matriz mal condicionada.

```
% Tamanho do sistema a resolver.
n = 5;

% Para perturbar as linhas de A.
v = 1E-14;

% Criando a primeira linha da matriz A como aleatória.
A = 10*rand(1, n);

% Criando as demais linhas de A quase como combinações lineares.
for i = 2:n, A(i, :) = i*A(1, :) + v*rand(1, n); end

% Vetor b impondo que x = [1,1,...,1].
b = A*ones(n, 1);

% Obtendo a solução e o resíduo na solução.
x = A\b; r = b - A*x;
```

Ao executar o Código A.1, temos os seguintes resultados: uma matriz A muito próxima de singular

$$\mathsf{A} = \begin{bmatrix} 2,7603 & 6,7970 & 6,5510 & 1,6261 & 1,1900 \\ 5,5205 & 13,5941 & 13,1020 & 3,2522 & 2,3800 \\ 8,2808 & 20,3911 & 19,6529 & 4,8784 & 3,5699 \\ 11,0410 & 27,1881 & 26,2039 & 6,5045 & 4,7599 \\ 13,8013 & 33,9851 & 32,7549 & 8,1306 & 5,9499 \end{bmatrix},$$

com um número de condição bastante elevado $(2719 \cdot 10^{17})$ que faz com que a solução calculada seja irrelevante (ocorre a perda de cerca de 39 dígitos significativos).

O efeito que A mal condicionada causa, pode ser visto quando se calcula a solução do sistema $\mathsf{Ax} = \mathsf{b}$ e obtemos, erroneamente

$$\mathsf{x} = \begin{bmatrix} 4,6641, & 0,2338, & 0,3503, & 1,6739, & -0,4665 \end{bmatrix}^T,$$

e mesmo assim um resíduo "excelente"

$$\mathsf{r} = (1 \cdot 10^{-14}) \begin{bmatrix} 0,3553, & 0,7105, & 0, & 0, & 0 \end{bmatrix}^T.$$

A.1.3 Precisão da solução

Uma forma para verificar se a solução de um sistema de equações lineares algébricas é válida, é calcular o resíduo r. Para isso, seja $\tilde{\mathsf{x}}$ a solução encontrada via algum método numérico, então r é dado por

$$\mathsf{r} = \mathsf{b} - \mathsf{A}\tilde{\mathsf{x}}.$$

Teoricamente, se A é uma matriz não singular, temos que

$$||\tilde{\mathsf{x}} - \mathsf{x}|| = 0 \iff ||\mathsf{r}|| = 0,$$

todavia em situações práticas estas quantidades não são necessariamente pequenas simultaneamente (Heath, 2002). Considerando que existe uma perturbação $\delta\mathsf{A}$ de tal forma que a solução aproximada $\tilde{\mathsf{x}}$ do problema $\mathsf{Ax} = \mathsf{b}$ satisfaça

$$(\mathsf{A} + \delta\mathsf{A})\tilde{\mathsf{x}} = \mathsf{b},$$

então temos que

$$||\mathsf{r}|| = ||\mathsf{b} - \mathsf{A}\tilde{\mathsf{x}}|| = ||\delta\mathsf{A}\tilde{\mathsf{x}}|| \leq ||\delta\mathsf{A}|| \cdot ||\tilde{\mathsf{x}}||,$$

de onde se obtém a desigualdade que relaciona o resíduo relativo com uma variação relativa na matriz A, ou seja

$$\frac{||\mathsf{r}||}{||\mathsf{A}|| \cdot ||\tilde{\mathsf{x}}||} \leq \frac{||\delta\mathsf{A}||}{||\mathsf{A}||}.$$

Logo, um resíduo grande implica um erro grande, e portanto o algoritmo usado é instável, ou seja, um algoritmo estável produz sempre uma solução com resíduo pequeno. O erro e o resíduo relacionam-se também por

$$||\tilde{\mathsf{x}} - \mathsf{x}|| = ||\mathsf{A}^{-1}(\mathsf{A}\tilde{\mathsf{x}} - \mathsf{b})|| = || - \mathsf{A}^{-1}\mathsf{r}|| \leq ||\mathsf{A}^{-1}|| \cdot ||\mathsf{r}||$$

e logo

$$\frac{||\tilde{\mathsf{x}} - \mathsf{x}||}{||\tilde{\mathsf{x}}||} \leq \text{cond}(\mathsf{A})\frac{||\mathsf{r}||}{||\mathsf{A}|| \cdot ||\mathsf{x}||}.$$

Assim m resíduo pequeno implica um erro relativo pequeno na solução se e somente se A é bem condicionada.

Realizando uma perturbação $\delta\mathsf{b}$ no vetor b do sistema, ou uma perturbação $\delta\mathsf{A}$ na matriz A do sistema, temos que a solução x do problema será alterada respetivamente para $(\mathsf{x} + \delta\mathsf{x}_\mathsf{b})$ e $(\mathsf{x} + \delta\mathsf{x}_\mathsf{A})$. Com isso temos, respetivamente

$$\mathsf{A}(\mathsf{x} + \delta\mathsf{x}_\mathsf{b}) = \mathsf{b} + \delta\mathsf{b} \quad \text{e} \quad (\mathsf{A} + \delta\mathsf{A})(\mathsf{x} + \delta\mathsf{x}_\mathsf{A}) = \mathsf{b}.$$

Desenvolvendo, simultaneamente, ambas as equações temos

$$\mathsf{Ax} + \mathsf{A}\delta\mathsf{x}_\mathsf{b} = \mathsf{b} + \delta\mathsf{b} \quad \text{e} \quad \mathsf{Ax} + \mathsf{A}\delta\mathsf{x}_\mathsf{A} + \delta\mathsf{A}(\mathsf{x} + \delta\mathsf{x}_\mathsf{A}) = \mathsf{b}$$

$$\mathsf{Ax} = \mathsf{b} \Rightarrow \mathsf{A}\delta\mathsf{x}_\mathsf{b} = \delta\mathsf{b} \Rightarrow \delta\mathsf{x}_\mathsf{b} = \mathsf{A}^{-1}\delta\mathsf{b} \quad \text{e} \quad \mathsf{Ax} = \mathsf{b} \Rightarrow \delta\mathsf{x}_\mathsf{A} = -\mathsf{A}^{-1}\delta\mathsf{A}(\mathsf{x} + \delta\mathsf{x}_\mathsf{A})$$

$$\mathsf{b} = \mathsf{Ax} \Rightarrow ||\mathsf{b}|| \leq ||\mathsf{A}|| \cdot ||\mathsf{x}||$$

$$\delta\mathsf{x_b} = \mathsf{A}^{-1}\delta\mathsf{b} \Rightarrow ||\delta\mathsf{x_b}|| \leq ||\mathsf{A}^{-1}|| \cdot ||\delta\mathsf{b}|| \quad \text{e} \quad ||\delta\mathsf{x_A}|| \leq ||\mathsf{A}^{-1}|| \cdot ||\delta\mathsf{A}|| \cdot ||\mathsf{x} + \delta\mathsf{x_A}||$$

e portanto

$$\frac{||\delta\mathsf{x_b}||}{||\mathsf{x}||} \leq \text{cond}(\mathsf{A})\frac{||\delta\mathsf{b}||}{||\mathsf{b}||} \quad \text{e} \quad \frac{||\delta\mathsf{x_A}||}{||\mathsf{x} + \delta\mathsf{x_A}||} \leq \text{cond}(\mathsf{A})\frac{||\delta\mathsf{A}||}{||\mathsf{A}||}.$$

Estas relações são capazes de fornecer informações sobre o mal condicionamento de A, quando mostrarem que uma perturbação grande ocorre na solução, para uma pequena perturbação $\delta\mathsf{b}$ ou $\delta\mathsf{A}$. Logo, pequenas perturbações relativas em b e/ou A podem conduzir a grandes erros (relativos) nas soluções. Com o valor de cond(A) podemos estimar quantos dígitos se perdem ao resolver um sistema de equações lineares (Heath, 2002). Seja d_p a quantidade de dígitos perdidos, temos então $d_p = \log_{10}(\text{cond}(\mathsf{A}))$.

A.2 Métodos diretos

Os métodos diretos para a resolução de sistemas de equações lineares algébricas caracterizam-se por conter um número finito de operações, dependente apenas do tamanho da matriz A e do vetor b. Estas operações conduzem exatamente à solução (caso ela exista), e não são usadas aproximações iniciais (Burden and Faires, 1993).

A.2.1 Método da eliminação de Gauss

Dado um sistema $\mathsf{Ax} = \mathsf{b}$, definimos por matriz aumentada a matriz $\overline{\mathsf{A}} = [\mathsf{A},\ \mathsf{b}] = [a_{ij}]_{n\times(n+1)}$, e chamamos de E_i a equação referente à linha i de $\overline{\mathsf{A}}$ e portanto $E_i \equiv \sum_{j=1}^{n} a_{ij}x_j = a_{i,n+1}$. Considerando que $a_{jj} \neq 0$, com as operações

$$E_i = E_i - \frac{a_{ik}}{a_{kk}}E_k, \quad k = 1(1)n-1, \quad i = k+1(1)n, \quad j = k(1)n$$

reescrevemos os elementos de $\overline{\mathsf{A}}$ de forma a anular todos os coeficientes em que $i > j$, fazendo com que a porção de $\overline{\mathsf{A}}$ referente a A seja triangular superior, e portanto os elementos x_i do vetor solução são encontrados por substituição retroativa

$$\begin{cases} x_n = \dfrac{a_{n,n+1}}{a_{nn}} \\ x_i = \dfrac{a_{i,n-1} - \displaystyle\sum_{j=i+1}^{n} a_{ij}x_j}{a_{ii}}, \quad i = n-1(-1)1 \end{cases}.$$

A.2.2 Fatorização LU

Dado um sistema $\mathsf{Ax} = \mathsf{b}$, a fatorização LU da matriz de coeficientes A consiste em obter duas matrizes: L (triangular inferior) e U (triangular superior) satisfazendo $\mathsf{A} = \mathsf{LU}$, e que portanto permitem expressar o sistema na forma

$$\mathsf{LUx} = \mathsf{b} \Leftrightarrow \begin{cases} \mathsf{Ly} = \mathsf{b} \Leftrightarrow y_i = b_i - \displaystyle\sum_{j=1}^{i-1} l_{ij} y_j, \quad i = 1(1)n \\ \\ \mathsf{Ux} = \mathsf{y} \Leftrightarrow x_i = \dfrac{y_i - \displaystyle\sum_{j=i+1}^{n} u_{ij} x_j}{u_{ii}}, \quad i = n(-1)1 \end{cases},$$

ou seja, o problema é transformado na resolução de dois sistemas triangulares: um por substituição progressiva e outro retroativa. Uma das vantagens deste processo está no facto de que múltiplas soluções com diferentes vetores b podem ser calculadas sem cálculos adicionais como nos casos da eliminação de Gauss.

Aplicando o mesmo processo de eliminação de Gauss (mas sem alterar os valores do vetor b) obtemos a matriz U, e ao passo que vamos obtendo U passamos a armazenar na matriz L os coeficientes

$$l_{ik} = \frac{a_{ik}}{a_{kk}}, \quad k = 1(1)n-1, \quad i = k+1(1)n.$$

Se $l_{ii} = 1,\ i = 1(1)n$, então $\mathsf{LU} = \mathsf{A}$.

A.2.3 Estabilidade da fatorização LU

As técnicas mostradas anteriormente falham se durante o processo for encontrado $a_{ii} = 0$ (pivô nulo), pois haverá divisão por zero. Por outro lado, se a_{ii} for suficientemente próximo de zero conduzirá a erros de arredondamento uma vez que sua grandeza é pequena comparada aos demais elementos.

Nestas condições é vantajoso procurar, em cada coluna, o maior coeficiente nas linhas abaixo do elemento pivô, e então efetuar a troca das linhas antes de cada etapa $k = 1(1)n-1$ de fatorização (a permutação das equações em nada altera o sistema). A esta técnica chamamos pivotagem parcial. Caso procuremos por um maior pivô não somente nas linhas mas também nas colunas, podemos minimizar ainda mais os erros de arredondamento (a permutação de colunas altera a ordem das variáveis incógnitas, e uma matriz de permutação é necessária para as reorganizar). A esta técnica chamamos pivotagem completa.

Seja A uma matriz $n \times n$ e $\mathsf{A} = \mathsf{LU}$ a fatorização LU de A com aritmética exata. Na precisão finita os cálculos irão aproximar L por $\tilde{\mathsf{L}}$ e U por $\tilde{\mathsf{U}}$. Vamos assumir que $\tilde{\mathsf{L}}\tilde{\mathsf{U}} = \mathsf{A} + \delta\mathsf{A}$. Na aritmética de máquina de precisão dupla ϵ, o limite superior do erro relativo devido ao arredondamento em aritmética de vírgula flutuante é $2,2 \cdot 10^{-16}$. Então os erros relativos para armazenar a matriz A são dados por

$$\max_{i,j} |e_{ij}| \leq \epsilon \max_{i,j} |a_{ij}|,$$

para $\delta\mathsf{A} = (e_{ij})$.

Relembremos que numa análise de erro, um algoritmo é estável quando produz a solução exata de um problema ligeiramente perturbado.

Definição A.1. *Um algoritmo $\tilde{f}$ para um problema f é estável se para cada x $f(\tilde{x}) = f(x)$ para algum $\tilde{x}$ com*

$$\frac{\|\tilde{x} - x\|}{\|x\|} = \mathcal{O}(\epsilon).$$

Teorema A.1. *Para a eliminação de Gauss sem pivotagem em computador, os calculados $\tilde{\mathsf{L}}$ e $\tilde{\mathsf{U}}$ satisfazem*

$$\tilde{\mathsf{L}}\tilde{\mathsf{U}} = \mathsf{A} + \delta\mathsf{A}, \quad \frac{\|\delta\mathsf{A}\|}{\|\mathsf{L}\|\|\mathsf{U}\|} \leq p(n)\epsilon = \mathcal{O}(\epsilon),$$

onde $p(n)$ depende da dimensão n mas não das entradas de matriz.

Demonstração. Conforme Demmel (1997), sabemos que

$$u_{jk} = a_{jk} - \sum_{i=1}^{j-1} l_{ji}u_{ik},\ j \leq k \quad \text{e} \quad l_{jk} = \frac{a_{jk} - \sum_{i=1}^{k-1} l_{ji}u_{ik}}{u_{kk}},\ j > k.$$

Em aritmética de vírgula flutuante temos, para $i \leq k$,

$$u_{jk} = \left(a_{jk} - \sum_{i=1}^{j-1} l_{ji}u_{ik}(1+\delta_i)\right)(1+\delta'), \quad \text{com} \quad \begin{cases} |\delta_i| \leq (j-1)\epsilon \\ |\delta'| \leq \epsilon \end{cases},$$

e então

$$\begin{aligned} a_{jk} &= \frac{u_{jk}}{1+\delta'}l_{jj} + \sum_{i=1}^{j-1} l_{ji}u_{ik}(1+\delta_i), \quad \text{uma vez que} \quad l_{jj} = 1 \\ &= \sum_{i=1}^{j} l_{ji}u_{ik} + \underbrace{\sum_{i=1}^{j} l_{ji}u_{ik}\delta_i}_{E_{jk}}, \quad 1+\delta_j = \frac{1}{1+\delta'}, \end{aligned}$$

e tomando as normas (1, ∞, Frobenius: uma vez que $\||\mathsf{X}|\| = \|\mathsf{X}\|$)

$$||\mathsf{E}|| \leq n\epsilon||\mathsf{L}||||\mathsf{U}||.$$

De forma análoga temos, para $j \geq k$,

$$l_{jk} = \left(\frac{\left(a_{jk} - \sum_{i=1}^{k-1} l_{ji}u_{ik}(1+\delta_i)\right)(1+\delta')}{u_{kk}}\right)(1+\delta''), \quad \text{com} \quad \begin{cases} |\delta_i| \leq (k-1)\epsilon \\ |\delta'| \leq \epsilon \\ |\delta''| \leq \epsilon \end{cases},$$

e então

$$a_{jk} = \frac{u_{kk}l_{jk}}{(1+\delta')(1+\delta'')} + \sum_{i=1}^{k-1} l_{ji}u_{ik}(1+\delta_i)$$

$$= \sum_{i=1}^{k} l_{ji}u_{ik} + \underbrace{\sum_{i=1}^{k} l_{ji}u_{ik}\delta_i}_{E_{jk}}, \quad 1+\delta_k = \frac{1}{(1+\delta')(1+\delta'')}, \quad |\delta_i| \leq n\epsilon.$$

Tomando as normas (1, ∞, Frobenius)

$$||\mathsf{E}|| \leq n\epsilon||\mathsf{L}||||\mathsf{U}||.$$

Agora, o sistema $\mathsf{LUx} = \mathsf{y}$, $\mathsf{Ly} = \mathsf{b}$, $\mathsf{Ux} = \mathsf{y}$ deve também ser resolvido com aritmética de vírgula flutuante:

$$(\mathsf{L}+\delta\mathsf{L})\tilde{\mathsf{y}} = \mathsf{b}, \quad |\delta\mathsf{L}| \leq n\epsilon|\mathsf{L}|$$

$$(\mathsf{U}+\delta\mathsf{U})\tilde{\mathsf{x}} = \tilde{\mathsf{y}}, \quad |\delta\mathsf{U}| \leq n\epsilon|\mathsf{U}|,$$

então

$$\begin{aligned}
\mathsf{b} &= (\mathsf{L}+\delta\mathsf{L})\tilde{\mathsf{y}} \\
&= (\mathsf{L}+\delta\mathsf{L})(\mathsf{U}+\delta\mathsf{U})\tilde{\mathsf{x}} \\
&= (\mathsf{LU}+\mathsf{L}\delta\mathsf{U}+\delta\mathsf{LU}+\delta\mathsf{L}\delta\mathsf{U})\tilde{\mathsf{x}} \\
&= (\mathsf{A}-\mathsf{E}+\mathsf{L}\delta\mathsf{U}+\delta\mathsf{LU}+\delta\mathsf{L}\delta\mathsf{U})\tilde{\mathsf{x}} \\
&= (\mathsf{A}+\delta\mathsf{A})\tilde{\mathsf{x}}, \quad \delta\mathsf{A} = -\mathsf{E}+\mathsf{L}\delta\mathsf{U}+\delta\mathsf{LU}+\delta\mathsf{L}\delta\mathsf{U}
\end{aligned}$$

e assim

$$\begin{aligned}
||\delta\mathsf{A}|| &\leq ||\mathsf{E}|| + ||\mathsf{L}\delta\mathsf{U}|| + ||\delta\mathsf{LU}|| + ||\delta\mathsf{L}\delta\mathsf{U}|| \\
&\leq n\epsilon||\mathsf{L}||||\mathsf{U}|| + 2n\epsilon||\mathsf{L}||||\mathsf{U}|| + n^2\epsilon^2||\mathsf{L}||||\mathsf{U}|| \\
&\leq 3n\epsilon||\mathsf{L}||||\mathsf{U}||.
\end{aligned}$$

□

Se $3n\epsilon\|\mathsf{L}\|\|\mathsf{U}\| = \mathcal{O}(\|\mathsf{A}\|)$ então o algoritmo é estável.

Definição A.2. *O factor de crescimento ρ_n da matriz A com dimensão $n \times n$ da eliminação de Gauss é dado por*

$$\rho_n = \frac{\max_{ij}|u_{ij}|}{\max_{ij}|a_{ij}|}.$$

Teorema A.2. *Para a eliminação de Gauss com pivotagem parcial em computador, os calculados $\tilde{\mathsf{L}}$ e $\tilde{\mathsf{U}}$ satisfazem*

$$\tilde{\mathsf{L}}\tilde{\mathsf{U}} = \tilde{\mathsf{P}}(\mathsf{A}+\delta\mathsf{A}), \quad \frac{\|\delta\mathsf{A}\|}{\|\mathsf{A}\|} \leq p(n)\rho_n\epsilon = \mathcal{O}(\rho_n\epsilon)$$

onde $\tilde{\mathsf{P}}$ é a matriz de permutação calculada, $p(n) = 3n^3$.

Demonstração. Notemos que com pivotagem parcial $\|\mathsf{L}\| \leq n$ (norma 1, ∞ e Frobenius) e $\|\mathsf{U}\| \leq n\rho_n\|\mathsf{A}\|$. □

Com pivotagem parcial, o fator de crescimento pode ainda ser tão grande quanto 2^{n-1} (no pior caso o tamanho das entradas podem dobrar em cada estágio), mas isto é extremamente raro. Na prática, há um baixo ou nulo crescimento no tamanho das entradas (Higham and Higham, 1989).

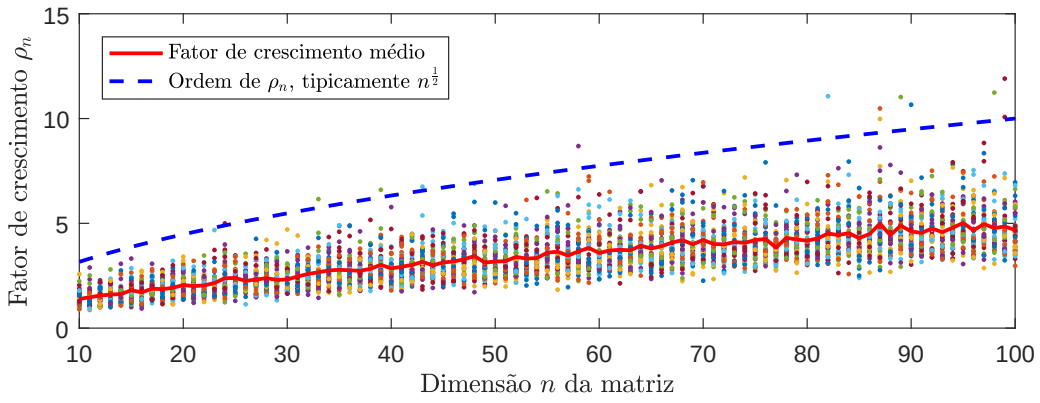

Figura A.1: Fatores de crescimento ρ_n para eliminação de Gauss para $n = 10(1)100$.

A Figura A.1 apresenta os fatores de crescimento médio ρ_n (em 50 experimentos) para a eliminação de Gauss com pivotagem parcial. O resultado evidência que o crescimento médio é menor que $n^{\frac{1}{2}}$ (raramente $n^{\frac{2}{3}}$).

Os códigos a seguir aplicam eliminação de Gauss e fatorização LU, ambos sem e com pivotagem parcial e completa, sendo estes parâmetros das rotinas.

Código A.2. Solução de um sistema de equações lineares algébricas por eliminação de Gauss.

```
function x = SYS_GAUSS(A, b, p)
% Solução de sistemas lineares por eliminação de Gauss.
% p = 0, sem pivitagem
%     1, com pivotagem parcial
%     2, com pivotagem completa.

% Tamanho de A.
n = length(A);

% Matriz de permutação caso pivotagem comleta.
if p == 2, P = eye(n); end

% Eliminação de Gauss.
for k = 1:n-1
    if p == 1 % Pivotagem parcial.
        [A, b] = p_parcial(A, b, k);
    elseif p== 2 % Pivotagem completa.
        [A, b, P] = p_completa(A, b, k, P);
    end
    b(k+1:n) = b(k+1:n)-A(k+1:n, k)/A(k, k)*b(k);
    A(k+1:n, k:n) = A(k+1:n, k:n)-A(k+1:n, k)/A(k, k)*A(k, k:n);
```

```
end

% Alocando memória.
x = zeros(n, 1);

% Substituição retroativa.
x(n) = b(n)/A(n, n);
for i = n-1:-1:1
    x(i) = (b(i)-A(i, i+1:end)*x(i+1:end))/A(i, i);
end

% Solução (des)permutada.
if p == 2, x = P*x; end

function [A, b] = p_parcial(A, b, k)
v = max(abs(A(k:end, k))); v = find(abs(A(k:end, k)) == v);
v = v(1) + k - 1;
aux = A(v, :); A(v, :) = A(k, :); A(k, :) = aux;
aux = b(v); b(v) = b(k); b(k) = aux;

function [A, b, P] = p_completa(A, b, k, P)
mc = max(abs(A(k:end, k:end))); ml = max(mc);
vc = find(mc == ml);   vl = find(abs(A(k:end, vc+k-1)) == ml);
vc = vc + k - 1;       vl = vl + k - 1;
aux = A(vl, :); A(vl, :) = A(k, :); A(k, :) = aux;
aux = A(:, vc); A(:, vc) = A(:, k); A(:, k) = aux;
aux = P(:, vc); P(:, vc) = P(:, k); P(:, k) = aux;
aux = b(vl); b(vl) = b(k); b(k) = aux;
```

Código A.3. Solução de um sistema de equações lineares algébricas por fatorização LU.

```
function x = SYS_LU(A, b, p)
% Solução de sistemas lineares por fatozização LU.
% p = 0, sem pivitagem
%     1, com pivotagem parcial
%     2, com pivotagem completa.

% Tamanho de A.
n = length(A);

% Matriz de permutação caso pivotagem comleta.
if p == 2, P = eye(n); end

% Alocando memória
U = A; L = eye(n);
y = zeros(n, 1); x = zeros(n, 1);

% Fatoriza A em L e U.
for k = 1:n-1
    if p == 1 % Pivotagem parcial.
        [L, U, b] = p_parcial(L, U, b, k);
    elseif p== 2 % Pivotagem completa.
        [L, U, b, P] = p_completa(L, U, b, k, P);
    end
    L(k+1:n, k) = U(k+1:n, k)/U(k, k);
```

```
    U(k+1:n, k:n) = U(k+1:n, k:n)-L(k+1:n, k)*U(k, k:n);
end

% Substituição progressiva para Ly=b.
for i = 1:n
   y(i) = b(i)-L(i,1:i)*y(1:i);
end

% Substituição retroativa para Ux=y.
x(n) = y(n)/U(n,n);
for i = n-1:-1:1
   x(i) = (y(i)-U(i,i+1:end)*x(i+1:end))/U(i,i);
end

% Solução (des)permutada.
if p == 2, x = P*x; end

function [L, U, b] = p_parcial(L, U, b, k)
v = max(abs(U(k:end, k)));  v = find(abs(U(k:end, k)) == v);
v = v(1) + k - 1;
aux = U(v, :); U(v, :) = U(k, :); U(k, :) = aux;
aux = L(v, :); L(v, :) = L(k, :); L(k, :) = aux;
aux = b(v); b(v) = b(k); b(k) = aux;

function [L, U, b, P] = p_completa(L, U, b, k, P)
mc = max(abs(U(k:end, k:end))); ml = max(mc);
vc = find(mc == ml);   vl = find(abs(U(k:end, vc+k-1)) == ml);
vc = vc + k - 1;       vl = vl + k - 1;
aux = U(vl, :); U(vl, :) = U(k, :); U(k, :) = aux;
aux = U(:, vc); U(:, vc) = U(:, k); U(:, k) = aux;
aux = L(vl, :); L(vl, :) = L(k, :); L(k, :) = aux;
aux = L(:, vc); L(:, vc) = L(:, k); L(:, k) = aux;
aux = P(:, vc); P(:, vc) = P(:, k); P(:, k) = aux;
aux = b(vl); b(vl) = b(k); b(k) = aux;
```

A.3 Métodos iterativos

Os métodos iterativos de resolução de um sistema de equações lineares algébricas resolvem o problema $\mathsf{Ax} = \mathsf{b}$ na forma

$$\mathsf{x}^{(k+1)} = \mathsf{H}\mathsf{x}^{(k)} + \mathsf{d}, \quad k \geq 0$$

onde $\mathsf{d} \in \mathbb{R}^n$, $\mathsf{H} \in \mathbb{R}^{n \times n}$ e $\mathsf{x}^{(0)} \in \mathbb{R}^n$ é uma aproximação inicial. A ideia desta classe de métodos não é obter a solução exata em um número finito de passos, mas sim uma aproximação da solução que satisfaça um critério de tolerância. Tal aproximação é calculada iterando o processo quantas vezes forem necessárias para atingir a referida precisão, e usando para tal, em cada iteração, a aproximação anterior para atualizar a nova. Todavia, para usar estes métodos, uma condição necessária e suficiente que garanta a convergência dos mesmos é enunciada:

Definição A.3. *Seja* A *uma matriz não singular associada a um sistema* $\mathsf{Ax} = \mathsf{b}$*, e*

$\sigma(\mathsf{A})$ *o conjunto de todos os seus valores próprios* λ*, chama-se então* **raio espetral***, e denota-se por* $\rho(\mathsf{A})$*, o número (Saad, 2003):*

$$\rho(\mathsf{A}) = \max_{\lambda \in \sigma(\mathsf{A})} |\lambda|.$$

Teorema A.3. *Um método iterativo* $\mathsf{x}^{(k+1)} = \mathsf{H}\mathsf{x}^{(k)} + \mathsf{d}$ *converge para a solução do sistema* $\mathsf{A}\mathsf{x} = \mathsf{b}$*, seja qual for o vetor inicial* $\mathsf{x}^{(0)}$*, se, e somente se, o raio espetral* $\rho(\mathsf{H})$ *satisfazer a desigualdade* $\rho(\mathsf{H}) < 1$*.*

Demonstração. Seja x^* a solução exata do sistema $\mathsf{A}\mathsf{x} = \mathsf{b}$, e $\mathsf{x}^{(k+1)}$ uma aproximação, temos que

$$\mathsf{x}^{(k+1)} - \mathsf{x}^* = \mathsf{H}\mathsf{x}^{(k)} + \mathsf{d} - (\mathsf{H}\mathsf{x}^* + \mathsf{d})$$

e então anulam-se apenas os vetores d resultando em

$$\mathsf{x}^{(k+1)} - \mathsf{x}^* = \mathsf{H}(\mathsf{x}^{(k)} - \mathsf{x}^*)$$

que pode ser escrito, por funcionar para o índice anterior, como

$$\mathsf{x}^{(k)} - \mathsf{x}^* = \mathsf{H}(\mathsf{x}^{(k-1)} - \mathsf{x}^*),$$

logo temos que

$$\begin{aligned} \mathsf{x}^{(k+1)} - \mathsf{x}^* &= \mathsf{H}^2(\mathsf{x}^{(k-1)} - \mathsf{x}^*) \\ &= \mathsf{H}^3(\mathsf{x}^{(k-2)} - \mathsf{x}^*) \\ &\vdots \\ &= \mathsf{H}^{k+1}(\mathsf{x}^{(0)} - \mathsf{x}^*), \end{aligned}$$

e assim podemos dizer que a sequência $[x_0,\ x_1,\ \ldots]$ converge para x^* se e somente se o $\lim_{k\to\infty} \mathsf{H}^k = 0$, e portanto $\rho(\mathsf{H}) < 1$. □

A.3.1 Métodos iterativos estacionários de Jacobi, Gauss-Seidel e SOR

Para estes três métodos podemos começar com a decomposição da matriz A do sistema $\mathsf{A}\mathsf{x} = \mathsf{b}$ de tal forma que

$$\mathsf{A} = \mathsf{D} - \mathsf{E} - \mathsf{F},$$

onde D representa a diagonal de A e, $-\mathsf{E}$ e $-\mathsf{F}$ as partes estritamente inferior e superior, respetivamente, à diagonal D.

O método de Jacobi consiste primeiramente em isolar as n incógnitas x_n, de tal forma que a equação $i,\ i = 1(1)n$, tenha a variável x_i isolada, ou seja

$$x_i^{(k+1)} = \frac{1}{a_{i,i}} \left(b_i - \sum_{j=1, j\neq i}^{n} a_{i,j} x_j^{(k)} \right), \quad i = 1(1)n, \tag{A.1}$$

onde $a_{i,j}$ são os elementos de A e b_i os de b. A partir de um vetor inicial de aproximação $\mathsf{x}^{(0)}$, o método segue iterativamente, até que se atinja a precisão necessária, com a renovação deste vetor, obtendo $\mathsf{x}^{(k+1)}$ a partir da aplicação de $\mathsf{x}^{(k)}$ no lado direito da equação (A.1).

A equação (A.1) pode ser escrita no formato matricial como

$$\mathsf{x}^{(k+1)} = \mathsf{D}^{-1}\Big(\mathsf{b} + (\mathsf{E}+\mathsf{F})\mathsf{x}^{(k)}\Big),$$

e portanto a matriz de iteração é $\mathsf{H} = \mathsf{D}^{-1}(\mathsf{E}+\mathsf{F})$.

Para o método de Gauss-Seidel a ideia de funcionamento é a mesma, a não ser pela diferença de que neste, em cada renovação do vetor $\mathsf{x}^{(k+1)}$ são usadas as componentes já conhecidas do mesmo, ou seja, atualizando-se imediatamente no processo iterativo, e portanto acelerando a convergência. Desta forma, cada componente do vetor aproximação da solução no método de Gauss-Seidel é obtido com

$$x_i^{(k+1)} = \frac{1}{a_{i,i}}\left(b_i - \sum_{j=1}^{i-1} a_{i,j}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{i,j}x_j^{(k)}\right), \quad i = 1(1)n. \tag{A.2}$$

O formato matricial para (A.2) passa a ser

$$\mathsf{x}^{(k+1)} = \mathsf{D}^{-1}\Big(\mathsf{b} + \mathsf{E}\mathsf{x}^{(k+1)} + \mathsf{F}\mathsf{x}^{(k)}\Big),$$

o que conduz à matriz de iteração $\mathsf{H} = (\mathsf{D}-\mathsf{E})^{-1}\mathsf{F}$.

Por fim, o método SOR (Successive Over Relaxation) surge com a ideia de ponderar através de um parâmetro ω, o resultado obtido com Gauss-Seidel e a iteração anterior, assim obtendo cada iteração seguinte, da forma

$$x_i^{(k+1)} = \omega\frac{1}{a_{i,i}}\left(b_i - \sum_{j=1}^{i-1} a_{i,j}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{i,j}x_j^{(k)}\right) + (1-\omega)x_i^{(k)}, \quad i = 1(1)n. \tag{A.3}$$

O formato matricial para (A.3) passa a ser

$$\mathsf{x}^{(k+1)} = \omega\mathsf{D}^{-1}\Big(\mathsf{b} + \mathsf{E}\mathsf{x}^{(k+1)} + \mathsf{F}\mathsf{x}^{(k)}\Big) + (1-\omega)\mathsf{x}^{(k)},$$

e então a matriz de iteração vem a ser $\mathsf{H} = (\mathsf{D}-\omega\mathsf{E})^{-1}\Big(\omega\mathsf{F} + (1-\omega)\mathsf{D}\Big)$.

Os métodos estacionários, como o de Jacobi ou Gauss-Seidel, são fáceis de usar, mas a sua eficiência e aplicação está limitada para sistemas em que a matriz A seja diagonalmente dominante ou simétrica definida positiva. São métodos interessantes apenas para uma introdução aos métodos iterativos ou como precondicionadores para métodos não estacionários.

O método de Gauss-Seidel geralmente tem uma convergência mais rápida que o método de Jacobi, todavia não costuma ser competitivo com os métodos não estacionários. Já o método SOR acelera a convergência do método de Gauss-Seidel ($\omega > 1$,

sobre-relaxação) e pode garantir a convergência quando Gauss-Seidel falha ($0 < \omega < 1$, sub-relaxação). A velocidade da convergência SOR está diretamente relacionada com a escolha de ω, e uma condição necessária para que haja convergência no método SOR é ter $\omega \in]0,\ 2[$ (Saad, 2003).

A.3.2 Método dos gradientes conjugados

O método dos gradientes conjugados (CG) é um método iterativo de solução de um sistema $\mathsf{Ax} = \mathsf{b}$, onde A deve ser uma matriz simétrica definida positiva, quando não o for, uma transformação $\mathsf{A}^T\mathsf{A}$ garante que tal ocorra. Este método trabalha ao usar as propriedades de ortogonalidade dos resíduos $\mathsf{r}^{(k)}$ e de conjugado das direções $\mathsf{d}^{(k)}$. Considerando

$$\mathsf{x}^{(k+1)} = \mathsf{x}^{(k)} + \alpha_k \mathsf{d}^{(k)} \quad \text{e} \quad \mathsf{d}^{(k+1)} = \mathsf{r}^{(k+1)} + \beta_k \mathsf{d}^{(k)},$$

os resíduos evoluem ao longo das iterações k por

$$\mathsf{r}^{(k+1)} = \mathsf{b} - \mathsf{Ax}^{(k+1)} = \mathsf{r}^{(k)} - \alpha_k \mathsf{Ad}^{(k)}. \tag{A.4}$$

Como $\mathsf{r}^{(k)}$ tem de ser por definição ortogonal a $\mathsf{r}^{(k+1)}$, usamos o produto interno $\left\langle \mathsf{r}^{(k)}, \mathsf{r}^{(k+1)} \right\rangle$ para obter

$$\alpha_k = \frac{\left\langle \mathsf{r}^{(k)}, \mathsf{r}^{(k+1)} \right\rangle}{\left\langle \mathsf{d}^{(k)}, \mathsf{Ad}^{(k)} \right\rangle}. \tag{A.5}$$

Por outro lado, devendo $\mathsf{d}^{(k)}$ e $\mathsf{d}^{(k+1)}$ ser direções A-conjugadas, $\left\langle \mathsf{d}^{(k+1)}, \mathsf{Ad}^{(k)} \right\rangle = 0$, resulta que

$$\beta_k = -\frac{\left\langle \mathsf{r}^{(k+1)}, \mathsf{Ad}^{(k)} \right\rangle}{\left\langle \mathsf{d}^{(k)}, \mathsf{Ad}^{(k)} \right\rangle}, \tag{A.6}$$

e, de (A.4), escrevemos

$$\mathsf{Ad}^{(k)} = \frac{\mathsf{r}^{(k)} - \mathsf{r}^{(k+1)}}{\alpha_k}$$

e que, fazendo o produto interno com $\mathsf{r}^{(k+1)}$, resulta

$$\left\langle \mathsf{r}^{(k+1)}, \mathsf{Ad}^{(k)} \right\rangle = -\frac{\left\langle \mathsf{r}^{(k+1)}, \mathsf{r}^{(k+1)} \right\rangle}{\alpha_k} \tag{A.7}$$

Por fim, de (A.5), (A.6) e (A.7) obtemos uma forma alternativa, que é mais viável em termos de cálculos computacionais, para os β_k, a saber

$$\beta_k = \frac{\left\langle \mathsf{r}^{(k+1)}, \mathsf{r}^{(k+1)} \right\rangle}{\left\langle \mathsf{r}^{(k)}, \mathsf{r}^{(k)} \right\rangle}.$$

Código A.4. Solução de um sistema de equações lineares algébricas por GC.

```
function [x, k] = SYS_GC(A, b)
%SYS_GC   Resolve o sistema Ax=b pelo método dos gradientes conjugados.

% A precisa ser simétrica.
b = A'*b; A = A'*A;

% Dimensão da matriz A.
n = length(A);

% Tolerância para o critério de paragem.
tol = 1e-16;

k = 1;

% Aproximação do primeiro vetor.
x(k, :) = zeros(1, n);

% Cálculo do primeiro resíduo.
r(k, :) = b - A*x(k, :)';

% Cálculo da primeira direção.
p(k, :) = r(k, :);

% Ciclo enquanto não atingir a tolerância.
while norm(r(k, :)) > tol
    alfa(k) = (p(k, :)*r(k, :)')/(p(k, :)*A*p(k, :)');

    % Próxima aproximação x.
    x(k+1, :) = x(k, :) + alfa(k)*p(k, :);

    % Próxima aproximação r.
    r(k+1, :) = r(k, :) - alfa(k)*(A*p(k, :)')';
    beta(k) = (r(k+1, :)*r(k+1, :)')/(r(k, :)*r(k, :)');

    %Próxima aproximação p.
    p(k+1, :) = r(k+1, :) + beta(k)*p(k, :);
    k = k+1;
end

% O resultatdo encontrado.
x = x(end,:);
```

A.3.3 Método do resíduo mínimo generalizado

O método do resíduo mínimo generalizado (GMRES) é um método iterativo não estacionário do tipo Krylov, e tem como base minimizar a norma do resíduo sobre todos os vetores em $\mathsf{x}^{(0)} + \mathsf{K}_m$, onde K_m é o subespaço de Krylov mostrado na Definição A.4.

Definição A.4. *Dada uma matriz* A *e um vetor* r *chama-se subespaço de Krylov de dimensão* m *ao conjunto das combinações* $\mathsf{A}^k\mathsf{r}, \quad k = 0(1)m-1$. *Portando* $\mathcal{K}_m(\mathsf{A}, \mathsf{r}) = \left\langle \mathsf{r}, \mathsf{Ar}, \ldots, \mathsf{A}^{m-1}\mathsf{r} \right\rangle$. *A matriz de Krylov associada é* $\mathsf{K}_m(\mathsf{A}, \mathsf{r}) = [\mathsf{r}, \ \mathsf{Ar}, \ \ldots, \ \mathsf{A}^{m-1}\mathsf{r}]$, *ou seja, cada uma de suas colunas são as* m *combinações lineares geradas.*

Teorema A.4. *Seja* A *uma matriz quadrada, então um vetor* $\hat{\mathsf{x}}$ *é o resultado de um método de projeção sobre* $\mathcal{K}_m$ *ortogonal a* $\mathsf{AK}_m = \mathcal{K}_m(\mathsf{A}, \mathsf{Ar}^{(0)})$ *com vetor inicial* $\mathsf{x}^{(0)}$ *se e somente se minimiza a norma-2 do vetor resíduo sobre o subespaço afim* $\mathsf{x}^{(0)} + \mathcal{K}_m$, $R(\hat{\mathsf{x}}) = \min_{\mathsf{x} \in \mathsf{x}^{(0)} + K_m} R(\mathsf{x})$, $R(\mathsf{x}) = ||\mathsf{b} - \mathsf{Ax}||_2$.

Demonstração. Ver Saad (2003). □

O resultado ótimo mostrado no Teorema A.4 é explorado na minimização dos resíduos no GMRES (Saad, 2003), e a implementação do mesmo transforma tal minimização em um problema de mínimos quadrados em $\mathbb{R}^m$, onde a partir de uma base ortonormal V_m de K_m, e de um vetor $\mathsf{z} \in \mathsf{K}_m$, $\mathsf{z} = \sum_{j=1}^{m} \mathsf{y}_j \mathsf{v}_j^m$, onde v_j^m é j-ésima coluna de V_m, temos que

$$\min_{\mathsf{x} \in \mathsf{x}^{(0)} + \mathsf{K}_m(\mathsf{A}, \mathsf{r}^{(0)})} ||\mathsf{b} - \mathsf{Ax}||_2 = \min_{\mathsf{z} \in \mathsf{K}_m(\mathsf{A}, \mathsf{r}^{(0)})} ||\mathsf{r}^{(0)} - \mathsf{Az}||_2 = \min_{\mathsf{y} \in \mathbb{R}^m} ||\mathsf{r}^{(0)} - \mathsf{AV}_m \mathsf{y}||_2. \tag{A.8}$$

Uma forma de evitar a multiplicação das matrizes A e V_m em cada iteração, pois isto requer um custo computacional elevado, é aplicar o processo de Arnoldi, que neste caso é oriundo do processo de ortogonalização de Gram-Schmidt modificado aplicado à sequência de Krylov. Ao considerar que o primeiro vetor é

$$\mathsf{v}_1 = \frac{\mathsf{r}^{(0)}}{||\mathsf{r}^{(0)}||_2},$$

temos, para $j = 1(1)m - 1$ que

$$\mathsf{v}_{j+1} = \frac{\mathsf{Av}_j - \sum_{i=1}^{j} \left((\mathsf{Av}_j)^T \mathsf{v}_i \right) \mathsf{v}_i}{\left|\left| \mathsf{Av}_j - \sum_{i=1}^{j} \left((\mathsf{Av}_j)^T \mathsf{v}_i \right) \mathsf{v}_i \right|\right|_2}$$

Código A.5. Arnoldi com ortogonalização de Gram-Schmidt modificada.

```
function v = arnoldi(A, b)
%arnoldi   Aplica o processo de Arnoldi (com relação a A e b) com
%   ortogonalização de Gram-Schmidt modificada.

x0 = rand(10, 1);

% Dimensão m do subespaço de Krylov.
m = length(A);

% Cálculo do resíduo.
r0 = b - A*x0;

% Primeiro vetor do processo.
v(1, :) = r0/norm(r0); h = zeros(m+1, 1);

% Para cada novo vetor até m.
```

```
for j = 1:m
   t = A*v(j, :)';
   for i = 1:j
       % Elemento da matriz de Hessenberg.
       h(i, j) = t'*v(i, :)';
       t = t - h(i, j)*v(i, :)';
   end
   % Elemento da matriz de Hessenberg.
   h(j+1, i) = norm(t);
   % Gera próximo v.
   v(j+1, :) = t/h(j+1, i);
end
```

O processo mostrado no Código A.5 gera os elementos $h_{i,j} = \left\langle \mathsf{A}\mathsf{v}_j, \mathsf{v}_i \right\rangle$, que formam a matriz de Hessenberg $\overline{\mathsf{H}}_m = [h_{i,j}]_{(m+1)\times m}$ superior. Assim, considerando que e_m é o m-ésimo vetor da base canónica e obtendo a matriz truncada $\mathsf{H}_m = [h_{i,j}]_{m\times m}$, são satisfeitas as relações (Vasconcelos, 1998):

$$\begin{aligned} \mathsf{A}\mathsf{V}_m &= \mathsf{V}_{m+1}\overline{\mathsf{H}}_m \\ \mathsf{A}\mathsf{V}_m &= \mathsf{V}_m\mathsf{H}_m + h_{m+1,m}\mathsf{v}_{m+1}\mathsf{e}_m^T \\ \mathsf{V}_m^T\mathsf{A}\mathsf{V}_m &= \mathsf{V}_m^T(\mathsf{V}_m\mathsf{H}_m + \mathsf{v}_m\mathsf{e}_m^T) = \mathsf{H}_m \end{aligned} \tag{A.9}$$

e assim, a norma (A.8) é obtida ao considerar $\rho = ||\mathsf{r}^{(0)}||_2$, e a partir de (A.9) usando a ortonormalidade das colunas de V_{m+1}, escrevemos:

$$||\mathsf{r}^{(0)} - \mathsf{A}\mathsf{V}_m\mathsf{y}||_2 = ||\rho\mathsf{v}_1 - \mathsf{V}_{m+1}\overline{\mathsf{H}}_m\mathsf{y}||_2$$

$$||\mathsf{r}^{(0)} - \mathsf{A}\mathsf{V}_m\mathsf{y}||_2 = ||\mathsf{V}_{m+1}(\rho\mathsf{e}_1 - \overline{\mathsf{H}}_m\mathsf{y})||_2$$

$$||\mathsf{r}^{(0)} - \mathsf{A}\mathsf{V}_m\mathsf{y}||_2 = ||\rho\mathsf{e}_1 - \overline{\mathsf{H}}_m\mathsf{y}||_2.$$

O Código A.6 foi adaptado de Barrett et al. (1994), e trata-se do método do resíduo mínimo generalizado para a solução de sistemas lineares.

Código A.6. Solução de um sistema de equações lineares algébricas por GMRES.

```
function [x, k] = SYS_GMRES(A, x, b, M, nsave, max_it, tol)
%SYS_GMRES   Resolve o sistema Ax=b pelo método do resíduo mínimo
%   generealizado. M é um precondicionador, nsave é o número de iterações
%   entre os reinícios, max_it é o número máximo de iterações e tol é a
%   tolerância desejada.

% Norma de b deve ser não nula.
nb = norm(b); if  (nb == 0), nb = 1; end

% Sai se a solução já foi encontrada.
r = M\(b-A*x); erro = norm(r)/nb; if (erro < tol), return, end

n = size(A, 1);
V(1:n, 1:nsave+1) = zeros(n, nsave+1); H(1:nsave+1, 1:nsave) = ...
    zeros(nsave+1, nsave);
cs(1: nsave) = zeros(nsave, 1); sn(1: nsave) = zeros(nsave, 1);
```

```
e1 = zeros(n, 1); e1(1) = 1;

% Início das iterações.
for k = 1:max_it

    % Armazena o resíduo.
    r = M\(b-A*x); V(:, 1) = r/norm(r); s = norm(r)*e1;

    % Constrói a base ortonormal usando Gram-Schmidt.
    for i = 1:nsave
        w = M\(A*V(:, i));
        for k = 1:i, H(k, i)= w'*V(:, k);  w = w - H(k, i)*V(:, k); end
        H(i+1, i) = norm(w);  V(:, i+1) = w/H(i+1, i);

        % Aplica a rotação dada.
        for k = 1:i-1
            temp = cs(k)*H(k, i) + sn(k)*H(k+1, i);
            H(k+1, i) = -sn(k)*H(k, i) + cs(k)*H(k+1, i); H(k, i) = temp;
        end

        % Da i-ésima matriz de rotação, aproxima a norma do resíduo.
        [cs(i), sn(i)] = rotacionamatriz(H(i,i), H(i+1,i));
        temp = cs(i)*s(i);
        s(i+1) = -sn(i)*s(i); s(i) = temp;
        H(i, i) = cs(i)*H(i, i) + sn(i)*H(i+1, i); H(i+1, i) = 0;
        erro  = abs(s(i+1))/nb;

        % Atualiza a aproximação.
        if (erro ≤ tol), y = H(1:i, 1:i)\s(1:i); x = x + V(:, 1:i)*y;
            break
        end
    end

    if ( erro ≤ tol ), break, end;
    y = H(1:nsave, 1:nsave)\s(1:nsave);

    % Atualiza a aproximação e o resíduo.
    x = x + V(:, 1:nsave)*y; r = M\(b-A*x);
    s(i+1) = norm(r); erro = s(i+1)/nb;

    % Avalia a convergência.
    if ( erro ≤ tol ), break, end;
end

% Rotação matricial.
function [c, s] = rotacionamatriz(a, b)
    if (b == 0), c = 1; s = 0;
    elseif (abs(b) > abs(a)), p = a/b;  s = 1/sqrt(1+p^2); c = p*s;
    else, p = b/a; c = 1/sqrt(1+p^2); s = p*c;
    end
```

O método GMRES é aplicável para matrizes não simétricas e conduz ao menor resíduo para um número fixo de passos de iteração. A dificuldade com este método é o crescimento do espaço de Krylov onde o problema é projetado. A fim de limitar

o esforço de memória para acomodar espaço crescente, o método é implementado com reinicialização.

A.3.4 Método dos gradientes biconjugados

O método dos gradientes biconjugados (BiCG) pode ser derivado do algoritmo de biortogonalização de Lanczos que resolve não somente o sistema original, mas também um sistema $\mathsf{A}^T\mathsf{x}^* = \mathsf{b}^*$. O Algoritmo para o mesmo é um processo de projeção ortogonal de $\mathcal{K}_m(\mathsf{A}, \mathsf{v}_1)$ em $\mathcal{K}_m(\mathsf{A}^T, \mathsf{w}_1)$, tomando

$$\mathsf{v}_1 = \frac{\mathsf{r}^{(0)}}{||\mathsf{r}^{(0)}||_2}.$$

O vetor w_1 é escolhido arbitrariamente com a condição $\left\langle \mathsf{v}_1, \mathsf{w}_1 \right\rangle \neq 0$, por exemplo $\mathsf{w}_1 = \mathsf{v}_1$. Como há um sistema $\mathsf{A}^T\mathsf{x}^* = \mathsf{b}^*$, então w_1 pode ser escolhido como sendo o resíduo inicial $\mathsf{b}^* - \mathsf{A}^T\mathsf{x}^{*(0)}$. Partimos então da decomposição LDU da matriz T_m como

$$\mathsf{T}_m = \mathsf{L}_m\mathsf{U}_m,$$

que é renovada a cada iteração, e a definição da matriz

$$\mathsf{P}_m = \mathsf{V}_m\mathsf{U}_m^{-1}.$$

Por fim a solução pode ser expressa como

$$\mathsf{x}_m = \mathsf{x}^{(0)} + \mathsf{V}_m\mathsf{T}_m^{-1}(||\mathsf{r}^{(0)}||\mathsf{e}_1)$$

$$\mathsf{x}_m = \mathsf{x}^{(0)} + \mathsf{V}_m\mathsf{U}_m^{-1}\mathsf{L}_m^{-1}(||\mathsf{r}^{(0)}||\mathsf{e}_1)$$

$$\mathsf{x}_m = \mathsf{x}^{(0)} + \mathsf{P}_m\mathsf{L}_m^{-1}(||\mathsf{r}^{(0)}||\mathsf{e}_1).$$

Da mesma forma com que ocorre em CG, a aproximação x_m é renovada a partir de x_{m-1} e similarmente os vetores r_j e r_j^* estão, respetivamente, na mesma direção de v_{j+1} e w_{j+1}. Assim eles formam uma sequência biortogonal, e é possível definir, de forma similar, a matriz

$$\mathsf{P}_m^* = \mathsf{W}_m(\mathsf{L}_m^{-1})^T.$$

Assim, pela propriedade dos conjugados, P_m e P_m^* estão relacionadas da seguinte forma

$$(\mathsf{P}_m^*)^T\mathsf{A}\mathsf{P}_m = \mathsf{L}_m^{-1}\mathsf{W}_m^T\mathsf{A}\mathsf{V}_m\mathsf{U}_m^{-1} = \mathsf{L}_m^{-1}\mathsf{T}_m\mathsf{U}_m^{-1} = \mathsf{I},$$

e assim podemos escrever o Código A.7 para BiCG.

Código A.7. Solução de um sistema de equações lineares algébricas por BiCG.

```
function [x, k] = SYS_BiCG(A, b, tol)
%SYS_BGSTAB   Resolve o sistema Ax=b pelo método dos gradientes
%   biconjugados com tolerância tol.

% Dimensão da matriz A.
n = length(A);

% Vetor aleatório como aproximação inicial.
x(:, 1) = rand(n, 1);

% Cálculo do primeiro resíduo.
r(:, 1) = b - A*x;

% Vetor para r2 inicial.
r2(:, 1) = b - A'*x;

p(:, 1) = r; p2(:, 1) = r2; k = 1;
while norm(r(:, k)) > tol || k < 100
    a(k) = (r(:, k)'*r2(:, k))/((A*p(:, k))'*p2(:, k));

    % Atualiza o vetor solução.
    x(:, k+1) = x(:, k) + a(k)*p(:, k);

    % Atualiza o resíduo.
    r(:, k+1) = r(:, k) - a(k)*A*p(:, k);
    r2(:, k+1) = r2(:, k) - a(k)*A'*p2(:, k);
    B(k) = (r(:, k+1)'*r2(:, k+1))/(r(:, k)'*r2(:, k));
    p(:, k+1) = r(:, k+1) + B(k)*p(:, k);
    p2(:, k+1) = r2(:, k+1) + B(k)*p2(:, k);
    k = k+1;
end

% O resultatdo encontrado.
x = x(:, end);
```

O método BiCG é aplicável para matrizes não simétricas e requer o produto de matrizes e vetores com a matriz de coeficientes e sua transposta, sendo este o seu aspeto negativo: requer o armazenamento e operação com a matriz transposta (Saad, 2003).

A.3.5 Método dos gradientes biconjugados estabilizados

O método dos gradientes biconjugados estabilizados (BiCGstab) foi proposto por Vorst (1992) para resolver sistemas de equações lineares algébricas envolvendo matrizes não simétricas, e portanto evitando a operação $\mathsf{A}^T\mathsf{A}$. O método consiste em produzir um resíduo

$$\mathsf{r}_j = p_j(\mathsf{A})q_j(\mathsf{A})\mathsf{r}^{(0)}, \tag{A.10}$$

onde p_j é escolhido para manter pequena a norma do resíduo em cada passo j e q_j é o polinómio de grau j de A. Como exemplo, $p_j(t)$ pode ter a forma

$$p_{j+1}(t) = (1 - \omega_j t)p_j(t). \tag{A.11}$$

No algoritmo para execução do método BiCGstab, a solução é atualizada de tal maneira que r_j é da forma (A.10), onde $p_j(\mathsf{A})$ é um polinómio de grau j satisfazendo (A.11). Assim

$$\begin{aligned} p_{j+1}q_{j+1} &= (1-\omega_j t)p_j(q_j - \alpha_j t r_j) \\ &= (1-\omega_j t)(p_j q_j - \alpha_j t p_j r_j), \end{aligned}$$

e

$$\begin{aligned} p_j r_j &= p_j(q_j + \beta_{j-1} r_{j-1}) \\ &= p_j q_j + \beta_{j-1}(1-\omega_{j-1}t)p_{j-1}r_{j-1}. \end{aligned}$$

Seja $\mathsf{r}_j = q_j(\mathsf{A})p_j(\mathsf{A})\mathsf{r}^{(0)}$ e $\mathsf{p}_j = p_j(\mathsf{A})r_j(\mathsf{A})\mathsf{r}^{(0)}$. Podendo ser verificado que

$$\begin{aligned} \mathsf{r}_{j+1} &= (\mathsf{I} - \omega_j \mathsf{A})(\mathsf{r}_j - \alpha_j \mathsf{A}\mathsf{p}_j) \\ \mathsf{p}_{j+1} &= \mathsf{r}_{j+1} + \beta_j(\mathsf{I} - \omega_j \mathsf{A})\mathsf{p}_j. \end{aligned}$$

Seja $\mathsf{s}_j = \mathsf{r}_j - \alpha_j \mathsf{A}\mathsf{p}_j$, então nós obtemos

$$\mathsf{r}_{j+1} = (\mathsf{I} - \omega_j \mathsf{A})\mathsf{s}_j.$$

O parâmetro ω_j é escolhido para minimizar a norma-2 de r_{j+1}, por exemplo

$$\omega_j = \frac{\left\langle \mathsf{A}\mathsf{s}_j, \mathsf{s}_j \right\rangle}{\left\langle \mathsf{A}\mathsf{s}_j, \mathsf{A}\mathsf{s}_j \right\rangle}.$$

Os parâmetros α_j e β_j são atualizados, respetivamente, por

$$\alpha_j = \frac{\left\langle \mathsf{r}_j, \mathsf{r}_2{}^{(0)} \right\rangle}{\left\langle \mathsf{A}\mathsf{p}_j, \mathsf{r}_2{}^{(0)} \right\rangle} \quad \text{e} \quad \beta_j = \frac{\alpha_j}{\omega_j} \frac{\left\langle \mathsf{r}_{j+1}, \mathsf{r}_2{}^{(0)} \right\rangle}{\left\langle \mathsf{r}_j, \mathsf{r}_2{}^{(0)} \right\rangle},$$

onde $\mathsf{r}_2{}^{(0)}$ satisfaz $\left\langle \mathsf{r}^{(0)}, \mathsf{r}_2{}^{(0)} \right\rangle \neq 0$ (Shen et al., 2011).

O Código A.8 apresenta a implementação para o método BiCGstab.

Código A.8. Solução de um sistema de equações lineares algébricas por BiCGstab.

```
function [x, j] = SYS_BICGSTAB(A, b, max_it, tol)
%SYS_BGSTAB   Resolve o sistema Ax=b pelo método dos gradientes
%   biconjugados estabilizados com tolerância tol.

% Dimensão da matriz A.
n = length(A);

% Vetor aleatório como aproximação inicial.
x(:, 1) = rand(n, 1);

% Cálculo do primeiro resíduo.
r(:, 1) = b - A*x;
```

```
% Vetor aleatório para r2 inicial.
r2(:, 1) = rand(n, 1);

p(:, 1) = r; k = 1;
while norm(r(:, k)) > tol || k < max_it
    a(k) = (r(:, k)'*r2(:, 1))/((A*p(:, k))'*r2(:, 1));
    s(:, k) = r(:, k) - a(k)*A*p(:, k);
    o(k) = ((A*s(:, k))'*s(:, k))/((A*s(:, k))'*A*s(:, k));

    % Atualiza o vetor solução.
    x(:, k+1) = x(:, k) + a(k)*p(:, k) + o(k)*s(:, k);

    % Atualiza o resíduo.
    r(:, k+1) = s(:, k) - o(k)*A*s(:, k);
    B(k) = (r(:, k+1)'*r2(:, 1))/(r(:, k)'*r2(:, 1))*a(k)/o(k);
    p(:, k+1) = r(:, k+1) + B(k)*(p(:, k) - o(k)*A*p(:, k));
    k = k+1;
end

% O resultatdo encontrado.
x = x(:, end);
```

A.4 Precondicionadores

Um método iterativo pode aproximar mais rapidamente a solução x de um sistema $\mathsf{Ax} = \mathsf{b}$ se a matriz de coeficientes A usufruir de algumas boas propriedades relacionadas à convergência, e uma importante destas, como visto na Secção A.1.2, é o número de condição da matriz A ser pequeno. O precondicionamento, portanto, consiste em fazer alguma transformação no sistema original $\mathsf{Ax} = \mathsf{b}$ (multiplicando este por matrizes P e Q, chamadas precondicionadores), mas de forma a manter, obviamente, a mesma solução x, e simultaneamente a convergir mais rapidamente. Para tal, as possibilidades de precondicionamento são mostradas em uma só equação (A.12):

$$(\mathsf{P}^{-1}\mathsf{AQ})(\mathsf{Q}^{-1}\mathsf{x}) = \mathsf{P}^{-1}\mathsf{b}, \tag{A.12}$$

onde o sistema modificado agora, ao considerar $\bar{\mathsf{A}} = \mathsf{P}^{-1}\mathsf{AQ}$, $\bar{\mathsf{x}} = \mathsf{Q}^{-1}\mathsf{x}$ e $\bar{\mathsf{b}} = \mathsf{P}^{-1}\mathsf{b}$, corresponde a

$$\bar{\mathsf{A}}\bar{\mathsf{x}} = \bar{\mathsf{b}}.$$

É fácil a tarefa de mostrar que mesmo com as matrizes precondicionadoras P e Q a solução x de (A.12) fica inalterada, pois

$$(\mathsf{P}^{-1}\mathsf{AQ})(\mathsf{Q}^{-1}\mathsf{x}) = \mathsf{P}^{-1}\mathsf{b}$$

$$\mathsf{Q}^{-1}x = (\mathsf{P}^{-1}\mathsf{AQ})^{-1}\mathsf{P}^{-1}\mathsf{b}$$

$$\mathsf{Q}^{-1}x = (\mathsf{AQ})^{-1}\mathsf{PP}^{-1}\mathsf{b}$$

$$\mathsf{Q}^{-1}\mathsf{x} = \mathsf{Q}^{-1}\mathsf{A}^{-1}\mathsf{PP}^{-1}\mathsf{b}$$

$$\mathsf{x} = \mathsf{Q}\mathsf{Q}^{-1}\mathsf{A}^{-1}\mathsf{P}\mathsf{P}^{-1}\mathsf{b}$$
$$\mathsf{x} = \mathsf{A}^{-1}\mathsf{b}.$$

Caso $\mathsf{Q} = \mathsf{I}$, onde I é a matriz identidade, temos que a equação (A.12) é escrita como

$$\mathsf{P}^{-1}\mathsf{A}\mathsf{x} = \mathsf{P}^{-1}\mathsf{b},$$

e a este chamamos precondicionamento à esquerda. De forma similar, se $\mathsf{P} = \mathsf{I}$, então a equação (A.12) é escrita como

$$\mathsf{A}\mathsf{Q}\mathsf{Q}^{-1}\mathsf{x} = \mathsf{b},$$

e a este chamamos precondicionamento à direita. Por fim, se $\mathsf{Q} \neq \mathsf{I}$ e $\mathsf{P} \neq \mathsf{I}$, o precondicionamento é dito ser misto (Saad, 2003). Portanto devemos procurar o precondicionador P, ou Q, ou ambos que satisfaçam $\text{cond}(\bar{\mathsf{A}}) \ll \text{cond}(\mathsf{A})$, ou ainda que os valores próprios de $\mathsf{P}^{-1}\mathsf{A}\mathsf{Q}$ estejam em uma nuvem próxima a unidade. Obviamente, o melhor precondicionador seria $\mathsf{P} = \mathsf{A}$, mas isso recai em encontrar o próprio A^{-1} e resolver o sistema em um passo, mas este não é o objetivo.

A.4.1 Precondicionador por fatorização ILU

A fatorização ILU (LU incompleta) dá-se a partir da geração das matrizes L (triangular inferior) e U (triangular superior), ao fazer uso da eliminação gaussiana com uma pequena modificação, a saber, desprezar alguns preenchimentos $l_{i,j}$ e/ou $u_{i,j}$ que originalmente, em A, são $a_{i,j} = 0$, o que pode manter o padrão de esparsidade de A. Estes preenchimentos, quando não desprezados, são comumente chamados na literatura de "fill-in". Como esta fatorização faz uso da eliminação de Gauss, devemos ter cuidado ao trabalhar com algum pivô nulo, o que causa a interrupção do algoritmo ao tentar dividir por 0. Há duas possibilidades para o padrão de esparsidade de LU: um é o escolhido antes da execução do algoritmo e o outro o moldado durante a execução do algoritmo, ao que respetivamente chamamos de fatorização estática ou por posição e fatorização dinâmica ou por valor (Vasconcelos, 1998).

Após obter as matrizes L e U, cada coluna j do precondicionador P é construída resolvendo sequencialmente os dois sistemas triangulares:

$$\mathsf{L}\bar{\mathsf{x}} = \mathsf{e}_j \quad \text{e} \quad \mathsf{U}\mathsf{P}_j = \bar{\mathsf{x}}$$

A.4.2 Precondicionador por minimização da norma de Frobenius

Como se sabe, o possível aparecimento de um pivô nulo na fatorização ILU compromete a execução da mesma, interrompendo instantaneamente. Para evitar esse problema podemos usar o precondicionador por minimização da norma de Frobenius, que consiste em aproximar diretamente a matriz inversa A^{-1}.

Sendo P a matriz aproximada a A^{-1}, o interesse é encontrá-la de tal forma que minimize a norma de Frobenius da matriz resíduo $\mathsf{I} - \mathsf{AP}$, $\mathsf{I} - \mathsf{PA}$ ou $\mathsf{I} - \mathsf{LAU}$:

$$F(\mathsf{P})_r = ||\mathsf{I} - \mathsf{AP}||_F^2, \tag{A.13}$$

$$F(\mathsf{P})_l = ||\mathsf{I} - \mathsf{PA}||_F^2 \tag{A.14}$$

ou

$$F(\mathsf{P})_{rl} = ||\mathsf{I} - \mathsf{LAU}||_F^2. \tag{A.15}$$

Uma matriz P cujo valor (A.13) é pequeno já seria uma aproximação da matriz inversa direita de A, da mesma forma que uma matriz P cujo valor (A.14) é suficientemente pequeno já seria uma boa aproximação da matriz inversa esquerda de A. Por fim, uma par direito-esquerdo (L e U) pode ser procurado para minimizar (A.15).

Conforme Saad (2003), pela similaridade de (A.13) e (A.14), apenas (A.13) e (A.15) são consideradas. A função objetivo (A.13) corresponde à soma das normas-2 das colunas individuais da matriz resíduo $\mathsf{I} - \mathsf{AP}$, ou seja

$$F(\mathsf{P})_r = ||\mathsf{I} - \mathsf{AP}||_F^2 = \sum_{j=1}^{m} ||\mathsf{e}_j - \mathsf{A}m||_2^2, \tag{A.16}$$

onde e_j e m_j são as j-ésimas colunas da matriz identidade e da matriz P respetivamente.

Uma das formas possíveis de minimizar a função objetivo (A.16) é considerá-la globalmente como uma função de uma matriz esparsa P, por exemplo através de um método tipo gradiente, ou então em alternativa, considerá-la localmente através da minimização das funções individuais

$$f_j(m_j) = ||\mathsf{e}_j - \mathsf{A}m_j||_2^2, \quad j = 1(1)n.$$

Apêndice B

Lista das rotinas da `Tau Toolbox`

Sumário

B.1 Classe e objeto @ctau

cos ..... Cosseno.
cosh ..... Cosseno hiperbólico.
ctau ..... Definição de classe para coeficientes.
exp ..... Exponencial.
log ..... Logaritmo natural.
minus ..... Operação de subtração.
mpower - Prop. 5.9 ..... Operação de potência.
mtimes ..... Operação de multiplicação.
plot ..... Plota resultados.

plus ..... Operação de soma.
sin ..... Seno.
sinh ..... Seno hiperbólico.
uminus ..... Operação de subtração unitária.
uplus ..... Operação de soma unitária.

B.2 Classe e objeto @dtau

diff ..... Operador de diferenciação.
dtau ..... Definição de classe para variável dependente.
fred ..... Operador de integração de Fredholm.
int ..... Operador de primitivação.
matrixC ..... Matrix C da tradução das condições do problema.
matrixCsys ..... Matriz C para sistemas.
matrixN - Prop. 5.5 ..... Matriz de diferenciação.
matrixO - Prop. 5.6 ..... Matriz de primitivação.
minus ..... Operação de subtração.
mrdivide ..... Operação de divisão.
mtimes ..... Operação de multiplicação.
plus ..... Operação de soma.
uminus ..... Operação de subtração unitára.
uplus ..... Operação de soma unitária.
volt ..... Operador de integração de Volterra.

B.3 Classe e objeto @itau

cos ..... Cosseno.
cosh ..... Cosseno hiperbólico.
coshi ..... Cosseno hiperbólico por interpolação.
coshm ..... Cosseno hiperbólico por função de matriz.
cosht ..... Cosseno hiperbólico por aproximação **tau**.
cosi ..... Cosseno por interpolação.
cosm ..... Cosseno por função de matriz.
cost ..... Cosseno por aproximação **tau**.
exp ..... Exponencial.
expi ..... Exponencial por interpolação.
expm ..... Exponencial por função de matriz.
expt ..... Exponencial por aproximação **tau**.
itau ..... Definição de classe para variável independente.
linspace ..... Vetor de pontos.
log ..... Logaritmo natural.
logi ..... Logaritmo natural por interpolação.

logm ..... Logaritmo natural por função de matriz.
logt ..... Logaritmo natural por aproximação **tau**.
matrixM - Prop. 5.4 ..... Matriz de incremento de grau.
minus ..... Operação de subtração.
mpower - Prop. 5.7 ..... Operação de potenciação.
mtimes ..... Operação de multiplicação.
plus ..... Operação de soma.
pow2orth ... Transformação da base das potências para base ortogonal.
powMrec - Prop. 5.7 ..... Potências da matriz M por recorrência.
sin ..... Seno.
sinh ..... Seno hiperbólico.
sinhi ..... Seno hiperbólico por interpolação.
sinhm ..... Seno hiperbólico por função de matriz.
sinht ..... Seno hiperbólico por aproximação **tau**.
sini ..... Seno por interpolação.
sinm ..... Seno por função de matriz.
sint ..... Seno por proximação **tau**.
sqrt ..... Raiz quadrada.
sqrti ..... Raiz quadrada por interpolação.
sqrtm ..... Raiz quadrada por função de matriz.
sqrtt ..... Raiz quadrada por aproximação **tau**.
uminus ..... Operação de subtração unitária.
uplus ..... Operação de soma unitária.

B.4 Classe e objeto @rtau

cos ..... Cosseno por aproximação **tau**.
cosh ..... Cosseno hiperbólico por aproximação **tau**.
exp ..... Exponencial por aproximação **tau**.
log ..... Logaritmo natural por aproximação **tau**.
minus ..... Operação de subtração.
mpower ..... Operação de potenciação.
mtimes ..... operação de multiplicação.
plus ..... Operação de soma.
rtau ..... Definição de classe para variável independente no rhs.
sin ..... Seno por aproximação **tau**.
sinh ..... Seno hiperbólico por aproximação **tau**.
uminus ..... Operação de subtração unitária.
uplus ..... Operação de soma unitária.

B.5 Construção das bases de polinómios ortogonais clássicos

chebyshevTpoints ..... Pontos de Chebyshev de primeira espécie.
chebyshevUpoints ..... Pontos de Chebyshev de segunda espécie.
gegenbauerCpoints ..... Pontos de Gegenbauer.
legendrePpoints ..... Pontos de Legendre.
orth2powmatrix .. Matriz de transformação de base ortogonal para pot.
orthobasispoints ..... Pontos de base ortogonal.
plotpoly ..... Plota polinómios ortogonais.
polbesselY ..... Polinómios de Bessel.
polchebyshevT ... Polinómios de Chebyshev de primeira espécie [-1 1].
polchebyshevTab .. Polinómios de Chebyshev de primeira espécie [a b].
polchebyshevU .... Polinómios de Chebyshev de segunda espécie [-1 1].
polchebyshevUab ... Polinómios de Chebyshev de segunda espécie [a b].
polgegenbauerC ..... Polinómios de Gegenbauer [-1 1].
polgegenbauerCab ..... Polinómios de Gegenbauer [a b].
polhermiteH ..... Polinómios de Hermite.
pollaguerreL ..... Polinómios de Laguerre.
pollegendreP ..... Polinómios de Legendre [-1 1].
pollegendrePab ..... Polinómios de Legendre [a b].
pow2orthmatrix ... Matriz de transformação da base das pot. para ort.
shift2ab ..... Shift no intervalo de otogonalidade.

B.6 Exemplos de problemas aproximados pela **Tau Toolbox**

linear_ide_fredholm_Hosseini2002 .....
linear_ide_fredholm_non_polynomial_coefficients .....
linear_ide_fredholmvolterra .....
linear_ide_fredholmvolterra_2 .....
linear_ide_fredholmvolterra_basic .....
linear_ide_fredhom .....
linear_ide_volterra .....
linear_ide_volterra_Hosseini2002 .....
linear_ide_wikipedia .....
linear_ie_volterra_Yang2013Rabbani2007_ex1 .....
linear_ie_volterra_Yang2013Rabbani2007_ex2 .....
linear_ode .....
linear_ode_Legendre .....
linear_ode_OlverTownsend .....
linear_ode_bvp_piecewise .....
linear_ode_sign_function_with_non_plynomial_coefficients .....

B.7 Avaliação dos operadores

B.8 Avaliação em bases de polinómios ortogonais

B.9 Interpolação em bases de polinómios ortogonais

interporth ..... Interpolação em base de polinómios ortogonais.
interporth2 ..... Interpolação 2D em base de polinómios ortogonais.
interporth2_error ..... Erro na interpolação 2D.
interporth2inc ..... Interpolação 2D incremental.
interporthinc ..... Interpolação incremental.

B.10 Produto de polinómios em bases ortogonais

chebypolypow ..... Potência dos coeficientes na base de Chebyshev.
chebypolyprod ..... Produto dos coeficientes na base de Chebyshev.
legpolypow ..... Potência dos coeficientes na base de Legendre.
legpolyprod ..... Produto dos coeficientes na base de Legendre.
orthopolyprod ..... Produto dos coeficientes em base ortogonal.

B.11 TauGui - tau graphical user interface

createmfile_advanced_ode ..... mfile para edo.
createmfile_advanced_ode_pw ..... mfile para edo parcelar.
createmfile_advanced_sys ..... mfile para sietema de edo's.
createmfile_advanced_sys_pw ... mfile para sistema de edo's parcelar.
createmfile_beginner ..... mfile na versão elementar.
findsolver ..... Encontra o solver mais apropriado para o problema.
getalldata ..... Coleta todos os dados da interface.
informations ..... Informações técnicas.
taugui ..... Interface gráfica para o método **tau**.

B.12 TauSolvers - Funções nível básico

schur_iter ..... Compementos de Schur iterativo GMRES.
schur_luinc .. Complementos de Schur com decomposição LU incremental.
schur_nflu ..... Complementos de Shur com bloco LU não fixado.
schursolver ..... Método **tau** iterativo com comp. de Schur.
schursolverpw ..... Método **tau** iteratico com comp. de Schur parcelar.
tauode ..... Método **tau** para edo.
tauodenewton ..... Método **tau** para edo linearizada.
tauodepw ..... Método **tau** parcelar para edo.
tauodepwpvf ..... Método **tau** parcelar para prob. valor de fronteira.

tausolver ..... Método **tau**.
tausys ..... Método **tau** para sistemas de edo's.
tausysnewton ..... Método **tau** para sistemas de edo's linearizadas.
tausyspw ..... Método **tau** parcelar para sistemas de edo's.

B.13 Testes das Proposições

TEST_Mrec_vs_Mformal ..... Testa Proposição 5.7.
TEST_Mrec_vs_Mformal_problem ..... Testa Proposição 5.7.
TEST_coef_fun_by_Schurcomplements ..... Testa complementos de Schur .
TEST_coeflinearization_vs_convolution ..... Testa Proposição 5.9.
TEST_differents_kind_of_orthoval ..... Testa Proposição 5.1.
TEST_invVAV_vs_recA ..... Testa Proposições 5.4, 5.5, 5.6.
TEST_invVAV_vs_recA_problem ..... Testa Proposições 5.4, 5.5, 5.6.
TEST_invVAV_vs_recA_problem2 ..... Testa Proposições 5.4, 5.5, 5.6.
TEST_invVAV_vs_recA_spy ..... Testa Proposições 5.4, 5.5, 5.6.
TEST_invV_vc_W ..... Testa Proposição 5.3.
TEST_invV_vc_W_problem ..... Testa Proposição 5.3.
TEST_polyval_vs_orthoval ..... Testa Proposição 5.1.
TEST_polyval_vs_orthoval_kernel ..... Testa Proposição 5.1.

B.14 Ferramentas para o método tau

altura ..... Altura do operador.
basis2basis ..... Mudança de baseortogonal para base ortogonal.
changesupc ..... Muda tipo de condições.
colorR2B ..... Gera tons de cores do vermelho para o azul.
deriveP ..... Deriva elementos da base ortogonal.
eorth ..... j-ésimo vetor de uma base ortogonal.
evalonly ..... Avaliador seletivo.
evalonlynewton ..... Avaliador seletivo para problemas linearizados.
fc ..... Tratamento de coefiente funcional não polinomial.
findint ..... Enontra termo integral.
findkernel ..... Encontra núcleo do termo integral.
fun2orth ..... Função para base ortogonal.
isivp ..... Verifica se é problema de valor inicial.
matrixT .... Matriz que traduz o operador e as condições de contorno.
maxerror ..... Erro máximo.
mno ..... Matrizes M, N e O na base das potências.
myspy ..... Spy de matriz no formato cityplot.
poly2str ..... Polinómio para string.
prodcoef ..... Produto de coefcientes.

readcondcell ..... Lê condições no formato de célula.
readcondcellsys .... Lê condições para sistemas no formato de célula.
reorg2poly ..... Reorganiza vetor para polnómio.
reorg2vec ..... Reorganiza polinómio para vetor.
solvingthesystem .. Resolve o sist. linear associado ao problema **tau**.
spliteq ..... Reparte equações em LRS e RHS.
str2poly ..... Srting para polinómio.
tau .....Criação dos objetos **tau** de variável dep. e indep.
tausettings ..... Configurações para escolher formulações.
tausystem ..... Gera T e b do sistema Ta=b associado ao método **tau**.

Notation and Conventions	meaning
x, object	independent variable
y, object	dependent variable
n, j $\in \mathbb{Z}_0^+$	polynomial degree
M, N, O $\in \mathbb{R}^{n\times n}$	operational matrices
a, p, q	coefficients in a orthogonal basis
k $\in \mathbb{Z}_0^+$	diffentiation order
k $\in \mathbb{Z}^-$	integration order
ode	integro-differential problem
conds	conditions
prec	precision
func	function
inc	increment
v_a, v_b, v_c	coefficient vectors
$\mathcal{Z}$, basis	orthogonal basis
domain	interval of orthogonality

About this work

The tau method belongs to the class of spectral methods and is intended to numerically compute polynomial approximate solutions of differential problems. In order to obtain a solution with good approximation properties, in these methods, the residual, resulting from the approximation of the solution by a polynomial, is projected on an orthogonal polynomial basis and conditions are imposed to ensure a small residual, in a suitable norm. The efficient and robust implementation of the method, obtained with this work, results from the conjugation of the work in three areas: Numerical Analysis, Numerical Linear Algebra and Programming.

Examples 1: Differential equation

$$\begin{cases} \frac{d^2}{dx^2}y + x^2 y = x^3, \ x \in]0,\ 5[\\ y(0) = 0,\ y'(0) = 1 \end{cases}$$

```
[x, y] = tau(ChebyshevT,[0 5],20);
ode = 'diff(y,2)+x^2*y=x^3';
conds = {'y(0)=0';'y''(0)=1'};
a = tausolver(x,y,ode,conds)
```

Example 2: Differential equation

$$\begin{cases} \cos(x)\frac{d}{dx}y + e^{x^2} y = 0, \ x \in]-1,\ 1[\\ y(-1) = 1 \end{cases}$$

```
[x, y] = tau(ChebyshevU,[-1 1],8)
ode = 'cos(x)*diff(y)+exp(x^2)*y=0'
conds = {'y(-1)=1'}
a = tausolver(x,y,ode,conds)
```

Example 3: Volterra integro-differential equation

$$\begin{cases} \frac{d}{dx}y + 2y + 5\int_{-1}^{x} y dt = 0, \ x \in]-1,\ 1[\\ y(-1) = 1 \end{cases}$$

```
[x, y]=tau('ChebyshevT',[0 5],20);
ode='diff(y)+2*y+5*volt(y)= 1';
conds='y(0)=0';
a=tausolver(x,y,ode,conds);
```

Example 4: Nonlinear differential equation

$$\begin{cases} \frac{d}{dx}y + y^2(y-1) = 0, \ x \in]0,\ 1400[\\ y(-1) = 1 \end{cases}$$

```
[x, y]=tau('LegendreP',[0, 1400],100);
ode='diff(y)+3*yo^2*y-2*yo*y=2*yo^3-yo^2';
conds='y(0)=1/700';
a=tauodenewton(x,y,ode,conds,'pieces',100);
```


Tau Toolbox

for the solution of integro-differential problems

Quick Reference Guide
Release 1.0
May 2017

Univ. of Porto & CMUP
José M. A. Matos
M. S. Trindade
P. B. Vasconcelos

Obtaining Tau Toolbox
www.fc.up.pt/tautoolbox

Create tau objects	default
`x = itau(basis,domain,n)`	
`y = dtau(basis,domain,n)`	
`[x,y] = tau(basis,domain,n)`	
`[x,y] = tau(basis,domain)`	n=5
`[x,y] = tau(basis)`	domain=[-1,1], n=5
`[x,y] = tau`	basis=$\mathcal{T}$, domain=[-1,1], n=5

basis	meaning
`'ChebyshevT'`	Chebyshev of 1st kind $\mathcal{T}$
`'ChebyshevU'`	Chebyshev of 2nd kind $\mathcal{U}$
`'LegendreP'`	Legendre $\mathcal{P}$
`'GegenbauerC'`, α	Gegenbauer(α) $\mathcal{C}$
`'HermiteH'`	Hermite $\mathcal{H}$
`'LaguerreL'`	Laguerre $\mathcal{L}$
`'BesselY'`	Bessel $\mathcal{Y}$

Solve integro-differential problems
`tausolver(x,y,ode,conds,opts)`
`tauode(x,y,ode,conds,opts)`
`tauodepw(x,y,ode,conds,opts)`
`tauodepwpvf(x,y,ode,conds,opts)`
`tausys(x,y,ode,conds,opts)`
`tausyspw(x,y,ode,conds,opts)`
`schursolver(x,y,ode,conds,prec,opts)`
`schursolverpw(x,y,ode,conds,prec,opts)`
`tauodenewton(x,y,ode,conds,prec,opts)`
`tausysnewton(x,y,ode,conds,prec,opts)`

opts	default
`'pieces'`	1
`'precond'`	'no'
`'apsol'`	0
`'resid'`	0
`'coeff'`	1
`'spy'`	'0'

Operational matrices
`M = matrixM(x)`
`N = matrixN(y)`
`O = matrixO(y)`

Polynomial operations	meaning
`mpower(x, j)` or `x^j`	scalar product x^j
`diff(y, k)`	differentiation $\frac{d^k}{dx^k}y$
`int(y, k)`	primitive $\int \dots \int y dx^k$
`fred(y, 'K(x,t)')`	Fredholm int. $\int_a^b K(x,t)y(t)dt$
`volt(y, 'K(x,t)')`	Volterra int. $\int_a^x K(x,t)y(t)dt$

Polynomial evaluation	meaning
`orthoval(x,pts,opts):`	
`orthovalv(a,pts,domain,basis)`	$\mathcal{Z}\mathsf{a}\|_{pts}$
`orthovalvj(j,pts,domain,basis)`	$Z_j\|_{pts}$
`orthovald(N,k,pts,domain,basis)`	$\frac{d^k}{dx^k}\mathcal{Z}\|_{pts}$
`orthovalM(a,M,domain,basis)`	$\mathcal{Z}\mathsf{a}\|_{\mathsf{M}}$
`orthovalMj(j,M,domain,basis)`	$Z_j\|_{\mathsf{M}}$

opts	meaning
`'coef',a`	vector of coefficients
`'j',j`	polynomial degree
`'difforder',n`	differentiation order

Polynomial evaluation with operations	meaning
`taudiff(step,x,a,k)`	$\frac{d^k}{dx^k}\mathcal{Z}\mathsf{a}$
`tauint(step,x,a,k)`	$\int \dots \int \mathcal{Z}\mathsf{a}dx^k$
`taufred(step,x,a,'K(x,t)')`	$\int_a^b K(x,t)\mathcal{Z}\mathsf{a}dt$
`tauvolt(step,x,a,'K(x,t)')`	$\int_a^x K(x,t)\mathcal{Z}\mathsf{a}dt$

Linear system solution	meaning
`[T, b] = tausystem(x,y,ode,conds)`	$\mathsf{Ta} = \mathsf{b}$
`a = solvingthesystem(T,b)`	$\mathsf{Ta} = \mathsf{b}$

Base operations	meaning
`W = orth2powmatrix(x)`	$\mathsf{p}_{\mathcal{X}} = \mathsf{W}\mathsf{p}_{\mathcal{Z}}$
`V = pow2orthmatrix(x)`	$\mathsf{p}_{\mathcal{Z}} = \mathsf{V}\mathsf{p}_{\mathcal{X}}$

Basis building	meaning
`polchebyshevTab(n,domain)`	$\mathsf{V}_{\mathcal{T}}$
`polchebyshevUab(n,domain)`	$\mathsf{V}_{\mathcal{U}}$
`pollegendrePab(n,domain)`	$\mathsf{V}_{\mathcal{P}}$
`polgegenbauerCab(n,domain)`	$\mathsf{V}_{\mathcal{C}}$
`pollaguerreL(n)`	$\mathsf{V}_{\mathcal{L}}$
`polhermiteH(n)`	$\mathsf{V}_{\mathcal{H}}$
`polbesselY(n)`	$\mathsf{V}_{\mathcal{Y}}$

Gaussian quadrature roots
`chebyshevTpoints(n,domain)`
`chebyshevUpoints(n,domain)`
`legendrePpoints(n,domain)`
`gegenbauerCpoints(n,domain)`
`orthobasispoints(x)`

Non-polynomial coefficients	
`funi(x)`	by interpolation
`funm(x)`	by Taylor series for matrix
`funt(x)`	by tau method

where `fun` can be:
`sin`, `cos`, `sinh`, `cosh`, `exp` and `log`.

Orthogonal interpolation
`interporth(x,func)`
`interporthinc(x,func,prec,inc)`
`interporth2(x,func)`
`interporth2inc(x,func,prec,inc)`

Polynomial products	meaning
`orthopolyprod(p,q,v_a,v_b,v_c)`	$\mathsf{p}_{\mathcal{Z}}\mathsf{q}_{\mathcal{Z}}$
`chebypolyprod(p,q)`	$\mathsf{p}_{\mathcal{T}}\mathsf{q}_{\mathcal{T}}$
`legpolyprod(p,q)`	$\mathsf{p}_{\mathcal{P}}\mathsf{q}_{\mathcal{P}}$

Referências Bibliográficas

S. Abbasbandy and A. Taati. Numerical solution of the system of nonlinear Volterra integro-differential equations with nonlinear differential part by the operational tau method and error estimation. *Journal of Computational and Applied Mathematics*, 231(1):106–113, 2009.

M. Abramowitz and I. A. Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*. Number 55. Courier Dover Publications, 1972.

M. H. AliAbadi and S. Shahmorad. A matrix formulation of the tau method for Fredholm and Volterra linear integro-differential equations. *Korean Journal of Computational & Applied Mathematics*, 9(2):497–507, 2002.

R. Askey. *Orthogonal polynomials and special functions*. Society for Industrial and Applied Mathematics, 1975.

R. Barrett, M. Berry, T. F. Chan, Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. V. der Vorst. *Templates for the solution of linear systems: building blocks for iterative methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1994.

J. P. Boyd. *Chebyshev and fourier spectral methods*. Dover Publications, 2001.

R. L. Burden and J. D. Faires. *Numerical analysis*. 1993.

C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang. *Spectral Methods: fundamentals in single domains*. Springer, 2010.

T. S. Chihara. *An introduction to orthogonal polynomials*, volume 13 of *Ellis Horwood series in mathematics and its applications*. Gordon and Breach, 1978.

C. W. Clenshaw. A note on the summation of chebyshev series. *Mathematics of Computation*, 9(51):118–120, 1955.

R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth. On the Lambertw function. *Advances in Computational Mathematics*, 5, 1996.

M. R. Crisci and E. Russo. An extension of ortiz recursive formulation of the tau method to certain linear systems of ordinary differential equations. *Mathematics of computation*, 163(41):27–42, 1983.

M. R. da Silva. *Aproximação de funções no sentido do método τ*. Revista da Universidade de Coimbra, 1990.

M. Dehghan and R. Salehi. The numerical solution of the non-linear integro-differential equations based on the meshless method. *Journal of Computational and Applied Mathematics*, 236(9):2367–2377, 2012.

J. W. Demmel. *Applied numerical linear algebra*. Society for Industrial and Applied Mathematics, 1997.

B. A. Finlayson and L. E. Scriven. The method of weigted residuals - a review. *Aplied Mechanics Reviews*, 19(9):735–748, 1966.

A. Gavina. O método espectral tau para problemas de controlo ótimo, 2016.

A. Gavina, J. Matos, and P. Vasconcelos. Improving the accuracy of chebyshev tau method for nonlinear differential problems. *Mathematics in Computer Science*, 10(2): 279–289, 2016.

A. Gil, J. Segura, and N. M. Temme. *Numerical methods for special functions*. Society for Industrial and Applied Mathematics, 2007.

G. H. Golub and J. H. Welsch. Calculation of Gauss quadrature rules. *Mathematics of Computation*, 23(106):221–230, 1969.

D. Gottlieb and S. A. Orszag. *Numerical analysis of spectral methods: theory and Applications*, volume 26. Society for Industrial and Applied Mathematics, 1977.

W. Hahn. *Uber die jacobischen polynome und zwei verwandre polynomklassen*. Number 39. Math. Z., 1935.

M. T. Heath. *Scientific computing, an introductory survey*, volume 2. McGraw-Hill New York, 2002.

N. J. Higham. *Functions of matrices: theory and computation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008. ISBN 978-0-898716-46-7.

N. J. Higham and D. J. Higham. Large growth factors in gaussian elimination with pivoting. *Society for Industrial and Applied Mathematics Journal on Matrix Analysis and Applications*, 10(2):155–164, 1989.

S. M. Hosseini and S. Shahmorad. Numerical solution of a class of integro-differential equations by the tau method with an error estimation. *Applied Mathematics and Computation*, 136(2):559–570, 2003a.

S. M. Hosseini and S. Shahmorad. Tau numerical solution of Fredholm integro-differential equations with arbitrary polynomial bases. *Applied Mathematical Modelling*, 27(2): 145–154, 2003b.

H. Jeffreys and B. S. Jeffreys. *Weierstrass's theorem on approximation by polynomials*. Cambridge Univ. Press Cambridge, England, UK, 1988.

C. Lanczos. *Trigonometric interpolation of empirical and analitical functions*, volume 17. Journal of mathematics and physics, 1938.

C. Lanczos. *Applied analysis*. Prentice Hall, INC, 1956.

K. M. Liu and C. K. Pan. The automatic solution to systems of ordinary differential equations by the tau method. *Computers and Mathematics with Applications*, 38(9): 197–210, 1999.

J. P. Mahmound, M. Y. R. Ardabili, and A. Shahorad. Numerical solution of the systems of Fredholm integro-differential equations by the tau method. *Applied Mathematics and Computation*, 168(1):465–478, 2005.

R. Mathias. Approximation of matrix-valued functions. *Journal on Matrix Analysis and Applications*, 14(4):1061–1063, 1993.

J. Matos. O método-τ em termos operacionais, 1994.

J. Matos, M. J. Rodrigues, and P. B. Vasconcelos. New implementation of the tau method for PDEs. 164-165:555–567, 2004.

J. Matos, M. J. Rodrigues, and J. C. de Matos. Avoiding similarity transformations in the operational tau method. 2017.

P. Mokhtary and F. Ghoreishi. Convergence analysis of the operational tau method for Abel-type Volterra integral equations. *Eletronic Transitions on Numerical Analysis*, 41:289–305, 2014.

S. Namasivayam and E. L. Ortiz. Best approximation and the numerical solution of partial differential equations with the tau method. *Portugaliae Mathematica*, 40(1): 97–119, 1981.

S. Olver and A. Townsend. A fast and well-conditioned spectral method. *Society for Industrial and Applied Mathematics Review*, 55(3):462–489, 2013.

P. Onumanyi and E. Ortiz. Numerical solution of stiff and singularly perturbed boundary value problems with a segmented-adaptive formulation of the tau method. *Mathematics of Computation*, 43(167):189–203, 1984.

E. L. Ortiz. Step by step tau method - Part i. Piecewise polynomial approximations. *Computers and Mathematics with Applications*, 1(3-4):381–392, 1975.

E. L. Ortiz. *On the numerical solution of nonlinear and functional differential equations with the tau method*, volume 679, pages 127–139. Springer, 1978.

E. L. Ortiz and A. P. N. Dinh. Linear recursive schemes associated with some nonlinear partial differential equations in one dimension and the tau method. *Journal on Mathematical Analysis*, 18(2):452–464, 1987.

E. L. Ortiz and H. Samara. An operational approach to the tau method for the numerical solution of nonlinear differential equations. *Computing*, 1(27):15–25, 1981.

M. Rabbani, K. Maleknejad, and N. Aghazadeh. Numerical computational solution of the Volterra integral equations system of the second kind by using an expansion method. *Applied Mathematics and Computation*, 187(2):1143–1146, 2007.

M. Y. Rahimi, S. Shahmorad, F. Talati, and A. Tari. An operational method for the numerical solution of two dimentional linear Fredholm integral equations with an error estimation. *Bulletin of the Iranian Mathematical Society*, 36(2):119–132, 2010.

M. J. Rodrigues and J. Matos. A tau method for nonlinear dynamical systems. *Numerical Algorithms*, 62(4):583–600, 2012.

Y. Saad. *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics, 2003.

L. Saeedi, A. Tari, and S. H. M. Masuleh. Numerical solution of some nonlinear Volterra integral equations of the first kind. *Applications and Applied Mathematics*, 8(1):214–226, 2013.

J. Shen, T. Tang, and L.-L. Wang. *Spectral methods: algorithms, analysis and applications*, volume 41. Springer, 2011.

P. K. Suetin. Classical orthogonal polynomials. 2001.

G. Szego. *Orthogonal polynomials*, volume 23. American Mathematical Society, 1939.

L. N. Trefethen. *Finite difference and spectral methods for ordinary and partial differential equations*. Cornell University, 1994.

L. N. Trefethen. *Spectral methods in Matlab*. Society for Industrial and Applied Mathematics, 2000.

L. N. Trefethen. *Approximation theory and approximation practice*. Society for Industrial and Applied Mathematics, 2013.

M. Trindade, J. Matos, and P. B. Vasconcelos. Towards a Lanczos' τ-method toolkit for differential problems. *Mathematics in Computer Science*, 10(3):313–329, 2016.

P. B. Vasconcelos. Paralelização de algoritmos de álgebra linear numérica com aplicação a mecânica de fluidos computacionais, 1998.

H. A. V. D. Vorst. A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. *Journal on Scientific and Statistical Computing*, 13(2): 631–644, 1992.

L. H. Yang, H. Y. Li, and J. R. Wang. Solving a system of linear Volterra integral equations using the modified reproducing kernel method. *Abstract and Applied Analysis*, 2013.

O. C. Zienkiewicz and Y. K. Cheung. *The finite method in structural and continuum mechanics*. McGraw-Hill, London, 1967.