Luís Tiago de Freixo Ramos Paiva

Cálculo recursivo dos aproximantes de Frobenius-Padé: teoria, estabilidade e condicionamento

Departamento de Matemática Aplicada
Faculdade de Ciências da Universidade do Porto
Faculdade de Engenharia da Universidade do Porto
Janeiro 2004

Luís Tiago de Freixo Ramos Paiva

Cálculo recursivo dos aproximantes de Frobenius-Padé: teoria, estabilidade e condicionamento

Tese submetida à Faculdade de Ciências da Universidade do Porto para obtenção do grau de Mestre em Métodos Computacionais em Ciências e Engenharia

Departamento de Matemática Aplicada
Faculdade de Ciências da Universidade do Porto
Faculdade de Engenharia da Universidade do Porto
Janeiro 2004

Silvia Gama

Aos meus pais,

À minha avó,

À minha irmã,

À Joana.

Agradecimentos

À Professora Doutora Zélia Rocha e ao Professor Doutor José Matos, meus orientadores, o meu sincero agradecimento pelo apoio científico, a disponibilidade, o incentivo e a amizade que me dedicaram. Agradeço-lhes terem-me dado a conhecer o cerne da Análise Numérica e a Aproximação de Frobenius-Padé. Devo-lhes ainda a cuidada revisão desta tese. Foi sem dúvida um prazer e um orgulho imenso ter trabalhado com dois excepcionais investigadores.

Ao Professor Doutor Sílvio Gama e ao Professor Doutor Laginha Palma, membros da Coordenação de Mestrado, agradeço a atenção e a oportunidade concedida para o meu desenvolvimento como Matemático e Investigador.

À minha namorada Joana, o meu profundo agradecimento pelo incansável suporte emocional e pelo forte apoio e interesse que demonstrou no meu percurso.

À minha irmã, agradeço as sugestões linguísticas enriquecedoras e o tempo inextinguível que me dedicou durante a elaboração deste trabalho.

Aos meu pais, o meu ilustre agradecimento por todo o apoio que me concederam e pelo orgulho que me depositam. Esta tese pertence-lhes.

Ao Departamento de Matemática Aplicada da Universidade do Porto e à Faculdade de Engenharia da Universidade do Porto, agradeço todas as facilidades concedidas.

Resumo

O nome Frobenius-Padé é atribuído à classe de aproximantes racionais que, no sentido da definição de Frobenius, generalizam a aproximação de Padé a séries ortogonais. De forma mais clara, se conhecermos um desenvolvimento formal em série de polinómios ortogonais $\{P_i\}_{i\geq 0}$ da função $f(z)$, i.e., $f(z) = \sum_{i=0}^{\infty} f_i P_i(z)$, para cada par de valores inteiros não negativos p e q, designa-se por aproximante de Frobenius-Padé (AFP), $[p/q]_f^P$, a função racional $N^{[p/q]}(z)/D^{[p/q]}(z)$, onde $N^{[p/q]}(z) = \sum_{i=0}^{p} a_i P_i(z)$ e $D^{[p/q]}(z) = \sum_{i=0}^{q} b_i P_i(z)$ são polinómios de grau respectivamente p e q, tal que se anulam tantos coeficientes quanto os possíveis da série do erro $D^{[p/q]}(z)f(z) - N^{[p/q]}(z)$.

Os AFP $[p/q]_f^P$, tal como os aproximantes de Padé, dispõem-se numa tabela de dupla entrada correspondendo p ao número de linha e q ao número de coluna que designaremos por Tabela de Frobenius-Padé (TFP).

Nesta tese deduzem-se novas relações de recorrência envolvendo os coeficientes do numerador, do denominador e da série do erro de aproximantes pertencentes a duas diagonais descendentes adjacentes da TFP. A introdução de mais um parâmetro livre nas relações de recorrência revelou um aumento substancial da estabilidade no cálculo dos AFP.

São apresentados e programados, em Fortran90 e no Mathematica, dois algoritmos de cálculo recursivo dos aproximantes baseados nestas relações.

Um algoritmo corresponde à implementação directa das relações encontradas. O outro corresponde a uma modificação do primeiro, onde intervém uma decomposição LU com pivotagem parcial, via método de Doolittle, com vista ao melhoramento da estabilidade numérica do cálculo de quatro parâmetros τ, λ, η e ρ que se revelam fundamentais nas relações de recorrência.

Os programas em Fortran90 foram adaptados de forma a utilizar a biblioteca CADNA (Control of Accuracy and Debugging Numerical Applications) que tem por função avaliar o efeito acumulativo dos erros de cálculo considerando apenas os algarismos correctamente calculados.

As capacidades de cálculo formal e numérico do Mathematica, assim como as funcionalidades do CADNA, foram essenciais para efectuar o estudo aqui apresentado. Nomeadamente, a análise do condicionamento do cálculo dos AFP exigiu realizar os cálculos formalmente com o Mathematica. Por outro lado, a estabilidade do cálculo dos coeficientes dos AFP foi avaliada com o auxilio do CADNA.

Os programas foram testados com séries de polinómios de Jacobi (Legendre e Chebyshev de segunda espécie), de Hermite, de Laguerre e de Bessel. Os resultados obtidos revelam:

- um melhoramento significativo da estabilidade numérica no cálculo destes aproximantes relativamente aos programas anteriormente desenvolvidos,
- pequenas variações nos dados iniciais, ou seja, nos coeficientes f_i, produzem variações limitadas nos parâmetros τ, λ, η e ρ e também nos coeficientes a_i do numerador e b_i do denominador dos aproximantes,
- pequenas perturbações nos coeficientes a_i e b_i são absorvidas no cálculo dos aproximantes $[p/q]_f^P$ tornando-se praticamente imperceptíveis,
- o bom condicionamento do problema em causa, uma vez que a perturbação nos f_i transmite-se apenas em verdadeira grandeza aos aproximantes $[p/q]_f^P$,
- a boa qualidade da aproximação da soma da série fornecida pelos aproximantes $[p/q]_f^P$ relativamente à soma parcial correspondente $\sum_{i=0}^{p+2q} f_i P_i$, uma vez que o cálculo de $[p/q]_f^P$ exige o conhecimento de $\{f_i\}_{i=0}^{p+2q}$.

No que diz respeito ao algoritmo modificado, os resultados revelam valores dos parâmetros τ, λ, η e ρ diferentes dos obtidos sem decomposição LU. Estes novos valores conduzem a um ligeiro melhoramento de alguns dos coeficientes a_i e b_i, melhoramento esse que se traduz de forma visível no cálculo dos aproximantes $[p/q]_f^P$. Esse aumento de qualidade não se revela muito expressivo devido ao facto do problema em causa ser bem condicionado, já que pequenas modificações nos coeficientes a_i e b_i, sejam no sentido de melhorar ou de piorar a sua qualidade, se propagam de forma praticamente imperceptível ao cálculo dos AFP.

Em conclusão, os programas desenvolvidos neste trabalho constituem um avanço no cálculo estável dos AFP relativamente ao software anteriormente existente. O problema do condicionamento do cálculo destes aproximantes foi, nesta tese, pela primeira vez estudado. Os testes realizados incluem séries ortogonais dos quatro tipos de polinómios clássicos.

Abstract

Frobenius-Padé is the name given to the class of rational approximants which, in the sense of Frobenius definition, generalizes the Padé approximation to the orthogonal series. In other words, suppose you know a formal series expansion of $f(z)$ in orthogonal polynomials $\{P_i\}_{i\geq 0}$, i.e., $f(z) = \sum_{i=0}^{\infty} f_i P_i(z)$. For each pair of non-negative integers p and q, the Frobenius-Padé approximant (AFP) of order $[p/q]_f^P$ is the rational function $N^{[p/q]}(z)/D^{[p/q]}(z)$, where $N^{[p/q]}(z) = \sum_{i=0}^{p} a_i P_i(z)$ and $D^{[p/q]}(z) = \sum_{i=0}^{q} b_i P_i(z)$ are polynomials of degree p and q, respectively, such that, in the series $D^{[p/q]}(z)f(z) - N^{[p/q]}(z)$ as many of the first terms as possible are equal to zero.

The AFP $[p/q]_f^P$, as well as the Padé approximants, can be arranged in a double entry table, where p corresponds to the row and q to the column, that is called Frobenius-Padé Table (TFP).

In this thesis new recursive relations are deduced involving the coefficients of the numerator, denominator and error series of the approximants belonging to two adjacent descendant diagonals of the TFP. The introduction of a new parameter in the recursive relations revealed a substantial increase of the numerical stability of the calculus of the AFP.

Two algorithms for the recursive evaluation of AFP are implemented in Fortran90 and Mathematica.

The first one corresponds it a direct implementation. The second uses a LU decomposition with partial pivoting (Doolittle method). The purpose of the latter is to improve the numerical stability in the evaluation of the parameters τ, λ, η and ρ. These parameters play a crucial role in the study of the recursive relations.

The programmes written in Fortran90 were adapted in order use the CADNA library (Control of Accuracy and Debugging Numerical Applications) to evaluate the cumulative effects of the computational errors and to consider only the digits that are correct in the results.

The capabilities of the formal and numerical calculus of Mathematica, as well as CADNA, were essential tools to perform the study presented in this thesis. Namely, the analysis of the conditioning of the AFP calculus was performed in Mathematica using formal calculation. On the other hand, the stability of the calculus of the AFP coefficients was evaluated using the CADNA library.

The programmes were tested with series of Jacobi (Legendre and Chebyshev of the second kind), Hermite, Laguerre and Bessel polynomials. The results reveal:

- a significant improvement of the numerical stability of the calculus of theses approximants in order to the pre-existing programmes presented,

- small variations on the initial data, that is, in the coefficients f_i of the series expansion of $f(z)$ in orthogonal polynomials, produce limited variations on the numerator and denominator coefficients,
- small perturbations on the coefficients a_i and b_i are absorbed in the calculus of the approximants $[p/q]_f^P$ becoming imperceptive in practice,
- the great conditioning of the discussed problem, if you introduce some effective uncertainties on the f_i, reducing their initial accuracy, they propagate in the same magnitude along the calculus of the approximants $[p/q]_f^P$,
- the good quality of the approximation to the sum of the series given by the approximants $[p/q]_f^P$ when compared with the result obtained by the partial sum $\sum_{i=0}^{p+2q} f_i P_i$, since the calculus of $[p/q]_f^P$ demands the knowledge of $\{f_i\}_{i=0}^{p+2q}$.

With regard to the modified algorithm, the results reveal that the value of the parameters τ, λ, η and ρ are different from those obtained without LU decomposition. These new values lead to a slight improvement of the coefficients a_i and b_i. This improvement is obvious in the calculus of the approximants $[p/q]_f^P$ due to the discussed problem being well conditioned. Any small changes in the coefficients a_i and b_i, either done with the purpose of improving or degrading their quality, don't propagate in the AFP procedure.

In conclusion, the programmes developed in this work establish an advance on the calculus of the AFP when compared to the pre-existent software. The conditioning calculus issue of these approximants was, in the thesis, studied for the first time ever. The testes involve orthogonal series of the four types of classic polynomials.

Résumé

Le nom de Frobenius-Padé est attribué à la classe des approximats rationnels qui, dans le sens de la définition de Frobenius, généralise l'approximation de Padé de séries orthogonaux. C'est-à-dire, si un développement formel en série de polynômes orthogonaux $\{P_i\}_{i\geq 0}$ est donné pour la fonction $f(z)$, i.e., $f(z) = \sum_{i=0}^{\infty} f_i P_i(z)$, pour chaque paire de valeurs entières non négatives p e q, on désigne par l'approximant de Frobenius-Padé (AFP), $[p/q]_f^P$, la fonction rationnelle $N^{[p/q]}(z)/D^{[p/q]}(z)$, où $N^{[p/q]}(z) = \sum_{i=0}^{p} a_i P_i(z)$ et $D^{[p/q]}(z) = \sum_{i=0}^{q} b_i P_i(z)$ sont des polynômes des degrés respectivement p e q, tels que sont nuls les premiers coefficients de la série de l'erreur $D^{[p/q]}(z)f(z) - N^{[p/q]}(z)$, jusqu'à l'ordre aussi grande qui possible.

De la même manière que les approximants de Padé, les AFP $[p/q]_f^P$ se disposent dans um table où p correspond à la ligne et q à la colonne. Cette table se désigne par la Table des approximats de Frobenius-Padé (TFP).

Dans ce travail on déduit des nouvelles relations de récurrence entre les coefficients du numérateur, du dénominateur et de la série de l'erreur des approximats qui appartiennent à deux diagonales descendantes adjacentes dans la TFP. La introduction d'un nouveau paramètre dans les relations de récurrence a révèle un augmentation considérable de la stabilité du calcul des AFP.

Basés sur ces relations, deux algorithmes pour le calcul récursif des approximats sont présentés et programmés en Fortran90 et en Mathematica.

Un algorithme correspond à l'implémentation directe des relations trouvées. L'autre correspond à une modification du premier où il intervient une décomposition LU avec pivotage partiale, selon la méthode de Doolittle, en vue de l'amélioration de la stabilité numérique du calcul de quatre paramètres τ, λ, η et ρ qui se révèlent fondamentaux dans les relations de récurrence.

Les programmes en Fortran90 ont été adaptés pour pouvoir utiliser la bibliothèque CADNA (Control of Accuracy and Debugging Numerical Applications) qui a comme fonction évaluer l'effet cumulatif des erreurs de calcul an considérant seulement les chiffres correctement calculés.

Les capacités de calcul formel et numérique du Mathematica, ainsi que les fonctionnalités du CADNA, ont été essentielles pour effectuer l'étude ici présenté. Notamment, l'analyse du conditionnement du problème de calcul des AFP a exigé la réalisation les calculs formellement avec Mathematica. D'autre part, la stabilité du calcul des coefficients des AFP a été évalué avec l'aide du CADNA.

Les programmes ont été testés avec des séries de polynômes de Jacobi (Legendre et

Chebyshev de seconde espèce), de Hermite, de Laguerre et de Bessel. Les résultats obtenus révèlent:

- une amélioration significative de la stabilité numérique du calcul de ces approximats relativement aux programmes antérieurement développés,
- des petites variations dans les donnés initiales, c'est-à-dire, dans les coefficients f_i, produisent des variations limités dans les parametres τ, λ, η et ρ et aussi dans les coefficients a_i du numérateur et b_i du dénominateur des approximats,
- des petites perturbations dans les coefficients a_i et b_i sont absorbées dans le calcul des approximats $[p/q]_f^P$ devenant ainsi pratiquement inaperçues,
- le bon conditionnement du problème en cause, vu que la perturbation des f_i se transmet avec le même grandeur aux approximants $[p/q]_f^P$,
- la bonne qualité d'approximation de la somme de la série fournie par les approximats $[p/q]_f^P$ relativement à la somme partiale correspondante $\sum_{i=0}^{p+2q} f_i P_i$, sachant que le calcul de $[p/q]_f^P$ exige la connaissance de $\{f_i\}_{i=0}^{p+2q}$.

Par rapport à l'algorithme modifié, les résultats révèlent des valeurs des paramètres τ, λ, η et ρ différents de ceux obtenus sans la décomposition LU. Ces nouvelles valeurs conduisent à une légère amélioration de quelque uns des coefficients a_i et b_i, qui se traduit de façon visible dans le calcul des approximats $[p/q]_f^P$. Cette augmentation de qualité ne se révèle pas très expressive dû au fait que le problème en cause est bien conditionné, c'est-à-dire, des petites modifications dans les coefficients a_i et b_i, dans le sens d'une amélioration ou d'une dégradation de sa qualité, se propagent de façon imperceptible au calcul des AFP.

En conclusion, les programmes développés dans ce travail constituent une avance dans le calcul stable des AFP par rapport aux logiciels antérieurement existant. Le problème du conditionnement du calcul de ces approximats a été, dans cette thèse, pour la première fois étudié. Les testes réalisés incluent des séries orthogonaux de quatre types de polynômes classiques.

Conteúdo

Capítulo 1

Introdução e motivação

Dada uma função $f(z)$ de que se conhecem, pelo menos os primeiros coeficientes do seu desenvolvimento formal em série de potências

$$f(z) = \sum_{i \geq 0} f_i z^i \tag{1.1}$$

define-se o aproximante de Padé (AP) de $f(z)$ como sendo a função racional

$$[p/q]_f(z) = N(z)/D(z) \tag{1.2}$$

onde $N(z)$ e $D(z)$ são polinómios de grau p e q respectivamente, determinados pela condição de o desenvolvimento em série de Maclaurin de $[p/q]_f(z)$ coincidir com o de $f(z)$ tão longe quanto possível [1, 2, 3, 4].

Assim, o cálculo do aproximante de Padé $[p/q]_f(z)$ consiste na determinação dos $p+1$ coeficientes de

$$N(z) = \sum_{i=0}^{p} a_i z^i \tag{1.3}$$

e dos $q+1$ coeficientes de

$$D(z) = \sum_{i=0}^{q} b_i z^i. \tag{1.4}$$

Uma vez que em (1.2) existe um factor multiplicativo não nulo arbitrário entre $N(z)$ e $D(z)$, temos $p+q+1$ coeficientes a determinar, o que sugere que, normalmente, podemos esperar do aproximante que a sua expansão em série de potências coincida com a da função até à ordem z^{p+q}. Utilizando notação de séries formais, temos a seguinte definição.

Definição 1.1 *[1, 2, 3, 4] O aproximante de Padé, $[p/q]_f(z)$, é uma função racional $N(z)/D(z)$ tal que*

$$\begin{aligned} &D(z)f(z) - N(z) = \mathcal{O}(z^{p+q+1}) \\ &\partial N \leq p,\ \partial D \leq q \end{aligned} \tag{1.5}$$

O símbolo ∂P utiliza-se para designar o grau do polinómio P e o símbolo $\mathcal{O}(z^n)$ para designar uma série cujos coeficientes até ao termo de grau $n-1$ são nulos.

Substituindo (1.1), (1.3) e (1.4) em (1.5) encontramos as equações

$$\sum_{i+j=k} b_i f_j = a_k,\ k = 0, \ldots, p \tag{1.6}$$

$$\sum_{i+j=k} b_i f_j = 0,\ k = p+1, \ldots, p+q \tag{1.7}$$

As equações (1.7) constituem um sistema homogéneo de q equações lineares nas $q+1$ incógnitas $b_0, \ldots, b_q$. Existe assim uma infinidade de soluções correspondentes ao parâmetro livre, por exemplo, b_0. Fixando $b_0 = 1$ obtemos uma solução única. Esta escolha assegura que o aproximante $[p/q]_f$ está definido no ponto zero. Calculados os b_i, $i = 0, \ldots, q$, com (1.6) calcula-se directamente os valores de a_i, $i = 0, \ldots, p$.

O trabalho desenvolvido nesta tese, envolve uma classe de aproximantes racionais que constituem uma generalização da aproximação de Padé. A generalização consiste em aplicar a mesma definição de aproximação a outro tipo de séries, que não as séries de potências.

Seja $f(z)$ uma função definida por um desenvolvimento em série ortogonal

$$f(z) = \sum_{i \geq 0} f_i P_i(z) \tag{1.8}$$

onde $\{P_i\}_{i \geq 0}$ constitui uma família de polinómios ortogonais, relativamente a uma funcional linear u, i.e.,

$$\langle u, P_i P_j \rangle = \mu_i \delta_{ij}, \quad \text{com } \mu_i \neq 0, \quad i, j \geq 0.$$

Decorre que $f_i = \frac{1}{\mu_i} \langle u, f P_i \rangle$.

Nos exemplos mais conhecidos existe uma representação integral de u, ou seja,

$$\langle u, \cdot \rangle \equiv \int_a^b \cdot(x) \omega(x) dx, \qquad -\infty \leq a < b \leq +\infty, \tag{1.9}$$

onde $\omega(x)$ é designada por *função peso* que satisfaz determinadas propriedades.

Assim, teremos

$$\langle u, P_i P_j \rangle \equiv \int_a^b P_i(x) P_j(x) \omega(x) dx = \mu_i \delta_{ij}, \qquad i, j \geq 0, \tag{1.10}$$

com $\mu_i = \langle u, P_i P_i \rangle = \|P_i\|_2^2 \neq 0$, $i \geq 0$.

Os aproximantes a considerar, designados por aproximantes de Frobenius-Padé (AFP) de $f(z)$ [12], constituem a função racional

$$[p/q]_f^P(z) = \frac{N(z)}{D(z)} \equiv \frac{\sum_{i=0}^p a_i P_i(z)}{\sum_{i=0}^q b_i P_i(z)} \tag{1.11}$$

tal que

$$D(z)f(z) - N(z) = \mathcal{O}(P_{p+q+1}(z)) \tag{1.12}$$

onde $\mathcal{O}(P_{p+q+1}(z))$ designa uma série ortogonal com os primeiros $p+q+1$ termos nulos [17, 18].

Os aproximantes de Frobenius-Padé $[p/q]_f^P$, tal como os aproximantes de Padé, dispõem-se numa tabela de dupla entrada correspondendo p ao número de linha e q ao número de coluna que designaremos por Tabela de Frobenius-Padé (TFP).

O tema deste trabalho consiste na análise e melhoramento da estabilidade numérica no cálculo recursivo dos coeficientes a_i do numerador e b_i do denominador dos AFP, assim como no estudo do condicionamento do cálculo destes aproximantes.

O cálculo recursivo dos AFP foi objecto de uma tese de doutoramento [18] onde, em particular, se apresentam relações de recorrência, algoritmos e testes correspondentes ao cálculo de duas diagonais descendentes adjacentes na TFP. O estudo e melhoramento da estabilidade deste cálculo constitui a principal motivação deste trabalho.

A tese de Mestrado aqui exposta foi precedida de um trabalho de Seminário [19] onde o programa escrito em Fortran90, desenvolvido em [18], foi alterado no sentido de implementar a biblioteca CADNA (Control of Accuracy and Debugging Numerical Applications), quer em precisão simples quer em precisão dupla, com vista ao tratamento do efeito acumulativo dos erros de cálculo (ver apêndices C e E). Esse trabalho de Seminário é apresentado de forma sucinta na secção 3.1 desta tese.

Já em [18], os testes numéricos sugerem uma forte presença de instabilidade, confirmada por testes suplementares realizados neste trabalho. Na secção 3.1, é feita uma análise de resultados comparando os programas, executados com e sem o CADNA. Tivemos oportunidade de confirmar, com a implementação do CADNA, que realmente existem instabilidades e com o auxilio desta biblioteca tomámos conhecimento do tipo de instabilidades e do local onde as instabilidades são produzidas.

Um dos principais objectivos deste trabalho foi ultrapassar as instabilidades numéricas, detectadas pelo CADNA no algoritmo de [18], modificando as expressões as relações de recorrência correspondentes no sentido de evitar a perda de precisão. Esse objectivo foi alcançado com êxito.

Neste trabalho apresenta-se relações de recorrência inéditas entre os numeradores $N(z)$ e denominadores $D(z)$ de quatro AFP consecutivos pertencentes a duas diagonais descendentes adjacentes da TFP. Consequentemente, deduzimos novas relações de recorrência entre os coeficientes a_i dos numeradores e b_i dos denominadores desses quatro aproximantes. Finalmente, a partir destes coeficientes são construídos os aproximantes $[p/q]_f^P$ utilizando a formula (1.11).

Descrevemos seguidamente o conteúdo deste trabalho.

O capítulo 2 trata do problema do cálculo recursivo de sucessões de aproximantes. Neste capítulo, apresentam-se resultados novos, estabelecendo novas relações entre as componentes, i.e., entre os coeficientes do numerador, do denominador e da série do erro, de aproximantes pertencentes a duas diagonais descendentes adjacentes na TFP. A introdução de mais um parâmetro livre nas relações de recorrência revelou um aumento substancial da estabilidade

no cálculo dos AFP. Estas relações, generalizando resultados análogos na tabela de Padé [3, 4], permitem percorrer várias diagonais da tabela e mostram-se apropriadas para a implementação em algoritmos de cálculo recursivo dos aproximantes.

Correspondentes a estas novas relações, surge um novo algoritmo de cálculo apresentado na secção 2.4. Este algoritmo foi avaliado e comparado com o anterior algoritmo descrito em [18] em termos da estabilidade do cálculo e qualidade de aproximação da série.

Os testes realizados usando o novo algoritmo revelaram que este é suficientemente estável para calcular as aproximações racionais com grande precisão. As sucessões de aproximantes calculadas, produzem valores convergentes para os valores exactos.

Com o objectivo de progredir no sentido do aumento da estabilidade do cálculo dos AFP, modificámos o novo algoritmo substituindo quatro fórmulas algébricas, correspondentes ao cálculo de quatro parâmetros τ, λ, η e ρ, pela resolução de um sistema 4×4 usando uma decomposição LU com pivotagem parcial. Para as várias famílias de polinómios clássicos [11, 14, 15, 16, 18], o algoritmo modificado foi testado, tendo revelado um ligeiro melhoramento nalguns dos coeficientes que se traduz de forma visível ao cálculo dos AFP, demonstrando uma das vantagens da dissolução da estrutura triangular ou quase-triangular das matrizes em causa.

No capítulo 4 analisámos o condicionamento do problema de cálculo dos AFP. Para estudar o efeito da propagação de pequenas perturbações nos dados iniciais, os coeficientes do desenvolvimento em série ortogonal da função a aproximar foram perturbados no Mathematica e com esses valores perturbados foram construídos os aproximantes.

Há que distinguir três sub-problemas neste estudo: o primeiro corresponde ao cálculo dos parâmetros τ, λ, η e ρ; o segundo corresponde ao cálculo dos coeficientes a_i e b_i e finalmente o terceiro ao cálculo dos aproximantes $[p/q]_f^P$ propriamente ditos, i.e.,

$$f_i - (1) \rightarrow (\tau, \lambda, \eta, \rho) - (2) \rightarrow (a_i, b_i) - (3) \rightarrow [p/q]_f^P .$$

Constatámos que o problema (3) de cálculo dos aproximantes $[p/q]_f^P$ a partir dos coeficientes a_i do numerador $N^{[p/q]}(z)$ e b_i do denominador $D^{[p/q]}(z)$ revela-se altamente bem condicionado pois as perturbações nos a_i e b_i são absorvidas, produzindo aproximantes $[p/q]_f^P = \frac{N^{[p/q]}(z)}{D^{[p/q]}(z)}$ de excelente qualidade.

Verificámos que a perturbação inicial nos f_i é ampliada no cálculo dos parâmetros (1) e dos coeficientes (2), para depois ser substancialmente reduzida no cálculo efectivo dos aproximantes $[p/q]_f^P$ (3). Assim se observarmos o problema principal, partindo do cálculo dos f_i e terminando no cálculo dos aproximantes $[p/q]_f^P$, podemos afirmar que se trata de um problema muito bem condicionado pois a perturbação nos f_i transmite-se apenas em verdadeira grandeza aos aproximantes.

Em conclusão, os programas desenvolvidos neste trabalho constituem um avanço no cálculo estável dos AFP relativamente ao software anteriormente existente. O problema do condicionamento do cálculo destes aproximantes foi, nesta tese, pela primeira vez estudado. Os testes realizados incluíram séries ortogonais dos quatro tipos de polinómios clássicos, o que é inédito na literatura sobre o tema.

Vejamos de que forma podemos prosseguir com o trabalho desenvolvido nesta tese.

Uma vez que conseguimos um programa robusto e eficaz para o cálculo recursivo dos aproximantes de Frobenius-Padé será nosso objectivo aplicar este procedimento a problemas complexos da Física e Engenharia.

Em [18] apresenta-se um conjunto de relações de recorrência na TFP que permite estabelecer diversos algoritmos de cálculo recursivo de sucessões de aproximantes. Alguns desses algoritmos foram programados e testados, outros não.

O trabalho apresentado nesta tese deve ser aplicado a cada um destes procedimentos de forma a melhorar a estabilidade do cálculo correspondente. Além disso, o estudo do condicionamento, tal como aqui é exposto, deve repetir-se para todas as relações.

Em [8] e [18, capítulo 5] são introduzidos pela primeira vez os aproximantes de Frobenius-Padé (d-AFP) para séries de polinómios d-ortogonais ($d > 1$). Neste trabalho consideramos apenas $d = 1$. O cálculo dos d-AFP, para $d > 1$, efectua-se de forma análoga ao caso $d = 1$, embora tecnicamente seja bastante mais complexo. Assim, existem várias relações de recorrência entre d-AFP, que conduzem a algoritmos de cálculo recursivo análogos aos apresentados nesta tese. Os testes realizados revelam que o cálculo destes aproximantes comporta-se numericamente de forma semelhante ao caso $d = 1$ [8]. O trabalho desenvolvido nesta tese poderá extender-se adequadamente ao cálculo dos d-AFP.

Por fim, referimos o conteúdo dos apêndices que figuram no final desta tese.

Os gráficos e tabelas dos vários exemplos dos capítulos 2 e 4, que suportam as conclusões apresentadas neste trabalho, podem ser consultadas nos apêndices (A) e (B) respectivamente.

No apêndice (C) apresentamos as funcionalidades da biblioteca CADNA (Control of Accuracy and Debugging Numerical Applications) e ilustramos de que forma deve ser implementada [5].

O CADNA, utilizando aritmética estocástica, tem como principal objectivo estimar o efeito da propagação dos erros de arredondamento e de cálculo. Assistidos pelas funções e subrotinas da biblioteca CADNA temos acesso ao tipo de instabilidades numéricas que ocorrem no programa e utilizando um *debugger* são identificadas as expressões que causam as instabilidades.

No apêndice (D) são apresentadas algumas explicações sobre o funcionamento do software Mathematica [22]. A necessidade de extremo rigor na análise e interpretação dos vários resultados produzidos pelo Mathematica exigiu um estudo pormenorizado da forma como este software efectua o cálculo. Os aspectos focados prendem-se essencialmente com os comandos e funções que foram utilizados para a realização deste trabalho.

No apêndice (E) apresentamos o código do programa, em Fortran90, que implementa o novo algoritmo descrito na secção 2.4, bem como a parte fundamental do código do programa que implementa o algoritmo modificado. Ambos se encontram alterados no sentido de utilizar a biblioteca CADNA. O correspondente programa em Mathematica foi omitido por seguir exactamente as mesmas instruções do programa em Fortran90, salvo algumas questões que se prendem apenas com a sintaxe do software.

Capítulo 2

Novas fórmulas de cálculo dos Aproximantes de Frobenius-Padé

Este capítulo trata do problema do cálculo recursivo de sucessões de AFP [8, 12, 17, 18].

Deduzem-se resultados novos, estabelecendo relações inéditas entre as componentes, i.e., entre os coeficientes do numerador, do denominador e da série do erro, de aproximantes pertencentes a duas diagonais descendentes adjacentes na TFP.

Considerando as novas relações de recorrência surge um novo algoritmo de cálculo. Este novo algoritmo foi testado e os resultados revelam um aumento expressivo da estabilidade no cálculo dos coeficientes dos AFP.

2.1 Notações e definições

Seja $\{P_i(z)\}_{i\geq 0}$ uma família de polinómios ortogonais relativamente a uma funcional linear ou forma u, i.e.

$$\langle u, P_i P_j \rangle = \mu_i \delta_{ij}, \quad i,j = 0,1,\ldots$$

onde $\mu_i = \langle u, P_i P_i \rangle = \|P_i(z)\|_2^2 \neq 0$ e δ_{ij} é o símbolo de Kronecker. Desta forma, $\{P_i(z)\}_{i\geq 0}$, entre outras propriedades características conhecidas, satisfaz uma relação de recorrência de ordem 2

$$P_{i+1}(z) = \frac{z-\beta_i}{\alpha_i} P_i(z) - \frac{\gamma_i}{\alpha_i} P_{i-1}(z)$$

$$P_0(z) = 1, \quad P_1(z) = \frac{z-\beta_0}{\alpha_0}$$

tal que os coeficientes $\{\alpha_i\}_{i\geq 0}$, $\{\beta_i\}_{i\geq 0}$ e $\{\gamma_i\}_{i\geq 0}$ existem e $\alpha_i \neq 0$ e $\gamma_i \neq 0$ para $i \geq 1$.

Seja $f(z)$ uma função de variável complexa, dada pelo desenvolvimento ortogonal

$$f(z) = \sum_{i\geq 0} f_i P_i(z)$$

então

$$f_i = \frac{1}{\mu_i} \langle u, f P_i \rangle$$

Para estas funções, definem-se os seguintes aproximantes racionais.

Definição 2.1 *[12, 17, 18] Para cada par de valores $p, q \in \mathbb{N}$, define-se o aproximante de Frobenius-Padé (AFP) de ordem (p, q) de $f(z)$, como sendo a função racional*

$$[p/q]_f^P(z) = \frac{N^{[p/q]}(z)}{D^{[p/q]}(z)} \equiv \frac{\sum_{i=0}^{p} a_i P_i(z)}{\sum_{i=0}^{q} b_i P_i(z)}$$

tal que

$$D^{[p/q]}(z) f(z) - N^{[p/q]}(z) = \mathcal{O}(P_{p+q+1}(z)),$$

onde $\mathcal{O}(P_{p+q+1}(z))$ designa uma série cujos coeficientes até ao termo de grau $p+q+1$ são nulos.

Desta definição resulta que os coeficientes $a_0, \ldots, a_p$ e $b_0, \ldots, b_q$ satisfazem a igualdade

$$\sum_{i=0}^{q} b_i P_i(z) f(z) - \sum_{i=0}^{p} a_i P_i(z) = \sum_{i \geq p+q+1} e_i P_i(z). \tag{2.1}$$

Tal como na aproximação de Padé, os AFP podem representar-se numa tabela bidimensional, onde $[p/q]_f^P(z)$ ocupa a posição definida pela linha p e a coluna q. Esta tabela designa-se por *Tabela de Frobenius-Padé* (TFP).

Definição 2.2 *[18] A tabela de Frobenius-Padé é normal se e só se:*

1. *Para cada par (p, q), temos que $\partial N^{[p/q]} = p$ e $\partial D^{[p/q]} = q$. Isto implica que todos os aproximantes de uma TFP são distintos;*
2. *O coeficiente do primeiro termo da expansão em série do erro (2.1), e_{p+q+1}, é não nulo.*

Um AFP $[p/q]_f^P(z)$ diz-se normalizado se e só se $b_q = 1$, i.e.,:

$$[p/q]_f^P(z) = \frac{\sum_{k=0}^{p} a_k P_k(z)}{P_q(z) + \sum_{j=0}^{q-1} b_j P_j(z)}.$$

Em todo este trabalho consideremos TFP normais e AFP normalizados. Assim sendo, temos sempre

$$\begin{cases} N^{[p/q]}(z) = \sum_{i=0}^{p} a_i^{[p/q]} P_i(z), & a_p^{[p/q]} \neq 0 \\ D^{[p/q]}(z) = \sum_{i=0}^{q} b_i^{[p/q]} P_i(z), & b_q^{[p/q]} = 1 \\ N^{[p/q]}(z) - D^{[p/q]}(z) f(z) = \sum_{i \geq p+q+1} e_i^{[p/q]} P_i(z), & e_{p+q+1}^{[p/q]} \neq 0 \end{cases}. \tag{2.2}$$

2.2 Relações de recorrência na Tabela de Frobenius-Padé

Tal como no caso da aproximação de Padé, onde é possível estabelecerem-se relações envolvendo elementos adjacentes da tabela de Padé [3, 4], também no caso da aproximação de Frobenius-Padé se podem encontrar relações entre elementos adjacentes da TFP.

Para estabelecer as relações de recorrência entre os coeficientes de aproximantes adjacentes na TFP, é necessário recorrer repetidas vezes às fórmulas que estabelecem a relação entre os coeficientes do desenvolvimento ortogonal de uma dada função e os coeficientes do desenvolvimento ortogonal do produto dessa função pelo monómio z. Tais fórmulas encontram-se estabelecidas na seguinte proposição.

Proposição 2.1 *[8, 18] Se* $\{P_i\}_{i\geq 0}$ *é a família de polinómios ortogonais que satisfaz a relação de recorrência*

$$\begin{cases} P_{i+1}(z) = \frac{\beta_i - z}{\alpha_i} P_i(z) + \frac{\gamma_i}{\alpha_i} P_{i-1}(z), \ i \geq 1 \\ P_0(z) = 1, \ P_1(z) = \frac{z-\beta_0}{\alpha_0}, \quad \alpha_i, \gamma_i \neq 0, \ i \geq 0 \end{cases}, \tag{2.3}$$

e

1. $f(z)$ *é a série*
$$f(z) = \sum_{i\geq 0} f_i P_i(z),$$

2. $p(z)$ *é o polinómio*
$$p(z) = \sum_{i=0}^{n} p_i P_i(z),$$

3. $e(z)$ *é a série truncada*
$$e(z) = \sum_{i\geq n} e_i P_i(z),$$

então:

1.
$$zf(z) = \sum_{i\geq 0} g_i P_i(z), \quad \begin{cases} g_0 = \beta_0 f_0 + \gamma_1 f_1 \\ g_i = \alpha_{i-1} f_{i-1} + \beta_i f_i + \gamma_{i+1} f_{i+1}, \quad i \geq 1 \end{cases}, \tag{2.4}$$

2.
$$zp(z) = \sum_{i=0}^{n+1} h_i P_i(z), \quad \begin{cases} h_0 = \beta_0 p_0 + \gamma_1 p_1 \\ h_i = \alpha_{i-1} p_{i-1} + \beta_i p_i + \gamma_{i+1} p_{i+1}, \quad i = 1, \ldots, n-1 \\ h_n = \alpha_{n-1} p_{n-1} + \beta_n p_n \\ h_{n+1} = \alpha_n p_n \end{cases} \tag{2.5}$$

e

3.

$$ze(z) = \sum_{i \geq n-1} d_i P_i(z), \quad \begin{cases} d_{n-1} = \gamma_n e_n \\ d_n = \beta_n e_n + \gamma_{n+1} e_{n+1} \\ d_i = \alpha_{i-1} e_{i-1} + \beta_i e_i + \gamma_{i+1} e_{i+1}, \quad i \geq n+1 \end{cases} . \tag{2.6}$$

Os seguintes resultados permitem progredir na TFP calculando uma sucessão de elementos pertencentes a duas diagonais descendentes adjacentes.

Proposição 2.2 *[18] Sejam $p, q \geq 1$ tais que os aproximantes $[p-1/q-1]_f^P$, $[p/q-1]_f^P$ e $[p/q]_f^P$ existem e satisfazem as condições de normalidade. Definindo*

$$\begin{cases} N^{[p+1/q]} = \lambda N^{[p-1/q-1]} + (\eta + z) N^{[p/q-1]} + \rho N^{[p/q]} \\ D^{[p+1/q]} = \lambda D^{[p-1/q-1]} + (\eta + z) D^{[p/q-1]} + \rho D^{[p/q]} \end{cases} \quad \begin{matrix} \bullet \\ \bullet \ \bullet \\ * \end{matrix}$$

com

$$\begin{cases} \lambda = -\gamma_{p+q} \frac{e_{p+q}^{[p/q-1]}}{e_{p+q-1}^{[p-1/q-1]}} \\ \eta = -\frac{1}{e_{p+q}^{[p/q-1]}} (\lambda e_{p+q}^{[p-1/q-1]} + d_{p+q}^{[p/q-1]}) \\ \rho = -\frac{1}{e_{p+q+1}^{[p/q]}} (\lambda e_{p+q+1}^{[p-1/q-1]} + \eta e_{p+q+1}^{[p/q-1]} + d_{p+q+1}^{[p/q-1]}) \end{cases}$$

e

$$\Delta = \alpha_{q-1} + \rho.$$

Se $\Delta \neq 0$ então

$$\frac{N^{[p+1/q]}}{D^{[p+1/q]}} = [p+1/q]_f^P$$

é o AFP tal que $b_q^{[p+1/q]} = \Delta$. Caso contrário, a tabela Frobenius-Padé é não normal, sendo

$$D^{[p+1/q]} f - N^{[p+1/q]} = \mathcal{O}(P_{p+q+2})$$

com $\partial(N^{[p+1/q]}) = p+1$ e $\partial(D^{[p+1/q]}) \leq q-1$.

Proposição 2.3 *[18] Sejam $p \geq 0$ e $q \geq 1$ tais que os aproximantes $[p/q-1]_f^P$, $[p/q]_f^P$ e $[p+1/q]_f^P$ existem e satisfazem as condições de normalidade. Definindo*

$$\begin{cases} N^{[p+1/q+1]} = \lambda N^{[p/q-1]} + (\eta + \tau z) N^{[p/q]} + \rho N^{[p+1/q]} \\ D^{[p+1/q+1]} = \lambda D^{[p/q-1]} + (\eta + \tau z) D^{[p/q]} + \rho D^{[p+1/q]} \end{cases} \quad \begin{matrix} \bullet \ \bullet \\ \bullet \ * \end{matrix}$$

com

$$\begin{cases} \tau = 1/\alpha_q \\ \lambda = -\frac{\gamma_{p+q+1}}{\alpha_q} \frac{e_{p+q+1}^{[p/q]}}{e_{p+q}^{[p/q-1]}} \\ \eta = -\frac{1}{e_{p+q+1}^{[p/q]}} (e_{p+q+1}^{[p/q-1]} \lambda + d_{p+q+1}^{[p/q]} \tau) \\ \rho = -\frac{1}{e_{p+q+2}^{[p+1/q]}} (e_{p+q+2}^{[p/q-1]} \lambda + e_{p+q+2}^{[p/q]} \eta + d_{p+q+2}^{[p/q]} \tau) \end{cases},$$

então $N^{[p+1/q+1]}/D^{[p+1/q+1]} = [p+1/q+1]_f^P$ é o AFP normalizado com $b_{q+1}^{[p+1/q+1]} = 1$.

Em [18] é apresentado um algoritmo que, utilizando intercaladamente as duas relações

$$\begin{array}{ccc} \bullet & & \\ \bullet & \bullet & \\ * & & \end{array} \quad \text{e} \quad \begin{array}{ccc} & \bullet & \bullet \\ & \bullet & * \end{array},$$

permite calcular uma sucessão de aproximantes dispostos em duas diagonais adjacentes. Note-se que os aproximantes da primeira coluna da tabela são as somas parciais da série dada, i.e., $[p/0]_f^P(z) = \sum_{i=0}^{p} f_i P_i(z)$.

Este algoritmo foi programado e testado com diversos exemplos [18]. Os resultados obtidos exibem um aumento progressivo dos erros relativos no cálculo dos coeficientes dos aproximantes à medida que se progride nas diagonais descendentes, revelando a existência de instabilidade numérica nos cálculos produzidos por este algoritmo.

2.3 Novas relações de recorrência para o cálculo de duas diagonais descendentes adjacentes na TFP

No sentido de melhorar a estabilidade e o condicionamento do problema de cálculo dos parâmetros λ, η e ρ na relação de recorrência $\begin{array}{ccc} \bullet & & \\ \bullet & \bullet & \\ * & & \end{array}$ pode introduzir-se mais um parâmetro, τ nas igualdades da proposição 2.2, a determinar impondo as condições de normalidade (2.2) nos aproximantes envolvidos.

Para $p, q \geq 1$ consideremos a relação

$$\begin{cases} N^{[p+1/q]} = \lambda N^{[p-1/q-1]} + (\eta + \tau z) N^{[p/q-1]} + \rho N^{[p/q]} \\ D^{[p+1/q]} = \lambda D^{[p-1/q-1]} + (\eta + \tau z) D^{[p/q-1]} + \rho D^{[p/q]} \end{cases} \qquad \begin{array}{ccc} \bullet & & \\ \bullet & \bullet & \\ * & & \end{array}$$

onde λ, η, τ e ρ são números reais, independentes de z, dependentes dos índices p e q, a determinar impondo (2.2).

Considerando a igualdade $N^{[p+1/q]} = \lambda N^{[p-1/q-1]} + (\eta + \tau z) N^{[p/q-1]} + \rho N^{[p/q]}$, podemos expandir os vários termos para encontrar condições entre os coeficientes dos numeradores

$$\begin{aligned} \sum_{i=0}^{p+1} a_i^{[p+1/q]} P_i &= \lambda \sum_{i=0}^{p-1} a_i^{[p-1/q-1]} P_i + \eta \sum_{i=0}^{p} a_i^{[p/q-1]} P_i + \tau z \sum_{i=0}^{p} a_i^{[p/q-1]} P_i + \rho \sum_{i=0}^{p} a_i^{[p/q]} P_i \\ &= \lambda \sum_{i=0}^{p-1} a_i^{[p-1/q-1]} P_i + \eta \sum_{i=0}^{p} a_i^{[p/q-1]} P_i + \tau \sum_{i=0}^{p+1} g_i^{[p/q-1]} P_i + \rho \sum_{i=0}^{p} a_i^{[p/q]} P_i \end{aligned}$$

onde os coeficientes $g_i^{[p/q-1]}$ são obtidos por (2.5).

Note-se que para esta igualdade ser válida é necessário que os polinómios de ambos os membros tenham o mesmo grau. Recorde-se que os coeficientes estão normalizados de forma que $a_p^{[p/q-1]} \neq 0$ o que implica que $\partial \left(\sum_{i=0}^{p} a_i^{[p/q-1]} P_i \right) = p$.

O facto de $\partial \left(\sum_{i=0}^{p+1} a_i^{[p+1/q]} P_i \right) = p+1$ obriga $\partial \left(\tau z \sum_{i=0}^{p} a_i^{[p/q-1]} P_i \right) = p+1$, logo, para que a igualdade se verifique, é condição necessária que $\tau \neq 0$.

Consideremos agora $D^{[p+1/q]} = \lambda D^{[p-1/q-1]} + (\eta + \tau z) D^{[p/q-1]} + \rho D^{[p/q]}$. Temos que

$$\begin{aligned}\sum_{i=0}^{q} b_i^{[p+1/q]} P_i &= \lambda \sum_{i=0}^{q-1} b_i^{[p-1/q-1]} P_i + \eta \sum_{i=0}^{q-1} b_i^{[p/q-1]} P_i + \tau z \sum_{i=0}^{q-1} b_i^{[p/q-1]} P_i + \rho \sum_{i=0}^{q} b_i^{[p/q]} P_i \\ &= \lambda \sum_{i=0}^{q-1} b_i^{[p-1/q-1]} P_i + \eta \sum_{i=0}^{q-1} b_i^{[p/q-1]} P_i + \tau \sum_{i=0}^{q} h_i^{[p/q-1]} P_i + \rho \sum_{i=0}^{q} b_i^{[p/q]} P_i\end{aligned}$$

onde os coeficientes $h_i^{[p/q-1]}$ são obtidos por (2.5).

Analisando os coeficientes de maior índice, extraímos a primeira igualdade

$$b_q^{[p+1/q]} = \tau h_q^{[p/q-1]} + \rho b_q^{[p/q]}.$$

Segundo a normalização estabelecida tem-se que $b_q^{[p+1/q]} = 1$ e $b_q^{[p/q]} = 1$. Usando (2.5), $h_q^{[p/q-1]} = \alpha_{q-1} b_{q-1}^{[p/q-1]}$. Simplificando por normalidade, obtemos $h_q^{[p/q-1]} = \alpha_{q-1}$ e conseguimos a primeira equação

$$\tau \alpha_{q-1} + \rho = 1. \tag{2.7}$$

Por último façamos uma análise análoga das séries dos erros. Consideremos

$$\begin{aligned}\sum_{i \geq p+q+2} e_i^{[p+1/q]} P_i &= \lambda \sum_{i \geq p+q-1} e_i^{[p-1/q-1]} P_i + \eta \sum_{i \geq p+q} e_i^{[p/q-1]} P_i + \tau z \sum_{i \geq p+q} e_i^{[p/q-1]} P_i + \rho \sum_{i \geq p+q+1} e_i^{[p/q]} P_i \\ &= \lambda \sum_{i \geq p+q-1} e_i^{[p-1/q-1]} P_i + \eta \sum_{i \geq p+q} e_i^{[p/q-1]} P_i + \tau \sum_{i \geq p+q-1} d_i^{[p/q-1]} P_i + \rho \sum_{i \geq p+q+1} e_i^{[p/q]} P_i\end{aligned}$$

onde os coeficientes $d_i^{[p/q-1]}$ são obtidos por (2.6).

Atendendo que $\{P_i\}_{i \geq 0}$ é uma base de polinómios, em ambos os membros desta igualdade, os coeficientes dos mesmos polinómios coincidem. Assim, como no membro esquerdo da igualdade a série do erro começa em P_{p+q+2}, no membro direito os coeficientes de P_{p+q-1}, P_{p+q} e P_{p+q+1} devem anular-se.

Considerando os coeficientes dos termos de grau $p+q-1$, $p+q$ e $p+q+1$, concluímos que é necessário

$$\begin{cases} \lambda e_{p+q-1}^{[p-1/q-1]} + \tau d_{p+q-1}^{[p/q-1]} = 0 \\ \lambda e_{p+q}^{[p-1/q-1]} \eta + e_{p+q}^{[p/q-1]} + \tau d_{p+q}^{[p/q-1]} = 0 \\ \lambda e_{p+q+1}^{[p-1/q-1]} + \eta e_{p+q+1}^{[p/q-1]} + \tau d_{p+q+1}^{[p/q-1]} + \rho e_{p+q+1}^{[p/q]} = 0 \end{cases}$$

para que o segundo membro constitua uma série de ordem $p+q+2$.

Da proposição 2.1 vem que $d_{p+q-1}^{[p/q-1]} = \gamma_{p+q} e_{p+q}^{[p/q-1]}$. Desta forma, as equações precedentes juntamente com (2.7) constituem o seguinte sistema

$$\begin{cases} \tau \alpha_{q-1} + \rho = 1 \\ \tau \gamma_{p+q} e_{p+q}^{[p/q-1]} + \lambda e_{p+q-1}^{[p-1/q-1]} = 0 \\ \tau d_{p+q}^{[p/q-1]} + \lambda e_{p+q}^{[p-1/q-1]} + \eta e_{p+q}^{[p/q-1]} = 0 \\ \tau d_{p+q+1}^{[p/q-1]} + \lambda e_{p+q+1}^{[p-1/q-1]} + \eta e_{p+q+1}^{[p/q-1]} + \rho e_{p+q+1}^{[p/q]} = 0 \end{cases} . \tag{2.8}$$

Seja

$$\Delta = \det \begin{pmatrix} \alpha_{q-1} & 0 & 0 & 1 \\ \gamma_{p+q} e_{p+q}^{[p/q-1]} & e_{p+q-1}^{[p-1/q-1]} & 0 & 0 \\ d_{p+q}^{[p/q-1]} & e_{p+q}^{[p-1/q-1]} & e_{p+q}^{[p/q-1]} & 0 \\ d_{p+q+1}^{[p/q-1]} & e_{p+q+1}^{[p-1/q-1]} & e_{p+q+1}^{[p/q-1]} & e_{p+q+1}^{[p/q]} \end{pmatrix}$$

o determinante da matriz associada, então, como resultado desta análise, fica demonstrada a seguinte proposição.

Proposição 2.4 *Sejam $p, q \geq 1$ tais que os aproximantes $[p-1/q-1]_f^P$, $[p/q-1]_f^P$ e $[p/q]_f^P$ existem e satisfazem as condições de normalidade e seja $\Delta \neq 0$. Defina-se*

$$\begin{cases} N^{[p+1/q]} = \lambda N^{[p-1/q-1]} + (\eta + \tau z) N^{[p/q-1]} + \rho N^{[p/q]} \\ D^{[p+1/q]} = \lambda D^{[p-1/q-1]} + (\eta + \tau z) D^{[p/q-1]} + \rho D^{[p/q]} \end{cases} \quad \begin{matrix} \bullet & \\ \bullet & \bullet \\ & * \end{matrix}$$

com

$$\begin{cases} \tau = \frac{1}{\Delta} \left(e_{p+q-1}^{[p-1/q-1]} e_{p+q}^{[p/q-1]} e_{p+q+1}^{[p/q]} \right) \\ \lambda = -\frac{\gamma_{p+q} e_{p+q}^{[p/q-1]}}{e_{p+q-1}^{[p-1/q-1]}} \tau \\ \eta = -\frac{1}{e_{p+q}^{[p/q-1]}} \left(d_{p+q}^{[p/q-1]} \tau + e_{p+q}^{[p-1/q-1]} \lambda \right) \\ \rho = -\frac{1}{e_{p+q+1}^{[p/q]}} \left(d_{p+q+1}^{[p/q-1]} \tau + e_{p+q+1}^{[p-1/q-1]} \lambda + e_{p+q+1}^{[p/q-1]} \eta \right) \end{cases} \quad (2.9)$$

Então

$$\frac{N^{[p+1/q]}}{D^{[p+1/q]}} = [p+1/q]_f^P$$

é o AFP normalizado tal que $b_q^{[p+1/q]} = 1$.

Note-se que as igualdades (2.9) resultam da resolução do sistema (2.8).

Desta proposição resulta um corolário que permite calcular os coeficientes do numerador, do denominador e da série do erro do aproximante $[p+1/q]_f^P(z)$.

Corolário 2.1 *Nas condições da proposição anterior, os coeficientes dos aproximantes e da série do erro satisfazem as seguintes relações*

$$\begin{cases} a_{p+1}^{[p+1/q]} = \tau g_{p+1}^{[p/q-1]} = \tau \alpha_p a_p^{[p/q-1]} \\ a_p^{[p+1/q]} = \tau g_p^{[p/q-1]} + \eta a_p^{[p/q-1]} + \rho a_p^{[p/q]} \\ a_i^{[p+1/q]} = \tau g_i^{[p/q-1]} + \lambda a_i^{[p-1/q-1]} + \eta a_i^{[p/q-1]} + \rho a_i^{[p/q]}, \ i = 0, \ldots, p-1 \\ b_i^{[p+1/q]} = \tau h_i^{[p/q-1]} + \lambda b_i^{[p-1/q-1]} + \eta b_i^{[p/q-1]} + \rho b_i^{[p/q]}, \ i = 0, \ldots, q-1 \\ e_i^{[p+1/q]} = \tau d_i^{[p/q-1]} + \lambda e_i^{[p-1/q-1]} + \eta e_i^{[p/q-1]} + \rho e_i^{[p/q]}, \ i = p+q+2, \ldots, L_{p+1,q} \end{cases}$$

onde $L_{p+1,q}$ representa uma constante a determinar e os valores g_i, h_i e d_i são calculados a partir dos coeficientes a_i, b_i e e_i correspondentes, utilizando as fórmulas da proposição 2.1.

Para a relação $\begin{matrix} \bullet & \bullet \\ \bullet & * \end{matrix}$ é igualmente possível desenvolver as expressões dos numeradores e dos denominadores com o objectivo de conseguir equações que nos permitam encontrar os coeficientes dos aproximantes em causa.

Para $p, q \geq 1$ tem-se

$$\begin{cases} N^{[p+1/q+1]} = \lambda N^{[p/q-1]} + (\eta + \tau z) N^{[p/q]} + \rho N^{[p+1/q]} \\ D^{[p+1/q+1]} = \lambda D^{[p/q-1]} + (\eta + \tau z) D^{[p/q]} + \rho D^{[p+1/q]} \end{cases} \quad \begin{matrix} \bullet & \bullet \\ \bullet & * \end{matrix}$$

Novamente, consideremos $N^{[p+1/q+1]} = \lambda N^{[p/q-1]} + (\eta + \tau z) N^{[p/q]} + \rho N^{[p+1/q]}$.

$$\begin{aligned} \sum_{i=0}^{p+1} a_i^{[p+1/q+1]} P_i &= \lambda \sum_{i=0}^{p} a_i^{[p/q-1]} P_i + \eta \sum_{i=0}^{p} a_i^{[p/q]} P_i + \tau z \sum_{i=0}^{p} a_i^{[p/q]} P_i + \rho \sum_{i=0}^{p+1} a_i^{[p+1/q]} P_i \\ &= \lambda \sum_{i=0}^{p} a_i^{[p/q-1]} P_i + \eta \sum_{i=0}^{p} a_i^{[p/q]} P_i + \tau \sum_{i=0}^{p+1} g_i^{[p/q]} P_i + \rho \sum_{i=0}^{p+1} a_i^{[p+1/q]} P_i \end{aligned}$$

onde os coeficientes $g_i^{[p/q]}$ são obtidos a partir de (2.5).

Desta igualdade, dado que os coeficientes a_i respeitam as condições de normalidade, decorre que $\tau g_{p+1}^{[p/q]} + \rho a_{p+1}^{[p+1/q]} = a_{p+1}^{[p+1/q+1]} \neq 0$.

Analisemos agora as relações entre os coeficientes dos denominadores $D^{[p+1/q+1]} = \lambda D^{[p/q-1]} + (\eta + \tau z) D^{[p/q]} + \rho D^{[p+1/q]}$.

$$\begin{aligned} \sum_{i=0}^{q+1} b_i^{[p+1/q+1]} P_i &= \lambda \sum_{i=0}^{q-1} b_i^{[p/q-1]} P_i + \eta \sum_{i=0}^{q} b_i^{[p/q]} P_i + \tau z \sum_{i=0}^{q} b_i^{[p/q]} P_i + \rho \sum_{i=0}^{q} b_i^{[p+1/q]} P_i \\ &= \lambda \sum_{i=0}^{q-1} b_i^{[p/q-1]} P_i + \eta \sum_{i=0}^{q} b_i^{[p/q]} P_i + \tau \sum_{i=0}^{q+1} h_i^{[p/q]} P_i + \rho \sum_{i=0}^{q} b_i^{[p+1/q]} P_i \end{aligned}$$

onde os valores $h_i^{[p/q]}$ se obtêm a partir de (2.5).

A igualdade conseguida através da comparação dos coeficientes de maior grau é $b_{q+1}^{[p+1/q+1]} = \tau h_{q+1}^{[p/q]}$. Mais uma vez, devido à normalização, $b_{q+1}^{[p+1/q+1]} = 1$ e, como $h_{q+1}^{[p/q]} = \alpha_q b_q^{[p/q]} = \alpha_q$ por (2.5), tem-se que $\tau \alpha_q = 1$.

Analisemos de igual forma as relações entre as séries dos erros

$$\begin{aligned} \sum_{i \geq p+q+3} e_i^{[p+1/q+1]} P_i &= \lambda \sum_{i \geq p+q} e_i^{[p/q-1]} P_i + \eta \sum_{i \geq p+q+1} e_i^{[p/q]} P_i + \tau z \sum_{i \geq p+q+1} e_i^{[p/q]} P_i + \rho \sum_{i \geq p+q+2} e_i^{[p+1/q]} P_i \\ &= \lambda \sum_{i \geq p+q} e_i^{[p/q-1]} P_i + \eta \sum_{i \geq p+q+1} e_i^{[p/q]} P_i + \tau \sum_{i \geq p+q} d_i^{[p/q]} P_i + \rho \sum_{i \geq p+q+2} e_i^{[p+1/q]} P_i \end{aligned}$$

onde as quantidades $d_i^{[p/q]}$ se encontram a partir de (2.6).

A partir dos coeficientes de grau $p+q$, $p+q+1$ e $p+q+2$ obtemos as seguintes equações:

$$\begin{cases} \lambda e_{p+q}^{[p/q-1]} + \tau d_{p+q}^{[p/q]} = 0 \\ \lambda e_{p+q+1}^{[p/q-1]} + \eta e_{p+q+1}^{[p/q]} + \tau d_{p+q+1}^{[p/q]} = 0 \\ \lambda e_{p+q+2}^{[p/q-1]} + \eta e_{p+q+2}^{[p/q]} + \tau d_{p+q+2}^{[p/q]} + \rho e_{p+q+2}^{[p+1/q]} = 0 \end{cases}.$$

Da proposição 2.1, em particular de (2.6), vem que $d_{p+q}^{[p/q]} = \gamma_{p+q+1} e_{p+q+1}^{[p/q]}$ e então o sistema a resolver é o seguinte:

$$\begin{cases} \tau\alpha_q = 1 \\ \tau\gamma_{p+q+1} e_{p+q+1}^{[p/q]} + \lambda e_{p+q}^{[p/q-1]} = 0 \\ \tau d_{p+q+1}^{[p/q]} + \lambda e_{p+q+1}^{[p/q-1]} \eta e_{p+q+1}^{[p/q]} = 0 \\ \tau d_{p+q+2}^{[p/q]} + \lambda e_{p+q+2}^{[p/q-1]} + \eta e_{p+q+2}^{[p/q]} + \rho e_{p+q+2}^{[p+1/q]} = 0 \end{cases}. \tag{2.10}$$

Note-se que o sistema (2.10) é triangular inferior e regular uma vez que, por hipótese de normalidade, os elementos da diagonal são não nulos.

Como resultado desta análise obtemos a seguinte proposição.

Proposição 2.5 *Sejam $p \geq 0$ e $q \geq 1$ tais que os aproximantes $[p/q-1]_f^P$, $[p/q]_f^P$ e $[p+1/q]_f^P$ existem e satisfazem as condições de normalidade. Defina-se*

$$\begin{cases} N^{[p+1/q+1]} = \lambda N^{[p/q-1]} + (\eta + \tau z) N^{[p/q]} + \rho N^{[p+1/q]} \\ D^{[p+1/q+1]} = \lambda D^{[p/q-1]} + (\eta + \tau z) D^{[p/q]} + \rho D^{[p+1/q]} \end{cases} \quad \begin{matrix} \bullet & \bullet \\ \bullet & * \end{matrix}$$

com

$$\begin{cases} \tau = \frac{1}{\alpha_q} \\ \lambda = -\frac{\gamma_{p+q+1} e_{p+q+1}^{[p/q]}}{e_{p+q}^{[p/q-1]}} \tau \\ \eta = -\frac{1}{e_{p+q+1}^{[p/q]}} \left(d_{p+q+1}^{[p/q]} \tau + e_{p+q+1}^{[p/q-1]} \lambda \right) \\ \rho = -\frac{1}{e_{p+q+2}^{[p+1/q]}} \left(d_{p+q+2}^{[p/q]} \tau + e_{p+q+2}^{[p/q-1]} \lambda + e_{p+q+2}^{[p/q]} \eta \right) \end{cases}. \tag{2.11}$$

Então

$$\frac{N^{[p+1/q]}}{D^{[p+1/q]}} = [p+1/q]_f^P$$

é o AFP normalizado tal que $b_q^{[p+1/q]} = 1$.

O sistema (2.11) decorre da resolução por substituição descendente do sistema (2.10).

Um resultado imediato desta proposição figura no seguinte corolário que permite calcular os coeficientes do numerador, do denominador e da série do erro do aproximante $[p+1/q+1]_f^P$.

Corolário 2.2 *Nas condições da proposição anterior, os coeficientes dos aproximantes e do erro satisfazem a relação*

$$\begin{cases} a_{p+1}^{[p+1/q]} = \tau g_{p+1}^{[p/q]} + \rho a_{p+1}^{[p+1/q]} = \alpha_p \tau a_p^{[p/q]} + \rho a_{p+1}^{[p+1/q]} \\ a_i^{[p+1/q]} = \lambda a_i^{[p/q-1]} + \eta a_i^{[p/q]} + \tau g_i^{[p/q]+\rho a_i^{[p+1/q]}}, \quad i = 0, \ldots, p \\ b_q^{[p+1/q+1]} = \eta + \tau h_q^{[p/q]} + \rho \\ b_i^{[p+1/q+1]} = \lambda b_i^{[p/q-1]} + \eta b_i^{[p/q]} + \tau h_i^{[p/q]} + \rho b_i^{[p+1/q]}, \quad i = 0, \ldots, q-1 \\ e_i^{[p+1/q]} = \lambda e_i^{[p/q-1]} + \eta e_i^{[p/q]} + \tau d_i^{[p/q]} + \rho e_i^{[p+1/q]}, \quad i = p+q+3, \ldots, L_{p+1,q} \end{cases}$$

onde $L_{p+1,q}$ representa uma constante a determinar e os valores g_i, h_i e d_i são calculados a partir dos coeficientes a_i, b_i e d_i correspondentes, utilizando as fórmulas da proposição 2.1.

2.4 Descrição do novo algoritmo

Os resultados das proposições anteriores podem utilizar-se intercaladamente permitindo calcular uma sucessão de aproximantes dispostos em duas diagonais descendentes adjacentes. Sendo $\bullet_0 = [p/0]_f^P$, $\bullet_1 = [p+1/0]_f^P$ e $\bullet_2 = [p+1/1]_f^P$, com as fórmulas anteriores, podemos calcular a sucessão de aproximantes $*_3 = [p+2/1]_f^P$, $*_4 = [p+2/2]_f^P$, $*_5 = [p+3/2]_f^P$, $*_6 = [p+3/3]_f^P, \ldots, *_{2i-1} = [p+i/i-1]_f^P, *_{2i} = [p+i/i]_f^P, \ldots$, com a seguinte disposição na tabela

$$\begin{array}{ccccccc} \bullet_0 & & & & & & \\ \bullet_1 & \bullet_2 & & & & & \\ & *_3 & *_4 & & & & \\ & & \ddots & \ddots & & & \\ & & & & *_{2J-1} & *_{2J} & \end{array}$$

Este procedimento encontra-se esquematizado no seguinte algoritmo.

Algoritmo
$$\begin{array}{ccc} & \bullet & \\ \bullet & & \bullet \\ & * & * \end{array}$$

Dados: $p,\ J \in \mathbb{N},\ \{f_i\}_{i=0}^{p+3J}$

Calcula: $\left\{[p+q/q-1]_f^P,\ [p+q/q]_f^P\right\}_{q=2}^{J}$

Parâmetros: $\{\alpha_i,\ \beta_i,\ \gamma_i\}_{i=0}^{p+3J}$

Valores iniciais:
$$\begin{aligned} & a_{q,-1} = b_{q,-1} = e_{q,p+q} = 0, \quad q = 0, \ldots, 2J \\ & a_{2q,p+q+1} = b_{2q,q+1} = 0,\ b_{2q,q} = 1, \quad q = 0, \ldots, J \\ & a_{2q-1,p+q+1} = b_{2q-1,q} = 0,\ b_{2q-1,q-1} = 1, \quad q = 1, \ldots, J \\ & e_{q,i} = 0, \quad i = 0, \ldots, p+q, \quad q = 0, \ldots, 2J \\ & a_{0,i} = f_i, \quad i = 0, \ldots, p \\ & a_{1,i} = f_i, \quad i = 0, \ldots, p+1 \\ & e_{0,i} = f_i, \quad i = p+1, \ldots, p+3J \\ & e_{1,i} = f_i, \quad i = p+2, \ldots, p+3J \end{aligned}$$

Calcular:

I $[p+1/1]_f^P$

$d_{0,i} = \alpha_{i-1} f_{i-1} + \beta_i f_i + \gamma_{i+1} f_{i+1}, \quad i = p+2, \ldots, p+3J-1$

$b_{2,0} = \frac{\beta_0}{\alpha_0} - \frac{1}{\alpha_0 f_{p+2}} d_{0,p+2}$

$a_{2,0} = b_{2,0} f_0 + \frac{\gamma_1}{\alpha_0} f_1$

$a_{2,i} = (b_{2,0} - \frac{\beta_0}{\alpha_0}) f_i + \frac{1}{\alpha_0}(\alpha_{i-1} f_{i-1} + \beta_i f_i + \gamma_{i+1} f_{i+1}), \quad i = 1, \ldots, p+1$

$e_{2,i} = (b_{2,0} - \frac{\beta_0}{\alpha_0}) f_i + \frac{1}{\alpha_0} d_{0,i}, \quad i = p+3, \ldots, p+3J-1$

Se $e_{2,p+3} = 0$, **então** pára.

Para $q = 2, \ldots, J$

II $[p+q/q-1]_f^P$

$d_{2q-3,i} = \alpha_{i-1} e_{2q-3,i-1} + \beta_i e_{2q-3,i} + \gamma_{i+1} e_{2q-3,i+1}, \quad i = p+2q-2, \ldots, p+3J-q$

$g_{2q-3,i} = \alpha_{i-1} a_{2q-3,i-1} + \beta_i a_{2q-3,i} + \gamma_{i+1} a_{2q-3,i+1}, \quad i = 0, \ldots, p+q-1$

$h_{2q-3,i} = \alpha_{i-1} b_{2q-3,i-1} + \beta_i b_{2q-3,i} + \gamma_{i+1} b_{2q-3,i+1}, \quad i = 0, \ldots, q-2$

$\Delta = \alpha_{q-2} e_{2q-4,p+2q-3} e_{2q-3,p+2q-2} e_{2q-2,p+2q-1} -$
$-\gamma_{p+2q-2} e_{2q-3,p+2q-2} (e_{2q-4,p+2q-2} e_{2q-3,p+2q-1} - e_{2q-3,p+2q-2} e_{2q-4,p+2q-1}) +$
$+e_{2q-4,p+2q-3} (d_{2q-3,p+2q-2} e_{2q-3,p+2q-1} - e_{2q-3,p+2q-2} d_{2q-3,p+2q-1})$

Se $\Delta = 0$, **então** pára.

$\tau_1 = e_{2q-4,p+2q-3} e_{2q-3,p+2q-2} e_{2q-2,p+2q-1} / \Delta$

$\lambda_1 = -\gamma_{p+2q-2} e_{2q-3,p+2q-2} \tau_1 / e_{2q-4,p+2q-3}$

$\eta_1 = -(d_{2q-3,p+2q-2} \tau_1 + e_{2q-4,p+2q-2} \lambda_1) / e_{2q-3,p+2q-2}$

$\rho_1 = -(d_{2q-3,p+2q-1} \tau_1 + e_{2q-4,p+2q-1} \lambda_1 + e_{2q-3,p+2q-1} \eta_1) / e_{2q-2,p+2q-1}$

$a_{2q-1,p+q} = \alpha_{p+q-1} a_{2q-3,p+q-1} \tau_1$

$a_{2q-1,p+q-1} = g_{2q-3,p+q-1} \tau_1 + a_{2q-3,p+q-1} \eta_1 + a_{2q-2,p+q-1} \rho_1$

$a_{2q-1,i} = g_{2q-3,i} \tau_1 + a_{2q-4,i} \lambda_1 + a_{2q-3,i} \eta_1 + a_{2q-2,i} \rho_1, \quad i = 0, \ldots, p+q-2$

$b_{2q-1,i} = h_{2q-3,i} \tau_1 + b_{2q-4,i} \lambda_1 + b_{2q-3,i} \eta_1 + b_{2q-2,i} \rho_1, \quad i = 0, \ldots, q-2$

$e_{2q-1,i} = d_{2q-3,i} \tau_1 + e_{2q-4,i} \lambda_1 + e_{2q-3,i} \eta_1 + e_{2q-2,i} \rho_1, \quad i = p+2q, \ldots, p+3J-q$

Se $e_{2q-1,p+2q} = 0$, **então** pára.

III $[p+q/q]_f^P$

$d_{2q-2,i} = \alpha_{i-1} e_{2q-2,i-1} + \beta_i e_{2q-2,i} + \gamma_{i+1} e_{2q-2,i+1}, \quad i = p+2q-1, \ldots, p+3J-q$

$g_{2q-2,i} = \alpha_{i-1} a_{2q-2,i-1} + \beta_i a_{2q-2,i} + \gamma_{i+1} a_{2q-2,i+1}, \quad i = 0, \ldots, p+q-1$

$h_{2q-2,i} = \alpha_{i-1} b_{2q-2,i-1} + \beta_i b_{2q-2,i} + \gamma_{i+1} b_{2q-2,i+1}, \quad i = 0, \ldots, q-1$

$\tau_2 = 1/\alpha_{q-1}$

$\lambda_2 = -\gamma_{p+2q-1} e_{2q-2,p+2q-1} \tau_2 / e_{2q-3,p+2q-2}$

$\eta_2 = -(d_{2q-2,p+2q-1} \tau_2 + e_{2q-3,p+2q-1} \lambda_2) / e_{2q-2,p+2q-1}$

$\rho_2 = -(d_{2q-2,p+2q} \tau_2 + e_{2q-3,p+2q} \lambda_2 + e_{2q-2,p+2q} \eta_2) / e_{2q-1,p+2q}$

$a_{2q,p+q} = \alpha_{p+q-1} a_{2q-2,p+q-1} \tau_2 + a_{2q-1,p+q} \rho_2$

$a_{2q,i} = g_{2q-2,i} \tau_2 + a_{2q-3,i} \lambda_2 + a_{2q-2,i} \eta_2 + a_{2q-1,i} \rho_2, \quad i = 0, \ldots, p+q-1$

$b_{2q,q-1} = h_{2q-2,q-1} \tau_2 + \eta_2 + \rho_2$

$b_{2q,i} = h_{2q-2,i} \tau_2 + b_{2q-3,i} \lambda_2 + b_{2q-2,i} \eta_2 + b_{2q-1,i} \rho_2, \quad i = 0, q-2$

$e_{2q,i} = d_{2q-2,i} \tau_2 + e_{2q-3,i} \lambda_2 + e_{2q-2,i} \eta_2 + e_{2q-1,i} \rho_2, \quad i = p+2q+1, \ldots, p+3J-q$

Se $e_{2q,p+2q+1} = 0$, **então** pára.

Resultados:

Para $q = 1, \ldots, J$

$$
\begin{array}{ll}
a_{2q-1,i}, & i = 0, \ldots, p+q \\
b_{2q-1,i}, & i = 0, \ldots, q+2 \\
a_{2q,i}, & i = 0, \ldots, p+q \\
b_{2q,i}, & i = 0, \ldots, q-1 \\
e_{2q-1,i}, & i = p+2q, \ldots, p+3J-q \\
e_{2q,i}, & i = p+2q+1, \ldots, p+3J-q
\end{array}
$$

Obtidos os valores dos coeficientes efectua-se o cálculo dos numeradores, denominadores e uma aproximação da série dos erros dos AFP:

$$
\begin{array}{l}
N^{[p+q/q-1]} = \sum_{i=0}^{p+q} a_{2q-1,i} P_i \\
D^{[p+q/q-1]} = \sum_{i=0}^{q-2} b_{2q-1,i} P_i + P_{q-1} \\
N^{[p+q/q]} = \sum_{i=0}^{p+q} a_{2q,i} P_i \\
D^{[p+q/q]} = \sum_{i=0}^{q-1} b_{2q,i} P_i + P_q \\
D^{[p+q/q-1]} f - N^{[p+q/q-1]} = \sum_{i=p+2q}^{p+3J-q} e_{2q-1,i} P_i + \mathcal{O}(P_{p+3J-q+1}) \\
D^{[p+q/q]} f - N^{[p+q/q]} = \sum_{i=p+2q+1}^{p+3J-q} e_{2q,i} P_i + \mathcal{O}(P_{p+3J-q+1})
\end{array}
$$

O número de coeficientes f_i necessários inicialmente, bem como o número suficiente de termos e_i a calcular em cada iteração, foram objecto de estudo em [18], estudo esse que se revelou fundamental para a elaboração do algoritmo apresentado.

Para avaliar a sua fiabilidade, o algoritmo descrito foi programado e testado com funções de que se conhecem simultaneamente um desenvolvimento ortogonal com fórmulas exactas para os coeficientes e uma fórmula analítica fechada que permita avaliar a função, em qualquer abcissa do intervalo de aproximação, com a mesma precisão de cálculo dos aproximantes no programa. Na secção seguinte, apresentam-se vários exemplos de aplicação do algoritmo. Os resultados são ilustrados com os erros relativos na aproximação da função utilizando os aproximantes avaliados com os coeficientes calculados pelas fórmulas apresentadas no algoritmo e com os algarismos significativos correctamente calculados presentes nesses coeficientes.

Capítulo 3

Estabilidade do cálculo dos Aproximantes de Frobenius-Padé

Este capítulo trata da estabilidade do cálculo recursivo de sucessões de AFP.

Iniciamos a análise da estabilidade deste problema efectuando um estudo pormenorizado dos programas em Fortran90 existentes na literatura [18] no sentido de avaliar e localizar as instabilidades presentes nos cálculos [19]. Este trabalho foi realizado com o auxílio do CADNA (ver apêndice C).

Com o objectivo de progredir no sentido do aumento da estabilidade do cálculo dos AFP, modificamos o novo algoritmo substituindo a dado ponto quatro fórmulas algébricas correspondentes ao cálculo de quatro parâmetros τ, λ, η e ρ, intervenientes nas relações de recorrência, pela resolução de um sistema 4×4 usando uma decomposição LU com pivotagem parcial. Os testes deste algoritmo modificado revelam um suave melhoramento no cálculo dos quatro parâmetros citados que se traduz de forma considerável ao cálculo dos AFP, demonstrando o efeito da pivotagem parcial.

3.1 Localização da instabilidade numérica

O principal objectivo desta secção é analisar a propagação dos erros de arredondamento efectuados pela aritmética de vírgula flutuante e a consequente perda de precisão nos valores dos coeficientes dos AFP, calculados com o algoritmo recursivo apresentado em [18]. Tal algoritmo foi programado em Fortran90, utilizando aritmética de precisão dupla, e testado com uma função da qual se conhece um desenvolvimento ortogonal com coeficientes exactos.

Consideremos o seguinte exemplo.

Exemplo 3.1 *Os polinómios de Legendre $P_i(z)$, normalizados por $P_i(1) = 1$, $i \geq 0$, satisfazem a relação de recorrência*

$$P_{i+1}(z) = \frac{z-\beta_i}{\alpha_i}P_i(z) - \frac{\gamma_i}{\alpha_i}P_{i-1}(z), \quad \gamma_i \neq 0,\ i \geq 1$$

$$P_0(z) = 1, \quad P_1(z) = \frac{z-\beta_0}{\alpha_0},$$

com

$$\begin{cases} \alpha_i = \frac{i+1}{2i+1}, & \beta_i = 0, \quad \gamma_i = \frac{i}{2i+1}, \quad i \geq 1 \\ \alpha_0 = 1, & \beta_0 = 0 \end{cases} . \tag{3.1}$$

Partindo da função geradora dos polinómios de Legendre

$$f(z) = \frac{1}{\sqrt{1-2az+a^2}} = \sum_{n\geq 0} a^n P_n(z), \quad -1 < z < 1$$

temos que

$$\{\alpha_i, \beta_i, \gamma_i\}_{i\geq 0}$$

são os parâmetros e

$$\{f_i = a^i\}_{i\geq 0}$$

são os dados do programa.

Este exemplo aparece também tratado em [17, 18]. Tomando o mesmo valor do parâmetro $a = 0.8$ aí considerado

$$f(z) = \frac{1}{\sqrt{1.64 - 1.6z}}, \ -1 < z < 1, \quad \{f_i = 0.8^i\}_{i\leq 0}.$$

O algoritmo em questão foi reprogramado em Fortran90, utilizando aritmética de precisão simples e dupla. O programa citado foi alterado no sentido de implementar a biblioteca CADNA (ver apêndice C.3), em precisão simples e dupla, com vista ao tratamento da instabilidade numérica detectada.

Os vários programas produzidos foram testados e para a função tratada calcularam-se diversas sucessões de aproximantes. Em ambos os casos, em precisão simples calcularam-se os primeiros 20 aproximantes, i.e., até ao elemento $[p + 10/10]_f^P(z)$ e em precisão dupla calcularam-se os primeiros 40 aproximantes, ou seja, até ao elemento $[p+20/20]_f^P(z)$. Como resultados obtivemos os coeficientes a_i do numerador, b_i do denominador e e_i da série do erro e respectivos algarismos significativos correctamente calculados dos aproximantes em causa.

Resultados numéricos

Para as quantidades tidas como exemplo considere-se a seguinte notação: ω^{sp} e ω^{dp} representam respectivamente resultados em precisão simples e dupla obtidos sem o CADNA e ω^* e ω^{**} referem-se a resultados em precisão simples e dupla calculados com o CADNA.

Executado o programa original (sem implementação do CADNA) de cálculo recursivo dos coeficientes de Frobenius-Padé, com precisão simples e dupla, os resultados merecem desde já algumas considerações. Os algarismos comuns, contados da esquerda para a direita, dos resultados obtidos com as duas precisões devem, em princípio, estar correctamente calculados. Ilustremos com um exemplo:

$$[2/2]_f^P: \quad a_0^{sp} = \underline{0.27794814}E+01 \quad a_0^{dp} = \underline{0.2779483}046979128E+01.$$

Embora a propagação dos erros de cálculo exista num e noutro caso, os resultados em precisão dupla garantem um maior número de algarismos significativos o que nos leva a acreditar que os primeiros algarismos sublinhados estejam correctamente calculados.

Estes algarismos comuns deveriam aparecer nos resultados correspondentes obtidos com o CADNA, porém constatamos algumas diferenças notórias que resultam da aritmética aleatória.

Uma vantagem inerente à utilização do CADNA é o facto dos resultados serem exibidos apresentando apenas os algarismos significativos correctamente calculados. Verificámos que o CADNA fornece uma resposta prudente, exibindo sempre um número menor de algarismos significativos do que os comuns às precisões simples e dupla. Na sequência do exemplo anterior, obtivemos

$$[2/2]_f^P : \quad a_0^{sp} = \underline{\underline{0.27794}}814E+01 \quad a_0^* = \underline{0.27794}E+01.$$

Comparando os resultados, em precisão simples, obtidos com e sem CADNA constatamos, que os obtidos com este último exibem um número decrescente de algarismos significativos, tal como seria de esperar dado o efeito acumulativo dos erros de cálculo. Com precisão simples, a partir do aproximante $[5/5]_f^P$, o CADNA já não fornece resultados, i.e., não existem algarismos significativos correctamente calculados, o que significa que atingimos um resultado sem precisão, @.0.

$$[5/5]_f^P : \quad a_0^{sp} = -.34182022E+03 \quad a_0^* = @.0$$

Durante a execução do programa implementado com o CADNA é criado o ficheiro cadna_stability_f90.lst que contém a lista de instabilidades numeradas por ordem de ocorrência (ver apêndice C.4). No ficheiro das instabilidades constatamos a existência de instabilidades no aproximante $[5/5]_f^P$.

No caso particular com $p, q = 0, \ldots, 10$, este ficheiro apresenta 704 instabilidades mas apenas de dois tipos distintos:

- unstable multiplication: ambos os operandos são não significativos
- unstable division: o denominador é não significativo

À partida sabíamos que os valores e_i, tratando-se dos coeficientes da série do erro e conhecidos previamente através da execução algoritmo original de [18], seriam bastante pequenos e portanto as fórmulas que exibem divisões ou multiplicações por estas quantidades são, provavelmente, uma fonte de instabilidade.

Recorrendo a um *debugger*, o CADNA, para cada uma das instabilidades exibidas em cadna_stability_f90.lst, indica a linha de código correspondente a cada instabilidade produzida. Estes resultados são guardados num ficheiro idb.out. Consideremos a seguinte informação contida neste ficheiro:

	cadna_stability_f90.lst	idb.out
1	UNSTABLE DIVISION.	146
...		...
61	UNSTABLE MULTIPLICATION.	146
62	UNSTABLE DIVISION.	146
...		...

Podemos constatar que a linha 146 do programa é a principal linha a causar instabilidade, tendo atingido os dois tipos de instabilidade.

$$146 \quad \rho_1 = \frac{-1}{e_{2q-2,p+2q-1}}(e_{2q-4,p+2q-1}\lambda_1 + e_{2q-3,p+2q-1}\eta_1 + d_{2q-3,p+2q-1})$$

As nossas expectativas foram confirmadas. Esta expressão, considerada instável pelo CADNA, apresenta uma multiplicação e uma divisão por valores e_i e d_i e portanto enquadra-se no perfil que tínhamos descrito. Lembremos que os coeficientes d_i são obtidos por multiplicação por um monómio e são também quantidades pequenas (ver proposição 2.1). Esta instrução, por estar inserida num ciclo, sempre que esse ciclo é percorrido uma vez mais é atingida a instabilidade, motivo pelo qual esta linha é exibida várias vezes no ficheiro idb.out.

Outras fórmulas que devem manter-se sob vigilância aparecem nas linhas 114 e 115

	cadna_stability_f90.lst	idb.out
...		...
9	UNSTABLE DIVISION.	114
10	UNSTABLE DIVISION.	115
...		...

devido à existência de divisões por Δ.

$$114 \quad a_{2q-1,p+q} = \alpha_{p+q-1}a_{2q-3,p+q-1}/\Delta$$
$$115 \quad a_{2q-1,p+q-1} = (\eta_1 a_{2q-3,p+q-1} + g_{2q-3,p+q-1} + \rho_1 a_{2q-2,p+q-1})/\Delta$$

Se analisarmos o algoritmo apresentado em [18] verificamos que $\Delta = \alpha_{q-2} + \rho_1$ depende do valor de α_{q-2}, que é um dos parâmetros que intervêm na relação de recorrência dos polinómios ortogonais (3.1) e é conhecido exactamente e não é fonte de instabilidade, e depende também de ρ_1 que, como vimos, é instável.

Na tabela 3.1, na posição (p, q) é evidenciado o número de algarismos significativos correctamente calculados do coeficiente com menor precisão do numerador e do denominador do aproximante de Frobenius-Padé $[p/q]_f^P$, obtido com o CADNA usando precisão simples.

Observamos que esse número decresce rapidamente à medida que se progride ao longo das diagonais. A partir da coluna $q = 5$, os erros podem considerar-se até muito grandes dada a falta de algarismos significativos correctos nos coeficientes. Este resultado contrasta com a razoável aproximação obtida com estes aproximantes exibida na tabela 3.2.

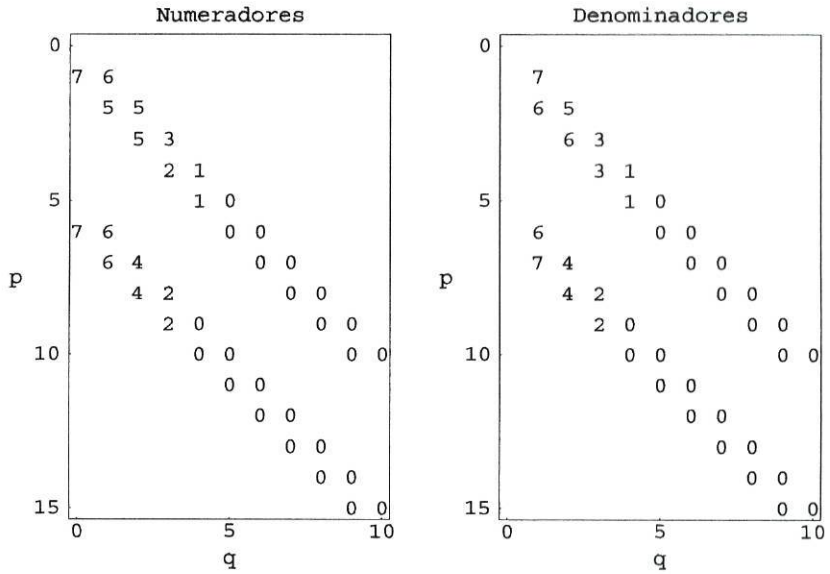


Tabela 3.1: Algarismos significativos correctamente calculados do coeficiente com menor precisão do numerador e do denominador, obtidos com o algoritmo de [18] programado em Fortran90 com *precisão simples* e utilizando a biblioteca CADNA

A partir dos coeficientes a_i e b_i obtidos pelo algoritmo programado em Fortran90, os AFP são construídos no Mathematica, i.e., calcula-se

$$[p/q]_f^P = \frac{\sum_{i=0}^{p} a_i P_i(z)}{\sum_{i=0}^{q} b_i P_i(z)}$$

utilizando os comandos que fornecem os polinómios ortogonais e se encontram disponíveis no Mathematica. Em seguida, os aproximantes são comparados com os valores exactos da soma da série $f(z)$.

Devido à existência de uma singularidade da função no ponto $z = 1.64/1.6 = 1.025$, em [18], os aproximantes são testados em dois pontos do intervalo de aproximação: um perto do extremo do intervalo e da singularidade, $z = 0.9$; o outro pertencente a uma região *bem comportada*, $z = -0.5$. A tabela 3.2 mostra, para cada posição (p, q), o número de algarismos significativos exactos na aproximação $[p/q]_f^P(z)$,i.e.,

$$\left[-\log_{10} \left(\left| \frac{f(z) - [m/n]_f^P(z)}{f(z)} \right| \right) \right],$$

para estes dois valores de z.

Ainda que todas as considerações tomadas até este parágrafo tenham sido baseadas em resultados obtidos em precisão simples, algumas delas podem ser igualmente depreendidas para precisão dupla.

Em dupla precisão, com o CADNA, as instabilidades foram detectadas mais tarde do que em precisão simples nas mesmas linhas do programa que as obtidas com esta última. Uma

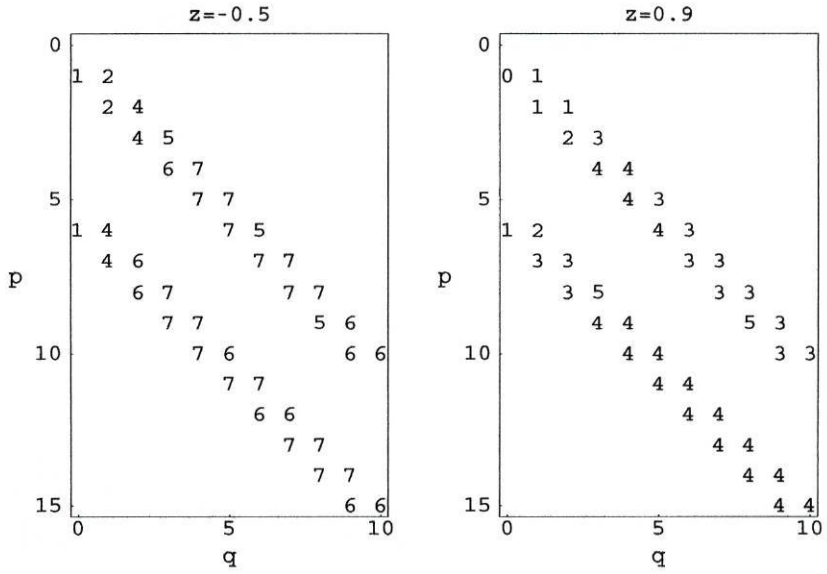


Tabela 3.2: $\left[-\log_{10}\left(\left|\frac{f(z)-[p/q]_f^P(z)}{f(z)}\right|\right)\right]$ calculado no Mathematica, com os $[p/q]_f^P(z)$ construídos a partir dos resultados do algoritmo de [18] programado em Fortran90 com o CADNA utilizando *precisão simples*, para $f(z) = \frac{1}{\sqrt{1.64-1.6z}}$

vantagem inerente à utilização da precisão dupla é o facto de podermos calcular mais aproximantes, embora o número de algarismos significativos correctamente calculados diminua substancialmente, como se vê na tabela 3.3.

Um pormenor curioso, que não passa despercebido, é o fenómeno que ocorre nos coeficientes dos aproximantes $[p/q]_f^P$ com $p, q \geq 6$.

$$[6/6]_f^P : \quad a_0^{dp} = 0.2793998362040806E + 04 \quad a_0^{**} = 0.1903832E + 004$$

A partir do aproximante $[6/6]_f^P$, os coeficientes ω^{dp} e ω^{**} correspondentes deixam de ter qualquer algarismo em comum. Este evento tende a agravar-se, já que a partir do aproximante $[7/7]_f^P$, os coeficientes diferem no sinal e até mesmo na ordem de grandeza, com efeito

$$[7/7]_f^P : \quad a_0^{dp} = -.1973501682127148E + 05 \quad a_0^{**} = 0.6529048E + 003.$$

Daqui concluímos que os resultados obtidos em precisão dupla sem o CADNA, no cálculo dos coeficientes dos aproximantes, não têm qualquer significado a partir do aproximante $[6/6]_f^P$, daí a utilidade do CADNA.

Por último resta-nos apenas verificar a qualidade, em termos do erro relativo, dos aproximantes construídos com os coeficientes a_i e b_i calculados em dupla precisão. Na

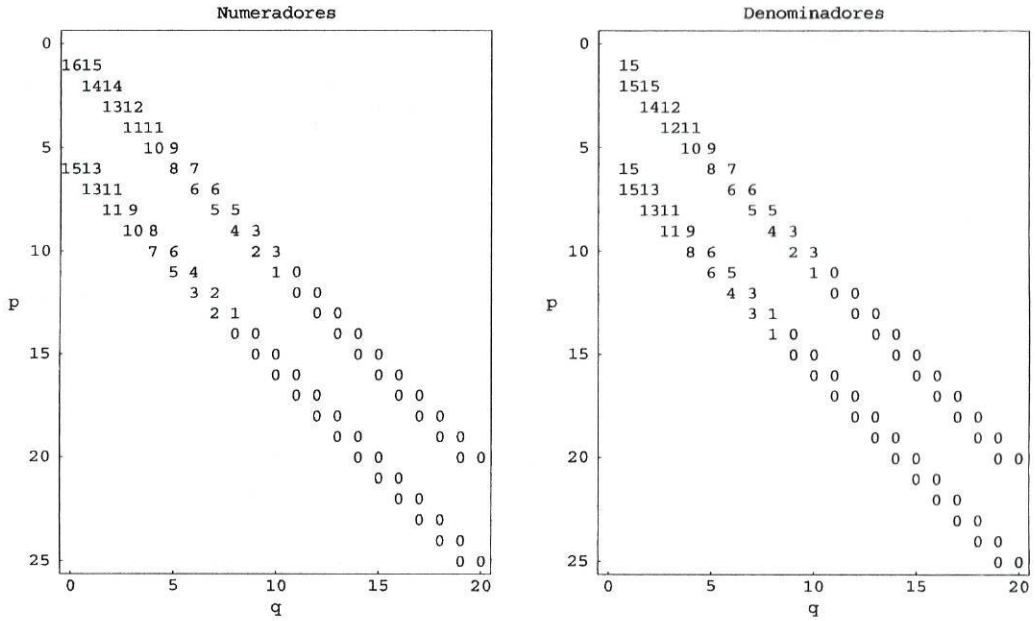


Tabela 3.3: Algarismos significativos correctamente calculados do coeficiente com menor precisão do numerador e do denominador, obtidos com o algoritmo de [18] programado em Fortran90 com *precisão dupla* e utilizando a biblioteca CADNA

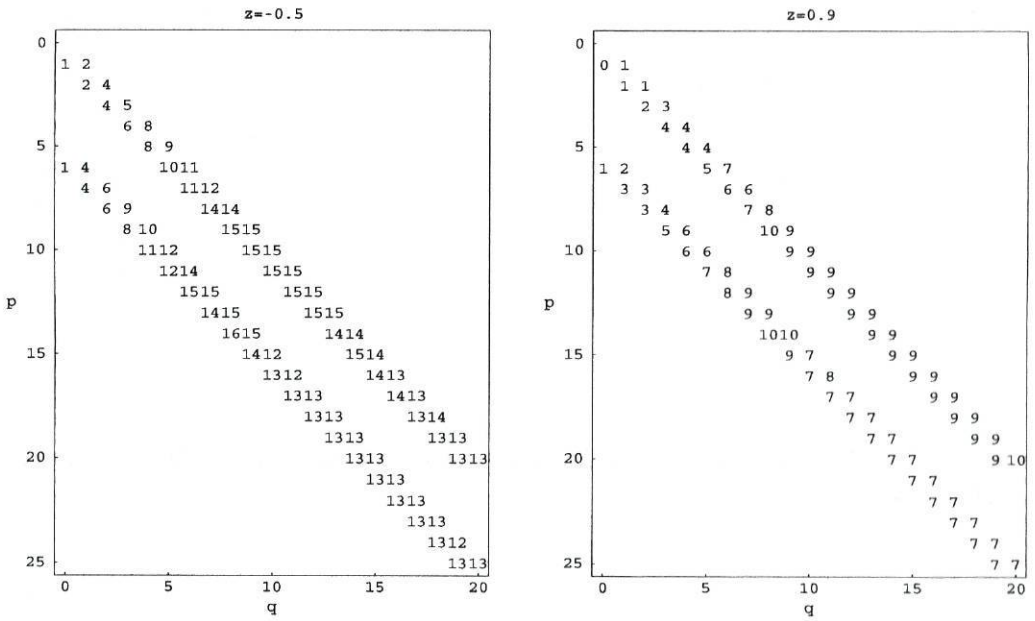


Tabela 3.4: $\left[-\log_{10}\left(\left|\frac{f(z)-[p/q]_f^P(z)}{f(z)}\right|\right)\right]$ calculado no Mathematica, com os $[p/q]_f^P(z)$ construídos a partir dos resultados do algoritmo de [18] programado em Fortran90 com o CADNA utilizando *precisão dupla*, para $f(z) = \frac{1}{\sqrt{1.64-1.6z}}$

tabela 3.4 figura, para cada posição (p, q), o número de algarismos significativos exactos na aproximação $[p/q]_f^P(z)$ para os pontos $z = -0.5$ e $z = 0.9$.

Tal como em precisão simples, os resultados sugerem que o comportamento dos aproximantes, embora razoável, é perturbado pela singularidade da função.

Conclusões

A utilização do CADNA é bastante positiva, dada a rapidez de execução que o assiste, a clara identificação das instabilidades numéricas e das linhas de programa correspondentes e, em especial, a obtenção de resultados credíveis, uma vez que só são exibidos os algarismos correctamente calculados.

Como se pode ver no apêndice (C), a implementação do CADNA no programa em Fortran90 exige seis passos suplementares que corresponde essencialmente à declaração da biblioteca, declaração das variáveis estocásticas, inicialização da aritmética estocástica e alteração das instruções de escrita de forma a apresentar os resultados estocásticos com o respectivo número de algarismos significativos.

No problema em estudo, o CADNA permitiu localizar claramente, com a utilização de um *debugger*, as linhas do programa que produziam instabilidade numérica. A respectiva execução não se torna demasiado pesada em problemas de média dimensão, atendendo a que os tempos de execução vêm multiplicados sensivelmente por cinco. Este aumento de tempo é desprezável quando comparado com o tempo necessário para efectuar cálculo formal no Mathematica, como é realizado em [18]. No problema em causa, obtivemos sempre os resultados de forma rápida, sendo difícil de constatar a diferença de tempo de execução com e sem a implementação do CADNA.

O algoritmo mostra-se bastante sensível no cálculo de uma sucessão de coeficientes, fornecendo resultados com precisão decrescente. Verifica-se que uma das operações mais instáveis, em termos de propagação de erros, consiste no cálculo do parâmetro ρ. Este facto é justificável uma vez que as instruções correspondentes envolvem quocientes em que os coeficientes dos erros e_i figuram em denominador. Estas quantidades são pequenas em valor absoluto, pelo menos a partir de certa ordem. Assistiu-se a uma perda de precisão progressiva no cálculo dos aproximantes.

3.2 Resultados obtidos com a implementação directa do novo algoritmo

O algoritmo descrito na secção anterior e o algoritmo apresentado em [18] estabelecem relações a quatro termos de progressão ao longo de duas diagonais descendentes adjacentes da TFP.

Com o objectivo de analisar a propagação dos erros de arredondamento efectuados pela aritmética de vírgula flutuante e a consequente perda progressiva de precisão nos valores dos coeficientes dos AFP, os algoritmos foram programados em Fortran90 utilizando aritmética de precisão simples e dupla. Os programas citados foram alterados no sentido de implementar

a biblioteca CADNA (ver apêndice C) em ambas as precisões, com vista ao tratamento da instabilidade numérica detectada. O código fonte do programa referente ao algoritmo apresentado na secção 2.4, adaptado para a utilização do CADNA, encontra-se no apêndice (E).

Os vários programas produzidos foram testados com funções das quais se conhecem os desenvolvimentos ortogonais com coeficientes exactos. Para cada uma das funções tratadas, calcularam-se as sucessões de aproximantes correspondentes aos valores de teste $p = 0$ e $p = 5$. Desta forma, usando precisão simples, construíram-se os primeiro 20 aproximantes da sucessão, i.e., até ao elemento $[p + 10/10]_f^P(z)$ e em precisão dupla construíram-se os primeiro 40 aproximantes, ou seja, até ao elemento $[p + 20/20]_f^P(z)$.

Com o objectivo de ilustrar os resultados obtidos, os programas foram testados com pelo menos um representante de cada família de polinómios ortogonais clássicos.

Exemplo 2.1 *Consideremos mais uma vez a família dos polinómios de Legendre $P_i(z)$ e a série*

$$f(z) = \frac{1}{\sqrt{1 - 2az + a^2}} = \sum_{n \geq 0} a^n P_n(z), \quad -1 < z < 1,$$

com $a = 0.8$.

A análise e interpretação dos resultados foi estruturada em três fases. A primeira fase abrangeu o estudo sobre a estabilidade do cálculo dos parâmetros τ, λ, η e ρ que intervêm nas relações de recorrência e respectiva qualidade em termos do número de algarismos significativos correctamente calculados. Em segundo lugar, calculados esses parâmetros, analisámos a qualidade dos coeficientes a_i do numerador e b_i denominador produzidos para os aproximantes de dois conjuntos de diagonais descendentes adjacentes. Por último, realizámos uma análise qualitativa dos aproximantes construídos, em dois pontos estratégicos, utilizando os coeficientes em causa. Nos dois primeiros casos é exibido o número de algarismos significativos correctamente calculados, fornecido pelo CADNA. No último caso, o aproximante é comparado, no Mathematica, com a soma da série.

Resultados numéricos

Da análise das relações de recorrência das proposições 2.4 e 2.5, dos corolários 2.1 e 2.2 que as procedem e do novo algoritmo da secção 2.4 concluímos que os parâmetros τ, λ, η e ρ são determinantes no cálculo das componentes dos AFP. No entanto, como vimos na secção 3.1, verifica-se que as operações mais sensíveis, em termos de propagação de erros, consistem precisamente no cálculo dessa quantidades. Este facto justifica-se porque expressões de τ, λ, η e ρ envolvem quocientes em que os coeficientes dos erros e_i, geralmente quantidades pequenas em valor absoluto, pelo menos a partir de certa ordem, figuram em denominador.

Na tabela 3.5 estão representados, na posição do aproximante $[p/q]_f^P$ da TFP, os algarismos significativos dos parâmetros τ, λ, η e ρ, obtidos com a biblioteca CADNA, constando na primeira coluna os resultados obtidos com o novo algoritmo e na segunda os resultados obtidos com o algoritmo descrito em [18].

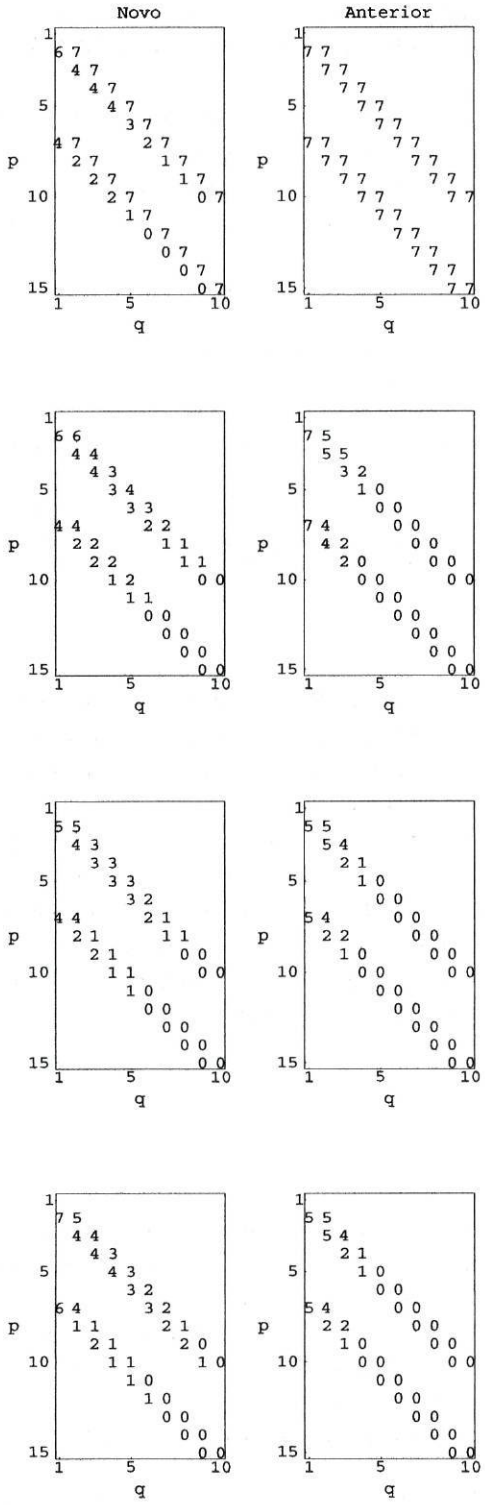


Tabela 3.5: Algarismos significativos dos parâmetros τ, λ, η e ρ, respectivamente, calculados com o novo algoritmo e com o algoritmo de [18], em Fortran90 e utilizando a biblioteca CADNA em precisão simples, para $f(z) = \frac{1}{\sqrt{1.64-1.6z}}$

Dado que nas expressões da relação $\overset{\bullet}{\underset{*}{\bullet\ \bullet}}$ do algoritmo de [18] se fixa $\tau = 1$, nos elementos nas posições $(q, q-1)$ e $(q+5, q-1)$, para $q = 2, \ldots, 10$, da primeira tabela da segunda coluna, τ apresenta precisão máxima.

O facto de, em ambas as colunas, nos elementos nas diagonais (q, q) e $(q+5, q)$, para $q = 2, \ldots, 10$, o parâmetro τ também apresentar sete algarismos significativos correctamente calculados deve-se à possibilidade do Fortran90 calcular o valor de $\tau = 1/\alpha_q$ com exactidão, mantendo durante todo o processo de cálculo a precisão máxima.

No entanto, para o novo algoritmo, nas diagonais adjacentes, i.e., nos elementos $(q, q-1)$ e $(q+5, q-1)$, para $q = 2, \ldots, 10$, esse fenómeno já não se verifica, devido à complexidade da expressão que define τ.

Atendendo ao facto do cálculo sucessivo dos parâmetros τ, λ, η e ρ depender sempre dos parâmetros precedentes é de esperar um decréscimo progressivo da precisão dos parâmetros τ, λ, η e ρ por esta ordem. Este fenómeno verifica-se em ambos os algoritmos embora se revele de forma mais acentuada no algoritmo de [18].

Confirmando as nossas expectativas, a introdução de mais um parâmetro livre nas relações de recorrência de cálculo dos coeficientes dos AFP, constitui uma mais valia no aumento da estabilidade do novo algoritmo. De facto, os resultados expostos na primeira coluna da tabela 3.5, obtidos com o novo algoritmo, mostram-se expressivamente melhores do que os apresentados na segunda coluna, obtidos com o algoritmo de [18].

Dado que nas equações dos coeficientes a_i e b_i intervêm os parâmetros e que o cálculo destes coeficientes é também recursivo, é de esperar que os coeficientes a_i e b_i venham afectados de erros de cálculo que se propagam à medida que se avança na TFP.

Na tabela 3.6, na posição (p, q), é evidenciado o número de algarismos significativos correctamente calculados do coeficiente com menor precisão do numerador e do denominador do aproximante $[p/q]_f^P$, obtidos com o CADNA. Nota-se um aumento substancial da precisão dos coeficientes no novo algoritmo relativamente ao de [18], permitindo construir cerca de mais dez aproximantes de Frobenius-Padé. As tabelas de resultados obtidos com o algoritmo de [18] encontram-se na secção 3.1.

A tabela 3.7 mostra, para cada posição (p, q), o número de algarismos significativos exactos na aproximação $[p/q]_f^P(z)$, i.e.,

$$\left[-\log_{10} \left(\left| \frac{f(z) - [p/q]_f^P(z)}{f(z)} \right| \right) \right].$$

Verifica-se um melhoramento expressivo nos AFP conseguidos com o novo algoritmo relativamente ao algoritmo de [18] (ver tabela 3.1 da secção 3.1), até mesmo numa zona perto da singularidade.

Como se pode observar na tabela 3.7, os resultados sugerem que o comportamento dos aproximantes é perturbado pela singularidade da função $z = 1.64/1.6 = 1.025$, dada a qualidade da aproximação calculada num ponto próximo do extremo do intervalo e da singularidade, $z = 0.9$ e noutro pertencente a uma região *bem comportada*, $z = -0.5$.

Passando a aritmética de vírgula flutuante de dupla precisão, os resultados revelam-se

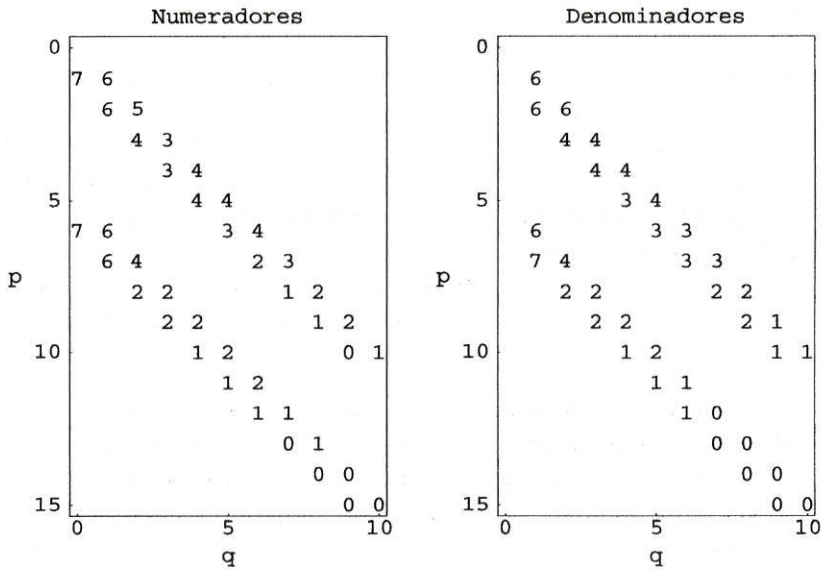


Tabela 3.6: Algarismos significativos correctamente calculados do coeficiente com menor precisão do numerador e do denominador, obtidos com o novo algoritmo programado em Fortran90 com *precisão simples* e utilizando a biblioteca CADNA

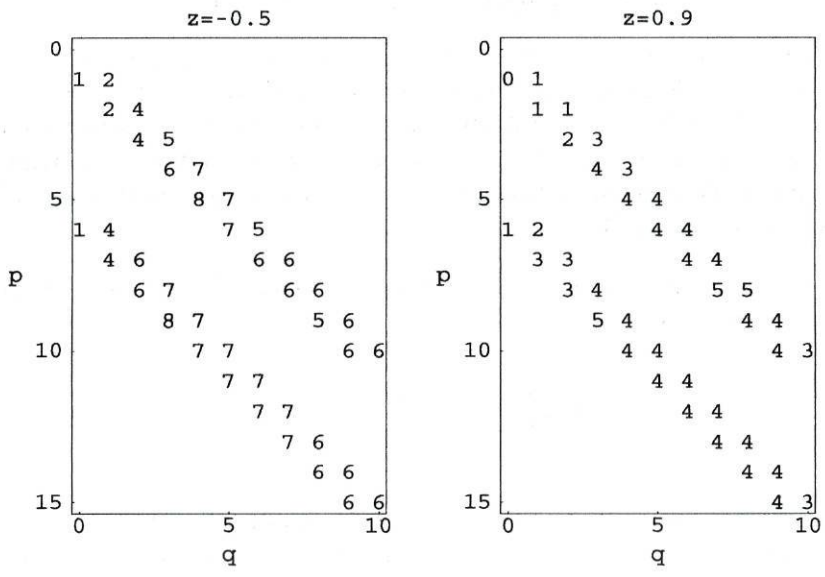


Tabela 3.7: $\left[-\log_{10}\left(\left|\frac{f(z)-[p/q]_f^P(z)}{f(z)}\right|\right)\right]$ calculado no Mathematica, com os $[p/q]_f^P(z)$ construídos a partir dos resultados do novo algoritmo programado em Fortran90 com *precisão simples* e utilizando o CADNA, para $f(z) = \frac{1}{\sqrt{1.64-1.6z}}$

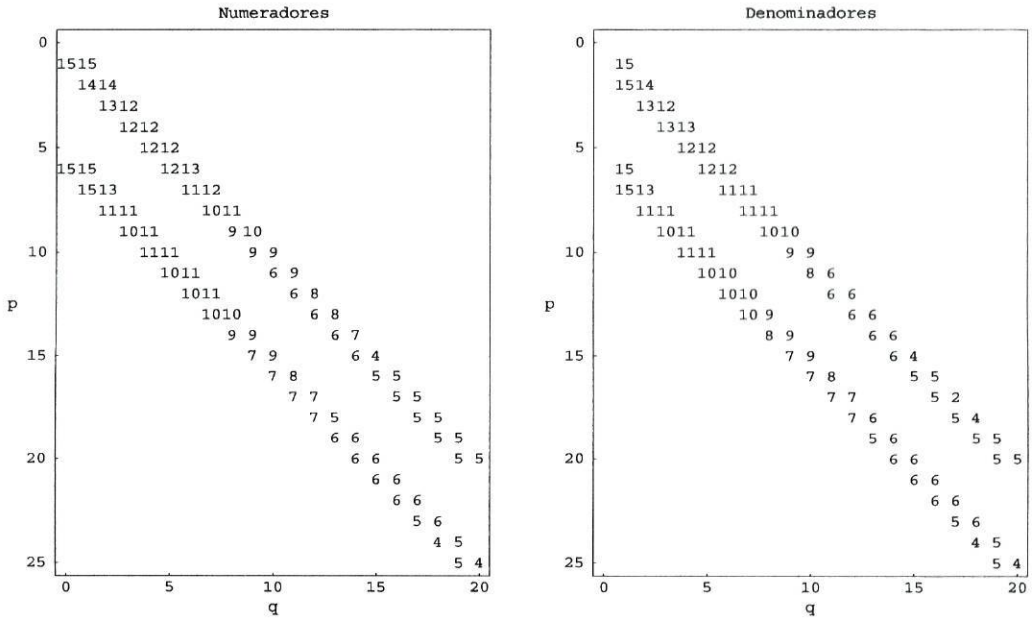


Tabela 3.8: Algarismos significativos correctamente calculados do coeficiente com menor precisão do numerador e do denominador, obtidos com o novo algoritmo programado em Fortran90 com *precisão dupla* e utilizando a biblioteca CADNA

em consonância com os analisados anteriormente.

Na tabela 3.8, recorrendo novamente ao CADNA, verifica-se um aumento bastante significativo da precisão dos coeficientes do numerador e do denominador dos AFP, conseguindo cerca de mais de duas dezenas de aproximantes do que os obtidos com o algoritmo de [18] (ver tabela 3.3 da secção 3.1).

Uma vez mais, na tabela 3.9, assiste-se a uma diminuição substancial do erro relativo da aproximação, revelando a boa qualidade com que os AFP aproximam a soma da série.

Toda esta análise vem demonstrar o aumento expressivo da estabilidade do novo algoritmo de cálculo dos aproximantes de Frobenius-Padé.

Por último resta referir que, dada a perda repentina de precisão perto do ponto $z = 1$ devido à proximidade da singularidade $z = 1.025$, um segundo estudo foi realizado tomando o valor $a = 0.2$ para parâmetro. Para este novo valor temos

$$f^*(z) = \frac{1}{\sqrt{1.04 - 0.4z}}, \quad -1 < z < 1$$

que tem uma singularidade em $z = 2.6$ que se encontra afastada do limite superior do intervalo. A análise pormenorizada descrita neste exemplo foi efectuada para a nova função $f^*(z)$, tendo os resultados expressado as mesmas características, exceptuando a súbita perda de precisão perto de $z = 1$, permitindo, desta forma, supor que se podem generalizar as conclusões.

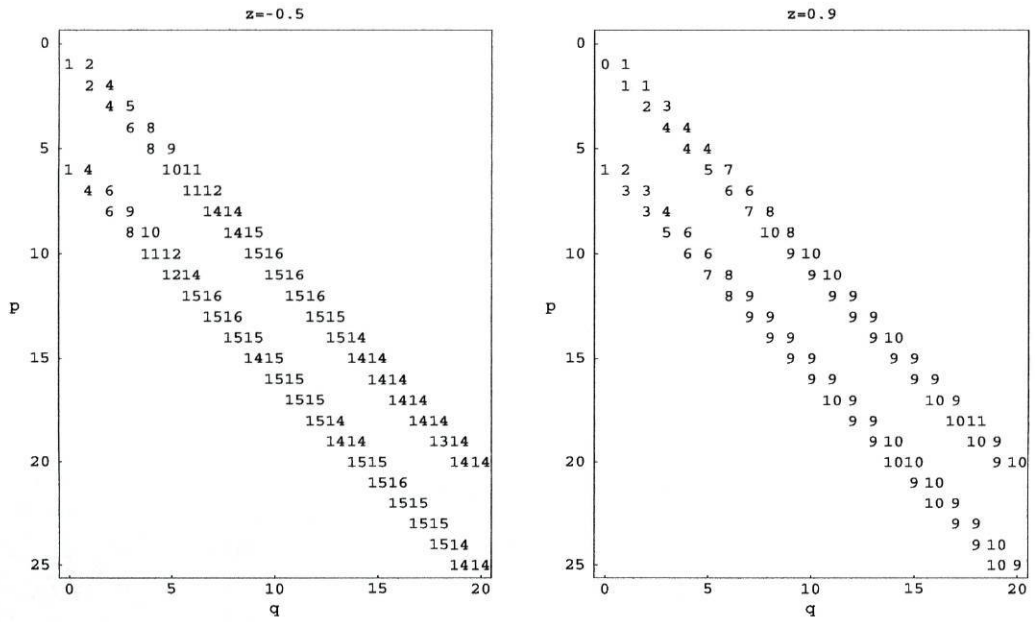


Tabela 3.9: $\left[-\log_{10}\left(\left|\frac{f(z)-[p/q]_f^P(z)}{f(z)}\right|\right)\right]$ calculado no Mathematica, com os $[p/q]_f^P(z)$ construídos a partir dos resultados do novo algoritmo programado em Fortran90 com *precisão dupla* e utilizando o CADNA, para $f(z) = \frac{1}{\sqrt{1.64-1.6z}}$

Conclusões

A introdução de mais um parâmetro livre nas relações de recorrência revelou um aumento significativo da estabilidade do cálculo dos AFP. O algoritmo correspondente a estas novas relações de recorrência foi avaliado e comparado com o anterior algoritmo apresentado em [18] em termos da estabilidade do cálculo e qualidade de aproximação da série.

O novo algoritmo mostra-se bastante mais estável no cálculo de uma sucessão de coeficientes, fornecendo resultados com precisão satisfatória.

Assistiu-se a um aumento de precisão substancial no cálculo dos parâmetros τ, λ, η e ρ, quando comparados com os resultados de [18].

O cálculo dos coeficientes a_i dos numeradores e b_i dos denominadores dos AFP revelou-se também significativamente mais estável, produzindo os aproximantes $[p/q]_f^P$ com melhor qualidade quando comparados com os resultados de algoritmos anteriormente desenvolvidos.

Com o objectivo de avaliar o novo algoritmo desenvolvido em termos de estabilidade de cálculo, este foi testado com outros exemplos de desenvolvimentos de polinómios ortogonais clássicos. Serão apenas apresentados os resultados obtidos com aritmética de precisão simples visto que estes reproduzem antecipadamente as mesmas características dos resultados em dupla precisão. Devido às semelhanças que os resultados obtidos apresentam, a sua discussão será realizada em simultâneo. Todos resultados referentes a estes exemplos são apresentados

no anexo (A).

Os coeficientes dos aproximantes, bem como os primeiros termos da série do erro, neste algoritmo, são calculados pelo programa implementado em Fortran90, adaptado para utilizar a biblioteca CADNA, e escritos num ficheiro com 8 ou 16 algarismos significativos, dependendo da precisão que se considera para os cálculos. No Mathematica implementou-se uma rotina que lê estes coeficientes e, utilizando as suas próprias rotinas de cálculo dos polinómios ortogonais, constrói os aproximantes de Frobenius-Padé.

Consideremos os seguintes desenvolvimentos ortogonais e respectivas funções a aproximar.

Exemplo 3.2 *A partir da fórmula [13]*

$$z^{\mu} = \Gamma(\mu+\beta+1)\sum_{n=0}^{\infty} \frac{(-1)^n(2n+\gamma)\Gamma(n+\gamma)(-\mu)_n}{\Gamma(n+\beta+1)\Gamma(n+\gamma+\mu+1)} R_n^{(\alpha,\beta)}(z), \quad 0<z<1$$

válida para

$$\alpha,\ \beta > -1, \quad -Re(\mu) < \min(1+\beta, 3/4+\beta/2),$$

onde $R_n^{(\alpha,\beta)}(z)$ *representa o polinómio de Jacobi de grau* n *definido no intervalo* $[0,1]$, *com peso* $w(z) = (1-z)^{\alpha}z^{\beta}$ *e* $\gamma = \alpha+\beta+1$.

Tomando $\alpha = \beta = 1/2$ *e* $\mu = -1/2$, *obtemos*

$$f(z) = \frac{1}{\sqrt{z}} = \frac{16}{\pi}\sum_{n=0}^{\infty}(-1)^n \frac{n+1}{(2n+1)(2n+3)} U_n(z), \quad 0<z<1,$$

onde $U_n(z)$ *é o polinómio de Chebyshev de segunda espécie de grau* n *definido no intervalo* $[0,1]$, *normalizado pela condição* $U_n(1) = n+1$. *Estes polinómios satisfazem a relação de recorrência [11]*

$$U_{n+1}(z) = 2(2z-1)U_n(z) - U_{n-1}(z).$$

Assim temos

$$\left\{\alpha_i = \frac{1}{4}, \beta_i = \frac{1}{2}, \gamma_i = \frac{1}{4}\right\}_{i\leq 0}$$

e

$$\left\{f_i = \frac{16}{\pi}(-1)^i \frac{i+1}{(2i+1)(2i+3)}\right\}_{i\leq 0}.$$

Exemplo 3.3 *Considere-se a série [10]*

$$f(z) = \frac{1}{2}e^{z/2} = \sum_{i\geq 0}(-1)^i L_i^{(\alpha)}(z), \quad 0<z<\infty.$$

Os polinómios de Laguerre $L_i^{(\alpha)}(z)$, *normalizados com coeficiente principal* $k_i = \frac{(-1)^i}{i!}$, *satisfazem a relação de recorrência*

$$\begin{cases} L_{i+1}^{(\alpha)}(z) = \frac{2i+\alpha+1-z}{i+1}L_i^{(\alpha)}(z) - \frac{i+\alpha}{i+1}L_{i-1}^{(\alpha)}(z), & i \geq 1 \\ L_0^{(\alpha)}(z) = 1, \quad L_1^{(\alpha)}(z) = 1+\alpha-z \end{cases}.$$

Considerando $\alpha = 0$, *temos*

$$\{\alpha_i = -(i+1), \beta_i = 2i+1, \gamma_i = -i\}_{i\leq 0}$$

e

$$\{f_i = (-1)^i\}_{i\leq 0}.$$

Exemplo 3.4 *Considere-se a série de Hermite [10]*

$$f(z) = e^{2z-1} = \sum_{i\geq 0} \frac{1}{i!} H_i(z), \quad 0 < z < 2.$$

Os polinómios de Hermite $H_i(z)$, *normalizados com coeficiente principal* $k_i = 2^i$, *satisfazem a relação de recorrência*

$$H_{i+1}(z) = 2zH_i(z) - 2iH_{i-1}(z).$$

Assim temos

$$\left\{\alpha_i = \frac{1}{2}, \beta_i = 0, \gamma_i = i\right\}_{i\leq 0}$$

e

$$\left\{f_i = \frac{1}{i!}\right\}_{i\leq 0}.$$

Exemplo 3.5 *Considere-se a série [15, 16]*

$$f(z) = \frac{1}{\sqrt{1-z}} e^{\frac{1}{1+\sqrt{1-z}}} = \sum_{i\geq 0} \frac{1}{2^i i!} B_i(z), \quad -\infty < z < +\infty.$$

Os polinómios mónicos de Bessel $B_i(z)$, *satisfazem a relação de recorrência*

$$\begin{cases} B_{i+1}(z) = (z-\beta_i)B_i(z) + \gamma_i B_{i-1}(z), & i \geq 1 \\ B_0(z) = 1, \quad B_1(z) = z - \beta_0 \end{cases}.$$

Assim temos

$$\beta_0 = -1, \left\{\beta_i = 0, \gamma_i = -\frac{1}{(2i-1)(2i+1)}\right\}_{i\leq 1}$$

e

$$\left\{f_i = \frac{1}{2^i i!}\right\}_{i\leq 0}.$$

Analogamente ao exemplo envolvendo polinómios de Legendre, para testar a propagação dos erros de cálculo no algoritmo, o programa em Fortran90 foi executado para valores de $p \in \{0, 5\}$. Para as várias famílias de polinómios ortogonais consideradas, o resultado do programa é o conjunto dos coeficientes dos aproximantes $[p+q/q-1]_f^P(z)$ e $[p+q/q]_f^P(z)$ para $q = 1, \ldots, 10$.

Seguindo o mesmo plano de estudo, começámos por analisar o efeito da propagação dos erros de cálculo verificado no valor dos parâmetros τ, λ, η e ρ.

Visto que, conforme [19], o CADNA considera a expressão do parâmetro ρ como sendo a principal geradora de instabilidades e que a qualidade deste parâmetro depende da qualidade dos restantes, serão apenas apresentados os resultados relativos a ρ (ver tabela A.1). Os resultados constatam uma vez mais a eficiência e estabilidade do novo algoritmo.

Os resultados relativos ao número de dígitos correctamente calculados presentes no coeficiente de menor precisão do numerador e do denominador do aproximante $[p/q]_f^P$ permitem reforçar as conclusões efectuadas até ao momento (ver tabelas A.2 e A.3). As expressões do novo algoritmo são consideravelmente mais estáveis e, como consequência imediata, os resultados obtidos apresentam maior precisão e por isso maior credibilidade.

A qualidade dos coeficientes permite obter as séries do numerador e do denominador, i.e., $N^{[p/q]}(z) = \sum_{i=0}^{p} a_i P_i$ e $D^{[p/q]}(z) = \sum_{i=0}^{q} b_i P_i$, com maior precisão, o que torna o cálculo dos aproximantes mais robusto e eficiente.

Para qualquer série, a aproximação mais fácil de se obter consegue-se através das suas somas parciais. Assim, uma aproximação possível para cada função $f(z) = \sum_{i\geq 0} f_i P_i$ obtém-se considerando uma soma parcial da série, i.e., $f(z) \approx \sum_{i\geq 0}^{k} f_i P_i$ com $k \in \mathbb{N}$ e quanto maior for k melhor será a aproximação.

Desta forma, sabendo que para construir o aproximante $[p/q]_f^P$ é necessário conhecer ou calcular os primeiros $p+2q+1$ coeficientes do desenvolvimento em série ortogonal de $f(z)$, podemos verificar a real qualidade dos aproximantes comparando-os com as somas parciais associadas.

Visto que o custo computacional do cálculo do aproximante $[p/q]_f^P$ é superior ao custo do cálculo da soma parcial $\sum_{i\geq 0}^{p+2q} f_i P_i$, a aproximação de Frobenius-Padé só faz sentido se a sua qualidade for superior à soma parcial da série.

Usando o argumento anterior e a veracidade expressa nos resultados que o assiste (ver figura A.1), podemos afirmar com clareza que este algoritmo é verdadeiramente mais estável do que os anteriormente desenvolvidos.

Por último resta referir que o estudo em aritmética de vírgula flutuante em dupla precisão foi realizado, tendo os resultados numéricos revelado as mesmas propriedades e daí resultado conclusões semelhantes.

3.3 Modificação do novo algoritmo utilizando uma decomposição LU

Nas duas relações de recorrência das proposições 2.2 e 2.3, uma alternativa às formulas (2.9) e (2.11) consiste em resolver os sistemas (2.8) e (2.10) utilizando uma *decomposição LU com pivotagem parcial.*

A decomposição LU, quando utilizada conjuntamente com uma técnica de pivotagem, confere estabilidade numérica ao cálculo da solução de um sistema. Recordemos sucintamente no que consiste a decomposição LU [21].

Dado um sistema de n equações a n incógnitas $Ax = b$, a ideia é escrever a matriz A como o produto de duas matrizes: $LU = A$, onde L é uma matriz triangular inferior e U uma matriz triangular superior. Consideremos $A = (a_{ij})$, $L = (l_{ij})$ e $U = (u_{ij})$.

Para $n = 4$ obtemos

$$\begin{pmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

A existência de uma factorização $A = LU$ implica que os elementos das matrizes A, L e U satisfaçam as relações

$$a_{ij} = \sum_{m=1}^{n} l_{im} u_{mj}, \quad i, j = 1, \ldots, n. \tag{3.2}$$

Como as matrizes L e U são triangulares e, além disso, L possui diagonal unitária, temos $n(n-1)/2$ elementos de L e $n(n+1)/2$ elementos de U para determinar, num total de n^2 elementos. A expressão (3.2) fornece exactamente n^2 relações, pelo que se poderá pensar em utilizá-la como via para calcular os l_{ij} e os u_{ij}. Embora (3.2) seja uma expressão não-linear envolvendo produtos destes coeficientes, é possível tornear esta dificuldade mediante uma escolha judiciosa da sequência de cálculos.

Em primeiro lugar registemos, dado o carácter de matrizes triangulares de L e U, a validade das expressões

$$l_{im} = 0 \quad se \quad m > i \quad e \quad u_{mj} = 0 \quad se \quad m > j,$$

pelo que, no somatório em (3.2), basta fazer variar m de 1 a $\min(i, j)$, pois os restantes termos são necessariamente nulos.

Consideremos a seguinte sequência de cálculos baseada no varrimento de A por colunas.

Primeira coluna de A

Do que se acabou de dizer, resulta imediatamente que

$$a_{i1} = \sum_{m=1}^{n} l_{im} u_{m1} = l_{i1} u_{11}, \quad i = 1, \ldots, n,$$

donde, recordando que L tem diagonal unitária e no pressuposto que $u_{11} \neq 0$, extraímos as relações

$$u_{11} = a_{11}, \quad l_{i1} = \frac{a_{i1}}{u_{11}}, \quad i = 2, \ldots, n.$$

Ficam assim completamente determinadas as primeiras colunas de U e de L.

Segunda coluna de A

Por outro lado, também é verdade que

$$a_{i2} = \sum_{m=1}^{n} l_{im} u_{m2} = l_{i1} u_{12} + l_{i2} u_{22}, \quad i = 1, \ldots, n.$$

Daqui se tira que

$$a_{12} = l_{11} u_{12}, \quad a_{22} = l_{21} u_{12} + l_{22} u_{22}$$

e, portanto,

$$u_{12} = a_{12}, \quad u_{22} = a_{22} - l_{21}u_{12}.$$

Logo, se $u_{22} \neq 0$,

$$l_{i2} = \frac{a_{i2} - l_{i1}u_{12}}{u_{22}}, \quad i = 3, \ldots, n.$$

As segundas colunas de U e L ficam, assim, totalmente calculadas.

Aplicando sucessivas vezes este processo é possível obter todos os elementos das matrizes U e L, e que as expressões gerais que os determinam são

$$\begin{array}{rcll} u_{ij} & = & a_{ij} - \sum_{m=1}^{i-1} l_{im}u_{mj}, & i = 2, 3, \ldots, n, \quad j = i, i+1, \ldots, n \\ l_{ji} & = & \frac{1}{u_{ii}}\left(a_{ji} - \sum_{m=1}^{i-1} l_{jm}u_{mi}\right), & i = 2, 3, \ldots, n-1, \quad j = i+1, i+2, \ldots, n \end{array}$$

Estas expressões pressupõem que os u_{ii} são todos diferentes de zero.

O algoritmo apresentado consubstancia o chamado *Método de Doolittle* para obter a factorização de uma matriz em termos de uma matriz L triangular inferior de diagonal unitária e uma matriz U triangular superior.

O *algoritmo de Doolittle* envolve o conjunto das equações necessárias para calcular todos os elementos l_{ij} e u_{ij} da seguinte forma:

Algoritmo de Doolittle

$u_{11} = a_{11}$
para i **de** 2 **até** n
  $u_{1i} = a_{1i}$
  $l_{i1} = a_{i1}/u_{11}$
fim
para i **de** 2 **até** $n-1$
  $u_{ii} = a_{ii} - \sum_{m=1}^{i-1} l_{im}u_{mi}$
  para j **de** $i+1$ **até** n
    $u_{ij} = a_{ij} - \sum_{m=1}^{i-1} l_{im}u_{mj}$
    $l_{ji} = \frac{1}{u_{ii}}\left(a_{ji} - \sum_{m=1}^{i-1} l_{jm}u_{mi}\right)$
  fim
fim
$u_{nn} = a_{nn} - \sum_{m=1}^{i-1} l_{nm}u_{mn}$

Ao algoritmo de Doolittle é aplicada a técnica de *pivotagem parcial* que passamos a descrever. Neste processo designamos por *pivot* os elementos u_{ii} que aparecem em denominador no cálculo de l_{ji}. Quanto maior for o $|u_{ii}|$ mais estável é o cálculo de l_{ji}.

De acordo com esta técnica, são candidatos a pivots os elementos

$$\omega_k = a_{ki} - \sum_{m=1}^{i-1} l_{km}u_{mi}, \quad k = i, \ldots, n$$

resultantes da, eventual, permuta das linhas i e k, com $k \geq i$, da matriz A. Deve escolher-se para pivot o elemento ω_k que tiver maior valor absoluto, mais precisamente, seja p um índice

tal que

$$|\omega_p| = \max_{i \leq k \leq n} |\omega_k|.$$

Se $p \neq i$ procede-se à permuta das linhas p e i. Esta permutação de linhas traduz-se na decomposição LU pela pré-multiplicação por uma matriz de permutação P. Assim, obtemos $PA = LU$ ou $L^{-1}PA = U$.

Realizada esta decomposição, a resolução do sistema $PAx = Pb \Leftrightarrow LUx = Pb$ efectua-se resolvendo dois sistemas triangulares, um superior e outro inferior

$$\begin{cases} Ly = Pb \\ Ux = y \end{cases}.$$

Na tentativa de melhorar a estabilidade do algoritmo apresentado na secção 2.4, procurando retardar o efeito da propagação dos erros de arredondamento, em vez de usarmos as formulas solução dos sistemas (2.8) e (2.10), construímos as matrizes correspondentes a esses sistemas de equações e recorremos a uma decomposição LU com pivotagem parcial.

Com o objectivo de avaliar o melhoramento da estabilidade deste algoritmo modificado, este foi programado em Fortran90 e adaptado para utilizar a biblioteca CADNA, usando aritmética de precisão simples e dupla, recorrendo a subrotinas retiradas de [20]. Os resultados obtidos foram comparados com o algoritmo apresentado na secção 2.4.

3.4 Resultados obtidos com a implementação modificada do novo algoritmo

Este algoritmo foi implementado em Fortran90, recorrendo a rotinas padrão de resolução de sistemas de equações lineares utilizando uma decomposição LU com pivotagem parcial, com aritmética de precisão simples e dupla. Com o objectivo de ilustrar os resultados obtidos, o programa foi testado com os exemplos seguintes:

Exemplo 2.1 *Consideremos novamente a família de polinómios de Legendre $P_i(z)$ e a série*

$$f(z) = \frac{1}{\sqrt{1 - 2az + a^2}} = \sum_{n \geq 0} a^n P_n(z)$$

com $a = 0.8$.

Mais uma vez, visto que os resultados em precisão simples revelam precocemente as mesmas características dos resultados obtidos em precisão dupla, são apresentados apenas os primeiros.

Um facto a considerar está relacionado com a existência de vários zeros nos elementos das matrizes em causa. Como é sabido, a resolução de matrizes quase-esparsas através de uma decomposição LU pode produzir instabilidades devido à acumulação de resíduos causadas pelas diversas multiplicações por 0., i.e., elementos nulos na precisão da máquina.

No sentido de minimizar este problema, uma vez que o algoritmo foi programado em Fortran90 utilizando aritmética de vírgula flutuante em precisão simples, os elementos nulos das matrizes foram declarados em precisão dupla.

(p,q)	$\frac{\rho-\rho^{LU}}{\rho}$	$\frac{\rho-\rho^{F}}{\rho}$	(p,q)	$\frac{\rho-\rho^{LU}}{\rho}$	$\frac{\rho-\rho^{F}}{\rho}$
$(2,1)$	$\mathbf{0}$	1.1×10^{-7}	$(2,2)$	$\mathbf{7.6\times10^{-7}}$	1.0×10^{-6}
$(3,2)$	2.3×10^{-6}	1.8×10^{-7}	$(3,3)$	$\mathbf{3.5\times10^{-5}}$	4.9×10^{-5}
$(4,3)$	$\mathbf{2.9\times10^{-4}}$	3.2×10^{-4}	$(4,4)$	4.2×10^{-3}	4.2×10^{-3}
$(5,4)$	2.0×10^{-2}	2.0×10^{-2}	$(5,5)$	2.4×10^{-1}	2.4×10^{-1}
$(6,5)$	1.3×10^{2}	9.6×10^{1}	$(6,6)$	1.0	1.0
$(7,6)$	1.7	1.7	$(7,7)$	3.9	3.5
$(8,7)$	1.2	1.2	$(8,8)$	$\mathbf{3.3}$	4.5
$(9,8)$	$\mathbf{7.2\times10^{-1}}$	7.5×10^{-1}	$(9,9)$	$\mathbf{2.8}$	4.2
$(10,9)$	9.4×10^{-1}	8.2×10^{-1}	$(10,10)$	$\mathbf{8.0\times10^{-1}}$	8.3×10^{-1}
$(11,10)$	3.4	3.0	$(11,11)$	8.4×10^{-1}	8.0×10^{-1}
$(12,11)$	2.2	2.2	$(12,12)$	7.7×10^{-1}	7.4×10^{-1}
$(13,12)$	1.2	1.2	$(13,13)$	1.7	1.7
$(14,13)$	7.3×10^{-1}	6.1×10^{-1}	$(14,14)$	$\mathbf{7.6\times10^{-1}}$	8.3×10^{-1}
$(15,14)$	$\mathbf{1.6}$	1.9	$(15,15)$	9.9×10^{-1}	9.4×10^{-1}
$(16,15)$	$\mathbf{4.4\times10^{1}}$	1.0×10^{2}	$(16,16)$	1.0	1.0
$(17,16)$	5.4	4.9	$(17,17)$	1.1	1.1
$(18,17)$	$\mathbf{1.6\times10^{1}}$	1.5×10^{2}	$(18,18)$	$\mathbf{9.6\times10^{-1}}$	1.0
$(19,18)$	$\mathbf{1.9}$	5.6	$(19,19)$	$\mathbf{2.7}$	3.7
$(20,19)$	$\mathbf{3.7\times10^{-1}}$	7.5×10^{1}	$(20,20)$	1.0	7.7×10^{-1}

Tabela 3.10: $\left|\frac{\rho-\rho^{*}}{\rho}\right|$ calculado no Mathematica, onde ρ é calculado exactamente usando cálculo formal no Mathematica e ρ^{*} corresponde simultaneamente ao ρ^{LU} calculado com o algoritmo modificado e ao ρ^{F} calculado com o novo algoritmo, ambos programados em Fortran90 com *precisão simples*, para $f(z)=\frac{1}{\sqrt{1.64-1.6z}}$

Resultados numéricos

A utilização de uma decomposição LU tinha como objectivo conferir estabilidade numérica ao cálculo dos parâmetros τ, λ, η e ρ pela acção da pivotagem parcial, que corresponde a colocar o elemento de módulo máximo em denominador. Observamos que as entradas das matrizes geradas no método de Doolittle são todas sensivelmente da mesma ordem de grandeza pelo que as trocas possíveis não conferem alteração substancial à grandeza do pivot. Por essa razão não é de esperar que o efeito da pivotagem produza um melhoramento muito significativo.

Constatámos que no cálculo dos parâmetros τ^{LU}, λ^{LU}, η^{LU} e ρ^{LU}, efectuado recorrendo a uma decomposição LU, a pivotagem é efectuada geralmente uma vez por sistema e mais frequentemente no cálculo dos elementos da sub-diagonal.

Os resultados revelam pequenas alterações nos valores dos parâmetros τ^{LU}, λ^{LU}, η^{LU} e ρ^{LU} quando comparados com os valores dos parâmetros τ^{F}, λ^{F}, η^{F} e ρ^{F} obtidos através do algoritmo descrito em 2.4.

Uma vez que o parâmetro ρ^{LU} é o mais vulnerável a erros de cálculo, na tabela 3.10 representam-se os erros relativos de ρ^{LU}, $\left|\frac{\rho-\rho^{LU}}{\rho}\right|$, e de ρ^{F}, $\left|\frac{\rho-\rho^{F}}{\rho}\right|$, relativamente ao valor exacto de ρ calculado efectuando cálculo formal no Mathematica.

(p,q)	$\left\|\left(\frac{a_i-a_i^{LU}}{a_i}\right)^p_{i=0}\right\|_\infty$	$\left\|\left(\frac{a_i-a_i^{F}}{a_i}\right)^p_{i=0}\right\|_\infty$	$\left\|\left(\frac{b_i-b_i^{LU}}{b_i}\right)^p_{i=0}\right\|_\infty$	$\left\|\left(\frac{b_i-b_i^{F}}{b_i}\right)^p_{i=0}\right\|_\infty$
$(1,0)$	0	0	0	0
$(1,1)$	0	0	0	0
$(2,1)$	$\mathbf{4.9 \times 10^{-7}}$	1.1×10^{-6}	1.3×10^{-7}	1.3×10^{-7}
$(2,2)$	$\mathbf{7.0 \times 10^{-7}}$	1.5×10^{-6}	7.0×10^{-7}	6.7×10^{-7}
$(3,2)$	5.5×10^{-6}	1.5×10^{-6}	9.2×10^{-7}	4.7×10^{-7}
$(3,3)$	$\mathbf{2.8 \times 10^{-5}}$	4.0×10^{-5}	$\mathbf{2.7 \times 10^{-5}}$	3.8×10^{-5}
$(4,3)$	$\mathbf{4.2 \times 10^{-4}}$	4.6×10^{-4}	$\mathbf{1.9 \times 10^{-4}}$	2.0×10^{-4}
$(4,4)$	3.2×10^{-3}	3.2×10^{-3}	3.1×10^{-3}	3.1×10^{-3}
$(5,4)$	2.0×10^{-2}	2.0×10^{-2}	1.2×10^{-2}	1.2×10^{-2}
$(5,5)$	1.8×10^{-1}	1.8×10^{-1}	1.8×10^{-1}	1.7×10^{-1}
$(6,5)$	9.3×10^{1}	7.0×10^{1}	7.8×10^{1}	5.9×10^{1}
$(6,6)$	9.9×10^{-1}	9.9×10^{-1}	9.8×10^{-1}	9.8×10^{-1}
$(7,6)$	1.3	1.3	1.1	1.1
$(7,7)$	1.5	1.4	1.5	1.4
$(8,7)$	2.7	2.7	1.0	1.0
$(8,8)$	1.0	1.0	1.0	1.0
$(9,8)$	3.9	3.9	1.0	1.0
$(9,9)$	1.0	1.0	1.0	1.0
$(10,9)$	**2.2**	2.4	1.0	1.0
$(10,10)$	1.0	1.0	1.0	1.0
$(11,10)$	5.7	5.7	1.0	1.0
$(11,11)$	1.0	1.0	1.0	1.0
$(12,11)$	1.0×10^{1}	1.0×10^{1}	1.0	1.0
$(12,12)$	1.0	1.0	1.0	1.0
$(13,12)$	1.4×10^{1}	1.4×10^{1}	1.0	1.0
$(13,13)$	1.9	1.9	1.0	1.0
$(14,13)$	1.6×10^{1}	1.6×10^{1}	1.0	1.0
$(14,14)$	**1.8**	1.9	1.0	1.0
$(15,14)$	$\mathbf{2.3 \times 10^{1}}$	2.4×10^{1}	1.0	1.0
$(15,15)$	2.0	2.0	1.0	1.0
$(16,15)$	$\mathbf{2.0 \times 10^{2}}$	5.6×10^{2}	**1.0**	1.9
$(16,16)$	2.2	2.2	1.0	1.0
$(17,16)$	1.3×10^{1}	1.3×10^{1}	1.0	1.0
$(17,17)$	2.4	2.4	1.0	1.0
$(18,17)$	$\mathbf{5.7 \times 10^{1}}$	4.1×10^{2}	**1.0**	1.3
$(18,18)$	**2.5**	2.6	1.0	1.0
$(19,18)$	7.6×10^{1}	5.0	1.0	1.0
$(19,19)$	9.9	3.7	1.0	1.0
$(20,19)$	$\mathbf{7.1 \times 10^{1}}$	8.1×10^{1}	1.0	1.0
$(20,20)$	**2.9**	3.0	1.0	1.0

Tabela 3.11: $\left\|\left(\frac{s_i-s_i^*}{s_i}\right)^p_{i=0}\right\|_\infty$ calculado no Mathematica, onde s_i representa simultaneamente os coeficientes a_i e b_i calculados exactamente usando cálculo formal no Mathematica e s_i^* corresponde simultaneamente aos coeficientes a_i^{LU} e b_i^{LU} calculados com o algoritmo modificado ou aos a_i^F e b_i^F calculados com o novo algoritmo, ambos programados em Fortran90 com *precisão simples*, para $f(z) = \frac{1}{\sqrt{1.64-1.6z}}$

Na tabela 3.11, para cada par (p, q) , estão representados os erros relativos:

- dos coeficientes a_i^{LU} do numerador e b_i^{LU} do denominador do aproximante $\overline{[p/q]}_f^P$, calculados recorrendo ao algoritmo que implementa a decomposição LU e utilizando os parâmetros τ^{LU}, λ^{LU}, η^{LU} e ρ^{LU}, i.e., $\left\| \left(\frac{a_i - a_i^{LU}}{a_i} \right)_{i=0}^{p} \right\|_\infty$ e $\left\| \left(\frac{b_i - b_i^{LU}}{b_i} \right)_{i=0}^{q} \right\|_\infty$ respectivamente,

- dos coeficientes a_i^F do numerador e b_i^F do denominador do aproximante $[p/q]_f^P$, calculados recorrendo ao novo algoritmo descrito na secção 2.4 e utilizando os parâmetros τ^F, λ^F, η^F e ρ^F, i.e., $\left\| \left(\frac{a_i - a_i^F}{a_i} \right)_{i=0}^{p} \right\|_\infty$ e $\left\| \left(\frac{b_i - b_i^F}{b_i} \right)_{i=0}^{q} \right\|_\infty$ respectivamente,

relativamente aos valores exactos de a_i e b_i calculados efectuando cálculo formal no Mathematica.

Nas tabelas 3.10 e 3.11 a negrito figuram os resultados onde o novo algoritmo modificado foi mais eficiente.

Embora o melhoramento dos coeficientes a_i^{LU} do numerador e b_i^{LU} do denominador seja pouco substancial, este traduz-se de forma visível nos aproximantes $\overline{[p/q]}_f^P$ produzidos com estes coeficientes.

Os gráficos da figura 3.1 representam, em escala logarítmica, várias curvas dos erros relativos dos aproximantes $[p/q]_f^P(z)$, $\left| \frac{f(z) - [p/q]_f^P(z)}{f(z)} \right|$, encontrados na secção 3.2 (a vermelho) e dos aproximantes $\overline{[p/q]}_f^P(z)$, $\left| \frac{f(z) - \overline{[p/q]}_f^P(z)}{f(z)} \right|$, construídos com os resultados da decomposição LU (a verde).

Os gráficos evidenciam um aumento considerável da qualidade de certos aproximantes obtidos utilizando o algoritmo modificado, construídos a partir de valores a_i^{LU} e b_i^{LU} com qualidade ligeiramente superior aos obtidos via implementação directa do algoritmo descrito na secção 2.4.

Uma vez mais, foi analisada a interferência causada pela proximidade da singularidade $z = 1.025$ ao intervalo $[-1, 1]$. Considerando uma função $f^*(z)$ com uma singularidade em $z = 2.6$, os resultados apresentam características semelhantes, exceptuando a perda progressiva de precisão perto de $z = 1$.

Conclusões

Os resultados do algoritmo modificado, desenvolvido com vista ao melhoramento da estabilidade numérica do cálculo dos quatro parâmetros τ^{LU}, λ^{LU}, η^{LU} e ρ^{LU}, revelam novos valores apresentando frequentemente um suave melhoramento em relação aos resultados obtidos via implementação directa do algoritmo 2.4.

No que diz respeito à qualidade dos coeficientes a_i^{LU} do numerador e b_i^{LU} do denominador, assistimos a um ligeiro melhoramento, reflexo da qualidade dos parâmetros.

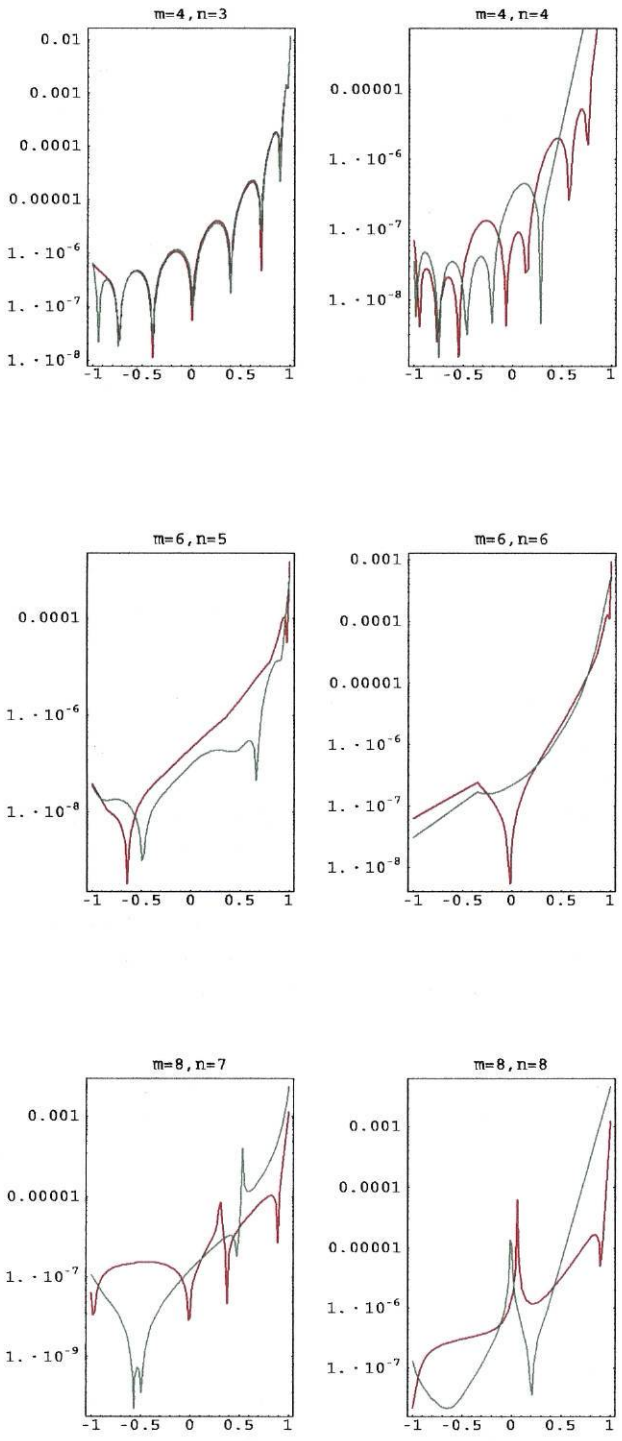


Figura 3.1: Erros relativos, em escala logarítmica, para uma amostra de aproximantes $[m/n]_f^P(z)$ calculados com a implementação directa do novo algoritmo, $\left|\frac{f(z)-[p/q]_f^P(z)}{f(z)}\right|$ a vermelho, e de aproximantes $\overline{[m/n]}_f^P(z)$ homólogos calculados com o algoritmo modificado, $\left|\frac{f(z)-\overline{[p/q]}_f^P(z)}{f(z)}\right|$ a verde, utilizando Fortran90 em precisão simples

No entanto, esse ligeiro melhoramento nalguns dos coeficientes, conduz a um aumento visível da qualidade de certos aproximantes $[p/q]_f^P$, evidenciado através da comparação com a soma parcial da série.

Voltando aos exemplos 3.2, 3.3, 3.4 e 3.5, consideremos novamente as várias famílias de polinómios ortogonais e respectivas séries.

Tal com no exemplo 3.1, no cálculo dos parâmetros τ^{LU}, λ^{LU}, η^{LU} e ρ^{LU} efectuado recorrendo a uma decomposição LU, a pivotagem realiza-se, em geral, uma vez por sistema e mais frequentemente no cálculo dos elementos da sub-diagonal. Os resultados revelam pequenas alterações nos valores dos parâmetros citados quando comparados com os valores exactos (ver tabelas A.4, A.6, A.8 e A.10).

O melhoramento dos coeficientes a_i^{LU} do numerador e b_i^{LU} do denominador revela-se pouco expressivo (ver tabelas A.5, A.7, A.9 e A.11), embora se traduza de forma essencial no cálculo de certos aproximantes $[p/q]_f^P$, apresentado estes um aumento significativo da qualidade em relação aos AFP conseguidos através da implementação do algoritmo descrito em 2.4 (ver figuras A.2, A.3, A.4 e A.5).

3.5 Conclusões

A introdução de mais um parâmetro livre nas relações de recorrência de cálculo dos AFP, conduziu a um novo algoritmo de cálculo substancialmente mais estável. Este algoritmo foi avaliado e comparado com o anterior algoritmo apresentado em [18] em termos da estabilidade do cálculo e qualidade de aproximação da série.

Os programas foram testados com séries de polinómios de Jacobi (Legendre e Chebyshev de segunda espécie), de Hermite, de Laguerre e de Bessel, tendo os resultados numéricos obtidos revelado, de forma homogénea, um melhoramento significativo da estabilidade numérica no cálculo dos quatro parâmetros τ, λ, η e ρ e dos coeficientes do numerador e do denominador dos AFP relativamente aos programas anteriormente desenvolvidos na literatura. As sucessões de aproximantes calculadas produzem valores convergentes para os valores exactos.

Os resultados do algoritmo modificado, desenvolvido com vista ao melhoramento da estabilidade numérica do cálculo dos quatro parâmetros citados, revelam novos valores que conduzem a um ligeiro melhoramento dos coeficientes a_i e b_i. O melhoramento verificado, evidenciado na comparação com a soma parcial da série que, como vimos, é um tipo de aproximação possível, traduz-se de forma visível ao cálculo de alguns aproximantes $[p/q]_f^P$.

As capacidades de cálculo formal e numérica do Mathematica, assim como as funcionalidades do CADNA usado com os programas Fortran90, foram essenciais para efectuar o estudo aqui apresentado. A estabilidade do cálculo dos coeficientes das componentes dos AFP foi avaliada com o auxilio do CADNA.

Capítulo 4

Condicionamento do cálculo dos Aproximantes de Frobenius-Padé

Este capítulo é dedicado ao estudo do condicionamento do cálculo dos AFP.

São introduzidos os conceitos de erros iniciais, estabilidade de um algoritmo e condicionamento de um problema. É apresentada a fórmula dos acréscimos finitos para o estudo do efeito dos erros iniciais na avaliação de funções de várias variáveis.

Para avaliar a propagação de pequenas perturbações nos dados iniciais, os coeficientes f_i do desenvolvimento em série ortogonal da função a aproximar foram perturbados e foi analisada a forma como essa perturbação se transmite aos cálculos dos parâmetros τ, λ, η e ρ, dos coeficientes a_i do numerador e b_i do denominador dos AFP e dos próprios aproximantes $[p/q]_f^P$. Os resultados numéricos obtidos permitem afirmar que se trata de um problema muito bem condicionado pois a perturbação nos f_i transmite-se apenas em verdadeira grandeza aos aproximantes.

4.1 Introdução

4.1.1 Erros iniciais

Devemos ter sempre presente o facto de que os dados de um problema estão frequentemente afectados de erros, ora porque resultam de experiências físicas sujeitas à capacidade de máquinas e/ou observação, ora porque a representação dos dados em vírgula flutuante está sujeita aos respectivos erros de representação. Erros desta natureza nos dados de um problema são designados por *erros iniciais*. Estes erros estão presentes antes e independentemente da aplicação de qualquer método numérico para resolver o problema.

Há problemas cuja solução é muito sensível a variações nos dados. Queremos dizer que, para certos problemas, erros nos dados quase desprezáveis, podem originar variações muito grandes na solução.

4.1.2 Análise dos erros

Consideremos uma função f de n variáveis independentes: $x_1, \ldots, x_n$.

Sendo x_i^0 valores aproximados de x_i, com $i = 1(1)n$, ao tomarmos $w_0 = f(x_1^0, \ldots, x_n^0)$ para valor aproximado de $w = f(x_1, \ldots, x_n)$, cometemos o erro

$$\Delta w_0 = w - w_0 = f(x_1, \ldots, x_n) - f(x_1^0, \ldots, x_n^0)$$

que provém dos erros existentes nos dados

$$\Delta x_i^0 = x_i - x_i^0, \quad i = 1(1)n.$$

Designado por M e M_0 os pontos de coordenadas $(x_1^0 + \Delta x_1^0, \ldots, x_n^0 + \Delta x_n^0)$ e $(x_1^0, \ldots, x_n^0)$, supondo que f é contínua em $[M_0, M]$ e que admite derivadas parciais

$$f_{x_i} = \frac{\partial f}{\partial x_i},$$

em $]M_0, M[$, a *fórmula dos acréscimos finitos* permite escrever [9]

$$\Delta w_0 = \sum_{i=1}^{n} f_{x_i}(P) \Delta x_i^0, \tag{4.1}$$

onde P é um ponto interior do segmento $\overline{M_0 M}$, i.e., $P = M_0 + \theta(M - M_0)$, $0 < \theta < 1$, ou ainda, P é um ponto de coordenadas $(x_1^0 + \theta \Delta x_1^0, \ldots, x_n^0 + \theta \Delta x_n^0)$.

Se as derivadas parciais de f forem contínuas em M_0 e se os erros Δx_i^0, $i = 1(1)n$, forem suficientemente pequenos, poderemos escrever, menosprezando quantidades de 2^a ordem em Δx_i^0,

$$\Delta w_0 \approx \sum_{i=1}^{n} f_{x_i}(M_0) \Delta x_i^0,$$

$$\overline{\Delta w_0} \lesssim \sum_{i=1}^{n} |f_{x_i}(M_0)| \overline{\Delta x_i^0}$$

e, supondo $w_0 = f(M_0) \neq 0$,

$$\left| \frac{\overline{\Delta w_0}}{w_0} \right| \lesssim \sum_{i=1}^{n} \left| \frac{f_{x_i}(M_0)}{f(M_0)} \right| \overline{\Delta x_i^0}, \tag{4.2}$$

obtendo assim estimativas para os erros absoluto e relativo máximos de w_0, que, evidentemente, não nos devem satisfazer em projectos de alta precisão. Nesse caso não substituiríamos P por M_0, escreveríamos

$$\overline{\Delta w_0} \leq \sum_{i=1}^{n} |f_{x_i}(P)| \overline{\Delta x_i^0} \tag{4.3}$$

e deixaríamos P variar continuamente entre os dois pontos $(x_1^0 - \Delta x_1^0, \ldots, x_n^0 - \Delta x_n^0)$ e $(x_1^0 + \Delta x_1^0, \ldots, x_n^0 + \Delta x_n^0)$, i.e., tomaríamos $P = (x_1^0 + \theta \overline{\Delta x_1^0}, \ldots, x_n^0 + \theta \overline{\Delta x_n^0})$, $-1 < \theta < 1$ e calcularíamos

$$S_{x_i} = \sup |f_{x_i}(P)|,$$

para, finalmente, obter

$$\overline{\Delta w_0} \leq \sum_{i=1}^{n} S_{x_i} \overline{\Delta x_i^0}. \tag{4.4}$$

Num problema matemático, cuja resolução em computador envolva vários cálculos, dever-se-á realizar um estudo, como o efectuado acima, de cada fórmula utilizada para se avaliar de que forma os erros nos argumentos destas fórmulas se propagam aos respectivos resultados. No entanto, na prática, em problemas que envolvam cálculos extensos e complexos, não é esse o método utilizado. Nessas condições, procede-se a um estudo numérico heurístico tal como o efectuado neste trabalho.

4.1.3 Condicionamento e estabilidade

Devido à existência dos chamados erros iniciais, o problema que resolvemos na prática nem sempre coincide com o problema colocado. Como é evidente, só terão interesse os resultados que não difiram muito da solução exacta do problema em causa.

Um problema em que pequenas alterações nos dados iniciais ou parâmetros originam grandes variações na solução diz-se um problema *mal condicionado*. Se esta situação não ocorrer diz-se que é um problema *bem condicionado*.

Na utilização dum método numérico para resolução de um problema matemático deverá ter-se em conta o comportamento dos erros de arredondamento que necessariamente se cometem ao longo da execução dos cálculos. Com efeito, cada operação aritmética executada em computador é realizada de forma inexacta conduzindo a resultados afectados de erros, que por sua vez se vão propagar caso estes resultados sejam reutilizados posteriormente noutros cálculos. Este fenómeno está relacionado com a noção de *estabilidade* de um método numérico.

Dizemos que um método numérico é *estável* se a acumulação dos erros durante os cálculos não tem influência significativa no resultado final. Caso contrário o método diz-se *instável*.

É claro que um algoritmo implementado num problema mal condicionado nunca poderá ser estável. Por outro lado, num problema bem condicionado, um método estável produz sempre bons resultados.

4.2 Efeito da perturbação nos dados iniciais (f_i)

Nesta secção pretendemos avaliar o efeito da propagação dos erros nos dados iniciais, i.e., nos coeficientes f_i do desenvolvimento em série ortogonal de $f(z)$.

Para podermos realizar com rigor a análise do condicionamento deste problema de cálculo torna-se necessário considerar três sub-problemas distintos: o primeiro correspondente ao cálculo dos parâmetros τ, λ, η e ρ; o segundo corresponde ao cálculo dos coeficientes a_i e b_i e finalmente o terceiro ao cálculo dos aproximantes $[p/q]_f^P$ propriamente ditos.

Nos testes numéricos apresentados, utilizam-se funções satisfazendo dois requisitos: os coeficientes do seu desenvolvimento em série ortogonal podem calcular-se, pelo menos teoricamente, até uma ordem tão longe quanto o necessário, com a precisão de cálculo da

máquina; a função é descrita por uma fórmula analítica exacta, permitindo calcular o seu valor, pelo menos teoricamente, em cada ponto do intervalo de aproximação, com a precisão de cálculo da máquina. O primeiro destes requisitos permite calcular as sucessões de aproximantes tão longe quanto se queira, sem restrições sobre os graus dos numeradores e dos denominadores. Permite também isolar os erros inerentes aos processos de cálculo implementados com o algoritmo, dos erros nos dados, ou seja, dos erros nos coeficientes da série. O segundo requisito prende-se com a necessidade de analisar o algoritmo comparando os resultados, tanto quanto possível, com os valores exactos da função. Na maioria dos casos não temos acesso ao primeiro requisito, dado que só se conhecem os primeiros coeficientes do desenvolvimento ortogonal da função em causa.

Em geral, os coeficientes f_i do desenvolvimento ortogonal são calculados recorrendo à resolução de integrais que, como é sabido, depende de métodos numéricos que produzem resultados com um certo grau de incerteza. Nesse caso estamos perante um problema com erros iniciais.

O CADNA fornece-nos a possibilidade de perturbar os dados de um problema com o objectivo de verificar a influência do número de algarismos significativos exactos dos dados iniciais no resultado.

A subrotina data_st(X,ER_X) permite ao utilizador introduzir um erro relativo de uma determinada ordem nos dados iniciais, reduzindo o número de algarismos significativos exactos. Este procedimento permite constatar até que ponto o número de algarismos significativos exactos do resultado depende do número de algarismos significativos exactos dos dados iniciais. Neste caso em particular, tomámos para X os coeficientes f_i e considerámos ER_X$= 5 \times 10^{-12}$, i.e.,

$$\left| \frac{f_i - \widetilde{f}_i}{f_i} \right| \leq 5 \times 10^{-12}.$$

Com os coeficientes perturbados $\widetilde{f}_i$ executámos o algoritmo descrito na secção 2.4, utilizando aritmética de vírgula flutuante em dupla precisão, para vários exemplos correspondentes às diferentes famílias de polinómios ortogonais.

Os gráficos apresentados, assim como as tabelas, são gerados pela função Plot do software Mathematica. Os coeficientes dos aproximantes, bem como os primeiros termos da série do erro, neste algoritmo, são calculados pelo programa implementado em Fortran90 utilizando a biblioteca CADNA e escritos num ficheiro, apenas com algarismos significativos correctamente calculados. No Mathematica implementou-se uma rotina que lê estes coeficientes e, utilizando as suas próprias rotinas de cálculo dos polinómios ortogonais, constrói os AFP.

Consideremos novamente o exemplo da série de Legendre que é utilizado frequentemente na literatura sobre a aproximação de Frobenius-Padé.

Exemplo 2.1 *Consideremos novamente a família de polinómios de Legendre e a série*

$$f(z) = \frac{1}{\sqrt{1 - 2az + a^2}} = \sum_{n \geq 0} a^n P_n(z)$$

com $a = 0.8$.

Resultados numéricos

Nas tabelas 4.1 e 4.2 figuram respectivamente, na posição do aproximante $[p/q]_f^P$ da TFP, os algarismos significativos dos parâmetros perturbados $\widetilde{\tau}$, $\widetilde{\lambda}$, $\widetilde{\eta}$ e $\widetilde{\rho}$ e dos parâmetros produzidos sem perturbação τ, λ, η e ρ, obtidos com a biblioteca CADNA.

Atendendo ao facto do cálculo de cada um dos parâmetros τ, λ, η e ρ depender sempre dos parâmetros precedentes e à perda de precisão progressiva de τ, assistimos mais uma vez ao decréscimo da precisão dos restantes parâmetros.

No entanto, comparando as duas tabelas, verificámos que o efeito destrutivo da propagação dos erros iniciais não se revela de carácter explosivo, i.e.,

$$\left|\frac{f_i - \widetilde{f_i}}{f_i}\right| \leq \xi \Rightarrow \left|\frac{\phi_i - \widetilde{\phi_i}}{\phi_i}\right| \leq \xi,$$

onde $\widetilde{\phi_i}$ e ϕ_i representam os parâmetros obtidos, respectivamente, com e sem a perturbação.

Na verdade, se considerarmos ambas as situações, com e sem perturbação, verificámos que, tanto na tabela 4.1 como na 4.2, a perda de algarismos significativos ao longo das diagonais apresenta o mesmo comportamento, i.e., durante o processo de cálculo, em ambos os casos, perdemos o mesmo número de algarismos significativos. Tivemos oportunidade de constatar também que, no caso dos parâmetros perturbados, começámos com uma diferença de quatro algarismos significativos em relação aos parâmetros não perturbados e no final do processo mantivemos a mesma diferença.

Mais precisamente, este estudo empírico revela que os erros nos dados iniciais, f_i, propagam-se em verdadeira grandeza ao cálculo dos parâmetros τ, λ, η e ρ.

Dado que nas equações dos coeficientes a_i e b_i intervêm os parâmetros τ, λ, η e ρ e sabendo que os erros produzidos no cálculo dos parâmetros se propagam à medida que se avança nos cálculos, só podemos esperar que, na melhor da hipóteses, também neste problema o efeito da propagação dos erros iniciais não se revele explosivo.

Na tabela 4.3, na posição (p, q), é evidenciado o número de algarismos significativos correctos do coeficiente com menor precisão do numerador e do denominador do aproximante $\widetilde{[p/q]}_f^P$, fornecidos pelo CADNA. Nota-se uma diminuição substancial da precisão dos coeficientes, mas mais uma vez se verifica que estamos perante um problema bem condicionado, dado que o efeito da propagação dos erros se traduz de forma linear, i.e.

$$\left|\frac{f_i - \widetilde{f_i}}{f_i}\right| \leq \xi \Rightarrow \left|\frac{\phi_i - \widetilde{\phi_i}}{\phi_i}\right| \leq \xi \Rightarrow \left|\frac{s_i - \widetilde{s_i}}{s_i}\right| \leq \xi,$$

onde $\widetilde{s_i}$ e s_i representam os coeficientes a_i do numerador e b_i do denominador obtidos, respectivamente, com e sem a perturbação. Os resultados sem perturbação encontram-se na tabela 3.8.

Tal como no estudo dos parâmetros, também aqui constatámos que em ambos os casos, com e sem perturbação, perdemos o mesmo número de algarismos significativos ao logo das diagonais e que a diferença inicial entre os algarismos significativos dos coeficientes com e sem perturbação se mantém até ao final processo, resultado que nos permite concluir que os

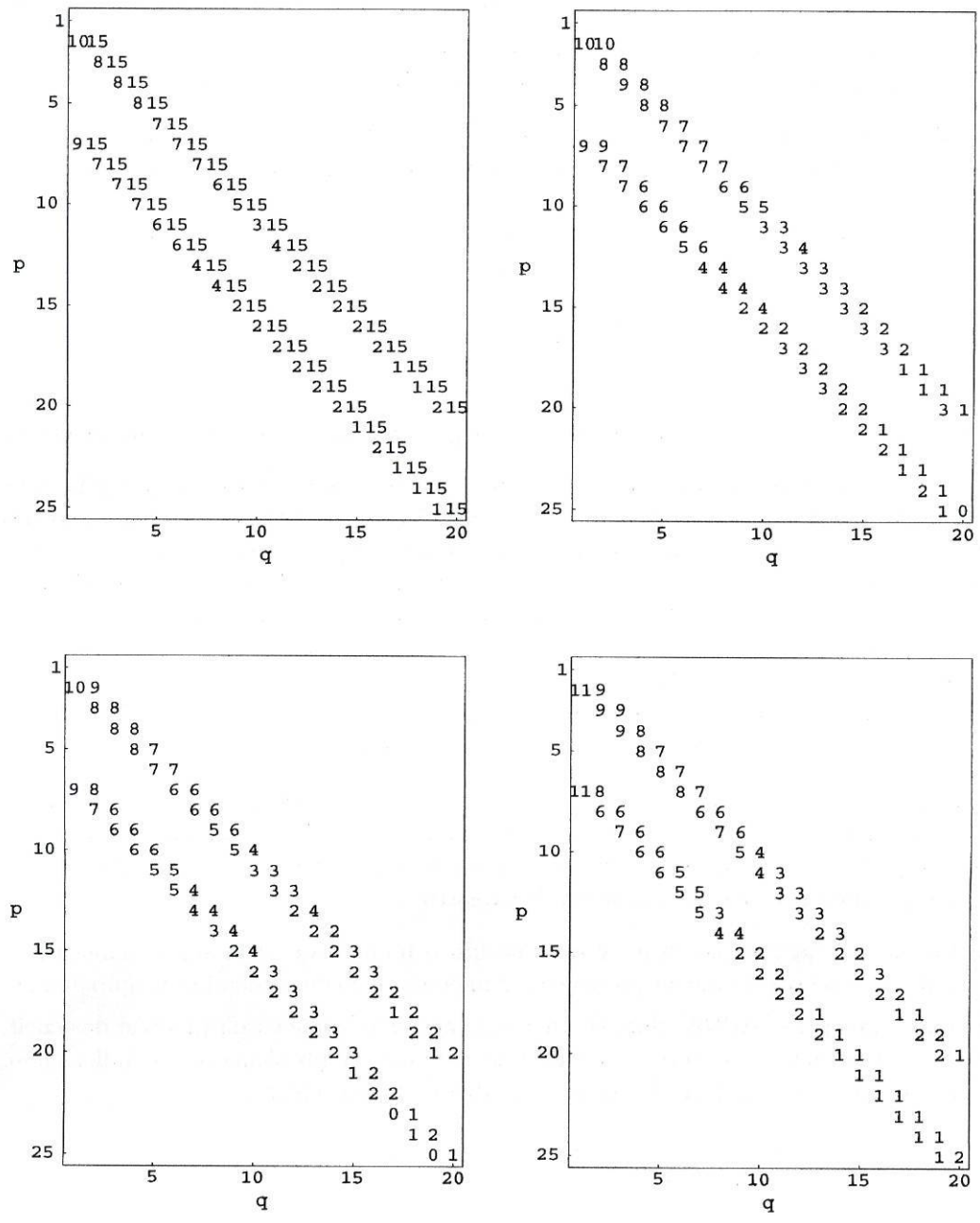


Tabela 4.1: Algarismos significativos correctamente calculados dos parâmetros $\widetilde{\tau}$, $\widetilde{\lambda}$, $\widetilde{\eta}$ e $\widetilde{\rho}$ respectivamente, obtidos a partir dos $\widetilde{f}_i$ perturbados com um erro relativo da ordem de 5×10^{-12}, calculados com o novo algoritmo programado em Fortran90 com precisão dupla e utilizando o CADNA

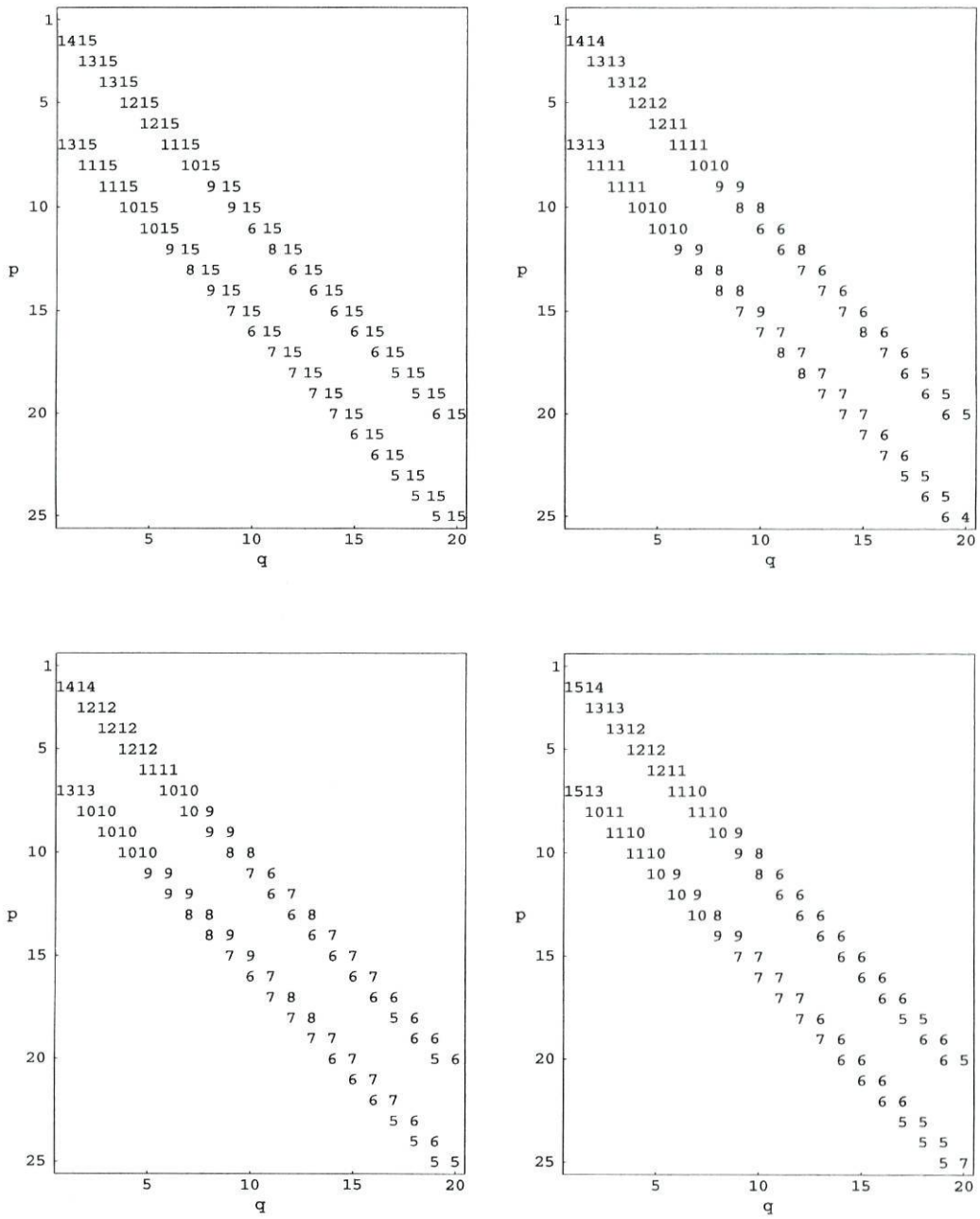


Tabela 4.2: Algarismos significativos correctamente calculados dos parâmetros τ, λ, η e ρ respectivamente, obtidos a partir dos f_i, calculados com o novo algoritmo programado em Fortran90 com precisão dupla e utilizando o CADNA

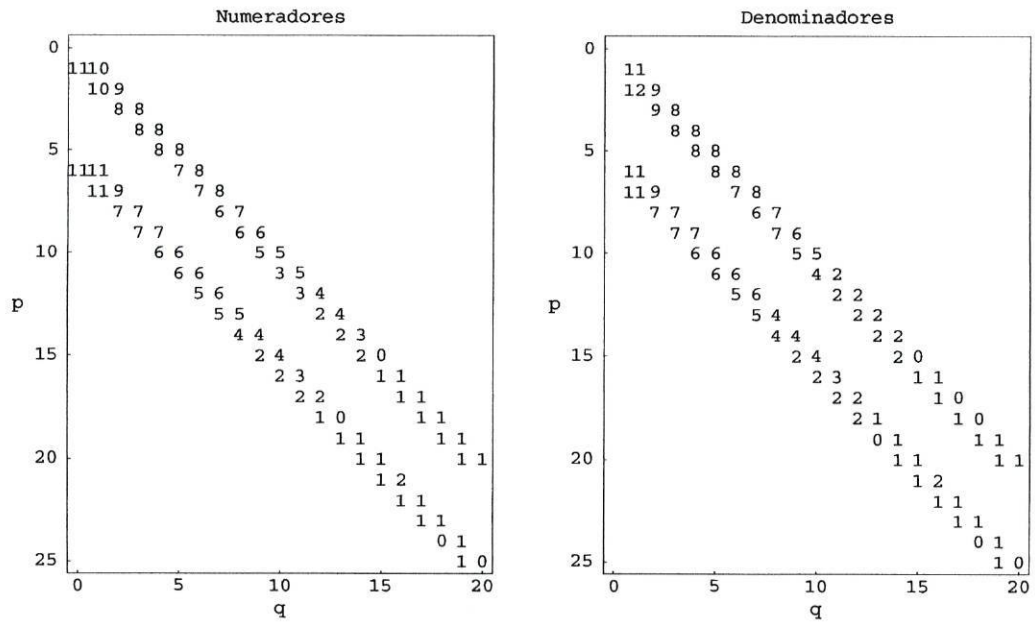


Tabela 4.3: Algarismos significativos correctamente calculados do coeficiente com menor precisão do numerador e do denominador, obtidos a partir dos $\widetilde{f}_i$ perturbados com um erro relativo da ordem de 5×10^{-12}, calculados com o novo algoritmo programado em Fortran90 com precisão dupla e utilizando a biblioteca CADNA

erros nos dados iniciais, f_i, propagam-se em verdadeira grandeza ao cálculo dos coeficientes a_i e b_i.

Os gráficos da figura 4.1 representam, em escala logarítmica, as curvas dos erros relativos para uma amostra de aproximantes $\widetilde{[m/n]}_f^P(z)$ calculados com os valores $\widetilde{f}_i$ perturbados e de aproximantes $[m/n]_f^P(z)$ homólogos calculados com os coeficientes f_i correctos.

Verifica-se que a repentina perda de qualidade dos aproximantes perto do limite superior do intervalo $[-1, 1]$ é efeito da presença da singularidade em $z = 1.025$.

Quando o valor $p+q+1$ é pequeno, a ordem de aproximação $D^{[p/q]}(z)f(z) - N^{[p/q]}(z) = \mathcal{O}(P_{p+q+1}(z))$ é pequena, de modo que a diferença entre $[p/q]_f^P = \frac{N^{[p/q]}}{D^{[p/q]}} = \frac{\sum_{i=0}^{p} a_i P_i}{\sum_{i=0}^{q-1} b_i P_i + P_q}$ e $\widetilde{[p/q]}_f^P = \frac{\widetilde{N}^{[p/q]}}{\widetilde{D}^{[p/q]}} = \frac{\sum_{i=0}^{p} \widetilde{a}_i P_i}{\sum_{i=0}^{q-1} \widetilde{b}_i P_i + P_q}$ não é visível se compararmos cada um deles com a série $f(z)$, uma vez que a propagação dos erros nos dados não afectou os algarismos que os aproximantes têm coincidentes com os da série. Por esse motivo, os gráficos dos aproximantes inferiores ao $[6/6]_f^P$ coincidem.

Quando a ordem de aproximação cresce, a propagação dos erros iniciais vem afectar algarismos significativos no aproximante perturbado que deveriam estar coincidentes com a série. Assim, os gráficos citados não são mais coincidentes e afastam-se um do outro à medida que progredimos na ordem de aproximação. Este facto é esclarecido nos gráficos da figura 4.1.

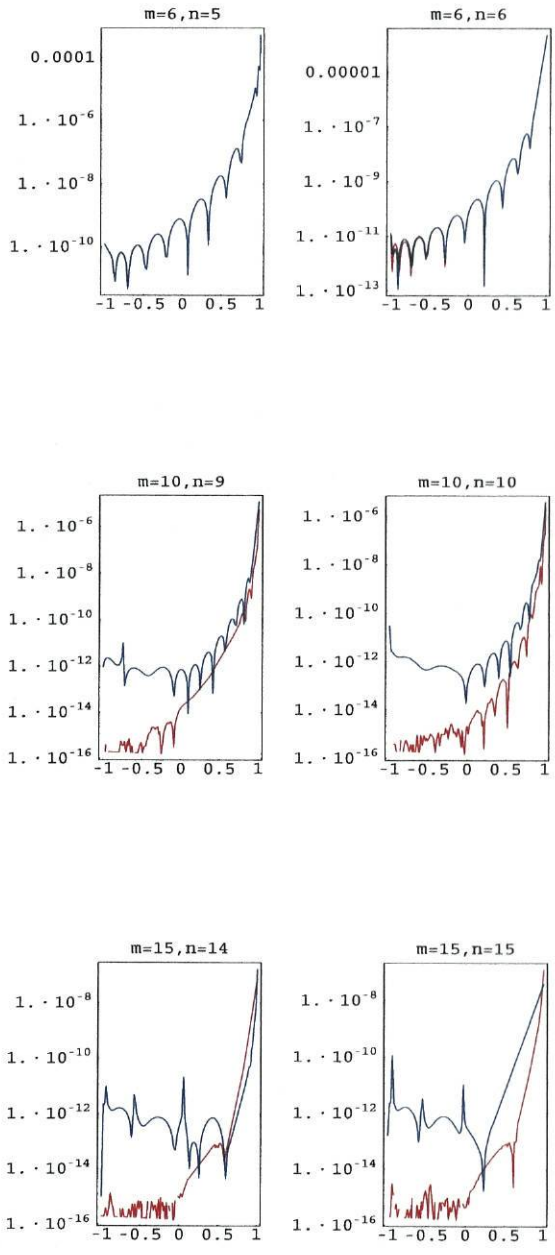


Figura 4.1: Erros relativos, em escala logarítmica, para uma amostra de aproximantes $\widetilde{[m/n]}_f^P(z)$ calculados com os valores $\widetilde{f}_i$ perturbados, $\left|\frac{f(z)-\widetilde{[m/n]}_f^P(z)}{f(z)}\right|$ a azul, e de aproximantes $[m/n]_f^P(z)$ homólogos calculados com os coeficientes f_i , $\left|\frac{f(z)-[m/n]_f^P(z)}{f(z)}\right|$ a vermelho

Para averiguar o efeito causado pela proximidade da singularidade ao intervalo, um segundo estudo foi realizado considerando $a = 0.2$. Para este novo valor obtemos a função

$$f^*(z) = \frac{1}{\sqrt{1.04 - 0.4z}}, \quad -1 < z < 1.$$

Os resultados revelaram as mesmas propriedades, exceptuando a súbita perda de precisão perto de $z = 1$, permitindo, desta forma, supor que se podem generalizar as conclusões.

Conclusões

Quando introduzimos uma perturbação nos coeficientes f_i do desenvolvimento em série ortogonal de $f(z)$, esta propaga-se ao cálculo dos parâmetros τ, λ, η e ρ, que por sua vez provocam perturbações nos coeficientes a_i do numerador e b_i do denominador.

No entanto, verifica-se que o efeito da propagação dos erros iniciais não tem carácter explosivo, tendo os resultados revelado que, à medida que se avança na TFP, a propagação dos erros é efectuada de forma aparentemente linear.

Se observarmos o problema principal, partindo do cálculo dos f_i e terminando no cálculo dos aproximantes $[p/q]_f^P$, podemos afirmar que se trata de um problema muito bem condicionado pois a perturbação nos f_i transmite-se apenas em verdadeira grandeza aos coeficientes dos aproximantes. Verificamos que as perturbações nos a_i e b_i são absorvidas, de tal maneira que a perturbação inicial nos f_i é propagada ao cálculo dos parâmetros e dos coeficientes, para depois ser significativamente reduzida no cálculo efectivo dos aproximantes $[p/q]_f^P$.

Voltando aos exemplos 3.2, 3.3, 3.4 e 3.5, consideremos novamente as várias famílias de polinómios ortogonais e respectivas séries.

Para todas as séries consideradas, a perturbação produzida nos coeficientes f_i difundiu-se ao cálculo dos parâmetros τ, λ, η e ρ, introduzindo, consequentemente, erros nos coeficientes a_i do numerador e b_i do denominador (ver tabelas B.1, B.3, B.5 e B.7).

O efeito da propagação dos erros iniciais apresenta, uma vez mais, carácter limitado, dado que os resultados revelam uma propagação aparentemente linear (ver tabelas B.2, B.4, B.6 e B.8).

Tal como no exemplo 3.1, o problema principal, i.e., partindo do cálculo dos f_i e terminando no cálculo dos aproximantes $[p/q]_f^P$, mostra-se muito bem condicionado pois a perturbação nos f_i propaga-se ao logo do cálculo dos aproximantes sem aumentar de proporção. Novamente, assistimos a uma absorção das perturbações nos a_i e b_i, de tal maneira que a perturbação inicial nos f_i se transmite em verdadeira grandeza ao cálculo dos parâmetros e dos coeficientes mas depois é absorvida no cálculo efectivo dos aproximantes $[p/q]_f^P$ (ver tabelas B.1, B.2, B.3 e B.4).

4.3 Efeito da perturbação nos coeficientes (a_i e b_i)

O estudo apresentado nesta secção surge com o objectivo de analisar o condicionamento do cálculo dos AFP a partir dos coeficientes a_i e b_i. Nesse sentido é conveniente abstrair os erros de cálculo numérico cometidos ao efectuar as operações do algoritmo. Tal é possível partindo de uma representação exacta dos dados do problema e efectuando cálculo formal no Mathematica (ver apêndice (D)).

Vamos realizar um estudo empírico, recorrendo especialmente à análise gráfica, dada a complexidade do desenvolvimento da fórmula dos acréscimos finitos verificada para alguns exemplos.

Os coeficientes a_i do numerador e b_i do denominador provêm de um algoritmo recursivo que, regra geral, fornece resultados afectados pela acumulação dos erros produzidos à medida que se progride nos cálculos.

Com o objectivo de avaliar a propagação dos erros nos coeficientes a_i e b_i aos AFP $[p/q]_f^P$, vamos perturbar, utilizando o Mathematica, os coeficientes produzidos formalmente pelo algoritmo descrito na secção 2.3. Com estes coeficientes perturbados aleatoriamente são construídos os AFP que, para os vários exemplos testados, se revelam quase tão bons quanto os exactos.

Com o Mathematica podemos simular a presença de uma perturbação nos coeficientes a_i e b_i. Aos coeficientes em causa é adicionada uma perturbação aleatória

$$\xi = \left(\frac{Random[Integer, \{0, 100\}]}{Random[Integer, \{100, 200\}]} - \frac{1}{2} \right) 10^{-\alpha}$$

sendo α a ordem de precisão da perturbação. O comando `Random[Integer,{a,b}]` fornece-nos um número inteiro aleatório pertencente ao intervalo $[a, b]$ e, efectuando as devidas transformações, conseguimos que a perturbação seja um número racional, para que o Mathematica efectue cálculo formal, e que $|\xi| \leq \frac{1}{2} \times 10^{-\alpha}$. Com estes novos valores perturbados $\widetilde{a_i} = a_i + \xi_i$ e $\widetilde{b_i} = b_i + \xi_i$ executámos o algoritmo descrito na secção 2.3 para vários exemplos correspondentes às diferentes famílias de polinómios ortogonais.

Os gráficos apresentados são gerados pela instrução Plot do software Mathematica. Os coeficientes a_i do numeradores e b_i dos denominadores dos aproximantes, bem como os primeiros termos da série do erro neste algoritmo, são calculados pelo programa implementado no Mathematica usando cálculo formal. Em seguida, implementou-se uma rotina que perturba os coeficientes a_i e b_i e, utilizando as suas próprias rotinas de cálculo dos polinómios ortogonais, constrói os AFP.

Vejamos os seguintes exemplos.

Exemplo 2.1 *Consideremos uma vez mais a família de polinómios de Legendre e a série*

$$f(z) = \frac{1}{\sqrt{1 - 2az + a^2}} = \sum_{n \geq 0} a^n P_n(z)$$

com $a = 0.8$.

Usando cálculo formal no Mathematica e efectuando várias perturbações de diferentes

ordens obtemos valores $\widetilde{a_i}$ e $\widetilde{b_i}$ alterados que serão utilizados para construir os aproximantes $\widetilde{[q/q-1]}_f^P(z)$ e $\widetilde{[q/q]}_f^P(z)$ com $q = 1, \ldots, 20$.

Resultados numéricos

Considerando na expressão $|\xi| \leq \frac{1}{2} \times 10^{-\alpha}$, os coeficientes a_i do numerador e b_i do denominador foram perturbados considerando $\alpha \in \{8, 12, 14\}$. Desta forma, obtemos três sucessões de aproximantes distintas.

Os gráficos da figura 4.2 representam, em escala logarítmica, as curvas dos erros relativos

$$\left| \frac{f(z) - \left(\widetilde{[m/n]}_f^P(z) \right)_\alpha}{f(z)} \right|$$

para uma amostra dos aproximantes $\left(\widetilde{[m/n]}_f^P(z) \right)_\alpha$ calculados.

Graficamente, as curvas do erro relativo dos aproximantes perturbados tomando $\alpha = 8$ são representadas a azul, as curvas do erro relativo dos aproximantes perturbados considerando $\alpha = 10$ surgem a verde e a vermelho figuram as curvas do erro relativo dos aproximantes perturbados fixando $\alpha = 14$.

Como podemos constatar pela análise gráfica, os valores dos erros relativos, obtidos por comparação com a série, para os diferentes aproximantes são reduzidos, aumentando significativamente à medida que nos aproximamos da singularidade. Algumas das curvas chegam a ser quase coincidentes exibindo o insignificante efeito da propagação dos erros iniciais nos coeficientes a_i e b_i.

Mais uma vez tivemos em consideração o efeito causado pela proximidade ao intervalo da singularidade e os resultados, obtidos através de um segundo estudo realizado tomando $a = 0.2$, apresentam as mesmas características, excepto a súbita perda de precisão perto de $z = 1$, permitindo supor que se podem generalizar as conclusões.

O problema do cálculo dos AFP a partir dos coeficientes do numerador e do denominador revela-se muito bem condicionado, conseguindo-se aproximantes de óptima qualidade.

Conclusões

As curvas dos erros relativos dos aproximantes revelam que o efeito da propagação dos erros nos coeficientes sobre os aproximantes é praticamente nulo, não afectando de forma visível a qualidade dos AFP.

O problema de cálculo dos aproximantes $[p/q]_f^P$ a partir dos coeficientes a_i do numerador $N^{[p/q]}(z)$ e b_i do denominador $D^{[p/q]}(z)$ é extremamente bem condicionado, produzindo aproximantes $[p/q]_f^P = \frac{N^{[p/q]}(z)}{D^{[p/q]}(z)}$ de excelente qualidade no que diz respeito à aproximação da função $f(z)$ em causa.

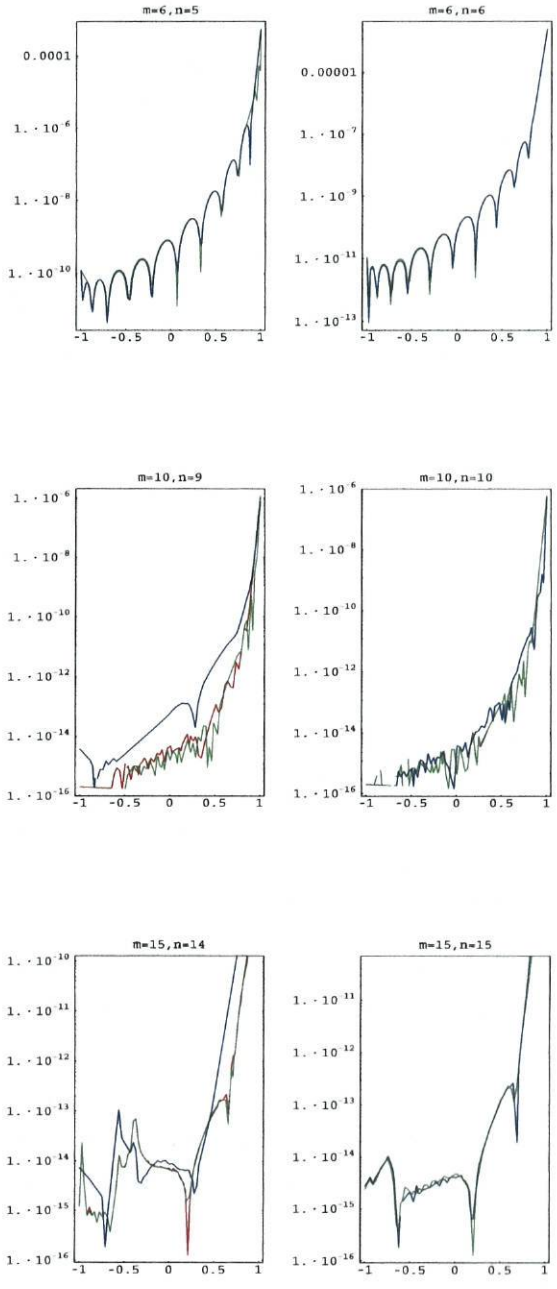


Figura 4.2: Erros relativos, em escala logarítmica, para uma amostra de aproximantes $\widetilde{[m/n]}_f^P(z)$ calculados com os valores $\widetilde{a_i}$ e $\widetilde{b_i}$ perturbados, $\left|\frac{f(z)-\left(\widetilde{[m/n]}_f^P(z)\right)_\alpha}{f(z)}\right|$, representados a azul para $\alpha = 8$, a verde para $\alpha = 10$ e a vermelho para $\alpha = 14$, usando cálculo formal no Mathematica

Tomando novamente aos exemplos 3.2, 3.3, 3.4 e 3.5, consideremos as várias famílias de polinómios ortogonais e respectivas séries.

Realizando um estudo semelhante para cada uma das séries dos exemplos citados, constatámos, mais uma vez, que os valores dos erros relativos, obtidos por comparação com a série para os diferentes aproximantes, são reduzidos. Algumas das curvas chegam a ser coincidentes exibindo o imperceptível efeito da propagação dos erros nos coeficientes a_i e b_i (ver tabelas B.5, B.6, B.7 e B.8).

Para todos os exemplos tratados nesta tese, o cálculo dos AFP $[p/q]_f^P$ a partir dos coeficientes a_i do numerador $N^{[p/q]}(z)$ e b_i do denominador $D^{[p/q]}(z)$ revela-se muito bem condicionado, produzindo, sempre, aproximantes $[p/q]_f^P = \frac{N^{[p/q]}(z)}{D^{[p/q]}(z)}$ de excepcional qualidade no que diz respeito à aproximação da função $f(z)$ em causa, até mesmo quando temos coeficientes a_i e b_i com baixa precisão.

4.4 Conclusões

Quando provocamos uma perturbação nos coeficientes f_i do desenvolvimento em série ortogonal de $f(z)$, esta propaga-se ao cálculo dos parâmetros τ, λ, η e ρ, que por sua vez provocam uma diminuição da precisão dos coeficientes a_i do numerador e b_i do denominador.

No entanto, verifica-se que o efeito da propagação dos erros iniciais não tem carácter explosivo, tendo os resultados revelado que, à medida que se avança na TFP, a propagação dos erros é efectuada de forma aparentemente constante.

Se observarmos o problema que parte do cálculo dos f_i e termina no cálculo dos aproximantes $[p/q]_f^P$ podemos afirmar que se trata de um problema muito bem condicionado pois a perturbação nos f_i transmite-se apenas em verdadeira grandeza aos coeficientes dos aproximantes. Verificamos que as perturbações nos a_i e b_i são absorvidas, de tal maneira que a perturbação inicial nos f_i é propagada ao cálculo dos parâmetros e dos coeficientes na mesma ordem de grandeza, para depois ser bruscamente reduzida no cálculo efectivo dos aproximantes $[p/q]_f^P$.

O problema de cálculo dos aproximantes $[p/q]_f^P$ a partir dos coeficientes a_i do numerador $N^{[p/q]}(z)$ e b_i do denominador $D^{[p/q]}(z)$ é extremamente bem condicionado, produzindo aproximantes $[p/q]_f^P = \frac{N^{[p/q]}(z)}{D^{[p/q]}(z)}$ de excelente qualidade no que diz respeito à aproximação da função $f(z)$ em causa.

O segundo sub-problema, i.e., perturbar o valor dos parâmetros τ, λ, η e ρ e verificar de que forma a perturbação se propaga ao cálculo dos coeficientes, não foi realizado explicitamente. Esse trabalho está implícito na análise dos resultados do algoritmo modificado, uma vez que este produziu ligeiras alterações nos valores dos parâmetros, alterações essas que, como vimos, se transmitem ao cálculo dos AFP.

Capítulo 5

Conclusões

Um dos principais objectivos deste trabalho foi localizar e identificar as instabilidades do processo de cálculo dos aproximantes de Frobenius-Padé, bem como ultrapassar essas instabilidades encontrando novas relações de recorrência que conferiram maior estabilidade ao cálculo dos aproximantes $[p/q]_f^P$.

A introdução de mais um parâmetro livre nas relações de recorrência de cálculo dos AFP, conduziu a um novo algoritmo de cálculo substancialmente mais estável. Este algoritmo foi avaliado e comparado com o anterior algoritmo apresentado em [18] em termos da estabilidade do cálculo e qualidade de aproximação da série.

Os resultados do algoritmo modificado, desenvolvido com vista ao melhoramento da estabilidade numérica do cálculo dos quatro parâmetros citados, revelam novos valores que conduzem a um ligeiro melhoramento dos coeficientes a_i e b_i. O melhoramento verificado, evidenciado na comparação com a soma parcial da série que, como vimos, é um tipo de aproximação possível, traduz-se de forma visível ao cálculo de alguns aproximantes $[p/q]_f^P$.

Os testes realizados incluíram séries ortogonais dos quatro tipos de polinómios clássicos, o que é inédito na literatura sobre o tema. Assim, os programas foram testados com séries de polinómios de Jacobi (Legendre e Chebyshev de segunda espécie), de Hermite, de Laguerre e de Bessel, tendo os resultados numéricos obtidos revelado, de forma homogénea, um melhoramento significativo da estabilidade numérica no cálculo dos quatro parâmetros τ, λ, η e ρ e dos coeficientes do numerador e do denominador dos AFP relativamente aos programas anteriormente desenvolvidos na literatura. As sucessões de aproximantes calculadas produzem valores convergentes para os valores exactos.

Numa segunda etapa deste trabalho concentrámos a nossa atenção sobre o condicionamento do cálculo dos aproximantes de Frobenius-Padé. Com esse objectivo, realizámos um estudo inédito seguindo a metodologia adequada a este tipo de problemas: perturbámos os dados iniciais e analisámos de que forma essas pequenas perturbações se propagam ao longo do processo de cálculo aproximantes $[p/q]_f^P$.

Constatámos que quando provocamos uma perturbação nos coeficientes f_i do desenvolvimento em série ortogonal de $f(z)$, esta propaga-se ao cálculo dos parâmetros τ, λ, η e ρ, que por sua vez provocam uma diminuição da precisão dos coeficientes a_i do numerador e

b_i do denominador. No entanto, verifica-se que o efeito da propagação dos erros iniciais não tem carácter explosivo, tendo os resultados revelado que, à medida que se avança na TFP, a propagação dos erros é efectuada de forma aparentemente constante.

Se observarmos o problema que parte do cálculo dos f_i e termina no cálculo dos aproximantes $[p/q]_f^P$ podemos afirmar que se trata de um problema muito bem condicionado pois a perturbação nos f_i transmite-se apenas em verdadeira grandeza aos coeficientes dos aproximantes. Verificamos que as perturbações nos a_i e b_i são absorvidas, de tal maneira que a perturbação inicial nos f_i é propagada ao cálculo dos parâmetros e dos coeficientes na mesma ordem de grandeza, para depois ser bruscamente reduzida no cálculo efectivo dos aproximantes $[p/q]_f^P$.

O problema de cálculo dos aproximantes $[p/q]_f^P$ a partir dos coeficientes a_i do numerador $N^{[p/q]}(z)$ e b_i do denominador $D^{[p/q]}(z)$ é extremamente bem condicionado, produzindo aproximantes $[p/q]_f^P = \frac{N^{[p/q]}(z)}{D^{[p/q]}(z)}$ de excelente qualidade no que diz respeito à aproximação da função $f(z)$ em causa.

O segundo sub-problema, i.e., perturbar o valor dos parâmetros τ, λ, η e ρ e verificar de que forma a perturbação se propaga ao cálculo dos coeficientes, não foi realizado explicitamente. Esse trabalho está implícito na análise dos resultados do algoritmo modificado, uma vez que este produziu ligeiras alterações nos valores dos parâmetros, alterações essas que, como vimos, se transmitem ao cálculo dos AFP.

De realçar as capacidades de cálculo formal e numérica do Mathematica, assim como as funcionalidades do CADNA usado com os programas Fortran90, que se revelaram essenciais para efectuar o estudo aqui apresentado. Nomeadamente, a análise do condicionamento do cálculo dos AFP exigiu realizar os cálculos formalmente com o Mathematica e, por outro lado, a estabilidade do cálculo dos coeficientes das componentes dos AFP foi avaliada com o auxilio do CADNA.

Em conclusão, os programas desenvolvidos neste trabalho constituem um avanço no cálculo estável dos AFP relativamente ao software anteriormente existente. O problema do condicionamento do cálculo destes aproximantes foi, nesta tese, pela primeira vez estudado.

Apêndice A

Gráficos e tabelas do capítulo 2

Os gráficos apresentados no capítulo 2, assim como as tabelas, são gerados pela função Plot do software Mathematica utilizando a opção `Compiled->False` (ver apêndice (D)).

Comecemos por enunciar novamente os exemplos que foram analisados nesta tese:

Exemplo 2.2 *A partir da fórmula [13]*

$$z^{\mu} = \Gamma(\mu+\beta+1)\sum_{n=0}^{\infty}\frac{(-1)^n(2n+\gamma)\Gamma(n+\gamma)(-\mu)_n}{\Gamma(n+\beta+1)\Gamma(n+\gamma+\mu+1)}R_n^{(\alpha,\beta)}(z), \quad 0<z<1$$

válida para

$$\alpha,\ \beta > -1, \quad -Re(\mu) < \min(1+\beta, 3/4+\beta/2),$$

onde $R_n^{(\alpha,\beta)}(z)$ *representa o polinómio de Jacobi de grau n definido no intervalo* $[0,1]$, *com peso* $w(z) = (1-z)^{\alpha}z^{\beta}$ *e* $\gamma = \alpha+\beta+1$.

Tomando $\alpha = \beta = 1/2$ *e* $\mu = -1/2$, *obtemos*

$$f(z) = \frac{1}{\sqrt{z}} = \frac{16}{\pi}\sum_{n=0}^{\infty}(-1)^n\frac{n+1}{(2n+1)(2n+3)}U_n(z), \quad 0<z<1,$$

onde $U_n(z)$ *é o polinómio de Chebyshev de segunda espécie de grau n definido no intervalo* $[0,1]$, *normalizado pela condição* $U_n(1) = n+1$. *Estes polinómios satisfazem a relação de recorrência [11]*

$$U_{n+1}(z) = 2(2z-1)U_n(z) - U_{n-1}(z).$$

Assim temos

$$\left\{\alpha_i = \frac{1}{4}, \beta_i = \frac{1}{2}, \gamma_i = \frac{1}{4}\right\}_{i\leq 0}$$

e

$$\left\{f_i = \frac{16}{\pi}(-1)^i\frac{i+1}{(2i+1)(2i+3)}\right\}_{i\leq 0}.$$

Exemplo 2.3 *Considere-se a série [10]*

$$f(z) = \frac{1}{2}e^{z/2} = \sum_{i\geq 0}(-1)^i L_i^{(\alpha)}(z), \quad 0<z<\infty.$$

Os polinómios de Laguerre $L_i^{(\alpha)}(z)$, *normalizados com coeficiente principal* $k_i = \frac{(-1)^i}{i!}$, *satisfazem a relação de recorrência*

$$\begin{cases} L_{i+1}^{(\alpha)}(z) = \frac{2i+\alpha+1-z}{i+1} L_i^{(\alpha)}(z) - \frac{i+\alpha}{i+1} L_{i-1}^{(\alpha)}(z), & i \geq 1 \\ L_0^{(\alpha)}(z) = 1, \quad L_1^{(\alpha)}(z) = 1 + \alpha - z \end{cases}.$$

Considerando $\alpha = 0$, *temos*

$$\{\alpha_i = -(i+1), \beta_i = 2i+1, \gamma_i = -i\}_{i \leq 0}$$

e

$$\{f_i = (-1)^i\}_{i \leq 0}.$$

Exemplo 2.4 *Considere-se a série [10]*

$$f(z) = e^{2z-1} = \sum_{i \geq 0} \frac{1}{i!} H_i(z), \quad 0 < z < 2.$$

Os polinómios de Hermite $H_i(z)$, *normalizados com coeficiente principal* $k_i = 2^i$, *satisfazem a relação de recorrência*

$$H_{i+1}(z) = 2zH_i(z) - 2iH_{i-1}(z).$$

Assim temos

$$\left\{\alpha_i = \frac{1}{2}, \beta_i = 0, \gamma_i = i\right\}_{i \leq 0}$$

e

$$\left\{f_i = \frac{1}{i!}\right\}_{i \leq 0}.$$

Exemplo 2.5 *Considere-se a série [15, 16]*

$$f(z) = \frac{1}{\sqrt{1-z}} e^{\frac{1}{1+\sqrt{1-z}}} = \sum_{i \geq 0} \frac{1}{2^i i!} B_i(z), \quad -\infty < z < +\infty.$$

Os polinómios mónicos de Bessel $B_i(z)$, *satisfazem a relação de recorrência*

$$\begin{cases} B_{i+1}(z) = (z - \beta_i) B_i(z) + \gamma_i B_{i-1}(z), & i \geq 1 \\ B_0(z) = 1, \quad B_1(z) = z - \beta_0 \end{cases}.$$

Assim temos

$$\beta_0 = -1, \left\{\beta_i = 0, \gamma_i = -\frac{1}{(2i-1)(2i+1)}\right\}_{i \leq 1}$$

e

$$\left\{f_i = \frac{1}{2^i i!}\right\}_{i \leq 0}.$$

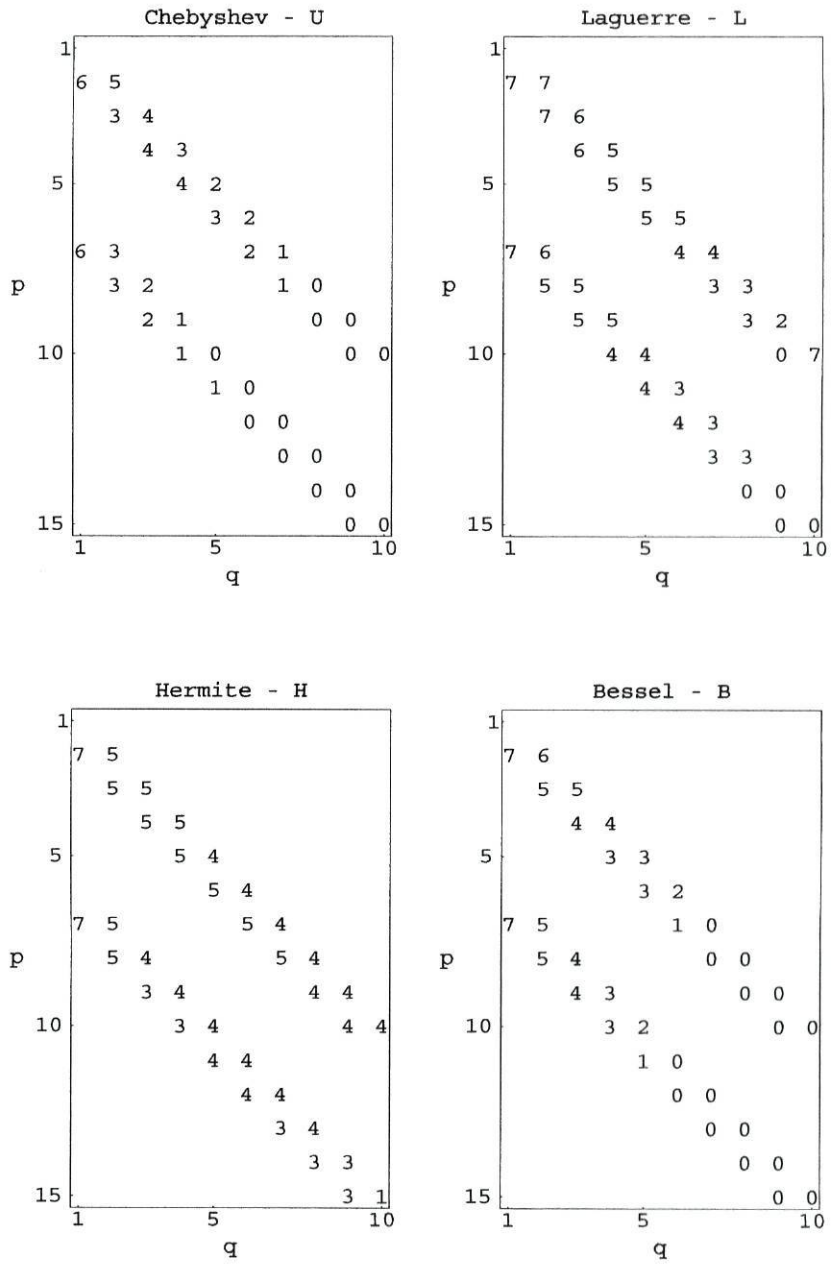


Tabela A.1: Para cada série, na posição (p, q), figura o número de algarismos significativos correctamente calculados do parâmetro ρ, calculados em Fortran90 em precisão simples, utilizando a biblioteca CADNA

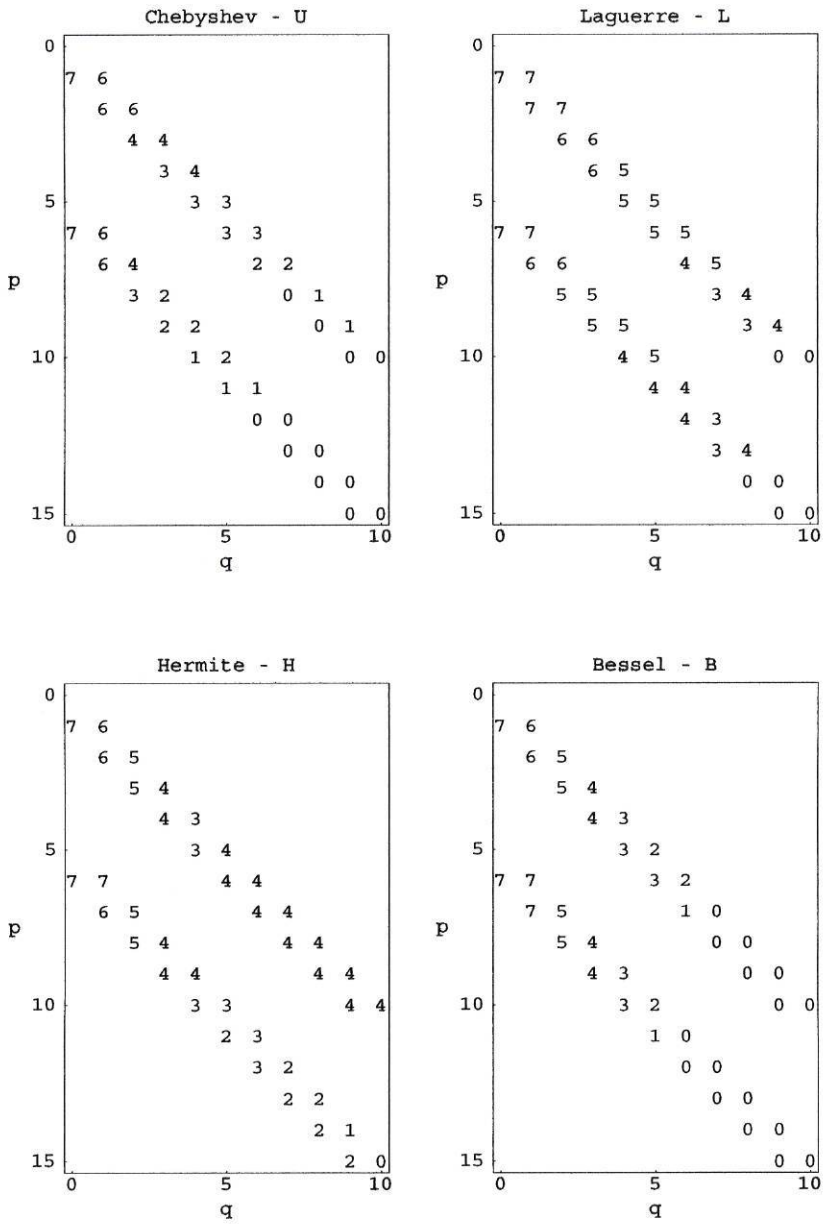


Tabela A.2: Para cada série, na posição (p, q), figura o número de algarismos significativos correctamente calculados do coeficiente de menor precisão do numerador, calculados em Fortran90 em precisão simples, utilizando a biblioteca CADNA

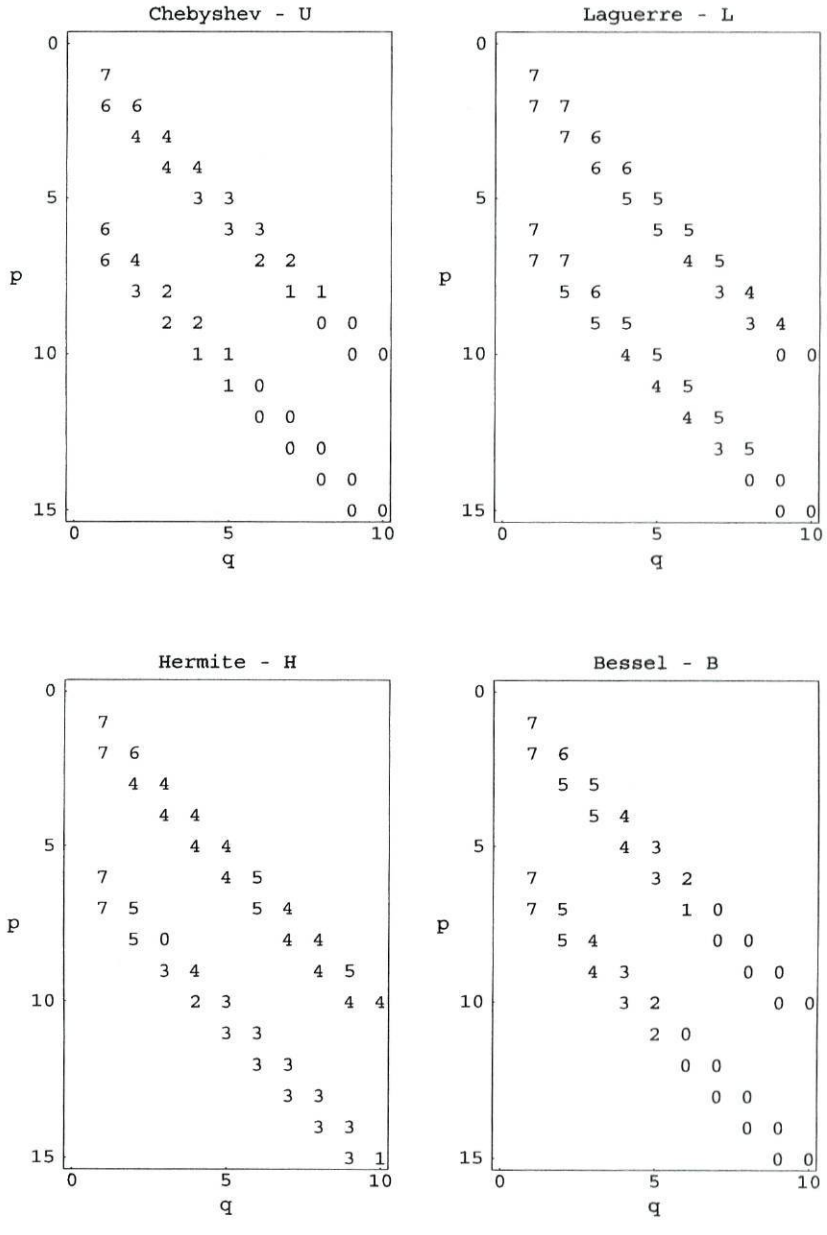


Tabela A.3: Para cada série, na posição (p, q), figura o número de algarismos significativos correctamente calculados do coeficiente de menor precisão do denominador, calculados em Fortran90 em precisão simples, utilizando a biblioteca CADNA

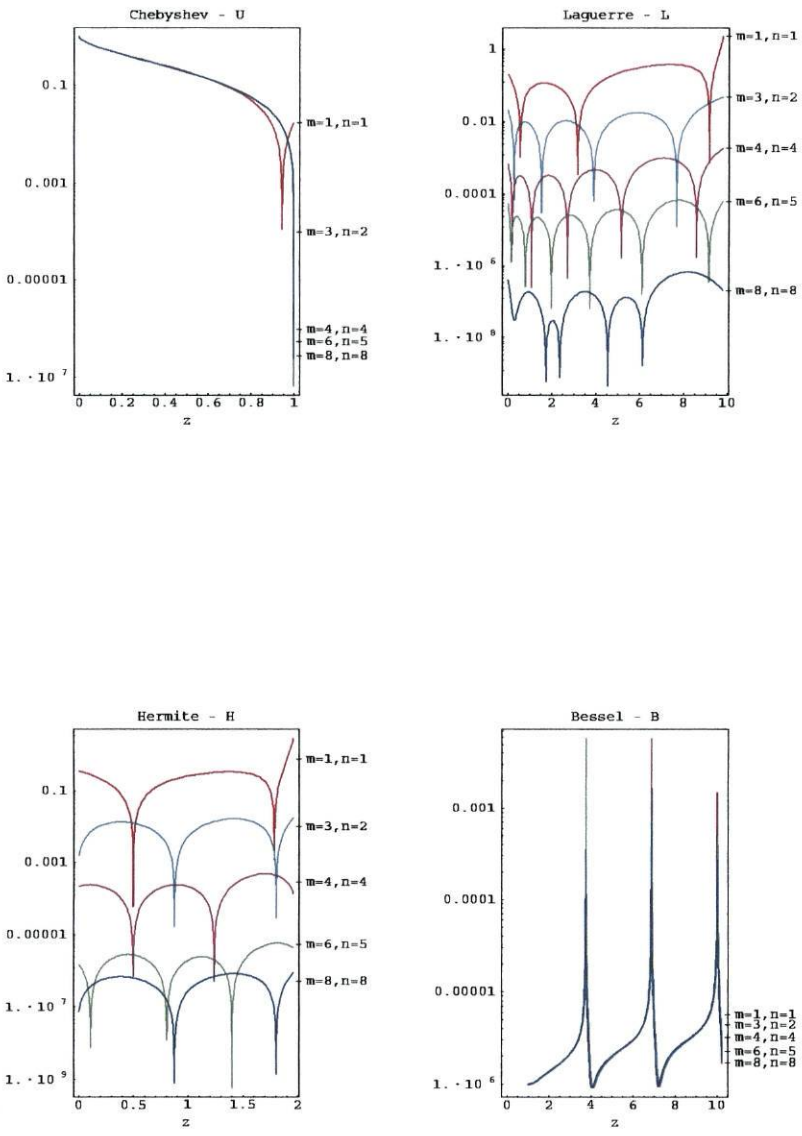


Figura A.1: Erros relativos, em escala logarítmica, para uma amostra de aproximantes $[m/n]_f^P(z)$ construídos a partir dos resultados do novo algoritmo programado em Fortran90 com *precisão dupla* e utilizando o CADNA, $\left|\frac{f(z)-[m/n]_f^U(z)}{f(z)}\right|$

(p,q)	$\left\lvert\frac{\rho-\rho^{LU}}{\rho}\right\rvert$	$\left\lvert\frac{\rho-\rho^{F}}{\rho}\right\rvert$	(p,q)	$\left\lvert\frac{\rho-\rho^{LU}}{\rho}\right\rvert$	$\left\lvert\frac{\rho-\rho^{F}}{\rho}\right\rvert$
$(2,1)$	0	0	$(2,2)$	$\mathbf{2.4\times10^{-6}}$	2.5×10^{-6}
$(3,2)$	$\mathbf{1.2\times10^{-5}}$	1.3×10^{-5}	$(3,3)$	1.8×10^{-4}	1.8×10^{-4}
$(4,3)$	$\mathbf{3.1\times10^{-4}}$	3.5×10^{-4}	$(4,4)$	$\mathbf{7.4\times10^{-4}}$	1.4×10^{-3}
$(5,4)$	$\mathbf{6.5\times10^{-3}}$	8.8×10^{-3}	$(5,5)$	$\mathbf{2.9\times10^{-1}}$	3.2×10^{-1}
$(6,5)$	$\mathbf{9.5\times10^{-1}}$	9.6×10^{1}	$(6,6)$	1.6	1.6
$(7,6)$	$\mathbf{3.7\times10^{-1}}$	3.8×10^{-1}	$(7,7)$	$\mathbf{1.8\times10^{-1}}$	2.7×10^{-1}
$(8,7)$	7.7×10^{-1}	7.1×10^{-1}	$(8,8)$	**1.5**	2.8
$(9,8)$	2.1×10^{1}	1.4	$(9,9)$	**1.1**	1.4
$(10,9)$	**2.2**	7.1	$(10,10)$	1.7	1.6
$(11,10)$	**2.0**	2.3	$(11,11)$	4.3	2.5
$(12,11)$	5.9×10^{-1}	1.6×10^{-1}	$(12,12)$	3.1	1.7
$(13,12)$	**3.3**	4.4	$(13,13)$	1.1	1.1
$(14,13)$	**8.4**	8.7	$(14,14)$	1.0	1.0
$(15,14)$	**2.9**	3.0	$(15,15)$	**1.1**	1.2
$(16,15)$	6.1	5.5	$(16,16)$	1.0	1.0
$(17,16)$	3.2×10^{-1}	1.9×10^{-1}	$(17,17)$	$\mathbf{9.2\times10^{-1}}$	9.4×10^{-1}
$(18,17)$	$\mathbf{8.3\times10^{-1}}$	1.3	$(18,18)$	9.3×10^{-1}	9.0×10^{-1}
$(19,18)$	$\mathbf{1.7\times10^{-1}}$	3.7×10^{1}	$(19,19)$	9.9×10^{-1}	9.9×10^{-1}
$(20,19)$	4.4	3.7	$(20,20)$	1.9	9.7×10^{-1}

Tabela A.4: $\left\lvert\frac{\rho-\rho^{*}}{\rho}\right\rvert$ calculado no Mathematica, onde ρ é calculado exactamente usando cálculo formal no Mathematica e ρ^{*} corresponde simultaneamente ao ρ^{LU} calculado com o algoritmo modificado e ao ρ^{F} calculado com o novo algoritmo, ambos programados em Fortran90 com *precisão simples*, relativamente à série de Chebyshev de 2^{a} espécie

(p,q)	$\left\|\left(\frac{a_i-a_i^{LU}}{a_i}\right)_{i=0}^{p}\right\|_\infty$	$\left\|\left(\frac{a_i-a_i^{F}}{a_i}\right)_{i=0}^{p}\right\|_\infty$	$\left\|\left(\frac{b_i-b_i^{LU}}{b_i}\right)_{i=0}^{q}\right\|_\infty$	$\left\|\left(\frac{b_i-b_i^{F}}{b_i}\right)_{i=0}^{q}\right\|_\infty$
$(1,0)$	0	0	0	0
$(1,1)$	1.0×10^{-7}	1.0×10^{-7}	0	0
$(2,1)$	4.1×10^{-7}	4.1×10^{-7}	0	0
$(2,2)$	2.0×10^{-6}	2.0×10^{-6}	1.8×10^{-6}	1.8×10^{-6}
$(3,2)$	$\mathbf{3.1 \times 10^{-5}}$	3.2×10^{-5}	$\mathbf{7.1 \times 10^{-6}}$	7.3×10^{-6}
$(3,3)$	1.3×10^{-5}	1.4×10^{-4}	1.3×10^{-4}	1.3×10^{-4}
$(4,3)$	$\mathbf{4.6 \times 10^{-4}}$	5.3×10^{-4}	$\mathbf{1.1 \times 10^{-4}}$	1.4×10^{-4}
$(4,4)$	$\mathbf{5.2 \times 10^{-4}}$	1.0×10^{-3}	$\mathbf{5.2 \times 10^{-4}}$	1.0×10^{-3}
$(5,4)$	$\mathbf{6.7 \times 10^{-3}}$	9.0×10^{-3}	$\mathbf{4.3 \times 10^{-3}}$	5.6×10^{-3}
$(5,5)$	$\mathbf{2.3 \times 10^{-1}}$	2.4×10^{-1}	$\mathbf{2.2 \times 10^{-1}}$	2.4×10^{-1}
$(6,5)$	$\mathbf{7.1 \times 10^{-1}}$	7.2×10^{-1}	$\mathbf{6.3 \times 10^{-1}}$	6.4×10^{-1}
$(6,6)$	**1.1**	1.2	1.1	1.1
$(7,6)$	1.1	1.1	1.1	1.1
$(7,7)$	1.1	1.1	1.1	1.1
$(8,7)$	1.9	1.8	1.0	1.0
$(8,8)$	**1.0**	1.1	**1.0**	1.1
$(9,8)$	2.7×10^{1}	1.1	1.5	1.0
$(9,9)$	1.0	1.0	1.0	1.0
$(10,9)$	3.7	3.4	1.0	1.0
$(10,10)$	1.4	1.3	1.0	1.0
$(11,10)$	6.7	6.7	1.0	1.0
$(11,11)$	4.0	2.6	1.4	1.1
$(12,11)$	8.0	6.3	1.0	1.0
$(12,12)$	1.2	1.2	1.0	1.0
$(13,12)$	1.6×10^{1}	1.5×10^{1}	1.0	1.0
$(13,13)$	1.5	1.5	1.0	1.0
$(14,13)$	5.2×10^{1}	5.0×10^{1}	1.0	1.0
$(14,14)$	1.8	1.7	1.0	1.0
$(15,14)$	1.6×10^{1}	1.5×10^{1}	1.0	1.0
$(15,15)$	2.1	2.1	1.0	1.0
$(16,15)$	3.9×10^{1}	3.3×10^{1}	1.0	1.0
$(16,16)$	2.4	2.4	1.0	1.0
$(17,16)$	3.6×10^{1}	3.5×10^{1}	1.0	1.0
$(17,17)$	**2.4**	2.5	1.0	1.0
$(18,17)$	3.1×10^{1}	2.6×10^{1}	1.0	1.0
$(18,18)$	2.5	2.5	1.0	1.0
$(19,18)$	$\mathbf{1.2 \times 10^{2}}$	2.0×10^{2}	**1.0**	1.1
$(19,19)$	2.6	2.6	1.0	1.0
$(20,19)$	$\mathbf{3.2 \times 10^{1}}$	7.8×10^{1}	1.0	1.0
$(20,20)$	4.3	2.7	1.0	1.0

Tabela A.5: $\left\|\left(\frac{s_i-s_i^*}{s_i}\right)_{i=0}^{k}\right\|_\infty$ calculado no Mathematica, onde k toma o valor p ou q, s_i representa simultaneamente os coeficientes a_i e b_i calculados exactamente usando cálculo formal no Mathematica e s_i^* corresponde simultaneamente aos coeficientes a_i^{LU} e b_i^{LU} calculados com o algoritmo modificado ou aos a_i^F e b_i^F calculados com o novo algoritmo, ambos programados em Fortran90 com *precisão simples*, relativamente à série de Chebyshev de 2^a espécie

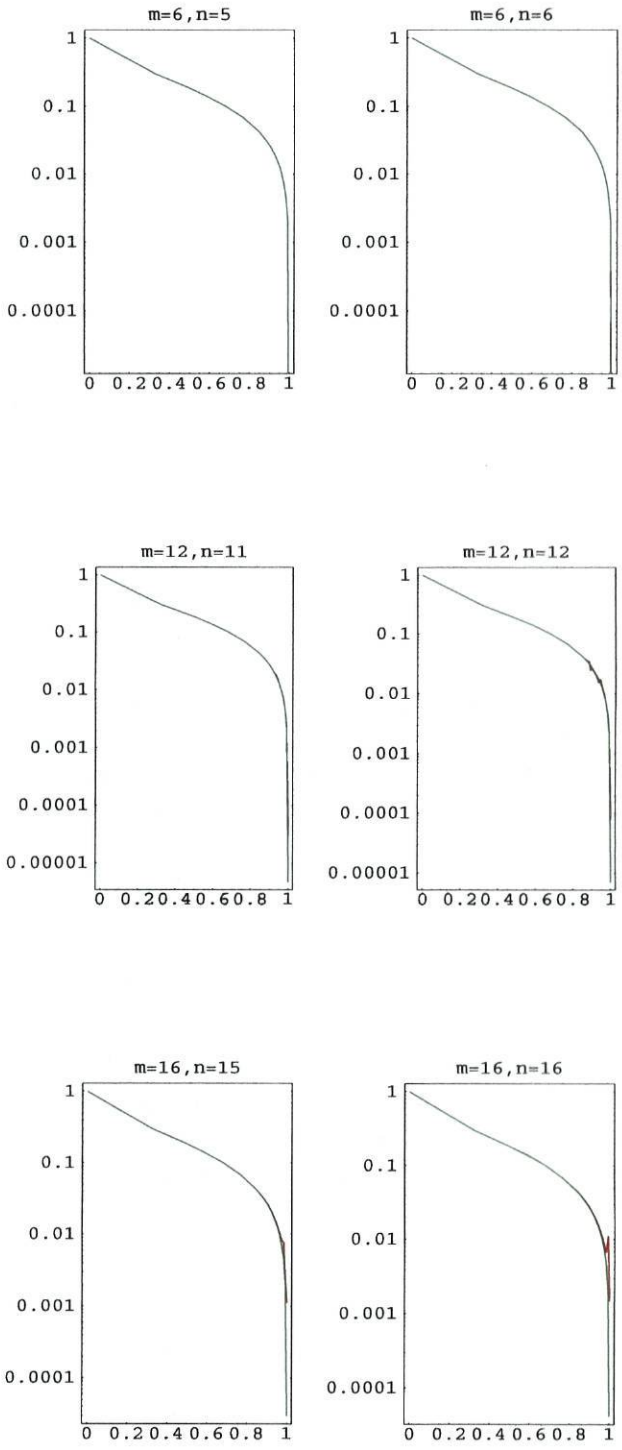


Figura A.2: Erros relativos, em escala logarítmica, para uma amostra de aproximantes $[m/n]_f^U(z)$ calculados com a implementação directa do novo algoritmo, $\left|\frac{f(z)-[p/q]_f^U(z)}{f(z)}\right|$ a vermelho, e de aproximantes $\overline{[m/n]}_f^U(z)$ homólogos calculados com o algoritmo modificado, $\left|\frac{f(z)-\overline{[p/q]}_f^U(z)}{f(z)}\right|$ a verde, utilizando Fortran90 em precisão simples, relativamente à série de Chebyshev de 2^a espécie e considerando $f(z) = 1/\sqrt{z}$

(p,q)	$\left\lvert\frac{\rho-\rho^{LU}}{\rho}\right\rvert$	$\left\lvert\frac{\rho-\rho^{F}}{\rho}\right\rvert$	(p,q)	$\left\lvert\frac{\rho-\rho^{LU}}{\rho}\right\rvert$	$\left\lvert\frac{\rho-\rho^{F}}{\rho}\right\rvert$
$(2,1)$	0	0	$(2,2)$	0	0
$(3,2)$	0	0	$(3,3)$	2.8×10^{-7}	1.0×10^{-7}
$(4,3)$	2.7×10^{-6}	7.3×10^{-7}	$(4,4)$	$\mathbf{3.7 \times 10^{-6}}$	4.3×10^{-6}
$(5,4)$	$\mathbf{7.7 \times 10^{-6}}$	1.3×10^{-5}	$(5,5)$	8.9×10^{-5}	3.3×10^{-5}
$(6,5)$	2.5×10^{-4}	1.8×10^{-4}	$(6,6)$	$\mathbf{2.1 \times 10^{-5}}$	1.3×10^{-4}
$(7,6)$	1.1×10^{-3}	3.3×10^{-4}	$(7,7)$	1.7×10^{-2}	1.3×10^{-3}
$(8,7)$	9.0×10^{-2}	8.4×10^{-4}	$(8,8)$	6.0×10^{-1}	6.2×10^{-3}
$(9,8)$	8.7×10^{-1}	4.4×10^{-3}	$(9,9)$	1.5	1.4
$(10,9)$	6.6	1.1	$(10,10)$	$\mathbf{6.8 \times 10^{-1}}$	1.6
$(11,10)$	9.7×10^{-1}	5.5×10^{-1}	$(11,11)$	$\mathbf{5.7}$	1.0×10^{1}
$(12,11)$	$\mathbf{2.1}$	1.6×10^{1}	$(12,12)$	1.8	7.6×10^{-1}
$(13,12)$	1.8×10^{1}	1.8×10^{1}	$(13,13)$	1.0	2.2×10^{-1}
$(14,13)$	7.9×10^{-1}	4.7×10^{-1}	$(14,14)$	2.1×10^{1}	1.1
$(15,14)$	1.1	7.3×10^{-1}	$(15,15)$	9.8	2.8
$(16,15)$	1.3	2.1×10^{1}	$(16,16)$	2.4	8.4×10^{-1}
$(17,16)$	$\mathbf{2.2 \times 10^{-1}}$	8.1×10^{-1}	$(17,17)$	$\mathbf{4.3 \times 10^{-1}}$	4.7×10^{-1}
$(18,17)$	$\mathbf{1.1}$	1.4	$(18,18)$	$\mathbf{1.3}$	1.8
$(19,18)$	$\mathbf{7.7 \times 10^{-1}}$	8.0×10^{-1}	$(19,19)$	$\mathbf{5.0}$	1.2×10^{1}
$(20,19)$	2.6	3.6×10^{-1}	$(20,20)$	$\mathbf{3.0}$	7.0

Tabela A.6: $\left\lvert\frac{\rho-\rho^{*}}{\rho}\right\rvert$ calculado no Mathematica, onde ρ é calculado exactamente usando cálculo formal no Mathematica e ρ^{*} corresponde simultaneamente ao ρ^{LU} calculado com o algoritmo modificado e ao ρ^{F} calculado com o novo algoritmo, ambos programados em Fortran90 com *precisão simples*, relativamente à série de Laguerre

(p,q)	$\left\|\left(\frac{a_i-a_i^{LU}}{a_i}\right)^p_{i=0}\right\|_\infty$	$\left\|\left(\frac{a_i-a_i^{F}}{a_i}\right)^p_{i=0}\right\|_\infty$	$\left\|\left(\frac{b_i-b_i^{LU}}{b_i}\right)^q_{i=0}\right\|_\infty$	$\left\|\left(\frac{b_i-b_i^{F}}{b_i}\right)^q_{i=0}\right\|_\infty$
$(1,0)$	0	0	0	0
$(1,1)$	0	0	0	0
$(2,1)$	0	0	0	0
$(2,2)$	0	0	0	0
$(3,2)$	0	0	0	0
$(3,3)$	3.6×10^{-7}	1.2×10^{-7}	1.0×10^{-7}	1.0×10^{-7}
$(4,3)$	3.7×10^{-6}	1.1×10^{-6}	1.2×10^{-6}	4.8×10^{-7}
$(4,4)$	9.4×10^{-6}	6.7×10^{-6}	$\mathbf{2.6 \times 10^{-6}}$	2.9×10^{-6}
$(5,4)$	1.6×10^{-5}	1.6×10^{-5}	1.1×10^{-5}	1.6×10^{-6}
$(5,5)$	1.3×10^{-4}	2.5×10^{-5}	5.9×10^{-5}	2.0×10^{-5}
$(6,5)$	3.0×10^{-4}	2.5×10^{-4}	$\mathbf{7.4 \times 10^{-5}}$	8.5×10^{-5}
$(6,6)$	$\mathbf{4.4 \times 10^{-4}}$	4.9×10^{-4}	$\mathbf{2.2 \times 10^{-5}}$	1.1×10^{-4}
$(7,6)$	1.7×10^{-3}	7.4×10^{-4}	6.8×10^{-4}	4.2×10^{-4}
$(7,7)$	2.3×10^{-2}	2.4×10^{-3}	1.0×10^{-3}	1.0×10^{-3}
$(8,7)$	1.1×10^{-1}	1.1×10^{-3}	4.6×10^{-3}	9.6×10^{-4}
$(8,8)$	9.9×10^{-1}	8.8×10^{-3}	3.9×10^{-2}	2.3×10^{-3}
$(9,8)$	1.1	1.6×10^{-2}	3.4×10^{-1}	1.6×10^{-2}
$(9,9)$	1.9	1.8	$\mathbf{5.4 \times 10^{-1}}$	8.7×10^{-1}
$(10,9)$	$\mathbf{8.2 \times 10^{-1}}$	1.4	7.5×10^{-1}	4.7×10^{-1}
$(10,10)$	**1.0**	1.5	$\mathbf{8.8 \times 10^{-1}}$	1.2
$(11,10)$	1.2	1.1	$\mathbf{8.3 \times 10^{-1}}$	1.1
$(11,11)$	1.5	2.4	1.7	1.1
$(12,11)$	**1.8**	3.3	1.0	8.3×10^{-1}
$(12,12)$	**1.0**	1.3	1.0	9.9×10^{-1}
$(13,12)$	2.0×10^{1}	1.0	1.0	1.0
$(13,13)$	1.0	1.0	1.0	1.0
$(14,13)$	1.0	1.0	1.0	1.0
$(14,14)$	1.0	1.0	1.0	1.0
$(15,14)$	1.0	1.0	1.0	1.0
$(15,15)$	1.0	1.0	1.0	1.0
$(16,15)$	**1.0**	1.1	1.0	1.0
$(16,16)$	1.0	1.0	1.0	1.0
$(17,16)$	1.0	1.0	1.0	1.0
$(17,17)$	1.0	1.0	1.0	1.0
$(18,17)$	1.0	1.0	1.0	1.0
$(18,18)$	1.0	1.0	1.0	1.0
$(19,18)$	1.0	1.0	1.0	1.0
$(19,19)$	1.0	1.0	1.0	1.0
$(20,19)$	1.0	1.0	1.0	1.0
$(20,20)$	1.0	1.0	1.0	1.0

Tabela A.7: $\left\|\left(\frac{s_i-s_i^*}{s_i}\right)^k_{i=0}\right\|_\infty$ calculado no Mathematica, onde k toma o valor p ou q, s_i representa simultaneamente os coeficientes a_i e b_i calculados exactamente usando cálculo formal no Mathematica e s_i^* corresponde simultaneamente aos coeficientes a_i^{LU} e b_i^{LU} calculados com o algoritmo modificado ou aos a_i^F e b_i^F calculados com o novo algoritmo, ambos programados em Fortran90 com *precisão simples*, relativamente à série de Laguerre

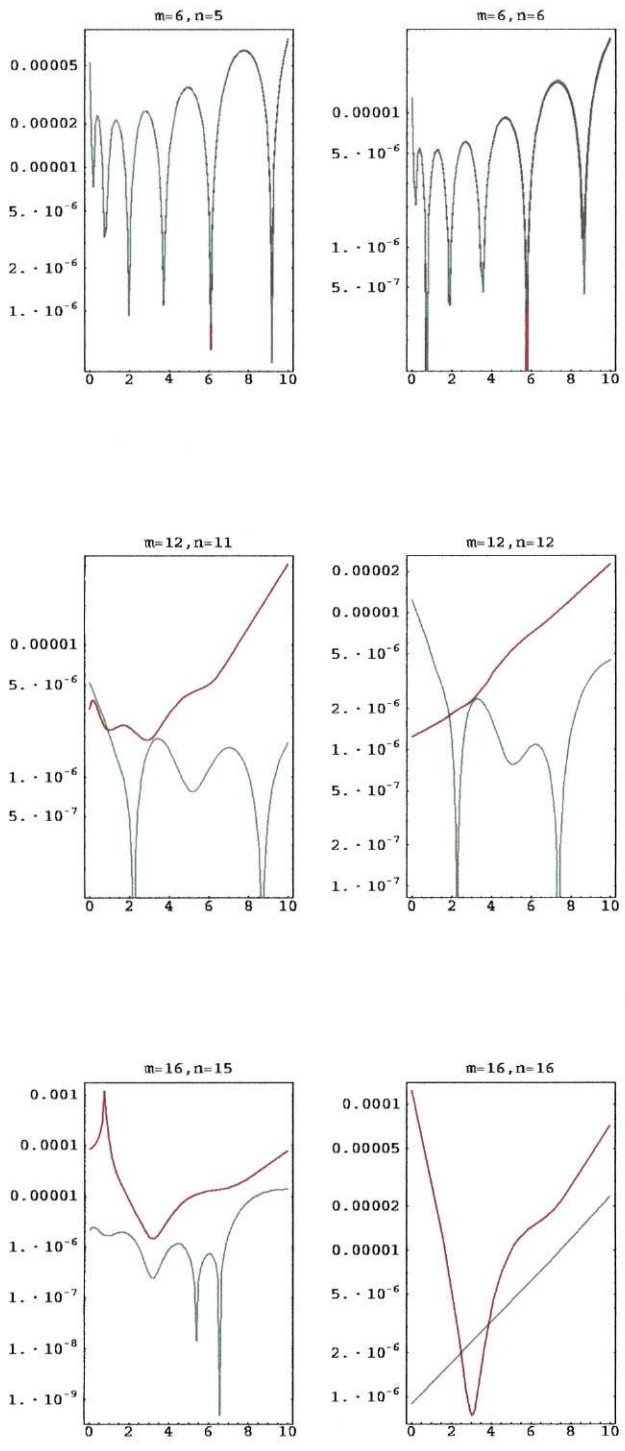


Figura A.3: Erros relativos, em escala logarítmica, para uma amostra de aproximantes $[m/n]_f^L(z)$ calculados com a implementação directa do novo algoritmo, $\left|\frac{f(z)-[p/q]_f^L(z)}{f(z)}\right|$ a vermelho, e de aproximantes $\overline{[m/n]}_f^L(z)$ homólogos calculados com o algoritmo modificado, $\left|\frac{f(z)-\overline{[p/q]}_f^L(z)}{f(z)}\right|$ a verde, utilizando Fortran90 em precisão simples, relativamente à série de Laguerre e considerando $f(z) = \frac{e^{z/2}}{2}$

(p,q)	$\left\lvert\frac{\rho-\rho^{LU}}{\rho}\right\rvert$	$\left\lvert\frac{\rho-\rho^{F}}{\rho}\right\rvert$	(p,q)	$\left\lvert\frac{\rho-\rho^{LU}}{\rho}\right\rvert$	$\left\lvert\frac{\rho-\rho^{F}}{\rho}\right\rvert$
$(2,1)$	0	0	$(2,2)$	9.5×10^{-7}	9.5×10^{-7}
$(3,2)$	1.3×10^{-6}	1.2×10^{-6}	$(3,3)$	$\mathbf{6.3\times10^{-7}}$	8.7×10^{-7}
$(4,3)$	1.1×10^{-5}	1.1×10^{-5}	$(4,4)$	1.8×10^{-4}	1.7×10^{-4}
$(5,4)$	5.3×10^{-4}	4.8×10^{-4}	$(5,5)$	2.9×10^{-3}	2.6×10^{-3}
$(6,5)$	6.1×10^{-3}	5.2×10^{-3}	$(6,6)$	2.0×10^{-2}	1.5×10^{-2}
$(7,6)$	2.9×10^{-2}	1.5×10^{-2}	$(7,7)$	$\mathbf{1.1\times10^{-2}}$	3.5×10^{-2}
$(8,7)$	$\mathbf{2.1\times10^{-1}}$	3.3×10^{-1}	$(8,8)$	6.3	4.7
$(9,8)$	1.1	1.1	$(9,9)$	3.2	2.2
$(10,9)$	2.1	1.7	$(10,10)$	3.1	2.9
$(11,10)$	1.0	1.0	$(11,11)$	1.2	8.7×10^{-1}
$(12,11)$	4.0×10^{-1}	7.7×10^{-1}	$(12,12)$	3.5	3.2
$(13,12)$	1.1	1.1	$(13,13)$	$\mathbf{1.0}$	3.5
$(14,13)$	9.4	2.5	$(14,14)$	1.2	1.9×10^{-1}
$(15,14)$	2.1	3.7×10^{-1}	$(15,15)$	$\mathbf{8.8\times10^{-1}}$	9.1×10^{-1}
$(16,15)$	4.5×10^{-1}	3.3×10^{-1}	$(16,16)$	5.6×10^{-1}	1.6×10^{-1}
$(17,16)$	$\mathbf{6.3\times10^{-1}}$	3.1×10^{2}	$(17,17)$	$\mathbf{5.5\times10^{-1}}$	8.9×10^{-1}
$(18,17)$	1.8	4.3×10^{-1}	$(18,18)$	1.0	2.8×10^{-2}
$(19,18)$	1.3	9.0×10^{-1}	$(19,19)$	8.7×10^{-1}	7.6×10^{-1}
$(20,19)$	9.8×10^{-1}	9.0×10^{-1}	$(20,20)$	$\mathbf{9.9\times10^{-1}}$	1.2

Tabela A.8: $\left\lvert\frac{\rho-\rho^{*}}{\rho}\right\rvert$ calculado no Mathematica, onde ρ é calculado exactamente usando cálculo formal no Mathematica e ρ^{*} corresponde simultaneamente ao ρ^{LU} calculado com o algoritmo modificado e ao ρ^{F} calculado com o novo algoritmo, ambos programados em Fortran90 com *precisão simples*, relativamente à série de Hermite

(p,q)	$\left\|\left(\frac{a_i-a_i^{LU}}{a_i}\right)_{i=0}^{p}\right\|_\infty$	$\left\|\left(\frac{a_i-a_i^{F}}{a_i}\right)_{i=0}^{p}\right\|_\infty$	$\left\|\left(\frac{b_i-b_i^{LU}}{b_i}\right)_{i=0}^{q}\right\|_\infty$	$\left\|\left(\frac{b_i-b_i^{F}}{b_i}\right)_{i=0}^{q}\right\|_\infty$
$(1,0)$	0	0	0	0
$(1,1)$	0	0	0	0
$(2,1)$	1.5×10^{-7}	1.2×10^{-7}	0	0
$(2,2)$	4.4×10^{-7}	2.0×10^{-7}	1.2×10^{-7}	0
$(3,2)$	1.4×10^{-6}	1.4×10^{-6}	5.9×10^{-7}	5.9×10^{-7}
$(3,3)$	1.2×10^{-6}	9.6×10^{-6}	5.1×10^{-6}	3.9×10^{-6}
$(4,3)$	3.6×10^{-5}	2.6×10^{-5}	1.0×10^{-5}	7.1×10^{-6}
$(4,4)$	2.1×10^{-4}	1.4×10^{-4}	6.9×10^{-5}	4.7×10^{-5}
$(5,4)$	2.4×10^{-3}	1.1×10^{-3}	4.7×10^{-5}	1.0×10^{-5}
$(5,5)$	1.1×10^{-3}	2.1×10^{-4}	2.5×10^{-4}	6.5×10^{-5}
$(6,5)$	$\mathbf{6.2 \times 10^{-4}}$	2.4×10^{-3}	$\mathbf{3.6 \times 10^{-4}}$	9.3×10^{-4}
$(6,6)$	$\mathbf{5.5 \times 10^{-3}}$	1.6×10^{-2}	$\mathbf{3.1 \times 10^{-3}}$	6.6×10^{-3}
$(7,6)$	$\mathbf{2.7 \times 10^{-2}}$	3.9×10^{-2}	$\mathbf{9.4 \times 10^{-3}}$	1.2×10^{-2}
$(7,7)$	$\mathbf{1.7 \times 10^{-1}}$	2.2×10^{-1}	$\mathbf{6.2 \times 10^{-2}}$	7.3×10^{-2}
$(8,7)$	4.3×10^{-1}	3.7×10^{-1}	1.4×10^{-1}	1.1×10^{-1}
$(8,8)$	8.9	4.4	3.2	1.5
$(9,8)$	2.5	2.3	8.7×10^{-1}	7.5×10^{-1}
$(9,9)$	**3.2**	3.6	1.3	1.2
$(10,9)$	8.0	6.8	1.7	1.7
$(10,10)$	$\mathbf{1.7 \times 10^{1}}$	1.9×10^{1}	1.2	1.2
$(11,10)$	4.9	4.5	1.0	1.0
$(11,11)$	3.9	3.1	1.0	1.0
$(12,11)$	3.3×10^{2}	9.5×10^{1}	4.0	9.6×10^{-1}
$(12,12)$	$\mathbf{8.7 \times 10^{1}}$	2.6×10^{2}	**1.4**	2.3
$(13,12)$	2.5×10^{1}	2.0×10^{1}	**1.2**	1.7
$(13,13)$	$\mathbf{5.9 \times 10^{1}}$	2.5×10^{2}	1.1	1.1
$(14,13)$	**2.8**	1.1×10^{2}	1.1	9.3×10^{-1}
$(14,14)$	$\mathbf{6.1 \times 10^{1}}$	1.6×10^{3}	1.0	9.3×10^{-1}
$(15,14)$	5.5×10^{2}	1.2×10^{2}	1.2	1.0
$(15,15)$	3.3×10^{2}	2.6×10^{2}	1.0	1.0
$(16,15)$	5.4×10^{2}	1.1×10^{2}	1.0	1.0
$(16,16)$	$\mathbf{1.8 \times 10^{2}}$	2.4×10^{2}	1.0	1.0
$(17,16)$	7.5×10^{2}	1.4×10^{2}	1.0	1.0
$(17,17)$	$\mathbf{9.7 \times 10^{1}}$	6.7×10^{2}	1.0	1.0
$(18,17)$	$\mathbf{1.4 \times 10^{2}}$	2.3×10^{2}	1.0	1.0
$(18,18)$	$\mathbf{6.1 \times 10^{1}}$	7.1×10^{2}	1.0	1.0
$(19,18)$	$\mathbf{1.1 \times 10^{2}}$	1.7×10^{2}	1.0	1.0
$(19,19)$	3.4×10^{1}	1.0×10^{1}	1.0	1.0
$(20,19)$	$\mathbf{7.6 \times 10^{1}}$	1.6×10^{3}	1.0	1.0
$(20,20)$	**9.0**	5.4×10^{1}	1.0	1.0

Tabela A.9: $\left\|\left(\frac{s_i-s_i^*}{s_i}\right)_{i=0}^{k}\right\|_\infty$ calculado no Mathematica, onde k toma o valor p ou q, s_i representa simultaneamente os coeficientes a_i e b_i calculados exactamente usando cálculo formal no Mathematica e s_i^* corresponde simultaneamente aos coeficientes a_i^{LU} e b_i^{LU} calculados com o algoritmo modificado ou aos a_i^F e b_i^F calculados com o novo algoritmo, ambos programados em Fortran90 com *precisão simples*, relativamente à série de Hermite

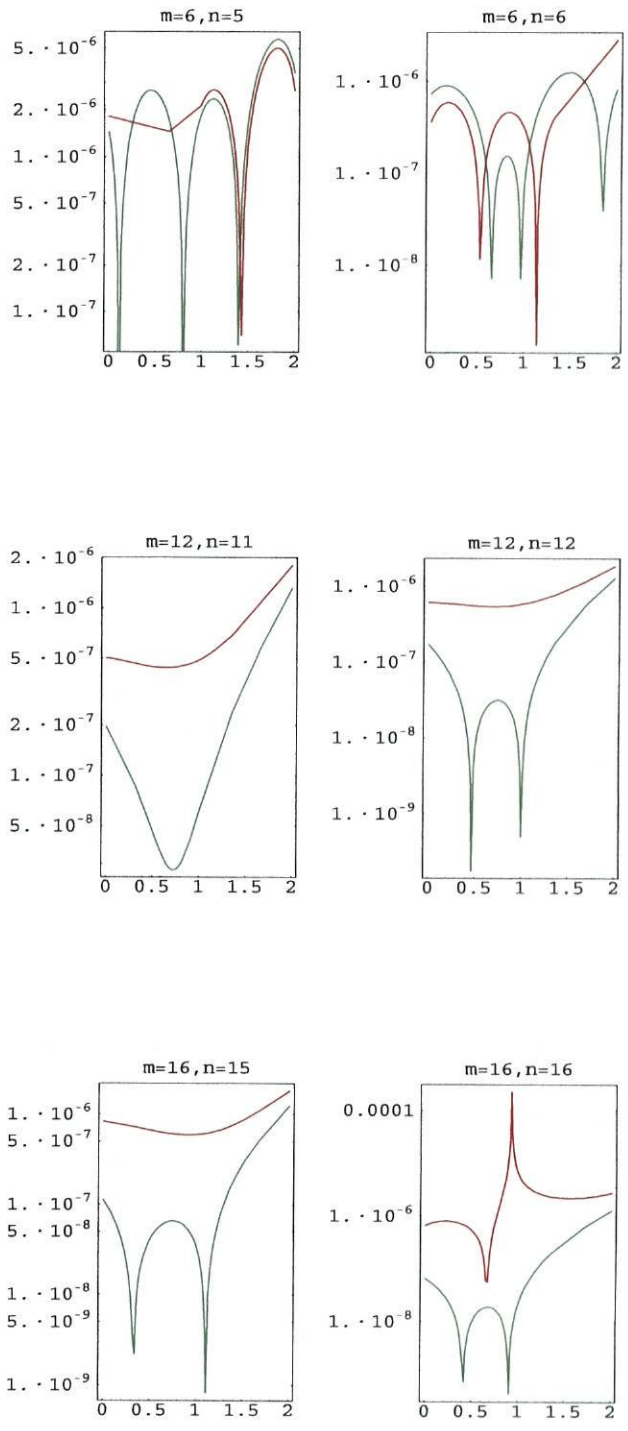


Figura A.4: Erros relativos, em escala logarítmica, para uma amostra de aproximantes $[m/n]_f^H(z)$ calculados com a implementação directa do novo algoritmo, $\left|\frac{f(z)-[p/q]_f^H(z)}{f(z)}\right|$ a vermelho, e de aproximantes $\overline{[m/n]}_f^H(z)$ homólogos calculados com o algoritmo modificado, $\left|\frac{f(z)-\overline{[p/q]}_f^H(z)}{f(z)}\right|$ a verde, utilizando Fortran90 em precisão simples, relativamente à série de Hermite e considerando $f(z) = e^{2z-1}$

(p,q)	$\left\lvert\frac{\rho-\rho^{LU}}{\rho}\right\rvert$	$\left\lvert\frac{\rho-\rho^{F}}{\rho}\right\rvert$	(p,q)	$\left\lvert\frac{\rho-\rho^{LU}}{\rho}\right\rvert$	$\left\lvert\frac{\rho-\rho^{F}}{\rho}\right\rvert$
$(2,1)$	0	0	$(2,2)$	0	0
$(3,2)$	$\mathbf{5.1 \times 10^{-7}}$	5.7×10^{-7}	$(3,3)$	4.7×10^{-6}	3.5×10^{-6}
$(4,3)$	1.6×10^{-5}	1.2×10^{-5}	$(4,4)$	6.1×10^{-5}	4.1×10^{-5}
$(5,4)$	1.2×10^{-4}	6.0×10^{-5}	$(5,5)$	2.1×10^{-4}	7.5×10^{-5}
$(6,5)$	1.1×10^{-4}	1.0×10^{-3}	$(6,6)$	$\mathbf{2.8 \times 10^{-3}}$	5.8×10^{-3}
$(7,6)$	$\mathbf{1.3 \times 10^{-2}}$	1.9×10^{-2}	$(7,7)$	$\mathbf{5.5 \times 10^{-2}}$	6.4×10^{-2}
$(8,7)$	2.2×10^{-1}	2.0×10^{-1}	$(8,8)$	3.5	2.1
$(9,8)$	9.7×10^{-1}	9.1×10^{-1}	$(9,9)$	2.8	4.2
$(10,9)$	1.7	1.7	$(10,10)$	**2.1**	2.5
$(11,10)$	$\mathbf{8.8 \times 10^{-1}}$	9.3×10^{-1}	$(11,11)$	$\mathbf{2.4 \times 10^{-1}}$	6.2×10^{-1}
$(12,11)$	2.7×10^{1}	7.9	$(12,12)$	$\mathbf{8.6 \times 10^{-1}}$	2.3
$(13,12)$	$\mathbf{4.4 \times 10^{-1}}$	4.7×10^{-1}	$(13,13)$	$\mathbf{4.6 \times 10^{-1}}$	1.0
$(14,13)$	$\mathbf{4.6 \times 10^{-1}}$	2.8×10^{1}	$(14,14)$	$\mathbf{9.9 \times 10^{-1}}$	7.0
$(15,14)$	2.8×10^{2}	9.9×10^{-1}	$(15,15)$	$\mathbf{6.5 \times 10^{-1}}$	6.4
$(16,15)$	9.6×10^{-1}	9.3×10^{-1}	$(16,16)$	1.1	1.1
$(17,16)$	1.2	1.1	$(17,17)$	1.2	5.5×10^{-1}
$(18,17)$	$\mathbf{5.7 \times 10^{-1}}$	1.3	$(18,18)$	$\mathbf{8.6 \times 10^{-1}}$	9.2×10^{-1}
$(19,18)$	$\mathbf{9.0 \times 10^{-2}}$	8.3×10^{-1}	$(19,19)$	**1.4**	3.1
$(20,19)$	$\mathbf{8.1 \times 10^{-1}}$	5.3	$(20,20)$	$\mathbf{7.3 \times 10^{-1}}$	1.0

Tabela A.10: $\left\lvert\frac{\rho-\rho^{*}}{\rho}\right\rvert$ calculado no Mathematica, onde ρ é calculado exactamente usando cálculo formal no Mathematica e ρ^{*} corresponde simultaneamente ao ρ^{LU} calculado com o algoritmo modificado e ao ρ^{F} calculado com o novo algoritmo, ambos programados em Fortran90 com *precisão simples*, relativamente à série de Bessel

(p,q)	$\left\|\left(\frac{a_i-a_i^{LU}}{a_i}\right)_{i=0}^{p}\right\|_\infty$	$\left\|\left(\frac{a_i-a_i^{F}}{a_i}\right)_{i=0}^{p}\right\|_\infty$	$\left\|\left(\frac{b_i-b_i^{LU}}{b_i}\right)_{i=0}^{q}\right\|_\infty$	$\left\|\left(\frac{b_i-b_i^{F}}{b_i}\right)_{i=0}^{q}\right\|_\infty$
$(1,0)$	1.2×10^{-8}	1.2×10^{-8}	0	0
$(1,1)$	2.1×10^{-8}	2.1×10^{-8}	2.0×10^{-8}	2.0×10^{-8}
$(2,1)$	$\mathbf{4.9 \times 10^{-7}}$	1.1×10^{-6}	1.3×10^{-7}	1.3×10^{-7}
$(2,2)$	$\mathbf{7.0 \times 10^{-7}}$	1.5×10^{-6}	7.0×10^{-7}	6.7×10^{-7}
$(3,2)$	5.5×10^{-6}	1.5×10^{-6}	9.2×10^{-7}	4.7×10^{-7}
$(3,3)$	$\mathbf{2.8 \times 10^{-5}}$	4.0×10^{-5}	$\mathbf{2.7 \times 10^{-5}}$	3.8×10^{-5}
$(4,3)$	$\mathbf{4.2 \times 10^{-4}}$	4.6×10^{-4}	$\mathbf{1.9 \times 10^{-4}}$	2.0×10^{-4}
$(4,4)$	3.2×10^{-3}	3.2×10^{-3}	3.1×10^{-3}	3.1×10^{-3}
$(5,4)$	2.0×10^{-2}	2.0×10^{-2}	1.2×10^{-2}	1.2×10^{-2}
$(5,5)$	1.8×10^{-1}	1.8×10^{-1}	1.8×10^{-1}	1.7×10^{-1}
$(6,5)$	9.3×10^{1}	7.0×10^{1}	7.8×10^{1}	5.9×10^{1}
$(6,6)$	9.9×10^{-1}	9.9×10^{-1}	9.8×10^{-1}	9.8×10^{-1}
$(7,6)$	1.3	1.3	1.1	1.1
$(7,7)$	1.5	1.4	1.5	1.4
$(8,7)$	2.7	2.7	1.0	1.0
$(8,8)$	1.0	1.0	1.0	1.0
$(9,8)$	3.9	3.9	1.0	1.0
$(9,9)$	1.0	1.0	1.0	1.0
$(10,9)$	**2.2**	2.4	1.0	1.0
$(10,10)$	1.0	1.0	1.0	1.0
$(11,10)$	5.7	5.7	1.0	1.0
$(11,11)$	1.0	1.0	1.0	1.0
$(12,11)$	1.0×10^{1}	1.0×10^{1}	1.0	1.0
$(12,12)$	1.0	1.0	1.0	1.0
$(13,12)$	1.4×10^{1}	1.4×10^{1}	1.0	1.0
$(13,13)$	1.9	1.9	1.0	1.0
$(14,13)$	1.6×10^{1}	1.6×10^{1}	1.0	1.0
$(14,14)$	**1.8**	1.9	1.0	1.0
$(15,14)$	$\mathbf{2.3 \times 10^{1}}$	2.4×10^{1}	1.0	1.0
$(15,15)$	2.0	2.0	1.0	1.0
$(16,15)$	$\mathbf{2.0 \times 10^{2}}$	5.6×10^{2}	**1.0**	1.9
$(16,16)$	2.2	2.2	1.0	1.0
$(17,16)$	1.3×10^{1}	1.3×10^{1}	1.0	1.0
$(17,17)$	2.4	2.4	1.0	1.0
$(18,17)$	$\mathbf{5.7 \times 10^{1}}$	4.1×10^{2}	**1.0**	1.3
$(18,18)$	**2.5**	2.6	1.0	1.0
$(19,18)$	7.6×10^{1}	5.0	1.0	1.0
$(19,19)$	9.9	3.7	1.0	1.0
$(20,19)$	$\mathbf{7.1 \times 10^{1}}$	8.1×10^{1}	1.0	1.0
$(20,20)$	**2.9**	3.0	1.0	1.0

Tabela A.11: $\left\|\left(\frac{s_i-s_i^*}{s_i}\right)_{i=0}^{k}\right\|_\infty$ calculado no Mathematica, onde k toma o valor p ou q, s_i representa simultaneamente os coeficientes a_i e b_i calculados exactamente usando cálculo formal no Mathematica e s_i^* corresponde simultaneamente aos coeficientes a_i^{LU} e b_i^{LU} calculados com o algoritmo modificado ou aos a_i^F e b_i^F calculados com o novo algoritmo, ambos programados em Fortran90 com *precisão simples*, relativamente à série de Bessel

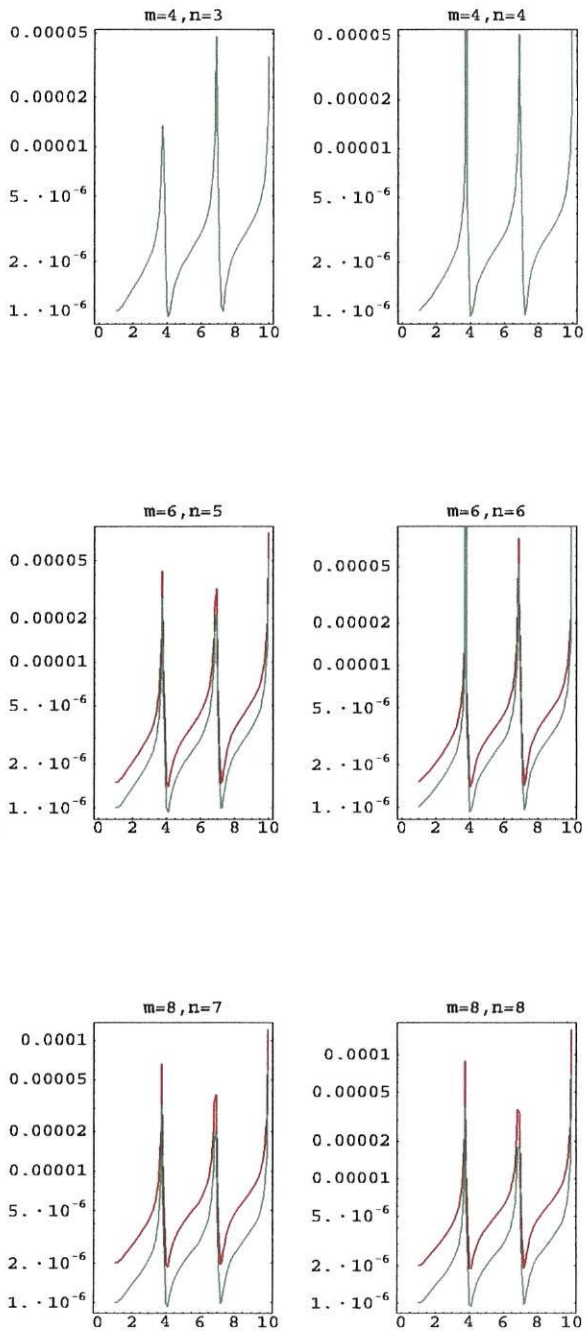


Figura A.5: Erros relativos, em escala logarítmica, para uma amostra de aproximantes $[m/n]_f^B(z)$ calculados com a implementação directa do novo algoritmo, $\left|\frac{f(z)-[p/q]_f^B(z)}{f(z)}\right|$ a vermelho, e de aproximantes $\overline{[m/n]}_f^B(z)$ homólogos calculados com o algoritmo modificado, $\left|\frac{f(z)-\overline{[p/q]}_f^B(z)}{f(z)}\right|$ a verde, utilizando Fortran90 em precisão simples, relativamente a série de Bessel e considerando $f(z)=\frac{1}{\sqrt{1-z}}e^{\frac{1}{1+\sqrt{1-z}}}$

Apêndice B

Gráficos e tabelas do capítulo 3

Os gráficos apresentados no capítulo 2, assim como as tabelas, são gerados pela função Plot do software Mathematica utilizando a opção `Compiled->False` (ver apêndice (D)).

Comecemos por enunciar novamente os exemplos que foram analisados nesta tese:

Exemplo 2.2 *A partir da fórmula [13]*

$$z^{\mu} = \Gamma(\mu+\beta+1)\sum_{n=0}^{\infty}\frac{(-1)^n(2n+\gamma)\Gamma(n+\gamma)(-\mu)_n}{\Gamma(n+\beta+1)\Gamma(n+\gamma+\mu+1)}R_n^{(\alpha,\beta)}(z), \quad 0<z<1$$

válida para

$$\alpha,\ \beta > -1, \quad -Re(\mu) < \min(1+\beta, 3/4+\beta/2),$$

onde $R_n^{(\alpha,\beta)}(z)$ *representa o polinómio de Jacobi de grau n definido no intervalo* $[0,1]$, *com peso* $w(z) = (1-z)^{\alpha}z^{\beta}$ *e* $\gamma = \alpha+\beta+1$.

Tomando $\alpha = \beta = 1/2$ *e* $\mu = -1/2$, *obtemos*

$$f(z) = \frac{1}{\sqrt{z}} = \frac{16}{\pi}\sum_{n=0}^{\infty}(-1)^n\frac{n+1}{(2n+1)(2n+3)}U_n(z), \quad 0<z<1,$$

onde $U_n(z)$ *é o polinómio de Chebyshev de segunda espécie de grau n definido no intervalo* $[0,1]$, *normalizado pela condição* $U_n(1) = n+1$. *Estes polinómios satisfazem a relação de recorrência [11]*

$$U_{n+1}(z) = 2(2z-1)U_n(z) - U_{n-1}(z).$$

Assim temos

$$\left\{\alpha_i = \frac{1}{4}, \beta_i = \frac{1}{2}, \gamma_i = \frac{1}{4}\right\}_{i\leq 0}$$

e

$$\left\{f_i = \frac{16}{\pi}(-1)^i\frac{i+1}{(2i+1)(2i+3)}\right\}_{i\leq 0}.$$

Exemplo 2.3 *Considere-se a série [10]*

$$f(z) = \frac{1}{2}e^{z/2} = \sum_{i\geq 0}(-1)^i L_i^{(\alpha)}(z), \quad 0<z<\infty.$$

Os polinómios de Laguerre $L_i^{(\alpha)}(z)$, *normalizados com coeficiente principal* $k_i = \frac{(-1)^i}{i!}$, *satisfazem a relação de recorrência*

$$\begin{cases} L_{i+1}^{(\alpha)}(z) = \frac{2i+\alpha+1-z}{i+1} L_i^{(\alpha)}(z) - \frac{i+\alpha}{i+1} L_{i-1}^{(\alpha)}(z), & i \geq 1 \\ L_0^{(\alpha)}(z) = 1, \quad L_1^{(\alpha)}(z) = 1 + \alpha - z \end{cases}$$

Considerando $\alpha = 0$, *temos*

$$\{\alpha_i = -(i+1), \beta_i = 2i+1, \gamma_i = -i\}_{i \leq 0}$$

e

$$\{f_i = (-1)^i\}_{i \leq 0}.$$

Exemplo 2.4 *Considere-se a série [10]*

$$f(z) = e^{2z-1} = \sum_{i \geq 0} \frac{1}{i!} H_i(z), \quad 0 < z < 2.$$

Os polinómios de Hermite $H_i(z)$, *normalizados com coeficiente principal* $k_i = 2^i$, *satisfazem a relação de recorrência*

$$H_{i+1}(z) = 2zH_i(z) - 2iH_{i-1}(z).$$

Assim temos

$$\left\{\alpha_i = \frac{1}{2}, \beta_i = 0, \gamma_i = i\right\}_{i \leq 0}$$

e

$$\left\{f_i = \frac{1}{i!}\right\}_{i \leq 0}.$$

Exemplo 2.5 *Considere-se a série [15, 16]*

$$f(z) = \frac{1}{\sqrt{1-z}} e^{\frac{1}{1+\sqrt{1-z}}} = \sum_{i \geq 0} \frac{1}{2^i i!} B_i(z), \quad -\infty < z < +\infty.$$

Os polinómios mónicos de Bessel $B_i(z)$, *satisfazem a relação de recorrência*

$$\begin{cases} B_{i+1}(z) = (z - \beta_i) B_i(z) + \gamma_i B_{i-1}(z), & i \geq 1 \\ B_0(z) = 1, \quad B_1(z) = z - \beta_0 \end{cases}$$

Assim temos

$$\beta_0 = -1, \left\{\beta_i = 0, \gamma_i = -\frac{1}{(2i-1)(2i+1)}\right\}_{i \leq 1}$$

e

$$\left\{f_i = \frac{1}{2^i i!}\right\}_{i \leq 0}.$$

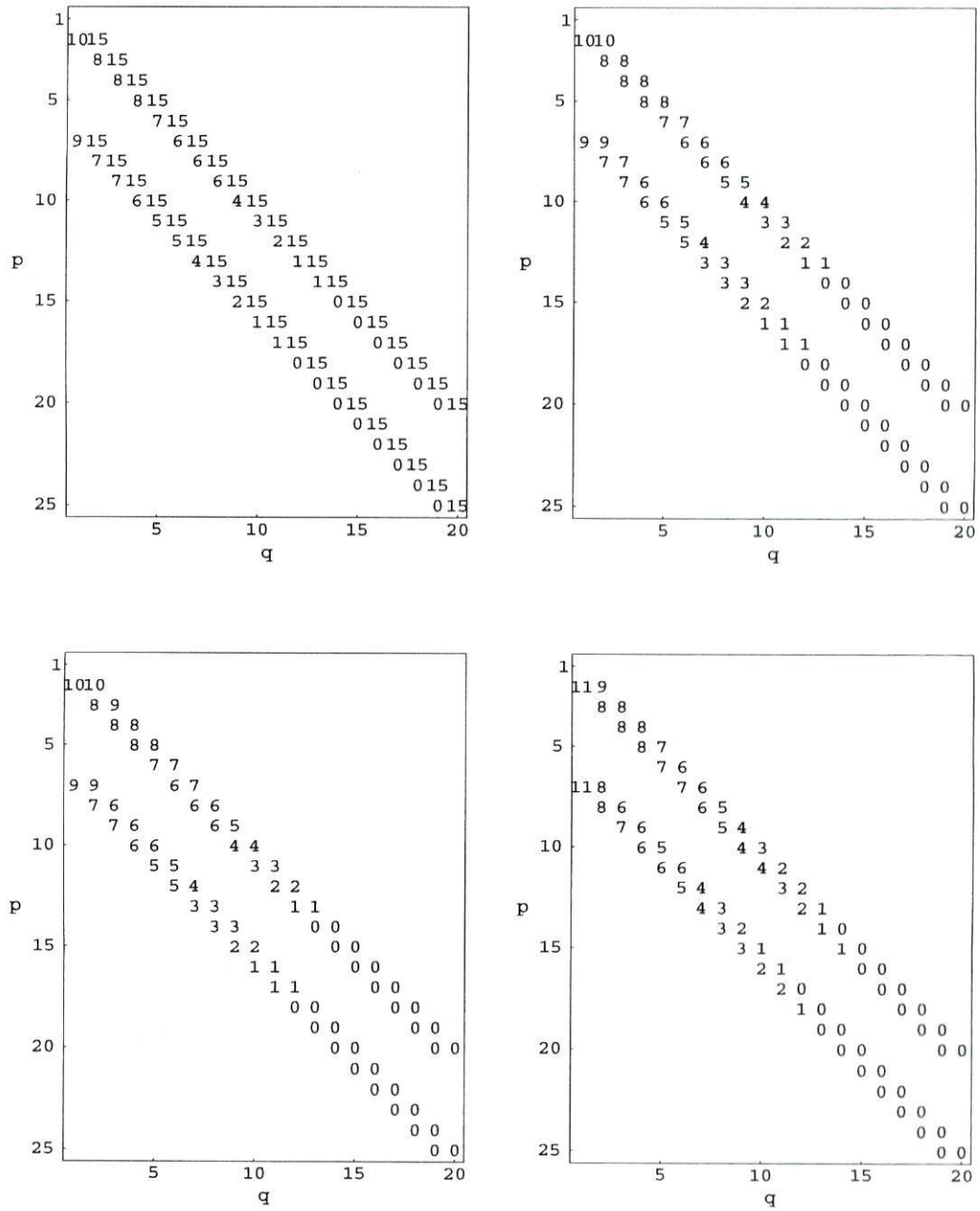


Tabela B.1: Algarismos significativos correctamente calculados dos parâmetros $\widetilde{\tau}$, $\widetilde{\lambda}$, $\widetilde{\eta}$ e $\widetilde{\rho}$ respectivamente, obtidos a partir dos $\widetilde{f}_i$ perturbados com um erro relativo da ordem de 5×10^{-12}, calculados com o novo algoritmo programado em Fortran90 com precisão dupla e utilizando a biblioteca CADNA, relativamente à série de Chebyshev de 2^a espécie

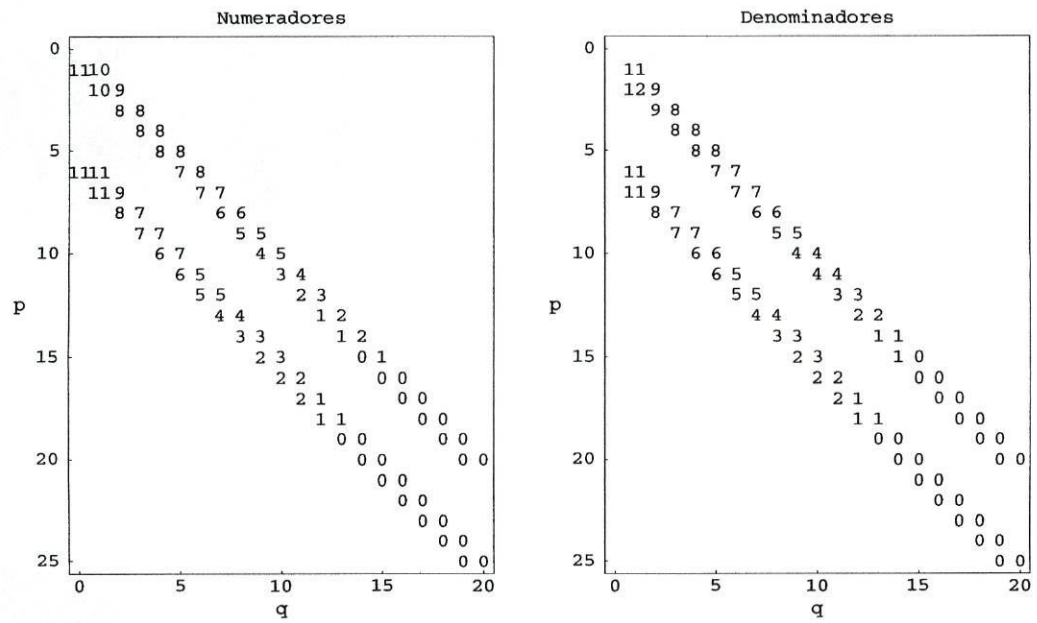


Tabela B.2: Algarismos significativos correctamente calculados do coeficiente com menor precisão do numerador e do denominador, obtidos a partir dos $\widetilde{f}_i$ perturbados com um erro relativo da ordem de 5×10^{-12}, calculados com o novo algoritmo programado em Fortran90 com precisão dupla e utilizando a biblioteca CADNA, relativamente à série de Chebyshev de 2^a espécie

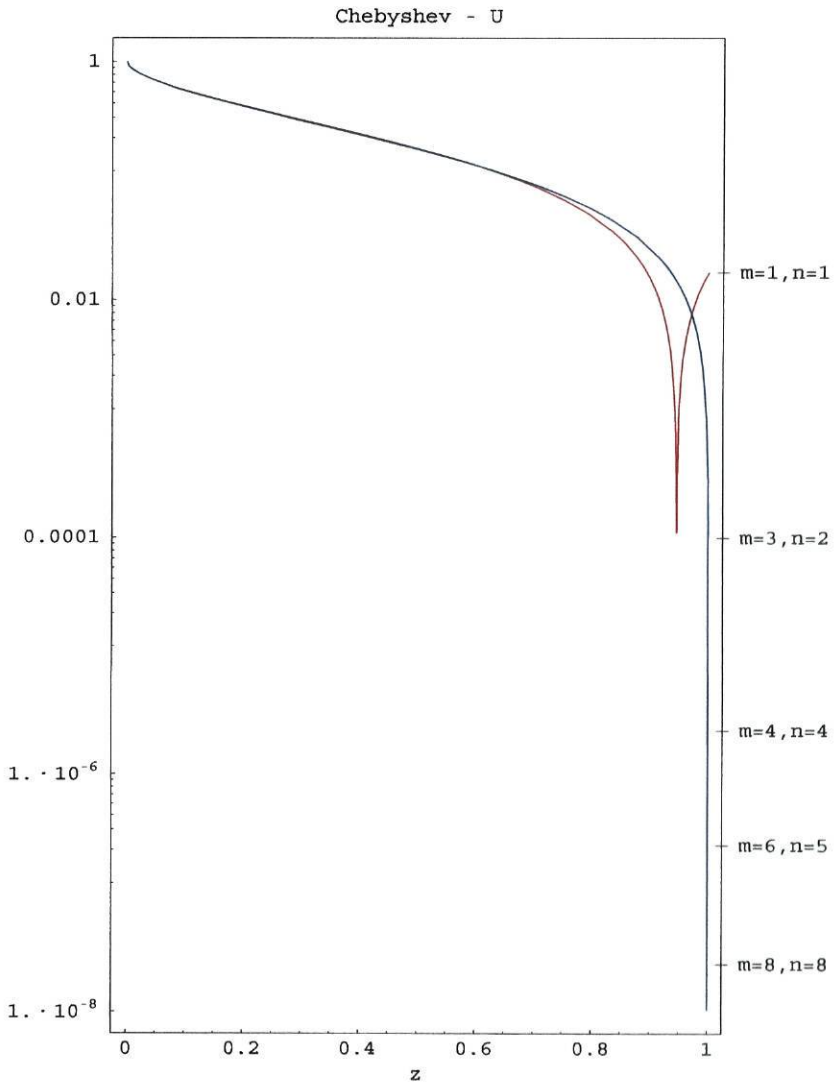


Figura B.1: Erros relativos, em escala logarítmica, para uma amostra de aproximantes $\widetilde{[m/n]}_f^U(z)$ calculados com os valores $\widetilde{f_i}$ perturbados com um erro relativo da ordem de 5×10^{-12}, $\left|\frac{f(z)-\widetilde{[m/n]}_f^U(z)}{f(z)}\right|$, relativamente à série de Chebyshev de 2^a espécie e considerando $f(z) = 1/\sqrt{z}$

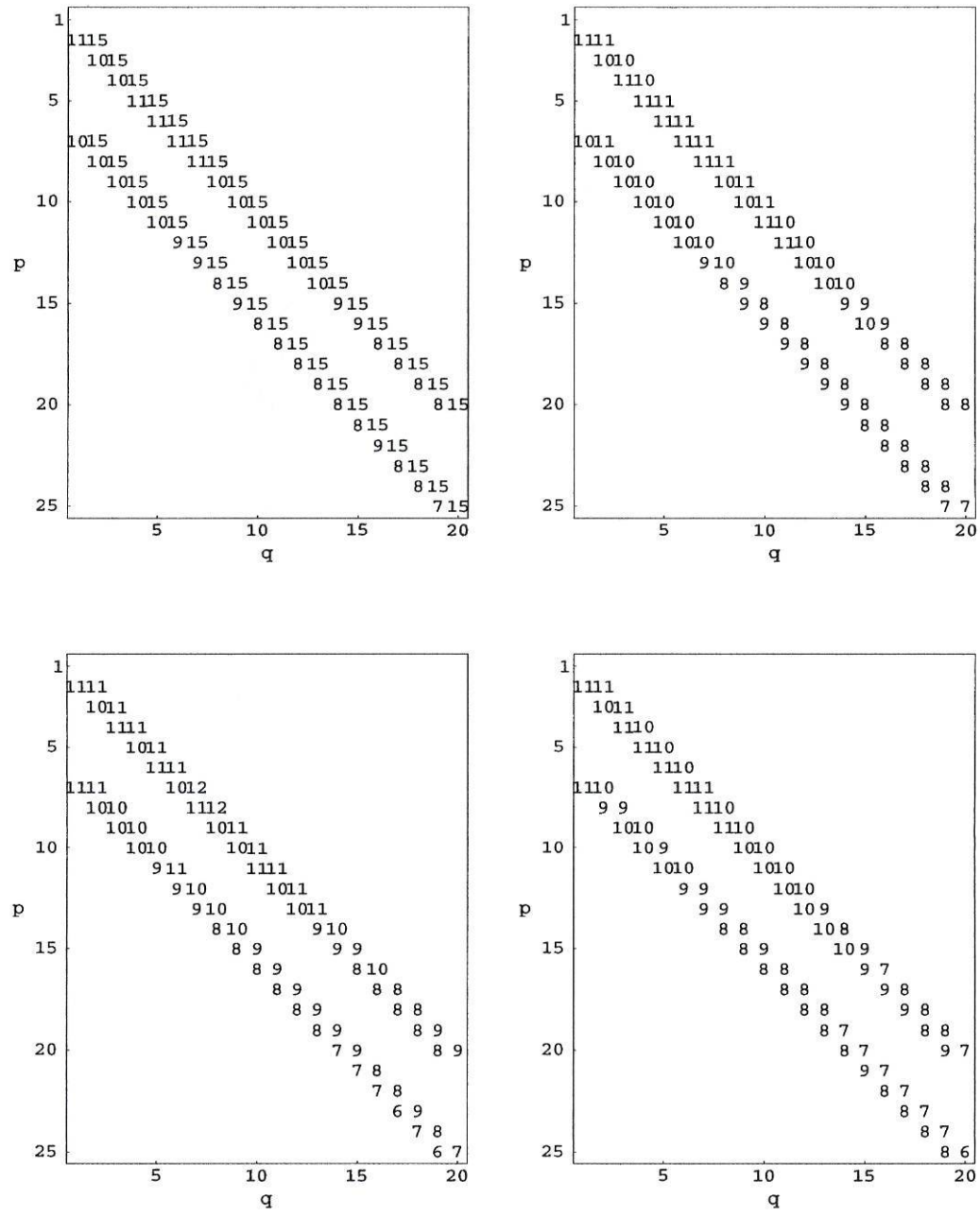


Tabela B.3: Algarismos significativos correctamente calculados dos parâmetros $\widetilde{\tau}$, $\widetilde{\lambda}$, $\widetilde{\eta}$ e $\widetilde{\rho}$ respectivamente, obtidos a partir dos $\widetilde{f}_i$ perturbados com um erro relativo da ordem de 5×10^{-12}, calculados com o novo algoritmo programado em Fortran90 com precisão dupla e utilizando a biblioteca CADNA, relativamente à série de Laguerre

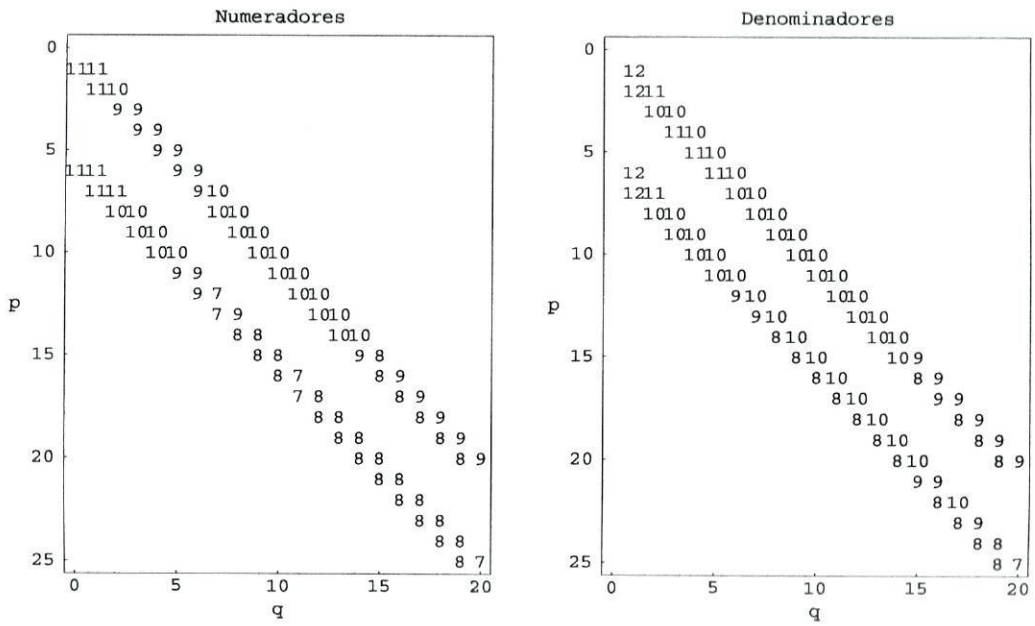


Tabela B.4: Algarismos significativos correctamente calculados do coeficiente com menor precisão do numerador e do denominador, obtidos a partir dos $\widetilde{f}_i$ perturbados com um erro relativo da ordem de 5×10^{-12}, calculados com o novo algoritmo programado em Fortran90 com precisão dupla e utilizando a biblioteca CADNA, relativamente à série de Laguerre

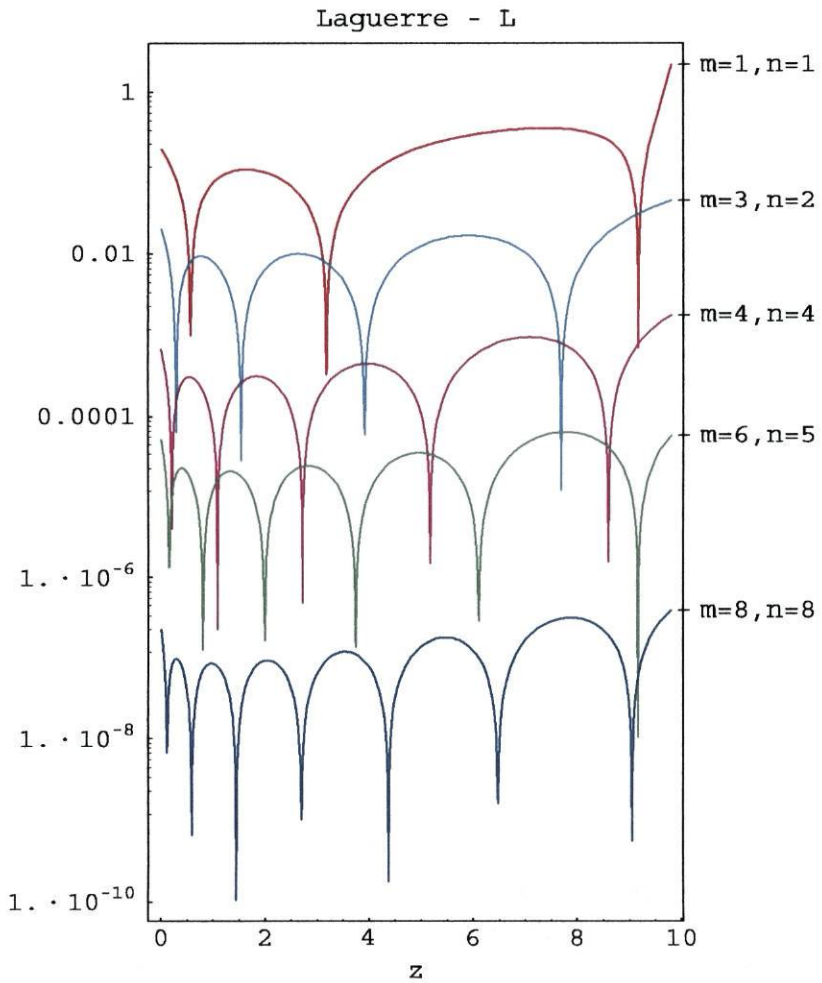


Figura B.2: Erros relativos, em escala logarítmica, para uma amostra de aproximantes $\widetilde{[m/n]}_f^L(z)$ calculados com os valores $\widetilde{f}_i$ perturbados com um erro relativo da ordem de 5×10^{-12}, $\left|\frac{f(z)-\widetilde{[m/n]}_f^L(z)}{f(z)}\right|$, relativamente à série de Laguerre e considerando $f(z) = \frac{e^{z/2}}{2}$

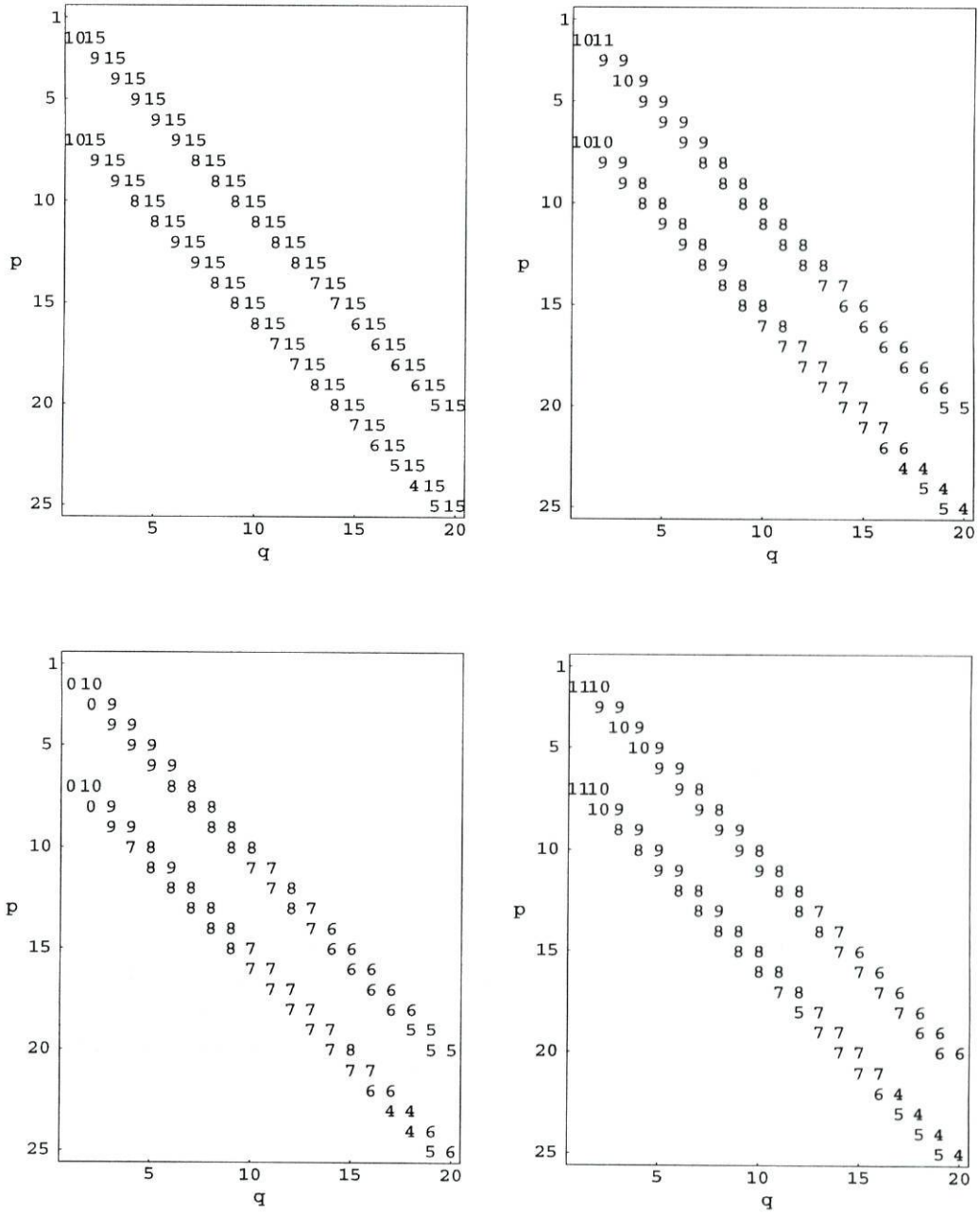


Tabela B.5: Algarismos significativos correctamente calculados dos parâmetros $\widetilde{\tau}$, $\widetilde{\lambda}$, $\widetilde{\eta}$ e $\widetilde{\rho}$ respectivamente, obtidos a partir dos $\widetilde{f}_i$ perturbados com um erro relativo da ordem de 5×10^{-12}, calculados com o novo algoritmo programado em Fortran90 com precisão dupla e utilizando a biblioteca CADNA, relativamente à série de Hermite

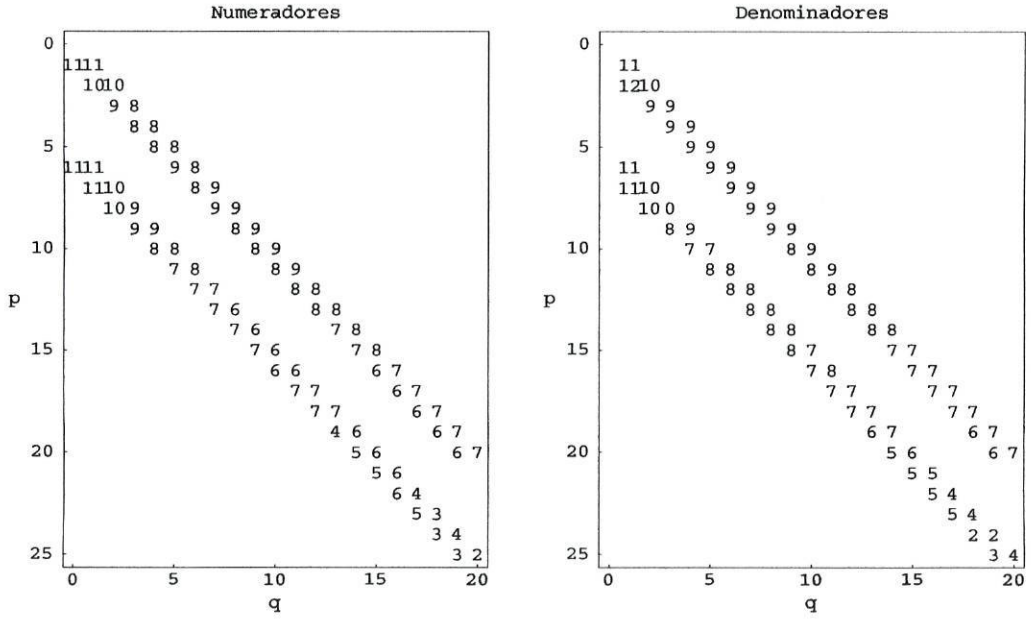


Tabela B.6: Algarismos significativos correctamente calculados do coeficiente com menor precisão do numerador e do denominador, obtidos a partir dos $\widetilde{f}_i$ perturbados com um erro relativo da ordem de 5×10^{-12}, calculados com o novo algoritmo programado em Fortran90 com precisão dupla e utilizando a biblioteca CADNA, relativamente à série de Hermite

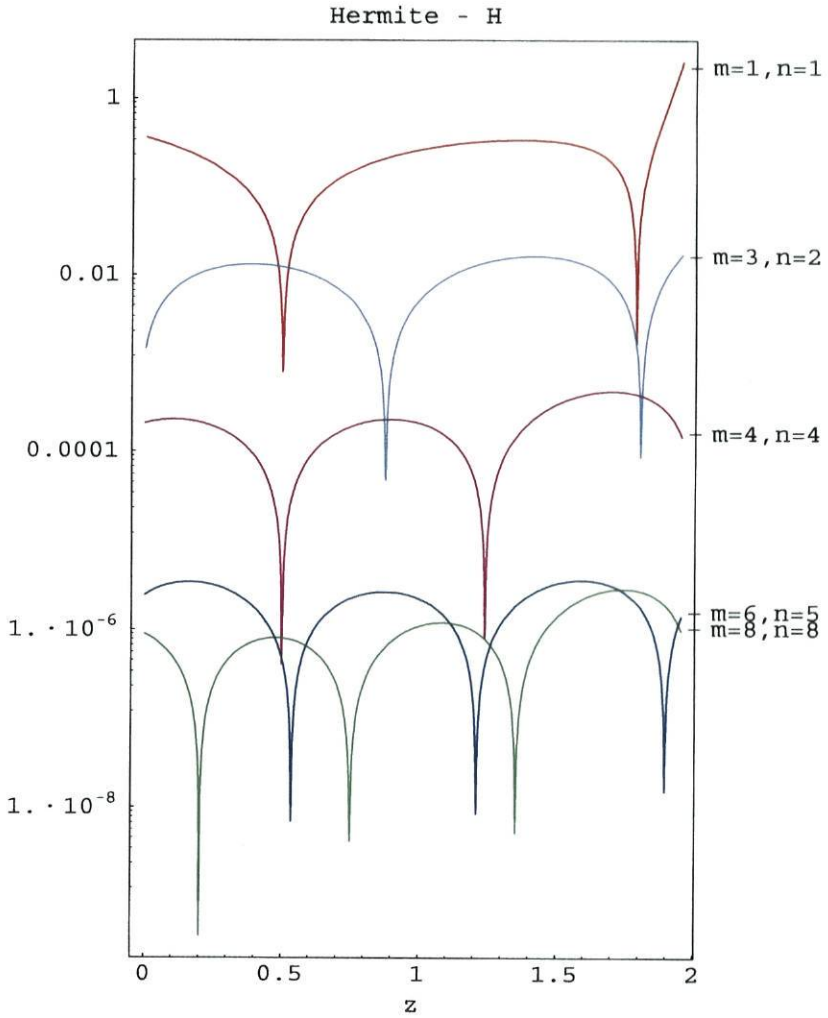


Figura B.3: Erros relativos, em escala logarítmica, para uma amostra de aproximantes $\widetilde{[m/n]}_f^H(z)$ calculados com os valores $\widetilde{f}_i$ perturbados com um erro relativo da ordem de 5×10^{-12}, $\left|\frac{f(z)-\widetilde{[m/n]}_f^H(z)}{f(z)}\right|$, relativamente à série de Hermite e considerando $f(z) = e^{2z-1}$

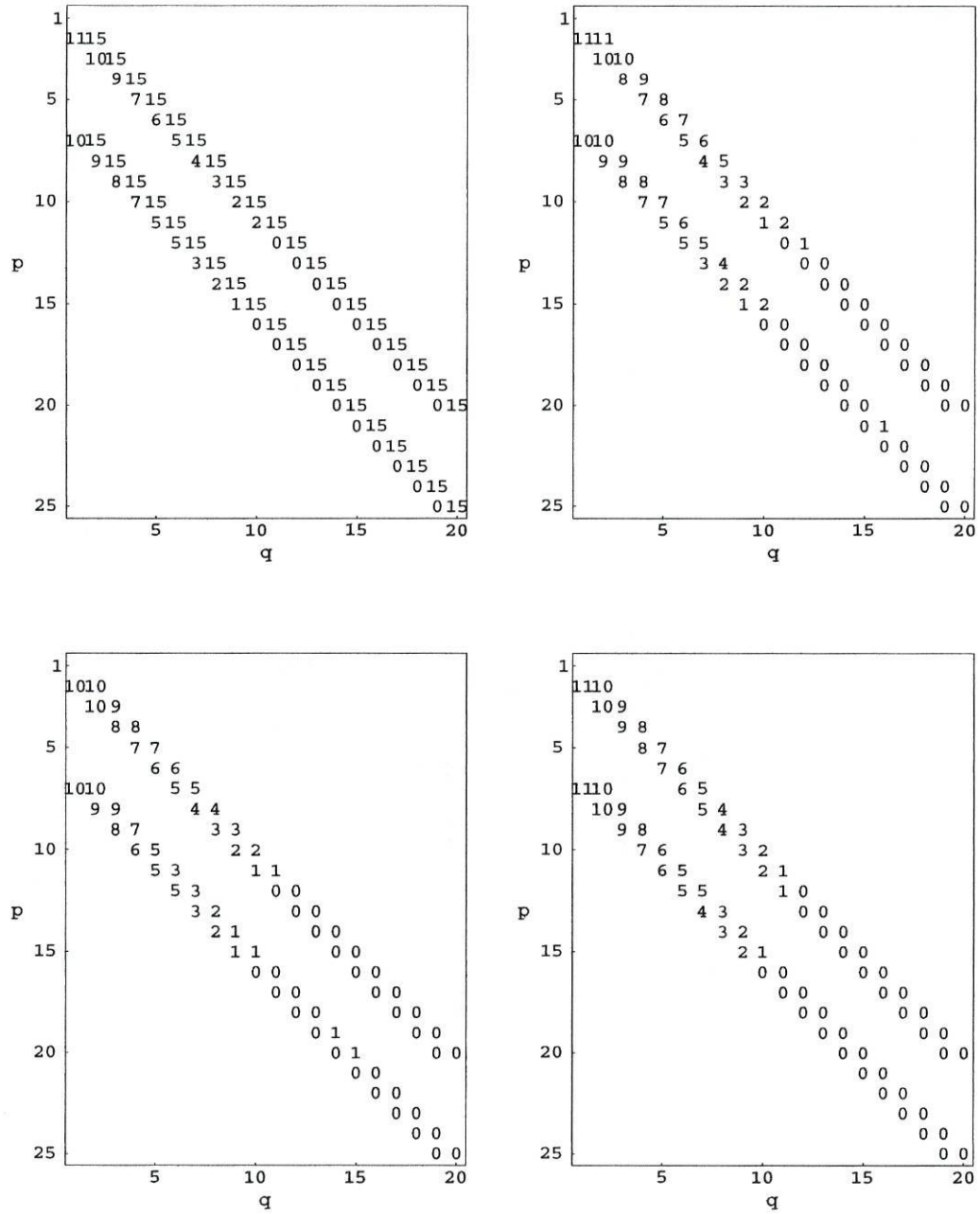


Tabela B.7: Algarismos significativos correctamente calculados dos parâmetros $\widetilde{\tau}$, $\widetilde{\lambda}$, $\widetilde{\eta}$ e $\widetilde{\rho}$ respectivamente, obtidos a partir dos $\widetilde{f}_i$ perturbados com um erro relativo da ordem de 5×10^{-12}, calculados com o novo algoritmo programado em Fortran90 com precisão dupla e utilizando a biblioteca CADNA, relativamente à série de Bessel

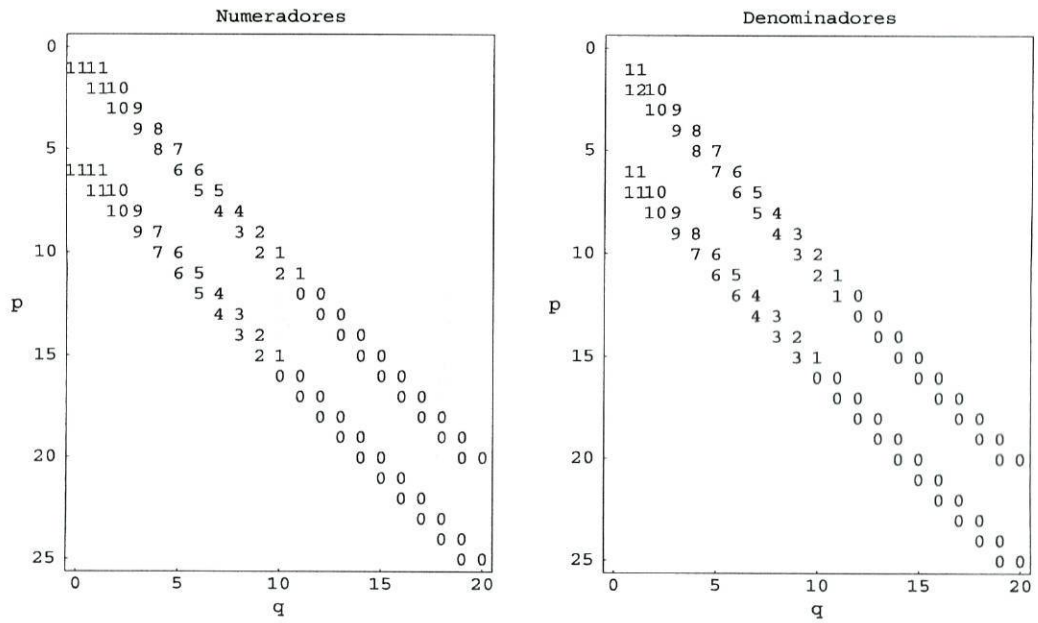


Tabela B.8: Algarismos significativos correctamente calculados do coeficiente com menor precisão do numerador e do denominador, obtidos a partir dos $\widetilde{f}_i$ perturbados com um erro relativo da ordem de 5×10^{-12}, calculados com o novo algoritmo programado em Fortran90 com precisão dupla e utilizando a biblioteca CADNA, relativamente à série de Bessel

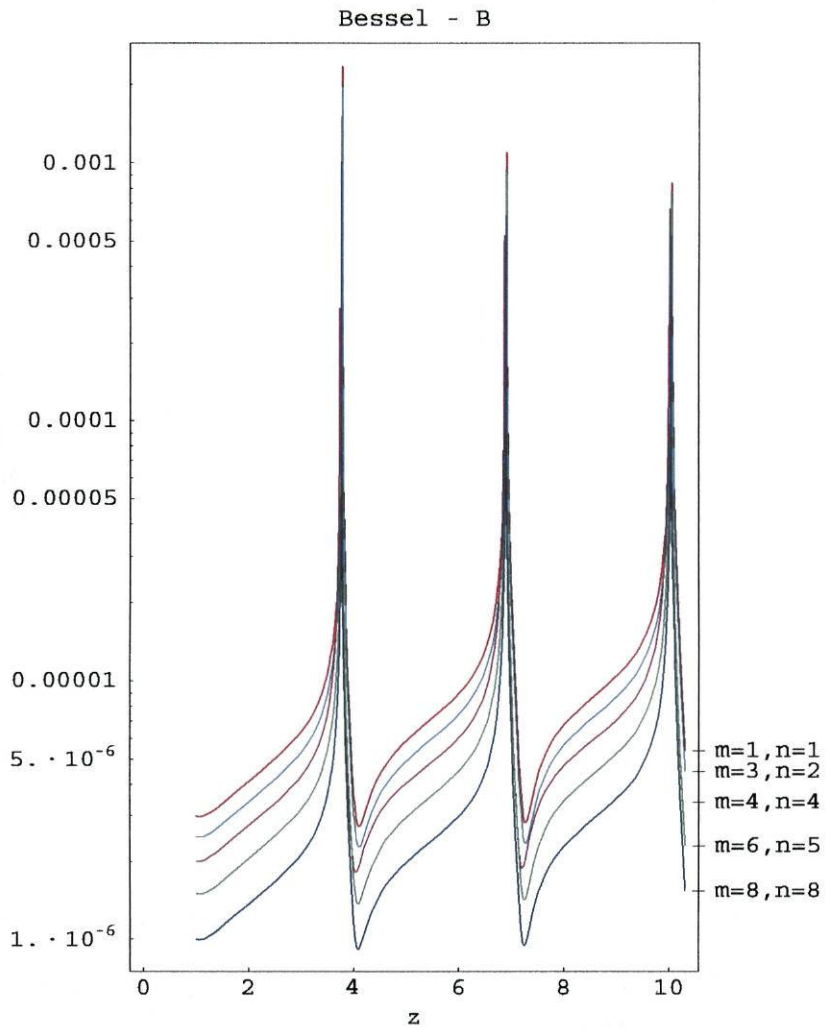


Figura B.4: Erros relativos, em escala logarítmica, para uma amostra de aproximantes $\widetilde{[m/n]}_f^B(z)$ calculados com os valores $\widetilde{f}_i$ perturbados com um erro relativo da ordem de 5×10^{-12}, $\left|\frac{f(z)-\widetilde{[m/n]}_f^B(z)}{f(z)}\right|$, relativamente à série de Bessel e considerando $f(z) = \frac{1}{\sqrt{1-z}} e^{\frac{1}{1+\sqrt{1-z}}}$

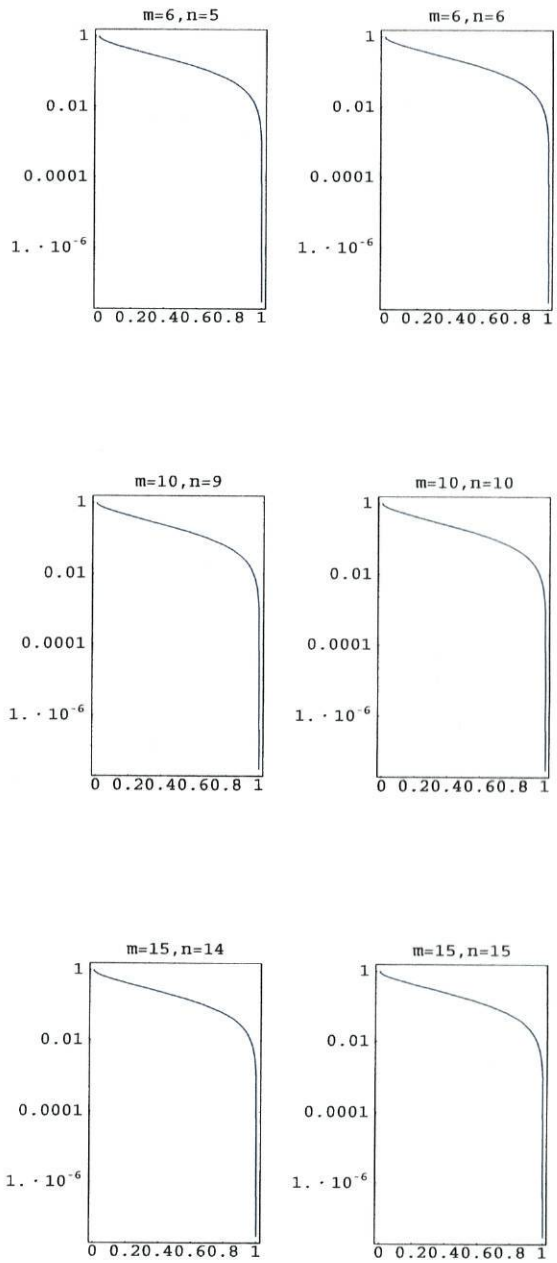


Figura B.5: Erros relativos, em escala logarítmica, para uma amostra de aproximantes $\widetilde{[m/n]}_f^P(z)$ calculados com os valores $\widetilde{a_i}$ e $\widetilde{b_i}$ perturbados com um erro relativo da ordem de $5 \times 10^{-\alpha}$, $\left|\frac{f(z)-\left(\widetilde{[m/n]}_f^P(z)\right)_\alpha}{f(z)}\right|$, representados a azul para $\alpha = 8$, a verde para $\alpha = 10$ e a vermelho para $\alpha = 14$, usando cálculo formal no Mathematica, relativamente à série de Chebyshev de 2^a espécie e considerando $f(z) = 1/\sqrt{z}$

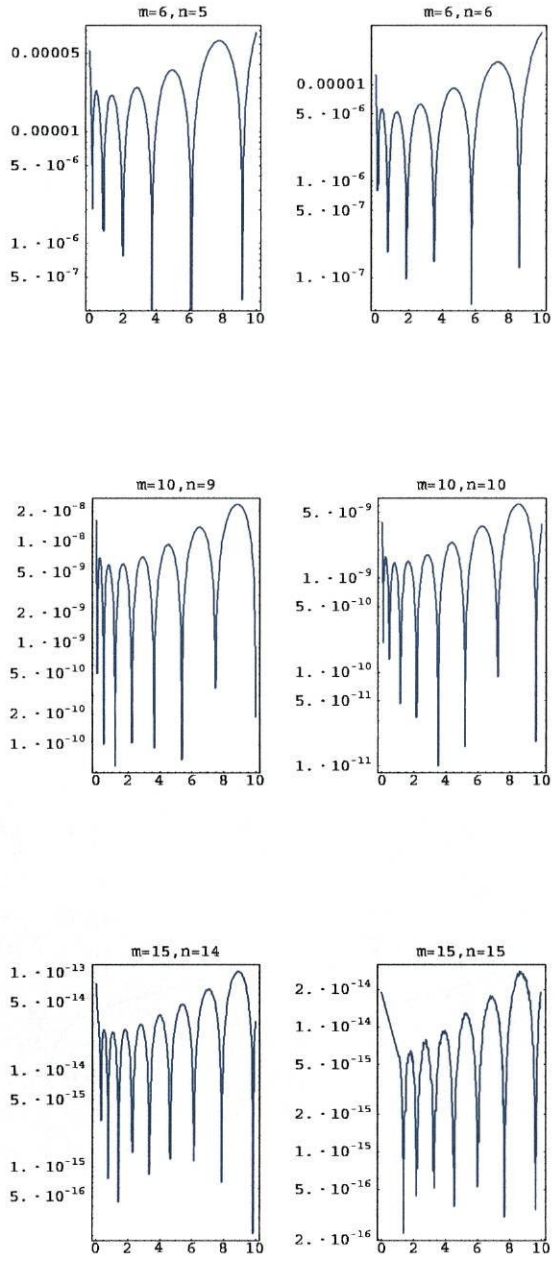


Figura B.6: Erros relativos, em escala logarítmica, para uma amostra de aproximantes $\widetilde{[m/n]}_f^P(z)$ calculados com os valores $\widetilde{a_i}$ e $\widetilde{b_i}$ perturbados com um erro relativo da ordem de $5 \times 10^{-\alpha}$, $\left| \frac{f(z) - \left(\widetilde{[m/n]}_f^P(z) \right)}{f(z)} \right|_{\alpha}$, representados a azul para $\alpha = 8$, a verde para $\alpha = 10$ e a vermelho para $\alpha = 14$, usando cálculo formal no **Mathematica**, relativamente à série de Laguerre e considerando $f(z) = \frac{e^{z/2}}{2}$

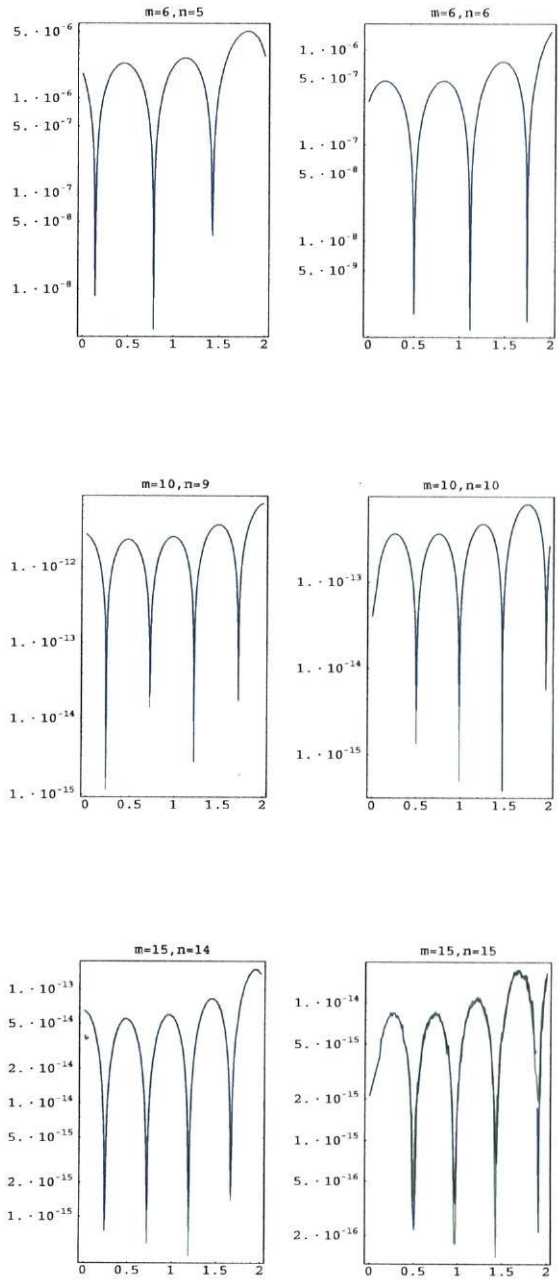


Figura B.7: Erros relativos, em escala logarítmica, para uma amostra de aproximantes $\widetilde{[m/n]}_f^P(z)$ calculados com os valores $\widetilde{a}_i$ e $\widetilde{b}_i$ perturbados com um erro relativo da ordem de $5 \times 10^{-\alpha}$, $\left| \frac{f(z) - \left(\widetilde{[m/n]}_f^P(z) \right)}{f(z)} \right|_{\alpha}$, representados a azul para $\alpha = 8$, a verde para $\alpha = 10$ e a vermelho para $\alpha = 14$, usando cálculo formal no Mathematica, relativamente à série de Hermite e considerando $f(z) = e^{2z-1}$

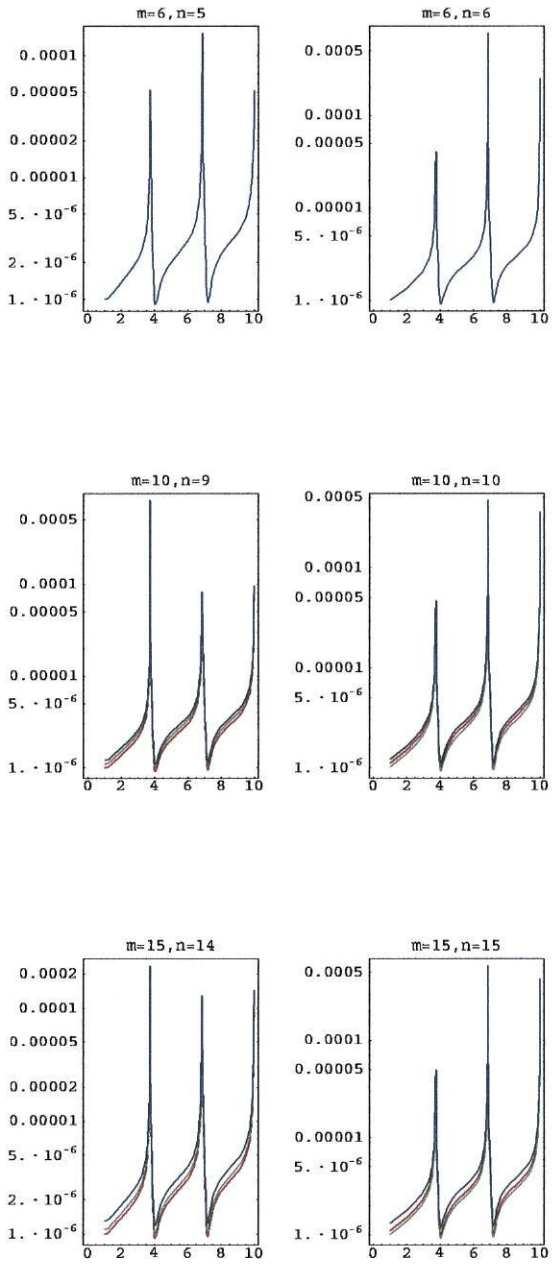


Figura B.8: Erros relativos, em escala logarítmica, para uma amostra de aproximantes $\widetilde{[m/n]}_f^P(z)$ calculados com os valores $\widetilde{a_i}$ e $\widetilde{b_i}$ perturbados com um erro relativo da ordem de $5 \times 10^{-\alpha}$, $\left|\frac{f(z)-\left(\widetilde{[m/n]}_f^P(z)\right)_\alpha}{f(z)}\right|$, representados a azul para $\alpha = 8$, a verde para $\alpha = 10$ e a vermelho para $\alpha = 14$, usando cálculo formal no Mathematica, relativamente à série de Bessel e considerando $f(z) = \frac{1}{\sqrt{1-z}} e^{\frac{1}{1+\sqrt{1-z}}}$

Apêndice C

Biblioteca CADNA

A aritmética de vírgula flutuante standard da IEEE apenas aproxima a aritmética exacta. Logo, quando um código de programação científico é executado num computador que respeita a IEEE standard os resultados não são exactos; a aproximação introduz um *erro de arredondamento* por cada operação.

Neste capítulo apresentamos as funcionalidades da biblioteca CADNA (Control of Accuracy and Debugging Numerical Applications) [5] que tem como principal objectivo estimar o efeito da propagação dos erros de arredondamento. O CADNA permite também identificar tipos de instabilidades numéricas e as linhas do programa onde figuram as expressões que as causam.

Os resultados obtidos com o CADNA exibem apenas os algarismos significativos que estão correctos, pelo que é típico aparecer resultados com um número decrescente de algarismos significativos, consequência da propagação crescente dos erros de arredondamento.

Para a implementação da biblioteca CADNA é necessário alterar substancialmente o código fonte do programa original.

C.1 Introdução ao CADNA

A validação de resultados numéricos é um verdadeiro problema na área da computação científica. Foi por bastante tempo ignorado pelos utilizadores e programadores mas é hoje reconhecido como sendo um tópico essencial.

A biblioteca CADNA permite o desenvolvimento de aplicações numéricas robustas e de alta performance. O CADNA fornece uma ajuda fundamental na investigação de comportamentos suspeitos de programas numéricos produzidos em ADA, C ou Fortran. O CADNA implementa o método CESTAC (Contrôle et Estimation Stochastique des Arrondis de Calcul), um modelo numérico estocástico de propagação de erros de arredondamento. A biblioteca CADNA permite, durante a execução do programa:

- estimar o valor do erro devido à propagação dos erros de arredondamento
- detectar instabilidades numéricas

- testar a sequência do programa
- estimar o número de algarismos significativos de todas os cálculos intermédios

O CADNA é baseado no método CESTAC (ver secção C.5), o qual estuda a propagação dos erros de arredondamento de um ponto de vista estocástico. A ideia básica é usar um processo de arredondamento aleatório para obter varias amostras de resultados de cada operação aritmética. O número de bits comuns nestas amostras estima o número de bits significativos do resultado real. Logo, a aritmética determinística do computador é substituída por uma *aritmética discreta aleatória.*

C.2 Objectivo da biblioteca CADNA

A aritmética usualmente utilizada em computadores para computação científica é a aritmética de vírgula flutuante. Visto que esta aritmética apenas aproxima a aritmética exacta, cada operação aritmética gera um erro de arredondamento. Logo, todos os resultados produzidos por computador têm inerente o chamado *erro computacional.* Este erro é proveniente de todos os erros de arredondamento produzidos durante a execução de todas as expressões elementares necessárias para obter o resultado.

O objectivo da biblioteca CADNA é responder à seguinte pergunta: *Qual o valor do erro computacional, gerado pela aritmética de vírgula flutuante, nos resultados produzidos por um dado programa executado num computador?*

Pretendemos estimar o erro de arredondamento de cada resultado através de um processo que é independente do programa e do algoritmo usados.

CADNA é uma biblioteca, mais precisamente, é um conjunto de tipos de variáveis, funções e subrotinas que podem ser utilizadas em qualquer programa desenvolvido em ADA, C ou Fortran. Esta biblioteca tem como principal objectivo estimar o efeito da propagação dos erros de arredondamento de todos os resultados numéricos obtidos por computador. Permite também avaliar os efeitos da imprecisão dos dados iniciais sobre os resultados finais.

Durante a execução do programa, à medida que são produzidas anomalias numéricas, certas mensagens são escritas num ficheiro particular chamado cadna_stability_f90.lst. Se este ficheiro não exibir nenhuma mensagem significa que o programa foi executado sem qualquer problema numérico. Os resultados são apresentados com o respectivo número de algarismos significativos exactos. Se o ficheiro em questão enumerar uma lista de mensagens então estas devem ser analisadas. Com a ajuda de um *debugger* associado ao compilador, o utilizador terá conhecimento das expressões que produzem as anomalias numéricas e poderá eventualmente alterar o código do programa no sentido de dissolver estas instabilidades.

C.3 Implementação utilizando o CADNA

O uso da biblioteca CADNA envolve seis passos [5]:

1. Declaração da biblioteca CADNA.

   A expressão use cadnafreeF deve ser introduzida no início de qualquer programa, função ou subrotina e antes da declaração das variáveis. Permite o uso de variáveis estocásticas e disponibiliza novas funções e subrotinas.

2. Substituição do tipo real ou real(8) por tipos estocásticos na declaração das variáveis.

   Para controlar a qualidade numérica de uma variável basta substituir o seu tipo standard por um tipo estocástico associado. Por exemplo:

Declaração standard	Declaração com CADNA
real :: a,b	type(single_st) :: a,b
real(8) :: c	type(double_st) :: c
real, dimension(6) :: d,e,f	type(single_st), dimension(6) :: d,e,f

   Pode utilizar-se variáveis implícitas declarando implicit type(single_st) (A-H,O-Z).

3. Inicialização de aritmética aleatória e criação do ficheiro cadna_stability_f90.lst que apresenta a lista de instabilidades numéricas geradas durante a execução do programa.

   Após a declaração das variáveis, deve ser chamada a subrotina cadna_init(k_instabilidades, semente) para criar o ficheiro cadna_stability_f90.lst e iniciar a aritmética aleatória. Ambos os argumentos desta subrotina são inteiros. O primeiro argumento pode tomar os valores:

   - Se k_instabilidades=-1, são exibidas todas as mensagens relacionadas com todas as instabilidades detectadas.
   - Se k_instabilidades=0, não é exibida nenhuma mensagem.
   - Se k_instabilidades=N (N positivo), são exibidas N mensagens relacionadas com as primeiras N instabilidades detectadas.

   O segundo argumento é opcional, pode tomar um valor entre 1 e 2^{314} e é utilizado para inicializar algumas variáveis internas na aritmética aleatória.

4. Possibilidade de perturbar os dados com o objectivo de verificar a influência do número de algarismos significativos exactos dos dados iniciais no resultado.

   A subrotina data_st(X,ER_X) permite ao utilizador introduzir algumas incertezas nos dados iniciais, reduzindo o número de algarismos significativos exactos. Este procedimento permite constatar até que ponto o número de algarismos significativos exactos do resultado depende do o número de algarismos significativos exactos dos dados iniciais.

   A subrotina data_st tem dois argumentos. O primeiro argumento X é estocástico. O segundo ER_X é opcional, é um argumento real que indica o erro relativo do primeiro.

5. Alteração das expressões de forma a apresentar os resultados estocásticos com o respectivo número de algarismos significativos.

   Antes de apresentar cada resultado estocástico, este deve ser transformado numa *string* utilizando a função str. Consequentemente também o formato de impressão deve ser modificado. Por exemplo:

Código Fortran90	Código Fortran90 com CADNA
real :: x	type(single_st) :: x
...	...
write(*,100) x	write(*,100) str(x)
100 format(10E15,7)	100 format(10a15)

6. Utilização de funções fornecidas pelo CADNA, por opção do utilizador.

   Entre outras, a função cestac(x) é uma função disponibilizada pelo CADNA que tem como argumento uma variável estocástica e retorna um inteiro exibindo o número de algarismos significativos exactos do argumento.

C.4 Utilização do *Debugger*

Quando um programa implementado com o CADNA é executado, é criado o ficheiro canda_stability_f90.lst que exibe uma lista de instabilidades numéricas geradas durante a execução do programa. O CADNA executa uma verificação extra dos erros e escreve neste ficheiro o tipo de instabilidade e a ordem respectiva de aparecimento. Cada instabilidade numérica detectada é associada a uma mensagem específica:

unstable multiplication: ambos os factores são não significativos

unstable division: o denominador é não significativo

unstable power function: um dos expoentes é não significativo

unstable mod function: o segundo argumento é não significativo

unstable dim function: a diferença entre dois factores é não significativa

unstable int function: o argumento não pode ser exactamente determinado devido à propagação dos erros de arredondamento

unstable nint function: o argumento não pode ser exactamente determinado devido à propagação dos erros de arredondamento

unstable sign function: o segundo argumento é não significativo

unstable branching: a diferença entre dois factores é não significativa (um zero estocástico); a expressão está associada a uma desigualdade

unstable log function: o argumento da função log é negativo ou não significativo

unstable log10 function: o argumento da função log10 é negativo ou não significativo

unstable sqrt function: o argumento da função sqrt é significativo mas negativo

unstable exponential function: o argumento da função exp é não significativo

Se o ficheiro no final da execução não contiver nenhuma mensagem, então podemos considerar os resultados credíveis. Se, por outro lado, o ficheiro apresentar algumas mensagens, então ocorreram anomalias numéricas que podem ter perturbado o resultado.

Após tomarmos conhecimento do tipo de instabilidades falta:

- encontrar qual a expressão que causa a instabilidade
- modificar o código nesse local no sentido de eliminar a instabilidade numérica

O *debugging* numérico pode ser realizado com um *debugger* simbólico como o idb da Intel para Linux.

Com o objectivo de encontrar a linha de código onde a instabilidade é causada deve ser colocada uma paragem sempre que a função instability da biblioteca CADNA é executada, já que esta função é chamada sempre que uma instabilidade numérica é detectada.

Utilizando o idb da Intel para Linux e o ficheiro idb.in (ver anexo E.5), o seguinte comando generalizado permite escrever em idb.out todos os locais onde as instabilidades numéricas são atingidas:

nohup idb *nome_do_executavel* < idb.in > idb.out &

C.5 Supervisão do método CESTAC

O objectivo do método CESTAC (Contrôle et Estimation Stochastique des Arrondis de Calcul), baseado no estudo dos erros de arredondamento do ponto de vista probabilístico, é estimar o efeito da propagação dos erros de arredondamento nos resultados obtidos computacionalmente utilizando aritmética de vírgula flutuante. Este método consiste em fazer propagar os erros de arredondamento de várias formas no sentido de diferenciar a parte estável da mantissa, considerada como sendo a parte significativa, da parte instável ou não significativa [6].

A primeira ideia básica do método CESTAC é substituir a aritmética usual de vírgula flutuante pela aritmética aleatória. A aritmética aleatória é obtida da aritmética usual de vírgula flutuante perturbando aleatoriamente, para cada operação aritmética, o bit menos significativo da mantissa do resultado. A segunda ideia básica é executar várias vezes o programa com esta nova aritmética de forma a obtermos diferentes resultados em cada execução.

Na prática, o método CESTAC consiste em:

- executar o mesmo programa N vezes (em geral N=2 ou N=3) simultaneamente utilizando aritmética aleatória, obtendo desta forma, para cada resultado intermédio R de qualquer operação aritmética de vírgula flutuante, um conjunto de N resultados diferentes de R_i, $i = 1, ..., N$,
- considerar o valor da média $\bar{R} = \sum_{i=1}^{N} R_i/N$ como sendo o resultado obtido,

- usar a distribuição $t - Student$ para estimar o intervalo de confiança para R e calcular o número de algarismos significativos exactos de $\bar{R}$.

Desta forma podemos analisar a propagação dos erros de arredondamento passo a passo e detectar instabilidades e resultados não significativos.

Apêndice D

Mathematica

Para analisar e comentar com maior exactidão os resultados produzidos pelo Mathematica efectuou-se um estudo detalhado deste software. Os aspectos focados prendem-se com os comandos e funções que foram utilizados para a realização deste trabalho.

Este apêndice surge na sequência do estudo efectuado, a partir do que é exposto em [7, Apêncide B] e [22].

D.1 Representação dos números

Existem quatro tipos distintos de números no Mathematica:

Integer: inteiro exacto de precisão arbitrária.

Racional: inteiro/inteiro na forma irredutível.
In[1]:=12344/2222
Out[1]=$\frac{6172}{1111}$

Real: um número real é identificado pela presença explícita de uma vírgula e pode ter qualquer número de dígitos.

O Mathematica distingue dois tipos de números reais: os números reais na *precisão da máquina* e os números reais de *alta precisão* e ele actua de forma diferente na presença de um ou de outro tipo.

Dizemos que um número real é de baixa precisão se o seu número de algarismos significativos é não superior à precisão da máquina, dada por exemplo por Precision[1.] (Precision[x] apresenta o número de algarismos significativos de x). Nos casos em que os números têm precisão inferior à precisão da máquina, o Mathematica acrescenta zeros à direita do último dígito significativo até atingir essa precisão.

Dizemos que um número real é de alta precisão se o seu número de algarismos significativos é superior à precisão da máquina.

O Mathematica supõe sempre que os dígitos que apresenta são correctos.

In[2]:=Precision[1.]
Out[2]=16
A precisão de um Pentium 4 é 16.

In[3]:=Precision[1.2345]
Out[3]=16
O número 1.2345 encontra-se na precisão da máquina porque é interpretado como sendo 1.234500000000000.

In[4]:=Precision[1.12345678912345678912345678912345678 9]
Out[4]=36
É um número real de alta precisão.

Complex: complexo da forma *número+númeroI.*
In[5]:=4+7/8I
Out[5]=$4+\frac{7}{8}i$
In[6]:=4+5.6I
Out[6]=$4+5.6i$

O Mathematica tem um outro tipo de objecto indivisível designado por *Symbol.* Um símbolo é o mais simples objecto do Mathematica com um nome. O nome de um símbolo é constituído por um conjunto de letras ou algarismos, começando obrigatoriamente por uma letra. O Mathematica distingue as letras maiúsculas das minúsculas. A primeira letra de um símbolo pertencente ao sistema é sempre maiúscula, por convenção.
x, y1, t - símbolos definidos pelo utilizador
Pi, I - símbolos pertencentes ao sistema

Os números inteiros e racionais são tratados pelo Mathematica como exactos. Por outro lado, os números reais são tratados como aproximados.

Num sistema simbólico como o Mathematica, podemos também representar os números reais de forma exacta. Por exemplo, o símbolo Pi é uma representação exacta da constante matemática π. Se quisermos encontrar uma aproximação de Pi, somos obrigados a utilizar a função de avaliação numérica `N[]` (`N[expr,n]` que fornece a expressão `expr` com `n` algarismos significativos).
In[7]:=N[Pi,6]
Out[7]=3.14159
In[8]:=N[Pi,32]
Out[8]=3.1415926535897932384626433832795

Uma das mais importantes capacidades do Mathematica é fazer manipulação simbólica e cálculo numérico de alta precisão. Em ambas as situações podemos obter resultados exactos ou aproximados.

D.2 Formas de Cálculo

O Mathematica permite-nos representar números reais com um número qualquer de dígitos.

Em geral o Mathematica distingue três tipos de cálculo:

- Cálculo formal
  Os cálculos são realizados utilizando manipulação simbólica (precisão infinita).
- Cálculo numérico na precisão da máquina
  Os cálculos são processados com precisão=16.
- Cálculo numérico de alta precisão
  Os cálculos são efectuados com precisão>16.

D.2.1 Cálculo formal

Quando o Mathematica utiliza cálculo formal para processar os cálculos, o que faz é tratar as diversas parcelas como símbolos e tenta simplificar os cálculos sem recorrer ao valor numérico das expressões.

```
In[9]:=1/2*Sqrt[3]+123+7/2*Sqrt[3]+4+2/6*Log[22/2]
Out[9]=127+4Sqrt[3]+Log[11]/3
```

D.2.2 Cálculo na precisão da máquina

Ao utilizarmos a função `N[expr,n]`, o Mathematica tenta avaliar a expressão `expr` com `n` dígitos de precisão. Se usarmos `N[expr]` e não explicitarmos especificamente a precisão, o Mathematica irá usar a precisão da máquina.

```
In[10]:=1./2*Sqrt[3.]+123.+7/2*Sqrt[3.]+4+2/6*Log[22./2]
Out[10]=134.728
```

A presença da vírgula faz com que o Mathematica forneça uma aproximação do resultado.

Os cálculos, na precisão da máquina, são efectuados utilizando 16 algarismos significativos e o resultado também ele possuí 16 algarismos, no entanto o Mathematica apresenta, por convenção, apenas 6 dígitos. Se o utilizador necessitar de visualizar mais dígitos terá forçosamente que aplicar a função `SetPrecision[expr,n]`.

```
In[11]:=SetPrecision[1./2*Sqrt[3.]+123.+7/2*Sqrt[3.]+4+2/6*Log[22./2],16]
Out[11]=134.7275016545416
```

Quando são efectuados cálculos em alta precisão o Mathematica conserva a precisão dos resultados e apresenta apenas os dígitos correctamente calculados. Por outro lado, quando utilizamos a precisão da máquina, o Mathematica fornece resultados nesta precisão, apresentando-os com todos os dígitos quer estejam correcta ou incorrectamente calculados.

```
In[12]:=N[1.111111111111111-1.111111111111000,16]
Out[12]=1.110223024625157×10^-16

In[13]:=N[1.111111111111111111111111-1.111111111111111111111000,24]
Out[13]=1.1×10^-21
```

Quando surgem nos cálculos valores na precisão da máquina todo o cálculo é efectuado na precisão da máquina. No entanto, mesmo quando utilizamos aritmética na precisão da máquina, o Mathematica tenta sempre arredondar os resultados de forma a preservar a sua veracidade. O facto de obtermos resultados com um grau de incerteza quando efectuamos cálculo numérico na precisão da máquina resulta do facto de o Mathematica usar precisão fixa nestes cálculos, no sentido de obter maior eficiência computacional.

A computação efectuada usando a precisão da máquina é realizada utilizando directamente as capacidades numéricas do processador do computador. Como consequência, os cálculos efectuados com a precisão da máquina são produzidos rapidamente.

Um ponto importante é o facto de todos os processadores possuírem um hardware ou microcódigo particularmente concebido para efectuar cálculos de vírgula flutuante para uma determinada precisão. O Mathematica serve-se deste facto quando avalia expressões numericamente utilizando a precisão da máquina.

Todos os números na precisão da máquina são representados no Mathematica como números em vírgula flutuante de dupla precisão.

A principal vantagem de usar as capacidades de um sistema de vírgula flutuante é a rapidez de execução. Os cálculos executados em alta precisão, que não utilizam directamente tais capacidades, são geralmente mais lentos do que os cálculo efectuados na precisão da máquina.

Existem várias desvantagens no uso de um sistema de vírgula flutuante. Uma já mencionada é o facto de forçar que todos os números tenham precisão fixa, independentemente da precisão verdadeira que possam ter. A segunda é que o tratamento de números na precisão da máquina pode variar ligeiramente de um processador para outro. Desta forma, quando utilizamos 16 bytes de precisão, o Mathematica fica à mercê do sistema de aritmética de vírgula flutuante de cada processador. Se a aritmética de vírgula flutuante for diferente em dois computadores, é possível obter-se resultados ligeiramente diferentes.

A computação em precisão fixa torna os cálculos mais eficientes, mas sem uma análise cuidada nunca podemos ter a certeza do número correcto de algarismos significativos presentes no resultado.

D.2.3 Cálculo de alta precisão

Quando queremos obter resultados com precisão superior à precisão da máquina, devemos recorrer à função `N[expr,n]` com $n > 16$. Desde que seja introduzida uma precisão superior a 16, o Mathematica procurará obter um resultado com a precisão desejada. A função `N[]` utiliza um processo adaptativo para controlar a precisão interna dos cálculos no sentido de atingir resultados com a precisão pedida pelo utilizador.

```
In[8]:=N[1/2*Sqrt[3]+123+7/2*Sqrt[3]+4+2/6*Log[22/2],100]
Out[8]=134.72750165454163268879709989201186590104512329988732823729608382085125660893961238601626964969653920
```

No entanto, retomando o exemplo anterior, não devemos aplicar função `N[]` à expressão `1./2*Sqrt[3.]+123.+7/2*Sqrt[3.]+4+2/6*Log[22./2]`, mas sim a uma expressão equivalente que tenha precisão superior ou igual à desejada.

```
In[14]:=N[1./2*Sqrt[3.]+123.+7/2*Sqrt[3.]+4+2/6*Log[22./2],45]
Out[14]=134.7275016545416

In[15]:=N[N[1/2*Sqrt[3]+123+7/2*Sqrt[3]+4+2/6*Log[22/2],50],45]
Out[15]=134.727501654541632688797099892011865901045123
```

Ao efectuar qualquer tipo de cálculo numérico é esperado que os resultados venham afectados de erros de arredondamento. Quando aumentamos a ordem de precisão numérica, estes erros normalmente tornam-se cada vez mais pequenos. Um bom método para verificar os resultados é efectuar o mesmo cálculo com precisões crescentes.

Quando processamos cálculos em alta precisão, o Mathematica avalia a precisão de todos os valores que intervêm nos cálculos, i.e., avalia o número de dígitos do resultado que podem vir afectados de erros provenientes de cálculos anteriores. O Mathematica corrige a precisão de forma que no resultado não apareçam os dígitos afectados de erro. Este procedimento assegura que todos os dígitos fornecidos pelo Mathematica são correctos. Em geral o Mathematica tenta produzir os resultados com a precisão mais alta possível, precisão essa que depende da precisão dos dados iniciais.

Os algoritmos internos que o Mathematica utiliza para avaliar funções matemáticas estão construídos de forma a manter tanta precisão quanto possível. Na maioria dos casos as funções do Mathematica fornecem resultados com tanta precisão quanto a fornecida pelo utilizador. Noutros casos é simplesmente impraticável realizá-lo e o Mathematica fornece resultados com precisão inferior. Se os dados iniciais forem introduzidos em alta precisão o Mathematica irá usar alta precisão nos seus cálculos internos, e usualmente é possível obter-se resultados em alta precisão.

O Mathematica trata os números em alta precisão como aproximações de quantidades onde um certo número de dígitos são conhecidos e outros não. Quando o Mathematica trabalha com potenciais efeitos de propagação de erros em alta precisão, assume que os dígitos afectados são independentes do resultado. Desta forma, se dois números são gerados da mesma forma num cálculo, alguns dos seus dígitos afectados de erro podem ser iguais. Então, quando estes números são, por exemplo, subtraídos, estes dígitos podem cancelar-se. Por tomar em consideração estes dígitos afectados de erro e considerando-os independentes do resultado, o Mathematica não efectuará tais cancelamentos. Em parte torna-se uma desvantagem já que os algoritmos numéricos por vezes esperam que haja cancelamentos entre dígitos errados de diferentes números, conseguindo resultados com maior precisão.

D.3 Precisão dos resultados

Os números definidos em alta precisão podem conter um número qualquer de dígitos e a sua precisão é ajustada à medida que se progride nos cálculos. Por outro lado, os números definidos na precisão da máquina contêm um número fixo de dígitos e a sua precisão mantém-se à medida que se avança nos cálculos.

A precisão do resultados de uma função pode depender de uma forma complicada da precisão do dados iniciais. Funções que variam rapidamente em geral fornecem resultados com pouca precisão, já que à variação dos resultados estão associadas incertezas maiores

do que as existentes nos dados iniciais. Funções que estão perto de constantes podem eventualmente dar resultados com maior precisão do que a precisão dos dados iniciais.

Por vezes torna-se interessante realizar o mesmo cálculo de formas diferentes e verificar se os resultados têm precisões diferentes. Geralmente, uma vez perdida a precisão num cálculo é praticamente impossível recupera-la e não podemos esquecer que quando perdemos precisão estamos efectivamente a perder informação sobre o resultado. O facto de diferentes formas de fazer o mesmo cálculo fornecerem resultados numéricos diferentes significa, entre outras coisas, que comparações entre aproximações de números reais devem ser tratadas com cuidado.

A igualdade entre duas expressões também depende da precisão com com estas expressões são avaliadas, visto que para testar se dois números reais são iguais, o Mathematica calcula a sua diferença e testa se o resultado é zero para uma determinada precisão. Dois resultados podem ser iguais para uma certa precisão e podem ser completamente distintos para uma outra precisão.

D.4 Instrução Plot

A instrução `Plot`, tal como em muitas outras linguagens, permite-nos representar graficamente funções fornecendo uma vasta gama de opções por forma a efectuar uma representação mais precisa e personalizada.

Um pormenor importante e intuitivo é o facto do `Plot` apenas poder representar os resultados com um número limitado de pontos, e, por vezes, omite bastante informação.

Já que o `Plot` precisa de avaliar os resultados várias vezes, é importante que essa avaliação seja efectuada tão rápida quanto possível. Como consequência, o Mathematica, em geral, compila a função a representar para um pseudo-código que é executado com bastante eficiência. Uma das principais desvantagens é que o pseudo-código apenas permite efectuar operações na precisão da máquina, o que praticamente impossibilita a análise gráfica de problemas de elevada sensibilidade.

Se a função que queremos representar graficamente exige operações de alta precisão devemos indicar no `Plot` a opção (`Compiled->False`).

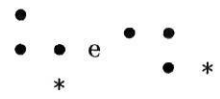
Apêndice E

Programas

Neste apêndice figura o programa principal correspondente ao novo algoritmo e a parte fundamental do programa que implementa uma decomposição LU, bem como as subrotinas utilizadas no cálculo dos aproximantes de Frobenius-Padé pertencentes a duas diagonais descendentes adjacentes da Tabela de Frobenius-Padé.

E.1 Frobenius_Pade.f90

O algoritmo descrito na secção 2.4 foi programado em Fortran90 implementando a biblioteca CADNA para o cálculo dos aproximantes de Frobenius-Padé, através de um processo recursivo utilizando intercaladamente as duas relações:




```
1     PROGRAM Frobenius_Pade
2
3     USE CADNAFREEF    ! Inicialização da biblioteca CADNA
4     IMPLICIT NONE
5     CHARACTER :: POf
6     INTEGER :: i, j, k, p, q
7     TYPE(DOUBLE_ST), DIMENSION(:,:), ALLOCATABLE :: a, b, d, e, g, h
8     TYPE(DOUBLE_ST), DIMENSION(:), ALLOCATABLE :: alpha, beta, gamma, f
9     TYPE(DOUBLE_ST) :: Delta, tau, lambda, eta, rho
10
11 !  Cria o ficheiro 'cadna_stability_f90.lst' que apresenta uma lista de
12 !  instabilidades do programa. '-1': listar todas as instabilidades
13    call CADNA_INIT(-1)
14
15 !  Definição dos dados
16    open(10, FILE='./dados.txt')
17
```

```
18 !   Linha da tabela a ser calculada
19     read (10,*) p
20 !   Coluna onde deve parar
21     read (10,*) J
22
23 !   Definição dos Polinómios Ortogonais
24     allocate (alpha(0:p+3*J), beta(0:p+3*J), gamma(0:p+3*J))
25     read (10,*) POf
26     call Polinomios_Ortogonais(alpha,beta,gamma,POf,p+3*J)
27
28 !   Coeficientes do desenvolvimento ortogonal da função a ser aproximada
29     allocate (f(0:p+3*J))
30     call Coeficientes_Ortogonais(f,POf,p+3*J)
31
32 !   Valores Iniciais
33     allocate (a(0:2*J,-1:p+J),b(0:2*J,-1:J),e(0:2*J,0:p+3*J))
34
35     do q=0,2*J
36
37         a(q,-1)=0.d0
38         b(q,-1)=0.d0
39         e(q,p+q)=0.d0
40
41         do i=0,p+q
42             e(q,i)=0.d0
42         enddo
44
45     enddo
46
47     do q=0,J
48         a(2*q,p+q+1)=0.d0
49         b(2*q,q)=1.d0
50         b(2*q,q+1)=0.d0
51     enddo
52
53     do q=1,J
54         a(2*q-1,p+q+1)=0.d0
55         b(2*q-1,q-1)=1.d0
56         b(2*q-1,q)=0.d0
57     enddo
58
59     do i=0,p
60         a(0,i)=f(i)
61     enddo
62
63     do i=p+1,p+3*J
64         e(0,i)=f(i)
```

```
65      enddo
66
67      do i=0,p+1
68          a(1,i)=f(i)
69      enddo
70
71      do i=p+2,p+3*J
72          e(1,i)=f(i)
73      enddo
74
75
76  !   [p+1/1]
77
78      allocate (d(0:2*J-2,p+2:p+3*J-1), g(1:2*J-2,0:p+J-1), h(1:2*J-2,0:J-1))
79
80      do i=p+2,p+3*J-1
81          d(0,i)=alpha(i-1)*f(i-1)+beta(i)*f(i)+gamma(i+1)*f(i+1)
82      enddo
83
84      b(2,0)=beta(0)/alpha(0)-d(0,p+2)/(f(p+2)*alpha(0))
85      a(2,0)=b(2,0)*f(0)+gamma(1)/alpha(0)*f(1)
86
87      do i=1,p+1
88          a(2,i)=(b(2,0)-beta(0)/alpha(0))*f(i)+(alpha(i-1)*f(i-1)+ &
89     &           beta(i)*f(i)+gamma(i+1)*f(i+1))/alpha(0)
90      enddo
91
92      do i=p+3,p+3*J-1
93          e(2,i)=(b(2,0)-beta(0)/alpha(0))*f(i)+d(0,i)/alpha(0)
94      enddo
95
96      open(40, FILE='./parametros_newcad.txt')
97      open(50, FILE='./parametros_algsig.txt')
98
99      do q=2,J
100
101 !   [p+q/q-1]
102
103         do i=p+2*q-2,p+3*J-q
104             d(2*q-3,i)=alpha(i-1)*e(2*q-3,i-1)+beta(i)*e(2*q-3,i)+gamma(i+1)* &
105    &                   e(2*q-3,i+1)
106         enddo
107
108         do i=0,p+q-1
109             g(2*q-3,i)=alpha(i-1)*a(2*q-3,i-1)+beta(i)*a(2*q-3,i)+gamma(i+1)* &
110    &                   a(2*q-3,i+1)
111         enddo
```

```
112
113         do i=0,q-2
114             h(2*q-3,i)=alpha(i-1)*b(2*q-3,i-1)+beta(i)*b(2*q-3,i)+gamma(i+1)* &
115     &                   b(2*q-3,i+1)
116         enddo
117
118         Delta=alpha(q-2)*e(2*q-4,p+2*q-3)*e(2*q-3,p+2*q-2)*e(2*q-2,p+2*q-1)- &
119     &        gamma(p+2*q-2)*e(2*q-3,p+2*q-2)*(e(2*q-4,p+2*q-2)*e(2*q-3,p+2*q-1)- &
120     &        e(2*q-3,p+2*q-2)*e(2*q-4,p+2*q-1))+e(2*q-4,p+2*q-3)* &
121     &        (d(2*q-3,p+2*q-2)*e(2*q-3,p+2*q-1)-e(2*q-3,p+2*q-2)*d(2*q-3,p+2*q-1))
122
123         tau=(e(2*q-4,p+2*q-3)*e(2*q-3,p+2*q-2)*e(2*q-2,p+2*q-1))/Delta
124         lambda=-gamma(p+2*q-2)*e(2*q-3,p+2*q-2)/e(2*q-4,p+2*q-3)*tau
125         eta=-(d(2*q-3,p+2*q-2)*tau+e(2*q-4,p+2*q-2)*lambda)/e(2*q-3,p+2*q-2)
126         rho=-(d(2*q-3,p+2*q-1)*tau+e(2*q-4,p+2*q-1)*lambda+e(2*q-3,p+2*q-1)*eta)/ &
127     &       e(2*q-2,p+2*q-1)
128
129         write (40,*) p+q
130         write (40,*) q-1
131         write (40,100) STR(tau)
132         write (40,100) STR(lambda)
133         write (40,100) STR(eta)
134         write (40,100) STR(rho)
135
136         write (50,*) p+q
137         write (50,*) q-1
138         write (50,*) CESTAC(tau)
139         write (50,*) CESTAC(lambda)
140         write (50,*) CESTAC(eta)
141         write (50,*) CESTAC(rho)
142
143         a(2*q-1,p+q)=alpha(p+q-1)*a(2*q-3,p+q-1)*tau
144         a(2*q-1,p+q-1)=eta*a(2*q-3,p+q-1)+tau*g(2*q-3,p+q-1)+rho*a(2*q-2,p+q-1)
145
146         do i=0,p+q-2
147             a(2*q-1,i)=lambda*a(2*q-4,i)+eta*a(2*q-3,i)+tau*g(2*q-3,i)+rho*a(2*q-2,i)
148         enddo
149
150         do i=0,q-2
151             b(2*q-1,i)=lambda*b(2*q-4,i)+eta*b(2*q-3,i)+tau*h(2*q-3,i)+rho*b(2*q-2,i)
152         enddo
153
154         do i=p+2*q,p+3*J-q
155             e(2*q-1,i)=lambda*e(2*q-4,i)+eta*e(2*q-3,i)+tau*d(2*q-3,i)+rho*e(2*q-2,i)
156         enddo
157
158
```

```
159 !   [p+q/q]
160
161         do i=p+2*q-1,p+3*J-q
162             d(2*q-2,i)=alpha(i-1)*e(2*q-2,i-1)+beta(i)*e(2*q-2,i)+gamma(i+1)* &
163     &                   e(2*q-2,i+1)
164         enddo
165
166         do i=0,p+q-1
167             g(2*q-2,i)=alpha(i-1)*a(2*q-2,i-1)+beta(i)*a(2*q-2,i)+gamma(i+1)* &
168     &                   a(2*q-2,i+1)
169         enddo
170
171         do i=0,q-1
172             h(2*q-2,i)=alpha(i-1)*b(2*q-2,i-1)+beta(i)*b(2*q-2,i)+gamma(i+1)* &
173     &                   b(2*q-2,i+1)
174         enddo
175
176         tau=1/alpha(q-1)
177         lambda=-gamma(p+2*q-1)*e(2*q-2,p+2*q-1)/e(2*q-3,p+2*q-2)*tau
178         eta=-(d(2*q-2,p+2*q-1)*tau+e(2*q-3,p+2*q-1)*lambda)/e(2*q-2,p+2*q-1)
179         rho=-(d(2*q-2,p+2*q)*tau+e(2*q-3,p+2*q)*lambda+e(2*q-2,p+2*q)*eta)/ &
180     &       e(2*q-1,p+2*q)
181
182         write (40,*) p+q
183         write (40,*) q
184         write (40,100) STR(tau)
185         write (40,100) STR(lambda)
186         write (40,100) STR(eta)
187         write (40,100) STR(rho)
188
189         write (50,*) p+q
190         write (50,*) q
191         write (50,*) CESTAC(tau)
192         write (50,*) CESTAC(lambda)
193         write (50,*) CESTAC(eta)
194         write (50,*) CESTAC(rho)
195
196         a(2*q,p+q)=alpha(p+q-1)*a(2*q-2,p+q-1)*tau+a(2*q-1,p+q)*rho
197
198         do i=0,p+q-1
199             a(2*q,i)=lambda*a(2*q-3,i)+eta*a(2*q-2,i)+tau*g(2*q-2,i)+rho*a(2*q-1,i)
200         enddo
201
202         b(2*q,q-1)=eta+tau*h(2*q-2,q-1)+rho
203
204         do i=0,q-2
205             b(2*q,i)=lambda*b(2*q-3,i)+eta*b(2*q-2,i)+tau*h(2*q-2,i)+rho*b(2*q-1,i)
```

```
206         enddo
207
208         do i=p+2*q+1,p+3*J-q
209             e(2*q,i)=lambda*e(2*q-3,i)+eta*e(2*q-2,i)+tau*d(2*q-2,i)+rho*e(2*q-1,i)
210         enddo
211
212     enddo
213
214     deallocate(d, g, h, alpha, beta, gamma)
215
216
217 !   Escreve os coeficientes dos aproximantes
218
219     open(20, FILE='./coefs_newcad.txt',ACCESS= 'APPEND')
220     do q=1,J
221         write(20,*) p+q
222         write(20,*) q-1
223
224 !       Numerador de [p+q/q-1]
225         do i=0,p+q
226             write(20,100) STR(a(2*q-1,i))
227         enddo
228
229 !       Denominador de [p+q/q-1]
230         do i=0,q-2
231             write(20,100) STR(b(2*q-1,i))
232         enddo
233
234 !       Coeficientes do erro
235         write(20,*) 3*(J-q)+1
236         do i=p+2*q,p+3*J-q
237             write(20,100) STR(e(2*q-1,i))
238         enddo
239
240
241         write(20,*) p+q
242         write(20,*) q
243
244 !       Numerador de [p+q/q]
245         do i=0,p+q
246             write(20,100) STR(a(2*q,i))
247         enddo
248
249 !       Denominador de [p+q/q]
250         do i=0,q-1
251             write(20,100) STR(b(2*q,i))
252         enddo
```

```
253
254 !       Coeficientes do erro
255         write(20,*) 3*(J-q)
256         do i=p+2*q+1,p+3*J-q
257             write(20,100) STR(e(2*q,i))
258         enddo
259     enddo
260
261
262 !   Escreve os algarismos significativos coeficientes dos aproximantes
263
264     open(30, FILE='./algsig_newcad.txt',ACCESS= 'APPEND')
265     do q=1,J
266         write(30,*) p+q
267         write(30,*) q-1
268
269 !       Numerador de [p+q/q-1]
270         do i=0,p+q
271             write(30,*) CESTAC(a(2*q-1,i))
272         enddo
273
274 !       Denominador de [p+q/q-1]
275         do i=0,q-2
276             write(30,*) CESTAC(b(2*q-1,i))
277         enddo
278
279 !       Coeficientes do erro
280         write(30,*) 3*(J-q)+1
281         do i=p+2*q,p+3*J-q
282             write(30,*) CESTAC(e(2*q-1,i))
283         enddo
284
285
286         write(30,*) p+q
287         write(30,*) q
288
289 !       Numerador de [p+q/q]
290         do i=0,p+q
291             write(30,*) CESTAC(a(2*q,i))
292         enddo
293
294 !       Denominador de [p+q/q]
295         do i=0,q-1
296             write(30,*) CESTAC(b(2*q,i))
297         enddo
298
299 !       Coeficientes do erro
```

```
300         write(30,*) 3*(J-q)
301         do i=p+2*q+1,p+3*J-q
302             write(30,*) CESTAC(e(2*q,i))
303         enddo
304     enddo
305
306     deallocate(a, b, e)
307
308     close(10)
309     close(20)
310     close(30)
311     close(40)
312     close(50)
313
314 100 FORMAT(10a22)
315
316     END PROGRAM Frobenius_Pade
```

E.2 Frobenius_Pade_LU.f90

As linhas apresentadas representam a parte fundamental do código do programa que implementa o algoritmo modificado, omitindo-se as instruções comuns com o programa original `Frobenius_Pade.f90`. Também este programa se encontram alterado no sentido de utilizar a biblioteca CADNA.

```
  1     PROGRAM Frobenius_Pade_LU
  2
  3     USE CADNAFREEF    ! Inicialização da biblioteca CADNA
  4     IMPLICIT NONE
  5     CHARACTER :: POf
  6     INTEGER :: i, j, k, p, q
  7     TYPE(DOUBLE_ST), DIMENSION(:,:), ALLOCATABLE :: a, b, d, e, g, h, mat
  8     TYPE(DOUBLE_ST), DIMENSION(:), ALLOCATABLE :: alpha, beta, gamma, f, indices, vecsol
  9     TYPE(DOUBLE_ST) :: tau, lambda, eta, rho, ntrocas
 10
...     Linhas do programa Frobenius_Pade.f90
 98
 99     do q=2,J
100
101 !   [p+q/q-1]
102
103         do i=p+2*q-2,p+3*J-q
104             d(2*q-3,i)=alpha(i-1)*e(2*q-3,i-1)+beta(i)*e(2*q-3,i)+gamma(i+1)* &
105     &                   e(2*q-3,i+1)
```

```
106         enddo
107
108         do i=0,p+q-1
109             g(2*q-3,i)=alpha(i-1)*a(2*q-3,i-1)+beta(i)*a(2*q-3,i)+gamma(i+1)* &
110     &                   a(2*q-3,i+1)
111         enddo
112
113         do i=0,q-2
114             h(2*q-3,i)=alpha(i-1)*b(2*q-3,i-1)+beta(i)*b(2*q-3,i)+gamma(i+1)* &
115     &                   b(2*q-3,i+1)
116         enddo
117
            allocate(mat(4,4), vecsol(4), indices(4))

            ! Construção da matriz e do vector solução.
            do i=1,4
                vecsol(i)=0.d0
                do k=1,4
                    mat(i,k)=0.d0
                enddo
            enddo

            vecsol(1)=1.d0

            mat(1,1)=alpha(q-2)
            mat(1,4)=1.d0
            mat(2,1)=gamma(p+2*q-2)*e(2*q-3,p+2*q-2)
            mat(2,2)=e(2*q-4,p+2*q-3)
            mat(3,1)=d(2*q-3,p+2*q-2)
            mat(3,2)=e(2*q-4,p+2*q-2)
            mat(3,3)=e(2*q-3,p+2*q-2)
            mat(4,1)=d(2*q-3,p+2*q-1)
            mat(4,2)=e(2*q-4,p+2*q-1)
            mat(4,3)=e(2*q-3,p+2*q-1)
            mat(4,4)=e(2*q-2,p+2*q-1)

            ! Decomposição LU e resolução do sistema.
            call Decomp_LU(indices,ntrocas,mat,4)
            call LU_BackSubs(vecsol,mat,indices,4)

            tau=vecsol(1)
            lambda=vecsol(2)
            eta=vecsol(3)
            rho=vecsol(4)

            deallocate(mat, vecsol, indices)
142
```

```
143         a(2*q-1,p+q)=alpha(p+q-1)*a(2*q-3,p+q-1)*tau
144         a(2*q-1,p+q-1)=eta*a(2*q-3,p+q-1)+tau*g(2*q-3,p+q-1)+rho*a(2*q-2,p+q-1)
145
146         do i=0,p+q-2
147             a(2*q-1,i)=lambda*a(2*q-4,i)+eta*a(2*q-3,i)+tau*g(2*q-3,i)+rho*a(2*q-2,i)
148         enddo
149
150         do i=0,q-2
151             b(2*q-1,i)=lambda*b(2*q-4,i)+eta*b(2*q-3,i)+tau*h(2*q-3,i)+rho*b(2*q-2,i)
152         enddo
153
154         do i=p+2*q,p+3*J-q
155             e(2*q-1,i)=lambda*e(2*q-4,i)+eta*e(2*q-3,i)+tau*d(2*q-3,i)+rho*e(2*q-2,i)
156         enddo
157
158
159 !   [p+q/q]
160
161         do i=p+2*q-1,p+3*J-q
162             d(2*q-2,i)=alpha(i-1)*e(2*q-2,i-1)+beta(i)*e(2*q-2,i)+gamma(i+1)* &
163     &                   e(2*q-2,i+1)
164         enddo
165
166         do i=0,p+q-1
167             g(2*q-2,i)=alpha(i-1)*a(2*q-2,i-1)+beta(i)*a(2*q-2,i)+gamma(i+1)* &
168     &                   a(2*q-2,i+1)
169         enddo
170
171         do i=0,q-1
172             h(2*q-2,i)=alpha(i-1)*b(2*q-2,i-1)+beta(i)*b(2*q-2,i)+gamma(i+1)* &
173     &                   b(2*q-2,i+1)
174         enddo
175
            allocate(mat(4,4), vecsol(4), indices(4))

            ! Construção da matriz e do vector solução.
            do i=1,4
                vecsol(i)=0.d0
                do k=1,4
                    mat(i,k)=0.d0
                enddo
            enddo

            vecsol(1)=1.d0

            mat(1,1)=alpha(q-1)
            mat(2,1)=gamma(p+2*q-1)*e(2*q-2,p+2*q-1)
```

```
        mat(2,2)=e(2*q-3,p+2*q-2)
        mat(3,1)=d(2*q-2,p+2*q-1)
        mat(3,2)=e(2*q-3,p+2*q-1)
        mat(3,3)=e(2*q-2,p+2*q-1)
        mat(4,1)=d(2*q-2,p+2*q)
        mat(4,2)=e(2*q-3,p+2*q)
        mat(4,3)=e(2*q-2,p+2*q)
        mat(4,4)=e(2*q-1,p+2*q)

        ! Decomposição LU e resolução do sistema.
        call Decomp_LU(indices,ntrocas,mat,4)
        call LU_BackSubs(vecsol,mat,indices,4)

        tau=vecsol(1)
        lambda=vecsol(2)
        eta=vecsol(3)
        rho=vecsol(4)

        deallocate(mat, vecsol, indices)
195
196     a(2*q,p+q)=alpha(p+q-1)*a(2*q-2,p+q-1)*tau+a(2*q-1,p+q)*rho
197
198     do i=0,p+q-1
199         a(2*q,i)=lambda*a(2*q-3,i)+eta*a(2*q-2,i)+tau*g(2*q-2,i)+rho*a(2*q-1,i)
200     enddo
201
202     b(2*q,q-1)=eta+tau*h(2*q-2,q-1)+rho
203
204     do i=0,q-2
205         b(2*q,i)=lambda*b(2*q-3,i)+eta*b(2*q-2,i)+tau*h(2*q-2,i)+rho*b(2*q-1,i)
206     enddo
207
208     do i=p+2*q+1,p+3*J-q
209         e(2*q,i)=lambda*e(2*q-3,i)+eta*e(2*q-2,i)+tau*d(2*q-2,i)+rho*e(2*q-1,i)
210     enddo
211
212 enddo
213
... Linhas do programa Frobenius_Pade.f90
315
316 END PROGRAM Frobenius_Pade_LU
```

E.3 Polinomios_Ortogonais.f90

Os vectores $\{\alpha\}_{i\geq 0}$, $\{\beta\}_{i\geq 0}$ e $\{\gamma\}_{i\geq 0}$ constituem os coeficientes da relação de recorrência $P_{i+1} = \frac{z-\beta_i}{\alpha_i} P_i + \frac{\gamma_i}{\alpha_i} P_{i-1}$. O parâmetro F define a família de polinómios ortogonais.

```
1     subroutine Polinomios_Ortogonais(M,a,b,c,F)
2
3     USE CADNAFREEF
4     IMPLICIT NONE
5     CHARACTER F
6     INTEGER M, i
7     TYPE(DOUBLE_ST) :: a(0:M), b(0:M), c(0:M)
8
9     select case (F)
10
11    case ('P')    ! Polinómios de Legendre - P
12    do i=0,M
13        a(i)=.5*(i+1)/(i+.5)
14        b(i)=0.d0
15        c(i)=.5*i/(i+.5)
16    enddo
17
18    case ('T')    ! Polinómios de Chebyshev 1a espécie - T
19    do i=0,M
20        a(i)=5.d-1
21        b(i)=0.d0
22        c(i)=5.d-1
23    enddo
24
25    case ('U')    ! Polinómios de Chebyshev 2a espécie - U
26    do i=0,M
27        alpha(i)=25.d-2
28        beta(i)=5.d-1
29        gamma(i)=25.d-2
30    enddo
31
32    case ('L')    ! Polinómios de Laguerre - L (alfa=0)
33    do i=0,M
34        a(i)=-(i+1.d0)
35        b(i)=2.d0*i+1
36        c(i)=-1.d0*i
37    enddo
38
39    case ('H')    ! Polinómios de Hermite - H
40    do i=0,M
41        a(i)=5.d-1
42        b(i)=0.d0
```

```
43          c(i)=i
44      enddo
45
46      case ('B')     ! Polinómios de Bessel - B
47      a(0)=1.d0
48      b(0)=-1.d0
49      c(0)=1.d0/((2*i-1)*(2*i+1))
50      do i=1,M
51          a(i)=1.d0
52          b(i)=0.d0
53          c(i)=-1.d0/((2*i-1)*(2*i+1))
54      enddo
55
56      case default  !  Potências de x
57      do i=0,M
58          a(i)=1.d0
59          b(i)=0.d0
60          c(i)=0.d0
61      enddo
62
63      end select
64
65      return
66      END SUBROUTINE Polinomios_Ortogonais
```

E.4 Coeficientes_Ortogonais.f90

Subrotina de cálculo dos coeficientes do desenvolvimento ortogonal da função a ser aproximada. O parâmetro COf define a função a considerar para uma determinada família de polinómios ortogonais.

```
1       SUBROUTINE Coeficientes_Ortogonais(M,COf,f)
2
3       USE CADNAFREEF      ! Inicialização da biblioteca CADNA
4       IMPLICIT NONE
5       CHARACTER, INTENT(IN) :: COf
6       INTEGER, INTENT(IN) :: M
7       INTEGER :: k
8       TYPE(DOUBLE_ST), DIMENSION(0:M), INTENT(OUT) :: f(0:M)
9       TYPE(DOUBLE_ST) :: pi
10      TYPE(DOUBLE_ST) :: a
11      REAL, PARAMETER :: pi_aux=3.1415927
12      REAL :: a_aux
13
14      select case (COf)
15
```

```
16      case ('P')    ! Polinómios de Legendre - P
17      open (20, FILE='./parametro.txt')
18      read(20,*) a_aux
19      close (20)
20      a=a_aux
21      f(0)=1.d0
22      do k=1,M
23          f(k)=a*f(k-1)
24      enddo
25
26      case ('U')    ! Polinómios de Chebyshev 2a espécie - U
27      pi=pi_aux
28      a_aux=16/pi
29      a=a_aux
30      do k=0,M
31          f(k)=a*(-1)**k*(k+1)/((2*k+1)*(2*k+3))
32      enddo
33
34      case ('L')    ! Polinómios de Laguerre - L (alfa=0)
35      f(0)=1.d0
36      do k=1,M
37          f(k)=-f(k-1)
38      enddo
39
40      case ('H')    ! Polinómios de Hermite - H
41      f(0)=1.d0
42      do k=1,M
43          f(k)=f(k-1)/k
44      enddo
45
46      case ('B')    ! Polinómios de Bessel - B
47      f(0)=1.d0
48      do k=1,M
49          f(k)=f(k-1)/(2*k)
50      enddo
51
52      end select
53
54      return
55      END SUBROUTINE Coeficientes_Ortogonais
```

E.5 idb.in

Ficheiro utilizado para a execução do *debugger*.

```
1     stop in instability
2     run
3     where
4     cont
5     where
6     cont

...

      ! Tantas vezes quanto o número de instabilidades a localizar.
      where
      cont

...

   where
   cont
   end
```

Referências

[1] George A. Baker, Jr. and Peter Graves-Morris. Padé approximants. In Gian-Carlo Rota, editor, *Encyclopedia of Mathematics and its Applications*, volume 59. Cambridge University Press, 1996.

[2] Claude Brezinski. Rational approximation to formal power series. *Journal of Approximation Theory*, 25:295–317, 1979.

[3] Claude Brezinski. *Padé-Type Approximation and General Orthogonal Polynomials.* Birkhäuser, Basel, 1980.

[4] Claude Brezinski and Jeannette Van Iseghem. Padé approximations. In P.G. Ciarlet and J.L. Lions, editors, *Handbook of Numerical Analysis*, volume III. North-Holland, Amsterdam, 1994.

[5] Jean-Marie Chesneaux and Jean Vignes. *CADNA for FORTRAN source codes - User's Guide.* http://www.lip6.fr/Cadna, 2002.

[6] Jean-Marie M. Chesneaux and Ana C. Matos. Breakdown and near-breakdown control in the CGS algorithm using stochastic arithmetic. *Numerical Algorithms*, 11(1-4):99–116, 1996.

[7] Zélia da Rocha. *Applications de la Théorie de la Biorthogonalité.* Tese de Doutoramento, Université des Sciences et Technologies de Lille, Fevereiro 1994.

[8] Zélia da Rocha and José M.A. Matos. Frobenius-Padé Approximants for d-ortogonal series: theory and computational aspects. A ser publicado em *Applied Numerical Mathematics.*

[9] Manuel Rogério da Silva. *Sebenta de Análise Numérica I.* Departamento de Matemática Aplicada da Faculdade de Ciências da Universidade do Porto, 1980.

[10] Philip J. Davis. *Interpolation and Approximation.* Dover, New York, 1975.

[11] Urs W. Hochstrasser. Orthogonal polynomials. In Milton Abramowitz and Irene A. Stegun, editors, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, pages 771–802. Dover, New York, 1970.

[12] Jonas T. Holdeman, Jr. A method for the approximation of functions defined by formal series expansions in orthogonal polynomials. *Mathematics of Computation*, 23:275–287, 1969.

[13] Yudell L. Luke. The special functions and their approximations. In Richard Bellman, editor, *Mathematics in Science and Engineering*, volume 53-I. Academic Press, 1969.

[14] P. Maroni. Une théorie algébrique des polynômes orthogonaux. Application aux polynômes orthogonaux semi-classiques. In C. Brezinski, L. Gori, and A. Ronveaux, editors, *Orthogonal Polynomials and their applications*, IMACS, pages 95–130. J.C. Baltzer AG, Scientific Publishing Co, 1991.

[15] P. Maroni. Variations around classical orthogonal polynomials. Connected problems. *Jornal of Computational Applied Mathematics*, 48:133–155, 1993.

[16] P. Maroni. Fonctions eulériennes. Polynômes orthogonaux classiques. *Techniques de L'Ingénieur, Traité Généralités, Sciences Fondamentales*, 48:133–155, 1994.

[17] Ana C. Matos. Recursive computation of Padé-Legendre approximants and some acceleration properties. *Numerische Mathematik*, 89(3):535–560, 2001.

[18] José M.A. Matos. *Algoritmos de cálculo dos aproximantes de Frobenius-Padé e generalizações.* Tese de Doutoramento, Departamento de Matemática Aplicada da Faculdade de Ciências da Universidade do Porto, 2003.

[19] Luís T. Paiva. Estabilidade numérica no cálculo recursivo dos aproximantes de Frobenius-Padé: utilização do CADNA. Relatório de Seminário, Departamento de Matemática Aplicada, Faculdade de Ciências da Universidade do Porto, Junho 2003.

[20] William H. Press. *Numerical Recipes in Fortran 90.* Cambridge University Press, 2^{nd} edition, 1996.

[21] Maria Raquel Valença. *Análise Numérica.* Universidade Aberta, 1996.

[22] Stephen Wolfram. *The Mathematica Book.* Wolfram Research, Cambridge, 4^{th} edition, 2001.