UNIVERSIDADE TÉCNICA DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

$$y(t) - \int_0^t p(t,s)y(s)ds = g(t)$$

Métodos Numéricos para Equações Integrais de Volterra com Núcleo Singular

Maria Isabel da Conceição Santos Reis dos Santos
(Licenciada)

dissertação para a obtenção do grau de mestre em
Matemática Aplicada

JANEIRO 1996

Métodos Numéricos para Equações Integrais de Volterra com Núcleo Singular

Maria Isabel da Conceição Santos Reis dos Santos

Tese submetida para provas de mestrado em
Matemática Aplicada

Departamento de Matemática

Instituto Superior Técnico

Lisboa

15 de Janeiro de 1996

Tese realizada sob a orientação da

Professora Doutora Maria Teresa Romãozinho Marques Diogo

e do

Professor Doutor Pedro Miguel Rita da Trindade e Lima

Professores Auxiliares do Departamento de Matemática

Instituto Superior Técnico

Nome: Maria Isabel da Conceição Santos Reis dos Santos

Título: Métodos Numéricos para Equações Integrais de Volterra com Núcleo Singular

Palavras chave

Métodos Numéricos
Equações Integrais
Métodos de Colocação
Aceleração de Convergência

Keywords

Numerical Methods
Integral Equations
Collocation Methods
Convergence Acceleration

RESUMO

Palavras chave - Métodos Numéricos, Equações Integrais,
Métodos de Colocação, Aceleração de Convergência.

Este trabalho trata a solução numérica de equações integrais de Volterra por métodos de colocação em espaços de funções seccionalmente polinomiais (*splines*). Em particular, é considerada uma equação integral de Volterra de segunda espécie com núcleo fracamente singular, para a qual provamos um resultado de existência e unicidade de solução.

Alguns métodos de ordem pouco elevada, nomeadamente os métodos de Euler e ponto médio (ordem um) e o método dos trapézios (ordem dois) são aplicados à referida equação usando técnicas de integração produto. A convergência desses métodos é analisada teoricamente e ilustrada com alguns exemplos numéricos. Obtemos ainda um resultado sobre a estabilidade do método de Euler.

Finalmente, com o objectivo de melhorar a precisão das aproximações obtidas pelo método de Euler, é investigada a aplicação de técnicas de extrapolação. São utilizados os algoritmos E e Richardson, conseguindo-se resultados melhores comparados com métodos de colocação de ordens mais elevadas, tais como o método dos trapézios e Simpson, com menor esforço computacional.

ABSTRACT

Keywords - Numerical Methods, Integral Equations,
Collocation Methods, Convergence Acceleration.

This work is concerned with the numerical solution of Volterra integral equations by collocation methods in certain polynomial splines spaces. In particular, a second kind Volterra integral equation with weakly singular kernel is considered and an existence and uniqueness result is obtained.

Some low order product integration methods, which include the so-called Euler method and the midpoint method (both of order one), and the trapezoidal method (order two) are applied to the above refered to equation. The convergence of these methods is analysed and illustrated by means of some numerical examples. A result concerning the stability of the Euler method is also obtained.

Finally, in order to improve the low accuracy of the approximations obtained by the Euler method, the application of certain extrapolation techniques is investigated. The E-algorithm and Richardson method are applied, achieving approximations better than those obtained with higher order product integration methods, such as the trapezoidal and Simpson methods, with less computational effort.

Índice

Lista de Figuras

Lista de Tabelas

Agradecimentos

Pela contribuição dada para a execução deste trabalho, pretendo expressar o meu reconhecimento muito particular:

Aos Profs. Teresa Diogo e Pedro Lima na qualidade de meus orientadores científicos e, muito especialmente, pela sua constante disponibilidade e encorajamento. O seu acompanhamento empenhado tem sido fundamental para a minha formação científica e profissional.

Ao Instituto Superior Técnico pelas condições de trabalho únicas que me facultou e pelo apoio informático imprescindível à realização deste trabalho.

Notação

Qualquer somatório com o seu limite superior menor que o seu limite inferior é considerado zero.

Qualquer produto com o seu limite superior menor que o seu limite inferior é considerado um.

$:=$	igual por definição.
$=$	identicamente igual.
$\approx$	aproximadamente igual.
$f(t) = O(g(t))$	$f(t)$ é da ordem $g(t)$ quando t se aproxima de a, isto é,
$quando\, t \to a$	$f(t)/g(t)$ é limitado quando $t \to a$.
	Ao símbolo O é chamado símbolo O de Landau.
$f(t) \sim g(t)$	$f(t)$ tem o comportamento assimptótico como $g(t)$
$quando\, t \to a$	quando t se aproxima de a: O erro entre f e g tende
	para zero quando $t \to a$ (i.e., $\lim_{t\to a}(f(t)/g(t)) = 1$).
$A \cup B$	união dos conjuntos A e B.
$\aleph$	$:= \{1, 2, \ldots\}$, o conjunto dos números naturais.
$\Re$	o conjunto dos números reais
$[a, b]$	$:= \{x \in \Re : a \leq x \leq b\}$, intervalo fechado.
(a, b)	$:= \{x \in \Re : a < x < b\}$, intervalo aberto.
$(a, b], [a, b)$	intervalo aberto à esquerda, aberto à direita.
I	$:= [0, T]$, $t > 0$ e T finito.
S	$:= \{(t, s) \in \Re : 0 \leq s \leq t \leq T\}$.
y_n	$:= y(t_n)$, valor da função y no ponto t_n.
π_m	conjunto de polinómios reais de grau menor ou
	igual a m.
$C[a, b], C(I)$	é o espaço das funções reais de valor real que são
	contínuas em $[a, b]$ ou em I.
$C^r[a, b], C^r(I)$	é o espaço das funções reais de valor real que são r
	vezes continuamente diferenciáveis em $[a, b]$ ou em I.
$\|\mathbf{x}\|_\infty$	$:= \max_{1\leq i\leq m} \|x_i\|$ denota a norma do máximo ou,
	norma em l_∞ para $\mathbf{x} := (x_1, \ldots, x_m)^T \in \Re^m$.
$exp(t)$	representa o valor da função exponencial para t.
$log(t)$	é o logaritmo natural de t.
$sinal(x)$	denota o sinal de x. É igual a $-$ de x fôr negativo e
	é igual a $+$ se x fôr positivo.

$\sum_{j=k}^{m} a_j$	é por definição igual a 0 se $m < k$ e igual a $a_k + \ldots + a_m$ se $k \leq m$.
$\prod_{j=k}^{m} a_j$	é por definição igual a 1 se $m < k$ e igual a $a_k \ldots a_m$ se $k \leq m$.
Π_N	denota a partição, ou rede, para o intervalo I da forma $0 = t_0 < t_1 < \ldots < t_N = T$ (onde o índice N indica a dependência dos pontos da rede)
Π_h	denota a partição, ou rede, para o intervalo I da forma $a = t_0 < t_1 < \ldots < t_N = b$ (onde o índice N indica a dependência dos pontos da rede)
h	$:= \max_{0 \leq n \leq N-1} h_n$, com $h_n := t_{n+1} - t_n$, representa o diâmetro da rede Π_N. A rede Π_N diz-se uniforme se $h_n = h$ para todo o n.
$h \downarrow 0$ ($h \uparrow 0$)	significa que h tende para 0 por valores reais positivos (negativos).

Capítulo 1

Introdução

Desde o trabalho de Abel, em 1823, os especialistas em análise numérica têm mostrado um interesse contínuo em equações integrais. Os nomes de muitos matemáticos conhecidos, entre os quais Cauchy, Fredholm, Hilbert, Volterra e outros, estão associados a esta área. Existem duas razões principais para este interesse. Em alguns casos, como no trabalho de Abel, as equações integrais são o modelo natural para representar diversas situações físicas interessantes. Por outro lado, os operadores integrais, transformadas, e equações integrais são ferramentas úteis dos analistas clássicos para o estudo das equações diferenciais.

Na modelação matemática a ênfase tem sido posta nas equações diferenciais. Recentemente, contudo, esta situação tem começado a alterar-se. O desenvolvimento de Análise Matemática na área da teoria assimptótica tem permitido um maior conhecimento das propriedades das soluções das equações integrais. Este facto tem conduzido ao uso mais frequente dessas equações para modelos matemáticos, o que, por sua vez tem impulsionado o estudo de novos métodos numéricos para equações integrais. Em outros casos certas equações diferenciais parciais em duas variáveis são equivalentes a equações integrais numa variável. Vários exemplos encontram-se em problemas de condução de calor e difusão (ver [31] pp. 23). Este

facto tem conduzido a avanços em métodos para resolver equações integrais de fronteira, facto que tem atraído muitos engenheiros.

1.1 Classificação das Equações de Volterra

Uma equação integral é uma equação onde a incógnita, geralmente uma função de uma ou mais variáveis, ocorre sob o sinal de integral. Esta definição, algo geral, permite muitas formas específicas e, na prática, surgem muitos tipos distintos. A teoria clássica das equações integrais distingue entre equações de Fredholm e equações de Volterra. Nas equações de Fredholm a região de integração é fixa, enquanto nas equações de Volterra a região de integração depende de alguma forma das variáveis independentes.

Uma equação da forma

$$y(t) - \int_a^t K(t, s, y(s))ds = g(t), \qquad a \leq t \leq T < \infty,$$

é uma equação de Volterra de *segunda espécie*. Neste caso a incógnita é $y(s)$. O segundo membro da equação, $g(t)$, e o *núcleo* $K(t, s, y)$ supõem-se conhecidos. Esta equação é uma das várias formas da equação de Volterra.

Em muitas aplicações práticas tem interesse o comportamento da solução em todo o eixo real. No entanto, em cálculos numéricos é necessário em qualquer caso considerar T finito.

Para simplificar a notação podemos, sem perda de generalidade, escolher o domínio da variável independente tal que o limite inferior seja 0 e considerar apenas a equação

$$y(t) - \int_0^t K(t, s, y(s))ds = g(t), \qquad 0 \leq t \leq T \leq \infty. \tag{1.1}$$

Uma equação de Volterra de *primeira espécie* é uma equação da forma

$$\int_0^t K(t, s, y(s))ds = g(t), \qquad 0 \leq t \leq T \leq \infty. \tag{1.2}$$

Sob certas condições de diferenciabilidade esta equação pode, contudo, ser reduzida a uma equação de segunda espécie, por exemplo por diferenciação.

No caso em que o núcleo tem a forma

$$K(t, s, y(s)) = k(t, s)y(s). \tag{1.3}$$

a equação correspondente diz-se linear.

Equações com núcleos singulares têm interesse prático. Quando o núcleo $K(t,s,y)$ é singular, podemos escrevê-lo na forma

$$K(t,s,y) = p(t,s)k(t,s,y)$$

onde $p(t,s)$ representa a parte com comportamento não regular. A equação correspondente tem a forma

$$y(t) = g(t) + \int_0^t p(t,s)k(t,s,y(s))ds, \qquad 0 \leq t \leq T. \tag{1.4}$$

Equações com núcleos integráveis mas não limitados são chamadas equações integrais com singularidades fracas. Como caso particular de núcleo fracamente singular é de interesse o seguinte

$$K(t,s,y(s)) = (t-s)^{-\alpha}k(t,s,y), \qquad 0 < \alpha < 1. \tag{1.5}$$

A equação integral correspondente é chamada equação de Abel generalizada.

1.2 Relação com Problemas de valor inicial

As equações integrais são usadas no estudo das propriedades das equações diferenciais. Uma equação diferencial da forma

$$y'(t) = F(t,y(t)), \quad t \geq 0, \quad y(0) = y_0, \tag{1.6}$$

pode ser convertida por integração na equação de Volterra

$$y(t) = y_0 + \int_0^t F(s,y(s))ds. \tag{1.7}$$

Este é frequentemente o ponto de partida para a exploração das propriedades qualitativas da solução da equação diferencial, por exemplo, no que diz respeito à existência e unicidade.

1.3 Existência e Unicidade de Solução

Em casos extremamente simples, é possível encontrar soluções para equações de Volterra convertendo a equação integral em diferencial. Isto só é possível quando o núcleo da equação integral exibe propriedades especiais podendo-se obter um sistema equivalente de equações diferenciais ordinárias. Os teoremas enunciados nesta secção estão demonstrados em [31] (pp. 52-53 e pp. 62-63).

Vamos estudar a solução da equação (1.1) considerando várias hipóteses sobre o núcleo. Os resultados de existência e unicidade são aqui referidos admitindo que $K(t,s,y)$ satisfaz uma condição de Lipschitz em relação ao terceiro argumento. O uso desta condição simplifica a análise mas torna a aplicação dos resultados limitada. Em muitos casos a condição de Lipschitz não é satisfeita, e aí temos de examinar um comportamento mais geral. Notemos contudo, que no caso de uma equação linear, ou seja, com o núcleo da forma (1.3), aquela condição é automaticamente satisfeita.

Teorema 1.1 *Se na equação (1.1) se verificarem as condições*

(i) $K(t,s,y)$ *é contínuo em* $0 \leq s \leq t \leq T$, $-\infty < y < \infty$,

(ii) $g(t)$ *é contínuo em* $0 \leq t \leq T$,

(iii) $K(t,s,y)$ *satisfaz uma condição de Lipschitz em* y, *isto é, para quaisquer* u_1 *e* u_2 *reais finitos,* $|K(t,s,u_1) - K(t,s,u_2)| \leq L|u_1 - u_2|$, *onde* L *é independente de* t, s, u_1 *e* u_2,

então a equação integral (1.1) possui uma única solução contínua em $0 \leq t \leq T$, *para todo o* T *finito.* □

No teorema anterior assumimos que o núcleo é limitado. De uma forma mais geral, quando ele é singular temos o seguinte resultado:

Teorema 1.2 *Considere-se a equação (1.4) onde*

(i) $k(t,s,y)$ *é contínuo em* $0 \leq s \leq t \leq T$, $-\infty < y < \infty$,

(ii) $g(t)$ *é contínuo em* $0 \leq t \leq T$,

(iii) *a condição de Lipschitz* $|k(t,s,u_1) - k(t,s,u_2)| \leq L|u_1 - u_2|$, *é satisfeita para* $0 \leq s \leq t \leq T$ *e todo o* u_1, $u_2 \in \Re$,

(iv) *para cada função contínua* h *e todo o* $0 \leq \tau_1 \leq \tau_2 \leq t$ *os integrais*

$$\int_{\tau_1}^{\tau_2} p(t,s)k(t,s,h(s))ds$$

e

$$\int_0^t p(t,s)k(t,s,h(s))ds$$

são funções contínuas em t,

(v) $p(t,s)$ *é absolutamente integrável em relação a* s *para todo o* $0 \leq t \leq T$,

(vi) *existem pontos* $0 = T_0 < T_1 < \ldots < T_N = T$ *tal que com* $t \geq T_i$

$$L \int_{T_i}^{min(t,T_{i+1})} |p(t,s)|ds \leq \alpha < 1,$$

(vii) *para todo o* $t \geq 0$

$$\lim_{\delta \to 0} \int_t^{t+\delta} |p(t+\delta,s)|ds = 0,$$

então a equação integral (1.4) possui uma única solução contínua em $0 \leq t \leq T$. □

1.4 Aplicações das Equações de Volterra

Antes de passarmos ao estudo das equações de Volterra procuraremos nesta secção apresentar algumas das suas aplicações práticas. O objectivo desta secção é dar a conhecer a utilidade das equações de Volterra sem dar excessiva atenção ao pormenor.

As equações de Volterra nascem naturalmente em certo tipo de problemas dependentes do tempo onde o comportamento no instante t depende não apenas do estado neste instante, mas também dos estados nos instantes anteriores. A solução da equação diferencial

$$y'(t) = f(t, y(t))$$

é completamente determinada para $t > t_0$ se for conhecido $y(t_0)$. Isto verifica-se para qualquer t_0. Informação anterior a $t = t_0$ é irrelevante para a solução após t_0. Existem, contudo, situações onde, para predizer a evolução futura do problema, temos que conhecer não apenas o estado corrente mas também a evolução até ao estado $y(t_0)$. Tais modelos são algumas vezes chamados sistemas com memória ou

problemas historico-dependentes. Se a dependência histórica puder ser representada por um termo da forma

$$\int_0^t K(t,s,y(s))ds,$$

então a equação modeladora é de Volterra (ver [31] pp. 14). Exemplos encontram-se em aplicações na teoria de sistemas mecânicos, circuitos eléctricos e sistemas de computadores (ver [31]).

As equações integrais nascem também em algumas situações onde as observações experimentais não se revelam ser a variável de interesse mas sim algum seu integral. Para calcular o actual valor dessa variável é necessário descobrir a solução de equações integrais. Alguns exemplos encontram-se em problemas de inferência experimental (ver [31]).

Nesta secção procuraremos apresentar o estado actual das equações de Volterra, não de um ponto de vista histórico mas, de um ponto de vista de área de aplicação prática. Para tal referiremos alguns trabalhos realizados em certas áreas onde as equações integrais de Volterra têm sido utilizadas.

Existem vários livros e artigos que tratam de aplicações práticas das equações de Volterra. Damos aqui algumas referências que dão ao leitor uma introdução para uma vasta literatura.

Condução de Calor

Com o envelhecimento do aço, os átomos de carbono móveis difundem-se através das fronteiras com uma velocidade várias ordens de grandeza superior aos átomos de crómio. Este efeito permite reduzir um sistema de duas equações de difusão a uma só equação integral de Volterra de segunda espécie. Este problema de condução de calor foi apresentado por Mann e Wolf em 1951 [33].

Quando um avião, projectil ou outro corpo percorre a atmosfera, mesmo a velocidades várias vezes superiores à do som, as camadas superficiais frontais permanecem laminares. Lighthill, em 1950 [25], ao equacionar a transferência de calor à superfície do projectil com a sua radiação obteve uma equação integral de Volterra não linear com núcleo singular, para velocidades várias vezes superiores à do som.

Discussão sobre equações da dinâmica de reactores nucleares pode ser encontrada em Levin e Nohel (1962) (ver [31] pp. 26).

Superfluidez

A teoria da superfluidez e da transferência de calor entre sólidos e gases foi tratada por Levinson em 1960 [24]. Neste trabalho, Levinson refere que certas situações encontradas na teoria da superfluidez conduzem a problemas mais complexos que os encontrados na transferência de calor entre sólidos e gases em condições fronteira não lineares, como é o caso do trabalho de Mann e Wolf em 1951 [33]. No entanto, e tal como no caso estudado por Mann e Wolf também aqui se pode exprimir o problema através de uma equação integral de Volterra não linear. Na mesma área podemos ainda encontrar os trabalhos de Roberts e Mann em 1951 [38], e Handlesman e Olmstead em 1972 [19].

Viscoelasticidade

A determinação da distribuição da tensão numa folha de vidro, quando arrefecida simetricamente em ambas as superfícies, conduz a uma equação integro-diferencial de Volterra. A viscoelasticidade das camadas centrais do vidro causam uma pressão excessiva que pode conduzir à deformação do vidro. Em [31] (pp. 26-27) encontram-se referências aos trabalhos de Rogers e Lee, Cameron e McKee, e outros.

Electricidade e Electrónica

O estudo do movimento de contacto entre um comboio eléctrico com a catenária foi estudado por Heyda e Thrackray, em 1981. A equação que descreve o deslocamento do ponto de contacto devido à força exercida pelo movimento é uma equação de Volterra de primeira ordem [35].

A modelação, da difusão lateral das implantações iónicas em semi-condutores conduz a uma equação de Abel com núcleo singular [35]. Esta equação foi estudada pelo próprio Abel, em 1881, tendo mostrado a existência de uma solução analítica.

1.5 Estrutura do Trabalho

O texto, aqui apresentado, que se segue foi organizado procurando, tanto quanto possível, oferecer em cada capítulo uma abordagem de complexidade e especificidade crescente.

Neste capítulo, 'Introdução', pretendeu-se não só apresentar as equações de Volterra, que iremos tratar de uma forma geral, bem como um conjunto de situações nas quais este tipo de equações são utilizadas para modelar problemas reais.

No capítulo seguinte — 'Métodos Numéricos para Equações Integrais' — analisam-se os métodos de integração numérica aplicáveis a equações do tipo que iremos estudar. Por serem particularmente importantes para este trabalho, dá-se, igualmente, relevo aos chamados métodos de quadratura directa, métodos de colocação nos espaços de splines polinomiais $S_r^d(Z_N)$ e $\tilde{S}_r^d(Z_N)$ (introduzidos também neste capítulo) e às fórmulas númericas de quadratura.

No capítulo 'Estudo de uma Equação Integral' apresenta-se o problema que nos propomos explorar em maior detalhe e que consiste num subtipo de equações de Volterra de segunda espécie onde o núcleo apresenta uma singularidade fraca, para o qual se obtém um resultado de existência e unicidade. Seguidamente, fazem-se algumas considerações teóricas e efectuam-se experiências com exemplos numéricos para métodos de colocação em espaços de splines polinomiais $\tilde{S}_r^d(Z_N)$ de ordens 0, 1 e 2 e $S_r^d(Z_N)$ de ordem 0. São obtidos resultados teóricos de convergência para os métodos de ordem pouco elevada como por exemplo Euler, ponto médio e Trapézios.

As soluções obtidas no capítulo anterior são sujeitas, no capítulo 'Aceleração de Convergência', a algoritmos de extrapolação. Estes algoritmos podem permitir obter resultados mais precisos, procurando-se oferecer uma abordagem a alguns destes algoritmos quer do ponto de vista teórico quer através de exemplos numéricos.

Por fim, apresentam-se algumas conclusões sobre a aplicação dos métodos realizados bem como testes de desempenho e perspectivas de trabalho futuro na mesma área face aos resultados obtidos.

Incluem-se ainda, em apêndice, as listagens dos programas de integração e aceleração numérica realizados. Na bibliografia podemos encontrar referências a livros e publicações consultados na execução deste trabalho e que representam um complemento para a compreensão dos problemas e soluções que surgem na resolução numérica de equações de Volterra.

Capítulo 2

Métodos Numéricos para Equações Integrais

Este capítulo apresenta alguns dos métodos numéricos utilizados para a resolução de equações de Volterra. Por serem particularmente importantes para este trabalho, dá-se relevo aos chamados métodos de quadratura directa e métodos de colocação nos espaços de splines polinomiais $S_r^d(Z_N)$. Como para a aplicação destes métodos é em geral necessário utilizar fórmulas numéricas de quadratura, na secção 2.1 fazemos referência a resultados clássicos da teoria de integração numérica. Para estudos detalhados sobre a resolução numérica de equações integrais de Volterra, referimos [1], [7], [12] e[31] (ver também [1], [5], [27]).

2.1 Alguns Resultados sobre Integração Numérica

As definições e teoremas apresentados nesta secção podem ser encontrados em [7], [23], [11], [44], [21], [12], [13] e [27].

Considere-se o integral definido

$$I(\phi) := \int_a^b \phi(t)dt, \qquad |a| < \infty, \quad |b| < \infty, \quad a < b, \tag{2.1}$$

a aproximemo-lo através da fórmula numérica

$$I_n(\phi) := \sum_{j=0}^{n} c_{n,j}\phi(t_{n,j}), \tag{2.2}$$

onde os $t_{n,j}, t_{n,0} < t_{n,1} < \ldots < t_{n,n}$, são pontos pertencentes ao intervalo $[a,b]$; os resultados desta secção podem ser facilmente extendidos ao caso em que $[a,b]$ não contém todos os pontos. As constantes $\{c_{n,j}\}$ designam-se por coeficientes. Define-se erro de quadratura, ou erro de aproximação, como sendo a diferença $I(\phi) - I_n(\phi)$ e será denotada por $E_n(\phi)$. À aproximação $I_n(\phi)$, na relação

$$I(\phi) = I_n(\phi) + E_n(\phi), \tag{2.3}$$

chamamos fórmula de quadratura com $(n+1)$ pontos.

Em geral, os coeficientes $\{c_{n,j}\}$ são escolhidos tal que $E_n(\phi)$ se anule para todas as funções de um certo espaço teste. Por exemplo, as fórmulas de quadratura baseadas em interpolação são obtidas quando esse espaço teste é uma certa classe de polinómios.

Definição 2.1 [1] *Uma sucessão de fórmulas de quadratura diz-se convergente para a função ϕ se $E_n(\phi) \to 0$ quando $n \to \infty$, ou seja, se $I_n(\phi) \to I(\phi)$ quando $n \to \infty$.* □

Os polinómios são uma parte importante na análise das fórmulas de quadratura e são usados para determinar a precisão das aproximações numéricas.

Definição 2.2 [2] *Uma fórmula de quadratura tem grau de precisão q se é exacta, isto é se $E_n(\phi) = 0$, para todos os polinómios ϕ de grau menor ou igual a q mas $E_n(t^{q+1}) \neq 0$.* □

[1] ver [7] pp. 51-52.

[2] ver [7] pp. 51-52.

Teorema 2.1 *Uma sucessão de fórmulas de quadratura do tipo (2.2) converge para todas as funções contínuas de valores reais em $[a,b]$ se e só se existe um número real c tal que*

$$\sum_{j=0}^{n} |c_{n,j}| \leq c \quad \forall n. \square$$

Para termos uma ideia da velocidade de convergência necessitamos da seguinte definição (ver [7]),

Definição 2.3 *Uma sucessão de fórmulas de quadratura da forma (2.2) tem ordem r se para toda a função ϕ suficientemente regular o erro de aproximação satisfaz $E_n(\phi) = O(h^r)$ quando $h \downarrow 0$, onde $h := \max_{1\leq j\leq n}(t_{n,j} - t_{n,j-1})$.* $\square$

2.1.1 Fórmulas de Quadratura Interpoladoras

Seja $P_n(t)$ o polinómio interpolador de grau $\leq n$, da função $\phi(t)$ nos $(n+1)$ pontos distintos $t_{n,0}, \ldots, t_{n,n}$, pertencentes ao intervalo $[a,b]$. Uma fórmula de integração interpoladora baseia-se em aproximar o integral da função, no intervalo $[a,b]$, pelo integral do seu polinómio interpolador $P_n(t)$.

Definição 2.4 — Fórmulas de Quadratura Interpoladoras
Seja $\phi \in C^{n+1}[a,b]$, e sejam $\{t_{n,j}\}_{j=0}^{n}$ pontos dados em $[a,b]$. Então uma fórmula de quadratura interpoladora define-se por

$$\int_a^b \phi(t)dt = \sum_{j=0}^{n} c_{n,j}\phi(t_{n,j}) + E_n(\phi), \tag{2.4}$$

onde

$$c_{n,j} = \int_a^b L_{n,j}(t)dt, \tag{2.5}$$

sendo $L_{n,j}(t)$ o j-ésimo polinómio de Lagrange

$$L_{n,j}(t) := \prod_{i=0, i\neq j}^{n} \frac{t - t_{n,i}}{t_{n,j} - t_{n,i}}. \square$$

Uma vez que $P_n(t) = \phi(t)$ para qualquer polinómio ϕ de grau $\leq n$, os coeficientes $c_{n,j}$ em (2.5) determinam uma fórmula de quadratura com grau de precisão de pelo menos n. Iremos ver posteriormente que o grau de precisão das fórmulas interpoladoras pode ser feito consideravelmente maior que n se os pontos $t_{n,j}$ forem escolhidos cuidadosamente.

As fórmulas interpoladoras são caracterizadas pelo teorema:

Teorema 2.2 *Uma fórmula de quadratura baseada em $n+1$ pontos distintos é uma fórmula interpoladora se e só se tem grau de precisão $q \geq n$.* □

O seguinte teorema dá-nos uma expressão para o erro de aproximação de uma fórmula de quadratura interpoladora:

Teorema 2.3 *Seja $\phi \in C^{n+1}[a,b]$. Então existe uma função $\theta(t)$ com valores no intervalo (a,b) tal que o erro de aproximação da fórmula de quadratura interpoladora (2.2) é dado por*

$$E_n(\phi) = \frac{1}{(n+1)!}\int_a^b \pi_n(t)\phi^{(n+1)}(\theta(t))dt,$$

onde

$$\pi_n(t) := \prod_{i=0}^{n}(t - t_{n,i}).$$

Tém-se ainda a seguinte majoração:

$$|E_n(\phi)| \leq \frac{1}{(n+1)!}(b-a)^{n+2}\alpha^{n+1}|\phi^{(n+1)}(\xi_n)|, \tag{2.6}$$

onde $\xi_n \in (a,b)$ e $\alpha := (t_{n,n} - t_{n,0})/(b-a)$. □

Note-se que a majoração (2.6) pode ser melhorada em certos casos especiais, tais como: Se os pontos $\{t_{n,j}\}$ estão relacionados com os zeros de certos polinómios ortogonais. Nestes casos, as fórmulas de quadratura verificam

$$I(\phi) = I_n(\phi) + c_n(b-a)^{q+2}\phi^{q+1}(\xi_n), \;\; \xi_n \in (a,b),$$

onde q é o grau de precisão da fórmula e c_n uma constante dependendo apenas de n.

2.1.1.1 Fórmulas de Quadratura de Newton-Côtes

Se nas fórmulas de quadratura interpoladoras definidas na definição 2.4 considerarmos os pontos igualmente espaçados obtemos as fórmulas de Newton-Côtes (ver [44] pp. 118-127). Se

$$t_{n,j} := C\underline{^{te}} + jh; \quad j = 0, 1, \ldots, n, \tag{2.7}$$

então os coeficientes $c_{n,j}$ podem ser expressos na forma

$$c_{n,j} = \frac{1}{h^n}\frac{(-1)^{n-j}}{j!(n-j)!}\int_a^b \frac{\pi_n(t)}{t - t_{n,j}}dt. \tag{2.8}$$

Observação 2.1 *Uma desvantagem das quadraturas de Newton-Côtes é que uma sucessão de fórmulas de Newton-Côtes não converge necessariamente para o verdadeiro valor do integral. Quando n tende para ∞ os coeficientes começam a assumir valores bastante elevados e por vezes também negativos, o que faz com que estas fórmulas sejam muito sensíveis aos erros de arredondamento (ver [11] pp. 38, 54). Os coeficientes $c_{n,j}$ negativos começam a aparecer com $n = 8$, por conseguinte as fórmulas de Newton-Côtes não são praticamente utilizadas para $n \geq 8$.*

• **Fórmulas de Newton-Côtes fechadas:** As fórmulas de Newton-Côtes dizem-se fechadas quando os pontos extremos do intervalo de integração pertencem ao conjunto (2.7); mais concretamente quando $t_{n,0} = a$ e $t_{n,n} = b$.

Prova-se que uma fórmula com $(n+1)$ pontos tem grau de precisão $(n+1)$ se n é par, e grau de precisão n se n for ímpar. Para ver demonstração consultar [7] pp. 58.

As fórmulas de Newton-Côtes fechadas mais usadas na resolução de equações integrais de Volterra são as seguintes,

(i) Regra de Euler:

$$\int_a^b \phi(t)dt \approx h \sum_{i=0}^{n-1} \phi_i, \quad h = \frac{b-a}{n},$$

(ii) Regra dos Trapézios:

$$\int_a^b \phi(t)dt \approx h \left[\frac{\phi_0 + \phi_n}{2} + \sum_{i=1}^{n-1} \phi_i \right], \quad h = \frac{b-a}{n},$$

(iii) Regra de Simpson:

$$\int_a^b \phi(t)dt \approx \sum_{i=1}^{n/2} \frac{h}{3} \left(\phi_{2i-2} + 4\phi_{2i-1} + \phi_{2i} \right), \quad h = \frac{b-a}{n},$$

• **Fórmulas de Newton-Côtes abertas:** A fórmulas de Newton-Côtes dizem-se abertas quando $t_{n,0} = a + h$ e $t_{n,n} = b - h$. Alguns exemplos encontram-se em [7].

Como exemplo referimos a

(i) Regra do ponto médio:

$$\int_a^b \phi(t)dt \approx h \sum_{i=0}^{n-1} \phi_{i+\frac{1}{2}}, \quad h = \frac{b-a}{n},$$

onde

$$\phi_{i+\frac{1}{2}} := \phi(t_{i+\frac{1}{2}}).$$

• **Fórmulas de Newton-Côtes quase abertas:** A fórmulas de Newton-Côtes dizem-se quase-abertas quando, ou $t_{n,0} = a$ e $t_{n,n} = b - h$, ou $t_{n,0} = a + h$ e $t_{n,n} = b$.

Para valores de n ímpar estas fórmulas de Newton-Côtes quase-abertas coincidem com as fórmulas abertas correspondentes a $n - 1$. Para valores de n par encontram-se exemplos em [7].

2.1.1.2 Fórmulas de Quadratura de Gauss-Legendre

O grau de precisão de uma fórmula de quadratura interpoladora pode ser consideravelmente maior que n escolhendo os pontos $\{t_{n,j}\}$ apropriadamente. O maior grau possível obtém-se escolhendo os pontos

$$t_{n,j} = \frac{b-a}{2} z_{n,j} + \frac{a+b}{2}, \qquad j = 0, 1, \ldots, n, \tag{2.9}$$

onde $\{z_{n,j}\}$ são os $n + 1$ zeros do polinómio de Legendre $P_{n+1}(t)$ ([7] pp. 59). Os coeficientes $\{c_{n,j}\}$ são então dados por

$$c_{n,j} = \frac{b-a}{(1 - z_{n,j})^2 [P'_{n+1}(z_{n,j})]^2}. \tag{2.10}$$

As fórmulas de quadratura abertas definidas por (2.9) e (2.10) dizem-se fórmulas de Gauss-Legendre. Note-se que neste caso os limites de integração não estão contidos no conjunto $\{t_{n,j}\}$. Prova-se que o grau de precisão destas fórmulas é $2n + 1$ e nenhuma outra fórmula de quadratura interpoladora possui um grau de precisão tão elevado.

Observação 2.2 *Outra vantagem em relação às fórmulas de Newton-Côtes é que uma sucessão de fórmulas de Gauss-Legendre converge para o verdadeiro valor de* $I(\phi)$ *quando* $n \to \infty$ *para qualquer* $\phi \in C[a, b]$.

2.1.1.3 Fórmulas de Quadratura de Radau e Lobatto

As fórmulas de quadratura de Radau e Lobatto obtêm-se fixando um ponto extremo ou ambos no conjunto $\{t_{n,j}\}$ respectivamente e escolhendo os restantes pontos tal que o grau de precisão seja tão grande quanto possível. Estas fórmulas são usadas no desenvolvimento de métodos de Runge-Kutta de ordem elevada para equações de Volterra.

• **Fórmulas de Radau I:** As fórmulas de Radau I são definidas pelos pontos

$$t_{n,j} = \frac{b-a}{2} z_{n,j} + \frac{b+a}{2}, \quad j = 1, \ldots, n, \quad t_{n,0} = a,$$

onde os $\{z_{n,j}\}$ são os n zeros do polinómio

$$\frac{P_n(t) + P_{n+1}(t)}{t+1},$$

com P_n sendo o polinómio de Legendre de grau n (ver [7] pp. 60-62). Os coeficientes $c_{n,j}$ são dados por

$$c_{n,j} = \frac{b-a}{2(1-z_{n,j})} \frac{1}{[P_n'(z_{n,j})]^2}, \quad j = 1, \ldots, n-1; \quad c_{n,0} = \frac{b-a}{(n+1)^2}.$$

• **Fórmulas de Radau II:** As fórmulas de Radau II são definidas pelos pontos $\overline{t}_{n,j} = a + b - t_{n,n-j}$ e coeficientes $\overline{c}_{n,j} = c_{n,n-j}$, onde $t_{n,j}$ e $c_{n,j}$ correspondem às fórmulas de Radau I.

As fórmulas de Radau em $(n+1)$ pontos têm grau de precisão $2n$.

• **Fórmulas de Lobatto:** As fórmulas de Lobatto são definidas pelos pontos

$$t_{n,j} = \frac{b-a}{2} z_{n,j} + \frac{b+a}{2}, \quad j = 1, \ldots, n-1, \quad t_{n,0} = a, \quad t_{n,n} = b,$$

onde os $\{z_{n,j}\}$ são os $n-1$ zeros do polinómio $P_n'(t)$, onde $P_n(t)$ é o polinómio de Legendre de grau n ([7] pp. 62-63). Os coeficientes $c_{n,j}$ são dados por

$$c_{n,j} = \frac{b-a}{n(n+1)} \frac{1}{[P_n(z_{n,j})]^2}, \quad j = 1, \ldots, n-1; \quad c_{n,0} = c_{n,n} = \frac{b-a}{n(n+1)}.$$

As fórmulas de Lobatto em $(n+1)$ pontos têm grau de precisão $2n-1$.

2.1.2 Fórmulas de Quadratura Compostas

Uma fórmula de quadratura composta obtém-se dividindo o intervalo $[a, b]$ em ν subintervalos e aplicando sucessivamente uma dada fórmula de quadratura em cada um dos ν subintervalos. Supondo que os subintervalos têm comprimento igual a $\beta - \alpha$, podemos escrever (ver [7] pp. 85):

Definição 2.5 — Fórmulas de Quadratura Compostas
Uma fórmula de quadratura composta é definida por

$$I_n(\phi) := \sum_{i=1}^{\nu} \sum_{j=0}^{m} c_j \phi_j^{(i)}, \quad \nu(\beta - \alpha) = b - a, \tag{2.11}$$

onde os $\{c_j\}$ *são os coeficientes de uma dada fórmula com* $(n+1)$ *pontos e onde* $t_0^{(i)}, \ldots, t_m^{(i)}$ *são os* $(m+1)$ *pontos no i-ésimo subintervalo.* □

Teorema 2.4 [3] *Considere-se a fórmula de quadratura com* $(m+1)$ *pontos no intervalo* $[\alpha, \beta]$ *com os coeficientes* $\{c_j\}$*, e seja o erro de aproximação dado pela expressão*

$$c(\beta - \alpha)^{r+1} \phi^{(r)}(\xi), \quad \xi \in [\alpha, \beta],$$

sempre que $\phi \in C^r[\alpha, \beta]$*, onde* c *é uma constante dependente de* c_j *mas não de* α, β *ou* ϕ*. Então para toda a função* $\phi \in C^r[a, b]$ *a fórmula composta (2.11) tem um erro de aproximação que satisfaz*

$$\lim_{\nu \to \infty} \nu^r E_n(\phi) = c(b-a)^r [\phi^{(r-1)}(b) - \phi^{(r-1)}(a)].$$

Além disso, a sucessão de fórmulas de quadratura (2.11) tem ordem r*.* □

2.1.3 A Técnica de Integração Produto

Para um melhor entendimento desta subsecção iremos ilustrá-la com um exemplo. Seja $\phi(t) = \sqrt{t}\cos(t)$ e considere-se a regra dos trapézios composta aplicada ao integral

$$\int_0^1 \sqrt{t}\cos(t)dt.$$

O erro de aproximação varia aproximadamente como $h^{3/2}$, ou seja, a ordem r desta sucessão de fórmulas de quadratura é aproximadamente 1.5, enquanto que geralmente a regra dos trapézios composta tem exactamente ordem 2. Usando a regra de Simpson composta continuamos a ter ordem 1.5. A razão é que a ordem r é obtida apenas para $\phi \in C^r[0,1]$. No nosso exemplo temos $\phi \in C[0,1]$, mas $\phi \notin C^1[0,1]$ logo o teorema 2.4 não pode ser aplicado para $r \geq 1$. Observamos que quando a função integranda é contínua e aplicamos as fórmulas de quadratura interpoladoras podemos perder precisão; para isso basta que a derivada de alguma ordem superior não exista. A integração produto (ver [21] pp. 272-277) é uma técnica adequada e conveniente para problemas da forma

$$\int_a^b w(t)\psi(t)dt, \tag{2.12}$$

[3] ver [7] pp. 85.

onde ψ é suficientemente diferenciável e w é apenas integrável em $[a,b]$. O seu conceito foi introduzido, em 1954, por Young [47] no contexto da resolução numérica de equações integrais.

Seja $P_n(t)$ o polinómio interpolador de grau $\leq n$, da função ψ nos $n+1$ pontos distintos $t_{n,0},\ldots,t_{n,n}$, pertencentes ao intervalo $[a,b]$. Então sabemos que

$$\psi(t) \simeq P_n(t) = \sum_{j=0}^{n} \psi(t_{n,j}) L_{n,j}(t),$$

onde $L_{n,j}(t)$ designa o j-ésimo polinómio de Lagrange

$$L_{n,j}(t) := \prod_{i=0, i\neq j}^{n} \frac{t - t_{n,i}}{t_{n,j} - t_{n,i}}.$$

Uma fórmula de integração produto baseia-se em aproximar $\psi(t)$, no integral (2.12), pelo seu polinómio interpolador $P_n(t)$ e depois integrar a função resultante $w(t)P_n(t)$, isto é,

$$\int_a^b w(t)\psi(t)dt \simeq \int_a^b w(t)P_n(t)dt = \sum_{j=0}^{n} \psi(t_{n,j}) \int_a^b L_{n,j}(t)w(t)dt.$$

A definição que se segue foi baseada em [7] pp. 87:

Definição 2.6 — Fórmulas de Integração Produto
Sejam $\psi \in C^{n+1}[a,b]$ e w integrável e não negativa em $[a,b]$, e sejam $\{t_{n,j}\}_{j=0}^{n}$ pontos dados em $[a,b]$. Então a fórmula de quadratura integração produto é definida por

$$\int_a^b w(t)\psi(t)dt = \sum_{j=0}^{n} c_{n,j}\psi(t_{n,j}) + E_n(w,\psi),$$

onde

$$c_{n,j} := \int_a^b L_{n,j}(t)w(t)dt, \quad L_{n,j}(t) := \prod_{i=0, i\neq j}^{n} \frac{t - t_{n,i}}{t_{n,j} - t_{n,i}}.\square$$

Em seguida apresentamos dois resultados. O primeiro (ver [7] pp. 87 para demonstração) dá-nos uma expressão para o erro quando $\psi \in C^{n+1}[a,b]$ e o segundo (ver [7] pp. 89) é um teorema mais geral que nos permite obter a expressão para o erro quando ψ não está contido na classe $C^{n+1}[a,b]$.

Teorema 2.5 *Seja $\psi \in C^{n+1}[a,b]$. Se os coeficientes $\{c_{n,j}\}$ são limitados, então existe uma função $\theta(t)$ com valores em $[a,b]$ tal que*

$$E_n(w,\psi) = \frac{1}{(n+1)!} \int_a^b w(t)\pi_n(t)\psi^{(n+1)}(\theta(t))dt,$$

onde

$$\pi_n(t) := \prod_{j=0}^{n} (t - t_{n,j}).\square$$

Teorema 2.6 *Assuma-se que fórmula de quadratura*

$$I_n(w,\psi) := \sum_{j=0}^{n} c_{n,j}\psi(t_{n,j}),$$

aproximando

$$I(w,\psi) := \int_a^b w(t)\psi(t)dt,$$

tem grau de precisão q. Se $w(t)$ é não negativa (com $w(t) \neq 0$) e integrável em $[a,b]$, e se $\psi \in C^m[a,b]$, onde $1 \leq m \leq q+1$, então o erro de quadratura

$$E_n(w,\psi) := I(w,\psi) - I_n(w,\psi)$$

pode ser expresso na forma

$$E_n(w,\psi) = \int_a^b K_m(t)\psi^{(m)}(t)dt,$$

onde

$$K_m(t) := \int_a^b w(x)\frac{(x-t)_+^{m-1}}{(m-1)!}dx - \sum_{j=0}^{n} c_{n,j}\frac{(t_{n,j}-t)_+^{m-1}}{(m-1)!}.\square$$

Observação 2.3 *As fórmulas definidas em 2.1.3 podem ser aplicadas repetidamente como foi descrito em 2.1.2. Observe-se que, na regra dos trapézios composta convencional, os pesos de quadratura são constantes enquanto que neste caso dependem dos pontos t_i e têm de ser calculados em cada ponto t_i. Este é o preço que temos que pagar para obtermos maior ordem de convergência. Actualmente nos cálculos podemos aplicar a integração produto apenas em regiões onde a integranda não é regular e não em todo o intervalo; isto é, no caso $\phi(t) = \sqrt{t}\cos(t)$ é suficiente restringir a aplicação de uma regra de integração produto numa vizinhança fixa da origem.*

2.2 Equações de Volterra com núcleo regular

Uma vez que poucas equações de Volterra, encontradas na prática, podem ser resolvidas explicitamente, é frequentemente necessário recorrer a técnicas numéricas.

Existe, como já foi dito no capítulo anterior, uma ligação entre as equações integrais de Volterra e os problemas de valor inicial em equações diferenciais. Não é, portanto, de estranhar que alguns dos métodos numéricos utilizados para equações de Volterra sejam generalizações de métodos utilizados para equações diferenciais ordinárias.

Consideremos a equação (1.1) sujeita às condições do teorema 1.1 , da secção 1.3. Desta forma garantimos a existência e unicidade de uma solução contínua. Os métodos numéricos que a seguir descrevemos para a resolução deste tipo de equações baseiam-se nestas condições, e as conclusões são válidas desde que estas se verifiquem.

2.2.1 Métodos de Quadratura Directa

Os métodos de quadratura directa obtêm-se restringindo a equação (1.1) a uma rede (ou partição) finita $\{t_n : n = 0, \ldots, N\}$, do intervalo $I := [a, b]$, e depois aproximando o integral por uma fórmula de quadratura interpoladora. Neste caso a solução exacta é aproximada por um elemento $\mathbf{u}_N \in \Re^{N+1}$, o qual representa os valores aproximados nos pontos da rede. Por simplicidade de exposição, admitimos que $a = 0$, $b = T$ e a rede é uniforme, ou seja, $t_n = nh$, $n = 0, \ldots, N$, onde $h > 0$ é o comprimento do passo. Sejam $\{w_{n,j}\}_{j=0}^{n}$ constantes para $n = \kappa, \ldots, N$, então um método de quadratura directa define-se por

$$y_n = g_n + h \sum_{j=0}^{n} w_{n,j} K(t_n, t_j, y_j), \quad n = \kappa, \ldots, N \,, \tag{2.13}$$

onde $g_n := g(t_n)$ e $y_0 = g_0$ *e* $y_1, y_2, \ldots, y_{\kappa-1}$ são valores iniciais dados.

Este método traduz-se num sistema de $N - \kappa + 1$ equações não lineares nas incógnitas y_n, $n = \kappa, \ldots N$. Quando as fórmulas de quadratura utilizadas em (2.13) são fechadas então $w_{n,n}$ são não nulos para $n = \kappa, \ldots, N$ e o método diz-se *implícito*. Caso contrário o método diz-se *explícito*.

Exemplos destes métodos baseados nas fórmulas de quadratura de Newton-Côtes e de Gregory podem ser encontrados em [9], [27], [31] e [7].

Seguidamente damos algumas definições e um teorema que nos dá condições suficientes de convergência para um método da forma (2.13) (ver [31] pp. 100-102).

Definição 2.7 *Um método da forma (2.13) diz-se convergente se*

$$\lim_{h \to 0, Nh = T} \max_{0 \le n \le N} |y(t_n) - y_n| = 0. \square$$

Definição 2.8 *Diz-se que um método tem ordem p se o erro de aproximação satisfaz*

$$\max_{0 \le n \le N} |y(t_n) - y_n| = O(h^p), \ \textit{quando} \ h \downarrow 0,$$

onde $h := \max_{0 \le j \le N-1} h_j = \max_{0 \le j \le N-1} (t_{j+1} - t_j)$, *sendo* $\{t_n\}_{n=0}^{N}$ *pontos dados no intervalo* $[a, b]$, *com* $t_0 = a$ *e* $t_N = b$. $\square$

Definição 2.9 *Seja S a classe de equações da forma (1.1). Se para toda a equação pertencente à classe S, existe uma constante c (independente de h, mas geralmente dependente de K e y) tal que*

$$|E(h,t_n)| = |\int_0^{t_n} K(t_n,s,y(s))ds - h\sum_{j=0}^{n} w_{n,j}K(t_n,t_j,y_j)| \leq ch^p,$$

então o método (2.13) diz-se consistente de ordem p em S. □

Teorema 2.7 *Suponhamos verificadas as condições seguintes:*

(i) *a solução $y(t)$ de (1.1) e o núcleo $K(s,t,y)$ são tais que o método de aproximação (2.13) é consistente de ordem p com (1.1),*

(ii) *os pesos satisfazem*

$$sup_{n,i}|\omega_{n,i}| \leq \Omega < \infty,$$

(iii) *os erros iniciais $y_n - y(t_n)$, $n = 0,1,\ldots,\kappa-1$ verificam*

$$|y(t_n) - y_n| = O(h^{p-1}).$$

Então o método (2.13) é convergente com ordem de convergência maior ou igual a p. □

2.2.2 Métodos de Colocação em $S_r^d(Z_N)$

Um método de colocação, para uma dada equação funcional, baseia-se em procurar a solução aproximada u num espaço de funções de dimensão finita escolhido de forma adequada, ao qual chamamos espaço de aproximação Ω, de tal forma que u satisfaça a equação num certo conjunto finito de pontos. Em geral, o espaço de aproximação é um subespaço do espaço que contém a solução.

Nesta secção estudaremos métodos de colocação para equações de Volterra lineares com núcleo regular, onde os espaços de aproximação serão os espaços de splines polinomiais $S_r^d(Z_N)$.

2.2.2.1 Os Espaços de Splines Polinomiais $S_r^d(Z_N)$

Considerem-se o inteiro N (fixo) e a rede (ou partição)

$$\Pi_N : 0 = t_0 < t_1 < \ldots < t_N = T, \quad N \geq 1,$$

no intervalo $I := [0, T]$. Iremos supôr que a rede Π_N é *quase-uniforme*, isto é, se

$$h_n := t_{n+1} - t_n, \;\; h_{min} := \min_n h_n, \;\; h_{max} := \max_n h_n,$$

então existe uma constante γ, independente de N, tal que para qualquer N natural,

$$\frac{h_{max}}{h_{min}} \leq \gamma.$$

Esta suposição tem muita importância na análise da convergência dos métodos de colocação em espaços de splines polinomiais. Associemos à rede anterior o conjunto de pontos interiores,

$$Z_N := \{t_n : n = 1, \ldots, N-1\},$$

e os subintervalos,

$$\upsilon_0 := [t_0, t_1], \quad \upsilon_n := (t_n, t_{n+1}], \quad 1 \leq n \leq N-1.$$

Seja $\overline{Z}_N := Z_N \bigcup \{T\}$.

A definição seguinte, de espaços $S_r^d(Z_N)$ encontra-se em [7] pp. 236:

Definição 2.10 *Sejam r e d inteiros dados tais que $-1 \leq d \leq m-1$, e seja π_r o espaço dos polinómios de grau menor ou igual a r. O espaço linear das splines polinomiais de grau r associado aos nós Z_N define-se por:*

$$S_r^d(Z_N) := \{u : u \in C^d(I), u(t)|_{t \in \upsilon_n} = u_n(t) \in \pi_r, \quad n = 0, 1, \ldots, N-1\}.$$

Os elementos de $S_r^{-1}(Z_N)$ podem ter descontinuidades de salto nos nós Z_N. □

No caso em que $d = r-1$, obtêm-se os espaços $S_r^{r-1}(Z_N)$ aos quais chamaremos espaços de splines clássicos. No capítulo 3 utilizaremos também os espaços $\tilde{S}_r^d(Z_N)$, $-1 \leq d \leq m-1$, definidos do seguinte modo:

$$\tilde{S}_r^d(Z_N) := \{u : u \in C^d(I), u(t)|_{t \in \tilde{\upsilon}_n} = u_n(t) \in \pi_r, \quad n = 0, 1, \ldots, N-1, u(T) = u(t_N)\},$$

onde,

$$\tilde{\upsilon}_n := [t_n, t_{n+1}), \quad 0 \leq n \leq N-1.$$

Note-se que os espaços $\tilde{S}_r^{-1}(Z_N)$ se destinguem dos espaços $S_r^{-1}(Z_N)$ por os seus elementos serem funções contínuas à esquerda nos pontos Z_N. No caso em que $d > -1$, $\tilde{S}_r^d(Z_N)$ coincide com $S_r^d(Z_N)$.

2.2.2.2 Bases para o espaço $S_r^d(Z_N)$

O espaço linear $S_r^d(Z_N)$ tem dimensão $N(r-d)+d+1$, $-1 \leq d \leq r-1$, e uma base para este espaço é dada por

$$\{1, t, \ldots, t^r, (t-t_1)_+^{r-j}, \ldots, (t-t_{N-1})_+^{r-j}, j = 0, 1, \ldots, r-d-1\},$$

ou seja, qualquer elemento de $S_r^d(Z_N)$ possui uma representação única da forma

$$u(t) = \sum_{i=0}^{r} \alpha_i t^i + \sum_{l=1}^{N-1} \sum_{j=0}^{r-d-1} \beta_{l,j} (t-t_l)_+^{r-j},$$

onde as funções $(t-t_l)_+^{r-j}$ são as funções potência truncadas

$$(t-t_l)_+^{r-j} = \begin{cases} (t-t_l)^{r-j} & \text{se} \quad t \geq t_l \\ 0 & \text{se} \quad t < t_l \end{cases}.$$

Esta representação é contudo pouco conveniente do ponto de vista computacional porque o cálculo da função u perto do extremo superior do intervalo I necessita do cálculo de todas, ou quase todas, as funções base. Para ultrapassar esta dificuldade é possível construir uma base mais simétrica e de suporte compacto fazendo certa combinação linear de funções potência truncadas t_+^k. Os elementos desta base são chamados $B-splines$ (ver [7] pp. 236-239, [22] pp. 333-352, [18] pp. 229-272).

Os espaços de splines polinomiais de maior importância nas aplicações das equações de Volterra são os espaços $S_r^{-1}(Z_N)$ e $S_r^0(Z_N)$. Isto deve-se a certo tipo de ligação entre métodos de colocação para problemas de valor inicial, em espaços de splines polinomiais, e métodos de colocação, em espaços do mesmo tipo, para equações integrais de Volterra.

2.2.2.3 A Equação de Colocação

Seja $X(N)$, o conjunto dos pontos de colocação (finito e contido em I), dado por

$$X(N) := \bigcup_{n=0}^{N-1} X_n, \tag{2.14}$$

onde $X_n := \{t_{n,j} := t_n + c_j h_n : 0 \leq c_1 < \ldots < c_{r-d} \leq 1\}$. Os pontos $c_1, \ldots, c_{r-d}$ chamam-se parâmetros de colocação. Devemos então determinar o elemento $u \in S_r^d(Z_N)$ que verifica a equação (1.1) no conjunto $X(N)$. Obtém-se

$$u_n(t_{n,j}) = g(t_{n,j}) + \int_0^{t_n} k(t_{n,j}, s, u(s))ds + \int_{t_n}^{t_{n,j}} k(t_{n,j}, s, u_n(s))ds,$$
$$1 \leq j \leq r-d, \;\; n = 0, \ldots, N-1.$$

Usando a definição 2.10 podemos reescrever, a equação anterior, numa forma que realça o carácter recursivo do método,

$$u_n(t_{n,j}) = g(t_{n,j}) + \sum_{j=0}^{n-1} \int_{t_j}^{t_{j+1}} k(t_{n,j}, s, u_j(s))ds + \int_{t_n}^{t_{n,j}} k(t_{n,j}, s, u_n(s))ds, \quad (2.15)$$
$$1 \leq j \leq r-d, \;\; n = 0, \ldots, N-1.$$

Os espaços que vamos utilizar correspondem à escolha $d = -1$ e $r = m-1$. Utilizando esta escolha de parâmetros e fazendo a mudança de variável $s = t_i + \nu h_i$ na equação (2.15) obtém-se

$$\begin{aligned} u_n(t_{n,j}) = g(t_{n,j}) &+ \sum_{j=0}^{n-1} h_j \int_0^1 k(t_{n,j}, t_j + \nu h_j, u_j(t_j + \nu h_j))ds \\ &+ h_n \int_0^{c_j} k(t_{n,j}, t_n + \nu h_n, u_n(t_n + \nu h_n))ds, \qquad (2.16) \\ &\qquad 1 \leq j \leq m, \;\; n = 0, \ldots, N-1, \end{aligned}$$

onde

$$u_n(t_n + \nu h_n) = \sum_{j=1}^{m} L_j(\nu) u_n(t_{n,j}), \quad t_n + \nu h_n \in \upsilon_n, \quad n = 0, \ldots, N-1, \qquad (2.17)$$

sendo L_j o j-ésimo polinómio de Lagrange fundamental, para os parâmetros de colocação, dado por

$$L_j(\nu) := \prod_{i=1, i \neq j}^{m} \frac{c_i - \nu}{c_i - c_j}. \qquad (2.18)$$

O conjunto dos pontos de colocação é agora dado por $X_n := \{t_{n,j} := t_n + c_j h_n : 0 \leq c_1 < \ldots < c_m \leq 1\}$.

Observe-se que quando $m \geq 2$ e os pontos t_n e t_{n+1} pertencem a X_n, ou seja $c_1 = 0$ e $c_m = 1$, e a aproximação de colocação definida por (2.15) é contínua:

Teorema 2.8 [4] *Sejam g e k funções contínuas. Se os parâmetros $\{c_j\}$ de colocação são tais que $0 = c_1 < c_2 < \ldots < c_{m-1} < c_m = 1$, então a aproximação de colocação definida por (2.15), para $1 \leq j \leq m$, $n = 0, \ldots, N-1$, é um elemento de $S^0_{m-1}(Z_N)$.* □

• Convergência Global

Vamos agora abordar o problema da convergência do método de colocação (2.16). O teorema que apresentamos (ver [7] pp. 251-253) diz-nos qual é a melhor ordem global de convergência que se consegue obter quando aproximamos a

[4] ver [7] pp. 248.

solução y da equação (1.1) pela aproximação de colocação u solução da equação de colocação (2.16).

O erro cometido quando aproximamos y por u é dado por $e := y - u$ e a sua restrição ao intervalo v_n designa-se por e_n. Observe-se que a função erro e irá ter um número finito de saltos. Defina-se a distância de y a u no intervalo $I := [0, T]$ por

$$\|e\|_\infty := sup\{|e_n(t)| : t \in v_n, n = 0, \ldots, N-1\},$$

e seja

$$h := h_{max} = \max_n(t_{n+1} - t_n),$$

o diâmetro da rede Π_N.

Teorema 2.9 *Suponha-se que, em (1.1), as funções g e K pertencem a $C^m(I)$ e $C^m(S)$ respectivamente e que K verifica a condição de Lipschitz do teorema 1.1. Então existe um $\overline{h} > 0$ tal que a equação de colocação (2.16) define para cada $h \in (0, \overline{h})$ um único elemento $u \in S_{m-1}^{-1}(Z_N)$. O erro cometido, quando aproximamos a solução exacta y da equação (1.1) pela aproximação u, satisfaz, para qualquer escolha dos parâmetros de colocação $\{c_j\}$ com $0 \leq c_1 < \ldots < c_m \leq 1$ e para todas as sucessões de redes quase-uniformes,*

$$\|e\|_\infty \leq Ch^m, \tag{2.19}$$

onde C representa uma constante finita independente de h (mas dependendo de $\{c_j\}$). □

• Superconvergência Local em $\overline{Z}_N$

Anteriormente referimos que a aproximação de colocação $u \in S_{m-1}^{-1}(Z_N)$ da solução da equação integral (1.1) possui, sob hipóteses aproximadas de regularidade para g e K, a ordem global de convergência $p = m$; esta ordem é a melhor possível. Agora, daremos resultados que nos informam se ao considerarmos apenas os pontos da rede $\{t_n\}$ iremos conseguir maior ordem de convergência. Ou seja, será que existe um inteiro $p^* > p = m$ tal que, quando $h \downarrow 0$ e $Nh \leq \gamma T$ (sucessões de redes quase-uniformes),

$$|e(t_n)| \leq C^* h^{p^*} \ \ para\ todo\ o\ t_n \in \overline{Z}_N := Z_N \cup \{T\},$$

para uma escolha adequada dos parâmetros de colocação $\{c_j\}$. Dizemos que a aproximação u é *superconvergente localmente*, em $\overline{Z}_N$, se encontrarmos um p^* nas condições anteriores (ver [42], [7] pp. 284).

O teorema seguinte (ver [7] pp. 255-257) mostra que existem várias formas de obter superconvergência local.

Teorema 2.10 *Seja $u \in S_{m-1}^{-1}(Z_N)$ a aproximação de colocação da solução da equação integral (1.1) e admita-se que g e K satisfazem $g \in C^{2m-\nu}(I)$, $K \in C^{2m-\nu}(S)$, para algum $\nu \in \{0,1,2\}$ e com $m \geq \lfloor \nu/2 \rfloor + 1$. Considere-se que K é lipchitziana na segunda variável. Seja P_m o polinómio de Legendre de grau m.*

(a) *Se os parâmetros de colocação $\{c_j\}$ são os zeros de $P_{m-1}(2s-1) - P_m(2s-1)$ (isto é, os pontos de Radau II em $(0,1]$), então temos, para $\nu = 1$,*

$$\max_{t_n \in \overline{Z}_N} |e(t_n)| = O(h^{2m-1}), \ \text{quando } h \downarrow 0, \ Nh \leq \gamma T.$$

(b) *Se os parâmetros de colocação $\{c_j\}$ são os zeros de $s(1-s)P'_{m-1}(2s-1)$ (isto é, os pontos de Lobatto para $[0,1]$), e se $\nu = 2$ então segue que*

$$\max_{t_n \in \overline{Z}_N} |e(t_n)| = O(h^{2m-2}), \ \text{quando } h \downarrow 0, \ Nh \leq \gamma T.$$

(c) *Se os parâmetros de colocação $\{c_j\}$ são os zeros de $P_m(2s-1)$ (isto é, os pontos de Gauss para $(0,1)$), e se $\nu = 0$ obtemos*

$$\max_{t_n \in \overline{Z}_N} |e(t_n)| = O(h^m), \ \text{quando } h \downarrow 0, \ Nh \leq \gamma T;$$

por outras palavras, colocação nos pontos de Gauss não implica superconvergência local em $\overline{Z}_N$.

(d) *Se, contudo, os primeiros $m-1$ parâmetros de colocação $\{c_j\}$ são os zeros de $P_{m-1}(2s-1)$, e se $c_m = 1$, então segue que com $\nu = 2$*

$$\max_{t_n \in \overline{Z}_N} |e(t_n)| = O(h^{2m-2}), \ \text{quando } h \downarrow 0, \ Nh \leq \gamma T;$$

isto é, encontramos a mesma ordem de superconvergência local como para os pontos de Lobatto. □

Observação 2.4 *Pode demonstrar-se que o erro de colocação em cada ponto de $\overline{Z}_N$ é da mesma ordem do erro associado com a fórmula de quadratura interpoladora cujos nós são os parâmetros de colocação $\{c_j\}$, desde que o último destes parâmetros satisfaça $c_m = 1$. Note-se que a lista de casos de superconvergência local dados no teorema (2.10) não é exaustivo. Entre outras possibilidades mencionamos o caso onde os parâmetros de colocação são equidistantes, isto é, onde $c_j = \frac{j-1}{m-1}$, para $j = 1, \ldots, m$. As fórmulas de quadratura em m pontos baseadas nestes parâmetros são as fórmulas de Newton-Côtes. Se m é ímpar, e se os m parâmetros de colocação $\{c_j\}$ são igualmente espaçados (implicando em particular $c_1 = 0$, $c_m = 1$), então resulta superconvergência local de ordem $p^* = m+1$ em $\overline{Z}_N$. Por exemplo, se*

$m = 3$, então a ordem global de convergência é dada por $p = m = 3$, enquanto que $\overline{Z}_N$ temos a ordem $p^* = m + 1 = 4$. Se $c = 1/2$, e se os integrais (2.16) são aproximados pela fórmula de quadratura interpoladora em três pontos, o método correspondente reduz-se ao método de Simpson.

Observação 2.5 *Observemos que os resultados desta secção, para equações com núcleo regular, permanecem válidas para equações com núcleo fracamente singular do tipo (1.5) e que possuem uma solução y pertencente a $C^m(I)$.*

2.3 Equações de Volterra com Núcleo Fracamente Singular

Até agora apenas considerámos equações de Volterra com núcleos regulares (limitados) onde a regularidade da solução é determinada pela regularidade do núcleo dado e da função inicial. Se admitirmos núcleos com singularidades fracas então as soluções resultantes são em geral não regulares perto do ponto inicial do intervalo de integração (isto é, as suas derivadas são ilimitadas). Isto significa que, se um método numérico é para possuir uma ordem elevada de convergência, temos de ter em conta, de alguma forma, o comportamento singular da solução exacta perto desse ponto.

2.3.1 Métodos de Integração Produto

Os métodos descritos para equações com núcleo regular podem ser aplicados a equações com núcleo singular, utilizando a chamada técnica de integração produto. Por exemplo, métodos de quadratura directa para a equação (1.4) obtêm-se restringindo (1.4) a uma rede $\{t_n\}_{n=0}^N$ em $[a, b]$ e aplicando uma sucessão de fórmulas de quadratura integração produto (ver secção 2.1.3). Ou seja, em vez da expressão (2.13) para equações com núcleo regular, temos agora

$$y_n = g_n + \sum_{j=0}^{n} c_{n,j} k(t_n, t_j, y_j), \quad n = 0, \ldots, N \ ,$$

onde

$$g_n := g(t_n), \quad c_{n,j} := \int_0^{t_n} L_j(t) p(t_n, t) dt, \quad L_j(t) := \prod_{i=0}^{n} \frac{t - t_i}{t_j - t_i}.$$

2.3.2 Métodos de Colocação em $S_r^d(Z_N)$

Na secção 2.2.2 usámos funções spline polinomiais de $S_{m-1}^d(Z_N)$, com $d \in \{-1, 0\}$, para aproximar as soluções de equações integrais de Volterra de segunda espécie com núcleos limitados. No caso em que os núcleos não são limitados, são os seguintes os principais resultados sobre colocação nos espaços referidos: se as funções dadas $g(t)$ e $k(t, s, y)$ são m vezes continuamente diferenciáveis, nos seus respectivos domínios, segue que a solução correspondente tem o mesmo grau de regularidade em I. Por conseguinte, isto implica que a aproximação de colocação do espaço de splines anterior exibe a ordem global de convergência (óptima) $p = m$. Contudo, se admitirmos núcleos contendo uma singularidade fraca da forma $(t - s)^{-\alpha}$, com $0 < \alpha < 1$, e se empregarmos sucessões de redes quasi-uniformes, então a ordem global de convergência da aproximação desce para $p = 1 - \alpha$, independentemente da escolha do grau $m - 1$. Isto deve-se ao facto de a regularidade de $g(t)$ e $k(t, s, y)$ agora dar origem a uma solução cuja primeira derivada perto de $t = 0$ se comporta como $y'(t) \sim t^{-\alpha}$. Uma maneira de recuperar a ordem de convergência óptima é usar redes graduadas (ver [7] e [6]) para reflectir a estrutura de $y(t)$ perto de $t = 0$.

Contudo, nestas malhas o comprimento do passo inicial torna-se muito pequeno à medida que N aumenta, podendo resultar num acumular considerável de erros de arredondamento nos cálculos subsequentes.

Nesta tese consideramos apenas métodos de colocação aplicados a equações integrais cuja solução satisfaz certas condições de regularidade, que serão especificadas para cada método.

Capítulo 3

Estudo de uma Equação Integral

Pretende-se resolver numericamente, uma equação integral de Volterra de segunda espécie fracamente singular, utilizando métodos de integração produto fazendo colocação nos espaços $\tilde{S}_m^{-1}(Z_N)$ com $m = 0, 1, 2$ e $S_0^{-1}(Z_N)$ (ver secção 2.2.2).

O objectivo é comparar os métodos utilizando dois exemplos típicos de funções g do segundo membro. Averiguar até que ponto é útil aumentar o grau das splines polinomiais de aproximação. Para isso vamos ter em conta a ordem, precisão, regularidade exigida para as condições iniciais e tempos de computação.

Na secção 3.1 será apresentado o problema a resolver. Nesta inclui-se, além duma perspectiva histórica, uma formulação do problema. É ainda obtido um resultado de existência e unicidade de solução. Nas secções seguintes são estudados separadamente diversos métodos de colocação. Conclui-se com uma comparação entre alguns destes métodos.

3.1 Introdução

O problema que é tratado neste trabalho tem como origem o facto de determinados problemas de transferência de calor poderem ser modelados através da equação

$$f(x) = h(x) - \frac{1}{\sqrt{\pi}} \int_x^\infty \frac{1}{t} \left(log \frac{t}{x} \right)^{-1/2} f(t)dt. \tag{3.1}$$

Depois de Bartoshevich, que estudou pela primeira vez esta equação, Sizonenko [40] demonstrou que se $g \in L_2(0, +\infty)$, então a equação tem uma solução em $L_2(0, +\infty)$ dada por,

$$f(x) = \frac{1}{2}h(x) + \frac{d}{dx} \int_x^\infty \frac{1}{t} \left(\int_{log(t/x)}^\infty erfc(u^{1/2})du - erfc\left(\left(log \frac{t}{x} \right)^{-1/2} \right) \right) h(t)dt, \tag{3.2}$$

onde $erfc(x) = 1 - erf(x)$ sendo $erf(x)$ dada por

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-x^2} dx.$$

Mais tarde (em 1982) Rooney [39] provou que a equação (3.1) tem a seguinte solução em $L_2(0, +\infty)$ (através do estudo em espaços mais gerais) dada por uma expressão mais simples do que (3.2):

$$f(x) = h(x) + \int_x^\infty \frac{1}{t} \left(\frac{t}{x} erfc\left(\left(log \frac{t}{x} \right)^{1/2} \right) - \pi^{-1/2} \left(log \frac{t}{x} \right)^{-1/2} \right) h(t)dt. \tag{3.3}$$

Embora (3.3) se apresente com uma forma mais simples ela não é útil do ponto de vista prático. Note-se que deste ponto de vista pode interessar saber o valor da solução $f(x)$ para vários valores de x (em geral constrói-se uma tabela com estes valores). Ao aproximar numericamente o integral (3.3), o número de operações exigidas, para obter uma precisão aceitável, pode ser tão grande que torna o método ineficiente.

Nota-se, no entanto, que se utilizarmos as mudanças de variável $t = 1/s$ e $x = 1/u$, na equação (3.1), obtemos a seguinte equação linear de segunda espécie equivalente:

$$F(u) = H(u) - \frac{1}{\sqrt{\pi}} \int_0^u \frac{1}{\sqrt{ln(u/s)}} \frac{1}{s} F(s)ds, \;\; u > 0, \tag{3.4}$$

onde

$$F(u) := f\left(\frac{1}{u} \right), \quad H(u) := h\left(\frac{1}{u} \right).$$

Se observarmos atentamente o segundo membro de (3.4), verificamos que o integral

$$\int_0^u \frac{1}{s\sqrt{ln(u/s)}} ds$$

é divergente, o que faz surgirem dificuldades na resolução numérica daquela equação.

Aplicando a aproximação utilizada por Diogo [14], considera-se

$$y(t) := t^{-\mu}F(t), \;\; g(t) := t^{-\mu}H(t), \;\; \mu > 0,$$

e substitui-se em (3.4), resultando a equação

$$y(t) = g(t) - \int_0^t p(t,s)y(s)ds, \;\; t \in I := [0,T], \tag{3.5}$$

onde

$$p(t,s) := \frac{1}{\sqrt{\pi}} \frac{1}{\sqrt{ln(t/s)}} \left(\frac{s}{t}\right)^{\mu} \frac{1}{s}, \;\; \mu > 0. \tag{3.6}$$

É de salientar que assim se obtém

$$\int_0^t p(t,s)ds = \frac{1}{\mu} = constante.$$

Em [14] provou-se um resultado de existência e unicidade de solução do problema (3.5), (3.6) e obtiveram-se soluções numéricas utilizando os métodos de Euler e trapézios. Foi feita uma análise teórica da convergência destes dois métodos chegando-se à conclusão que eles têm ordens de convergência 1 e 2 respectivamente. Também se transformou a equação (3.5) na equação seguinte, cujo núcleo, embora ilimitado, apresenta-se com uma expressão mais simples:

$$y(t) - \int_0^t p_2(t,s)y(s)ds = g_2(t), \quad t \in I := [0,T], \quad T > 0, \tag{3.7}$$

onde

$$p_2(t,s) = \left(\frac{s}{t}\right)^{\mu} \frac{1}{s}, \quad \mu > 0, \tag{3.8}$$

$$g_2(t) = -\int_0^t p(t,s)g(s)ds + g(t). \tag{3.9}$$

É na forma (3.7), (3.8), (3.9) que o problema vai ser estudado neste trabalho.

3.1.1 Formulação do Problema

Dada a grande dificuldade em obter soluções das equações integrais de Volterra foi necessário desenvolver e adaptar métodos para equações específicas. Neste trabalho procuraremos investigar métodos numéricos de colocação que foram adaptados à resolução de equações de Volterra de segunda espécie de uma forma específica.

Considere a equação (3.5) sendo $y(t)$ uma função desconhecida e $g(t)$ uma função dada. O teorema seguinte (ver [14] pp. 94) permite simplificar o tratamento da equação (3.5).

Teorema 3.1 *A equação (3.5), com $p(t,s)$ dado por (3.6), é equivalente à equação (3.7), com $p_2(t,s)$ e $g_2(t)$ dados por (3.8) e (3.9) respectivamente.*

Demonstração Consideremos a equação

$$y(s) + \int_0^s p(s,\lambda)y(\lambda)d\lambda = g(s), \tag{3.10}$$

onde $0 \leq s \leq T,\ T > 0$ e $p(s,\lambda)$ é dado por (3.6). Multiplicando ambos os membros da equação (3.10) por $p(t,s)$ e integrando a equação resultante de 0 a t, obtemos,

$$\int_0^t p(t,s)y(s)ds + \int_0^t \int_0^s p(t,s)p(s,\lambda)y(\lambda)d\lambda ds = \int_0^t p(t,s)g(s)ds.$$

A fórmula de Dirichlet afirma que

$$\int_0^t \int_0^s \phi(s,\lambda)d\lambda ds = \int_0^t \int_\lambda^t \phi(s,\lambda)dsd\lambda,$$

desde que ambos os integrais existam. Aplicando este resultado à equação anterior vem,

$$\int_0^t p(t,s)y(s)ds + \int_0^t y(\lambda) \int_\lambda^t p(t,s)p(s,\lambda)dsd\lambda = \int_0^t p(t,s)g(s)ds. \tag{3.11}$$

Observa-se que, uma vez que (3.5) se verifica o primeiro integral do primeiro membro da equação (3.11) é igual a $g(t) - y(t)$. Fazendo uso desta observação podemos escrever a equação (3.11) na forma,

$$y(t) - \int_0^t y(\lambda) \int_\lambda^t p(t,s)p(s,\lambda)dsd\lambda = g(t) - \int_0^t p(t,s)g(s)ds. \tag{3.12}$$

Uma vez que

$$\int_\lambda^t p(t,s)p(s,\lambda)ds = \left(\frac{\lambda}{t}\right)^\mu \frac{1}{\lambda},$$

concluimos que a equação (3.12) pode ser escrita na forma (3.7). □

3.1.2 Um Resultado de Existência e Unicidade

Quando estamos perante um problema de análise numérica a primeira coisa a fazer é averiguar as condições de existência e unicidade de solução. O teorema seguinte diz-nos que se g for suficientemente regular então o problema (3.7), (3.8), (3.9), tem solução única com a mesma regularidade que g.

Teorema 3.2 *Seja V_m um espaço normado que é definido por*

$$V_m = \{\phi : \phi(t) \in C^m(I)\}, \quad I = [0,T], \quad T > 0, \quad m \in \aleph_0,$$

com a norma

$$\|\phi\|_m = \max\left\{\left|\frac{d^j\phi}{dt^j}\right|, t \in I, 0 \leq j \leq m\right\}.$$

Se $g \in V_m$ e $\mu > 1$ então a equação (3.7) possui uma solução única $y \in V_m$.

Demonstração *Seja v um elemento arbitrário de V_m e defina-se o operador $S : V_m \to V_m$ da forma seguinte:*

$$S(v) = \int_0^t p_2(t,s)v(s)ds + g_2(t), \quad t \in I.$$

Considerando $u = S(v)$ e $s = \lambda t$ podemos escrever

$$u(t) = \int_0^1 \lambda^{\mu-1} v(\lambda t)d\lambda + g_2(t).$$

Uma vez que $v \in C^m(I)$ e $g \in C^m(I)$ obtemos

$$u^{(j)}(t) = \int_0^1 \lambda^{\mu-1+j} v^{(j)}(\lambda t)d\lambda + g_2^{(j)}(t), \quad 0 \le j \le m.$$

Sendo assim se considerarmos dois elementos $u_1 = S(v_1)$ e $u_2 = S(v_2)$ vem

$$|u_1^{(j)} - u_2^{(j)}| \le \int_0^1 \lambda^{\mu-1+j} |v_1^{(j)}(\lambda t) - v_2^{(j)}(\lambda t)|d\lambda \le \frac{1}{\mu+j}\|v_1 - v_2\|_m, \ 0 \le j \le m.$$

Segue-se que

$$\|u_1 - u_2\|_m \le \frac{1}{\mu}\|v_1 - v_2\|_m,$$

ou seja,

$$\|S(v_1) - S(v_2)\|_m \le \frac{1}{\mu}\|v_1 - v_2\|_m.$$

A desigualdade anterior diz-nos que S é uma contracção desde que μ seja superior a 1. Uma vez que V_m é um espaço de Banach, e aplicando o teorema do ponto fixo de Banach, podemos afirmar que, para $\mu > 1$, S tem um e só um ponto fixo $y \in V_m$, ou seja, o problema (3.7) tem solução única. □

3.1.3 Resolução Numérica

Vamos considerar duas aplicações da equação (3.7) onde as soluções exactas são $y(t) = t^\alpha$ e $y(t) = t^\alpha \ln t$:

- $y(t) = t^\alpha$

Neste caso a função $g_2(t)$ toma a forma

$$\begin{aligned} g_2(t) &= t^\alpha - \int_0^t \frac{s^{\mu-1}}{t^\mu} s^\alpha ds \\ &= t^\alpha - \frac{1}{t^\mu}\frac{t^{\mu+\alpha}}{\mu+\alpha} \\ &= t^\alpha \left(1 - \frac{1}{\mu+\alpha}\right) \end{aligned}$$

- $y(t) = t^\alpha \ln t$

Neste caso a função $g_2(t)$ toma a forma,

$$\begin{aligned} g_2(t) &= t^\alpha \ln t - \int_0^t \frac{s^{\mu-1}}{t^\mu} s^\alpha ln s ds \\ &= t^\alpha \ln t - \frac{1}{t^\mu} \int_0^t s^{\mu+\alpha-1} ln s ds \\ &= t^\alpha \ln t - \frac{1}{t^\mu} \left(\frac{t^{\mu+\alpha}}{\mu+\alpha} \ln t - \int_0^t \frac{s^{\mu+\alpha+1}}{\mu+\alpha} ds \right) \\ &= t^\alpha \ln t - \frac{1}{t^\mu} \left(\frac{t^{\mu+\alpha}}{\mu+\alpha} \ln t - \frac{t^{\mu+\alpha}}{(\mu+\alpha)^2} \right) \\ &= t^\alpha \ln t - \frac{t^\alpha}{\mu+\alpha} \left(\ln t - \frac{1}{\mu+\alpha} \right) \\ &= t^\alpha \left(\ln t - \frac{1}{\mu+\alpha} \left(\ln t - \frac{1}{\mu+\alpha} \right) \right) \end{aligned}$$

O parâmetro α está relacionado com a regularidade da solução e μ com a singularidade do núcleo. A existência de solução depende destes dois parâmetros de acordo com o teorema 3.2.

Com o objectivo de obter soluções aproximadas para a equação (3.7) iremos utilizar métodos de colocação nos espaços $S_r^d(Z_N)$ e $\tilde{S}_r^d(Z_N)$. Para tal começaremos por efectuar colocações em $S_0^{-1}(Z_N)$ e $\tilde{S}_0^{-1}(Z_N)$, seguidamente em $\tilde{S}_1^{-1}(Z_N)$ com o objectivo de obter melhores ordens de convergência, e por fim em $\tilde{S}_2^{-1}(Z_N)$ para uma ordem de convergência ainda melhor.

Assim teremos: numa primeira fase, vamos obter soluções numéricas por métodos de colocação de três ordens de convergência:

Colocação em $S_0^{-1}(Z_N)$ e $\tilde{S}_0^{-1}(Z_N)$: para o qual esperaremos ordens de convergência dadas pelos teoremas 3.3, 3.4 e 3.5 e onde estudaremos os casos da tabela (3.1).

• Euler	$O(h)$
• À direita	$O(h)$
• Ponto médio	$O(h)$

Tabela 3.1: Colocação em $S_0^{-1}(Z_N)$ e $\tilde{S}_0^{-1}(Z_N)$.

Colocação em $\tilde{S}_1^{-1}(Z_N)$: para o qual esperaremos ordens de convergência dadas pelos teoremas 3.6 e 2.10, e onde estudaremos os casos da tabela (3.2).

• Lobatto (Trapésios)	$O(h^2)$	$c_0 = 0$	$c_1 = 1$
• Radau II	$O(h^3)$	$c_0 = 1/2$	$c_1 = 1$
• Gauss	$O(h^2)$	$c_0 = 1/3$	$c_1 = 1$

Tabela 3.2: Colocação em $\tilde{S}_1^{-1}(Z_N)$.

Colocação em $\tilde{S}_2^{-1}(Z_N)$: para o qual esperaremos ordens de convergência dadas pelo teorema 2.10, e onde estudaremos os casos da tabela (3.3).

• Lobatto (Simpson)	$O(h^4)$	$c_0 = 0$	$c_1 = 1/2$	$c_2 = 1$
• Radau II	$O(h^5)$	$c_0 = (3-\sqrt{3})/6$	$c_1 = (3+\sqrt{3})/6$	$c_2 = 1$
• Gauss	$O(h^3)$	$c_0 = (4-\sqrt{6})/10$	$c_1 = (4+\sqrt{6})/10$	$c_2 = 1$

Tabela 3.3: Colocação em $\tilde{S}_2^{-1}(Z_N)$.

Nos gráficos deste trabalho as linhas contínuas servem apenas para visualização, pois apenas os pontos indicados refletem resultados obtidos. Para cada um destes métodos serão estudadas as soluções exactas $y(t) = t^\alpha$ e $y(t) = t^\alpha \log(t)$, sendo designadas por exemplo para o método de Euler por `euler` e `euler(log)` respectivamente.

A utilização dos vários casos irá permitir identificar a precisão das soluções face ao seu custo computacional; iremos comparar os vários casos observando e analisando a precisão das soluções face ao tempo de computação. Também procuraremos identificar, para os vários casos, situações de superconvergência.

O cálculo experimental, das ordens de convergência dos métodos, vai ser feito com base no seguinte:

Suponhamos que estamos a aproximar a solução da nossa equação na rede $t_0, t_1, \ldots, t_N$. Se o método tem ordem p (ver definição 2.8), então desprezando os termos de ordem superior em h, verifica-se

$$E_n^h \approx ch^p, \quad h \to 0, \tag{3.13}$$

onde E_n^h representa o módulo do erro absoluto cometido, no ponto t_n, quando se utiliza um passo h. Se, em vez de utilizarmos o passo h, utilizarmos $h/2$ então

$$E_n^{h/2} \approx c\left(\frac{h}{2}\right)^p, \quad h \to 0. \tag{3.14}$$

Dividindo as equações (3.13) e (3.14) membro a membro obtém-se:

$$\frac{E_n^h}{E_n^{h/2}} \approx 2^p, \quad h \to 0.$$

Tirando o logaritmo na equação anterior vem,

$$p \approx \frac{\log_{10}\left(E_n^h / E_n^{h/2}\right)}{\log_{10}(2)}, \quad h \to 0. \tag{3.15}$$

Esta é uma forma prática de obter uma estimativa para a ordem de convergência de um método numérico.

Nas várias experiências numéricas iremos ter em conta as ordens e regularidades (da solução exacta y) sugeridas nos teoremas 3.3, 3.4, 3.5, 3.6, 2.9 e 2.10 e que são resumidas nas tabelas 3.4 e 3.5 (ver ainda observação 2.4).

Método	Regularidade de y	Ordem Global de Convergência
Euler	C^1	1
Direita	C^1	1
Médio	C^1	1
Trapézios	C^2	2
Simpson	C^3	3

Tabela 3.4: Relação entre métodos de colocação e ordens globais esperadas.

Método	Regularidade de y	Ordem Local de Convergência
Radau II (em S_1^{-1})	C^3	3
Gauss (em S_1^{-1})	C^4	2
Radau II (em S_2^{-1})	C^5	5
Gauss (em S_2^{-1})	C^6	3

Tabela 3.5: Relação entre métodos de colocação e ordens locais esperadas.

Numa segunda fase, com o objectivo de melhorar a precisão sem aumentar muito o custo computacional, utilizaremos o método de extrapolação de Richardson e algoritmo-E. Estes métodos serão aplicados apenas nos casos com menor ordem de convergência, à partida, e onde poderemos obter maiores ganhos com a aplicação de métodos de extrapolação.

A precisão e o desempenho dos métodos vai ser analizada em termos do erro relativo e do número de algarismos significativos. Lembremo-nos que: diz-se que o número aproximado $\overline{x}$ tem as algarismos significativos, relativamente ao valor exacto x, se

$$as := -log_{10}|\delta_x|,$$

onde $\delta_x := (x - \overline{x})/x$ é o erro relativo de $\overline{x}$ relativamente a x. Em particular, o número de algarismos significativos no ponto extremo do intervalo, $t_N = T$, é o valor

$$as := -\log_{10} \frac{|y(T) - u(T)|}{|y(T)|}, \quad T = t_N,$$

onde y e u representam as soluções exacta e aproximada respectivamente.

Uma outra forma de determinar a ordem de convergência aproximada num ponto recorre ao número de algarismos significativos. Esta forma permite relacionar o número de algarismos significativos com a ordem de convergência do método num ponto. Assim, a ordem de convergência aproximada num determinado ponto é dada por

$$p \approx \frac{as(h/2) - as(h)}{\log_{10}(2)}, \quad h \to 0, \tag{3.16}$$

onde $as(h)$ e $as(h/2)$ são os números de algarismos significativos correspondentes à utilização do método com os passos h e $h/2$, respectivamente; note-se que (3.16) é equivalente a (3.15).

Os resultados numéricos apresentados neste trabalho foram efectuados num computador `DEC 10000 Alpha AXP` configurado com 2 processadores, 256 Mbytes de RAM e uma capacidade de cálculo em vírgula flutuante de 400 milhões de instruções por segundo (400 MFLOPS). Assim, os tempos de cálculo apresentados são referentes a esta arquitectura, podendo os valores variar de várias ordens de grandeza quando se utilizar os programas em anexo em outras arquitecturas.

3.2 Colocação nos Espaços $\tilde{S}_0^{-1}(Z_N)$ e $S_0^{-1}(Z_N)$

Os métodos de colocação em $\tilde{S}_0^{-1}(Z_N)$ e $S_0^{-1}(Z_N)$ baseiam-se em aproximar a solução exacta (parte regular da integranda) por uma função seccionalmente constante. A aproximação consiste em utilizar uma função seccionalmente constante que iguala a parte regular da integranda num certo ponto do intervalo. Consoante o ponto escolhido se encontra no extremo esquerdo, direito, ou no meio do intervalo, o método obtido designa-se por Euler, direita e ponto médio, respectivamente. A existência de uma singularidade no núcleo da integranda, decomponível numa parte regular e uma parte não regular, conduz a técnicas de integração produto.

Estes métodos têm uma precisão bastante baixa, mas a situação pode ser melhorada através do uso de algumas técnicas de extrapolação.

A regra de integração produto, baseada na regra dos rectângulos, tem uma ordem de grandeza do erro $O(h)$ [31]. Se esta ordem de grandeza fôr preservada na solução da equação integral (e veremos que em geral isto é verdade), então podemos esperar que

$$|u(t_n) - y(t_n)| = O(h),$$

onde u é a solução aproximada e y é a solução exacta da referida equação integral. É de esperar que estes métodos possuam, em geral, baixa precisão para um valor razoável de h. Contudo, eles têm interesse prático devido à simplicidade dos pesos de integração e da resolução da equação de colocação.

3.2.1 Método de Euler

Começa-se com um método numérico bastante simples que se chama método de Euler em analogia com o método correspondente em equações diferenciais ordinárias. O método de Euler é um método de colocação no espaço de funções seccionalmente constantes $\tilde{S}_0^{-1}(Z_N)$ definido por

$$\tilde{S}_0^{-1}(Z_N) := \{u : u(t) = u(t_n) \in \pi_0, t \in [t_n, t_{n+1}), n = 0, \ldots, N-1, u(T) = u(t_N)\},$$

onde $Z_N := \{t_n : n = 1, \ldots, N-1\}$ é o conjuntos dos pontos interiores associados à rede

$$\Pi_N : 0 = t_0 < t_1 < \ldots < t_N = T, \;\; N \geq 1,$$

no intervalo $[0, T]$. (ver figura 3.1)

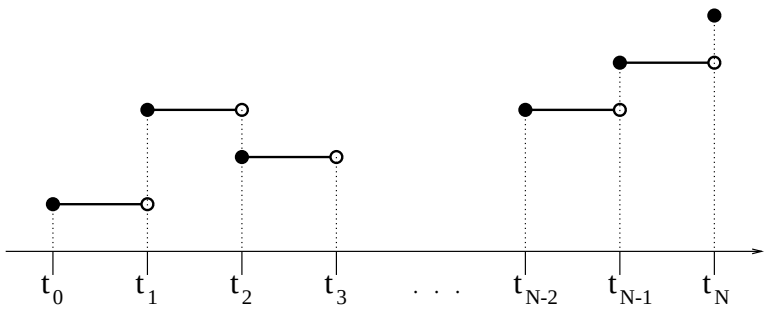

Figura 3.1: Aproximação seccionalmente constante (à esquerda).

3.2.1.1 Construção

Pretende-se encontrar o elemento $u \in \tilde{S}_0^{-1}(Z_N)$ que satisfaz a equação de colocação

$$\begin{aligned} u(t_n) &= g_2(t_n) + \int_0^{t_n} p_2(t_n, s)u(s)ds, \ \ n = 1, \ldots, N, \\ u(t_0) &:= y(0) = \frac{\mu}{\mu - 1} g_2(0), \end{aligned} \tag{3.17}$$

onde $y(0)$ é calculado a partir da equação (3.7), considerando a transformação de variável $s = \lambda t$ e tomando o limite de t quando t tende para zero.

Admitindo que em cada subintervalo $[t_i, t_{i+1})$ a função u é constante e igual a $u(t_i)$, podemos escrever

$$u(t_n) = g_2(t_n) + \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} p_2(t_n, s)ds u(t_i), \ \ n = 1, \ldots, N. \tag{3.18}$$

Uma vez que $p_2(t, s) = (s/t)^\mu / s$ e $t_i = ih$, $i = 0, 1, \ldots N$, o integral que surge em (3.18) pode ser calculado exactamente e a equação de colocação (3.18) toma a forma:

$$u(t_n) = g_2(t_n) + \sum_{i=0}^{n-1} \frac{1}{\mu n^\mu}((i+1)^\mu - i^\mu)u(t_i), \ \ n = 1, \ldots, N.$$

Tirando para fora do somatório os termos que não dependem do seu índice i, obtemos o algoritmo correspondente ao método de Euler:

$$\begin{aligned} u(t_0) &= \frac{\mu}{\mu - 1} g_2(t_0), \\ u(t_n) &= g_2(t_n) + \frac{1}{\mu n^\mu} \sum_{i=0}^{n-1} ((i+1)^\mu - i^\mu)u(t_i), \ \ n = 1, \ldots, N. \end{aligned} \tag{3.19}$$

3.2.1.2 Convergência

Nesta secção iremos demonstrar que, sob certas condições, o método de Euler, quando aplicado à equação (3.7) tem ordem de convergência global $O(h)$.

Teorema 3.3 *Suponhamos que g, na equação (3.7), pertence à classe $C^1(I)$ e $\mu > 1$, então a equação de colocação (3.17), (3.18) possui uma solução única $u \in \tilde{S}_0^{-1}(Z_N)$. Além disso, a função erro $e(t) := y(t) - u(t)$ satisfaz a desigualdade*

$$\|e\|_\infty \leq C_1 h,$$

onde $h := t_{i+1} - t_i$, $i = 0, 1, \ldots, N-1$, C_1 é uma constante independente de h e

$$\|e\|_\infty := \max_{t \in I} |e(t)|. \tag{3.20}$$

Demonstração Seja

$$e_n(t) := y(t) - u(t_n), \;\; t \in [t_n, t_{n+1}),$$

a função erro correspondente ao intervalo $[t_n, t_{n+1})$ e escrevamos a equação (3.7) na forma

$$y(t) = g_2(t) + \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} p_2(t,s)y(s)ds, \;\; t \in I. \tag{3.21}$$

Esta equação verifica-se para todo o t no intervalo I, em particular nos pontos $t_0, t_1, \ldots, t_N$. Subtraindo a equação (3.21), no ponto t_n, à equação de colocação (3.18) obtemos

$$\begin{aligned} e_n(t_n) &= y(t_n) - u(t_n) \\ &= \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} p_2(t_n,s)y(s)ds - \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} p_2(t_n,s)u(t_i)ds, \;\; n = 1, \ldots, N. \end{aligned}$$

Aplicando a propriedade aditiva dos integrais podemos concluir que

$$e_n(t_n) = \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} p_2(t_n,s) \left(y(s) - u(t_i)\right) ds, \;\; n = 1, \ldots, N,$$

ou seja,

$$e_n(t_n) = \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} p_2(t_n,s)e_i(s)ds, \;\; n = 1, \ldots, N. \tag{3.22}$$

Uma vez que $g \in C^1(I)$ e $\mu > 1$ sabemos que, pelo teorema 3.2, $y \in C^1(I)$. Por conseguinte, para cada $s \in [t_i, t_{i+1})$ podemos afirmar que

$$e_i(s) = e_i(t_i) + (s - t_i)y'(\xi_i), \;\; \xi_i \in (t_i, s), \;\; i = 0, 1, \ldots, N.$$

Mas, como $s \in [t_i, t_{i+1})$, $(s - t_i) \leq h$ vem

$$e_i(s) = e_i(t_i) + O(h), \;\; i = 0, 1, \ldots, N, \tag{3.23}$$

e sendo assim, podemos escrever a equação (3.22) na forma

$$e_n(t_n) = \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} p_2(t_n,s)e_i(t_i)ds + \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} p_2(t_n,s)ds\, O(h), \;\; n = 1, \ldots, N. \tag{3.24}$$

Uma vez que $p_2(t,s) = (s/t)^\mu/s$ vem

$$\sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} p_2(t_n,s)ds = \int_0^{t_n} p_2(t_n,s)ds = \frac{1}{\mu}. \tag{3.25}$$

Substituindo este resultado na equação (3.24) e depois majorando obtemos

$$|e_n(t_n)| \leq \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} p_2(t_n,s)|e_i(t_i)|ds + \frac{1}{\mu}O(h), \;\; n = 1, \ldots, N.$$

Definindo

$$E_e := \max_{0\leq i\leq N} |e_i(t_i)|, \tag{3.26}$$

vem

$$|e_n(t_n)| \leq \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} p_2(t_n, s)ds E_e + O(h), \;\; n = 1, \ldots, N,$$

e utilizando (3.25) obtemos

$$|e_n(t_n)| \leq \frac{1}{\mu} E_e + O(h), \;\; n = 1, \ldots, N.$$

Por conseguinte, com base em (3.26), segue que

$$E_e \leq \frac{1}{\mu} E_e + O(h).$$

Mas $\mu > 1$ logo resolvendo a inequação em ordem a E_e obtém-se $E_e = O(h)$. Sendo assim concluimos que nos pontos da rede o método tem ordem 1.

Uma vez que o erro num ponto $s \in [t_i, t_{i+1})$ é a soma do erro no ponto t_i, da rede, com algo que é um $O(h)$ (ver (3.23)) podemos concluir que o método de Euler tem ordem global de convergência $O(h)$. □

3.2.2 Método à Direita

O método à direita é um método de colocação no espaço das funções seccionalmente constantes $S_0^{-1}(Z_N)$ definido por

$$S_0^{-1}(Z_N) := \{u : u(0) = u(t_0),\; u(t) = u(t_{n+1}) \in \pi_0,\; t \in (t_n, t_{n+1}],\; n = 0, \ldots, N-1\},$$

onde Z_N é definido como no caso do método de Euler. (ver figura 3.2)

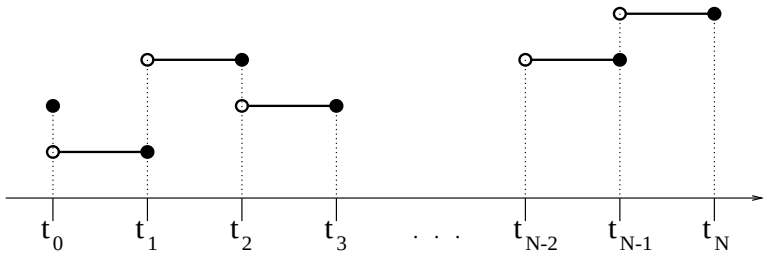

Figura 3.2: Aproximação seccionalmente constante (à direita).

3.2.2.1 Construção

Neste caso procuramos um elemento $u \in S_0^{-1}(Z_N)$ que satisfaça a equação de colocação

$$\begin{aligned} u(t_n) &= g_2(t_n) + \int_0^{t_n} p_2(t_n, s)u(s)ds, \quad n = 1, \ldots, N, \\ u(t_0) &:= y(0) = \frac{\mu}{\mu - 1} g_2(0), \end{aligned} \tag{3.27}$$

onde $y(0)$ é calculado como no método de Euler.

Uma vez que a função u é seccionalmente constante tomando o valor $u(t_{i+1})$ no intervalo $(t_i, t_{i+1}]$ a equação anterior toma a forma:

$$u(t_n) = g_2(t_n) + \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} p_2(t_n, s)ds u(t_{i+1}), \quad n = 1, \ldots, N. \tag{3.28}$$

De forma análoga à usada na secção 3.2.1 a equação de colocação (3.28) toma a forma

$$u(t_n) = g_2(t_n) + \sum_{i=0}^{n-1} \frac{1}{\mu n^\mu}((i+1)^\mu - i^\mu)u(t_{i+1}), \quad n = 1, \ldots, N,$$

ou seja,

$$u(t_n) = g_2(t_n) + \sum_{i=0}^{n-2} \frac{1}{\mu n^\mu}((i+1)^\mu - i^\mu)u(t_{i+1}) + \frac{1}{\mu n^\mu}(n^\mu - (n-1)^\mu)u(t_n) \quad n = 1, \ldots, N.$$

Note-se que $1 - \frac{n^\mu - (n-1)^\mu}{\mu n^\mu} \neq 0$, logo esta equação pode ser resolvida em ordem a $u(t_n)$. Por conseguinte, surge o seguinte algoritmo:

$$\begin{aligned} u(t_0) &= \frac{\mu}{\mu - 1} g_2(t_0), \\ u(t_n) &= \frac{1}{(\mu - 1)n^\mu + (n-1)^\mu} \left(\mu n^\mu g_2(t_n) + \sum_{i=0}^{n-1} (i^\mu - (i-1)^\mu)u(t_i) \right), \\ & \qquad n = 1, 2, \ldots, N. \end{aligned}$$

3.2.2.2 Convergência

Em relação a este método a ordem de convergência também é $O(h)$ e tém-se o seguinte resultado:

Teorema 3.4 *Suponhamos que g, na equação (3.7), pertence à classe $C^1(I)$ e $\mu > 1$, então a equação de colocação (3.27), (3.28) possui uma solução única $u \in S_0^{-1}(Z_N)$. Além disso, a função erro $e(t) := y(t) - u(t)$ satisfaz a desigualdade*

$$\|e\|_\infty \leq C_2 h,$$

onde $h := t_{i+1} - t_i$, $i = 0, 1, \ldots, N-1$, C_2 é uma constante independente de h e $\|e\|_\infty$ é definida como em (3.20).

Demonstração Seja

$$e_n(t) := y(t) - u(t_{n+1}),$$

então, de forma semelhante à utilizada na demonstração do teorema 3.3 obtemos a seguinte equação de erro:

$$e_n(t_{n+1}) = \sum_{i=0}^{n} \int_{t_i}^{t_{i+1}} p_2(t_{n+1}, s) e_i(s) ds, \quad n = 0, 1, \ldots, N-1. \tag{3.29}$$

Uma vez que $g \in C^1$ e $\mu > 1$ concluimos de novo que $y \in C^1(I)$. Por conseguinte, podemos desenvolver $e_i(s)$, $s \in (t_i, t_{i+1}]$, em série de Taylor em torno do ponto t_{i+1} até à primeira ordem e vem

$$e_i(s) = e_i(t_{i+1}) + (s - t_{i+1}) y'(\xi_i), \quad \xi_i \in (s, t_{i+1}), \quad s \in (t_i, t_{i+1}], \quad n = 0, 1, \ldots, N-1,$$

ou seja,

$$e_i(s) = e_i(t_{i+1}) + O(h), \quad i = 0, 1, \ldots, N-1. \tag{3.30}$$

Utilizando esta equação e o facto de que

$$\sum_{i=0}^{n} \int_{t_i}^{t_{i+1}} p_2(t_{n+1}, s) ds = \frac{1}{\mu}, \tag{3.31}$$

podemos escrever a equação (3.29) na forma

$$e_n(t_{n+1}) = \sum_{i=0}^{n} \int_{t_i}^{t_{i+1}} p_2(t_{n+1}, s) e_i(t_{i+1}) ds + \frac{1}{\mu} C h, \quad n = 0, 1, \ldots, N-1, \tag{3.32}$$

onde a constante C não depende de h. Defina-se, para abreviar,

$$w_{ni} := \int_{t_i}^{t_{i+1}} p_2(t_{n+1}, s) ds.$$

Deste modo a equação (3.32) pode ser reescrita sob a forma

$$e_n(t_{n+1})(1 - w_{nn}) = \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} p_2(t_{n+1}, s) e_i(t_{i+1}) ds + \frac{1}{\mu} C h, \quad n = 0, 1, \ldots, N-1,$$

donde

$$|e_n(t_{n+1})| \leq \sum_{i=0}^{n-1} \frac{|w_{ni}|}{|1-w_{nn}|}|e_i(t_{i+1})| + \frac{1}{\mu}\frac{|C|}{|1-w_{nn}|}h, \quad n = 0, 1, \ldots, N-1. \tag{3.33}$$

Uma vez que $|1 - w_{nn}| \neq 0$ e definindo

$$E_d := \max_{0 \leq i \leq N-1} |e_i(t_{i+1})|$$

a desigualdade (3.33) adquire a forma

$$|e_n(t_{n+1})| \leq E_d \sum_{i=0}^{n-1} \frac{|w_{ni}|}{|1-w_{nn}|} + |C_2|h, \quad n = 0, 1, \ldots, N-1. \tag{3.34}$$

Note-se que a sucessão

$$\sum_{i=0}^{n-1} \frac{|w_{ni}|}{|1-w_{nn}|} = \frac{n^\mu}{(\mu-1)(n+1)^\mu + n^\mu} = \frac{1}{(\mu-1)\frac{(n+1)^\mu}{n^\mu} + 1}$$

tende para $1/\mu$ por valores inferiores quando $n \to \infty$, sendo $1/\mu < 1$, por conseguinte existe uma constante real positiva α tal que,

$$\sum_{i=0}^{n-1} \frac{|w_{ni}|}{|1-w_{nn}|} \leq \alpha < 1,$$

logo podemos majorar o segundo membro da enequação (3.34) e escrever

$$|e_n(t_{n+1})| \leq E_d\alpha + |C_2|h,$$

donde

$$E_d \leq E_d\alpha + O(h).$$

Como $\alpha < 1$ conclui-se que $E_d = O(h)$, e uma vez que se verifica (3.30) fica garantida a ordem global de convergência $O(h)$ do método à direita. □

3.2.3 Método do Ponto Médio

Neste caso o espaço de colocação é o espaço $\tilde{S}_0^{-1}(Z_N)$ definido por

$$\tilde{S}_0^{-1}(Z_N) := \{u : u(t) = u(t_n) \in \pi_0, t \in [t_n, t_{n+1}), n = 0, \ldots, N-1, u(T) = u(t_N)\},$$

onde $Z_N := \{t_{n+\frac{1}{2}} := t_n + h/2, n = 0, 1, \ldots, N-1\}$ é o conjunto dos pontos de colocação associados à rede

$$\Pi_N : 0 = t_0 < t_1 < \ldots < t_N = T, \;\; N \geq 1,$$

no intervalo $[0, T]$. (ver figura 3.3)

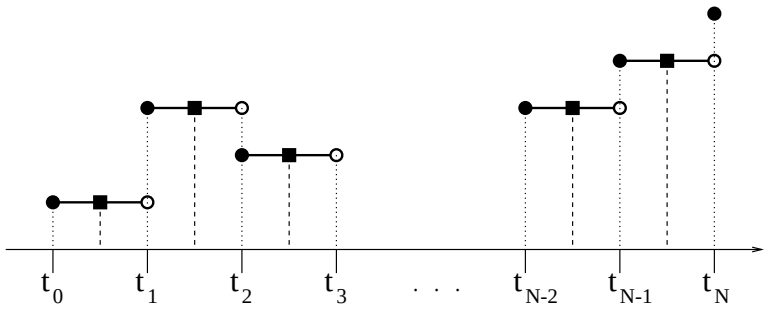

Figura 3.3: Aproximação seccionalmente constante (ao centro).

3.2.3.1 Construção

No método do ponto médio procuramos um elemento u do espaço $\tilde{S}_0^{-1}(Z_N)$ que satisfaça a equação de colocação:

$$u(t_{n+\frac{1}{2}}) = g_2(t_{n+\frac{1}{2}}) + \int_0^{t_{n+\frac{1}{2}}} p_2(t_{n+\frac{1}{2}}, s)u(s)ds, \quad n = 0, 1, \ldots, N-1. \tag{3.35}$$

Admitindo que u é seccionalmente constante tal que

$$u(s) = u(t_{n+\frac{1}{2}}), \quad s \in [t_n, t_{n+1}), \quad n = 0, 1, \ldots, N-1,$$

a equação anterior pode escrever-se na forma

$$u(t_{n+\frac{1}{2}}) = g_2(t_{n+\frac{1}{2}}) + \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} p_2(t_{n+\frac{1}{2}}, s)dsu(t_{i+\frac{1}{2}}) + \int_{t_n}^{t_{n+\frac{1}{2}}} p_2(t_{n+\frac{1}{2}}, s)dsu(t_{n+\frac{1}{2}}). \tag{3.36}$$

Para $n = 0$ obtém-se, da equação (3.36),

$$u(t_{0+\frac{1}{2}}) = g_2(t_{0+\frac{1}{2}}) + \int_0^{t_{0+\frac{1}{2}}} p_2(t_{0+\frac{1}{2}}, s)dsu(t_{0+\frac{1}{2}}).$$

Mas

$$\int_0^{t_{0+\frac{1}{2}}} p_2(t_{0+\frac{1}{2}}, s)ds = \frac{1}{\mu},$$

logo

$$u(t_{0+\frac{1}{2}}) = g_2(t_{0+\frac{1}{2}}) + \frac{1}{\mu}u(t_{0+\frac{1}{2}}).$$

Segue que,

$$u(t_{0+\frac{1}{2}}) = \frac{\mu}{\mu - 1}g_2(t_{0+\frac{1}{2}}). \tag{3.37}$$

Para $n = 1, 2, \ldots, N-1$ a equação (3.36) transforma-se em,

$$u(t_{n+\frac{1}{2}}) = g_2(t_{n+\frac{1}{2}}) + \frac{1}{\mu}\sum_{i=0}^{n-1} \frac{(i+1)^\mu - i^\mu}{(n+1/2)^\mu}u(t_{i+\frac{1}{2}}) + \frac{1}{\mu}\frac{(n+1/2)^\mu - n^\mu}{(n+1/2)^\mu}u(t_{n+\frac{1}{2}}), \tag{3.38}$$

onde se utilizou

$$\int_{t_i}^{t_k} p_2(t_{n+\frac{1}{2}}, s)ds = \frac{1}{\mu}\frac{k^\mu - i^\mu}{(n+1/2)^\mu}, \quad i = 1, \ldots, N-1.$$

Uma vez que

$$1 - \frac{1}{\mu}\frac{(n+\frac{1}{2})^\mu - n^\mu}{(n+\frac{1}{2})^\mu} \neq 0,$$

podemos resolver a equação (3.38) em ordem a $u(t_{n+\frac{1}{2}})$, e utilizando (3.37) obtém-se o seguinte esquema chamado método do ponto médio:

$$\begin{aligned} u(t_{0+\frac{1}{2}}) &= \frac{\mu}{\mu-1} g_2(t_{0+\frac{1}{2}}) \\ u(t_{n+\frac{1}{2}}) &= \frac{1}{(\mu-1)(n+\frac{1}{2})^\mu + n^\mu}\left(\mu\left(n+\frac{1}{2}\right)^\mu g_2(t_{n+\frac{1}{2}}) + \sum_{i=0}^{n-1}((i+1)^\mu - i^\mu)u(t_{i+\frac{1}{2}})\right), \\ & \qquad n = 1, 2, \ldots, N. \end{aligned}$$

3.2.3.2 Convergência

Em seguida apresentamos um resultado que nos diz que o método do ponto médio tem a mesma ordem de convergência dos métodos anteriores, $O(h)$:

Teorema 3.5 *Suponhamos que g, na equação (3.7), pertence à classe $C^1(I)$ e $\mu > 1$, então a equação de colocação (3.35) possui uma solução única $u \in \tilde{S}_0^{-1}(Z_N)$. Além disso, a função erro $e(t) := y(t) - u(t)$ satisfaz a desigualdade*

$$\|e\|_\infty \leq C_3 h,$$

onde $h := t_{i+1} - t_i$, $i = 0, 1, \ldots, N-1$, C_3 é uma constante independente de h e $\|e\|_\infty$ é definida como em (3.20).

Demonstração Seja

$$e_n(t) := y(t) - u(t_{n+\frac{1}{2}}), \quad t \in [t_n, t_{n+1}),$$

a função erro correspondente ao intervalo $[t_n, t_{n+1})$ e escrevamos a equação (3.7) no ponto $t_{n+\frac{1}{2}}$, onde $n = 0, 1, \ldots, N-1$:

$$\begin{aligned} y(t_{n+\frac{1}{2}}) &= g_2(t_{n+\frac{1}{2}}) + \sum_{i=0}^{n-1}\int_{t_i}^{t_{i+1}} p_2(t_{n+\frac{1}{2}}, s)y(s)ds \\ &\quad + \int_{t_n}^{t_{n+\frac{1}{2}}} p_2(t_{n+\frac{1}{2}}, s)y(s)ds \quad t \in I. \end{aligned}$$

Subtraindo esta equação à equação (3.36), e aplicando a propriedade aditiva dos integrais, obtém-se

$$\begin{aligned} e_n(t_{n+\frac{1}{2}}) &= y(t_{n+\frac{1}{2}}) - u(t_{n+\frac{1}{2}}) = \\ &= \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} p_2(t_{n+\frac{1}{2}}, s)(y(s) - u(t_{i+\frac{1}{2}}))ds \\ &\quad + \int_{t_n}^{t_{n+\frac{1}{2}}} p_2(t_{n+\frac{1}{2}}, s)(y(s) - u(t_{n+\frac{1}{2}}))ds, \end{aligned}$$

ou seja,

$$e_n(t_{n+\frac{1}{2}}) = \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} p_2(t_{n+\frac{1}{2}}, s)e_i(s)ds + \int_{t_n}^{t_{n+\frac{1}{2}}} p_2(t_{n+\frac{1}{2}}, s)e_n(s)ds. \tag{3.39}$$

Uma vez que $g \in C^1$ e $\mu > 1$ sabemos que, pelo teorema 3.2, $y \in C^1(I)$. Por conseguinte, para cada $s \in [t_i, t_{i+1})$ podemos afirmar que

$$e_i(s) = e_i(t_{i+\frac{1}{2}}) + (s - t_{i+\frac{1}{2}})y'(\xi_i), \xi_i \in (s, t_{i+\frac{1}{2}}). \tag{3.40}$$

Mas como $(s - t_{i+\frac{1}{2}}) \leq h$ vem

$$e_i(s) = e_i(t_{i+\frac{1}{2}}) + O(h),$$

e sendo assim, podemos escrever a equação (3.39) na forma:

$$\begin{aligned} e_n(t_{n+\frac{1}{2}}) &= \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} p_2(t_{n+\frac{1}{2}}, s)e_i(t_{i+\frac{1}{2}})ds + \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} p_2(t_{n+\frac{1}{2}}, s)dsO(h) \\ &+ \int_{t_n}^{t_{n+\frac{1}{2}}} p_2(t_{n+\frac{1}{2}}, s)e_n(t_{n+\frac{1}{2}})ds + \int_{t_n}^{t_{n+\frac{1}{2}}} p_2(t_{n+\frac{1}{2}}, s)dsO(h), \end{aligned}$$

ou seja,

$$\begin{aligned} e_n(t_{n+\frac{1}{2}}) &= \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} p_2(t_{n+\frac{1}{2}}, s)e_i(t_{i+\frac{1}{2}})ds \\ &+ \int_{t_n}^{t_{n+\frac{1}{2}}} p_2(t_{n+\frac{1}{2}}, s)e_n(t_{n+\frac{1}{2}})ds + \int_{0}^{t_{n+\frac{1}{2}}} p_2(t_{n+\frac{1}{2}}, s)dsO(h). \end{aligned}$$

Uma vez que $p_2(t, s) = (s/t)^\mu/\mu$ vem

$$\int_0^{t_{n+\frac{1}{2}}} p_2(t_{n+\frac{1}{2}}, s)ds = \frac{1}{\mu}, \tag{3.41}$$

logo

$$\begin{aligned} e_n(t_{n+\frac{1}{2}}) &= \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} p_2(t_{n+\frac{1}{2}}, s)e_i(t_{i+\frac{1}{2}})ds \\ &+ \int_{t_n}^{t_{n+\frac{1}{2}}} p_2(t_{n+\frac{1}{2}}, s)e_n(t_{n+\frac{1}{2}})ds + \frac{1}{\mu}O(h). \end{aligned}$$

Definindo

$$v_{ni} := \int_{t_i}^{t_{i+1}} p_2(t_{n+\frac{1}{2}}, s)ds, \quad i = 0, 1, \ldots, n-1.$$

$$v_{nn} := \int_{t_n}^{t_{n+\frac{1}{2}}} p_2(t_{n+\frac{1}{2}}, s)ds,$$

podemos escrever a equação anterior do seguinte modo:

$$(1 - v_{nn})e_n(t_{n+\frac{1}{2}}) = \sum_{i=0}^{n-1} v_{ni} e_i(t_{i+\frac{1}{2}}) + O(h).$$

Observando que $1 - v_{nn} \neq 0$ e definindo

$$E_m := \max_{0 \leq i \leq N-1} |e_i(t_{i+\frac{1}{2}})|,$$

obtém-se

$$|e_n(t_{n+\frac{1}{2}})| \leq E_m \sum_{i=0}^{n-1} \frac{|v_{ni}|}{|1 - v_{nn}|} + O(h). \tag{3.42}$$

Note-se que a sucessão

$$\sum_{i=0}^{n-1} \frac{|v_{ni}|}{|1 - v_{nn}|} = \frac{n^\mu}{(\mu - 1)(n + 1/2)^\mu + n^\mu} = \frac{1}{(\mu - 1)(1 + 1/(2n))^\mu + 1},$$

tende para $1/\mu$ por valores inferiores quando $n \to \infty$, sendo $1/\mu < 1$, segue que existe uma constante β, real positiva, tal que:

$$\sum_{i=0}^{n-1} \frac{|v_{ni}|}{|1 - v_{nn}|} \leq \beta < 1.$$

Sendo assim, obtém-se da enequação (3.42),

$$|e_n(t_{n+\frac{1}{2}})| \leq E_m \beta + O(h),$$

logo

$$E_m \leq E_m \beta + O(h).$$

Como $\beta < 1$ é garantido que $E_m = O(h)$. Além disso $e_i(s) = e_i(t_{i+\frac{1}{2}}) + O(h)$, donde se conclui que $\|e\|_\infty = O(h)$. □

3.2.4 Análise de Resultados em $\tilde{S}_0^{-1}(Z_N)$ e $S_0^{-1}(Z_N)$

Nos teoremas 3.3, 3.4, e 3.5 mostrou-se que, para que os métodos de colocação em S_0^{-1} convirjam com ordem 1, basta que g, em (3.9), seja de classe $C^1(I)$ e $\mu > 1$. Os resultados que se vão analisar correspondem a dois casos: t^α e $t^\alpha lnt$. A escolha de $\alpha = 1.5, 2.5, \ldots$ é feita para que as funções g correspondentes satisfaçam a condição referida. Ir-se-á escolher $\mu = 1.5 > 1$. O caso $\alpha = 0.5$ não está

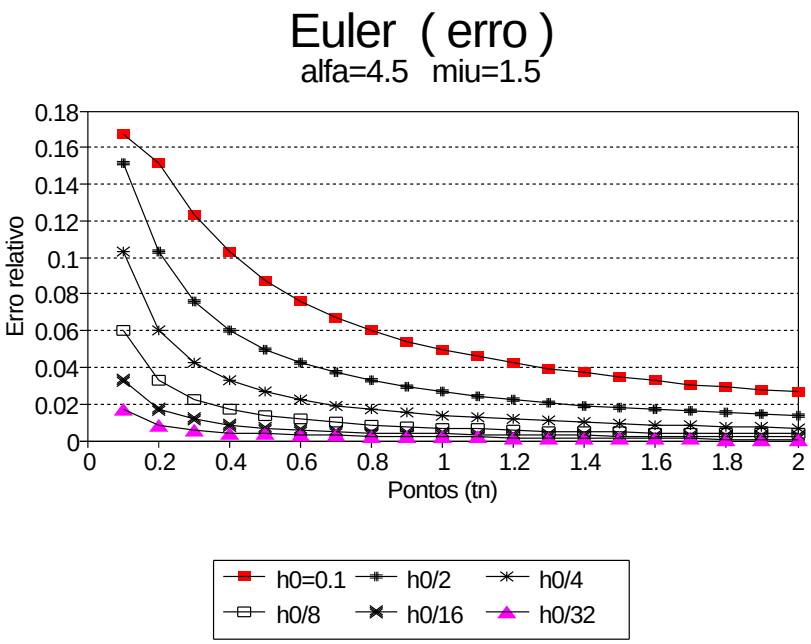

Figura 3.4: Erro do método de Euler para diversos valores de h.

englobado nas condições de convergência do método mas tem interesse averiguar o comportamento dos métodos face a valores de α não cobertos pelo teorema.

Observa-se que, à medida que t se aproxima do extremo direito do intervalo $t = 2.0$, o erro $e(t)$ vai diminuindo (ver figura 3.4). Esta propriedade de autocorrecção foi observada noutros exemplos pelo que é de suspeitar que ela seja algo inerente ao método e não dependente do problema a resolver. Além disso podemos observar que de facto a ordem é baixa, $O(h)$ (ver figura 3.5 e tabela 3.6). Apesar de tudo, em situações em que não seja necessário alta precisão, esta aproximação pode ser preferível a uma mais precisa, obtida a partir de um método mais com-

passo h	erro absoluto $E_N^h = y(2.0) - u(2.0)$	ordem de convergência $p \approx log_2(E_N^h / E_N^{h/2})$
0.1	1.943801630368	
0.05	0.945434447935	1.03983166
0.025	0.466152696718	1.02017482
0.0125	0.231441163189	1.01015713
0.00625	0.115312489823	1.00509670
0.003125	0.057554307456	1.00255297

Tabela 3.6: Ordem de convergência (método de Euler).

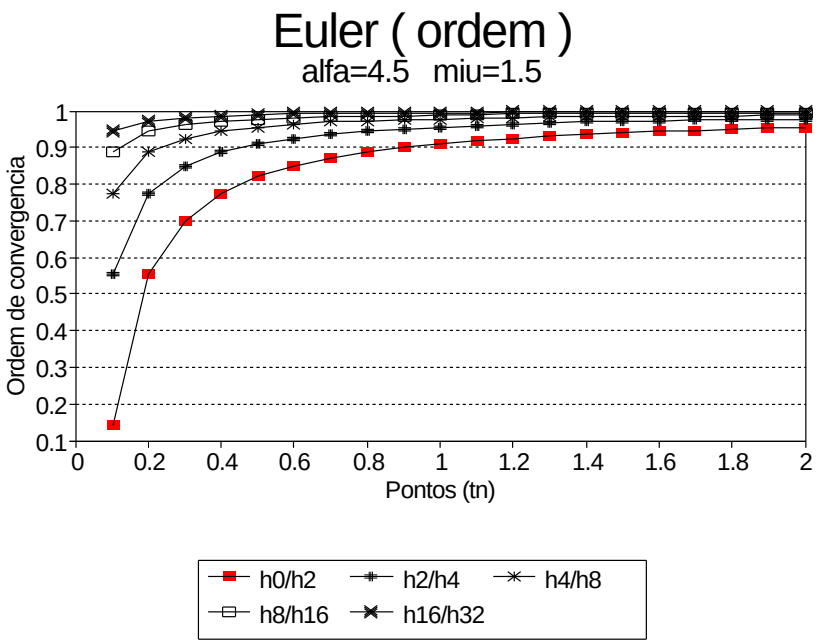

Figura 3.5: Ordem de convergência do método de Euler para diversos valores de h.

plicado. Observe-se que para se conseguir resultados um pouco mais precisos é necessário utilizar passos extremamente pequenos, o que involve trabalho computacional demasiadamente grande (tabela 3.7); ao passarmos de um passo $h = 0.1$ para $h = 0.00313$ gastamos 8340 vezes mais tempo e o resultado é 30 vezes mais preciso, ou seja, temos uma degradação da precisão por unidade de tempo de 278 vezes.

A grande vantagem deste método reside na sua simplicidade, podendo a sua baixa presisão ser ultrapassada utilizando técnicas de extrapolação (ver capítulo 4). Os resultados numéricos parecem indicar que o método é pouco sensível à variação dos parâmetros μ e α. Apesar de tudo para α perto de 1, 1.5, é onde se obtém um erro

tempo (s)	h	erro relativo
0.9760×10^{-3}	0.1	0.2642108×10^{-1}
0.2928×10^{-2}	0.05	0.1362985×10^{-1}
0.1366×10^{-1}	0.025	0.6922241×10^{-2}
0.5270×10^{-1}	0.0125	0.3488266×10^{-2}
0.2040×10^{0}	0.00625	0.1750959×10^{-2}
0.8140×10^{0}	0.00313	0.8771913×10^{-3}

Tabela 3.7: Relação entre os tempos de computação e erros para os diversos valores de h (método de Euler com $\alpha = 4.5$ e $\mu = 1.5$).

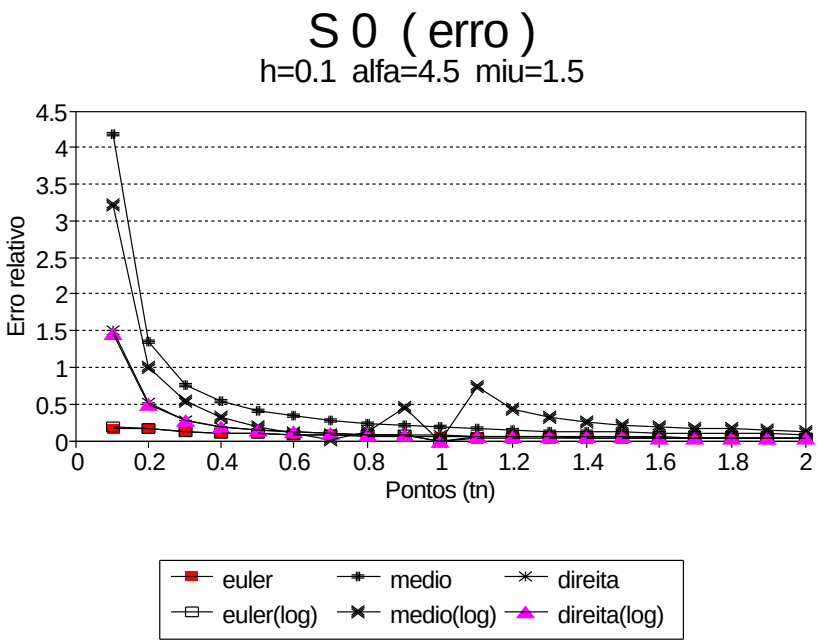

Figura 3.6: Erro relativo para os métodos de colocação em $\tilde{S}_0^{-1}(Z_N)$ e $S_0^{-1}(Z_N)$.

relativo menor; note-se que esta escolha corresponde a considerar $y \in C^1$, condição suficiente de convergência. Observa-se também robustez em relação à variação da condição inicial $g_2(t)$, μ e α; quando se muda de $y(t) = t^\alpha$ para $y(t) = t^\alpha lnt$, os seus erros relativos são idênticos (figuras 3.7, 3.8). Esta robustez em relação a α, μ e $g_2(t)$, já não se verifica no caso do método do ponto médio. Sendo assim, o método de Euler parece ser mais vantajoso que o método do ponto médio embora ambos sejam $O(h)$. Note-se que esta conclusão é baseada apenas no estudo de duas classes de funções. Pode acontecer que de uma forma geral isto não aconteça.

Para pontos próximo da singularidade, os valores aproximados, na vizinhança de $t = 0$, são melhores no método de Euler do que no método à direita, mas o valor aproximado da ordem de convergência aproxima-se de 1, por valores maiores ou menores que 1, consoante o método é o do ponto médio ou o de Euler. Como não se aconselha os métodos de ordem 1 com valores do passo h muito pequenos (devido ao esforço computacional) versus precisão, é preferível utilizar o método de Euler em vez do método à direita (figuras 3.6 e 3.9).

3.3 Colocação no Espaço $\tilde{S}_1^{-1}(Z_N)$

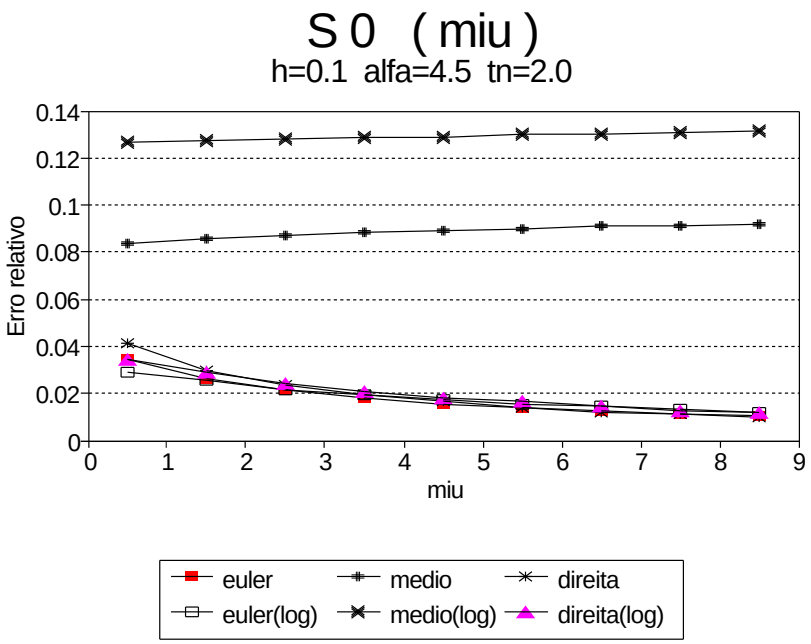

Figura 3.7: Dependência de μ para os métodos de colocação em $\tilde{S}_0^{-1}(Z_N)$ e $S_0^{-1}(Z_N)$.

Para contruir métodos de ordens superiores, é necessário utilizar regras de integração mais precisas. Sendo assim os métodos de colocação apresentados nesta secção baseiam-se em aproximar, a solução exacta, por uma função seccionalmente polinomial de grau 1. Quando a função aproximadora é contínua temos o método dos trapézios produto.

Consideremos a colocação no espaço das funções seccionalmente polinomiais de grau 1,

$$\begin{aligned}\tilde{S}_1^{-1}(Z_N) \quad := \{ \quad & u : u(t) = l_{1,n}(t)u(t_{1,n}) + l_{2,n}(t)u(t_{2,n}), u(t_{1,n}), u(t_{2,n}) \in \pi_0, \\ & t \in [t_n, t_{n+1}), n = 0, 1, \ldots, N-1, \\ & u(T) = l_{1,N-1}(T)u(t_{1,N-1}) + l_{2,N-1}(T)u(t_{2,N-1}) \ \},\end{aligned}$$

onde o conjunto dos pontos de colocação é dado por $X_n := \{t_{n,j} := t_n + c_j h_n, j = 1, 2 : 0 \leq c_1 \leq c_2 \leq 1\}$, e $Z_N := \{t_n : n = 1, \ldots, N-1\}$ são os pontos interiores associados à rede

$$\Pi_N : 0 = t_0 < t_1 < \ldots < t_N = T, \ \ N \geq 1,$$

no intervalo $[0, T]$, e os polinómios de Lagrange $l_{1,n}$ e $l_{2,n}$ são dados por

$$l_{1,n}(t) = \frac{t - t_{2,n}}{t_{1,n} - t_{2,n}} = \frac{t - t_{2,n}}{h(c_1 - c_2)},$$

$$l_{2,n}(t) = \frac{t - t_{1,n}}{t_{2,n} - t_{1,n}} = \frac{t - t_{1,n}}{h(c_2 - c_1)}.$$

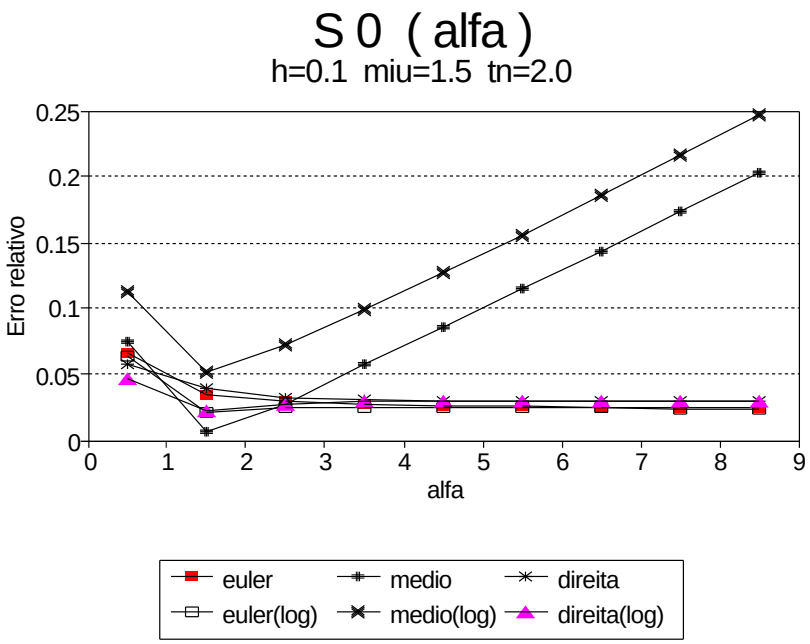

Figura 3.8: Dependência de α para os métodos de colocação em $\tilde{S}_0^{-1}(Z_N)$ e $S_0^{-1}(Z_N)$.

Para uma escolha particular dos parâmetros de colocação, $c_1 = 0$, $c_2 = 1$, geramos uma aproximação seccionalmente linear contínua, $u \in S_1^0(Z_N)$, mais precisamente, a função aproximadora pertence ao espaço linear de Hermite

$$S_1^0(Z_N) := \{u : u(t) = l_{1,n}(t)u(t_n) + l_{2,n}(t)u(t_{n+1}), u(t_n), u(t_{n+1}) \in \pi_0, t \in [t_n, t_{n+1}],$$

$$n = 0, 1, \ldots, N-1, u(T) = u(t_N)\},$$

onde

$$l_{1,n}(t) = \frac{t - t_{n+1}}{t_n - t_{n+1}} = \frac{t_{n+1} - t}{h}, \tag{3.43}$$

$$l_{2,n}(t) = \frac{t - t_n}{t_{n+1} - t_n} = \frac{t - t_n}{h}. \tag{3.44}$$

A este caso particular chama-se método dos trapézios produto em analogia com o método dos trapézios nas fórmulas de quadratura numérica.

3.3.1 Construção do Método dos Trapézios Produto

A equação de colocação para o problema proposto (3.7) é

$$u(t_n) = g_2(t_n) + \int_0^{t_n} p_2(t_n, s)u(s)ds, \quad n = 1, \ldots, N, \tag{3.45}$$

$$u(t_0) := y(0) = \frac{\mu}{\mu - 1} g_2(0), \tag{3.46}$$

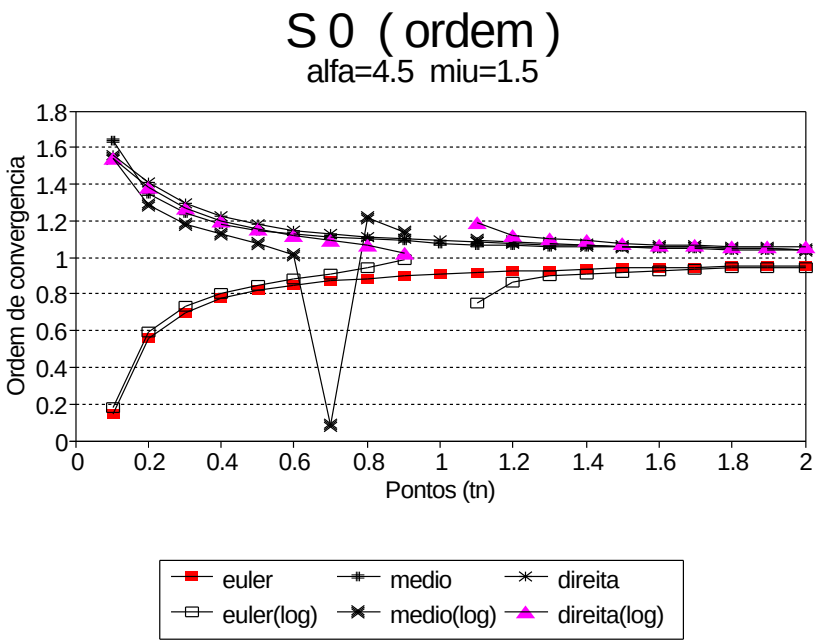

Figura 3.9: Ordens de convergência para os métodos de colocação em $\tilde{S}_0^{-1}(Z_N)$ e $S_0^{-1}(Z_N)$.

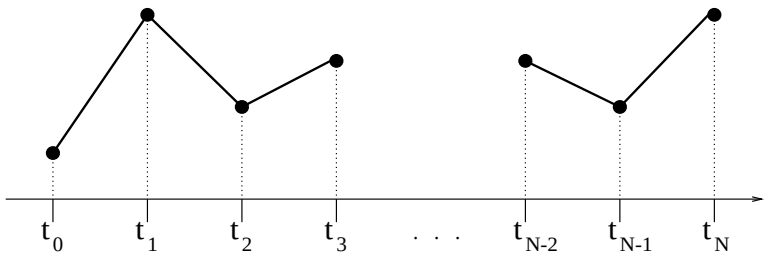

Figura 3.10: Aproximação seccionalmente linear.

onde $u \in S_1^0(Z_N)$ (ver figura 3.10) e $u(t_0)$ é definido como no método de Euler. Aproveitando a definição de $S_1^0(Z_N)$, utilizando a mudança de variável $s = t_i + \nu h$, $0 \leq \nu \leq 1$ e a propriedade aditiva dos integrais, obtemos

$$u(t_n) = g_2(t_n) + h \sum_{i=0}^{n-1} \int_0^1 p_2(t_n, t_i + \nu h) \left((1-\nu)u(t_i) + \nu u(t_{i+1})\right) d\nu,$$

$$n = 1, \ldots, N,$$

ou seja,

$$u(t_n) = g_2(t_n) + \sum_{i=0}^{n-1} \left(w_{1,i}u(t_i) + w_{2,i}u(t_{i+1})\right), \quad n = 1, \ldots, N, \tag{3.47}$$

onde os pesos $w_{1,i}$ e $w_{2,i}$ são dados respectivamente por

$$w_{1,i} := h \int_0^1 p_2(t_n, t_i + \nu h)(1-\nu)d\nu,$$

$$w_{2,i} := h \int_0^1 p_2(t_n, t_i + \nu h)\nu d\nu.$$

Uma vez que $p_2(t,s) = (s/t)^\mu/s$ os pesos $w_{1,i}$ e $w_{2,i}$ podem ser calculados exactamente e vem,

$$w_{2,i} := \frac{1}{n^\mu}\left(\frac{(i+1)^{\mu+1} - i^{\mu+1}}{\mu+1} - i\frac{(i+1)^\mu - i^\mu}{\mu}\right),$$

$$w_{1,i} := \frac{(i+1)^\mu - i^\mu}{\mu n^\mu} - w_{2,i},$$

para $i = 0, 1, \ldots, n-1$.

Observa-se que a equação (3.47) pode ser resolvida directamente para $u(t_n)$. Isto deve-se ao facto da equação integral (3.7) ser linear. Resolvendo a equação em ordem a $u(t_n)$ podemos concluir que o método dos trapézios produto, quando aplicado ao nosso problema, surge com a forma do seguinte algoritmo:

$$\begin{aligned}
u(t_0) &= \frac{\mu}{\mu-1} g_2(t_0), \\
\alpha_{n,\mu} &= (\mu^2-1)n^\mu + n^{\mu+1} - (n-1)^{\mu+1}, \\
\beta_{i,\mu} &= (i+1)^{\mu+1} - 2i^{\mu+1} + (i-1)^{\mu+1}, \\
u(t_n) &= \frac{1}{\alpha_{n,\mu}}\left(u(t_0) + \sum_{i=1}^{n-1}\beta_{i,\mu}u(t_i) + \mu(\mu+1)n^\mu g_2(t_n)\right), \quad n = 1, \ldots, N.
\end{aligned}$$

É de salientar que a exigência de continuidade faz com que se perca um grau de liberdade e consequentemente não se tenha de resolver um sistema de equações, como no caso geral.

3.3.2 Construção do Caso Geral

Neste caso pretende-se encontrar a solução $u \in \tilde{S}_1^{-1}(Z_N)$ que satisfaz o sistema:

$$u(t_{1,n}) = g_2(t_{1,n}) + \int_0^{t_{1,n}} p_2(t_{1,n}, s)u(s)ds, \tag{3.48}$$

$$u(t_{2,n}) = g_2(t_{2,n}) + \int_0^{t_{2,n}} p_2(t_{2,n}, s)u(s)ds, \tag{3.49}$$

onde $t_{j,n} := t_n + c_j h$, $j = 1, 2$, $n = 0, 1, \ldots, N$, $0 < c_1 < c_2 < 1$.

De maneira semelhante à utilizada no caso do método dos trapézios produto, o sistema (3.48) (3.49) assume a forma:

$$\begin{aligned} u(t_{1,n}) = g_2(t_{1,n}) &+ h\int_0^{c_1} p_2(t_{1,n}, t_n + \nu h)u_n(t_n + \nu h)d\nu \\ &+ h\sum_{i=0}^{n-1}\int_0^1 p_2(t_{1,n}, t_i + \nu h)u_i(t_i + \nu h)d\nu, \\ u(t_{2,n}) = g_2(t_{2,n}) &+ h\int_0^{c_2} p_2(t_{2,n}, t_n + \nu h)u_n(t_n + \nu h)d\nu \\ &+ h\sum_{i=0}^{n-1}\int_0^1 p_2(t_{2,n}, t_i + \nu h)u_i(t_i + \nu h)d\nu, \end{aligned}$$

onde

$$u_i(t_i + \nu h) = \frac{\nu - c_2}{c_1 - c_2}u(t_{1,i}) + \frac{\nu - c_1}{c_2 - c_1}u(t_{2,i}), \quad 0 \le \nu < 1,$$

com $i = 0, 1, \ldots, N-1$ e $n = 0, 1, \ldots, N$, ou seja,

$$\begin{aligned} u(t_{1,n}) = g_2(t_{1,n}) &+ h\left(u(t_{1,n})\int_0^{c_1} p_2(t_{1,n}, t_n + \nu h)\frac{\nu - c_2}{c_1 - c_2}d\nu\right. \\ &+ \left. u(t_{2,n})\int_0^{c_1} p_2(t_{1,n}, t_n + \nu h)\frac{\nu - c_1}{c_2 - c_1}d\nu\right) \\ &+ h\sum_{i=0}^{n-1}\left\{u(t_{1,i})\int_0^1 p_2(t_{1,n}, t_i + \nu h)\frac{\nu - c_2}{c_1 - c_2}d\nu\right. \\ &+ \left. u(t_{2,i})\int_0^1 p_2(t_{1,n}, t_i + \nu h)\frac{\nu - c_1}{c_2 - c_1}d\nu\right\}, \\ u(t_{2,n}) = g_2(t_{2,n}) &+ h\left(u(t_{1,n})\int_0^{c_2} p_2(t_{2,n}, t_n + \nu h)\frac{\nu - c_2}{c_1 - c_2}d\nu\right. \\ &+ \left. u(t_{2,n})\int_0^{c_2} p_2(t_{2,n}, t_n + \nu h)\frac{\nu - c_1}{c_2 - c_1}d\nu\right) \\ &+ h\sum_{i=0}^{n-1}\left\{u(t_{1,i})\int_0^1 p_2(t_{2,n}, t_i + \nu h)\frac{\nu - c_2}{c_1 - c_2}d\nu\right. \\ &+ \left. u(t_{2,i})\int_0^1 p_2(t_{2,n}, t_i + \nu h)\frac{\nu - c_1}{c_2 - c_1}d\nu\right\}. \end{aligned}$$

Resolvendo analiticamente os integrais que surgem nas equações anteriores, resulta o seguinte algoritmo, onde n varia entre 0 e N:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}\begin{bmatrix} u(t_{1,n}) \\ u(t_{2,n}) \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

onde,

$$\begin{aligned} a_{11} &= C + (n + c_1)A - B, \\ a_{12} &= B - (n + c_0)A, \\ b_1 &= Cg_2(t_{1,n}) + S, \end{aligned}$$

$$\begin{aligned} a_{21} &= (n+c_1)D - E, \\ a_{22} &= F + E - (n+c_0)D, \\ b_2 &= Fg_2(t_{2,n}) + S, \end{aligned}$$

sendo,

$$\begin{aligned} A &= \frac{(n+c_0)^{\mu} - n^{\mu}}{\mu}, \\ B &= \frac{(n+c_0)^{\mu+1} - n^{\mu+1}}{\mu+1}, \\ C &= (c_0 - c_1)(n+c_0)^{\mu}, \\ D &= \frac{(n+c_1)^{\mu} - n^{\mu}}{\mu}, \\ E &= \frac{(n+c_1)^{\mu+1} - n^{\mu+1}}{\mu+1}, \\ F &= (c_0 - c_1)(n+c_1)^{\mu}, \\ S &= \sum_{i=0}^{n-1} \left(\frac{(i+1)^{\mu+1} - i^{\mu+1}}{\mu+1} \left(u(t_{0,i}) - u(t_{1,i})\right) \right. \\ &\quad \left. + \frac{(i+1)^{\mu} - i^{\mu}}{\mu} (u(t_{1,i})(i+c_0) - u(t_{0,i})(i+c_1)) \right). \end{aligned}$$

Observe-se que neste algoritmo resolvemos um sistema de equações lineares; note-se que para calcular $u(t_{1,n})$ necessitamos de $t_{2,n}$ e não apenas pontos do intervalo $[0, t_{1,n}]$.

3.3.3 Convergência

O resultado que se segue diz-nos que o método dos trapézios produto, aplicado ao nosso problema, tem ordem de convergência global $O(h^2)$. Isto significa que a ordem de convergência é preservada quando se passa da quadratura integração produto para o método dos trapézios produto, aplicado ao nosso problema.

Teorema 3.6 *Suponhamos que g, na equação (3.7) pertence à classe $C^2(I)$ e $\mu > 1$, então a equação de colocação (3.46), (3.47) possui uma solução única $u \in S_1^0(Z_N)$. Além disso, a função erro $e(t) := y(t) - u(t)$ satisfaz a desigualdade*

$$\|e\|_{\infty} \leq C_2 h^2,$$

onde $h := t_{i+1} - t_i$, $i = 0, \ldots, N-1$, C_2 é uma constante independente de h e $\|e\|_{\infty}$ é definida como em (3.20).

Demonstração Considere-se $e_n(t) := y(t) - u_n(t)$, a função erro no intervalo $[t_n, t_{n+1})$. Uma vez que $y(t)$ verifica a equação (3.7) em todo o intervalo $[0,T]$ e $u(t_n)$ satisfaz (3.45), (3.46) podemos escrever

$$e_n(t_n) = \int_0^{t_n} p_2(t_n, s)y(s)ds - \int_0^{t_n} p_2(t_n, s)u(s)ds.$$

Integrando em cada subintervalo e depois somando vem,

$$e_n(t_n) = \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} p_2(t_n, s)\left(y(s) - l_{1,i}(s)u(t_i) - l_{2,i}u(t_{i+1})\right) ds.$$

Mas,

$$e_i(s) = y(s) - l_{1,i}(s)u(t_i) - l_{2,i}(s)u(t_{i+1}), \;\; s \in [t_i, t_{i+1}),$$

logo podemos escrever,

$$e_n(t_n) = \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} p_2(t_n, s)e_i(s)ds, \;\; n = 1, \ldots, N. \tag{3.50}$$

Uma vez que $g \in C^2(I)$ e $\mu > 1$, resulta que $y \in C^2(I)$, logo para $s \in [t, t_{i+1})$,

$$\begin{aligned} e_i(s) &= l_{1,i}(s)y(t_i) + l_{2,i}(s)y(t_{i+1}) + O(h^2) - l_{1,i}(s)u(t_i) - l_{2,i}(s)u(t_{i+1}) \\ &= l_{1,i}(s)e_i(t_i) + l_{2,i}(s)e_{i+1}(t_{i+1}) + O(h^2). \end{aligned}$$

Considerando

$$E := \max_{1 \le i \le N} |e_i(t_i)|,$$

observando-se que $l_{1,i}(s) \ge 0$ e $l_{2,i}(s) \ge 0$ e utilizando (3.43) e (3.44), obtém-se para $s \in [t_i, t_{i+1}]$,

$$|e_i(s)| \le \left(l_{1,i}(s) + l_{2,i}(s)\right) E + O(h^2) = E + O(h^2). \tag{3.51}$$

Aplicando este resultado na equação (3.50) surge a seguinte majoração:

$$|e_n(t_n)| \le \frac{1}{\mu} E + O(h^2),$$

donde,

$$E \le \frac{1}{\mu} E + O(h^2).$$

Mas $\mu > 1$ logo $E = O(h^2)$ e por conseguinte, utilizando a enequação (3.51), podemos concluir que $\|e\|_\infty = O(h^2)$, isto é, o método tem ordem 2.□

No caso geral em que a função aproximadora u é descontínua, ou seja, o método definido pelas equações (3.48) e (3.49), não foi possível obter uma demonstração teórica de convergência.

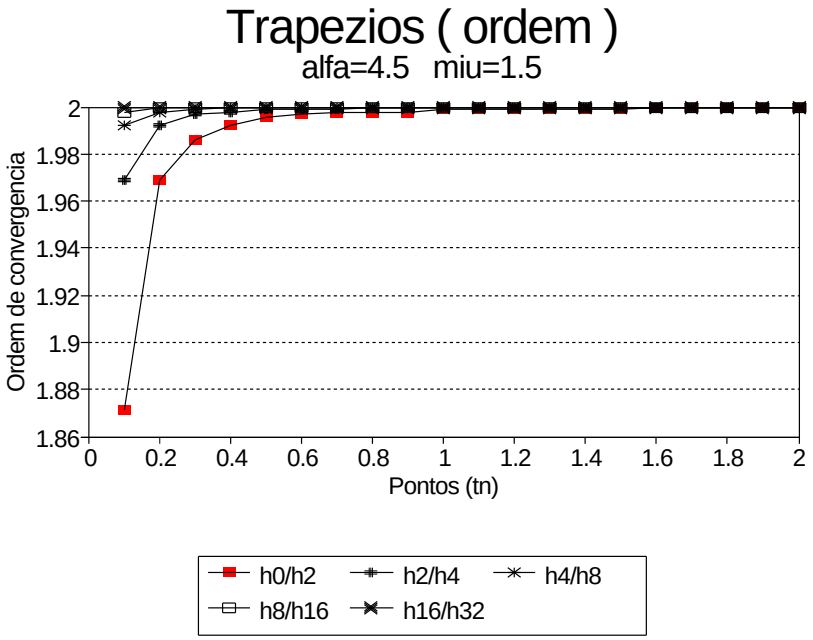

Figura 3.11: Ordem de convergência do método dos trapézios para vários valores de h.

3.3.4 Análise de Resultados em $\tilde{S}_1^{-1}$

Assim como na secção 3.2.4, também aqui o estudo é feito em relação aos exemplos típicos $y(t) = t^\alpha$ e $y(t) = t^\alpha lnt$.

Para $\alpha = 4.5$ e $\mu = 1.5$ estamos nas condições do teorema 3.6 e observa-se que de facto o método dos trapézios tem ordem 2 (figura 3.11 e tabela 3.8).

Em 1980 Brunner estudou a superconvergência local, em S_m^{-1}, quando o núcleo da equação de Volterra de segunda espécie é regular (ver teorema 2.10).

passo h	erro absoluto $E_N^h = y(2.0) - u(2.0)$	ordem de convergência $p \approx log_2(E_N^h / E_N^{h/2})$
0.1	0.024741775975	
0.05	0.006186749270	1.99969559
0.025	0.001546768893	1.99992391
0.0125	0.000386697322	1.99998098
0.00625	0.000096674649	1.99999524
0.003125	0.000024168682	1.99999881

Tabela 3.8: Ordem de convergência (método dos Trapézios).

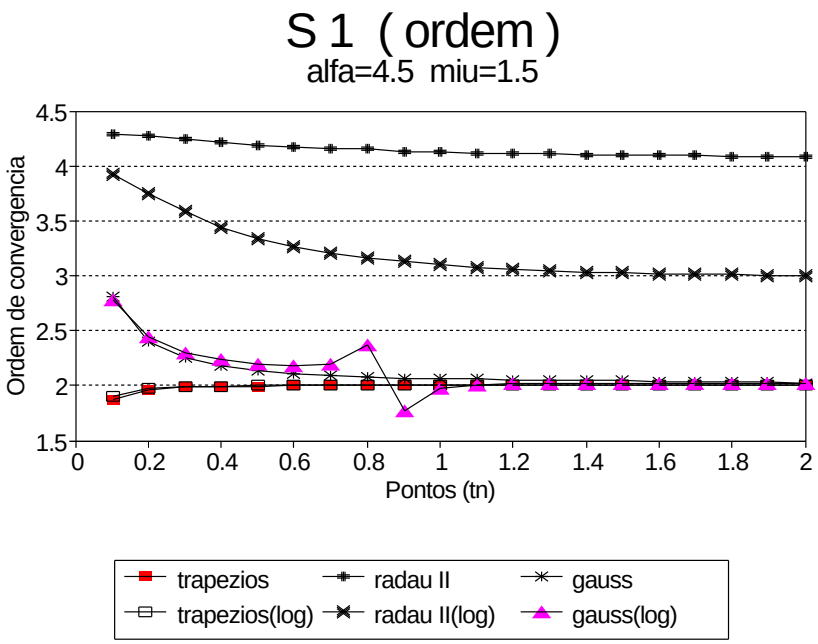

Figura 3.12: Ordem de convergência dos métodos de colocação em $\tilde{S}_1^{-1}(Z_N)$.

Assim como o método dos trapézios não perde a ordem de convergência quando se passa de um núcleo regular para o núcleo fracamente singular da nossa equação (ver teorema 3.6 e parágrafo anterior), é interessante averiguar, numericamente, se a superconvergência também se mantém com a escolha dos parâmetros feita para núcleos regulares descrita no teorema 2.10. Note-se que no teorema 2.10 exige-se que $g_2(t)$, ou seja $y(t)$, seja de classe C^3 no caso dos pontos de Radau II e classe C^4 no caso dos pontos de Gauss. Sendo assim ir-se-á considerar por exemplo $\alpha = 4.5$. Observa-se na figura 3.12 a superconvergência quando utilizamos os pontos de Radau II. É interessante notar que no caso $t^{4.5}$ consegue-se ordem 4, superior à esperada que seria 3. Em relação aos pontos de Gauss continua a não existir superconvergência.

É de salientar que o método que se apresenta ser numericamente mais robusto é o método de colocação com pontos de Radau II. Note-se que neste caso a auto-correcção é mais rápida do que nos outros métodos e a ordem de convergência é a melhor de todas (figuras 3.13 e 3.12).

método	tempo	erro relativo
Radau II	0.1952×10^{-2}	0.1100878×10^{-2}
Euler	0.9760×10^{-3}	0.2642108×10^{-1}

Tabela 3.9: Comparação dos tempos de computação face ao erro (com $h = 0.1$).

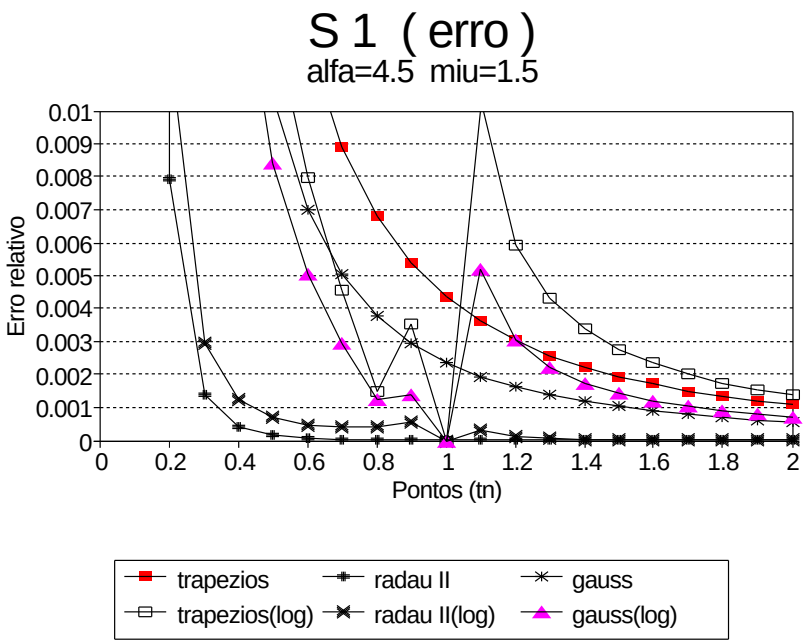

Figura 3.13: Erro relativo nos métodos de colocação em $\tilde{S}_1^{-1}(Z_N)$.

Observa-se que, para o mesmo valor do passo h, com um pouco mais de esforço computacional, do que o método de Euler, se consegue uma aproximação bastante mais precisa (tabela 3.9). Nomeadamente, obtemos uma precisão 24 vezes superior com uma duplicação do tempo computacional, ou seja, uma eficiência 12 vezes superior. Um inconveniente deste método é a exigência de regularidade de y. Uma vantagem do método de Euler é a sua pouca exigência de regularidade e ser extremamente simples, quando comparado com o método referido.

3.4 Colocação no Espaço $\tilde{S}_2^{-1}(Z_N)$

Os métodos desta secção consistem em aplicar a regra de integração produto, baseada em interpolação quadrática, ao termo integral do problema a resolver. Quando a função aproximadora, da solução exacta, é contínua então obtemos o método de Simpson.

Pretende-se encontrar um elemento $u \in \tilde{S}_2^{-1}(Z_N)$ que satisfaça o sistema de colocação

$$u(t_{1,n}) = g_2(t_{1,n}) + \int_0^{t_{1,n}} p_2(t_{1,n}, s)u(s)ds, \tag{3.52}$$

$$u(t_{2,n}) = g_2(t_{2,n}) + \int_0^{t_{2,n}} p_2(t_{2,n}, s)u(s)ds, \tag{3.53}$$

$$u(t_{3,n}) = g_2(t_{3,n}) + \int_0^{t_{3,n}} p_2(t_{3,n}, s)u(s)ds, \tag{3.54}$$

com $n = 0, 1, \ldots, N$. O espaço $\tilde{S}_2^{-1}(Z_N)$ é definido por

$$\begin{aligned}\tilde{S}_2^{-1}(Z_N) := \{ \quad & u : u(t) = u(t_{1,n})l_{1,n}(t) + u(t_{2,n})l_{2,n}(t) + u(t_{3,n})l_{3,n}(t), \\ & t \in [t_n, t_{n+1}), n = 0, 1, \ldots, N-2, \\ & u(T) = u(t_{1,N-1})l_{1,N-1}(t) + u(t_{2,N-1})l_{2,N-1}(t) + u(t_{3,N-1})l_{3,N-1}(t), \\ & t \in [t_{N-1}, t_N], u(t_{1,n}), u(t_{2,n}), u(t_{3,n}) \in \pi_0, n = 0, 1, \ldots, N \ \},\end{aligned}$$

onde os polinómios de Lagrange são,

$$l_{1,n}(t) = \frac{(t - t_{2,n})(t - t_{3,n})}{(t_{1,n} - t_{2,n})(t_{1,n} - t_{3,n})} = \frac{(t - t_{2,n})(t - t_{3,n})}{h^2(c_1 - c_2)(c_1 - c_3)},$$

$$l_{2,n}(t) = \frac{(t - t_{1,n})(t - t_{3,n})}{(t_{2,n} - t_{1,n})(t_{2,n} - t_{3,n})} = \frac{(t - t_{1,n})(t - t_{3,n})}{h^2(c_2 - c_1)(c_2 - c_3)},$$

$$l_{3,n}(t) = \frac{(t - t_{1,n})(t - t_{2,n})}{(t_{3,n} - t_{1,n})(t_{3,n} - t_{2,n})} = \frac{(t - t_{1,n})(t - t_{2,n})}{h^2(c_3 - c_1)(c_3 - c_2)},$$

e o conjunto dos pontos de colocação é dado por $X_n := \{t_{j,n} := t_n + c_j h_n, j = 1, 2, 3 : 0 \leq c_1 \leq c_2 \leq c_3 \leq 1\}$. A rede considerada é a mesma que nos métodos anteriores.

Se $c_1 = 0$, $c_2 = 1/2$ e $c_3 = 1$ então a aproximação, da solução da equação integral, vai ser seccionalmente quadrática e contínua, $u \in S_2^0(Z_N)$, ou seja, ela pertence ao espaço:

$$\begin{aligned}S_2^0(Z_N) := \{ \quad & u : u(t) = u(t_n)l_{1,n}(t) + u(t_{n+1/2})l_{2,n}(t) + u(t_{n+1})l_{3,n}(t), \\ & t \in [t_n, t_{n+1}), n = 0, 1, \ldots, N-2, \\ & u(T) = u(t_{N-1})l_{1,N-1}(t) + u(t_{N-1+1/2})l_{2,N-1}(t) + u(t_N)l_{3,N-1}(t), \\ & t \in [t_{N-1}, t_N], u(t_n), u(t_{n+1/2}) \in \pi_0, \forall n \ \},\end{aligned}$$

onde $u(t_{n+\frac{1}{2}}) := u(t_n + \frac{h}{2})$ e os polinómios de Lagrange são dados por:

$$l_{1,n}(t) = \frac{(t - t_{n+1/2})(t - t_{n+1})}{(t_n - t_{n+1/2})(t_n - t_{n+1})} = 2\frac{(t - t_{2,n})(t - t_{3,n})}{h^2},$$

$$l_{2,n}(t) = \frac{(t - t_n)(t - t_{n+1})}{(t_{n+1/2} - t_n)(t_{n+1/2} - t_{n+1})} = -4\frac{(t - t_n)(t - t_{n+1})}{h^2},$$

$$l_{3,n}(t) = \frac{(t - t_n)(t - t_{n+1/2})}{(t_{n+1} - t_n)(t_{n+1} - t_{n+1/2})} = 2\frac{(t - t_{1,n})(t - t_{2,n})}{h^2}.$$

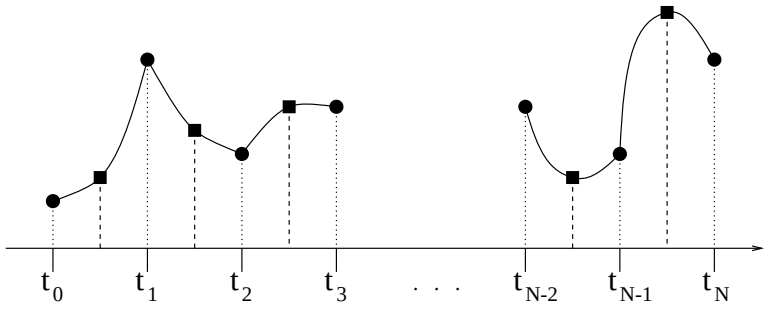

Figura 3.14: Aproximação seccionalmente quadrática.

Quando se utiliza o espaço anterior como espaço de aproximação na integração produto, aplicada às equações integrais, surge o método de Simpson Produto em analogia com o método de Simpson nas fórmulas de quadratura numérica (ver figura 3.14).

Em seguida iremos construir o métodos de colocação em $\tilde{S}_2^{-1}(Z_N)$. O estudo teórico da convergência não foi investigado para estes métodos.

3.4.1 Construção do Método de Simpson Produto

Considere-se o sistema de colocação (3.52), (3.53), (3.54), no espaço $S_2^0(Z_N)$. Neste caso a solução aproximadora é contínua e por isso perde-se um grau de liberdade. Por conseguinte, o sistema de colocação reduz-se ao seguinte sistema de duas equações:

$$u(t_{n+1/2}) = g_2(t_{n+1/2}) + \int_0^{t_{n+1/2}} p_2(t_{n+1/2}, s)u(s)ds, \tag{3.55}$$

$$u(t_{n+1}) = g_2(t_{n+1}) + \int_0^{t_{n+1}} p_2(t_{n+1}, s)u(s)ds, \tag{3.56}$$

onde se tem a condição inicial $u(t_0) = \frac{\mu}{\mu^{-1}} g_2(0)$ (ver secção 3.2.1.1) e

$$t_{n+1/2} = t_n + \frac{h}{2} = (n + 0.5)h, \;\; t_{n+1} = (n + 1)h, \;\; n = 0, 1, \ldots, N - 1.$$

Aplicando a propriedade aditiva dos integrais, e uma vez que $u \in S_2^0(Z_N)$, as equações (3.55) e (3.56) tomam a forma:

$$u(t_{n+1/2}) = g_2(t_{n+1/2}) + \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} p_2(t_{n+1/2}, s) u_i(s) ds + \int_{t_n}^{t_{n+1/2}} p_2(t_{n+1/2}, s) u_n(s) ds,$$

$$u(t_{n+1}) = g_2(t_{n+1}) + \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} p_2(t_{n+1}, s) u_i(s) ds + \int_{t_n}^{t_{n+1}} p_2(t_{n+1}, s) u_n(s) ds,$$

onde $n = 0, 1, \ldots, N-1$. Utilizando agora a mudança de variável $s = t_i + \nu h$, as equações anteriores, resultam em,

$$\begin{aligned} u(t_{n+1/2}) = g_2(t_{n+1/2}) &+ h \sum_{i=0}^{n-1} \int_0^1 p_2(t_{n+1/2}, t_i + \nu h) u_i(t_i + \nu h) d\nu \\ &+ h \int_0^{0.5} p_2(t_{n+1/2}, t_n + \nu h) u_n(t_n + \nu h) d\nu, \\ u(t_{n+1}) = g_2(t_{n+1}) &+ h \sum_{i=0}^{n-1} \int_0^1 p_2(t_{n+1}, t_i + \nu h) u_i(t_i + \nu h) d\nu \\ &+ h \int_0^1 p_2(t_{n+1}, t_n + \nu h) u_n(t_n + \nu h) d\nu, \end{aligned}$$

onde, para $i = 0, \ldots, N-1$,

$$u_i(t_i + \nu h) = u(t_i) L_{1,i}(\nu) + u(t_{i+1/2}) L_{2,i}(\nu) + u(t_{i+1}) L_{3,i}(\nu),$$

com,

$$\begin{aligned} L_{1,i}(\nu) &= 2(\nu - 0.5)(\nu - 1), \\ L_{2,i}(\nu) &= -4\nu(\nu - 1), \\ L_{3,i}(\nu) &= 2\nu(\nu - 0.5). \end{aligned}$$

Tendo em conta que as incógnitas do sistema são $u(t_{n+1/2})$ e $u(t_{n+1})$, resulta que se pretende resolver:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} u(t_{n+1/2}) \\ u(t_{n+1}) \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix},$$

onde,

$$\begin{aligned} a_{11} &= 1 - h \int_0^{0.5} p_2(t_{n+1/2}, t_n + \nu h) L_{2,n}(\nu) d\nu, \\ a_{12} &= -h \int_0^{0.5} p_2(t_{n+1/2}, t_n + \nu h) L_{3,n}(\nu) d\nu, \\ a_{21} &= -h \int_0^1 p_2(t_{n+1}, t_n + \nu h) L_{2,n}(\nu) d\nu, \\ a_{22} &= 1 - h \int_0^1 p_2(t_{n+1}, t_n + \nu h) L_{3,n}(\nu) d\nu, \\ b_1 &= g_2(t_{n+1/2}) + h \int_0^{0.5} p_2(t_{n+1/2}, t_n + \nu h) L_{1,n}(\nu) d\nu u(t_n) + \overline{S}(t_{n+1/2}), \end{aligned}$$

$$
\begin{aligned}
b_2 &= g_2(t_{n+1}) + h\int_0^1 p_2(t_{n+1}, t_n + \nu h) L_{1,n}(\nu) d\nu u(t_n) + \overline{S}(t_{n+1}), \\
\overline{S}(t) &= h\sum_{i=0}^{n-1} \Big(u(t_i) \int_0^1 p_2(t, t_i + \nu h) L_{1,i}(\nu) d\nu \\
&\quad + u(t_{i+1/2}) \int_0^1 p_2(t, t_i + \nu h) L_{2,i}(\nu) d\nu \\
&\quad + u(t_{i+1}) \int_0^1 p_2(t, t_i + \nu h) L_{3,i}(\nu) d\nu \Big).
\end{aligned}
$$

Resolvendo analiticamente os integrais das expressões anteriores, surge o seguinte algoritmo, para $n = 0, 1, \ldots, N-1$:

$$
\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} u(t_{n+1/2}) \\ u(t_{n+1}) \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix},
$$

onde,

$$
\begin{aligned}
a_{11} &= 1 + \frac{4}{(n+0.5)^\mu} I_2(n, 0.5), \\
a_{12} &= -\frac{2}{(n+0.5)^\mu} I_3(n, 0.5), \\
a_{21} &= \frac{4}{(n+1)^\mu} I_2(n, 1), \\
a_{22} &= 1 - \frac{2}{(n+1)^\mu} I_3(n, 1), \\
b_1 &= g_2(t_{n+1/2}) + \frac{1}{(n+0.5)^\mu} (2u(t_n) I_1(n, 0.5) + S), \\
b_2 &= g_2(t_{n+1}) + \frac{1}{(n+1)^\mu} (2u(t_n) I_1(n, 1) + S), \\
S &= 2\sum_{i=0}^{n-1} (u(t_i) I_1(i, 1) - 2u(t_{i+1/2}) I_2(i, 1) + u(t_{i+1}) I_3(i, 1)), \\
I_1(n, c) &= A(n, c) - (2n + 1.5) B(n, c) + (n + 0.5)(n + 1) C(n, c), \\
I_2(n, c) &= A(n, c) - (2n + 1) B(n, c) + n(n + 1) C(n, c), \\
I_3(n, c) &= A(n, c) - (2n + 0.5) B(n, c) + n(n + 0.5) C(n, c), \\
A(n, c) &= \frac{(n+c)^{\mu+2} - n^{\mu+2}}{\mu + 2}, \\
B(n, c) &= \frac{(n+c)^{\mu+1} - n^{\mu+1}}{\mu + 1}, \\
C(n, c) &= \frac{(n+c)^{\mu} - n^{\mu}}{\mu}.
\end{aligned}
$$

3.4.2 Construção do Caso Geral

O caso geral é quando a função aproximadora pode ser descontínua. Isto corresponde a considerar o sistema de colocação (3.52), (3.53), (3.54), no espaço

$\tilde{S}_2^{-1}(Z_N)$ com os parâmetros de colocação $0 < c_1 < c_2 < c_3 < 1$. Nesta situação ter-se-á que resolver, em cada passo do algoritmo, um sistema de equações de dimensão 3.

De forma análoga à utilizada na construção do método de Simpson Produto, ou seja, aplicando a propriedade aditiva dos integrais e utilizando, em seguida, a mudança de variável $s = t_i + \nu h$, o sistema de colocação (3.52), (3.53), (3.54), assume a forma:

$$\begin{aligned}
u(t_{1,n}) = g(t_{1,n}) &+ h\sum_{i=0}^{n-1}\int_0^1 p_2(t_{1,n}, t_i+\nu h)u_i(t_i+\nu h)d\nu \\
&+ h\int_0^{c_1} p_2(t_{1,n}, t_n+\nu h)u_n(t_n+\nu h)d\nu, \\
u(t_{2,n}) = g(t_{2,n}) &+ h\sum_{i=0}^{n-1}\int_0^1 p_2(t_{2,n}, t_i+\nu h)u_i(t_i+\nu h)d\nu \\
&+ h\int_0^{c_2} p_2(t_{2,n}, t_n+\nu h)u_n(t_n+\nu h)d\nu, \\
u(t_{3,n}) = g(t_{3,n}) &+ h\sum_{i=0}^{n-1}\int_0^1 p_2(t_{3,n}, t_i+\nu h)u_i(t_i+\nu h)d\nu \\
&+ h\int_0^{c_3} p_2(t_{3,n}, t_n+\nu h)u_n(t_n+\nu h)d\nu,
\end{aligned}$$

onde $n = 0, 1, \ldots, N-1$ e, para $i = 0, 1, \ldots, N-1$, tem-se

$$u_i(t_i+\nu h) = u(t_{1,i})L_{1,i}(\nu) + u(t_{2,i})L_{2,i}(\nu) + u(t_{3,i})L_{3,i}(\nu),$$

com,

$$\begin{aligned}
L_{1,i}(\nu) &= \frac{(\nu-c_2)(\nu-c_3)}{(c_1-c_2)(c_1-c_3)}, \\
L_{2,i}(\nu) &= \frac{(\nu-c_1)(\nu-c_3)}{(c_2-c_1)(c_2-c_3)}, \\
L_{3,i}(\nu) &= \frac{(\nu-c_1)(\nu-c_2)}{(c_3-c_1)(c_3-c_2)}.
\end{aligned}$$

Uma vez que neste caso as incógnitas são $u(t_{1,n})$, $u(t_{2,n})$ e $u(t_{3,n})$, resulta que se pretende resolver o sistema:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} u(t_{1,n}) \\ u(t_{2,n}) \\ u(t_{3,n}) \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix},$$

onde,

$$\begin{aligned}
a_{11} &= 1 - h\int_0^{c_1} p_2(t_{1,n}, t_n+\nu h)L_{1,n}(\nu)d\nu, \\
a_{12} &= -h\int_0^{c_1} p_2(t_{1,n}, t_n+\nu h)L_{2,n}(\nu)d\nu,
\end{aligned}$$

$$
\begin{aligned}
a_{13} &= -h\int_0^{c_1} p_2(t_{1,n}, t_n+\nu h)L_{3,n}(\nu)d\nu,\\
a_{21} &= -h\int_0^{c_2} p_2(t_{2,n}, t_n+\nu h)L_{1,n}(\nu)d\nu,\\
a_{22} &= 1-h\int_0^{c_2} p_2(t_{2,n}, t_n+\nu h)L_{2,n}(\nu)d\nu,\\
a_{23} &= -h\int_0^{c_2} p_2(t_{2,n}, t_n+\nu h)L_{3,n}(\nu)d\nu,\\
a_{31} &= -h\int_0^{c_3} p_2(t_{3,n}, t_n+\nu h)L_{1,n}(\nu)d\nu,\\
a_{32} &= -h\int_0^{c_3} p_2(t_{3,n}, t_n+\nu h)L_{2,n}(\nu)d\nu,\\
a_{33} &= 1-h\int_0^{c_3} p_2(t_{3,n}, t_n+\nu h)L_{3,n}(\nu)d\nu,\\
b_1 &= g_2(t_{1,n})+\overline{S}(t_{1,n}),\\
b_2 &= g_2(t_{2,n})+\overline{S}(t_{2,n}),\\
b_3 &= g_2(t_{3,n})+\overline{S}(t_{3,n}),\\
\overline{S}(t) &= h\sum_{i=0}^{n-1}\Big(u(t_{1,i})\int_0^1 p_2(t,t_i+\nu h)L_{1,i}(\nu)d\nu\\
&\quad +u(t_{2,i})\int_0^1 p_2(t,t_i+\nu h)L_{2,i}(\nu)d\nu\\
&\quad +u(t_{3,i})\int_0^1 p_2(t,t_i+\nu h)L_{3,i}(\nu)d\nu\Big).
\end{aligned}
$$

Resolvendo analiticamente os integrais anteriores, obtém-se o seguinte algoritmo, para $n = 0, 1, \ldots, N-1$:

$$
\begin{bmatrix} a_{11} & a_{12} & a_{13}\\ a_{21} & a_{22} & a_{23}\\ a_{31} & a_{32} & a_{33}\end{bmatrix}\begin{bmatrix} u(t_{1,n})\\ u(t_{2,n})\\ u(t_{3,n})\end{bmatrix} = \begin{bmatrix} b_1\\ b_2\\ b_3\end{bmatrix},
$$

onde,

$$
\begin{aligned}
a_{11} &= 1-\frac{1}{(n+c_1)^\mu}I_1(n,c_1),\\
a_{12} &= -\frac{1}{(n+c_1)^\mu}I_2(n,c_1),\\
a_{13} &= -\frac{1}{(n+c_1)^\mu}I_3(n,c_1),\\
a_{21} &= -\frac{1}{(n+c_2)^\mu}I_1(n,c_2),\\
a_{22} &= 1-\frac{1}{(n+c_2)^\mu}I_2(n,c_2),\\
a_{23} &= -\frac{1}{(n+c_2)^\mu}I_3(n,c_2),\\
a_{31} &= -\frac{1}{(n+c_3)^\mu}I_1(n,c_3),
\end{aligned}
$$

$$
\begin{aligned}
a_{32} &= -\frac{1}{(n+c_3)^{\mu}} I_2(n, c_3), \\
a_{33} &= 1 - \frac{1}{(n+c_3)^{\mu}} I_3(n, c_3), \\
b_1 &= g_2(t_{1,n}) + \frac{S}{(n+c_1)^{\mu}}, \\
b_2 &= g_2(t_{2,n}) + \frac{S}{(n+c_2)^{\mu}}, \\
b_3 &= g_2(t_{3,n}) + \frac{S}{(n+c_3)^{\mu}}, \\
S &= \sum_{i=0}^{n-1} (u(t_{1,i}) I_1(i,1) + u(t_{2,i}) I_2(i,1) + u(t_{3,i}) I_3(i,1)), \\
I_1(n,c) &= \frac{1}{(c_1-c_2)(c_1-c_3)} (A(n,c) - (2n + c_2 + c_3) B(n,c) \\
&\quad + (n+c_2)(n+c_3) C(n,c)), \\
I_2(n,c) &= \frac{1}{(c_2-c_1)(c_2-c_3)} (A(n,c) - (2n + c_1 + c_3) B(n,c) \\
&\quad + (n+c_1)(n+c_3) C(n,c)), \\
I_3(n,c) &= \frac{1}{(c_3-c_1)(c_3-c_2)} (A(n,c) - (2n + c_1 + c_2) B(n,c) \\
&\quad + (n+c_1)(n+c_2) C(n,c)), \\
A(n,c) &= \frac{(n+c)^{\mu+2} - n^{\mu+2}}{\mu+2}, \\
B(n,c) &= \frac{(n+c)^{\mu+1} - n^{\mu+1}}{\mu+1}, \\
C(n,c) &= \frac{(n+c)^{\mu} - n^{\mu}}{\mu}.
\end{aligned}
$$

Note-se que, por exemplo, para calcular $u(t_{1,n})$ utilizam-se os pontos $t_{2,n}$ e $t_{3,n}$ que estão fora do intervalo $[0, t_{1,n}]$ e além disso em cada passo do algoritmo resolvemos um sistema de equações lineares.

3.4.3 Análise de Resultados em $\tilde{S}_2^{-1}$

Tal como nas secções 3.2.4 e 3.3.4 iremos considerar, os exemplos típicos $y(t) = t^{\alpha}$ e $y(t) = t^{\alpha} lnt$.

As ordens de convergência dos métodos de colocação em S_0^{-1} e S_1^{-1}, dadas pelo teorema 2.9, para núcleos regulares permanecem válidas no caso do núcleo sin-

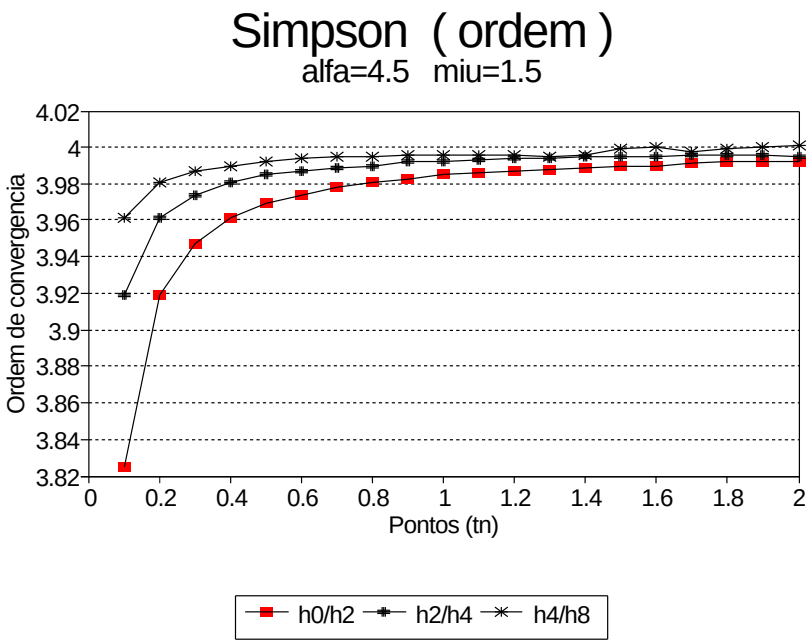

Figura 3.15: Ordem de convergência do método de Simpson.

gular estudado neste trabalho, nos exemplos estudados.[1] Isto leva-nos a suspeitar que, quando se faz a colocação em espaços de splines polinomiais de ordens mais elevadas, o mesmo se verifique. Em particular é de esperar que o mesmo comportamento se verifique quando se passa ao método de Simpson Produto, quando aplicado à nossa equação. No teorema 2.9 garante-se a convergência $O(h^3)$ para núcleos regulares no caso do método de Simpson, desde que $y \in C^3$. Considerou-se $\alpha = 4.5$, para se garantir $y \in C^3$, e obteve-se ordem aproximadamente 4 — uma ordem mais elevada que a prevista (figura 3.15); Os resultados numéricos indicam que a convergência é mais rápida do que $O(h^3)$. O teorema 2.9 garante $O(h^3)$, para núcleos regulares, mas esta ordem é apenas um limite inferior. Além disso, se $p_2(t,s)$ fosse constante, então o método de Simpson Produto reduzir-se-ía ao método de Simpson usual cujo erro se sabe ser $O(h^4)$. Note-se que $p_2(t,s)$ varia muito, perto do ponto $t = 0.0$, e pouco longe deste ponto. Sendo assim não é de estranhar que o valor aproximado da ordem de convergência, perto da singularidade, seja ligeiramente diferente de 4 mas este valor é recuperado rapidamente quando nos afastamos do ponto $t = 0.0$. É de suspeitar que a rapidez da recuperação tem a ver directamente com a ordem do método; note-se que em métodos de ordem baixa a recuperação parece ser mais lenta do que em métodos de ordem mais elevada. (ver tabela 3.10 e

[1] Apesar de o núcleo (3.8), da nossa equação, não verificar a condição (vii) do teorema 1.2 (contrariamente ao núcleo fracamente singular, do tipo Abel, dado por (1.5)), a observação 2.5 da secção 2.2.2 parece ser aplicável.

figura 3.15)

Observe-se que a ordem deixa de funcionar para valores de h muito pequenos, abaixo de 0.00625, os erros de arredondamento começam a ter uma influência considerável (ver tabela 3.10). Acontece que para h's muito pequenos o erro aumenta em vez de diminuir, pelo que se aconselha o uso deste método para valores de h não muito pequenos (por exemplo $h = 0.1$, $h = 0.05$ ou $h = 0.025$), necessidade de precisão razoável e funções g suficientemente regulares, C^3.

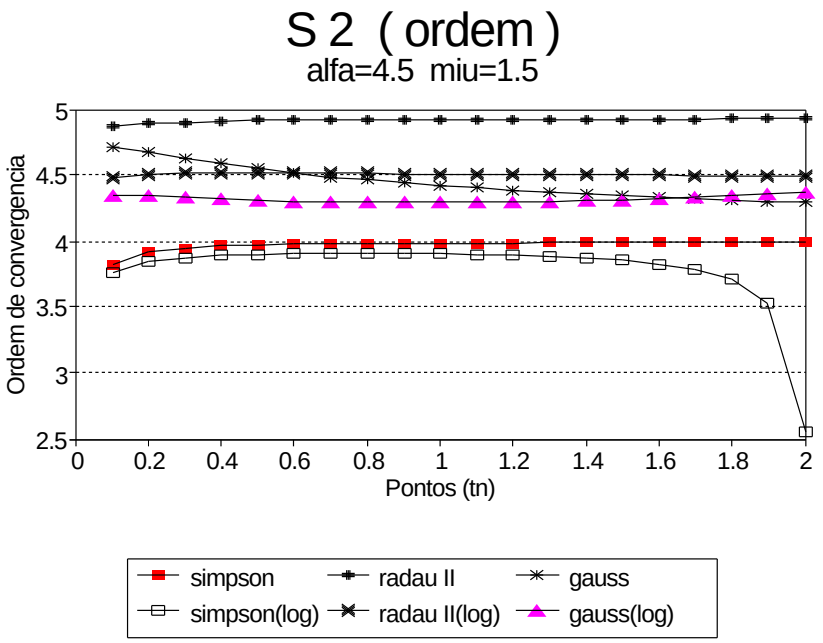

Figura 3.16: Ordem de convergência para os métodos de colocação em $\tilde{S}_2^{-1}(Z_N)$.

passo h	erro absoluto $E_N^h = y(2.0) - u(2.0)$	ordem de convergência $p \approx log_2(E_N^h / E_N^{h/2})$
0.1	0.000002869803	
0.05	0.000000180314	3.99236491
0.025	0.000000011305	3.99549241
0.0125	0.000000000706	4.00084354
0.00625	0.000000000044	4.00542496
0.003125	0.000000000015	1.54472858

Tabela 3.10: Ordem de convergência (método de Simpson).

Em relação à superconvergência local, em $\tilde{S}_2^{-1}$, consegue-se, numericamente, a melhor ordem de convergência, 5, quando se utilizam os pontos de Radau II

(figura 3.16). Isto verifica-se mesmo quando não se exige a regularidade do teorema 2.10, $g_2 \in C^5$; observe-se que os resultados foram obtidos para $\alpha = 4.5$.

3.5 Conclusões

Nesta secção pretende-se analisar em conjunto os métodos de colocação utilizados neste capítulo e ver em que medida o aumento do grau das splines polinomiais reduz o erro obtido para um mesmo passo e aumenta a ordem de convergência. Este estudo tem como objectivo determinar qual a melhor precisão com um tempo de computação limitado, caso se disponha de uma máquina com pouca capacidade de cálculo ou se deseje efectuar um número de cálculos muito elevado. Pertende-se igualmente determinar a melhor aproximação para obter uma precisão desejada com o mínimo tempo de computação.

O primeiro factor a ter em conta na escolha do método a utilizar diz respeito à ordem de convergência. Quanto maior for esta ordem, maior é a redução do erro com a redução a metade do passo utilizado.

Uma primeira observação dos resultados obtidos leva a concluir que quanto maior for o grau das splines polinomiais utilizadas maior é a ordem do método, pelo que poderia parecer conveniente escolher splines polinomiais do maior grau possível. No entanto, o aumento do grau das splines polinomiais obriga, como já vimos, a uma maior exigência na regularidade das funções a aproximar, o que limita a aplicabilidade dos métodos nas situações reais. Por outro lado, quanto maior for o grau das splines polinomiais maior é a complexidade do algoritmo a utilizar, o que conduz a maiores tempos na dedução dos métodos e maior probabilidade de cometer erros quer na sua dedução quer na sua implementação em computador.

As figuras 3.18 e 3.19 permitem determinar, para cada método, qual o erro relativo para um dado tempo de computação. O tempo de computação é função do passo, dado que ao reduzirmos o passo a metade estamos, aproximadamente, a duplicar o tempo de cálculo; pois os métodos são constituídos por um ciclo que é executado tantas vezes quantas o número de pontos da rede. Por esta razão encontramos em abcissas o logaritmo base 2 do tempo. Por outro lado, as ordenadas medem o erro relativo do método[2] pelo que a inclinação dos segmentos de recta são uma medida da ordem de convergência.

A figura 3.18 representa o desempenho dos métodos de colocação nos espaços $\tilde{S}_0^{-1}(Z_N)$, $S_0^{-1}(Z_N)$ e $\tilde{S}_1^{-1}(Z_N)$. Estes métodos são bem comportados, pois

[2]número de algarismos significativos a menos de um factor constante.

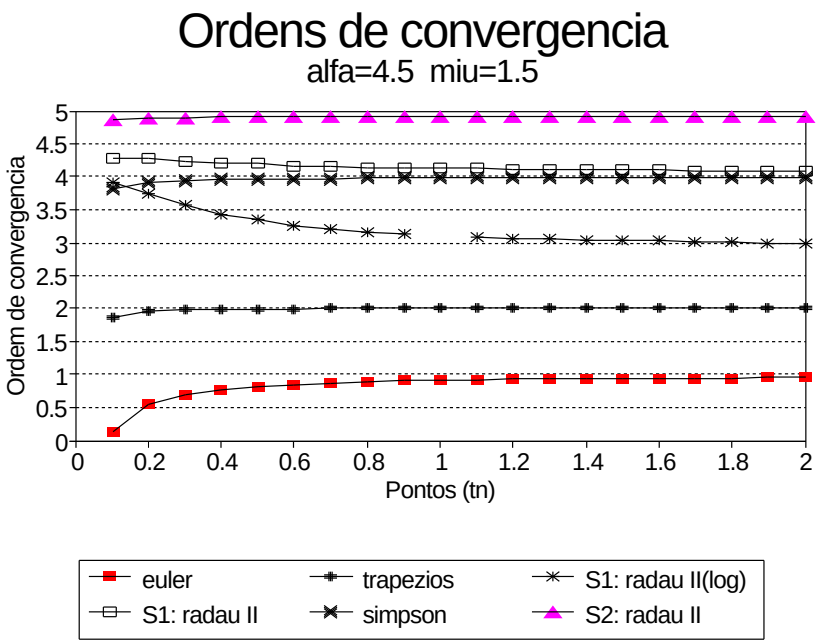

Figura 3.17: Ordens de convergência para vários métodos de colocação em $\tilde{S}_r^{-1}(Z_N)$.

obtemos rectas quase perfeitas, pelo que é lícito supor que obteremos precisões cada vez mais elevadas sempre que diminuirmos o passo (até atingirmos a unidade de arredondamento do tipo de dados em vírgula flutuante utilizado).

A figura 3.19 representa também o desempenho dos métodos de colocação em $\tilde{S}_2^{-1}(Z_N)$. Estes métodos apresentam um comportamento regular apenas para valores de h relativamente grandes. Para valores de h pequenos estes métodos começam a ser penalizados por erros de arredondamento elevados devido à necessidade de resolver, em cada passo, sistemas matriciais. Por oposição, os métodos mais simples, como o de Euler, permite aproveitar a capacidade de representação da máquina devido à simplicidade dos cálculos efectuados.

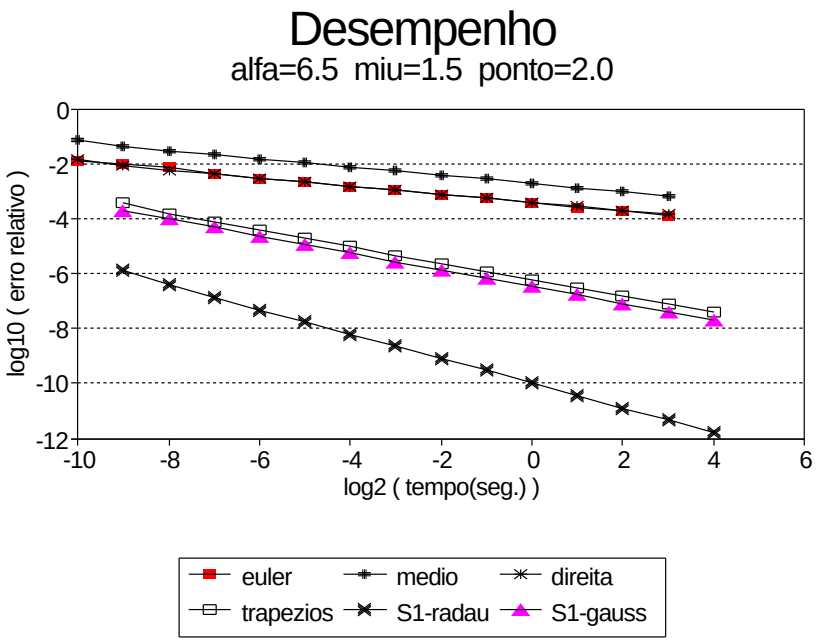

Figura 3.18: Desempenho dos métodos de colocação em $\tilde{S}_0^{-1}(Z_N)$, $S_0^{-1}(Z_N)$ e $\tilde{S}_1^{-1}(Z_N)$.

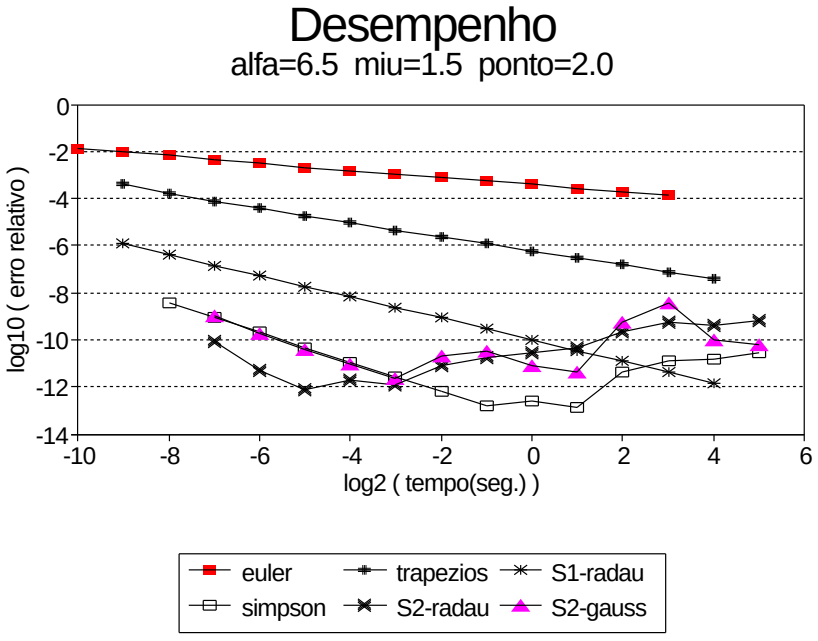

Figura 3.19: Desempenho de vários métodos de colocação em $\tilde{S}_r^{-1}(Z_N)$.

Capítulo 4

Aceleração de Convergência

A análise numérica, quando utilizada quer em engenharia quer na matemática aplicada, trabalha frequentemente com sucessões e séries. Estas são produzidas por métodos iterativos, técnicas de perturbação, procedimentos de aproximação dependentes de um parâmetro, *etc.* Muito frequentemente, na prática, essas sucessões ou séries convergem tão lentamente que tornam o seu uso desinteressante. É por esta razão que os métodos de aceleração de convergência têm sido estudados com grande interesse desde há vários anos em diversos domínios de aplicação (ver [3] e [2]). Estes métodos baseiam-se na ideia de extrapolação e conduzem, em muitos casos, à solução de problemas que seriam de outra forma insolúveis. Os métodos de extrapolação constituem, presentemente, uma área particular na análise numérica tendo ligações com muitas outras áreas importantes, tais como fracções contínuas (ver [4]) ou métodos de projecção entre outros. Eles também constituem a base de novos métodos para a solução de diversos problemas. Os métodos analíticos têm-se tornado cada vez mais importantes na matemática aplicada e na análise numérica, sendo de esperar que os métodos de extrapolação venham a ter um uso generalizado.

4.1 Introdução

Apesar da crescente eficiência dos computadores e dos algoritmos numéricos utilizados, as exigências de precisão impostas por muitas aplicações práticas são tais que o desenvolvimento actual ainda não permite alcançar. Desta forma, procurou-se obter algoritmos simples e gerais para ultrapassar este problema.

Um grande número de métodos utilizados em análise numérica e em matemática aplicada são métodos iterativos. Infelizmente, na prática, verifica-se que estes métodos convergem lentamente o que torna desaconselhável o seu uso. Por esta razão é frequente recorrer a métodos de aceleração de convergência.

Em muitos problemas de análise numérica a precisão de uma solução numérica é proporcional a uma potência do parâmetro da rede h. Os métodos menos dispendiosos são aqueles em que a precisão é proporcional à primeira ou segunda potências de h.

No início do século, Richardson formulou um método completamente novo para aumentar a precisão das soluções numéricas dos problemas lineares. Ele introduziu um meio que permite construir algoritmos usando soluções numéricas com vários parâmetros de aproximação distintos. Combinando de certa forma estas soluções ele conseguiu obter maior precisão. Esta é a ideia base para a construção de algoritmos numéricos para diversos problemas ligados às equações diferenciais e, posteriormente, generalizada aos problemas de equações integrais.

Quando se desenvolvem algoritmos simples e económicos, o comportamento da solução é de grande importância, sendo a classe de funções a que a solução pertence crítica para a aproximação. Na maioria das situações trabalhamos com um conjunto de dados que permite determinar o espaço funcional da solução. O uso desta informação inicial é essencial para a procura da classe de funções à qual a solução pertence e, consequentemente, à escolha do método de aproximação. Suponhamos que a solução de uma certa equação integral é obtida usando um método numérico de primeira ordem de precisão em relação ao diâmetro da rede h. Se este método for usado com os parâmetros h e $h/2$, então, de uma forma aproximada podemos afirmar que a segunda solução é duas vezes mais precisa que a primeira. Se o parâmetro escolhido fôr $h/3$, a solução será três vezes mais precisa que a primeira, e assim sucessivamente. Este processo de refinamento leva-nos a concluir que a precisão do resultado é proporcional ao diâmetro da rede. Assim, se necessitamos de um resultado 100 vezes mais preciso que a solução obtida com o parâmetro h, então deveremos utilizar um diâmetro 100 vezes inferior. O cálculo desta aproximação pode exigir um esforço de cálculo tão grande que torne inviável a aplicação do método.

Isto é especialmente verdade se estivermos a trabalhar com problemas a duas ou três dimensões, onde a dimensão dos vectores será inversamente proporcional ao quadrado e ao cubo de h.

Suponhamos que a solução exacta do problema a resolver é suficientemente regular. Admitamos que isso nos permite concluir que a solução aproximada tem um desenvolvimento em série de Taylor, em potências de h. Então, usando esse desenvolvimento em série, pode-se provar que uma certa combinação linear das três soluções aproximadas h, $h/2$, e $h/3$ provoca um aumento considerável na precisão. Neste caso, a ordem de precisão da combinação linear é $O(h^3)$. Se o parâmetro adimensional h for, por exemplo, $\frac{1}{10}$, a precisão da solução é da ordem de 10^{-3}. Note-se que sem se recorrer a tal procedimento seria necessário resolver o sistema aproximado com um parâmetro 10^{-3}. No caso unidimensional isto conduz-nos a uma diferença no número de nós proporcional a 10^3, enquanto com o procedimento indicado utilizaram-se apenas três soluções com um número de nós proporcional a 10, 20 e 30, respectivamente. No caso de problemas bidimensionais ou tridimensionais a comparação é ainda mais gritante. Podemos assumir que quanto maior for a regularidade das soluções exacta e aproximada maior será o grau de precisão a que poderemos chegar.

4.2 Aceleração de Convergência

Seja (S_n) uma sucessão de números (reais ou complexos) que converge para S. Iremos transformar a sucessão (S_n) noutra sucessão (T_n) e denotar por T essa transformação. Suponhamos que a nova sucessão (T_n) satisfaz, pelo menos para algumas classes particulares de sucessões convergentes (S_n), as seguintes propriedades (ver [3] pp. 1-2):

1. (T_n) converge para o mesmo limite que (S_n), ou seja, a transformação T é *regular* para a sucessão (S_n).

2. (T_n) converge para S mais depressa que (S_n), isto é, $\lim_{n\to\infty} \frac{(T_n - S)}{(S_n - S)} = 0$, ou seja, a transformação T *acelera a convergência* da sucessão (S_n), por outras palavras, a sucessão (T_n) *converge mais rapidamente* que (S_n).

Em geral estas propriedades não se verificam para todas as sucessões convergentes (S_n). Mais precisamente, foi provado, por Delahaye e Germain-Bonne, que não existe uma transformação universal T capaz de acelerar todas as sucessões

convergentes. Mesmo quando nos restringirmos a certas classes, como por exemplo a classe das sucessões monótonas, este resultado continua a ser válido. Por conseguinte, é sempre interessante encontrar e estudar novas transformações de sucessões uma vez que, de facto, cada uma delas está apta a acelerar a convergência de certas classes de sucessões.

É de salientar que transformações não lineares de sucessões têm usualmente melhores propriedades de aceleração que as lineares (elas aceleram classes maiores de sucessões) mas, por outro lado, elas nem sempre transformam uma sucessão convergente noutra sucessão convergente e, no caso de ser convergente os limites podem ser distintos. Por exemplo, no processo de Aitken (4.1), se (T_n) converge, então o seu limite é o mesmo que o limite da sucessão (S_n), mas pode acontecer que apesar de (S_n) convergir (T_n) possa ter dois pontos de acumulação distintos.

Exemplo 4.1 *Processo de soma (transformação linear)*

$$T_n = \frac{S_n + S_{n+1}}{2}, \quad n = 0, 1, \ldots$$

Neste caso obtemos um valor mais exacto de S a partir de S_n e S_{n+1}. Quais serão as sucessões cuja convergência este processo acelera ? Escrevendo

$$\frac{T_n - S}{S_n - S} = \frac{1}{2}\left(1 + \frac{S_{n+1} - S}{S_n - S}\right)$$

observamos que

$$\lim_{n\to\infty} \frac{T_n - S}{S_n - S} = 0,$$

se e só se

$$\lim_{n\to\infty} \frac{S_{n+1} - S}{S_n - S} = -1,$$

donde concluímos que, a classe de sucessões que esta transformação acelera, é muito restrita.

Exemplo 4.2 *Processo Δ^2 de Aitken (transformação não linear)*

$$T_n = \frac{S_n \times S_{n+2} - S_{n+1}^2}{S_{n+2} - 2S_{n+1} + S_n}, \quad n = 0, 1, \ldots \tag{4.1}$$

A classe de sucessões que este processo acelera é o conjunto das sucessões tais que existe $\lambda \in [-1; +1[$ tal que

$$\lim_{n\to\infty} \frac{S_{n+1} - S}{S_n - S} = -\lambda.$$

Observa-se que esta classe contém a do exemplo anterior.

Uma questão a ser posta é como construir transformações de sucessões. Uma forma é descobrir uma transformação a partir do seu núcleo, ou seja, do conjunto de sucessões para as quais $\exists S$ tal que $\forall n \geq N, T_n = S$. O interesse em falar do conceito de núcleo vem do facto de que se uma sucessão pertencer ao núcleo de uma transformação então essa transformação consegue acelerá-la. Observamos que se a sucessão a ser acelerada pertencer ao núcleo da transformação usada então, por construção, iremos ter $\forall n \geq N, T_n = S$.

Exemplo 4.3 *No caso do processo de soma torna-se simples concluir que o núcleo é o conjunto das sucessões da forma*

$$S_n = S + \alpha(-1)^n, \tag{4.2}$$

onde α é um escalar. Para o processo de Aitken o núcleo é o conjunto das sucessões da forma

$$S_n = S + a\lambda^n, \tag{4.3}$$

onde a e λ são escalares com $a \neq 0$ e $\lambda \neq 1$. Por conseguinte, o núcleo do processo de Aitken contém o núcleo do processo soma. Observamos que os núcleos dependem de vários parâmetros, S e α no primeiro caso, S, a e $\lambda(\neq 1)$ no segundo.

Existe um outro problema que deve ser mencionado neste momento. Quando usamos o processo de Aitken, o cálculo de T_n usa S_n, S_{n+1} e S_{n+2}. Para algumas sucessões é possível que $\lim_{n\to\infty} \frac{T_n - S}{S_n - S} = 0$ e que $\lim_{n\to\infty} \frac{T_n - S}{S_{n+1} - S}$ ou $\lim_{n\to\infty} \frac{T_n - S}{S_{n+2} - S}$ sejam diferentes de zero. Em particular se $\lim_{n\to\infty} \frac{S_{n+1} - S}{S_n - S} = 0$ então (T_n) obtida pelo processo Aitken converge mais depressa do que (S_n) e (S_{n+1}) mas nem sempre mais depressa que (S_{n+2}). Por conseguinte, no estudo de uma sucessão de transformações, será melhor observar a taxa $\frac{T_n - S}{S_{n+k} - S}$ onde S_{n+k} é o termo com maior índice usado no cálculo de T_n. Contudo tem de ser salientado que

$$\frac{T_n - S}{S_{n+k} - S} = \frac{T_n - S}{S_n - S} \times \frac{S_n - S}{S_{n+1} - S} \times \ldots \times \frac{S_{n+k-1} - S}{S_{n+k} - S},$$

o que mostra que se $\frac{T_n - S}{S_n - S}$ tende para zero e se $\frac{S_{n+1} - S}{S_n - S}$ é sempre diferente de zero e não tende para ele, então a taxa $\frac{T_n - S}{S_{n+k} - S}$ também tende para zero. Na prática, a exclusão do caso em que o limite de $\frac{S_{n+1} - S}{S_n - S}$ é nulo, não é uma restrição severa uma vez que, neste caso, (S_n) converge suficientemente rápido e não precisa de ser acelerada.

Claro que, usualmente, S é o limite da sucessão (S_n) mas este nem sempre é o caso e a questão precisa ser estudada. Por exemplo, no processo de Aitken, S é o limite de (S_n) se $|\lambda| < 1$. Se $|\lambda| > 1$, S_n diverge e S é frequentemente chamado o seu

anti-limite. Se $|\lambda| = 1$, (S_n) não tem limite nenhum ou apenas toma um conjunto finito de valores distintos e S é, neste caso, a sua média aritmética.

As duas expressões (4.2) e (4.3) dão uma forma explícita das sucessões que pertencem aos núcleos das transformações. Por esta razão iremos chamá-las as formas explícitas do núcleo. Contudo o núcleo pode também ser dado numa forma implícita, isto é, por meio de uma relação que se verifique ao longo de termos consecutivos da sucessão. Para a primeira transformação, é equivalente a escrever que, $\forall n$

$$S_{n+1} - S = -(S_n - S),$$

enquanto que, para o processo de Aitken, nós temos $\forall n$

$$S_{n+1} - S = \lambda(S_n - S).$$

Uma equação às diferenças como esta é chamada a forma implícita do núcleo porque ela não dá directamente (isto é, explicitamente) a forma das sucessões pertencentes ao núcleo mas apenas implicitamente como a solução desta equação às diferenças. Resolvendo esta equação às diferenças obtém-se a forma explícita do núcleo. Claro que, ambas as formas são equivalentes e dependem dos mesmos parâmetros.

4.3 Métodos de Extrapolação

Vamos ver como construir uma transformação de sucessões T a partir do seu núcleo. Pode acontecer também que tenhamos que encontrar o núcleo a partir de uma dada transformação para descobrirmos quais as sucessões que ela acelera, mas em geral isto é muito mais difícil.

Sejam $S, a_1, \ldots, a_p$, os parâmetros desconhecidos dos quais depende o núcleo da transformação T. Sabemos que a forma implícita do núcleo depende destes parâmetros e consiste numa relação ao longo de termos consecutivos da sucessão, isto é, uma relação da forma

$$R(S_n, \ldots, S_{n+q}, S) = 0, \quad \forall n \in \aleph, \tag{4.4}$$

que deve ser satisfeita $\forall n$, se e só se (S_n) pertence ao núcleo k_T da transformação T.

As definições seguintes foram baseadas em [3] pp. 5:

Definição 4.1 *Núcleo de uma Transformação – κ_T*

Uma sucessão S_n pertence ao núcleo de uma transformação de sucessões T se e só se a sucessão verifica a relação (4.4). □

Definição 4.2 *Método de extrapolação*

Uma transformação de sucessões $T : (S_n) \to (T_n)$ é um método de extrapolação se é tal que $\forall n \geq N, T_n = S$ se e só se $(S_n) \in k_T$. □

Sejam $S_n, S_{n+1}, \ldots, S_{n+p+q}$ dados, dependentes dos parâmetros desconhecidos $a_1, \ldots, a_p$, e seja $(x_n) \in k_T$ a sucessão satisfazendo as condições de interpolação

$$x_i = S_i, \quad i = n, \ldots, n+p+q.$$

Então, se (x_n) pertencer a k_T, ela satisfaz $\forall i$

$$R(x_i, \ldots, x_{i+q}, S) = 0.$$

Mas devido às condições de interpolação também temos

$$R(S_i, \ldots, S_{i+q}, S) = 0, \quad i = n, \ldots, n+p.$$

Isto é um sistema de $(p+1)$ equações com $(p+1)$ incógnitas $S, a_1, \ldots, a_p$ cuja solução depende de n. O valor que pretendemos encontrar é S logo basta-nos resolver este sistema para o obter e iremos denotá-lo por $S_{k,n}$.

Para resolvermos o sistema anterior temos que garantir a existência e unicidade de solução. Para isso assumimos que a derivada de R em relação à última variável é diferente de zero. Assim, pelo teorema da função implícita, garante-se a existência de uma função G (dependendo dos parâmetros $a_1, \ldots, a_p$) tal que

$$S = G(S_i, \ldots, S_{i+q})$$

para $i = n, \ldots, n+p$. A solução $S_{k,n} = S$ deste sistema depende de $S_n, \ldots, S_{n+k}$ e por conseguinte temos

$$S_{k,n} = F(S_n, \ldots, S_{n+k}).$$

Observação 4.1 *Se $(x_n) \notin k_T$ então $S_{k,n}$ não coincide com S, é apenas uma aproximação.*

4.4 Algoritmo–E

Nesta secção analisaremos o algoritmo–E que cobre a maioria dos restantes algoritmos de aceleração de convergência.

A desvantagem de um algoritmo tão genérico resulta, para cada caso específico, numa menor eficiência em termos de número de operações matemáticas e exigências de espaço computacional, que um algoritmo particularmente adaptado a cada caso.

A transformação–E é a transformação de sucessões mais geral que se conhece actualmente. Ela contém, como casos particulares, quase todas as transformações de sucessões conhecidas: extrapolação polinomial de Richardson, processo de soma, transformação de Shanks, a primeira generalização do algoritmo–ϵ, a transformação–G, transformação de Germain-Bonne, transformações generalizadas de Levin, processo p e extrapolação racional.

A transformação–E foi obtida, independentemente, por vários autores mas as duas formas mais gerais foram dadas por Havie e Brezinsky[3]. Ela pode ser vista como um caso particular da generalização do esquema de Neville-Aitken para o cálculo recursivo de interpolações polinomiais.

A transformação–E baseia-se na seguinte relação R que é válida para qualquer valor de n

$$S_n - S - a_1 g_1(n) - \ldots - a_k g_k(n) = 0,$$

onde os $g_i(n)$ são sucessões auxiliares dadas que podem depender de alguns termos da própria sucessão (S_n). Por outras palavras, assume-se que se (S_n) pertencer ao núcleo da transformação-E, então

$$S_n - S = a_1 g_1(n) + \ldots + a_k g_k(n). \tag{4.5}$$

Escrevendo esta relação para os índices $n, n+1, \ldots, n+k$ e resolvendo o sistema obtido em ordem à incógnita S, obtém-se

$$S = \frac{\begin{vmatrix} S_n & \ldots & S_{n+k} \\ g_1(n) & \ldots & g_1(n+k) \\ \vdots & & \vdots \\ g_k(n) & \ldots & g_k(n+k) \end{vmatrix}}{\begin{vmatrix} 1 & \ldots & 1 \\ g_1(n) & \ldots & g_1(n+k) \\ \vdots & & \vdots \\ g_k(n) & \ldots & g_k(n+k) \end{vmatrix}}. \tag{4.6}$$

Se (S_n) não pertencer ao núcleo, então o quociente (4.6) depende de k e de n e representa-se por $E_k^{(n)}$.

O núcleo desta transformação é definido pelo teorema seguinte, o qual se encontra em [3] pp. 57:

Teorema 4.1 $\forall n, E_k^{(n)} = S$ *se e só se* $\forall n$, $S_n = S + a_1 g_1(n) + \ldots + a_k g_k(n)$.□

Claro que se assume que o denominador de $E_k^{(n)}$ é diferente de zero. Também é bom referir que o núcleo da transformação $E_k : (S_n) \longmapsto (E_k^{(n)})$ depende das sucessões auxiliares $g_1, \ldots, g_k$. Assim, se a ordem de $g_1, \ldots, g_k$ é alterada, os núcleos das transformações $E_1, E_2, \ldots, E_{k-1}$ virão alterados em conformidade, mas E_k ficará inalterado.

Sucessões da forma

$$S_n = \frac{S + a_1 f_1(n) + \ldots + a_p f_p(n)}{1 + a_{p+1} h_1(n) + \ldots + a_{p+q} h_q(n)},$$

com $k = p + q$ também estão incluídas no núcleo E_k uma vez que a relação pode ser reescrita na forma

$$S_n = S + a_1 f_1(n) + \ldots + a_p f_p(n) - a_{p+1} S_n h_1(n) - \ldots - a_{p+q} S_n h_q(n)$$

ou

$$S_n = S + a_1 g_1(n) + \ldots + a_k g_k(n),$$

com $g_i(n) = f_i(n)$ para $i = 1, \ldots, p$ e $g_{i+p}(n) = S_n h_i(n)$ para $i = 1, \ldots, q$.

A escolha de $g_{2i-1}(n) = f_i(n)$, $g_{2i}(n) = S_n h_i(n)$ para $i = 1, \ldots, p$ quando $p = q$ pode ser igualmente feita, ou a escolha $g_i(n) = S_n h_i(n)$ para $i = 1, \ldots, q$ e $g_{i+q}(n) = f_i(n)$ para $i = 1, \ldots, p$, ou qualquer outra escolha. Assim, várias escolhas conduzem a valores diferentes de $E_i^{(n)}$ para $i = 1, \ldots, p + q - 1$ mas ao mesmo valor de $E_{p+q}^{(n)}$ para $i = 1, \ldots, p + q - 1$. Uma ordenação diferente dos g_i produz transformações intermédias E_i com núcleos diferentes e, portanto, esta ordem pode ser importante e adaptada às necessidades de cada caso particular.

A transformação–E inclui os seguintes casos particulares:

Processo de extrapolação de Richardson $g_i = x_n^i$ onde (x_n) é uma sucessão auxiliar.

Transformação–G $g_i(n) = x_{n+i-1}$ onde (x_n) é uma sucessão auxiliar.

Processo de soma $g_i(n) = x_i^n$ onde (x_n) é uma sucessão auxiliar.

Transformação de Shanks $g_i(n) = \Delta S_{n+i-1}$.

Transformação de Germain-Bonne $g_i(n) = (\Delta S_n)^i$.

Transformação generalizada de Levin $g_i(n) = x_n^{i-1}\Delta S_n/y_n$ onde (x_n) e (y_n) são sucessões auxiliares.

Primeira generalização do algoritmo–ϵ $g_i(n) = R^i(S_n\Delta x_n)/\Delta x_n$ onde (x_n) é uma sucessão auxiliar e R o operador diferença que generaliza Δ é definido por

$$R^{k+1}v_n = \Delta\left(\frac{R^k v_n}{\Delta x_n}\right) = R^k\Delta\left(\frac{v_n}{\Delta x_n}\right)$$

onde $v_n = S_n\Delta x_n$.

Processo p $g_1(n) = x_n$ e $g_i(n) = \Delta S_{n+i-2}$ para $i \geq 2$ onde (x_n) é uma sucessão auxiliar.

Extrapolação racional de Thiele $g_i(n) = x_n^i$ para $i = 1, \ldots, p, g_{i+p}(n) = S_n x_n^i$ para $i = 1, \ldots, p, k = 2p$ onde (x_n) é uma sucessão auxiliar.

O algoritmo recursivo que permite calcular os números $E_k^{(n)}$ para todo o k e n sem calcular os determinantes envolvidos na sua definição (4.6), é o algoritmo–E cujas regras normais são

$$\begin{aligned} E_0^{(n)} &= S_n, \quad n = 0, 1, \ldots \\ g_{0,i}^{(n)} &= g_i(n), \quad n = 0, 1, \ldots; \ i = 1, 2, \ldots \end{aligned}$$

Então para $k = 1, 2, \ldots$ e $n = 0, 1, \ldots$ temos

$$E_k^{(n)} = E_{k-1}^{(n)} - \frac{E_{k-1}^{(n+1)} - E_{k-1}^{(n)}}{g_{k-1,k}^{(n+1)} - g_{k-1,k}^{(n)}} \times g_{k-1,k}^{(n)},$$

onde os $g_{k-1,k}^{(n)}$ são quantidades auxiliares calculadas recursivamente através de

$$g_{k-1,k}^{(n)} = g_{k-2,k}^{(n)} - \frac{g_{k-2,k}^{(n+1)} - g_{k-2,k}^{(n)}}{g_{k-2,k-1}^{(n+1)} - g_{k-2,k-1}^{(n)}} \times g_{k-2,k-1}^{(n)}.$$

4.5 Método de Richardson

No princípio deste século, Richardson formulou um método que permite aumentar a precisão das soluções numéricas dos problemas lineares. Ele introduziu um processo que permite criar algoritmos que utilizam resultados de diferentes

aproximações numéricas com vários parâmetros de aproximação. A ideia de Richardson consiste em aplicar um conjunto de soluções aproximadas numa sucessão de redes por forma a obter soluções aproximadas com maior precisão.

A teoria da extrapolação de Richardson primeiramente foi formulada para problemas de equações diferenciais lineares e mais tarde generalizada; ela pode por exemplo ser aplicada a equações integrais lineares desde que se façam as devidas alterações.

O processo de extrapolação de Richardson é o caso particular do algoritmo-E em que a sucessão a ser transformada pode ser aproximada por uma expressão da forma

$$S_i = S + a_1(x_i) + a_2(x_i)^2 + \ldots + a_m(x_i)^m, \tag{4.7}$$

onde (x_i) é uma sucessão auxiliar, de números reais, tal que $x_i \neq x_j$ para $i \neq j$, ou seja, a relação (4.4), neste caso, é a seguinte:

$$S_i - S - a_1(x_i) - a_2(x_i)^2 - \ldots - a_m(x_i)^m = 0.$$

Escrevendo a relação anterior para $i = n, n+1, \ldots, n+m$ obtemos um sistema de $(m+1)$ equações a $(m+1)$ incógnitas, $S, a_1, a_2, \ldots, a_m$. Se (S_i) satisfizer a igualdade (4.7), $\forall i \in \aleph$, então resolvendo este sistema obtemos a solução

$$S = \frac{\begin{vmatrix} S_n & \ldots & S_{n+m} \\ x_n & \ldots & x_{n+m} \\ \vdots & & \vdots \\ x_n^m & \ldots & x_{n+m}^m \end{vmatrix}}{\begin{vmatrix} 1 & \ldots & 1 \\ x_n & \ldots & x_{n+m} \\ \vdots & & \vdots \\ x_n^m & \ldots & x_{n+m}^m \end{vmatrix}}, \tag{4.8}$$

onde o quociente (4.8) não depende de n nem de m. No caso de (S_i) não satisfazer (4.7) exactamente, a grandeza do segundo membro de (4.8) já depende de n e m e passamos a representá-la por $S_{n,m}$. Consequentemente, podemos afirmar que $S_{n,m} = S \forall n$ se e só se $S_n = S + a_1(x_n) + a_2(x_n)^2 + \ldots + a_m(x_n)^m$, ou seja, S_n pertence ao núcleo da transformada T, k_T, se e só se S_n satisfizer (4.7).

4.5.1 Construção do Algoritmo

Observamos que o cálculo de $S_{n,m}$ necessita do cálculo de dois determinantes de dimensão $(m+1)$ o que, como sabemos, involve cerca de $2(m+1)(m+1)!$

multiplicações; por exemplo para resolvermos um sistema de 7 equações temos cerca de 70560 multiplicações. Mesmo que consigamos um computador que faça os cálculos num tempo razoável não podemos fugir aos erros de arredondamento que vão afectar grandemente o resultado. Para evitar o cálculo destes determinantes vamos construir um algoritmo recursivo para resolver o sistema que nos propomos resolver.

A ideia da construção do algoritmo é, a partir de várias aproximações diferentes de S, obter uma aproximação mais precisa desta grandeza. Isto é conseguido eliminando o termo em $(x_i)^k$ na expressão

$$S = S_i - \sum_{\alpha=k}^{m} a_\alpha (x_i)^\alpha, \tag{4.9}$$

onde (x_i) é tal que $x_{i+k} = r^k x_i$ com $0 < r < 1$ e $k = 1, 2, \ldots$ (em geral $r = 1/2$ e $x_i = h$).

Sejam S_n e S_{n+1} as duas aproximações referidas, então

$$S = S_n - \sum_{\alpha=k}^{m} a_\alpha (x_n)^\alpha,$$

$$S = S_{n+1} - \sum_{\alpha=k}^{m} a_\alpha (x_{n+1})^\alpha.$$

Multiplicando a primeira equação por x_{n+k} e a segunda por x_n obtemos

$$x_{n+k} S = x_{n+k} S_n - \sum_{\alpha=k}^{m} a_\alpha (x_n)^\alpha x_{n+k},$$

$$x_n S = x_n S_{n+1} - \sum_{\alpha=k}^{m} a_\alpha (x_{n+1})^\alpha x_n.$$

Subtraíndo as duas equações anteriores vem

$$(x_n - x_{n+k}) S = x_n S_{n+1} - x_{n+k} S_n - \sum_{\alpha=k}^{m} a_\alpha [(x_{n+1})^\alpha x_n - (x_n)^\alpha x_{n+k}],$$

ou seja,

$$S = \frac{x_n S_{n+1} - x_{n+k} S_n}{x_n - x_{n+k}} - \sum_{\alpha=k}^{m} a_\alpha \frac{(x_{n+1})^\alpha x_n - (x_n)^\alpha x_{n+k}}{x_n - x_{n+k}}.$$

Mas como $x_{n+k} = r^k x_n$ obtemos

$$S = \frac{x_n S_{n+1} - x_{n+k} S_n}{x_n - x_{n+k}} - \sum_{\alpha=k}^{m} a_\alpha \frac{r^\alpha - r^k}{1 - r^k} (x_n)^\alpha.$$

Observamos que o termo em $(x_n)^k$ é nulo logo,

$$S = S_{k,n} - \sum_{\alpha=k+1}^{m} b_\alpha (x_n)^\alpha,$$

onde

$$b_\alpha = a_\alpha \frac{r^\alpha - r^k}{1 - r^k},$$

$$S_{k,n} = \frac{x_n S_{n+1} - x_{n+k} S_n}{x_n - x_{n+k}}.$$

$S_{k,n}$ é a aproximação de S que satisfaz (4.9), com a diferença de que o desenvolvimento do erro começa em $(x_n)^{k+1}$ em vez de $(x_n)^k$. Por conseguinte somos tentados a pensar que $S_{k,n}$ aproxima melhor S que S_n e S_{n+1}. É de salientar que, para obtermos uma nova aproximação de S, não necessitamos de conhecer os coeficientes a_α mas apenas k e as duas aproximações donde partimos. Sendo assim, definindo $S_{0,m} = S_m$ para $m = 0, 1, 2, \ldots$, obtemos o seguinte algoritmo

$$S_{0,m} = S_m, \qquad m = 0, 1, 2, \ldots \tag{4.10}$$

$$S_{k,m} = \frac{x_m S_{k-1,m+1} - x_{m+k} S_{k-1,m}}{x_m - x_{m+k}}, \quad k = 1, 2, \ldots; \;\; m = 0, 1, 2, \ldots \tag{4.11}$$

Observação 4.2 *Observamos que $S_{k,m}$ é o valor, no ponto zero, do polinómio interpolador $P(x)$, de grau $\leq (m - k + 1)$, que satisfaz $P(x_{m+j}) = S_{m+j}, j = 0, 1, \ldots, k$. É de salientar que a expressão anterior não é mais que o algoritmo de Neville-Aitken no ponto $x = 0$. Sendo assim, o processo de Richardson não é mais do que a extrapolação polinomial no ponto zero.*

Observação 4.3 *Prova-se que as quantidades $S_{k,m}$ são de facto as dadas pela razão de determinantes (4.8).*

4.6 Existência de Desenvolvimento Assimptótico

Nesta secção consideramos um problema, que é um caso particular do problema abstracto estudado por Marchuk [34], e enunciamos condições suficientes para a existência de desenvolvimentos assimptóticos, do erro $u^h - y$, quer em potências inteiras, quer em potências fraccionárias, do diâmetro da rede h. Observe-se que o desenvolvimento em potências inteiras é do tipo (4.12).

Sejam y e u^h as soluções exacta e aproximada de um certo problema. Suponha-se que u^h, obtida através de um algoritmo finito dependente de um parâmetro $h > 0$, converge para y quando $h \to 0$. Suponha-se ainda que a solução aproximada u^h possui um desenvolvimento assimptótico em potências do diâmetro da rede h, do tipo:

$$u^h = y + v_1 h + v_2 h^2 + \ldots + v_m h^m + \eta^h \quad com \quad \|\eta^h\| = O(h^{m+\beta}), \tag{4.12}$$

onde $v_k, k = 1, \ldots, m$ não depende de h.

Este desenvolvimento forma a base da extrapolação de Richardson (ver secção 4.5): O valor pretendido $u^0 := \lim_{h\to 0} u^h = y$ é aproximado por extrapolação a partir de vários valores $u^{h_\nu}, h_\nu > 0$.

Nesta secção consideramos um problema, que é um caso particular do problema abstracto estudado por Marchuk [34] pp. 16, e enunciamos condições suficientes para a existência de um desenvolvimento assimptótico do tipo (4.12), e de um mais geral que é um desenvolvimento assimptótico em potências não inteiras do tipo (4.22). Toda a análise feita foi baseada em Marchuk [34] pp. 16-24.

Seja $I := [a, b]$ um intervalo fechado em $\Re$. Considere-se que pretendemos resolver o problema (típico na física matemática):

$$Ly = g \ \ em \ \ I, \tag{4.13}$$

onde L é um operador linear integral e as funções y e g são definidas em I.

Nos espaços lineares de funções definidos nestes conjuntos iremos introduzir as classes de funções $P_k(I)$ e $M_k(I)$ que irão depender de um parâmetro inteiro $k \geq 0$. Geralmente estas classes caracterizam a regularidade do segundo membro da equação e da solução. A sua definição, para um dado problema, depende dos resultados conhecidos de existência de solução.

Posto isto comecemos com uma condição de existência e unicidade:

Condição 4.1 *Para todo o inteiro k, $0 \leq k \leq m$, e toda a função $g \in M_k(I)$ existe uma solução única $y \in P_k(I)$ de (4.13).*□

Para encontrarmos a solução numérica de (4.13) introduzimos a rede $\Pi_h \subset I$,

$$\Pi_h : a = t_0 < t_1 < \ldots < t_N = b,$$

onde $t_n = a + nh$, $n = 0, 1, \ldots, N$. Observe-se que a rede depende do parâmetro variável h, que pode ser tão pequeno quanto se queira. Iremos procurar uma solução aproximada no espaço de funções rede, isto é, funções de variável discreta definidas nos nós da rede. Considere-se o problema aproximado através do uso de um determinado método numérico

$$L_h u^h = g \ \ \ em \ \ \ \Pi_h, \tag{4.14}$$

onde L_h é um operador linear algébrico e u^h é a função rede que aproxima a solução y da equação integral inicial em Π_h. Consideremos a norma $\|.\|_{\Pi_h}$, definida por

$$\|y\|_{\Pi_h} := \max_{0\leq n\leq N} |y(t_n)|. \tag{4.15}$$

Iremos agora enunciar uma condição de estabilidade em termos destas normas caracterizando a solubilidade de (4.14) e a estabilidade da sua solução.

Condição 4.2 *Se a função rede ψ^h está definida em Π_h e é uma solução do problema*

$$L_h\psi^h = g^h \quad em \quad \Pi_h, \tag{4.16}$$

onde g^h é uma função rede com domínio de definição Π_h, então verifica-se a estimativa

$$\|\psi^h\|_{\Pi_h} \leq c\left(\|g^h\|_{\Pi_h}\right), \tag{4.17}$$

onde c é uma constante real positiva que não depende de h. □

Observe-se que a existência de uma solução única de (4.14) resulta de (4.17). Sabemos que, para cada g, o problema (4.14) tem uma solução única se o problema homogéneo correspondente, $L_h\xi_h = 0$, tem apenas a solução nula. Mas se considerarmos o problema homogéneo e aplicarmos (4.17) concluimos que $\|\xi^h\|_{\Pi_h} = 0$, donde $\xi^h = 0$ em Π_h.

Resta-nos enunciar a condição de consistência que está directamente relacionada com a aproximação de operadores de dimensão infinita através de operadores de dimensão finita.

Condição 4.3 *O desenvolvimento*

$$L_h\phi = L\phi + \sum_{j=1}^{k} h^j a_j + \sigma^h \quad em \quad \Pi_h, \tag{4.18}$$

é válido para toda a função $\phi \in P_k(I)$, onde $0 \leq k \leq m$. As funções a_j são independentes de h, $a_j \in M_{k-j}(I)$, e o termo restante σ^h verifica a estimativa

$$\|\sigma^h\|_{\Pi_h} \leq c_1 h^{k+\beta}, \tag{4.19}$$

onde a constante c_1 é independente de h, e β é independente de h, k e ϕ. □

Estas condições levam-nos a obter desenvolvimentos idênticos a (4.18) para a solução aproximada u^h de (4.14).

Teorema 4.2 *Suponhamos que as condições 4.1, 4.2 e 4.3 se verificam para (4.13), e (4.14) e $g \in M_m(I)$. Então a solução aproximada u^h tem o desenvolvimento*

$$u^h = y + \sum_{j=1}^{m} h^j v_j + \eta^h \quad em \quad \Pi_h. \tag{4.20}$$

As funções v_j são independentes de h, $v_j \in P_{m-j}(I)$, e o termo restante η^h verifica a estimativa

$$\|\eta^h\|_{\Pi_h} \leq c_2 h^{m+\beta}, \tag{4.21}$$

onde a constante c_2 é independente de h. □

A demonstração deste resultado encontra-se em [34] (pp. 18). Foi dito anteriormente que o desenvolvimento (4.20) do teorema 4.2 é a base do método de extrapolação de Richardson. Com efeito, iremos usar este desenvolvimento para melhorar a precisão das soluções aproximadas de (4.14). Suponhamos que as condições do teorema 4.2 são satisfeitas para um conjunto de redes $\overline{\Omega}_{h_k}$ com parâmetros

$$h_1 > h_2 > \ldots > h_{m+1} > 0.$$

Iremos assumir que estas redes têm intersecção não vazia, isto é,

$$\Pi_H = \bigcap_{k=1}^{m+1} \Pi_{h_k} \neq \emptyset.$$

De acordo com a condição 4.2 o sistema (4.14) tem uma solução única para cada valor do parâmetro $h = h_k$. Iremos denotar esta solução por u^{h_k}. Se u^H, definida em Π_H, fôr a solução extrapolada através do método de extrapolação de Richardson, então ela tem maior grau de precisão do que cada u^{h_k}:

Teorema 4.3 *Suponhamos que as redes Π_{h_k} com parâmetros $h_1 > h_2 > \ldots > h_{m+1} > 0$ satisfazem as condições do teorema 4.2 com a norma uniforme, para funções rede definidas em Π_h,*

$$\|v\|_{\Pi_h} = \max_{x \in \Pi_h} |v(x)|$$

e que se verifica a desigualdade

$$\frac{h_k}{h_{k+1}} \geq 1 + d_1, \qquad k = 1, \ldots, m,$$

com a constante $d_1 > 0$ independente de h_k. Então se Π_H fôr uma intersecção não vazia destas redes verifica-se a estimativa

$$\max_{\Pi_h} |u^H - y| \leq d_2 h_1^{m+\beta},$$

onde u^H é a solução extrapolada através do método de Richardson, y é a solução da equação integral (4.13) e d_2 é uma constante independente de h_k. □

Acontece por vezes que o desenvolvimento (4.18) contém potências não inteiras de h. Isto faz com que não seja garantida a aceleração de convergência utilizando o método de Richardson. Contudo podemos recorrer ao algoritmo-E. Por conseguinte ir-se-á modificar a condição 4.3 quando ocorrem potências não inteiras de h e ir-se-á provar o teorema correspondente sobre a existência de desenvolvimento assimptótico do erro $u^h - y$.

Condição 4.4 *O desenvolvimento*

$$L_h\phi = L\phi + \sum_{j=1}^{pq} h^{j/q} a_j + \sigma^h, \quad em \quad \Pi_h,$$

é válido para toda a função $\phi \in C^{p+1}$, p e q são inteiros tais que $p \geq 0$ e $q \geq 1$, $\mu > 1$ e as funções a_j são independentes de h, $a_j \in C^{p-j}(I)$, e o termo restante σ^h verifica a estimativa

$$\|\sigma^h\|_{\Pi_h} \leq c_3 h^{p+1}$$

onde a constante c_3 é independente de h, p, q e ϕ.□

O teorema seguinte é uma generalização do teorema 4.2 e permite obter desenvolvimentos, por exemplo, no caso de $\mu = 1/2$, do tipo

$$u^h = y + v_1 h + v_{1/2} h^{3/2} + v_2 h^2 + \ldots + O(h^{p+1}).$$

A sua demonstração é feita com base na demonstração do teorema 4.2 em [34] pp. 18-20.

Teorema 4.4 *Suponha-se que as condições 4.1, 4.2 e 4.4 se verificam para (4.13), e (4.14) e $f \in M_m(\Omega)$ e $g \in N_m(D)$. Então a solução aproximada u^h tem o desenvolvimento assimptótico*

$$u^h = y + \sum_{j=1}^{pq} h^{j/q} v_j + \eta^h, \quad em \quad \Pi_h, \tag{4.22}$$

onde as funções v_j são independentes de h, $v_j \in P_{p-j} := C^{p-j+1}(I)$ e o termo restante η^h satisfaz a estimativa

$$\|\eta^h\|_{\Pi} \leq c_4 h^{p+1}.$$

Demonstração Sejam $v_j \in C^{p-j+1}(I)$, $j = 1, \ldots, p$, um conjunto arbitrário de funções independentes de h. Usando estas funções e as duas soluções y e u^h defina-se a função rede

$$\eta^h := u^h - y - \sum_{j=1}^{pq} h^{j/q} v_j, \quad em\ \Pi_h. \tag{4.23}$$

Resolvendo (4.23) em ordem a u^h e substituindo a expressão resultante na equação (4.14) obtém-se

$$L_h y + \sum_{j=1}^{pq} h^{j/q} L_h v_j + L_h \eta^h = g, \quad em\ \Pi_h. \tag{4.24}$$

Usando a condição 4.4 podemos escrever os desenvolvimentos

$$L_h y = g + \sum_{i=1}^{pq} h^{i/q} a_{0,i} + \sigma_0^h \quad em \quad \Pi_h, \tag{4.25}$$

$$L_h v_j = L v_j + \sum_{i=1}^{(p-j)q} h^{i/q} a_{j,i} + \sigma_j^h \quad em \quad \Pi_h, \tag{4.26}$$

onde

$$a_{j,i} \in C^{p-j-i}(I) \tag{4.27}$$

são independentes de h e os termos restantes satisfazem

$$\|\sigma_j^h\|_{\Pi_h} \leq c_{j,1} h^{p+1-j}, \tag{4.28}$$

com $c_{j,1}$ independentes de h. Recorrendo às equações (4.25) e (4.26) podemos escrever (4.24) na seguinte forma:

$$g + \sum_{j=1}^{pq} h^{j/q} L v_j + \sum_{j=0}^{pq} h^{j/q} \sum_{i=1}^{(p-j)q} h^{i/q} a_{j,i} + \sum_{j=0}^{pq} h^{j/q} \sigma_j^h + L_h \eta^h = g, \quad \Pi_h. \tag{4.29}$$

Definindo $\xi_h := \sum_{j=1}^{pq} h^{j/q} \sigma_j^h$ e usando (4.28) obtém-se

$$\|\xi^h\|_{\Pi_h} \leq c_5 h^{p+1}, \tag{4.30}$$

onde

$$c_5 = \sum_{j=1}^{pq} c_{j,1}.$$

Reordenando os termos dos somatórios em (4.29) podemos obter

$$\sum_{j=1}^{p} h^{j/q} \left(L v_j + \sum_{i=1}^{j} a_{j-i,i} \right) + \xi^h + L_h \eta^h = 0, \quad em\ \Pi_h. \tag{4.31}$$

onde ξ^h satisfaz (4.30) para um conjunto arbitrário de funções $v_j \in C^{p-j+1}(I)$ e com η^h definida pela igualdade (4.23). Sendo assim podemos escolher as funções v_j, $j = 1, 2, \ldots, p$, de tal modo que satisfaçam as equações

$$L v_j = -\sum_{i=1}^{j} a_{j-i,i} \quad em \quad I. \tag{4.32}$$

Por conseguinte, por exemplo, a função v_1 é obtida a partir de

$$Lv_1 = -a_{0,1} \quad em \quad I.$$

Conclui-se, utilizando (4.27), que $a_{0,1} \in C^{p-1}(I)$. Consequentemente, pela condição 4.1, v_1 é definida de forma única e pertence ao espaço $C^{p-1}(I)$.

Agora suponha-se que as funções $v_j \in C^{p-j}(I)$ foram determinadas para $j = 1, \ldots, k$, onde $1 \leq k \leq p$. Então, uma vez que a condição 4.3 se verifica, (4.26) é válida para $j = 1, \ldots, k$ e satisfaz (4.27). Para $j = k+1$ em (4.32) temos

$$Lv_{k+1} = -\sum_{i=1}^{k+1} a_{k-i+1,i} \quad em \quad I, \tag{4.33}$$

onde, atendendo a (4.27), o segundo membro da equação pertence a $C^{p-k-1}(I)$. Uma vez que a condição 4.1 se verifica, segue que (4.33) tem solução única $v_{k+1} \in C^{p-k-1}(I)$. Note-se que v_{k+1} não depende de h porque nem o operador L, nem o segundo membro da equação dependem.

Sendo assim, apresentámos uma forma de construir as funções $v_j \in C^{p-j}(I)$, independentes de h, $j = 1, \ldots, p$. Estas funções v_j satisfazem (4.31) com ξ^h verificando (4.30), e a partir de (4.32), as relações (4.31) assumem a forma

$$L_h \eta^h = -\xi^h \quad em \quad I.$$

Uma vez que se verifica a condição 4.2, dela resulta que

$$\|\eta^h\|_{\Pi_h} \leq c\|\xi^h\|_{\Pi_h}.$$

Usando (4.30) obtém-se

$$\|\eta^h\|_{\Pi_h} \leq c_4 h^{p+1},$$

onde $c_4 = cc_5$.

A partir de (4.23) obtém-se

$$u^h = y + \sum_{j=1}^{pq} h^{j/q} v_j + \eta^h, \quad em \quad \Pi_h,$$

com as propriedades desejadas, o que conclui a demonstração.□

4.7 Aplicação ao Problema

O problema que estudamos neste trabalho, que é de grande interesse e importância do ponto de vista das aplicações, fornece resultados que podem ser extrapolados em relação ao diâmetro da rede. As equações integrais a uma dimensão, na ausência de condições de fronteira, são os exemplos mais simples que podemos utilizar para ilustrar os teoremas gerais da secção 4.6.

Considere-se a equação (3.7) nas condições do teorema 3.2. Este teorema garante-nos que a equação (3.7) possui uma solução única $y \in P_m$, ou seja, a condição 4.1 da secção 4.6 verifica-se para $I := [0,T]$ com as classes de funções $M_m(I) = P_m(I) = V_m$ (ver secção 3.1.2).

Iremos calcular a solução aproximada com o método de Euler, apresentado na secção 3.2.1, na rede $\Pi_h := \{t_i = ih, i = 0,1,\ldots,N\}$ com diâmetro de rede $h = T/N$, e $N \geq 1$, N inteiro:

$$\begin{aligned} u^h(t_0) &= \frac{\mu}{\mu-1} g_2(t_0), \\ u^h(t_n) &= g_2(t_n) + \frac{1}{\mu n^\mu} \sum_{i=0}^{n-1} ((i+1)^\mu - i^\mu) u^h(t_i), \quad n = 0,1,\ldots,N. \end{aligned}$$

Lema 4.1 *Suponhamos que são verificadas as condições de convergência do método de Euler, ou seja, $g_2 \in C^1$ e $\mu > 1$. Então verifica-se*

$$\|u^h\|_{\Pi_h} \leq c_3 \|g_2^h\|_{\Pi_h} \quad onde \quad c_3 = \frac{\mu}{\mu-1},$$

onde u^h é a solução aproximada de (3.19).

Demonstração Sabemos que a equação aproximada verifica

$$u^h(t) - \int_0^t p_2(t,s) u^h(s) ds = g_2(t), \quad t \in \Pi_h,$$

ou seja,

$$u^h(t_n) - \int_0^{t_n} p_2(t_n,s) u^h(s) ds = g_2(t_n), \quad n = 0,1,\ldots,N.$$

Aplicando a propriedade aditiva dos integrais podemos escrever

$$u^h(t_n) = \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} p_2(t_n,s) u^h(s) ds + g_2(t_n), \quad n = 0,1,\ldots,N.$$

Podemos então calcular a norma de u^h, de acordo com a definição 4.15, e vem

$$\begin{aligned}\|u^h\|_{\Pi_h} &= \max_{0\leq n\leq N}|u^h(t_n)| \\ &= \max_{0\leq n\leq N}\left|\sum_{i=0}^{n-1}\int_{t_i}^{t_{i+1}} p_2(t_n,s)ds u^h(t_i)+g_2(t_n)\right|.\end{aligned}$$

Aplicando a desigualdade triangular obtemos

$$\|u^h\|_{\Pi_h}\leq \max_{0\leq n\leq N}\left\{\sum_{i=0}^{n-1}\left|\int_{t_i}^{t_{i+1}} p_2(t_n,s)ds u^h(t_i)\right|+|g_2(t_n)|\right\}. \tag{4.34}$$

Segue que,

$$\|u^h\|_{\Pi_h}\leq \max_{0\leq n\leq N}\left\{\sum_{i=0}^{n-1}\left|\int_{t_i}^{t_{i+1}} p_2(t_n,s)ds\right||u^h(t_i)|+|g_2(t_n)|\right\}. \tag{4.35}$$

Uma vez que $\|u^h\|_{\Pi_h}=\max_{0\leq n\leq N}|u^h(t_n)|$ podemos escrever

$$\|u^h\|_{\Pi_h}\leq \max_{0\leq n\leq N}\left\{\sum_{i=0}^{n-1}\left|\int_{t_i}^{t_{i+1}} p_2(t_n,s)ds\right|\max_{0\leq n\leq N}|u^h(t_n)|+|g_2(t_n)|\right\}. \tag{4.36}$$

ou seja,

$$\|u^h\|_{\Pi_h}\leq \max_{0\leq n\leq N}\left\{\sum_{i=0}^{n-1}\left|\int_{t_i}^{t_{i+1}} p_2(t_n,s)ds\right|\|u^h\|_{\Pi_h}+|g_2(t_n)|\right\}. \tag{4.37}$$

Uma vez que $p_2(t,s)=(s/t)^\mu/s$ vem

$$\int_{t_i}^{t_{i+1}} p_2(t_n,s)ds=\frac{1}{\mu n^\mu}\left((i+1)^\mu-i^\mu\right),\mu>0. \tag{4.38}$$

Observamos que este integral é sempre positivo logo

$$\|u^h\|_{\Pi_h}\leq \max_{0\leq n\leq N}\left\{\sum_{i=0}^{n-1}\int_{t_i}^{t_{i+1}} p_2(t_n,s)ds\|u^h\|_{\Pi_h}+|g_2(t_n)|\right\}. \tag{4.39}$$

Atendendo a (4.38) vem

$$\sum_{i=0}^{n-1}\int_{t_i}^{t_{i+1}} p_2(t_n,s)ds=\sum_{i=0}^{n-1}\frac{1}{\mu n^\mu}\left((i+1)^\mu-i^\mu\right)=\frac{1}{\mu}.$$

Tendo em conta o resultado anterior e o facto de que $\max_{0\leq n\leq N}|g_2(t_n)|=\|g_2\|_{\Pi_h}$ vem,

$$\|u^h\|_{\Pi_h}\leq\frac{1}{\mu}\|u^h\|_{\Pi_h}+\|g_2\|_{\Pi_h},$$

segue que,

$$\frac{\mu-1}{\mu}\|u^h\|_{\Pi_h}\leq\|g_2\|_{\Pi_h}.$$

Mas $\mu > 1$ logo $(\mu - 1)/\mu > 0$, por conseguinte,

$$\|u^h\|_{\Pi_h} \leq c_3 \|g_2\|_{\Pi_h},$$

onde

$$c_3 = \frac{\mu}{\mu - 1},$$

como queriamos demonstrar. □

Sendo assim provámos que se verifica a condição de estabilidade 4.2 da secção 4.6, para o método de Euler (3.19).

O nosso objectivo é aplicar o teorema 4.2 para provar a existência de um desenvolvimento assimptótico do erro, com a forma (4.20), no caso de o método de Euler ser utilizado para aproximar a solução da equação (3.7). Aplicando a notação do teorema 4.2 a este caso concreto, defina-se o operador L através da igualdade:

$$Lu(t) := u(t) - \int_0^t p_2(t,s)u(s)ds, \quad 0 \leq t \leq T, \tag{4.40}$$

onde $p_2(t,s)$ é definido por (3.8). Analogamente, seja

$$L_h u(t) := u(t) - \sum_{i=0}^{n-1} \int_{ih}^{(i+1)h} p_2(t,s)u(t_i)ds, \quad 0 \leq t \leq T, \tag{4.41}$$

onde $h = T/N$.

Para verificar a condição 4.3 é necessário provar que os operadores, definidos por (4.40) e (4.41), satisfazem a igualdade (4.18), isto é, qualquer que seja a função $\phi \in P_k(I)$ existem funções $a_j \in M_{k-j}(I)$, independentes de h e tais que

$$L_h\phi = L\phi + \sum_{j=1}^{k} h^j a_j + \sigma^h \quad em\ \Pi_h, \tag{4.42}$$

Se considerarmos, como até aqui, as classes de funções $P_m(I) = M_m(I) = V_m(I)$, onde V_m são os espaços definidos no teorema 3.2 (sobre existência e unicidade de solução), a condição 4.3 não se verifica para este problema.

No entanto, os resultados numéricos que são apresentados na secção 4.9 e que foram obtidos com a aplicação de métodos de exptrapolação, sugerem que nos casos particulares considerados nesses exemplos, o teorema 4.2 pode ser aplicado e existe, de facto, um desenvolvimento assimptótico do erro, do tipo (4.20). Para demonstrar teoricamente este facto dever-se-á considerar a equação em classes de funções mais restritas, onde a solução satisfaça certas condições suplementares. Este problema deverá ser objecto de estudo posterior.

Observação 4.4 *A existência de desenvolvimento assimptótico para as regras de integração no caso de funções integrandas singulares foi estudado por Lyness e Ninham [32]. Hoog e Weiss [20] obtiveram resultados análogos para as regras de integração produto. Estes resultados sugerem que os erros dos métodos numéricos para equações integrais, do tipo considerado nesta tese, possam ter desenvolvimentos assimptóticos em potências não inteiras do passo h. A existência de tais desenvolvimentos é de grande interesse prático visto que eles permitem utilizar o algoritmo-E para obter aceleração de convergência [26].*

4.8 Realização

A aceleração de convergência, pelo método de Richardson, é garantida quando se tem um desenvolvimento assimptótico do tipo (4.20). Isto equivale a considerar $x_i = h_i$ no desenvolvimento (4.7). Por conseguinte o algoritmo de Richardson (4.10), (4.11), surge da seguinte forma:

$$S_{0,m} = S_m, \qquad m = 0, 1, 2, \ldots \tag{4.43}$$

$$S_{k,m} = \frac{h_m S_{k-1,m+1} - h_{m+k} S_{k-1,m}}{h_m - h_{m+k}}, \quad k = 1, 2, \ldots; \;\; m = 0, 1, 2, \ldots \tag{4.44}$$

Suponhamos que $h_0 = 0.1$ e apliquemos o método de Euler, à nossa equação, em $[0, 2]$. O algoritmo dá-nos valores nos pontos 0.0, 0.1, 0.2, ..., 2.0. Se aplicarmos o método de Euler com $h_1 = h_0/2 = 0.05$ obtêm-se os valores em 0.0, 0.05, 0.1, 0.15, 0.2, ..., 2.0. Note-se que os pontos 0.0, 0.1, 0.2, ..., 2.0, são comuns às duas partições. Para acelerarmos a convergência, num ponto, temos de ter várias aproximações desse ponto. Sendo assim faz sentido escolher, por exemplo,

$$h_i = \frac{h_0}{2^i}, \qquad i = 1, 2, \ldots,$$

com h_0 dado. Se $h_i = h_0/2^i$ então (4.44) resulta na forma mais simples,

$$S_{k,m} = \frac{S_{k-1,m+1} - 0.5^k S_{k-1,m}}{1 - 0.5^k}.$$

Após esta simplificação resta-nos apresentar o pseudo-código correspondente:

$$\begin{aligned}
&loop\ m = 1, N \\
&\quad S_{0,m} = S_m \\
&loop\ k = 1, N \\
&\quad loop\ m = 0, N - k \\
&\qquad S_{k,m} = \frac{S_{k-1,m+1} - 0.5^k S_{k-1,m}}{1 - 0.5^k}
\end{aligned}$$

Cada vez que é determinado o valor de $S_{k,m}$, a potência 0.5^k é calculada duas vezes. Dado que o cálculo das potências é dispendioso pode-se optimizar o algoritmo calculando a potência 0.5^k antes de entrar no ciclo em m. Por conseguinte, o número de operações do algoritmo de Richardson após a optimização, é

$$N\left(Pot + (Sub + Mult)(N + (N-1) + (N-2) + \ldots + 2 + 1)\right)$$

onde, N :=número de pontos para extrapolar, $Pot := 1$ potência 0.5^k, $Sub := 2$ subtrações, $Mult := 1$ multiplicação + 1 divisão. Sendo assim, o tempo de computação do algoritmo de Richardson é:

$$N\left(t_{Pot} + t_{SM}\ N\frac{N+1}{2}\right),$$

onde, t_{Pot} :=tempo que leva a calcular a potência 0.5^k e t_{SM} :=tempo que leva a fazer duas subtrações, uma multiplicação e uma divisão. Suponhamos que temos um desenvolvimento do tipo (4.22), ou seja, o erro é dado por:

$$u^h - y = \sum_{j=1}^{pq} h^{j/q} v_j + O(h^{p+1}).$$

Este é um dos casos com que nos vamos preocupar e podemos aplicar o algoritmo-E fazendo em (4.5) a identificação

$$g_j(n) = h_n^{j/q},$$

onde tal como anteriormente, vamos considerar $h_n = h_0/2^n$. Neste caso temos o seguinte pseudo-código:

$$\begin{aligned}
&loop\ n = 0, N \\
&\quad E_0^{(n)} = S_n \\
&loop\ i = 1, N \\
&\quad g_{0,i}^{(n)} = h_n^{i/q} \\
&\quad loop\ k = 1, N - i \\
&\qquad g_{k,i}^{(n)} = g_{k-1,i}^{(n)} - \frac{g_{k-1,i}^{(n+1)} - g_{k-1,i}^{(n)}}{g_{k-1,k}^{(n+1)} - g_{k-1,k}^{(n)}} \times g_{k-1,k}^{(n)} \\
&\qquad E_k^{(n)} = E_{k-1}^{(n)} - \frac{E_{k-1}^{(n+1)} - E_{k-1}^{(n)}}{g_{k-1,k}^{(n+1)} - g_{k-1,k}^{(n)}} \times g_{k-1,k}^{(n)}
\end{aligned}$$

4.9 Resultados

Obtiveram-se as aproximações da função $y(t) = t^\alpha$ com $\alpha = 4.5$ e $\mu = 1.5$ utilizando o método de Euler com $h = 0.1$, $h = 0.05$, ... Na tabela (4.1) podemos comparar os erros relativos com o método de Euler com e sem extrapolação de Richardson.

É interessante verificar que a extrapolação com apenas 2 pontos apresenta erros relativos da mesma ordem de grandeza que o método de Euler com passo $h = h_0/2^9$. A extrapolação com 8 pontos, que utiliza os resultados do método de Euler desde $h_0 = 0.1$ até $h = h_0/2^9$, apresenta erros relativos próximos da precisão da máquina (FORTRAN `double precision` permite cerca de 15 dígitos décimais exactos).

ponto	solução exacta	Euler $h = h_0 = 0.1$	Euler $h = h_0/2^9$	extrapolação (2 pontos)	extrapolação (8 pontos)
0.2	7.155×10^{-4}	1.507×10^{-1}	5.486×10^{-3}	5.501×10^{-2}	9.394×10^{-15}
0.4	1.619×10^{-2}	1.029×10^{-1}	2.744×10^{-3}	1.738×10^{-2}	2.142×10^{-16}
0.6	1.003×10^{-1}	7.612×10^{-2}	1.830×10^{-3}	8.351×10^{-3}	0.000
0.8	3.663×10^{-1}	6.014×10^{-2}	1.372×10^{-3}	4.885×10^{-3}	6.060×10^{-16}
1.0	1.000	4.964×10^{-2}	1.098×10^{-3}	3.200×10^{-3}	1.110×10^{-16}
1.2	2.271	4.223×10^{-2}	9.153×10^{-4}	2.257×10^{-3}	1.564×10^{-15}
1.4	4.545	3.674×10^{-2}	7.846×10^{-4}	1.677×10^{-3}	7.815×10^{-16}
1.6	8.289	3.251×10^{-2}	6.865×10^{-4}	1.295×10^{-3}	4.285×10^{-16}
1.8	14.084	2.915×10^{-2}	6.102×10^{-4}	1.029×10^{-3}	1.513×10^{-15}
2.0	22.627	2.642×10^{-2}	5.492×10^{-4}	8.386×10^{-4}	1.413×10^{-15}

Tabela 4.1: Comparação dos erros relativos com e sem extrapolação de Richardson para $y(t) = t^\alpha$ com $\alpha = 4.5$ e $\mu = 1.5$.

Observa-se aceleração de convergência com o método de Richardson, até à quarta coluna (ver figuras 4.3 e 4.4 e tabelas 4.2 e 4.4), o que leva a suspeitar que o erro tenha um desenvolvimento assimptótico do tipo

$$u^h - y = v_1 h + v_2 h^2 + \ldots + v_m h^m + O(h^{m+1}), \quad m = 1, 2, 3.$$

Na figura 4.1 podemos ver a evolução do erro relativo à medida que aumentamos o número de pontos a extrapolar, no caso de $y(t) = t^\alpha$ com $\alpha = 6.5$ e $\mu = 1.5$. Na comparação do desempenho, em relação ao mesmo exemplo (ver figura 4.2),

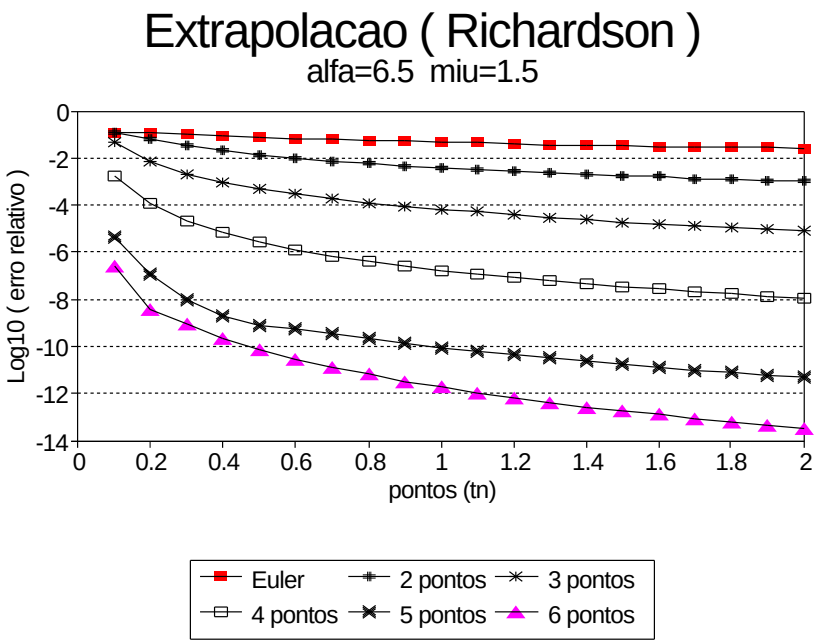

Figura 4.1: Erro do algoritmo de Richardson com o aumento do número de pontos de extrapolação.

verifica-se que o método de Richardson, embora comece em erros relativos grandes, apresenta um aumento de precisão muito superior aos métodos de colocação.

Na tabela 4.4 encontram-se as ordens de convergência aproximadas correspondentes a cada coluna, obtidas através do método de Richardson, para o exemplo $y(t) = t^{\alpha}$ com $\alpha = 4.5$ e $\mu = 1.5$, para o ponto $t_N = 2.0$. Observa-se que a ordem de convergência aumenta de coluna para coluna. Isto significa que de facto existe aceleração de convergência. Partimos de uma sucessão de valores aproximados e aplicámos transformações sucessivas. Após a primeira transformação obtemos uma sucessão que converge mais rapidamente que a anterior e assim sucessivamente; note-se que a ordem de convergência para a primeira coluna é 1, para a segunda é 2, para a terceira é 3 e para a quarta é 4.

Quando, através da aceleração de convergência atingimos 15 algarismos significativos (ver tabela 4.3), isto corresponde aos erros estarem próximos da unidade de arredondamento da máquina[1], logo é de esperar que não se consiga continuar a melhorar as aproximações. É por este facto que observamos, por exemplo, ordens nulas em alguns pontos da tabela 4.4.

[1] a mantissa é composta de 52 dígitos binários a que corresponde 15.65 algarismos significativos. Este valor nunca chega a ser atingido pois o último dígito binário resulta de arredondamentos das operações intermédias.

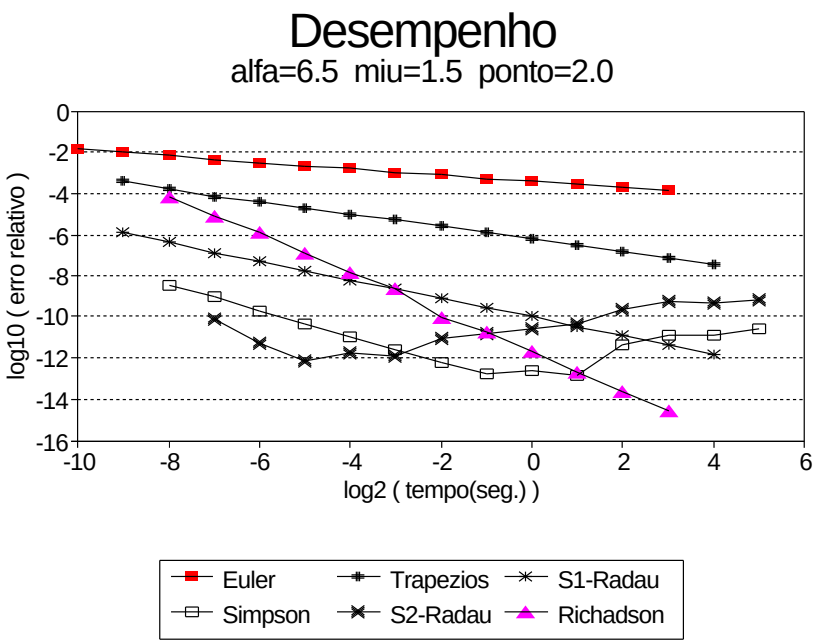

Figura 4.2: Comparação do desempenho da extrapolação de Richardson face aos métodos de colocação em $\tilde{S}_r^d(Z_N)$.

Na figura 4.5 apresentam-se os resultados obtidos, para o exemplo $y(t) = t^\alpha$ com $\alpha = 1$ e $\mu = 1.5$, para o ponto $t_N = 2.0$, através da aceleração de convergência utilizando os métodos:

Richardson: os resultados correspondentes a este método são designados por `richardson`.

algoritmo-E: nas experiências feitas com este método consideraram-se várias hipóteses de desenvolvimento assimptótico do erro:

hipótese	legenda gráfica
$e(t_N) = v_1 h + v_{1.5} h^{1.5} + v_2 h^2 + v_{2.5} h^{2.5} + \ldots$	alg-E(1.5)
$e(t_N) = v_1 h + v_2 h^2 + v_3 h^3 + \ldots$	alg-E(2)
$e(t_N) = v_1 h + v_{3.5} h^{3.5} + v_4 h^4 + \ldots$	alg-E(3.5)

Os resultados obtidos sugerem que o desenvolvimento assimptótico do erro, no ponto 2.0, é do tipo

$$e(t_N) = v_1 h + v_{1.5} h^{1.5} + v_2 h^2 + v_{2.5} h^{2.5} + v_3 h^3 + v_{3.5} h^{3.5} + \ldots$$

Observe-se que a convergência é acelerada significativamente quando incluímos todas as potências de h (inteiras e não inteiras). Em qualquer dos métodos

passo h	1^a coluna	2^a coluna	3^a coluna	4^a coluna
0.1	0.02642108			
0.05	0.013629849	0.000838618		
0.025	0.006922241	0.000214633	6.6384×10^{-6}	
0.0125	0.003488266	0.000054291	8.4414×10^{-7}	1.6400×10^{-8}
0.00625	0.001750959	0.000013653	1.0641×10^{-7}	1.0180×10^{-9}
0.003125	0.000877191	3.4232×10^{-6}	1.3356×10^{-8}	6.3123×10^{-11}
0.0015625	0.000439024	8.5705×10^{-7}	1.6730×10^{-9}	3.9209×10^{-12}
0.00078125	0.000219619	2.1442×10^{-7}	2.0934×10^{-10}	2.4399×10^{-13}
0.000390625	0.000109836	5.3625×10^{-8}	2.6182×10^{-11}	1.4177×10^{-14}
0.0001953125	0.000054925	1.3409×10^{-8}	3.2757×10^{-12}	9.1837×10^{-16}

Tabela 4.2: Erros relativos correspondentes às aproximações obtidas, para o ponto 2.0, pelo algoritmo de Richardson, considerando t^α com $\alpha = 4.5$ e $\mu = 1.5$.

a melhoria é a mesma da primeira coluna para a segunda. Isto sugere que deve existir um desenvolvimento assimptótico do erro com um termo principal na ordem de h, o qual é eliminado pela primeira transformada em cada um dos métodos de extrapolação. Deve existir um termo em $h^{1.5}$ pois só quando este é considerado é que conseguimos melhorar as aproximações; verifica-se o aumento de algarismos significativos de 6 para 9, da segunda coluna para a terceira.

Em relação aos termos de ordem superior ou igual a h^4 não se pode tirar conclusões a partir dos resultados numéricos, visto que, a ordem de grandeza desses termos é próxima da unidade de arredondamento do sistema numérico utilizado (aproximadamente 15 algarismos significativos).

1^a coluna	2^a coluna	3^a coluna	4^a coluna	5^a coluna	6^a coluna	7^a coluna	8^a coluna	9^a coluna	10^a coluna
1.6									
1.9	3.1								
2.2	3.7	5.2							
2.5	4.3	6.1	7.8						
2.8	4.9	7.1	9.0	11.1					
3.1	5.5	7.9	10.2	12.3	12.5				
3.4	6.1	8.8	11.4	13.6	13.9	14.5			
3.7	6.7	9.7	12.6	14.5	15.0	15.0	15.0		
4.0	7.3	10.6	13.8	15.0	15.0	15.0	15.0	15.0	
4.3	7.9	11.5	15.0	15.0	15.0	15.0	15.0	15.0	15.0

Tabela 4.3: Algarismos significativos no ponto 2.0 ($\alpha = 4.5$ e $\mu = 1.5$).

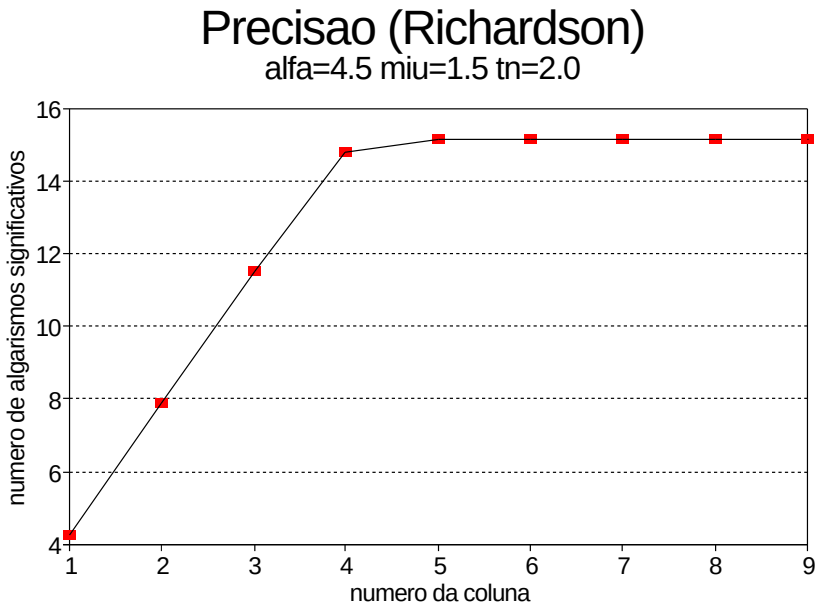

Figura 4.3: Aumento do número de algarismos significativos, por colunas, no algoritmo de Richardson.

1^a coluna	2^a coluna	3^a coluna	4^a coluna	5^a coluna	6^a coluna	7^a coluna	8^a coluna	9^a coluna
0.955								
0.977	1.966							
0.989	1.983	2.975						
0.994	1.992	2.988	4.010					
0.997	1.996	2.994	4.012	3.780				
0.999	1.998	2.997	4.009	4.394	4.684			
1.000	2.000	2.999	4.006	2.870	3.748	1.931		
1.000	2.000	2.999	4.105	1.931	0	0	0	
1.000	2.000	2.999	3.948	0	0	0	0	0

Tabela 4.4: Ordem de convergência considerando $h_0 = 0.1$, $\alpha = 4.5$, $\mu = 1.5$ e ponto 2.0.

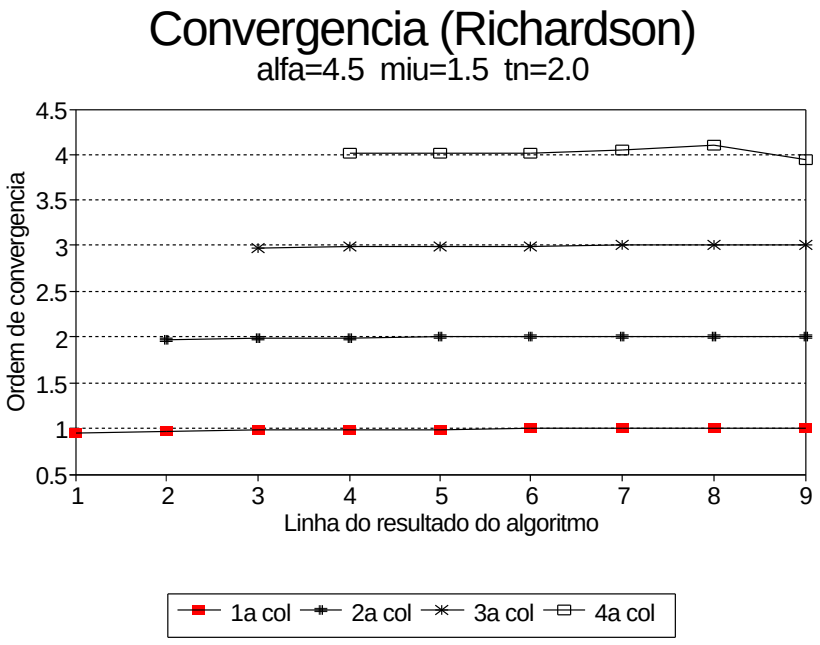

Figura 4.4: Variação da ordem de convergência de coluna para coluna no algoritmo de Richardson, no ponto 2.0.

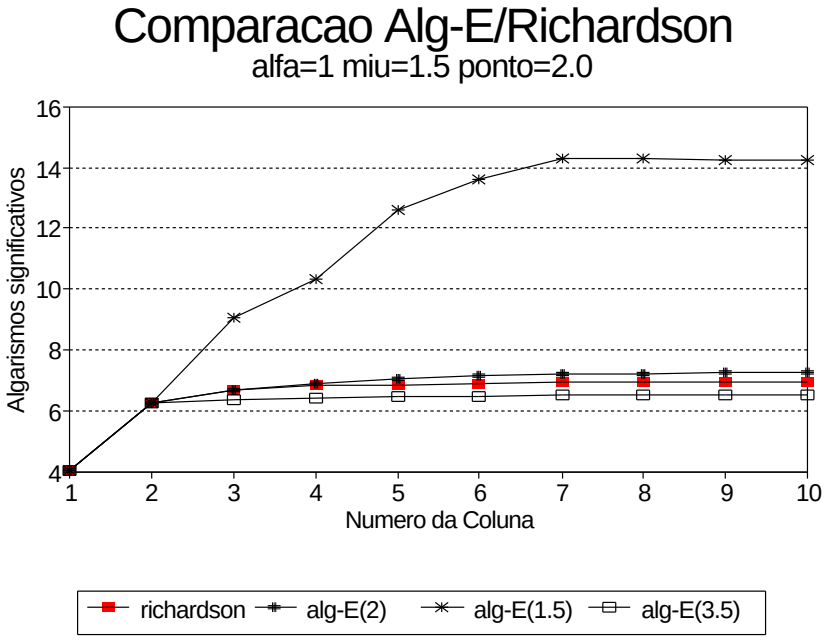

Figura 4.5: Comparação entre o algoritmo-E e o método de Richardson.

Capítulo 5

Conclusões

Neste trabalho considerou-se uma equação integral de Volterra, cujo núcleo $p_2(t,s)$ definido por (3.8) é singular no domínio $\{(t,s), 0 < s < t \leq T\}$ e tem a seguinte propriedade: o integral de $p_2(t,s)$ de 0 a t em relação a s é igual a uma constante não nula, para todo o t positivo. Consequentemente, em contraste com a maioria das equações singulares estudadas numericamente, não se verifica

$$\lim_{t\to 0} |\int_0^t p_2(t,s)ds| = 0.$$

Este facto dá origem a dificuldades na análise de convergência.

No capítulo 3, provou-se existência e unicidade de solução da equação (sob certas condições de regularidade), para certos valores de um parâmetro do núcleo. Foi também feito um estudo numérico da equação (3.7)-(3.9). Em particular, estudaram-se métodos de colocação baseados em aproximação por funções splines polinomiais seccionalmente constantes (método de Euler, método à direita e método do ponto médio) e provou-se convergência de ordem um. Estudou-se ainda o método dos trapézios em que a solução aproximada é uma função spline seccionalmente linear (contínua no intervalo $[0,T]$) e provou-se que esse método tem

ordem igual a dois. Implementou-se também o caso em que a aproximação linear é uma função descontínua, mas não se obteve prova teórica de convergência. Apresentaram-se vários resultados numéricos, confirmando as ordens teóricas obtidas. Esses resultados também sugerem que, para certas escolhas dos parâmetros de colocação, haja casos de superconvergência local, de acordo com o teorema 2.10. Investigou-se, numericamente, a aplicação de métodos de colocação usando splines quadráticos (de que é caso particular o chamado método de Simpson, em que a aproximação é uma função contínua no intervalo $[0, T]$). Para estes métodos não foi obtida prova de convergência teórica mas os resultados numéricos encorajam estudos futuros de métodos de colocação com splines de grau arbitrário m. Conjecturou-se que a ordem de convergência seja nesse caso igual a m. Contudo, devido à propriedade do núcleo da equação referida acima, a análise de convergência não é óbvia.

Este capítulo é finalizado com uma comparação dos diversos métodos estudados, aplicando os métodos e comparando os erros obtidos face ao tempo computacional por eles exigido. Conclui-se que o aumento do grau das splines polinomiais aumenta significativamente a precisão das aproximações, mas o aumento da complexidade dos cálculos introduz erros de arredondamento que vêm a comprometer os resultados. Por outro lado, a redução do passo não produz melhoras tão significativas como o aumento do grau das splines, mas permite obter precisões semelhantes com maior tempo computacional mas menor esforço na dedução e implementação dos métodos.

No capítulo 4, obtiveram-se resultados interessantes relativos à aceleração de convergência do método de Euler, por meio de técnicas de extrapolação. Verificou-se que, quando a solução da equação satisfaz certas condições de regularidade, e para certos valores do parâmetro μ do núcleo, a extrapolação de Richardson permite elevar a precisão dos resultados numéricos praticamente até à unidade de arredondamento do sistema de cálculo. Noutros casos, em que a extrapolação de Richardson não se revelou eficaz para acelerar a convergência, verificou-se que esse objectivo pode ser alcançado com a aplicação de outro método de extrapolação, o algoritmo-E.

Uma vez que o desempenho dos métodos de extrapolação está dependente da existência e da forma de desenvolvimentos assimptóticos do erro, neste trabalho dedicámo-nos a este estudo e obtivemos um resultado de existência de desenvolvimento assimptótico do erro em potências não inteiras do passo h, sob certas condições de regularidade. Obteve-se, ainda, um resultado teórico sobre a estabilidade para o método de Euler.

Os resultados obtidos neste capítulo encorajam a aplicação de técnicas de extrapolação a problemas do tipo considerado. Estas técnicas, associadas a mé-

todos de ordem baixa, permitem obter altos níveis de eficácia, podendo superar neste aspecto, os métodos de colocação baseados em splines de grau superior a 1. A aceleração de convergência permite obter ordens de convergência muito elevadas, pelo que com métodos simples podemos obter resultados muito exactos. A aceleração de convergência tem ainda a vantagem de permitir calcular simultaneamente as várias aproximações necessárias.

Assim, sempre que fôr possível aplicar aceleração de convergência é preferível devido ao reduzido tempo de computação, facilidade de implementação e simplicidade na dedução das expressões. Quando não é possível aplicar aceleração de convergência deve-se recorrer a métodos pouco complexos e que conduzam a expressões explícitas, a menos que a fraca capacidade de cálculo disponível seja uma limitação séria.

5.1 Trabalho Futuro

Alguns dos métodos aqui aplicados carecem da necessária base teórica. No entanto, e numa primeira fase era útil averiguar se determinados métodos ao serem aplicados apresentavam ordens de convergência aceitáveis. Só neste caso merece a pena dispender esforço numa certificação teórica; se os resultados forem desanimadores, qualquer que seja a certificação teórica encontrada, nunca seria demonstrável que seria possível produzir resultados melhores que os obtidos na prática. Assim, após os resultados aqui obtidos pode ser útil obter certificações teóricas para alguns métodos utilizados, com o objectivo de obter cobertura teórica para um maior número de casos que possam vir a surgir na realidade.

Os resultados numéricos obtidos sugerem que, para o problema estudado nesta tese e no caso do método de Euler, exista um desenvolvimento assimptótico do erro, em potências não inteiras de h. A existência de desenvolvimentos deste tipo é de grande interesse visto que eles permitem a utilização do algoritmo-E para podermos acelerar a convergência. Por conseguinte, tem interesse procurar obter uma certificação teórica para estes resultados numéricos.

Apêndice A

Programas

Neste apêndice incluem-se os ficheiros fonte dos programas referidos neste trabalho.

O programa `METODO` inclui os vários métodos de colocação, enquanto o programa `EXTRAP` inclui o algoritmo de extrapolação de Richardson. O programa `EALGOR` foi adaptado de um pacote de rotinas de extrapolação, apresentado por Brezinski [3]; apenas o programa principal foi reescrito, as rotinas `EALGO` e `GIKREC` não foram alteradas. A resolução dos sistemas de equações lineares utiliza o método de Crout com pesquisa parcial de pivot com equilibragem [10].

Os programas estão escritos em `FORTRAN 77` para maior portabilidade.

```
*---------------------------------------------------------------------*
      PROGRAM METODO
*---------------------------------------------------------------------*
*
*       Metodos:
*         0 - Exacta
*         1 - Euler
*         2 - Ponto Medio
*         3 - A Direita
*         4 - Trapezios
*         5 - S1 com pontos de Radau II
*         6 - S1 com pontos de Gauss
*         7 - Simpson
*         8 - S2 com pontos de Radau II
*         9 - S2 com pontos de Gauss
*
*       Casos:
*         1 - t**alfa
*         2 - t**alfa * log(t)
*
*       Ficheiro de dados
*          ALFA    MIU     B       H       CASO    METODO  LEVEL
*
*         ALFA parametro da solucao exacta Y(.) e G2(.)
*         MIU parametro da parte nao regular do nucleo
*   B extremo superior do intervalo [0,B]
*   H passo
*   CASO numero da funcao exacta a estudar (ver acima)
*   METODO numero do metodo a aplicar (ver acima)
*   LEVEL numero de reducoes do passo a executar H=H/2
*
*     Constantes:
*         NDIM numero maximo de pontos da rede
*         MAXLEV numero maximo de reducoes do passo
*     Variaveis:
*         U(0:NDIM) resultados (aproximacoes nos pontos da rede)
*         NS numero de pontos da rede
*         E(0:NDIM) erro absoluto relativo ao passo anterior (2*H)
*         O(0:NDIM) ordem aproximada de convergencia
*
*---------------------------------------------------------------------*

      PARAMETER (NDIM=50000,MAXLEV=10)

      DOUBLE PRECISION U(0:NDIM), ALFA, MIU, B, H, C0, C1, G2
      INTEGER CASO, METODO, NS
      DOUBLE PRECISION E(0:NDIM), O(0:NDIM)
```

```
      CHARACTER *20 MYFILE
      REAL DTIME, DELTA, TARRY(2)
      EXTERNAL DTIME

      WRITE(*,*) ' Nome do ficheiro de dados'
      READ(*,100) MYFILE
100   FORMAT(A20)
      OPEN(UNIT=1, FILE=MYFILE, STATUS='OLD')

      WRITE(*,*) ' Nome do ficheiro de resultados'
      READ(*,100) MYFILE
      OPEN(UNIT=2, FILE=MYFILE, STATUS='NEW')

      READ(1,*)
200   READ(1,*,END=700) ALFA, MIU, B, H, CASO, METODO, LEVEL

      IF (CASO .GE. 10 .OR. CASO .LT. 0) GOTO 200
      IF (LEVEL .LT. 1) LEVEL=1
      IF (LEVEL .GE. MAXLEV) LEVEL=MAXLEV

*         Executar o metodo tantas vezes quantas
*            o numero de reducoes do passo
      DO 600 J=1, LEVEL
        WRITE(2,300) ALFA, MIU, B, H, CASO, METODO
300     FORMAT(' Alfa=', D10.3,' Miu=',   D10.3, ' [0,', D8.2, ']',
     *         ' H=',    D10.3,' Caso=',  I1,    ' Metodo ',I2)

        NS = B / H
        U(0) = MIU/(MIU-1) * G2(CASO, ALFA, MIU, 0D0)

        DELTA=DTIME(TARRY)
        GOTO (1,2,3,4,5,6,7,8,9,10), METODO+1
1         CALL EXACTA (CASO, ALFA, H, NS, U)
        GOTO 500
2         CALL EULER (CASO, ALFA, MIU, H, NS, U)
        GOTO 500
3         CALL MEDIO (CASO, ALFA, MIU, H, NS, U)
        GOTO 500
4          CALL DIRTA (CASO, ALFA, MIU, H, NS, U)
        GOTO 500
5          CALL TRAPEZ (CASO, ALFA, MIU, H, NS, U)
        GOTO 500
6          CALL COLS1 (CASO, ALFA, MIU, H, NS, U, 1/3D0, 1D0)
        GOTO 500
7          CALL COLS1 (CASO, ALFA, MIU, H, NS, U, 0.5D0, 1D0)
        GOTO 500
8          CALL SIMPS (CASO, ALFA, MIU, H, NS, U)
```

```
        GOTO 500
9         C0 = (4 - DSQRT(6D0)) / 10D0
          C1 = (4 + DSQRT(6D0)) / 10D0
          CALL COLS2 (CASO, ALFA, MIU, H, NS, U, C0, C1, 1D0)
        GOTO 500
10        C0 = (3 - DSQRT(3D0)) / 6D0
          C1 = (3 + DSQRT(3D0)) / 6D0
          CALL COLS2 (CASO, ALFA, MIU, H, NS, U, C0, C1, 1D0)

500     CALL RESULT(CASO, ALFA, H, U, E, O, NS, J)

        H=H/2D0
600   CONTINUE

      GOTO 200

700   STOP
      END

*---------------------------------------------------------------------*
      SUBROUTINE RESULT(CASO, ALFA, H, U, E, O, NS, LEV)
*
*     Imprime os resultados da aplicacao dos metodos.
*
*---------------------------------------------------------------------*
      DOUBLE PRECISION U(0:NS), E(0:NS), O(0:NS), ALFA, H, Y
      DOUBLE PRECISION LOG2, SDV, AVG, EMAX, ERR, SOL
      INTEGER CASO, NS, LEV
      REAL DELTA, TARRY(2), DTIME
      EXTERNAL DTIME

      DELTA=DTIME(TARRY)
      LOG2=DLOG(2D0)

      M=1
      AVG=0D0
      DO 200 N=2**(LEV-1), NS, 2**(LEV-1)
        SOL=Y(CASO, ALFA, N*H)
        ERR=DABS(U(N) - SOL)
        IF (LEV .GT. 1) THEN
          O(M)=DLOG(E(M) / ERR) / LOG2
          AVG=AVG+O(M)
        ELSE
          O(M)=0D0
        ENDIF
        WRITE(2,100) N*H, U(N), DABS(ERR/SOL), O(M)
C       WRITE(2,100) N*H, U(N), ERR, O(M)
```

```
100     FORMAT(D12.4, ' ', D25.19, ' ', D25.19, ' ', D12.4)
        E(M)=ERR
        M=M+1
200   CONTINUE

      AVG=AVG/(M-1)
      SDV=0D0
      DO 300 N=1, M
        SDV=SDV+(O(N)-AVG)**2
300   CONTINUE

      EMAX=0D0
      DO 400 N=1, NS
        ERR=DABS(U(N) - Y(CASO, ALFA, N*H))
        IF (EMAX .LT. ERR) EMAX=ERR
400   CONTINUE

      WRITE(2,500) EMAX, DELTA, AVG, SDV/(M-2), TARRY(1), TARRY(2)
500   FORMAT('   Emax=', D25.19, ' Time=', I19,/,
     *       '   Media=', D25.19, ' Variancia=', D25.19,/,
     *       '   Elapsed=', D25.19, ' , ', D25.19)

      RETURN
      END

*-----------------------------------------------------------------*
      DOUBLE PRECISION FUNCTION G2 (CASO, ALFA, MIU, T)
*
*     Segundo membro da equacao integral.
*
*-----------------------------------------------------------------*

      DOUBLE PRECISION ALFA, MIU, T
      INTEGER CASO

      IF (CASO .EQ. 1) THEN
        G2=T**ALFA*(1D0-1D0/(MIU+ALFA))
      ELSE
        IF (T .EQ. 0D0) THEN
          G2=0D0
        ELSE
          G2=T**ALFA*(DLOG(T)-1D0/(MIU+ALFA)*(DLOG(T)-1D0/(MIU+ALFA)))
        ENDIF
      ENDIF

      RETURN
      END
```

```
*---------------------------------------------------------------------*
      DOUBLE PRECISION FUNCTION Y (CASO, ALFA, T)
*
*     Solucao exacta.
*
*---------------------------------------------------------------------*

      DOUBLE PRECISION T, ALFA
      INTEGER CASO

      IF (CASO .EQ. 1) THEN
        Y=T**ALFA
      ELSE
        IF (T .EQ. 0D0) THEN
          Y=0D0
        ELSE
          Y=T**ALFA * DLOG(T)
        ENDIF
      ENDIF

      RETURN
      END

*---------------------------------------------------------------------*
      SUBROUTINE EXACTA (CASO, ALFA, H, NS, U)
*
*     Calculo da solucao exacta.
*
*---------------------------------------------------------------------*

      DOUBLE PRECISION U(0:NS), ALFA, H, Y
      INTEGER CASO, NS

      DO 100 N=0, NS
         U(N)=Y(CASO, ALFA, N*H)
100   CONTINUE

      RETURN
      END

*---------------------------------------------------------------------*
      SUBROUTINE EULER (CASO, ALFA, MIU, H, NS, U)
*
*     Aplicacao do metodo de Euler.
*
```

```
*--------------------------------------------------------------------*

      DOUBLE PRECISION U(0:NS), ALFA, MIU, H, G2, S
      INTEGER CASO, NS

      DO 200 N=1, NS
         S=0D0
         DO 100 I=0, N-1
            S=S+ ((I+1)**MIU-I**MIU)*U(I)
100         CONTINUE
         U(N)=(1D0/(MIU*N**MIU))*S + G2(CASO, ALFA, MIU, N*H)
200   CONTINUE

      RETURN
      END

*--------------------------------------------------------------------*
      SUBROUTINE MEDIO (CASO, ALFA, MIU, H, NS, U)
*
*     Aplicacao do metodo do Ponto Medio.
*
*--------------------------------------------------------------------*

      DOUBLE PRECISION U(0:NS), ALFA, MIU, H, G2, S
      INTEGER CASO, NS

      U(0) = MIU / (MIU - 1) * G2(CASO, ALFA, MIU, H/2D0)
      DO 200 N=1, NS
         S=0D0
         DO 100 I=0, N-1
            S=S+ ((I+1)**MIU-I**MIU)*U(I)
100         CONTINUE
         U(N)=(1D0 / ((N + 0.5D0)**MIU * (MIU-1) + N**MIU)) *
     *        (MIU * (N + 0.5D0)**MIU *
     *         G2(CASO, ALFA, MIU, (N+0.5D0)*H) + S)
200    CONTINUE

      RETURN
      END

*--------------------------------------------------------------------*
      SUBROUTINE DIRTA (CASO, ALFA, MIU, H, NS, U)
*
*     Aplicacao do metodo 'a direita.
*
*--------------------------------------------------------------------*
```

```
      DOUBLE PRECISION U(0:NS), ALFA, MIU, H, G2, S
      INTEGER CASO, NS

      DO 200 N=1, NS
         S=0D0
         DO 100 I=1, N-1
            S=S+ (I**MIU-(I-1)**MIU)*U(I)
100         CONTINUE
         U(N)=(1D0/((MIU-1)*N**MIU+(N-1)**MIU)) * (S + MIU * N**MIU
     *              * G2(CASO, ALFA, MIU, N*H))
200   CONTINUE

      RETURN
      END

*---------------------------------------------------------------------*
      SUBROUTINE TRAPEZ (CASO, ALFA, MIU, H, NS, U)
*
*     Aplicacao do metodo dos Trapezios.
*
*---------------------------------------------------------------------*

      DOUBLE PRECISION U(0:NS), ALFA, MIU, H, G2, S
      INTEGER CASO, NS

      DO 200 N=1, NS

         S=0D0
         DO 100 I=1,N-1
            S=S+ ((I-1)**(MIU+1)-2D0*I**(MIU+1)+(I+1)**(MIU+1)) *U(I)
100         CONTINUE

         U(N)=1D0/(N**(MIU+1)+(MIU**2-1)*N**MIU-(N-1)**(MIU+1))
     *        *(U(0)+S+N**MIU*MIU*(MIU+1) * G2(CASO, ALFA, MIU, N*H))
200   CONTINUE

      RETURN
      END

*---------------------------------------------------------------------*
      SUBROUTINE COLS1 (CASO, ALFA, MIU, H, NS, U, C0, C1)
*                                        -1
*     Aplicacao do metodo de colocacao em S    (caso geral).
*                                        0
*---------------------------------------------------------------------*

      DOUBLE PRECISION U(0:NS), ALFA, MIU, H, C0, C1, G2, S
```

```
      INTEGER CASO, NS
      DOUBLE PRECISION A(2,2), B(2), D, U0(0:50000)
      DOUBLE PRECISION NM, NM1, NC0M, NC0M1, NC1M, NC1M1

      DO 200 N=0, NS-1

*        Matriz A        *

         NM = N**MIU
         NM1 = N**(MIU+1)
         NC0M = (N+C0)**MIU
         NC0M1 = (N+C0)**(MIU+1)
         NC1M = (N+C1)**MIU
         NC1M1 = (N+C1)**(MIU+1)

         A(1,1)=NC0M*(C0-C1)+(N+C1)*(NC0M-NM)/MIU-
     *          (NC0M1-NM1)/(MIU+1.0D0)
         A(1,2)=(NC0M1-NM1)/(MIU+1.D0)-(N+C0)*(NC0M-NM)/MIU
         A(2,1)=(N+C1)*(NC1M-NM)/MIU-(NC1M1-NM1)/(MIU+1.D0)
         A(2,2)=NC1M*(C0-C1)-(N+C0)*(NC1M-NM)/MIU+
     *          (NC1M1-NM1)/(MIU+1.D0)

*        Vector B        *

         S=0.0D0
         DO 100 I=0,N-1
            S=S+( (I+1.D0)**(MIU+1.D0)-I**(MIU+1.D0) )/(MIU+1.D0)*
     *        ( U0(I)-U(I+1) )+
     *        ( (I+1.D0)**MIU-I**MIU )/MIU*
     *        ( U(I+1)*(I+C0)-U0(I)*(I+C1) )
100         CONTINUE
         B(1)=(C0-C1)*NC0M * G2(CASO, ALFA, MIU, (N+C0)*H)+S
         B(2)=(C0-C1)*NC1M * G2(CASO, ALFA, MIU, (N+C1)*H)+S

*        Resolucao do Sistema Au = B        *

         D=A(1,1)*A(2,2)-A(1,2)*A(2,1)
         U0(N)=(B(1)*A(2,2)-B(2)*A(1,2))/D
         U(N+1)=(B(2)*A(1,1)-B(1)*A(2,1))/D

200   CONTINUE

      RETURN
      END

*-----------------------------------------------------------------*
      SUBROUTINE SIMPS (CASO, ALFA, MIU, H, NS, U)
```

```
*
*     Aplicacao do metodo de Simpson.
*
*-----------------------------------------------------------------*

      DOUBLE PRECISION U(0:NS), ALFA, MIU, H, G2, S
      INTEGER CASO, NS
      DOUBLE PRECISION U1(0:50000), U2(0:50000), INTEGR(0:1,0:2)
      DOUBLE PRECISION A(2,2), B(2), D

      DO 200 N=0, NS-1

*        Matriz A        *

         CALL SIMPSI(N, MIU, INTEGR)
         A(1,1)=1D0-INTEGR(0,1)
         A(1,2)=   -INTEGR(0,2)
         A(2,1)=   -INTEGR(1,1)
         A(2,2)=1D0-INTEGR(1,2)

*        Vector B        *

         S=0D0
         DO 100 I=0, N-1
            S=S+((I+1)**(MIU+2D0)-I**(MIU+2D0))/(MIU+2D0)*
     *       (U(I)-2*U1(I)+U2(I))+
     *       ((I+1)**(MIU+1D0)-I**(MIU+1D0))/(MIU+1D0)*
     *       (2D0*(2D0*I+1)*U1(I)-(2D0*I+1.5D0)*U(I)-
     *       (2D0*I+0.5D0)*U2(I))+
     *       ((I+1)**MIU-I**MIU)/MIU*
     *       ((I+1)*(I+0.5D0)*U(I)-
     *       2D0*I*(I+1)*U1(I)+I*(I+0.5D0)*U2(I))
100      CONTINUE
         B(1)=G2(CASO, ALFA, MIU, (N+0.5D0)*H)+INTEGR(0,0)*U(N)+
     *        2D0*S/(N+0.5D0)**MIU
         B(2)=G2(CASO, ALFA, MIU, (N+1D0)*H)+INTEGR(1,0)*U(N)+
     *        2D0*S/(N+1)**MIU

*        Resolucao do sistema Au=b        *

         D=A(1,1)*A(2,2)-A(1,2)*A(2,1)
         U1(N)=(B(1)*A(2,2)-B(2)*A(1,2))/D
         U2(N)=(B(2)*A(1,1)-B(1)*A(2,1))/D
         U(N+1)=U2(N)

200   CONTINUE
```

```
      RETURN
      END

*-----------------------------------------------------------------*
      SUBROUTINE SIMPSI(N, MIU, I)
*
*     Calculo auxiliar dos integrais necessarios ‘a execucao do
*        metodo de Simpson.
*
*-----------------------------------------------------------------*

      DOUBLE PRECISION MIU, I(0:1,0:2)
      DOUBLE PRECISION N0, N1, N2, K0, K1, K2, Z1, Z2, Z3

      N0 = N**MIU
      N1 = N**(MIU+1)
      N2 = N**(MIU+2)

      K0 = (N+0.5D0)**MIU
      K1 = (N+0.5D0)**(MIU+1)
      K2 = (N+0.5D0)**(MIU+2)

      Z1 = (K2 - N2) / (MIU + 2)
      Z2 = (K1 - N1) / (MIU + 1)
      Z3 = (K0 - N0) / MIU
      I(0,0) = 2D0/K0 * (Z1-Z2*(2*N+1.5D0)+Z3*(N+1D0)*(N+0.5D0))
      I(0,1) = -4D0/K0 * (Z1-Z2*(2*N+1D0)+Z3*N*(N+1D0))
      I(0,2) = 2D0/K0 * (Z1-Z2*(2*N+0.5D0)+Z3*N*(N+0.5D0))

      K0 = (N+1D0)**MIU
      K1 = (N+1D0)**(MIU+1)
      K2 = (N+1D0)**(MIU+2)

      Z1 = (K2 - N2) / (MIU + 2)
      Z2 = (K1 - N1) / (MIU + 1)
      Z3 = (K0 - N0) / MIU
      I(1,0) = 2D0/K0 * (Z1-Z2*(2*N+1.5D0)+Z3*(N+1D0)*(N+0.5D0))
      I(1,1) = -4D0/K0 * (Z1-Z2*(2*N+1D0)+Z3*N*(N+1D0))
      I(1,2) = 2D0/K0 * (Z1-Z2*(2*N+0.5D0)+Z3*N*(N+0.5D0))

      RETURN
      END

*-----------------------------------------------------------------*
      SUBROUTINE COLS2 (CASO, ALFA, MIU, H, NS, U, C0, C1, C2)
*                                          -1
*     Aplicacao do metodo de colocacao em S    (caso geral).
```

```
*                                    2
*-------------------------------------------------------------*

      DOUBLE PRECISION U(0:NS), ALFA, MIU, H, C0, C1, C2, G2
      INTEGER CASO, NS

      DOUBLE PRECISION U0(0:50000), U1(0:50000), INTEGR(0:2)
      DOUBLE PRECISION A(3,3), B(3), X(3), SOMA
      DOUBLE PRECISION NC0, NC1, NC2

      DO 400 N=0, NS-1

*        Matriz A        *

         NC0 = (N+C0)**MIU
         CALL COLS2I(N, N+C0, INTEGR, C0, C1, C2, MIU)
         A(1,1)=1.0D0-INTEGR(0)/NC0
         A(1,2)=     -INTEGR(1)/NC0
         A(1,3)=     -INTEGR(2)/NC0

         NC1 = (N+C1)**MIU
         CALL COLS2I(N, N+C1, INTEGR, C0, C1, C2, MIU)
         A(2,1)=     -INTEGR(0)/NC1
         A(2,2)=1.0D0-INTEGR(1)/NC1
         A(2,3)=     -INTEGR(2)/NC1

         NC2 = (N+C2)**MIU
         CALL COLS2I(N, N+C2, INTEGR, C0, C1, C2, MIU)
         A(3,1)=     -INTEGR(0)/NC2
         A(3,2)=     -INTEGR(1)/NC2
         A(3,3)=1.0D0-INTEGR(2)/NC2

*        Vector B        *

         SOMA = 0.0D0
         DO 100 I=0, N-1
            CALL COLS2I(I, I+1D0, INTEGR, C0, C1, C2, MIU)
            SOMA=SOMA+U0(I)*INTEGR(0)+
     *                  U1(I)*INTEGR(1)+
     *                  U(I+1)*INTEGR(2)
100         CONTINUE

         B(1)=G2(CASO, ALFA, MIU, (N+C0)*H)+SOMA/NC0
         B(2)=G2(CASO, ALFA, MIU, (N+C1)*H)+SOMA/NC1
         B(3)=G2(CASO, ALFA, MIU, (N+C2)*H)+SOMA/NC2

*        Resolucao do sistema A*X=B        *
```

```
         CALL RESOLV(A, B, X, 3)

         U0(N)=X(1)
         U1(N)=X(2)
         U(N+1)=X(3)

400   CONTINUE

      RETURN
      END

*---------------------------------------------------------------------*
      SUBROUTINE COLS2I(N, K, I, C0, C1, C2, MIU)
*
*     Calculo auxiliar dos integrais necessarios ‘a execucao do
*                                -1
*        metodo de colocacao em S    .
*                                2
*---------------------------------------------------------------------*

      DOUBLE PRECISION K, I(0:2), C0, C1, C2, MIU
      DOUBLE PRECISION Z1, Z2, Z3

      Z1=(K**(MIU+2D0)-N**(MIU+2D0))/(MIU+2D0)
      Z2=(K**(MIU+1D0)-N**(MIU+1D0))/(MIU+1D0)
      Z3=(K**MIU-N**MIU)/MIU

      I(0) = (Z1-Z2*(2D0*N+C1+C2)+Z3*(N+C1)*(N+C2))/((C0-C1)*(C0-C2))
      I(1) = (Z1-Z2*(2D0*N+C0+C2)+Z3*(N+C0)*(N+C2))/((C1-C0)*(C1-C2))
      I(2) = (Z1-Z2*(2D0*N+C0+C1)+Z3*(N+C0)*(N+C1))/((C2-C0)*(C2-C1))

      RETURN
      END
*---------------------------------------------------------------------*
      SUBROUTINE RESOLV(A, B, X, 3)
*
*     Rotina de interface com as rotinas de resolucao dos
*        sistemas de equacoes lineares.
*
*---------------------------------------------------------------------*
      DOUBLE PRECISION A(N,N), B(N), D(50000), X(N)
      INTEGER IPIVOT(50000)

      CALL FACTOR(A,3,D,IPIVOT,IFLAG)
      CALL SUBST(A,IPIVOT,B,3,X)
```

```
RETURN
END
```

```
*---------------------------------------------------------------------*
      PROGRAM EXTRAP
*---------------------------------------------------------------------*
*
*       Metodo de Richardson
*
*       Ficheiro de dados
*         Um valor por linha.
*
*       Resultados
*         Linhas do triangulo de resultados separados por uma
*            linha em branco.
*
*       Constantes:
*         NDIM numero maximo de pontos.
*
*       Variaveis:
*   S(0:NDIM,0:NDIM) matriz de aceleracao.
*
*---------------------------------------------------------------------*

      PARAMETER (NDIM=50)

      DOUBLE PRECISION P, S(0:NDIM,0:NDIM)
      CHARACTER *20 MYFILE
      REAL DTIME, DELTA, TARRY(2)
      EXTERNAL DTIME

      WRITE(*,*) ' Nome do ficheiro de dados'
      READ(*,100) MYFILE
100   FORMAT(A20)
      OPEN(UNIT=1, FILE=MYFILE, STATUS='OLD')

      WRITE(*,*) ' Nome do ficheiro de resultados'
      READ(*,100) MYFILE
      OPEN(UNIT=2, FILE=MYFILE, STATUS='NEW')

      DO 200 I=0, NDIM
200   READ(1,*,END=300) S(0,I)

      DELTA=DTIME(TARRY)
300   DO 500 K=1, I
        P = 0.5D0**K
        DO 400 M=0, I-K
400       S(K,M) = (S(K-1,M+1) - P * S(K-1,M)) / (1 - P)
500   CONTINUE
      DELTA=DTIME(TARRY)
```

```
      WRITE (2,*) DELTA

      DO 800 M=0, I-1
        WRITE(2,*) M
        DO 700 K=0, M
700       WRITE(2,900) S(K,M-K)
        WRITE(2,*)
800   CONTINUE
900   FORMAT(D25.19)

      STOP
      END
```

```
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

      PROGRAM EALGOR

C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      PARAMETER  (MAXCOL=50)
      INTEGER  IER, INIT, K, NBC
      DOUBLE PRECISION  EINIT, S(0:MAXCOL), E(0:MAXCOL), S0
      DOUBLE PRECISION  G(MAXCOL*(MAXCOL+1)), GINIT(2*MAXCOL)
      REAL        ALPHAI(MAXCOL), BETAI(MAXCOL)

      READ(*,*) N0
      DO 12 NBC=0, MAXCOL
12      READ(*,*,END=15) ALPHAI(NBC+1), BETAI(NBC+1), S(NBC)

15    INIT = 0
      DO 30 K = 1, NBC
        EINIT=S(K-1)
        S0=EINIT
        CALL GIKREC(N0, K, NBC-1, GINIT, ALPHAI, BETAI)
        CALL EALGO (E, NBC-1, INIT, 1.0D-30, EINIT, GINIT, G, IER)
        IF (IER .NE. 0) GOTO 800
DO 40 L = 0, K - 1
40   WRITE(*,202) E(L)
        WRITE(*,201) K, S0, EINIT
201     FORMAT(I12,2(D26.19))
202     FORMAT(D28.20)
   30 CONTINUE

      GOTO 900
800   WRITE (*,*) 'STOP DUE TO IER =', IER
      IF (IER.EQ.100) WRITE(*,*)  'DIVISION BY ZERO'
900   STOP
      END

C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

      SUBROUTINE EALGO (E, MAXCOL, INIT, EPS, EINIT, GINIT, G, IER)

C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C   DESCRIPTION:
C
C     SCALAR SEQUENCE EXTRAPOLATION VIA THE E-ALGORITHM
C
C   USAGE:
C     CALL EALGO (E, MAXCOL, INIT, EPS, EINIT, GINIT, G, IER)
```

```
C
C   ARGUMENTS:
C
C     E   - OUTPUT REAL VECTOR. IN THE CALLING PROCEDURE THE
C           DIMENSION OF E MUST BE (0:IE), WITH IE GREATER OR
C           EQUAL TO MAXCOL. THIS VECTOR CONTAINS, AFTER THE
C           k-TH CALL THE LAST COMPUTED ROW OF THE E-ARRAY:
C
C                               (k-1)  (k-2)       (0)
C         - IF k <= MAXCOL+1   E    , E     ,..., E
C                               0      1           k-1
C
C                               (k-1)  (k-2)       (k-1-MAXCOL)
C         - IF k > MAXCOL+1    E    , E     ,..., E
C                               0      1           MAXCOL
C
C     MAXCOL - INPUT INTEGER GIVING THE INDEX OF THE LAST COLUMN
C              OF THE E-ARRAY THAT THE USER WANTS TO COMPUTE
C                                  (n)
C              (THE LOWER INDEX OF E   ).
C                                  k
C
C     INIT   - INPUT/OUTPUT INTEGER TO BE SET TO ZERO BEFORE THE
C              FIRST CALL OF THE SUBROUTINE. ITS VALUE IS CHANGED
C              TO 1 BY THE SUBROUTINE DURING THE FIRST CALL.
C              FOR A NEW APPLICATION OF THE ALGORITHM THE USER
C              MUST SET AGAIN INIT TO ZERO.
C
C     EPS    - INPUT REAL VALUE USED IN TESTS FOR ZERO.
C              IF |X| < EPS, THEN X IS CONSIDERED TO BE ZERO.
C
C     EINIT  - INPUT/OUTPUT REAL VALUE. IT MUST CONTAIN THE VALUE
C              OF S    BEFORE THE k-TH CALL OF THE SUBROUTINE.
C                  k-1
C              IT CONTAINS AFTER THE k-TH CALL OF THE SUBROUTINE
C
C                     (0)
C                    E            IF  k <= MAXCOL+1
C                     k-1
C
C                     (k-1-MAXCOL)
C                    E            IF  k >  MAXCOL+1
C                     MAXCOL
C
C              THE SUBROUTINE RETURNS THE SEQUENCE
C
C            (0)   (0)   (0)         (0)      (1)      (2)
```

```
C          E   , E   , E   , ... , E      , E      , E      , ...
C           0     1     2           MAXCOL   MAXCOL   MAXCOL
C
C     GINIT  - INPUT REAL VECTOR. IN THE CALLING PROCEDURE THE
C              DIMENSION OF GINIT MUST BE GREATER OR EQUAL TO
C              2*MAXCOL. BEFORE THE k-TH CALL OF THE SUBROUTINE
C              IT MUST CONTAIN
C                    IF k = 1   GINIT(1)= G (0)
C                                         1
C
C                    IF k = 2   GINIT(1)= G (1)
C                                         1
C
C                    IF 3 <= k <= MAXCOL+1
C
C                       GINIT=[G (k-1), G (k-1), ... , G (k-1),
C                              1        2              k-2
C
C                             G (0), G (1), ... , G (k-1)]
C                              k-1    k-1          k-1
C
C                    IF k > MAXCOL+1
C
C                       GINIT=[G (k-1), G (k-1), ... , G (k-1)]
C                              1        2              MAXCOL
C
C     G      - WORKING REAL VECTOR IN WHICH THE LAST ROW AND THE
C              LAST COLUMN OF EACH G  ARRAY ARE SAVED FOR THE
C                                   i
C              NEXT CALL OF THE SUBROUTINE. IN THE CALLING
C              PROCEDURE THE DIMENSION OF G MUST BE GREATER OR
C              EQUAL TO MAXCOL*(MAXCOL+1) AND TO
C              MAXCOL*NUMBER_OF_CALLS.
C
C     IER    - OUTPUT INDEX WARNING/ERROR.
C              IER = 100   DIVISION BY ZERO IN THE SUBROUTINE.
C              IER = 200   CALL OF THE SUBROUTINE WITH A NON ZERO
C                          IER VALUE.
C
C   REMARKS:
C
C     ALL THE REAL ARGUMENTS PASSED FROM THE CALLING PROCEDURE
C     MUST BE IN DOUBLE PRECISION AND THE VECTORS E AND G MUST
C     NOT BE MODIFIED BY THE USER BETWEEN TWO CONSECUTIVE CALLS
C     OF THE SUBROUTINE.
C                              (0)
C     TO OBTAIN THE VALUE OF E   (J <= MAXCOL), THE USER MUST CALL
```

```
C                           J
C     THIS SUBROUTINE (J+1) TIMES.
C     IN THE CALLING PROCEDURE THE MINIMAL DIMENSIONS FOR THE
C     VECTOR ARGUMENTS ARE:
C       E(0:MAXCOL)
C       GINIT(2*MAXCOL)
C       G(MAXDIM),MAXDIM=MAX(MAXCOL*(MAXCOL+1),MAXCOL*NUMBER_OF_CALLS)
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      INTEGER IER, INIT, MAXCOL
      DOUBLE PRECISION EINIT, EPS
      DOUBLE PRECISION E(0:1), G(1), GINIT(1)

      DOUBLE PRECISION R
      INTEGER I, IND, INDI, INDI1, INDJ, INDM, J
      SAVE INDM

C               ... FIRST CALL OF THE SUBROUTINE
      IF (INIT .EQ. 0) THEN
         IER    = 0
         INDM   = 0
         INIT   = 1
         E(0)   = EINIT
         G(1)   = GINIT(1)
         RETURN
      END IF

C               ... NEXT CALLS OF THE SUBROUTINE
      IF (IER .NE. 0) THEN
         IER = 200
         RETURN
      END IF

      IF (MAXCOL .EQ. 0) THEN
         E(0) = EINIT
         RETURN
      END IF

C               ... INDM REPRESENTS THE (NUMBER_OF_CALLS - 1)
      INDM = INDM+1

C               ... ADD AN ELEMENT TO G  ARRAY AT EACH CALL
      G(1+INDM*MAXCOL) = GINIT(1)
      IF (INDM .NE. 1) THEN

C               ... COMPUTE AND SAVE IN G THE NECESSARY ELEMENTS
C                   OF THE G ARRAYS (ONLY FROM THE THIRD CALL)
```

```
C                                  i
      IF (INDM .GE. 3) THEN

C            ... COMPUTE THE NEXT ROW OF ALL THE PREVIOUS G
C                                                          i
C               ARRAYS (i = 2 TO INDM-1 IF INDM <= MAXCOL+1)
C                      (i = 2 TO MAXCOL IF INDM > MAXCOL+1)
C               AND SAVE IN G THE NEW COMPUTED ELEMENTS
         IND = MIN0(INDM-1,MAXCOL)
         DO 20 I = 2, IND
            DO 10 J = 1, I-1
               INDI = I + (J-1) * MAXCOL
               INDJ = J + INDM * MAXCOL
               R    = G(INDJ-MAXCOL) - G(INDJ)
               IF (DABS(R) .LT. EPS) THEN
                  IER = 100
                  RETURN
               END IF
               R  = GINIT(I) - (G(INDI)-GINIT(I))/R * G(INDJ)
               G(INDI)  = GINIT(I)
               GINIT(I) = R
 10         CONTINUE
            G(I+INDM*MAXCOL) = R
 20      CONTINUE
      END IF
      IF (INDM .LE. MAXCOL) THEN
C            ... COMPUTE THE NEW G     ARRAY  IF INDM <= MAXCOL
C                                 INDM
C                AND SAVE IN G THE RELATIVE LAST ROW AND COLUMN
         G(INDM) = GINIT(INDM)
         DO 40 I = 1, INDM
            INDI1 = INDM + I
            IND   = MIN0(I,INDM-1)
            DO 30 J = 1, IND
               INDI = J + I * MAXCOL
               INDJ = INDM + (J-1) * MAXCOL
               R    = G(INDI-MAXCOL) - G(INDI)
               IF (DABS(R) .LT. EPS) THEN
                  IER = 100
                  RETURN
               END IF
               R  = GINIT(INDI1) - (G(INDJ)-GINIT(INDI1))/R
     *               * G(INDI)
               G(INDJ)      = GINIT(INDI1)
               GINIT(INDI1) = R
 30         CONTINUE
            G(INDM+I*MAXCOL) = R
```

```
   40       CONTINUE
         END IF
      END IF
C                  ... COMPUTE THE NEW ELEMENTS OF THE NEXT ROW
C                      IN THE E-ARRAY
      IND = MIN0(INDM,MAXCOL)
      DO 50 I = 1, IND
         INDI = I + INDM * MAXCOL
         R    = G(INDI-MAXCOL) - G(INDI)
         IF (DABS(R) .LT. EPS) THEN
            IER = 100
            RETURN
         END IF
         R      = EINIT - (E(I-1) - EINIT)/R * G(INDI)
         E(I-1) = EINIT
         EINIT  = R
   50 CONTINUE

C                  ... SAVE FOR THE FINAL VALUE.
      E(IND) = R
      RETURN
      END
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

SUBROUTINE GIKREC(N0, K, MAXCOL, GINIT, ALPHAI, BETAI)

C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  Concerning the g(i,k), for example in the Shank's transformation
C    g1n=DSn, ( where DSn=S(n+1)-Sn )
C       g2n=DS(n+1)..... gin=DS(n+i-1) ,  i=1,2, ......
C
C So if for instance,   Sn=1/(n+1) , n=0,1,....   then
C gi(n)= -1/ ( (n+i+1)(n+i) ) thus
C gi,(k-1)= -1/ (  (k+i)(k+i-1) )  (see ...COMPUTATION OF
C Gi,(K-1) below )
C g(k-1),(i-1)= -1/ ( (k+i-1)(k+i-2) ) ( see ... COMPUTATION OF
C G(k-1),(i-1) below  )
C
C  Suppose that the i-th term of the error asymptotic expansion
C        has the form       h**(alpha i)*log(h)**(beta i)
C.....................................................................
      DOUBLE PRECISION GINIT(1), AHKL1, ARKL1
      DOUBLE PRECISION AHIL1, ARIL1, LNBETA
      REAL ALPHAI(1), BETAI(1)

C     ... COMPUTATION OF  g  (k-1)
C  i
```

```
      M = MAX0(1, K-2)
      IF (K .GT. MAXCOL+1) M = MAXCOL
      DO 10 I = 1,M
AHKL1 = 1.D0/(N0*2.D0**(K-1))
ARKL1 = DLOG(AHKL1)
IF (ARKL1 .LT. 0.D0) THEN
  LNBETA = -(DABS(ARKL1)**BETAI(I))
ELSE
  LNBETA = ARKL1**BETAI(I)
ENDIF
GINIT(I) = AHKL1**ALPHAI(I)*LNBETA
10    CONTINUE

C ... COMPUTATION OF g (i-1)
C             k-1
      IF (K .GE. 3 .AND. K .LE. MAXCOL+1) THEN
DO 20 I = 1,K
  AHIL1 = 1.D0/(N0*2.D0**(I-1))
  ARIL1 = DLOG(AHIL1)
  IF(ARIL1 .LT. 0.D0) THEN
    LNBETA = -(DABS(ARIL1)**BETAI(K-1))
  ELSE
    LNBETA = ARIL1**BETAI(K-1)
  ENDIF
  GINIT(M+I) = AHIL1**ALPHAI(K-1)*LNBETA
20 CONTINUE
      ENDIF

      RETURN
      END
```

```
************************************************************************
* SUBROTINA : FACTOR
*
* INPUT :
*    W - matriz de dimensao (N,N) contendo a matriz A de ordem N
*             a ser factorizada
*
*          N - ordem da matriz A
*
* OUTPUT :
*    W - matriz de dimensao N contendo a factorizacao
*             de P*A para alguma matriz de permutacao P
*             especificada por IPIVOT
*
*          IPIVOT - vector inteiro de dimensao N indicando
*                   que a linha IPIVOT(K) foi usada para
*                   eliminar X(K), K=1,...,N
*
*          IFLAG - um inteiro
*
*    =  1, se se efectuar um numero par mudancas
*    = -1, se se efectuar um numero impar mudancas
*    =  0, se a matriz triangular superior tiver um ou
*                         mais zeros na diagonal principal se
*                         efectuar um numero par mudancas
*
*          Logo, Determinante(A) = IFLAG*W(1,1)*...*W(N,N)
*    Se IFLAG.NE.0, entao o sistema linear  A*X=B  pode ser
*          resolvido para X pela chamada
*          CALL SUBST(W,IPIVOT,B,N,X)
*
* METODO :
*     Metodo de CROUT com pesquisa parcial de pivot com
*          equilibragem.
*
************************************************************************

      SUBROUTINE FACTOR(W,N,D,IPIVOT,IFLAG)
      INTEGER IFLAG, IPIVOT(N), I, ISTAR, J, K
      DOUBLE PRECISION D(N), W(N,N), AWIKOD, COLMAX
      DOUBLE PRECISION RATIO, ROWMAX, TEMP

      IFLAG=1
      DO 9 I=1,N
         IPIVOT(I)=I
         ROWMAX=0.0D0
         DO 5 J=1,N
```

```
            IF (DABS(W(I,J)).GT.ROWMAX) ROWMAX=DABS(W(I,J))
5         CONTINUE
          IF (ROWMAX.EQ.0.0D0) THEN
             IFLAG=0
            ROWMAX=1.0D0
          ENDIF
 D(I)=ROWMAX
9      CONTINUE

       IF (N.EQ.1) RETURN

*      factorizacao

       DO 20 K=1,N-1
          COLMAX=DABS(W(K,K))/D(K)
          ISTAR=K
          DO 13 I=K+1,N
            AWIKOD=DABS(W(I,K))/D(I)
            IF (AWIKOD.GT.COLMAX) THEN
               COLMAX=AWIKOD
               ISTAR=I
            ENDIF
13    CONTINUE
         IF (COLMAX.EQ.0.0D0) THEN
             IFLAG=0
          ELSE
             IF (ISTAR.GT.K) THEN

*     fazer K a linha pivot trocando-a pela linha escolhida ISTAR

               IFLAG=-IFLAG
              I=IPIVOT(ISTAR)
              IPIVOT(ISTAR)=IPIVOT(K)
              IPIVOT(K)=I
              TEMP=D(ISTAR)
              D(ISTAR)=D(K)
              D(K)=TEMP
              DO 15 J=1,N
          TEMP=W(ISTAR,J)
          W(ISTAR,J)=W(K,J)
   W(K,J)=TEMP
15              CONTINUE
            ENDIF

*      eliminar X(K) das linhas K+1, ..., N.

16      DO 19 I=K+1,N
```

```
              W(I,K)=W(I,K)/W(K,K)
              RATIO=W(I,K)
              DO 19 J=K+1,N
         W(I,J)=W(I,J)-RATIO*W(K,J)
19             CONTINUE
         ENDIF
20    CONTINUE
      IF (W(N,N).EQ.0.0D0) IFLAG=0

      RETURN
      END

*********************************************************************
* SUBROTINA : SUBST
*
* INPUT :
*    W, IPIVOT, N - output da subrotina FACTOR, aplicada ‘a
*   matriz A de ordem N
*    B - vector de dimensao N, dando o lado direito do
*              sistema a ser resolvido
*
* OUTPUT :
*    X - vector de dimensao N satisfazendo A*X=B
*
* METODO :
*    Consiste em resolver dois sistemas triangulares, um
*          fazendo substituicoes ascendentes e outro
*          substituicoes descendentes. Usa para isso a
*          factorizacao contida em W e em IPIVOT
*          (gerada em FACTOR)
*********************************************************************

      SUBROUTINE SUBST(W,IPIVOT,B,N,X)
      INTEGER IPIVOT(N),I,IP,J
      DOUBLE PRECISION B(N), W(N,N), X(N), SUM

      IF (N.LE.1) THEN
         X(1)=B(1)/W(1,1)
        RETURN
      ENDIF
      IP=IPIVOT(1)
      X(1)=B(IP)
      DO 15 I=2,N
         SUM=0.0D0
         DO 14 J=1,I-1
    SUM=W(I,J)*X(J)+SUM
14       CONTINUE
```

```
         IP=IPIVOT(I)
 X(I)=B(IP)-SUM
15    CONTINUE

      X(N)=X(N)/W(N,N)
      DO 20 K=N-1,1,-1
         SUM=0.0D0
         DO 19 J=K+1,N
    SUM=W(K,J)*X(J)+SUM
19       CONTINUE
         X(K)=(X(K)-SUM)/W(K,K)
20    CONTINUE

      RETURN
      END
```

Bibliografia

[1] C. Baker, "The State of the Art in the Numerical Treatment of Integral Equations", in The State of the Art in Numerical Analysis", Oxford University Press, 1987, ISBN 0-19-853614-3

[2] C. Brezinski, "Accélération de la Convergence en Analyse Numérique", Springer-Verlag, 1977, ISBN 0-387-08241-7

[3] C. Brezinski, M. Zaglia, "Extrapolations Methods, Theory and Practice", Elsevier Science Publishers B. V., 1991 ISBN 0-444-88814-4

[4] C. Brezinski, "History of Continued Fractions and Padé Approximants", Springer-Verlag, 1991, ISBN 3-540-15286-5

[5] H. Brunner, "A Survey of Recent Advances in the Numerical Treatment of Volterra Integral and Integro-differential Equations", Journal of Computation and Applied Mathematics, Vol 8, Nº 3, 1982

[6] H. Brunner, "The Numerical Solution of Weakly Singular Volterra Integral Equations by Collocation on Graded Meshes", Mathematics of Computation, Vol. 45, Nº 172, pp. 417-437, Outubro 1985

[7] H. Brunner, P. J. van der Houwen, "The Numerical Solution of Volterra Equations", North-Holland, 1986, ISBN 0-444-70073-0

[8] H. Brunner, "Collocation Methods for One-Dimensional Fredholm and Volterra Integral Equations", *in* The State of the Art in Numerical Analysis, Oxford University Press, 1987, ISBN 0-19-853614-3

[9] R. F. Cameron, S. McKee, "Product Integration Methods for Second-kind Abel Integral Equations" Journal of Computation and Applied Mathematics, Nº 11, pp. 1-10, 1984

[10] S. D. Conte, C. Boor, "Elementary Numerical Analysis An Algorithmic Approach", Mc Graw-Hill Book Company, 1986, ISBN 0-07-Y66228-2

[11] M. Crouzeix, A. L. Mignot, "Analyse Numérique des Equations Différentielles", Masson, Paris, 1984, ISBN 2-225-77341-6

[12] L. M. Delves, J. Walsh, "Numerical solutions of integral equations", Oxford University Press, 1974, ISBN 0-19-853342-X

[13] L. M. Delves, J. L. Mohamed, "Computational methods for integral equations", Cambridge University Press, 1985, ISBN 0-521-26629-7

[14] M. T. Diogo, "Collocation Type Methods for Volterra Integral Equations", PhD Thesis, University of Kent, March 1991

[15] T. O. Espelid, A. Genz, "Numerical Integration: Recent Developments, Software and Applications", Kluwer Academic Publishers, 1992, ISBN 0-7923-1583-9

[16] M. A. Golberg, "Numerical Solution of Integral Equations", Plenum Press, 1990, ISBN 0-306-43262-5

[17] R. Gorenflo, S. Vessella, "Abel Integral Equations, Analysis and Applications", Springer-Verlag, 1991, ISBN 0-387-53668-X NewYork

[18] G. Hämmerlin, K. Hoffmann, "Numerical Mathematics", Springer-Verlag, 1991, ISBN 0-387-97494-6

[19] R. A. Handelsman, W. E. Olmstead, "Asymptotic Solution to a Class of Nonlinear Volterra Integral Equations", SIAM Journal of Applied Mathematics, II, Nº 30, pp. 180-189, 404, 493, 1976

[20] F. de Hoog, R. Weiss, "Asymptotic Expansions for Product Integration", Mathematical Computations, Vol. 27, Nº 122, 1973, pp. 295-306.

[21] K., Atkinson, "An Introduction to Numerical Analysis", John Wiley and Sons, 1978 ISBN 0-471-02985-8

[22] D. Kincaid, W. Cheney, "Numerical Analysis", Books/Cole Publishing Co., 1991, ISBN 0-534-13014-3

[23] E. Kreyszig, "Introductory Functional Analysis with Applications", John-Wiley & Sons, 1978

[24] N. Levinson, "A Nonlinear Volterra Equation Arising in the Theory of Superfluidity", Journal of Mathematical Analysis and Applications I, I-11, 1960

[25] M. J. Lighthill, "Contribution to the Theory of Heat Transfer Through a Laminar Layer", 1950

[26] P. Lima, M. Graça, "Convergence Acceleration for boundary value problems with singularities using the E-algorithm", Journal of Computation and Applied Mathematics (Aceite para publicação)

[27] P. Linz, "The Numerical Solution of Volterra Integral Equations by Finite Difference Methods", MRC Tchnical Summary Report, Nº 825, University of Wisconsin, Madison, Novembro 1967

[28] P. Linz, "Numerical Methods for Volterra Integral Equations with Singular Kernels", SIAM, Vol 6, Nº 3, Setembro 1969

[29] P. Linz, "Numerical Methods for Volterra Integral Equations of the First Kind", Journal of Computing, Nº 12, pp. 393-397, 1969

[30] P. Linz, "Product Integration Methods for Volterra Integral Equations of the First Kind", BIT 11, pp. 413-421, 1971

[31] P. Linz, "Analytical and Numerical Methods for Volterra Equations", SIAM studies in Applied Mathematics, Philadelphia, 1985

[32] J. N. Lyness, B. W. Ninham, "Numerical Quadrature and Asymptotic Expansions", Mathematics of Computation, Nº 21, pp. 162-178, 1966

[33] W. R. Mann, F. Wolf, "Heat Transfer Between Solids and Gases under Nonlinear Boundary Conditions", 1950

[34] G. I. Marchuk, V. V. Shaidurov, "Difference Methods and Their Extrapolations", Springer-Verlag, 1983, ISBN 0-387-90794-7

[35] S. McKee, "Volterra Integral and Integro-differential Equations Arising from Problems in Engineering and Science", Bulletin of the Institute of Applied Mathematics, Nº 24, 1988

[36] A. N. Netravali, "Spline Approximation to the Solution of the Volterra Integral Equation of the Second Kind", Mathematics of Computation, Vol. 27, Nº 121, Janeiro 1973

[37] Peep UBA, "A Collocation Method with Cubic Splines for Multidimensional Weakly Singular Nonlinear Integral Equations", Journal of Integral Equations ana Applications, Volume 6, Number 2, Primavera 1994

[38] J. H. Roberts, W. R. Mann, "On a Certain Nonlinear Integral Equation of the Volterra Type", Bulletin of the American Mathematical Society, Vol. 56, 1950

[39] P. G. Rooney, "On an Integral Equation of Šub-Sizonenko", Glasgow Mathematical Journal, Nº 24, 1983

[40] Yu. A. Shub-Sizonenko, "Inversion of one Integral Operator by the Method of Expansion with Respect to Orthogonal Watson Operators", Siberian Mathematical Journal, Nº 20, Plenum Publishing Company, 1979

[41] M. Sibony, J. Mardon, "Approximations et Équations Différentielles", Hermann, Éditeurs des Sciences et des Arts, 1982, ISBN 2-7056-1406-2

[42] I. H. Sloan, "Superconvergence", *in* Numerical Solutions of Integral Equations, Plenum Press, 1990, ISBN 0-306-43262-5

[43] H. J. Stetter, "Asymptotic Expansions for the Error of Discretizations Algorithms for Non-linear Functional Equations", 1964

[44] J. Stoer, R. Bulirsch, "Introduction to Numerical Analysis", Springer-Verlag Inc., 1980, ISBN 0-387-90420-4

[45] W. Sweldens, R. Piessens, "Quadrature Formulae and Asymptotic Error Expansions for Wavelet Approximations of Smooth Functions", SIAM, Journal of Numerical Analysis, Vol. 31, Nº 4, pp. 1240-1264, Agosto 1994

[46] M. E. A. El Tom, "On the Numerical Stability of Spline Functions Approximations to Solutions of Volterra Integral Equations of the Second Kind", BIT 14, pp. 136-143, 1974

[47] A. Young, "The Application of Approximate product-Integration to the Numerical Solution of Integral Equations", The Unviversity, Liverpool, 1954