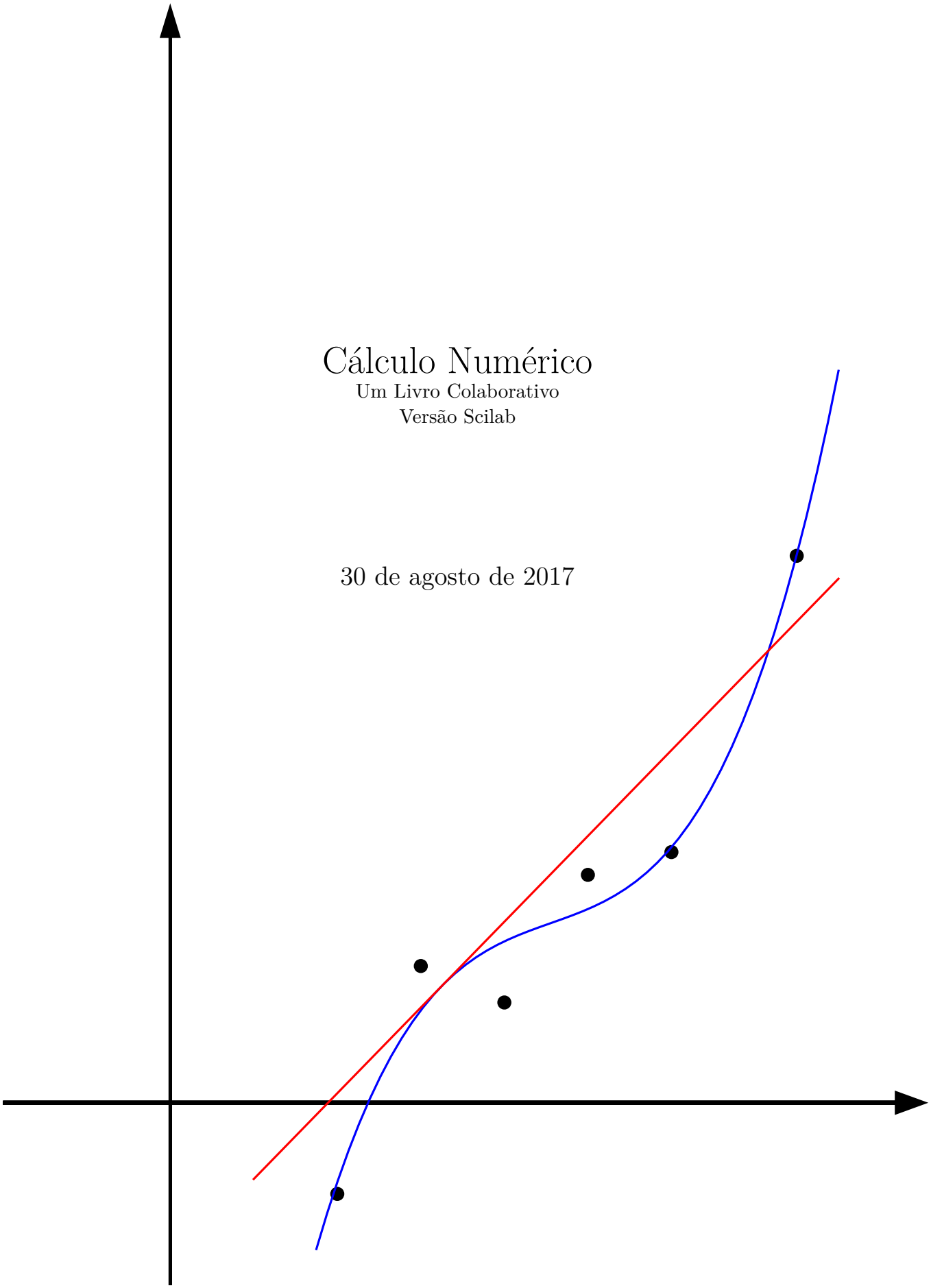
Cálculo Numérico

Um Livro Colaborativo

Versão Scilab

30 de agosto de 2017



Organizadores

Dagoberto Adriano Rizzotto Justo - UFRGS

Esequia Sauter - UFRGS

Fabio Souto de Azevedo - UFRGS

Leonardo Fernandes Guidi - UFRGS

Matheus Correia dos Santos - UFRGS

Pedro Henrique de Almeida Konzen - UFRGS

Licença

Nota dos organizadores

Estamos escrevendo este livro de forma colaborativa desde 2011 e, recentemente, decidimos por abrir a colaborações externas. Nosso objetivo é produzir um material didático no nível de graduação de excelente qualidade e de acesso livre pela colaboração entre professores e alunos de universidades, institutos de educação e demais interessados na análise, estudo e aplicação de métodos numéricos nos mais diversos ramos da ciência e da tecnologia.

O sucesso do projeto depende da colaboração! Edite você mesmo o livro, dê sugestões ou nos avise de erros e imprecisões. Toda a colaboração é bem vinda. Saiba mais visitando o site oficial do projeto:

`http://www.ufrgs.br/numerico`

Nós preparamos uma série de ações para ajudá-lo a participar. Em primeiro lugar, o acesso irrestrito ao livro pode se dar através do site oficial do projeto. Disponibilizamos o livro na versão original em PDF e versões adaptadas em HTML, EPUB e Slides. Além disso, o livro está escrito em código LaTeX disponível em repositório GitHub público.

Nada disso estaria completo sem uma licença apropriada à colaboração. Por isso, escolhemos disponibilizar o material do livro sob licença Creative Commons Atribuição-CompartilhaIgual 3.0 Não Adaptada (CC-BY-SA 3.0). Ou seja, você pode copiar, redistribuir, alterar e construir um novo material para qualquer uso, inclusive comercial. Leia a licença para maiores informações.

Desejamos-lhe ótimas colaborações!

Prefácio

Este livro busca abordar os tópicos de um curso de introdução ao cálculo numérico moderno oferecido a estudantes de matemática, física, engenharias e outros. A ênfase é colocada na formulação de problemas, implementação em computador da resolução e interpretação de resultados. Pressupõe-se que o estudante domine conhecimentos e habilidades típicas desenvolvidas em cursos de graduação de cálculo, álgebra linear e equações diferenciais. Conhecimentos prévios em linguagem de computadores é fortemente recomendável, embora apenas técnicas elementares de programação sejam realmente necessárias.

Nesta versão do livro, fazemos ênfase na utilização do **software** livre Scilab para a implementação dos métodos numéricos abordados. Recomendamos ao leitor ter à sua disposição um computador com o `Scilab` instalado. Não é necessário estar familiarizado com esta linguagem, mas recomendamos a leitura do Apêndice A, no qual apresentamos uma rápida introdução a este pacote computacional. Alternativamente, existem algumas soluções em nuvem que fornecem acesso ao `Scilab` via internet. Por exemplo, a plataforma virtual rollApp (`https://www.rollapp.com/app/scilab`) ou o Scilab on Cloud (`http://cloud.scilab.in/`).

Os códigos computacionais dos métodos numéricos apresentados no livro são implementados em uma abordagem didática. Isto é, temos o objetivo de que a implementação em linguagem computacional venha a auxiliar o leitor no aprendizado das técnicas numéricas que são apresentadas no livro. Implementações computacionais eficientes de técnicas de cálculo numérico podem ser obtidas na série de livros "Numerical Recipes", veja [10].

Sumário

Capítulo 1

Problemas de valor inicial

Neste capítulo, vamos estudar metodologias numéricas para aproximar a solução de problema de valor inicial (problema de valor inicial) para equações diferenciais ordinárias. Primeiramente, daremos atenção aos problemas de primeira ordem e, depois, mostraremos que estas técnicas podem ser estendidas para problemas e sistemas de ordem superior. Considere um problema de valor inicial de primeira ordem dado por:

$$\begin{aligned} u'(t) &= f(t, u(t)), \ \ t > t^{(1)} && \text{(1.1a)} \\ u(t^{(1)}) &= a \ \ \text{(condição inicial).} && \text{(1.1b)} \end{aligned}$$

A incógnita de um problema de valor inicial é uma função que satisfaz a equação diferencial (1.1a) e a condição inicial (1.1b).

Considere os próximos três exemplos:

Exemplo 1.0.1.

$$\begin{aligned} u'(t) &= t && \text{(1.2)} \\ u(0) &= 2 && \text{(1.3)} \end{aligned}$$

Exemplo 1.0.2.

$$\begin{aligned} u'(t) &= u(t) && \text{(1.4)} \\ u(0) &= 1 && \text{(1.5)} \end{aligned}$$

Exemplo 1.0.3.

$$\begin{aligned} u'(t) &= \operatorname{sen}\left(u(t)^2 + \operatorname{sen}\left(t\right)\right) && \text{(1.6)} \\ u(0) &= a && \text{(1.7)} \end{aligned}$$

A solução do primeiro exemplo é $u(t) = t^2/2 + 2$ pois satisfaz a equação diferencial e a condição inicial. A solução do segundo também é facilmente obtida: $u(t) = e^t$. Porém como podemos resolver o terceiro problema?

Para muitos problemas de valor inicial da forma (1.1), não é possível encontrar uma expressão analítica fechada, ou seja, sabe-se que a solução existe e é única, porém não podemos expressá-la em termos de funções elementares. Por isso é necessário calcular aproximações numéricas para a solução.

Existem uma enorme família de metodologias para construir soluções numéricas para problemas de valor inicial. Aqui, vamos nos limitar a estudar métodos que aproximam $u(t)$ em um conjunto finito de valores de t. Este conjunto de valores será chamado de **malha** e será denotado por $\{t^{(i)}\}_{i=1}^{N} = \{t^{(1)}, t^{(2)}, t^{(3)}, \ldots, t^{(N)}\}$. Desta forma, aproximamos a solução $u(t^{(i)})$ por $u^{(i)}$ em cada ponto da malha usando diferentes esquemas numéricos.

1.1 Método de Euler

Nesta seção, contruiremos o mais simples dos métodos para resolver problemas de valor inicial: o método de Euler com passo constante. Por passo constante, queremos dizer que os pontos da malha estão todos igualmente espaçados, isto é:

$$t^{(i)} = (i-1)h, \quad i = 1,2,\ldots,N.$$

onde h é passo, ou seja, a distância entre dois pontos da malha.

Considere, agora, o problema de valor inicial dado por:

$$u'(t) = f(t,u(t)), \quad t > t^{(1)} \tag{1.8}$$
$$u(t^{(1)}) = a. \tag{1.9}$$

Ao invés de tentar solucionar o problema para qualquer $t > t^{(1)}$, iremos aproximar $u(t)$ em $t = t^{(2)}$.

Integrando (1.8) de $t^{(1)}$ até $t^{(2)}$, obtemos:

$$\int_{t^{(1)}}^{t^{(2)}} u'(t)\,dt = \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\,dt \tag{1.10}$$
$$u(t^{(2)}) - u(t^{(1)}) = \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\,dt \tag{1.11}$$
$$u(t^{(1)}) = u(t^{(1)}) + \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\,dt \tag{1.12}$$

Seja u_n a aproximação de $u(t_n)$. Para obter o método numérico mais simples

aproximamos f em $[t^{(1)},t^{(2)}]$ pela função constante $f(t,u(t)) \approx f(t^{(1)},u^{(1)})$,

$$\begin{aligned} u^{(2)} &= u^{(1)} + f(t^{(1)},u^{(1)}) \int_{t^{(1)}}^{t^{(2)}} dt && (1.13)\\ u^{(2)} &= u^{(1)} + f(t^{(1)},u^{(1)})(t^{(2)} - t^{(1)}) && (1.14)\\ u^{(2)} &= u^{(1)} + hf(t^{(1)},u^{(1)}) && (1.15) \end{aligned}$$

Este procedimento pode ser repetido para $t^{(3)}$, $t^{(4)}$, ..., obtendo, assim, o chamado **método de Euler**:

$$\begin{aligned} u^{(n+1)} &= u^{(n)} + h\ f(t^{(n)},u^{(n)}), && (1.16)\\ u^{(1)} &= u^{(1)} = u(t^{(1)}) \text{ (condição inicial).} && (1.17) \end{aligned}$$

Exemplo 1.1.1. Considere o problema de valor inicial

$$\begin{aligned} u'(t) &= 2u(t)\\ u(0) &= 1 \end{aligned}$$

cuja solução é $u(t) = e^{2t}$. O método de Euler aplicado a este problema produz o esquema:

$$\begin{aligned} u^{(k+1)} &= u^{(k)} + 2hu^{(k)} = (1+2h)u^{(k)} && (1.18)\\ u^{(1)} &= 1, && (1.19) \end{aligned}$$

Suponha que queremos calcular o valor aproximado de $u(1)$ com $h = 0{,}2$. Então os pontos $t^{(1)} = 0$, $t^{(2)} = 0{,}2$, $t^{(3)} = 0{,}4$, $t^{(4)} = 0{,}6$, $t^{(5)} = 0{,}8$ e $t^{(6)} = 1{,}0$ formam os seis pontos da malha. As aproximações para a solução nos pontos da malha usando o método de Euler são:

$$\begin{aligned} u(0) &\approx u^{(1)} = 1\\ u(0{,}2) &\approx u^{(2)} = (1+2h)u^{(1)} = 1{,}4u^{(1)} = 1{,}4\\ u(0{,}4) &\approx u^{(3)} = 1{,}4u^{(2)} = 1{,}96\\ u(0{,}6) &\approx u^{(4)} = 1{,}4u^{(3)} = 2{,}744\\ u(0{,}8) &\approx u^{(5)} = 1{,}4u^{(4)} = 3{,}8416\\ u(1{,}0) &\approx u^{(6)} = 1{,}4u^{(5)} = 5{,}37824 \end{aligned}$$

Essa aproximação é bem grosseira quando comparamos com a solução do problema em $t = 1$: $u(1) = e^2 \approx 7{,}38906$. Não obstante, se tivéssemos escolhido um passo menor, teríamos obtido uma aproximação melhor. Veja tabela abaixo com valores obtidos com diferentes valores de passo h.

h	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
$u^{(N)}$	6,1917	6,7275	7,0400	7,2096	7,2980	7,3432	7,3660

De fato, podemos mostrar que quando h se aproxima de 0, a solução aproximada via método de Euler converge para a solução exata e^2. Para isto, basta observar que a solução da relação de recorrência (1.18) é dada por

$$u^{(k)} = (1+2h)^{k-1}.$$

Como $t^{(k)} = (k-1)h$ e queremos a solução em $t = 2$, a solução aproximada pelo método de Euler com passo h em é dada por:

$$u^{(k)} = (1+2h)^{k-1} = (1+2h)^{\frac{2}{h}}.$$

Aplicando o limite $h \to 0+$, temos:

$$\lim_{h\to 0+} (1+2h)^{\frac{2}{h}} = e^2.$$

Exercícios Resolvidos

ER 1.1.1. Aproxime a solução do problema de valor inicial

$$u'(t) = -0{,}5u(t) + 2 + t \tag{1.20}$$
$$u(0) = 8 \tag{1.21}$$

Usando os seguinte passos: $h = 10^{-1}$, $h = 10^{-2}$, $h = 10^{-3}$, $h = 10^{-4}$ e $h = 10^{-5}$ e compare a solução aproximada em $t = 1$ com a solução exata dada por:

$$u(t) = 2t + 8e^{-t/2} \Longrightarrow u(1) = 2 + 8e^{-1/2} \approx 6{,}85224527770107 \tag{1.22}$$

Solução. Primeramente itentificamos $f(t,u) = -0{,}5u + 2 + t$ e construímos o processo iterativo do método de Euler:

$$u^{(n+1)} = u^{(n)} + h(-0{,}5u^{(n)} + 2 + t^{(n)}), \quad n = 1,2,3,\ldots \tag{1.23}$$
$$u^{(1)} = 8 \tag{1.24}$$

O seguinte código pode ser usado para implementar no `Scilab` a recursão acima:

```
function u=euler(h,Tmax)
  u= 8;
  t= 0;
  itmax = Tmax/h;
```

```
  for n=1:itmax
    t= (n-1)*h;
    u= u + h*(-0.5*u+2+t);
   end
endfunction
```

o qual pode ser invocado da seguinte forma:

```
-->euler(1e-1,1)
 ans  =

    6.7898955
```

Podemos construir um vetor com as cinco soluções da seguinte forma:

```
 -->S=[euler(1e-1,1) euler(1e-2,1) euler(1e-3,1) euler(1e-4,1) euler(1e-5,1)]
 S  =

    6.7898956    6.846163     6.851639     6.852185     6.8522392
```

A seguinte tabela resume os resultados obtidos:

h	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
Euler	6,7898955	6,8461635	6,8516386	6,8521846	6,8522392
ε_{rel}	9,1e-03	8,9e-04	8,9e-05	8,9e-06	8,9e-07

$\Diamond$

Vamos agora, analisar o desempenho do método de Euler usando um exemplo mais complicado, porém ainda simples suficiente para que possamos obter a solução exata:

Exemplo 1.1.2. Considere o problema de valor inicial relacionado à equação logística:

$$\begin{aligned} u'(t) &= u(t)(1-u(t)) \\ u(0) &= 1/2 \end{aligned}$$

Podemos obter a solução exata desta equação usando o método de separação de variáveis e o método das frações parciais. Para tal escrevemos:

$$\frac{du(t)}{u(t)(1-u(t))} = dt$$

O termo $\frac{1}{u(t)(1-u(t))}$ pode ser decomposto em frações parciais como $\frac{1}{u} + \frac{1}{1-u}$ e chegamos na seguinte equação diferencial:

$$\left(\frac{1}{u(t)} + \frac{1}{1-u(t)}\right) du = dt.$$

Integrando termo-a-termo, temos a seguinte equação algébrica relacionando $u(t)$ e t:

$$\ln(u(t)) - \ln(1-u(t)) = t + C$$

Onde C é a constante de integração, que é definida pela condição inicial, isto é, $u = 1/2$ em $t = 0$. Substituindo, temos $C = 0$. O que resulta em:

$$\ln\left(\frac{u(t)}{1-u(t)}\right) = t$$

Equivalente a

$$\frac{u(t)}{1-u(t)} = e^t \Longrightarrow u(t) = (1-u(t))e^t \Longrightarrow (1+e^t)u(t) = e^t$$

E, finalmente, encontramos a solução exata dada por $u(t) = \frac{e^t}{1+e^t}$.

Vejamos, agora, o esquema iterativo produzido pelo método de Euler:

$$\begin{aligned} u^{(k+1)} &= u^{(k)} + hu^{(k)}(1-u^{(k)}), \\ u^{(1)} &= 1/2. \end{aligned}$$

O seguinte código pode ser usado para implementar no `Scilab` a recursão acima:

```
function u=euler(h,Tmax)
  u= .5;
  itmax = Tmax/h;
  for n=1:itmax
    u= u + h*u*(1-u);
   end
endfunction
```

o qual pode ser invocado da seguinte forma ($h = 1e-1, t = 2$):

```
-->euler(1e-1,2)
 ans  =

    0.8854273
```

Para fins de comparação, calculamos a solução exata e aproximada para alguns valores de t e de passo h e resumimos na tabela abaixo:

t	Exato	Euler $h = 0{,}1$	Euler $h = 0{,}01$
0	$1/2$	0,5	0,5
$1/2$	$\frac{e^{1/2}}{1+e^{1/2}} \approx 0{,}6224593$	0,6231476	0,6225316
1	$\frac{e}{1+e} \approx 0{,}7310586$	0,7334030	0,7312946
2	$\frac{e^2}{1+e^2} \approx 0{,}8807971$	0,8854273	0,8812533
3	$\frac{e^3}{1+e^3} \approx 0{,}9525741$	0,9564754	0,9529609

Exercícios

E 1.1.1. Resolva o problema de valor inicial a seguir envolvendo uma equação não autônoma, isto é, quando a função $f(t,u)$ depende explicitamente do tempo. Use passo $h = 0{,}1$ e $h = 0{,}01$. Depois compare com a solução exata dada por $u(t) = 2e^{-t} + t - 1$ nos intantes $t = 0$, $t = 1$, $t = 2$ e $t = 3$.

$$\begin{aligned} u'(t) &= -u(t) + t \\ u(0) &= 1, \end{aligned}$$

E 1.1.1. O esquema recursivo de Euler fica:

$$\begin{aligned} u^{(k+1)} &= u^{(k)} + h(-u^{(k)} + t^{(k)}) \\ u^{(1)} &= 1 \end{aligned}$$

Comparação:

t	Exato	Euler $h = 0{,}1$	Euler $h = 0{,}01$
0	1	1	1
1	$2e^{-1} \approx 0{,}7357589$	0,6973569	0,7320647
2	$2e^{-2} + 1 \approx 1{,}2706706$	1,2431533	1,2679593
3	$2e^{-3} + 2 \approx 2{,}0995741$	2,0847823	2,0980818

E 1.1.2. Resolva o prolema de valor inicial envolvendo uma equação não linear usando passo $h = 0{,}1$ e $h = 0{,}01$.

$$\begin{aligned} u'(t) &= \cos(u(t)) \\ u(0) &= 0, \end{aligned}$$

Depois compare com a solução exata dada por

$$u(t) = \tan^{-1}\left(\frac{e^{2t} - 1}{2e^t}\right).$$

nos intantes $t = 0$, $t = 1$, $t = 2$ e $t = 3$.

E 1.1.2.

$$\begin{aligned} u^{(k+1)} &= u^{(k)} + h\cos(u^{(k)}) \\ u^{(1)} &= 0 \end{aligned}$$

Comparação:

t	Exato	Euler $h = 0{,}1$	Euler $h = 0{,}01$
0	0	0	0
1	0,8657695	0.8799602	0.8671764
2	1,3017603	1.3196842	1.3035243
3	1,4713043	1.4827638	1.4724512

E 1.1.3. Resolva a seguinte problema de valor inicial linear com passo $h = 10^{-4}$ via método de Euler e compare a solução obtida com o valor exato $y(t) = e^{\text{sen}\,(t)}$ em $t = 2$:

$$\begin{aligned} y'(t) &= \cos(t)y(t) \\ y(0) &= 1. \end{aligned}$$

E 1.1.3. Aproximação via Euler: 2,4826529, exata: $e^{\text{sen}\,(2)} \approx 2{,}4825777$. Erro relativo aproximado: 3×10^{-5}.

1.2 O método de Euler melhorado

O método de Euler estudado na Seção 1.1 é aplicação bastante restrita devido à sua pequena precisão, isto é, normalmente precisamos escolher um passo h muito pequeno para obter soluções de boa qualidade, o que implica um número elevado de passos e, consequentemente, alto custo computacional.

Nesta seção, contruiremos o **método de Euler melhorado** ou **método de Euler modificado** ou, ainda, . Para tal, considere o problema de valor inicial dado por:

$$u'(t) = f(t,u(t)), \quad t > t^{(1)} \tag{1.25}$$

$$u(t^{(1)}) = a. \tag{1.26}$$

Assim como fizemos para o método de Euler, integramos (1.25) de $t^{(1)}$ até $t^{(2)}$ e obtemos:

$$\int_{t^{(1)}}^{t^{(2)}} u'(t)\,dt = \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\,dt \tag{1.27}$$

$$u(t^{(2)}) - u(t^{(1)}) = \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\,dt \tag{1.28}$$

$$u(t^{(2)}) = u(t^{(1)}) + \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\,dt \tag{1.29}$$

A invés de aproximar $f(t,u(t))$ como uma constante igual ao seu valor em $t = t^{(1)}$, aplicamos a regra do trapézio (ver **??**) à integral envolvida no lado direito da expressão, isto é:

$$\int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\,dt = \left[\frac{f\left(t^{(1)},u(t^{(1)})\right) + f\left(t^{(2)},u(t^{(2)})\right)}{2}\right] h + O(h^3) \qquad (1.30)$$

onde $h = t^{(2)} - t^{(1)}$. Como o valor de $u(t^{(2)})$ não é conhecido antes de o passo ser realizado, aproximamos seu valor aplicando o método de Euler:

$$\tilde{u}(t^{(2)}) = u(t^{(1)}) + hf\left(t^{(1)},u(t^{(1)})\right)$$

Assim obtemos:

$$\begin{aligned} u(t^{(2)}) &= u(t^{(1)}) + \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\,dt \\ &\approx u(t^{(1)}) + \left[\frac{f\left(t^{(1)},u(t^{(1)})\right) + f\left(t^{(2)},u(t^{(2)})\right)}{2}\right] h \\ &\approx u(t^{(1)}) + \left[\frac{f\left(t^{(1)},u(t^{(1)})\right) + f\left(t^{(2)},\tilde{u}(t^{(2)})\right)}{2}\right] h \end{aligned}$$

onde

$$\begin{aligned} k_1 &= f\left(t^{(1)},u(t^{(1)})\right) \\ \text{e} \\ k_2 &= f\left(t^{(2)},\tilde{u}(t^{(2)})\right) = f\left(t^{(2)},u(t^{(1)}) + hk_1\right) \end{aligned}$$

Portanto, o método recursivo de Euler melhorado assume a seguinte forma:

$$\begin{aligned} \tilde{u}^{(k+1)} &= u^{(k)} + hf(t^{(k)},u^{(k)}), \\ u^{(k+1)} &= u^{(k)} + \frac{h}{2}\left(f(t^{(k)},u^{(k)}) + f(t^{(k)},\tilde{u}^{(k)})\right), \\ u^{(1)} &= a \ \text{(condição inicial)}. \end{aligned}$$

Que pode ser escrito equivalentemente como:

$$\begin{aligned} k_1 &= f(t^{(k)},u^{(k)}), \\ k_2 &= f(t^{(k+1)},u^{(k)} + k_1), \\ u^{(k+1)} &= u^{(k)} + h\frac{k_1 + k_2}{2}, \\ u^{(1)} &= a \ \text{(condição inicial)}. \end{aligned}$$

Aqui k_1 e k_2 são variáveis auxiliares que representam as inclinações e devem ser calculadas a cada passo. Esta notação é compatível com a notação usada nos métodos de Runge-Kutta, uma família de esquemas iterativos para aproximar problemas de valor inicial, da qual o método de Euler e o método de Euler melhorado são casos particulares. Veremos os métodos de Runge-Kutta na Seção 1.6.

Exercícios Resolvidos

ER 1.2.1. Resolva pelo método de Euler melhorado problema de valor inicial do Exercício Resolvido 1.1.1:

$$u'(t) = -0{,}5u(t) + 2 + t \tag{1.31}$$
$$u(0) = 8 \tag{1.32}$$

Usando os seguinte passos: $h = 10^{-1}$, $h = 10^{-2}$, $h = 10^{-3}$, $h = 10^{-4}$ e $h = 10^{-5}$ e compare a solução aproximada em $t = 1$ com a solução obtida pelo método de Euler e a solução exata dada por:

$$u(t) = 2t + 8e^{-t/2} \Longrightarrow u(1) = 2 + 8e^{-1/2} \approx 6{,}85224527770107 \tag{1.33}$$

Solução. Primeramente itentificamos $f(t,u) = -0{,}5u + 2 + t$ e construímos o processo iterativo do método de Euler melhorado:

$$k_1 = f(t^{(n)},u^{(n)}) = -0{,}5u^{(n)} + 2 + t^{(n)} \tag{1.34}$$
$$\tilde{u} = u^{(n)} + hk_1 \tag{1.35}$$
$$k_2 = f(t^{(n+1)},\tilde{u}) = -0{,}5\tilde{u} + 2 + t^{(n+1)} \tag{1.36}$$
$$u^{(n+1)} = u^{(n)} + h(k_1 + k_2), \quad n = 1,2,3,\ldots \tag{1.37}$$
$$u^{(1)} = 8 \text{ (condição inicial).} \tag{1.38}$$

O seguinte código pode ser usado para implementar no `Scilab` a recursão acima:

```
function u=euler_mod(h,Tmax)
  u= 8;
  itmax = Tmax/h;

  for n=1:itmax
    t=(n-1)*h;
    k1 = (-0.5*u + 2 + t);
    u_til = u + h*k1;
    k2 = (-0.5*u_til + 2 + t + h);
    u= u + h * (k1 + k2)/2;
  end
endfunction
```

o qual pode ser invocado da seguinte forma:

```
-->euler_mod(1e-1,1)
 ans  =

    6.8532941
```

Podemos construir um vetor com as cinco soluções da seguinte forma:

```
 -->S=[euler_mod(1e-1,1) euler_mod(1e-2,1) euler_mod(1e-3,1) euler_mod(1e-4,1) euler_m
S  =

    6.8532949    6.8522554    6.8522454    6.8522453    6.8522453
```

A seguinte tabela resume os resultados obtidos:

h	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
Euler	6,7898955	6,8461635	6,8516386	6,8521846	6,8522392
ε_{rel}	9,1e-03	8,9e-04	8,9e-05	8,9e-06	8,9e-07
Euler mod.	6,8532949	6,8522554	6,8522454	6,8522453	6,8522453
ε_{rel}	1,5e-04	1,5e-06	1,5e-08	1,5e-10	1,5e-12

$\Diamond$

1.3 Solução de sistemas de equações diferenciais

Nas seções 1.1 e 1.2, construimos dois métodos numéricos para resolver problemas de valor inicial. Nestas seções, sempre consideremos problemas envolvendo equações diferenciais ordinárias de primeira ordem, isto é:

$$\begin{aligned} u'(t) &= f(t,u(t)), \quad t > t^{(1)} \\ u(t^{(1)}) &= a. \end{aligned}$$

Estas técnicas podem ser diretamente estendidas para resolver numericamente problemas de valor inicial envolvendo sistemas de equações diferenciais ordinárias de

primeira ordem, isto é:

$$
\begin{aligned}
u_1'(t) &= f_1(t,u_1(t),u_2(t),u_3(t),\ldots,u_n(t)), \quad t > t^{(1)}, && (1.39)\\
u_2'(t) &= f_2(t,u_1(t),u_2(t),u_3(t),\ldots,u_n(t)), \quad t > t^{(1)}, && (1.40)\\
u_3'(t) &= f_3(t,u_1(t),u_2(t),u_3(t),\ldots,u_n(t)), \quad t > t^{(1)}, && (1.41)\\
&\vdots && (1.42)\\
u_n'(t) &= f_n(t,u_1(t),u_2(t),u_3(t),\ldots,u_n(t)), \quad t > t^{(1)}, && (1.43)\\
u(t^{(1)}) &= a_1, && (1.44)\\
u(t^{(2)}) &= a_2, && (1.45)\\
u(t^{(3)}) &= a_3, && (1.46)\\
&\vdots && (1.47)\\
u(t^{(n)}) &= a_n. && (1.48)
\end{aligned}
$$

O Problema (1.39) pode ser escrito como um problema de primeira ordem envolvendo uma única incógnita, $u(t)$, dada como um vetor de funções $u_j(t)$, isto é:

$$
u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \\ \vdots \\ u_n(t) \end{bmatrix}
$$

De forma que o Problema (1.39) assuma a seguinte forma:

$$
\begin{aligned}
u'(t) &= f(t,u(t)), \quad t > t^{(1)}\\
u(t^{(1)}) &= a.
\end{aligned}
$$

onde

$$
f(t,u(t)) = \begin{bmatrix} f_1(t,u_1(t),u_2(t),u_3(t),\ldots,u_n(t)) \\ f_2(t,u_1(t),u_2(t),u_3(t),\ldots,u_n(t)) \\ f_3(t,u_1(t),u_2(t),u_3(t),\ldots,u_n(t)) \\ \vdots \\ f_n(t,u_1(t),u_2(t),u_3(t),\ldots,u_n(t)) \end{bmatrix}
$$

e

$$a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{bmatrix}$$

Veja o o Exemplo 1.49

Exemplo 1.3.1. Considere o problema de resolver numericamente pelo método de Euler o seguinte sistema de equações diferenciais ordinárias com valores iniciais:

$$\begin{aligned} x'(t) &= -y(t), && \text{(1.49a)} \\ y'(t) &= x(t), && \text{(1.49b)} \\ u(0) &= 1. && \text{(1.49c)} \\ v(0) &= 0. && \text{(1.49d)} \end{aligned}$$

Para aplicar o método de Euler a este sistema, devemos encarar as duas incógnitas do sistema como entradas de um vetor, ou seja, escrevemos:

$$u(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}.$$

e, portanto, o sistema pode ser escrito como:

$$\begin{bmatrix} x^{(k+1)} \\ y^{(k+1)} \end{bmatrix} = \begin{bmatrix} x^{(k)} \\ y^{(k)} \end{bmatrix} + h \begin{bmatrix} -y^{(k)} \\ x^{(k)} \end{bmatrix}.$$

Observe que este processo iterativo é equivalente a discretiza as equações do sistema uma-a-uma, isto é:

$$\begin{aligned} x^{(k+1)} &= x^{(k)} - hy^{(k)}, \\ y^{(k+1)} &= y^{(k)} + hx^{(k)}, \\ x^{(1)} &= 1, \\ y^{(1)} &= 0, \end{aligned}$$

Exercícios Resolvidos

ER 1.3.1. Resolva pelo método de Euler melhorado o seguinte problema de valor inicial para aproximar o valor de x e y entre $t = 0$ e $t = 1$:

$$\begin{aligned} x'(t) &= x(t) - y(t), \\ y'(t) &= x(t) - y(t)^3, \\ u(0) &= 1. \\ v(0) &= 0. \end{aligned}$$

Solução. Primeiramente, identificamos $u(t)$ como o vetor incógnita:

$$u(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}.$$

Depois aplicamos a recursão do método de Euler melhorado dada por:

$$\begin{aligned} \tilde{u}^{(k+1)} &= u^{(k)} + hf(t^{(k)}, u^{(k)}), \\ u^{(k+1)} &= u^{(k)} + \frac{h}{2}\left(f(t^{(k)}, u^{(k)}) + f(t^{(k)}, \tilde{u}^{(k)})\right), \end{aligned}$$

isto é:

$$\begin{aligned} \tilde{x}^{(k+1)} &= x^{(k)} + h\left(x^{(k)} - y^{(k)}\right) \\ \tilde{y}^{(k+1)} &= y^{(k)} + h\left(x^{(k)} - {y^{(k)}}^3\right) \\ x^{(k+1)} &= x^{(k)} + \frac{h}{2}\left[\left(x^{(k)} - y^{(k)}\right) + \left(\tilde{x}^{(k)} - \tilde{y}^{(k)}\right)\right] \\ y^{(k+1)} &= y^{(k)} + \frac{h}{2}\left[\left(x^{(k)} - {y^{(k)}}^3\right) + \left(\tilde{x}^{(k)} - {\tilde{y}^{(k)}}^3\right)\right] \end{aligned}$$

A tabela a seguir resume os resultados obtidos:

h		$t = 0{,}2$	$t = 0{,}4$	$t = 0{,}6$	$t = 0{,}8$	$t = 1{,}0$
10^{-2}	x	1.1986240	1.3890564	1.5654561	1.7287187	1.8874532
	y	0.2194288	0.4692676	0.7206154	0.9332802	1.0850012
10^{-3}	x	1.1986201	1.3890485	1.5654455	1.7287066	1.8874392
	y	0.2194293	0.4692707	0.7206252	0.9332999	1.0850259
10^{-4}	x	1.1986201	1.3890484	1.5653609	1.7287065	1.8874390
	y	0.2194293	0.4692707	0.7205062	0.9333001	1.0850262

A seguinte rotina pode ser usada para implementar a solução do sistema:

```
function [x,y]=euler_mod(h,Tmax,u1)
  itmax = Tmax/h;
  x=zeros(itmax+1);
  y=zeros(itmax+1);
  x(1)=u1(1);
  y(1)=u1(2);

  for n = 1:itmax
    t=(n-1)*h;
    kx1 = (x(n)-y(n));
    ky1 = (x(n)-y(n)^3);
    x_til = x(n) + h*kx1;
    y_til = y(n) + h*ky1;

    kx2 = (x_til-y_til);
    ky2 = (x_til-y_til^3);

    x(n+1)=x(n)+h*(kx1+kx2)/2;
    y(n+1)=y(n)+h*(ky1+ky2)/2;
  end
endfunction
```

Que pode ser invocada como:

```
h=1e-2
Tmax=1
itmax=Tmax/h
[x,y]=euler_mod(h,Tmax,[1,0]);
[x(1) x(1+itmax*.2) x(1+itmax*.4) x(1+itmax*.6) x(1+itmax*.8) x(1+itmax)]
[y(1) y(1+itmax*.2) y(1+itmax*.4) y(1+itmax*.6) y(1+itmax*.8) y(1+itmax)]
```

◊

1.4 Solução de equações e sistemas de ordem superior

Na Seção 1.3, estendemos os métodos de Euler e Euler melhorado visto nas seções 1.1 e 1.2 para resolver numericamente problemas de valor inicial envolvendo

sistemas de equações diferenciais ordinárias de primeira ordem. Nesta seção, estenderemos estas técnicas para resolver alguns tipos de problemas de ordem superior. Para tal, converteremos a equação diferencial em um sistema, incluindo as derivadas da incógnita como novas incógnitas. Vejamos um exemplo:

Exemplo 1.4.1. Resolva o problema de valor inicial de segunda ordem dado por

$$\begin{aligned} y'' + y' + y &= \cos(t), \\ y(0) &= 1, \\ y'(0) &= 0, \end{aligned}$$

A fim de transformar a equação diferencial dada em um sistema de equações de primeira ordem, introduzimos a substituição $w = y'$, de forma que obteremos o sistema:

$$\begin{aligned} y' &= w \\ w' &= -w - y + \cos(t) \\ y(0) &= 1 \\ w(0) &= 0 \end{aligned}$$

Este sistema pode ser resolvido usando as técnicas da Seção 1.3.

Exercícios resolvidos

ER 1.4.1. Considere o seguinte sistema envolvendo uma equação de segunda ordem e uma de primeira ordem:

$$\begin{aligned} x''(t) - (1 - 0{,}1z(t))x'(t) + x(t) &= 0 \\ 10z'(t) + z(t) &= x(t)^2 \end{aligned}$$

sujeito a condições iniciais dadas por:

$$\begin{aligned} x(0) &= 3 \\ x'(0) &= 0 \\ z(0) &= 10 \end{aligned}$$

Rescreva este sistema como um sistema de três equações de primeira ordem.

Solução. Definimos $y(t) = x'(t)$, pelo que o sistema se torna:

$$\begin{aligned} x'(t) &= y(t) \\ y'(t) - (1 - 0{,}1z(t))y(t) + x(t) &= 0 \\ 10z'(t) + z(t) &= x(t)^2 \end{aligned}$$

defina o vetor $u(t)$ como:

$$u(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix}$$

De forma que:

$$u'(t) = \begin{bmatrix} x'(t) \\ y'(t) \\ z'(t) \end{bmatrix} = \begin{bmatrix} y(t) \\ [1 - 0{,}1z(t)]\, y(t) - x(t) \\ [x(t)^2 - z(t)]\, /10 \end{bmatrix}$$

ou

$$u'(t) = \begin{bmatrix} u_1'(t) \\ u_2'(t) \\ u_3'(t) \end{bmatrix} = \begin{bmatrix} u_2(t) \\ [1 - 0{,}1u_3(t)]\, u_2(t) - u_1(t) \\ [u_1(t)^2 - u_3(t)]\, /10 \end{bmatrix}$$

sujeito às condições iniciais dadas por:

$$u'(0) = \begin{bmatrix} x'(0) \\ y'(0) \\ z'(0) \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \\ 10 \end{bmatrix}$$

$\Diamond$

Exercícios

E 1.4.1. Resolva o problema de valor inicial dado por

$$\begin{aligned} x' &= -2x + \sqrt{y} \\ y' &= x - y \\ x(0) &= 0 \\ y(0) &= 2 \end{aligned}$$

com passo $h = 2 \cdot 10^{-1}$ $h = 2 \cdot 10^{-2}$, $h = 2 \cdot 10^{-3}$ e $h = 2 \cdot 10^{-4}$ para obter aproximações para $x(2)$ e $y(2)$.

,

E 1.4.1.

h		$2 \cdot 10^{-2}$	$2 \cdot 10^{-2}$	$2 \cdot 10^{-3}$	$2 \cdot 10^{-4}$
Euler	x	0,4302019	0,4355057	0,4358046	0,4358324
	y	0,6172935	0,6457760	0,6486383	0,6489245
Euler mod,	x	0,4343130	0,4358269	0,4358354	0,4358355
	y	0,6515479	0,6489764	0,6489566	0,6489564

E 1.4.2. Considere o problema de segunda order dado por:

$$x''(t) + x'(t) + \operatorname{sen}(x(t)) = 1,$$

sujeito às condições iniciais dadas por:

$$\begin{aligned} x(0) &= 2, \\ x'(0) &= 0. \end{aligned}$$

Resolva numericamente para obter o valor de $x(0{,}5)$, $x(1)$, $x(1{,}5)$ e $x(2)$ com passo $h = 10^{-2}$ e $h = 10^{-3}$ via método de Euler modificado.

E 1.4.2.

h		$t = 0{,}5$	$t = 1{,}0$	$t = 1{,}5$	$t = 2{,}0$
10^{-3}	x	1,9023516	1,6564208	1,3124281	0,9168299
	y'	-0,3635613	-0,6044859	-0,7564252	-0,8072298
10^{-4}	x	1,9023552	1,6564243	1,3124309	0,9168319
	y'	-0,3635670	-0,6044930	-0,7564334	-0,8072397

1.5 Erro de truncamento

Nas seções 1.1 e 1.2, construimos dois métodos numéricos para resolver problemas de valor inicial. No Exercício Resolvido 1.2.1, vimos que o erro do método de Euler de do método de Euler melhorado caem quando se reduz o passo h, ademais, o erro do método de Euler melhorado cai conforme o quadrado de h, enquando o do método de Euler cai conforme h^2. Este fenômeno motiva a definção de **ordem de precisão**.

Definição 1.5.1. *O **erro de truncamento local** é definido como o erro introduzido em cada passo pelo truncamento da equação diferencial supondo conhecida a solução exata no início do intervalo. Um método numérico é dito ter **ordem de precisão** p se o erro de truncamento local for da ordem de h^{p+1}.*

Exemplo 1.5.1. O método de Euler tem erro de truncamento local de ordem 1. Para obter este resultado, observamos via expansão de Taylor que:

$$u(t+h) = u(t) + hu'(t) + \frac{h^2}{2}u''(t) + O(h^3).$$

Se escolhermos nesta expressão $t = t^{(n)}$ e, portanto, $t + h = t^{(n)} + h = t^{(n+1)}$, temos:

$$t^{(n+1)} = t^{(n)} + hu'(t^{(n)}) + \frac{h^2}{2}u''(t^{(n)}) + O(h^3)$$

Agora notamos que o termo principal do erro é dado por $\frac{h^2}{2}u''(t^{(n)})$, como a derivada segunda da solução não depende de h, o erro local de truncamento decresce conforme h^2 e assim a ordem de precisão do método é 1.

Definição 1.5.2. *O **erro de truncamento global** é definido como erro acumulado ao longo de todos os passo de resolução, supondo a condição inicial exata.*

A relação entre o erro de truncamento global e o erro de truncamento local depende da função $f(t,u)$ envolvida, sobre suficiente regularidade, o erro acumulado é da mesma ordem de grande do erro de truncamento local multiplicado pelo número de passos. Como o número de passos N necessários para calcular a solução de um problema de valor inicial no ponto $t = t_f$ é dado por $N = \frac{t_f}{h}$, temos que a erro de truncamento global é uma ordem inferior ao erro de truncamento local e equivale à ordem de precisão do método.

Usamos também a notação ETL para o erro de truncamento local e ETG para o erro de truncamento global. De forma que, para o método de Euler, temos:

$$ETL_{Euler} = O(h^2) \quad \text{e} \quad ETG_{Euler} = O(h).$$

Exemplo 1.5.2. Vamos obter o erro de truncamento local do método de Euler melhorado. Partimos da construção do esquema iterativo de Euler melhorado:

$$\int_{t^{(1)}}^{t^{(2)}} u'(t)\,dt = \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\,dt \tag{1.50}$$

$$u(t^{(2)}) - u(t^{(1)}) = \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\,dt \tag{1.51}$$

$$u(t^{(2)}) = u(t^{(1)}) + \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\,dt \tag{1.52}$$

Neste ponto, usamos o erro de truncamento do método de trapézios para aproximar a integral envolvida:

$$\int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\,dt = \frac{h}{2}\left[f\left(t^{(1)},u(t^{(1)})\right) + f\left(t^{(2)},u(t^{(2)})\right)\right] + O(h^3)$$

Assim, temos que o erro de truncamento local do método de Euler melhorado é $O(h^3)$ e, portanto, um método de ordem 2.

E 1.5.1. Aplique o método de Euler e o método de Euler melhorado para resolver o problema de valor inicial dado por

$$\begin{aligned} u' &= -2u + \sqrt{u} \\ u(0) &= 1 \end{aligned}$$

com passo $h = 10^{-1}$, $h = 10^{-2}$, $h = 10^{-3}$, $h = 10^{-4}$ e $h = 10^{-5}$ para obter aproximações para $u(1)$. Compare com a solução exata dada do problema dada por $u(t) = \left(1 + 2e^{-t} + e^{-2t}\right)/4$ através do erro relativo e observe a ordem de precisão do método.

E 1.5.1.

h	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
Euler	0,4495791	0,4660297	0,4675999	0,4677562	0,4677718
ε_{rel}	9,1e-03	8,9e-04	8,9e-05	8,9e-06	8,9e-07
Euler mod,	0,4686037	0,4677811	0,4677736	0,4677735	0,4677735
ε_{rel}	1,8e-03	1,6e-05	1,6e-07	1,6e-09	1,6e-11

A solução exata vale $u(1) = \frac{1+2e^{-1}+e^{-2}}{4} = \left(\frac{1+e^{-1}}{2}\right)^2 \approx 0.467773541395$.

E 1.5.2. Resolva o problema de valor inicial dado por

$$\begin{aligned} u' &= \cos(tu(t)) \\ u(0) &= 1 \end{aligned}$$

com passo $h = 10^{-1}$, $h = 10^{-2}$, $h = 10^{-3}$, $h = 10^{-4}$ e $h = 10^{-5}$ para obter aproximações para $u(2)$

E 1.5.2.

h	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
Euler	1,1617930	1,1395726	1,1374484	1,1372369	1,1372157
Euler mod	1,1365230	1,1372075	1,1372133	1,1372134	1,1372134

1.6 Métodos de Runge-Kutta explícitos

Nas seções anteriores, exploramos os métodos de Euler e Euler modificado para resolver problemas de valor inicial. Neste momento, deve estar claro ao leitor que o método de Euler melhorado produz soluções de melhor qualidade que o método de Euler para a maior parte dos problemas estudados. Isso se deve, conforme Seção 1.5, à ordem de precisão do método.

Os métodos de Runge-Kutta generalizam o esquema do método de Euler melhorado, inserindo mais estágios de cálculo e buscando ordens de precisão mais altas. Nesta seção, trataremos dos métodos de **Runge-Kutta explícitos**[1]

Para tal, considere o problema de valor inicial:

$$u'(t) = f(t,u(t)), \tag{1.53}$$
$$u(t_0) = a. \tag{1.54}$$

Integrando a EDO em $[t^{(n)},t^{(n+1)}]$ obtemos

$$u^{(n+1)} = u^{(n)} + \int_{t^{(n)}}^{t^{(n+1)}} f(t,u(t))\,dt \tag{1.55}$$

O método de Euler aproxima a integral no lado direito da expressão acima utilizando apenas o valor de $f(t,u)$ em $t = t^{(n)}$, o método de Euler melhorado aproxima esta integral utilizando os valores de $f(t,u)$ $t = t^{(n)}$ e $t = t^{(n+1)}$. Os métodos de Runge-Kutta explícitos admitem estágios intermediários, utilizando outros valores de $f(t,u)$ nos pontos $\{\tau_1,\tau_2,\ldots,\tau_\nu\}$ dentro do intervalo $[t_n,t^{(n+1)}]$. Veja esquema abaixo:

u_n			u_{n+1}
|			|
$t^{(n)}$			$t^{(n+1)}$
τ_1	τ_2	$\cdots$	τ_ν
$t^{(n)}$	$t^{(n)} + c_2h$	$\cdots$	$t^{(n)} + c_\nu h$

Observe que $\tau_j = t^{(n)} + c_jh$ com $0 = c_1 \leq c_2 \leq \cdots \leq c_\nu \leq 1$, isto é, o primeira ponto sempre coincide com o extremo esquerdo do intervalo, mas o último ponto não precisa ser o extremo direito. Ademais, um mesmo ponto pode ser usado mais de uma vez com aproximações diferentes para $u(\tau_j)$.

Desta forma, aproximamos a integral por um esquema de quadratura com ν pontos:

$$\int_{t^{(n)}}^{t^{(n+1)}} f(t,u(t))\,dt \approx u_n + h\sum_{j=1}^{\nu} b_j f\left(\tau_j,u(\tau_j)\right) \tag{1.56}$$

onde b_j são os pesos da quadratura numérica que aproxima a integral. Assim como na construção do método de Euler melhorado, não dispomos dos valores $u(\tau_j)$ antes

[1] Existem também os métodos implícitos que serão abordados na Seção 1.7. Ver Observação 1.6.1.

de calculá-los e, por isso, precisamos estimá-los com base nos estágios anteriores:

$$\begin{aligned}
\tilde{u}_1 &= u^{(n)} && (1.57)\\
\tilde{u}_2 &= u_n + ha_{21}k_1 && (1.58)\\
\tilde{u}_3 &= u_n + h\left[a_{31}k_1 + a_{32}k_2\right] && (1.59)\\
\tilde{u}_4 &= u_n + h\left[a_{41}k_1 + a_{42}k_2 + a_{43}k_3\right] && (1.60)\\
&\vdots && (1.61)\\
\tilde{u}_\nu &= u_n + h\left[a_{\nu 1}k_1 + a_{\nu 2}k_2 + \cdots + a_{\nu\nu}k_\nu\right] && (1.62)\\
u^{(n+1)} &= u_n + h[b_1k_1 + b_2k_2 + \cdots + b_\nu k_\nu], && (1.63)
\end{aligned}$$

onde $k_j = f\left(\tau_j, \tilde{u}_j\right)$, $A := (a_{ij})$ é a matriz Runge-Kutta (triangular inferior com diagonal zero), b_j são os pesos Runge-Kutta e c_j são os nós Runge-Kutta. Estes coeficientes podem ser escritos de forma compacta em uma tabela conforme a seguir:

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array} = \begin{array}{c|ccc} c_1 & 0 & 0 & 0 \\ c_2 & a_{21} & 0 & 0 \\ c_3 & a_{31} & a_{22} & 0 \\ \hline & b_1 & b_2 & b_3. \end{array} = \begin{array}{c|ccc} c_1 & & & \\ c_2 & a_{21} & & \\ c_3 & a_{31} & a_{22} & \\ \hline & b_1 & b_2 & b_3. \end{array}$$

Na tabela mais à direita, omitimos os termos obrigatoriamente nulos.

Exemplo 1.6.1. O método de Euler modificado pode ser escrito conforme:

$$\begin{aligned}
\tilde{u}_1 &= u^{(n)}\\
\tilde{u}_2 &= u^{(n)} + hk_1\\
u^{(n+1)} &= u^{(n)} + h\left[\frac{1}{2}k_1 + \frac{1}{2}k_2\right]
\end{aligned}$$

Identificando os coeficientes, obtemos $\nu = 2$, $c_1 = 0$, $c_2 = 1$, $a_{21} = 1$, $b_1 = \frac{1}{2}$ e $b_2 = \frac{1}{2}$. Escrevendo na forma tabular, temos:

$$\begin{array}{c|c} c & A \\ \hline & b \end{array} = \begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Observação 1.6.1. Nos métodos chamados explícitos, os elementos da diagonal principal (e acima dela) da matriz A devem ser nulos, pois a aproximação em cada

estágio é calculada com base nos valores dos estágios anteriores. Nos métodos implícitos, essa restrição é removida, neste caso, o cálculo da aproximação em um estágio envolve a solução de uma equação algébrica.

Observação 1.6.2. Além da condição fixa $c_1 = 0$, para que um método seja de ordem pelo menos unitária, isto é, $p \geq 1$.

1.6.1 Métodos de Runge-Kutta com dois estágios

Os métodos de Runge-Kutta com dois estágios ($\nu = 2$) são da seguinte forma:

$$\begin{aligned} \tilde{u}_1 &= u^{(n)} && (1.64)\\ \tilde{u}_2 &= u^{(n)} + ha_{21}k_1 && (1.65)\\ u^{(n+1)} &= u^{(n)} + h[b_1k_1 + b_2k_2], && (1.66) \end{aligned}$$

onde $k_1 = f(t^{(n)},u^{(n)})$ e $k_2 = f(t^{(n)} + c_2h,\tilde{u}_2)$

Assumindo suavidade suficiente em f, usamos o polinônio de Taylor:

$$\begin{aligned} k_2 &= f(t^{(n)} + c_2h, \tilde{u}_2) && (1.67)\\ &= f(t^{(n)} + c_2h, u^{(n)} + a_{21}hk_1) && (1.68)\\ &= f(t^{(n)}, u^{(n)}) + h\left[c_2\frac{\partial f}{\partial t} + a_{21}k_1\frac{\partial f}{\partial u}\right] + O(h^2) && (1.69)\\ &= k_1 + h\left(c_2\frac{\partial f}{\partial t} + a_{21}k_1\frac{\partial f}{\partial u}\right) + O(h^2) && (1.70) \end{aligned}$$

fazendo com que (1.66) se torne

$$\begin{aligned} u^{(n+1)} &= u^{(n)} + hb_1k_1 + hb_2k_2 && (1.71)\\ &= u_n + hb_1k1 + hb_2\left[k_1 + h\left(c_2\frac{\partial f}{\partial t} + a_{21}k_1\frac{\partial f}{\partial u}\right) + O(h^2)\right] && (1.72)\\ &= u_n + h(b_1 + b_2)k_1 + h^2b_2\left(c_2\frac{\partial f}{\partial t} + a_{21}k_1\frac{\partial f}{\partial u}\right) + O(h^3) && (1.73) \end{aligned}$$

Usando a equação diferencial ordinária que desejamos resolver e derivando-a em t, obtemos:

$$\begin{aligned} u'(t) &= f(t,u(t)), && (1.74)\\ u''(t) &= \frac{\partial f}{\partial t} + \frac{\partial f}{\partial u}u'(t) = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial u}f(t,u(t)). && (1.75) \end{aligned}$$

Agora, expandimos em série de Taylor a solução exata $u(t)$ em $t = t^{(n)}$,

$$\begin{aligned}
u(t^{(n+1)}) &= u(t^{(n)} + h) = u^{(n)} + hu'(t) + \frac{h^2}{2}u''(t) + O(h^3) && (1.76)\\
&= u^{(n)} + hf(t,u^{(n)}) + \frac{h^2}{2}\frac{d}{dt}f(t,u(t)) + O(h^3) && (1.77)\\
&= u^{(n)} + hf(t,u^{(n)}) + \frac{h^2}{2}\left(\frac{\partial f}{\partial t} + \frac{\partial f}{\partial u}u'(t^{(n)})\right) + O(h^3) && (1.78)\\
&= u^{(n)} + hf(t,u^{(n)}) + \frac{h^2}{2}\left(\frac{\partial f}{\partial t} + \frac{\partial f}{\partial u}f(t,u^{(n)})\right) + O(h^3) && (1.79)\\
&= u^{(n)} + hf(t,u^{(n)}) + \frac{h^2}{2}\left(\frac{\partial f}{\partial t} + \frac{\partial f}{\partial u}k_1\right) + O(h^3) && (1.80)
\end{aligned}$$

Finalmente comparamos os termos em (1.73) e (1.80) de forma a haver concordância na expansão de Taylor até segunda ordem, isto é, restanto apenas o erro de ordem 3 e produzindo um método de ordem de precisão $p = 2$:

$$b_1 + b_2 = 1, \quad b_2c_2 = \frac{1}{2} \quad \text{e} \quad a_{21} = c_2 \tag{1.81}$$

Este sistema é formada por três equações e quatro incógnitas, pelo que admite infinitas soluções. Para construir toda a família de soluções, escolha um parâmetro $\alpha \in (0,1]$ e defina a partir de (1.81):

$$b_1 = 1 - \frac{1}{2\alpha}, \quad b_2 = \frac{1}{2\alpha}, \quad c_2 = \alpha, \quad a_{21} = \alpha$$

Portanto, obtemos o seguinte esquema genérico:

$$\begin{array}{c|c} c & A \\ \hline & b \end{array} = \begin{array}{c|cc} c_2 & a_{21} & \\ \hline & b_1 & b_2 \end{array} = \begin{array}{c|cc} \alpha & \alpha & \\ \hline & \left(1-\frac{1}{2\alpha}\right) & \frac{1}{2\alpha} \end{array}, \quad 0 < \alpha \leq 1.$$

Algumas escolhas comuns são $\alpha = \frac{1}{2}$, $\alpha = \frac{2}{3}$ e $\alpha = 1$:

$$\begin{array}{c|cc} 0 & & \\ \frac{1}{2} & \frac{1}{2} & \\ \hline & 0 & 1 \end{array}, \qquad \begin{array}{c|cc} 0 & & \\ \frac{2}{3} & \frac{2}{3} & \\ \hline & \frac{1}{4} & \frac{3}{4} \end{array} \quad \text{e} \quad \begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}.$$

Note que a tabela da direita fornece o método Euler modificado ($\alpha = 1$). O esquema iterativo assume a seguinte forma:

$$\begin{aligned} \tilde{u}_1 &= u^{(n)} && (1.82)\\ \tilde{u}_2 &= u^{(n)} + h\alpha k_1 && (1.83)\\ u^{(n+1)} &= u^{(n)} + h\left[\left(1 - \frac{1}{2\alpha}\right)k_1 + \frac{1}{2\alpha}k_2\right], && (1.84) \end{aligned}$$

onde $k_1 = f(t^{(n)},u^{(n)})$ e $k_2 = f(t^{(n)} + h\alpha,\tilde{u}_2)$. Ou, equivalentemente:

$$\begin{aligned} k_1 &= f(t^{(n)},u^{(n)}) && (1.85)\\ k_2 &= f(t^{(n)} + h\alpha,u^{(n)} + h\alpha k_1) && (1.86)\\ u^{(n+1)} &= u^{(n)} + h\left[\left(1 - \frac{1}{2\alpha}\right)k_1 + \frac{1}{2\alpha}k_2\right], && (1.87) \end{aligned}$$

1.6.2 Métodos de Runge-Kutta com três estágios

Os métodos de Runge-Kutta com 3 estágios podem ser descritos na forma tabular como:

$$\begin{array}{c|ccc} 0 & & & \\ c_2 & a_{21} & & \\ c_3 & a_{31} & a_{32} & \\ \hline & b_1 & b_2 & b_3 \end{array}$$

Seguindo um procedimento similar ao da Seção 1.6.1, podemos obter as condições equivalentes às condições (1.81) para um método com $\nu = 3$ e ordem $p = 3$, as quais são:

$$\begin{aligned} b_1 + b_2 + b_3 &= 1, && (1.88)\\ b_2c_2 + b_3c_3 &= \frac{1}{2}, && (1.89)\\ b_2c_2^2 + b_3c_3^2 &= \frac{1}{3}, && (1.90)\\ b_3a_{32}c_2 &= \frac{1}{6}, && (1.91)\\ a_{21} &= c_2, && (1.92)\\ a_{31} + a_{32} &= c_3. && (1.93) \end{aligned}$$

Assim, temos 6 condições para determinar 8 incógnitas, o que implica a existência de uma enorme família de métodos de Runge-Kutta com três estágios e ordem

$p = 3$. Se fixarmos os coeficientes c_2 e c_3, podemos os outros de forma única:

$$\begin{aligned} b_1 &= 1 - \frac{1}{2c_2} - \frac{1}{2c_3} + \frac{1}{3c_2c_3} \\ b_2 &= \frac{3c_3 - 2}{6c_2(c_3 - c_2)} \\ b_3 &= \frac{2 - 3c_2}{6c_3(c_3 - c_2)} \\ a_{21} &= c_2 \\ a_{31} &= c_3 - \frac{1}{6b_3c_2} \\ a_{32} &= \frac{1}{6b_3c_2} \end{aligned}$$

Alguns exemplos de métodos de Runge-Kutta de 3 estágios são o método clássico de Runge-Kutta $\left(c_2 = \frac{1}{2} \text{ e } c_3 = 1\right)$ e o método de Nyström $\left(c_2 = c_3 = \frac{2}{3}\right)$:

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ 1 & -1 & 2 & \\ \hline & \frac{1}{6} & \frac{4}{6} & \frac{1}{6} \end{array} \quad \text{e} \quad \begin{array}{c|ccc} 0 & & & \\ \frac{2}{3} & \frac{2}{3} & & \\ \frac{2}{3} & 0 & \frac{2}{3} & \\ \hline & \frac{2}{8} & \frac{3}{8} & \frac{3}{8} \end{array} .$$

1.6.3 Métodos de Runge-Kutta com quatro estágios

As técnicas utilizadas nas seções 1.6.1 e 1.6.2 podem ser usadas para obter métodos de quarta ordem ($p = 4$) e quatro estágios ($\nu = 4$). As seguintes tabelas descrevem os dois esquemas mais conhecidos de Runge-Kutta quarta ordem com quatro estágios. O primeiro é denominado **método de Runge-Kutta 3/8** e o segundo é chamado de **método de Runge-Kutta clássico**.

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{3} & \frac{1}{3} & & & \\ \frac{2}{3} & -\frac{1}{3} & 1 & & \\ 1 & 1 & -1 & 1 & \\ \hline & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} \end{array} \quad \text{e} \quad \begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6} \end{array}$$

O método de Runge-Kutta clássico é certamente o mais notório dos métodos de Runge-Kutta e seu esquema iterativo pode ser escrito como a seguir:

$$\begin{aligned} k_1 &= f\left(t^{(n)},u^{(n)}\right) \\ k_2 &= f\left(t^{(n)}+h/2,u^{(n)}+k_1/2\right) \\ k_3 &= f\left(t^{(n)}+h/2,u^{(n)}+k_2/2\right) \\ k_4 &= f\left(t^{(n)}+h,u^{(n)}+k_3\right) \\ u^{(n+1)} &= u^{(n)}+h\frac{k_1+2k_2+2k_3+k_4}{6} \end{aligned}$$

A seguinte heurística, usando o método de Simpson para quadratura numérica, pode ajudar a compreender os estranhos coeficientes:

$$\begin{aligned} u(t^{(n+1)})-u(t^{(n)}) &= \int_{t^{(n)}}^{t^{(n+1)}} f(t,u(s))ds \\ &\approx \frac{h}{6}\left[f\left(t^{(n)},u(t^{(n)})\right)+4f\left(t^{(n)}+h/2,u(t^{(n)}+h/2)\right)\right. \\ &+ \left. f\left(t^{(n)}+hu(t^{(n)}+h)\right)\right] \\ &\approx h\frac{k_1+4(\frac{k_2+k_3}{2})+k_4}{6} \end{aligned}$$

onde k_1 e k_4 representam os valores de $f(t,u)$ nos extremos; k_2 e k_3 são duas aproximações diferentes para a inclinação no meio do intervalo.

Exercícios resolvidos

ER 1.6.1. Construa o esquema iterativo o método clássico de Runge-Kutta três estágios cuja tabela é dada a seguir:

0			
$\frac{1}{2}$	$\frac{1}{2}$		
1	-1	2	
	$\frac{1}{6}$	$\frac{4}{6}$	$\frac{1}{6}$

Solução.

$$\begin{aligned} k_1 &= f\left(t^{(n)},u^{(n)}\right) \\ k_2 &= f\left(t^{(n)}+h/2,u^{(n)}+k_1/2\right) \\ k_3 &= f\left(t^{(n)}+h,u^{(n)}-k_1+2k_2\right) \\ u^{(n+1)} &= u^{(n)}+h\frac{k_1+4k_2+k_4}{6} \end{aligned}$$

$\Diamond$

ER 1.6.2. Utilize o método clássico de Runge-Kutta três estágios para calcular o valor de $u(2)$ com passos $h = 10^{-1}$ e $h = 10^{-2}$ para o seguinte problema de valor inicial:

$$\begin{aligned} u'(t) &= -u(t)^2 + t, \\ u(0) &= 0. \end{aligned}$$

Aplicando o processo iterativo obtido no Problema Resolvido 1.6.1, obtemos a seguinte rotina:

```
function [y]= f(t,u)
    y= t-u**2
endfunction


function [u] = RK3_classico(h,Tmax,u1)
    itmax = Tmax/h;
    u=zeros(itmax+1)
    u(1)=u1

    for i = 1:itmax
        t=(i-1)*h
        k1 = f(t, u(i))
        k2 = f(t+h/2, u(i) + h*k1/2)
        k3 = f(t+h,   u(i) + h*(2*k2-k1))

        u(i+1) = u(i) + h*(k1+4*k2+k3)/6
    end
endfunction
```

A qual pode ser invocada com:

```
->sol=RK3_classico(1e-2,2,0);sol(201)
 ans  =

    1.1935760016451

-->sol=RK3_classico(1e-3,2,0);sol(2001)
 ans  =

    1.1935759753635
```

Exercícios

E 1.6.1. Aplique o esquema de Runge-Kutta segunda ordem com dois estágios cujos coeficientes são dados na tabela a seguir

0		
$\frac{2}{3}$	$\frac{2}{3}$	
	$\frac{1}{4}$	$\frac{3}{4}$

para resolver o problema de valor inicial dado por:

$$\begin{aligned} x'(t) &= \operatorname{sen}(x(t)), \\ x(0) &= 2. \end{aligned}$$

para $t = 2$ com $h = 1e-1$, $h = 1e-2$ e $h = 1e-3$. Expresse sua resposta com oito dígitos significativos corretos.

E 1.6.1. 2,9677921, 2,9682284 e 2,9682325.

E 1.6.2.

E 1.6.3. Resolva pelo método de Euler, Euler melhorado, Runge-Kutta clássico três estágios e Runge-Kutta clássico quatro estágios o problema de valor inicial tratados nos exercícios resolvidos 1.1.1 e 1.2.1 dado por:

$$u'(t) = -0{,}5u(t) + 2 + t \tag{1.94}$$

$$u(0) = 8 \tag{1.95}$$

Usando os seguinte passos: $h = 1$, $h = 10^{-1}$, $h = 10^{-2}$ e $h = 10^{-3}$ e compare a solução aproximada em $t = 1$ com as soluções obtidas com a solução exata dada por:

$$u(t) = 2t + 8e^{-t/2} \Longrightarrow u(1) = 2 + 8e^{-1/2} \approx 6{,}85224527770107 \tag{1.96}$$

E 1.6.3.

Euler	6,0000000	6,7898955	6,8461635	6,8516386
ε_{rel}	1,2e-01	9,1e-03	8,9e-04	8,9e-05
Euler mod,	7,0000000	6,8532949	6,8522554	6,8522454
ε_{rel}	2,2e-02	1,5e-04	1,5e-06	1,5e-08
RK_3	6,8333333	6,8522321	6,8522453	6,8522453
ε_{rel}	2,8e-03	1,9e-06	1,9e-09	1,8e-12
RK_4	6,8541667	6,8522454	6,8522453	6,8522453
ε_{rel}	2,8e-04	1,9e-08	1,9e-12	1,3e-15

E 1.6.4. Aplique o método de Euler, o método de Euler melhorado, o método clássico de Runge-Kutta três estágios e o método clássico de Runge-Kutta quatro

estágios para resolver o problema de valor inicial dado por

$$\begin{aligned} u' &= u + t \\ u(0) &= 1 \end{aligned}$$

com passo $h = 1$, $h = 10^{-1}$, $h = 10^{-2}$ e $h = 10^{-3}$ para obter aproximações para $u(1)$. Compare com a solução exata dada do problema dada por $u(t) = 2e^t - t - 1$ através do erro relativo e observe a ordem de precisão do método. Expresse a sua resposta com oito dígitos significativos para a solução e 2 dígitos significativos para o erro relativo.

E 1.6.4.

Euler	2,0000000	3,1874849	3,4096277	3,4338479
ε_{rel}	4,2e-01	7,2e-02	7,8e-03	7,9e-04
Euler mod	3,0000000	3,4281617	3,4364737	3,4365628
ε_{rel}	1,3e-01	2,4e-03	2,6e-05	2,6e-07
RK_3	3,3333333	3,4363545	3,4365634	3,4365637
ε_{rel}	3,0e-02	6,1e-05	6,5e-08	6,6e-11
RK_4	3,4166667	3,4365595	3,4365637	3,4365637
ε_{rel}	5,8e-03	1,2e-06	1,3e-10	1,2e-14

1.7 Métodos de Runge-Kutta implícitos

até aqui

1.8 O método de Euler implícito

Integrando o problema de valor inicial

$$u'(t) = f(t,u(t)) \tag{1.97}$$
$$u(t^{(1)}) = a \tag{1.98}$$

de $t^{(1)}$ até $t^{(2)}$ obtemos (como feito anteriormente)

$$u(t^{(2)}) = u(t^{(1)}) + \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\, dt \tag{1.99}$$

Entretanto se aproximarmos a função f por uma função constante $f(t,u(t)) \approx f(t^{(2)},u^{(2)})$, obteremos um novo método

$$u^{(2)} = u^{(1)} + f(t^{(2)},u^{(2)}) \int_{t^{(1)}}^{t^{(2)}} dt \tag{1.100}$$
$$u^{(2)} = u^{(1)} + hf(t^{(2)},u^{(2)}) \tag{1.101}$$

Generalizando este procedimento para t_n obtemos o **método de Euler implícito**

$$u_{n+1} = u_n + h\, f(t^{(n+1)}, u_{n+1}). \tag{1.102}$$

Note que este método é **implícito** (a equação é implícita) pois depende de u_{n+1} dos dois lados da equação. Se a função f for simples o suficiente, podemos resolver a equação isolando o termo u_{n+1}. Se isso não for possível, devemos usar um dos métodos vistos anteriormente para calcular as raízes da equação (por exemplo, método da bissecção e método de Newton).

Pode ser mostrado que o erro de truncamento local é

$$ETL^{n+1}_{EulImp} = \mathcal{O}(h^2).$$

portanto o método é de ordem 1. E o erro de truncamento global é

$$ETG^{n+1}_{EulImp} = \mathcal{O}(h).$$

Exemplo 1.8.1. Utilizando o **método de Euler implícito** para solucionar (2.18) obtemos

$$\begin{aligned}
u_{n+1} &= u_n + h\lambda u_{n+1}, && (1.103)\\
(1-h\lambda)u_{n+1} &= u_n, && (1.104)\\
u_{n+1} &= \left(\frac{1}{1-h\lambda}\right) u_n, && (1.105)\\
u_{n+1} &= \left(\frac{1}{1-h\lambda}\right)^2 u_{n-1}, && (1.106)\\
u_{n+1} &= \left(\frac{1}{1-h\lambda}\right)^{n+1}, \quad n = 0,1,\ldots && (1.107)
\end{aligned}$$

onde $u^{(1)} = 1$. Concluímos então que

$$\mathcal{D}_{EulImp} = \{z \in \mathbb{C} : \left|\frac{1}{1-z}\right| < 1\} \tag{1.108}$$

ou ainda,

$$\mathcal{D}_{EulImp} = \{z \in \mathbb{C} : |1-z| > 1\} \tag{1.109}$$

Capítulo 2

Em reestruturação

2.0.1 Ordem de precisão

A **precisão** de um método numérico que aproxima a solução de um problema de valor inicial é dada pela ordem do erro acumulado ao calcular a aproximação em um ponto $t^{(n+1)}$ em função do espaçamento da malha h.

Se $u(t^{(n+1)})$ for aproximado por u_{n+1} com erro da ordem $O(h^{p+1})$ dizemos que o método tem **ordem de precisão** p.

Queremos obter a ordem de precisão do método de Euler. Para isso, substituímos a EDO $u' = f(t,u)$ na expansão em série de Taylor

$$u(t^{(n+1)}) = u(t_n) + hu'(t_n) + h^2u''(t_n)/2 + \mathcal{O}(h^3) \tag{2.1}$$

e obtemos

$$u(t^{(n+1)}) = u(t_n) + hf(t_n,u(t_n)) + h^2u''(t_n)/2 + \mathcal{O}(h^3) \tag{2.2}$$

Subtraindo (2.2) do método de Euler

$$u_{n+1} = u_n + h\ f(t_n,u_n) \tag{2.3}$$

obtemos

$$\begin{aligned} e_{n+1} &= u_{n+1} - u(t^{(n+1)}) && (2.4)\\ &= u_n - u(t_n) + h(f(t_n,u(t_n) + e_n) - f(t_n,u(t_n))) + && (2.5)\\ &+ \frac{h^2}{2}u''_n + \mathcal{O}(h^3) && (2.6) \end{aligned}$$

Defina o **erro numérico** como $e_n = u_n - u(t_n)$ onde $u(t_n)$ é a solução exata e u_n é a solução aproximada. Assim

$$e_{n+1} = e_n + h(f(t_n,u(t_n) + e_n) - f(t_n,u(t_n))) + \tfrac{h^2}{2}u''_n + \mathcal{O}(h^3) \tag{2.7}$$

Usando a condição de Lipschitz em f temos

$$\begin{aligned} |e_{n+1}| &\leq |e_n| + h|f(t_n,u(t_n)+e_n) - f(t_n,u(t_n))| + \frac{h^2}{2}|u_n''| + \mathcal{O}(h^3) && (2.8)\\ &\leq |e_n| + hL|u(t_n)+e_n-u(t_n)| + \frac{h^2}{2}|u_n''| + \mathcal{O}(h^3) && (2.9)\\ &\leq |e_n| + hL|e_n| + \frac{h^2}{2}|u_n''| + \mathcal{O}(h^3) && (2.10)\\ &\leq (1+hL)|e_n| + \frac{h^2}{2}|u_n''| + \mathcal{O}(h^3) && (2.11) \end{aligned}$$

2.0.2 Erro de truncamento local

O **erro de truncamento local** é o erro cometido em **uma** iteração do método numérico supondo que a solução exata é conhecida no passo anterior.

Assim, supondo que a solução é exata em t_n ($|e_n| = 0$), obtemos que o ETL é

$$ETL_{Euler}^{n+1} = h^2/2|u''| + \mathcal{O}(h^3) = \mathcal{O}(h^2)$$

Como o $ETL = \mathcal{O}(h^2)$ temos que o método de Euler possui ordem 1.

2.0.3 Erro de truncamento global

O **erro de truncamento global** é o erro cometido durante **várias** iterações do método numérico.

Supondo que a solução exata é conhecida em $t^{(1)}$ ($\|e_1\| = 0$), então realizando $n = \frac{T}{h}$ iterações obtemos

$$\begin{aligned} ETG &= nETL && (2.12)\\ &= n[h^2/2|u''| + \mathcal{O}(h^3)] && (2.13)\\ &= Th/2|u''| + \mathcal{O}(h^2) && (2.14) \end{aligned}$$

ou seja

$$ETG_{Euler}^{n+1} = O(h)$$

Exercícios resolvidos

Em construção ... Gostaria de participar na escrita deste livro? Veja como em:

http://www.ufrgs.br/numerico/participe.html

Exercícios

2.1 Convergência, consistência e estabilidade

Nesta seção veremos três conceitos fundamentais em análise numérica: convergência, consistência e estabilidade.

2.1.1 Convergência

Um método é dito **convergente** se para toda EDO com f Lipschitz e todo $t > 0$ temos que

$$\lim_{h\to 0} |u_n - u(t_n)| = 0, \qquad \forall n$$

Convergência significa que a solução numérica tende a solução do problema de valor inicial.

Teorema 2.1.1. *O método de Euler é convergente.*

De fato, se f é Lipschitz-contínua e $|e_0| = 0$, temos que

$$\lim_{h\to 0} |e_{n+1}| \quad = \lim_{h\to 0} O(h) = 0 \tag{2.15}$$

2.1.2 Consistência

Definição 2.1.1. *Dizemos que um método numérico $R_h(u_n) = f$ é consistente com o problema de valor inicial $u'(t) = f$ se para qualquer $u(t)$*

$$\lim_{h\to 0} |u'(t_n) - R_h(u_n)| = 0, \quad \forall n \tag{2.16}$$

Isto é equivalente a

$$\lim_{h\to 0} \frac{ETL}{h} = 0 \tag{2.17}$$

2.1.3 Estabilidade

Definição 2.1.2. *Um método numérico é* ***estável*** *se*

$$|u_n - v_n| \leq C_1 |u^{(1)} - v_1|, \quad \forall n$$

Isto significa que dadas duas condições iniciais $u^{(1)}$ e v_1, teremos que as soluções u_n e v_n estarão a uma distância limitada por uma constante C_1 vezes $|u^{(1)} - v_1|$. Se $u^{(1)}$ e v_1 estiverem próximas então u_n e v_n estão também próximas dependendo da constante C_1 (obviamente C_1 depende da função f).

Considere o problema de valor inicial linear bem-posto

$$u'(t) = \lambda u(t), \quad u(0) = 1, \tag{2.18}$$

onde $\lambda \in \mathbb{C}$. Note que:

- Possui solução exata $u(t) = e^{\lambda t}$.
- O problema de valor inicial é **assintoticamente estável**, isto é, $\lim_{t\to\infty} u(t) = 0$, se e somente se $\Re\lambda < 0$.

Definição 2.1.3. *O **domínio de estabilidade linear** $\mathcal{D}$ do método numérico é o conjunto de todos $h\lambda \in \mathbb{C}$ tal que $\lim_{n\to\infty} u_n = 0$.*

Ou seja, $\mathcal{D}$ é o conjunto de todos $h\lambda$ para o qual o correto comportamento assintótico de (2.18) seja recuperado.

Exemplo 2.1.1. Utilizando o **Método de Euler** para solucionar (2.18) obtemos ($u^{(1)} = 1$)

$$\begin{aligned} u_{n+1} &= u_n + h\lambda u_n, && (2.19)\\ u_{n+1} &= (1+h\lambda)u_n, && (2.20)\\ u_{n+1} &= (1+h\lambda)^2 u_{n-1}, && (2.21)\\ u_{n+1} &= (1+h\lambda)^{n+1} u^{(1)} && (2.22)\\ u_{n+1} &= (1+h\lambda)^{n+1}, \quad n = 0,1,\ldots && (2.23) \end{aligned}$$

Para que o método de Euler seja estável, é necessário que h seja escolhido tal que $|1 + h\lambda| < 1$. Ou seja, $h\lambda$ deve estar em $\mathcal{D}_{Euler}$ onde

$$\mathcal{D}_{Euler} = \{z \in \mathbb{C} : |1 + z| < 1\} \tag{2.24}$$

é o interior de um disco no plano complexo de raio 1 e centro em $z = -1$.

Tal análise pode ser facilmente estendida para $u' = \lambda u + b$ (veja exercícios).

Para o caso EDO não linear, seja

$$u' = f(t,u), \quad t \geq t_0, \quad u(t_0) = u_0 \tag{2.25}$$

é comum requerer que $h\lambda_{n,k} \in \mathcal{D}$ onde $\lambda_{n,k}$ são os autovalores da matriz jacobiana $J_n := \frac{\partial f}{\partial u}|_{(t_n,u_n)}$, baseado na hipótese que o comportamento local da EDO é modelado por

$$u' = u_n + J_n(u - u_n) \tag{2.26}$$

Esta prática não é exata e fornece apenas uma ideia local do comportamento da EDO (podendo levar a conclusões errôneas).

Um dos teoremas mais importantes em análise numérica é o seguinte:

Teorema 2.1.2. *Um método numérico **consistente** para um problema de valor inicial bem-posto é **convergente** se e somente se ele é **estável**.*

Ele também é usado da seguinte forma:

Teorema 2.1.3. *Se um método numérico é **consistente** e **estável** em $[a,b]$ então ele é **convergente**.*

Exercícios resolvidos

Em construção ... Gostaria de participar na escrita deste livro? Veja como em:

http://www.ufrgs.br/numerico/participe.html

Exercícios

Em construção ... Gostaria de participar na escrita deste livro? Veja como em:

http://www.ufrgs.br/numerico/participe.html

2.2 O método de Euler implícito

Integrando o problema de valor inicial

$$u'(t) = f(t,u(t)) \tag{2.27}$$
$$u(t^{(1)}) = a \tag{2.28}$$

de $t^{(1)}$ até $t^{(2)}$ obtemos (como feito anteriormente)

$$u(t^{(2)}) = u(t^{(1)}) + \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\, dt \tag{2.29}$$

Entretanto se aproximarmos a função f por uma função constante $f(t,u(t)) \approx f(t^{(2)},u^{(2)})$, obteremos um novo método

$$u^{(2)} = u^{(1)} + f(t^{(2)},u^{(2)}) \int_{t^{(1)}}^{t^{(2)}} dt \tag{2.30}$$
$$u^{(2)} = u^{(1)} + hf(t^{(2)},u^{(2)}) \tag{2.31}$$

Generalizando este procedimento para t_n obtemos o **método de Euler implícito**

$$u_{n+1} = u_n + h\ f(t^{(n+1)},u_{n+1}). \tag{2.32}$$

Note que este método é **implícito** (a equação é implícita) pois depende de u_{n+1} dos dois lados da equação. Se a função f for simples o suficiente, podemos resolver a equação isolando o termo u_{n+1}. Se isso não for possível, devemos usar um dos métodos vistos anteriormente para calcular as raízes da equação (por exemplo, método da bissecção e método de Newton).

Pode ser mostrado que o erro de truncamento local é

$$ETL^{n+1}_{EulImp} = \mathcal{O}(h^2).$$

portanto o método é de ordem 1. E o erro de truncamento global é

$$ETG^{n+1}_{EulImp} = \mathcal{O}(h).$$

Exemplo 2.2.1. Utilizando o **método de Euler implícito** para solucionar (2.18) obtemos

$$\begin{aligned} u_{n+1} &= u_n + h\lambda u_{n+1}, && (2.33)\\ (1-h\lambda)u_{n+1} &= u_n, && (2.34)\\ u_{n+1} &= \left(\frac{1}{1-h\lambda}\right) u_n, && (2.35)\\ u_{n+1} &= \left(\frac{1}{1-h\lambda}\right)^2 u_{n-1}, && (2.36)\\ u_{n+1} &= \left(\frac{1}{1-h\lambda}\right)^{n+1}, \quad n = 0,1,\ldots && (2.37) \end{aligned}$$

onde $u^{(1)} = 1$. Concluímos então que

$$\mathcal{D}_{EulImp} = \{z \in \mathbb{C} : \left|\frac{1}{1-z}\right| < 1\} \tag{2.38}$$

ou ainda,

$$\mathcal{D}_{EulImp} = \{z \in \mathbb{C} : |1-z| > 1\} \tag{2.39}$$

Para que o método de Euler implícito seja estável, é necessário que h seja escolhido tal que $\left|\frac{1}{1-h\lambda}\right| < 1$, ou ainda, $|1-h\lambda| > 1$. Ou seja, $h\lambda$ deve estar em $\mathcal{D}_{EulImp}$ onde

$$\mathcal{D}_{EulImp} = \{z \in \mathbb{C} : |1-z| > 1\} \tag{2.40}$$

é o exterior de um disco no plano complexo de raio 1 e centro em $z = 1$.

Note que $\mathcal{D}_{EulImp}$ inclui todo o semiplano negativo. Portanto o método de Euler implícito imita a estabilidade assintótica da EDO linear sem restrição no passo h.

Definição 2.2.1. *Um método numérico é chamado* ***A-estável*** *ou* ***incondicionalmente estável*** *se seu domínio de estabilidade linear incluir todo o semiplano complexo com parte real negativa,*

$$\{z \in \mathbb{C} : \Re z < 0\} \subseteq \mathcal{D}$$

Portanto o método de Euler implícito é A-estável (incondicionalmente estável).

2.3 Método trapezoidal

O método de Euler aproxima f como uma constante no intervalo $[t^{(1)},t^{(2)}]$. Podemos melhorar isso usando a regra do trapézio,

$$u(t^{(2)}) = u(t^{(1)}) + \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\,dt \tag{2.41}$$

$$u^{(2)} = u^{(1)} + (t^{(2)} - t^{(1)})\left(\frac{1}{2}f(t^{(1)},u^{(1)}) + \frac{1}{2}f(t^{(2)},u^{(2)})\right) \tag{2.42}$$

motivando o **método trapezoidal**

$$u_{n+1} = u_n + \frac{h}{2}\left(f(t_n,u_n) + f(t^{(n+1)},u_{n+1})\right) \tag{2.43}$$

O método trapezoidal é dito **implícito**, pois para obter u_{n+1} é necessário calcular $f(t^{(n+1)},u_{n+1})$.

Entretanto, pode ser mostrado que o erro de truncamento local é

$$ETL_{Trap}^{n+1} = O(h^3)$$

portanto o método é de ordem 2. E o erro de truncamento global é

$$ETG_{Trap}^{n+1} = O(h^2)$$

Exemplo 2.3.1. Utilizando o **método trapezoidal** para solucionar (2.18) obtemos

$$u_{n+1} = \left(\frac{1+h\lambda/2}{1-h\lambda/2}\right)^{n+1}, \quad n = 0,1,\ldots \tag{2.44}$$

Concluímos então que

$$\mathcal{D}_{Tr} = \{z \in \mathbb{C} : \left|\frac{1+z/2}{1-z/2}\right| < 1\} \tag{2.45}$$

Note que $\mathcal{D}_{Tr} = \mathbb{C}^-$, o semiplano negativo. Portanto o método do trapézio imita a estabilidade assintótica da EDO linear sem restrição no passo h.

Exercícios resolvidos

Em construção ... Gostaria de participar na escrita deste livro? Veja como em:

http://www.ufrgs.br/numerico/participe.html

Exercícios

Em construção ... Gostaria de participar na escrita deste livro? Veja como em:

http://www.ufrgs.br/numerico/participe.html

2.4 O método de Heun

Também chamado de método de **Euler modificado**. A ideia é calcular primeiramente um valor intermediário $\tilde{u}$ usando o método de Euler explícito e usar esse valor na equação para o método do Trapézio. Ou seja, o **método de Heun** é

$$\tilde{u} = u_n + hf(t_n,u_n) \tag{2.46}$$

$$u_{n+1} = u_n + \frac{h}{2}\left(f(t_n,u_n) + f(t^{(n+1)},\tilde{u})\right) \tag{2.47}$$

Este é um exemplo de um método preditor-corretor.
Felizmente o erro de truncamento local continua sendo

$$ETL_{Heun}^{n+1} = O(h^3)$$

e o erro de truncamento global é

$$ETG_{Heun}^{n+1} = O(h^2)$$

Exercícios resolvidos

Em construção ... Gostaria de participar na escrita deste livro? Veja como em:

http://www.ufrgs.br/numerico/participe.html

Exercícios

E 2.4.1. Use o método de Euler melhorado para obter uma aproximação numérica do valor de $u(1)$ quando $u(t)$ satisfaz o seguinte problema de valor inicial

$$\begin{aligned} u'(t) &= -u(t) + e^{u(t)}, \\ u(0) &= 0, \end{aligned}$$

usando passos $h = 0{,}1$ e $h = 0{,}01$.
E 2.4.1. $u(1) \approx 1{,}317078$ quando $h = 0{,}1$ e $u(1) \approx 1{,}317045$.

E 2.4.2. Use o método de Euler e o método de Euler melhorado para obter aproximações numéricas para a solução do seguinte problema de valor inicial para $t \in [0,1]$:

$$\begin{aligned} u'(t) &= -u(t) - u(t)^2, \\ u(0) &= 1, \end{aligned}$$

usando passo $h = 0{,}1$. Compare os valores da solução exata dada por $u(t) = \frac{1}{2e^t - 1}$ com os numéricos nos pontos $t = 0$, $t = 0{,}1$, $t = 0{,}2$, $t = 0{,}3$, $t = 0{,}4$, $t = 0{,}5$, $t = 0{,}6$, $t = 0{,}7$, $t = 0{,}8$, $t = 0{,}9$, $t = 1{,}0$.

E 2.4.2.

t	Exato	Euler	Euler melhorado	Erro Euler	Erro Euler melhorado
0,0	1,	1,	1,	0,	0,
0,1	0,826213	0,8	0,828	0,026213	0,001787
0,2	0,693094	0,656	0,695597	0,037094	0,002502
0,3	0,588333	0,547366	0,591057	0,040967	0,002724
0,4	0,504121	0,462669	0,506835	0,041453	0,002714
0,5	0,435267	0,394996	0,437861	0,040271	0,002594
0,6	0,378181	0,339894	0,380609	0,038287	0,002428
0,7	0,330305	0,294352	0,332551	0,035953	0,002246
0,8	0,289764	0,256252	0,291828	0,033512	0,002064
0,9	0,255154	0,224061	0,257043	0,031093	0,001889
1,0	0,225400	0,196634	0,227126	0,028766	0,001726

No Scilab, esta tabela pode ser produzida com o código:

```
deff('du=f(u)','du=-u-u^2')
sol_Euler=Euler(f,0,1,10,1)'
sol_Euler_mod=Euler_mod(f,0,1,10,1)'
deff('u=u_exata(t)','u=1/(2*exp(t)-1)')
t=[0:.1:1]'
sol_exata=feval(t,u_exata)
tabela=[t sol_exata sol_Euler sol_Euler_mod abs(sol_exata-sol_Euler) abs(sol_exata-sol_Euler_mod)]
```

2.5 O método theta

Tanto o método de Euler quanto o método trapezoidal se encaixam no método

$$u_{n+1} = u_n + h(\theta f(t_n, u_n) + (1-\theta) f(t^{(n+1)}, u_{n+1})) \tag{2.48}$$

com $\theta = 1$ e $\theta = \frac{1}{2}$ respectivamente. O método é explícito somente para $\theta = 1$. Para $\theta = 0$, obtemos o método implícito de Euler.

Exercícios resolvidos

Em construção ... Gostaria de participar na escrita deste livro? Veja como em:

http://www.ufrgs.br/numerico/participe.html

Exercícios

Em construção ... Gostaria de participar na escrita deste livro? Veja como em:

http://www.ufrgs.br/numerico/participe.html

2.6 O método de Taylor

Uma maneira simples de aumentar a ordem dos métodos de Euler anteriormente descritos consiste em truncar a série de Taylor de $u(t+h)$:

$$u(t+h) = u(t) + hu'(t) + \frac{h^2}{2!}u''(t) + \frac{h^3}{3!}u'''(t) + \dots \tag{2.49}$$

Utilizando dois termos temos o método de Euler. Utilizando os três primeiros termos da série e substituindo $u'(t) = f(t,x)$ e $u''(t) = \frac{\partial f}{\partial t}(t,x)$ temos o **método de Taylor de ordem** 2

$$u_{n+1} = u_n + hf(t_n,u_n) + \frac{h^2}{2!}\frac{\partial f}{\partial t}(t_n,u_n) \tag{2.50}$$

O método de Taylor de ordem 3 é

$$u_{n+1} = u_n + hf(t_n,u_n) + \frac{h^2}{2!}\frac{\partial f}{\partial t}(t_n,u_n) + \frac{h^3}{3!}\frac{\partial^2 f}{\partial t^2}(t_n,u_n)$$

Exercícios resolvidos

Em construção ... Gostaria de participar na escrita deste livro? Veja como em:

http://www.ufrgs.br/numerico/participe.html

Exercícios

Em construção ... Gostaria de participar na escrita deste livro? Veja como em:

http://www.ufrgs.br/numerico/participe.html

2.7 Estabilidade dos métodos de Taylor

Exemplo 2.7.1. Prove que para um método de Taylor de ordem p para a EDO (2.18) temos

$$p(z) = 1 + z + \frac{z^2}{2!} + \frac{z^3}{3!} + \dots + \frac{z^p}{p!} \tag{2.51}$$

onde $u_n = (p(z))^n u_0$ e a região de estabilidade é dada por

$$\mathcal{D}_T = \{z \in \mathbb{C} : |p(z)| < 1\} \tag{2.52}$$

Trace as regiões de estabilidade para o método de Taylor para $p = 1, \dots, 6$ no mesmo gráfico.

Exemplo 2.7.2. Aproxime a solução do problema de valor inicial

$$\frac{du}{dt} = \text{sen } t \tag{2.53}$$
$$u(0) = 1 \tag{2.54}$$

para $t \in [0,10]$.

a. Trace a solução para $h = 0{,}16$, $0{,}08$, $0{,}04$, $0{,}02$ e $0{,}01$ para o método de Taylor de ordem 1, 2 e 3. (Trace todos de ordem 1 no mesmo gráfico, ordem 2 em outro gráfico e ordem 3 outro gráfico separado.)

b. Utilizando a solução exata, trace um gráfico do erro em escala logarítmica. Comente os resultados (novamente, em cada gráfico separado para cada método repita os valores acima)

c. Fixe agora o valor $h = 0{,}02$ e trace no mesmo gráfico uma curva para cada método.

d. Trace em um gráfico o erro em $t = 10$ para cada um dos métodos (uma curva para cada ordem) a medida que h diminui. (Use escala `loglog`)

Exercícios resolvidos

Em construção ... Gostaria de participar na escrita deste livro? Veja como em:

http://www.ufrgs.br/numerico/participe.html

Exercícios

Em construção ... Gostaria de participar na escrita deste livro? Veja como em:

http://www.ufrgs.br/numerico/participe.html

2.8 Métodos de passo múltiplo

Seja o problema de valor inicial

$$u'(t) = f(t,u(t)) \tag{2.55}$$
$$u(t_0) = a \tag{2.56}$$

Integrando a EDO em $[t^{(n+1)},t^{(n)}]$ obtemos

$$u_{n+1} = u_n + \int_{t_n}^{t^{(n+1)}} f(t,u(t))\, dt \tag{2.57}$$

Denote por $f_n \equiv f(t_n,u_n)$. Um método de passo simples utiliza f_{n+1} e f_n. Um método de passo múltiplo utiliza também s valores anteriores já calculados como $f_{n-1},f_{n-2},\ldots,f_{n-s}$, onde $s \geq 1$ inteiro.

$$u_{n+1} = u_n + h[b_s f_{n+1} + b_{s-1} f_n + \ldots + b_1 f_{n-s+2} + b_0 f_{n-s+1}] \tag{2.58}$$

Para conformidade com [9], translade $s-1$ índices,

$$u_{n+s} = u_{n+s-1} + h[b_s f_{n+s} + b_{s-1} f_{n+s-1} + \ldots + b_1 f_{n+1} + b_0 f_n] \tag{2.59}$$

e teremos

$$u_{n+s} = u_{n+s-1} + h \sum_{m=0}^{s} b_m f_{n+m} \tag{2.60}$$

De forma geral um **método de passo múltiplo** será

$$\sum_{m=0}^{s} a_m u_{n+m} = h \sum_{m=0}^{s} b_m f_{n+m} \tag{2.61}$$

Exercícios resolvidos

Em construção ... Gostaria de participar na escrita deste livro? Veja como em:

http://www.ufrgs.br/numerico/participe.html

Exercícios

Em construção ... Gostaria de participar na escrita deste livro? Veja como em:

http://www.ufrgs.br/numerico/participe.html

2.9 O método de Adams-Bashforth

Quando $a_s = 1$, $a_{s-1} = -1$, $a_m = 0$ para $m = s-2,\ldots,0$, $b_s = 0$ temos um método de Adams-Bashforth do tipo

$$u_{n+s} = u_{n+s-1} + h \sum_{m=0}^{s-1} b_m f_{n+m} \tag{2.62}$$

Note que os métodos de Adams-Bashforth são **explícitos** pois $b_s = 0$.

Exemplo 2.9.1. Vamos obter o método de Adams-Bashforth para $s = 4$ como

$$u_{n+4} = u_{n+3} + \int_{t_{n+3}}^{t_{n+4}} f(t,u(t))\,dt \tag{2.63}$$
$$u_{n+4} = u_{n+3} + h\sum_{m=0}^{3} b_m f_{n+m} \tag{2.64}$$
$$u_{n+4} = u_{n+3} + h[b_3 f_{n+3} + b_2 f_{n+2} + b_1 f_{n+1} + b_0 f_n] \tag{2.65}$$

Para isso devemos obter $[b_3,b_2,b_1,b_0]$ tal que o método seja exato para polinômios até ordem 3. Podemos obter esses coeficientes de maneira análoga a obter os coeficientes de um método para integração.

Supondo que os nós t_k estejam igualmente espaçados, e para facilidade dos cálculos, como o intervalo de integração é $[t_{n+3},t_{n+4}]$, translade t_{n+3} para a origem tal que $[t_n,t^{(n+1)},\ldots,t_{n+4}] = [-3h,-2h,-h,0,h]$.

Considere a base $[\phi_0(t),\ldots,\phi_3(t)] = [1,t,t^2,t^3]$ e substitua $f(t)$ por $\phi_k(t)$ obtendo

$$\int_0^h 1\,dt = h = h(b_0(1) + b_1(1) + b_2(1) + b_3(1))$$
$$\int_0^h t\,dt = \frac{h^2}{2} = h(b_0(0) + b_1(-h) + b_2(-2h) + b_3(-3h))$$
$$\int_0^h t^2\,dt = \frac{h^3}{3} = h(b_0(0)^2 + b_1(-h)^2 + b_2(-2h)^2 + b_3(-3h)^2)$$
$$\int_0^h t^3\,dt = \frac{h^4}{4} = h(b_0(0)^3 + b_1(-h)^3 + b_2(-2h)^3 + b_3(-3h)^3)$$

que pode ser escrito na forma matricial

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & -1 & -2 & -3 \\ 0 & 1 & 4 & 9 \\ 0 & -1 & -8 & -27 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1/2 \\ 1/3 \\ 1/4 \end{pmatrix} \tag{2.66}$$

Resolvendo o sistema obtemos

$$[b_0,b_1,b_2,b_3] = [-\frac{9}{24},\frac{37}{24},-\frac{59}{24},\frac{55}{24}]$$

fornecendo o **método de Adams-Bashforth de** 4 **estágios**

$$u_{n+4} = u_{n+3} + \tfrac{h}{24}[55f_{n+3} - 59f_{n+2} + 37f_{n+1} - 9f_n] \tag{2.67}$$

Exercícios resolvidos

Em construção ... Gostaria de participar na escrita deste livro? Veja como em:

http://www.ufrgs.br/numerico/participe.html

Exercícios

E 2.9.1. Mostre que o método de Adams-Bashforth para $s = 2$ é dado por

$$u_{n+2} = u_{n+1} + \tfrac{h}{2}[3f_{n+1} - f_n] \tag{2.68}$$

E 2.9.2. Mostre que o método de Adams-Bashforth para $s = 3$ é dado por

$$u_{n+3} = u_{n+2} + \tfrac{h}{12}[23f_{n+2} - 16f_{n+1} + 5f_n] \tag{2.69}$$

2.10 O método de Adams-Moulton

Quando $a_s = 1$, $a_{s-1} = -1$, $a_m = 0$ para $m = s-2,\ldots,0$, $b_s \neq 0$ temos um método de Adams-Moulton do tipo

$$u_{n+s} = u_{n+s-1} + h\sum_{m=0}^{s} b_m f_{n+m} \tag{2.70}$$

Note que os métodos de Adams-Moulton são implícitos pois $b_s \neq 0$.

Exemplo 2.10.1. Vamos obter o método de Adams-Moulton para $s = 3$ como

$$u_{n+3} = u_{n+2} + \int_{t_{n+3}}^{t_{n+4}} f(t,u(t))\,dt \tag{2.71}$$

$$u_{n+3} = u_{n+2} + h\sum_{m=0}^{3} b_m f_{n+m} \tag{2.72}$$

$$u_{n+3} = u_{n+2} + h[b_3 f_{n+3} + b_2 f_{n+2} + b_1 f_{n+1} + b_0 f_n] \tag{2.73}$$

Para isso devemos obter $[b_3,b_2,b_1,b_0]$ tal que o método seja exato para polinômios até ordem 3. Podemos obter esses coeficientes de maneira análoga a obter os coeficientes de um método para integração.

Supondo que os nós t_k estejam igualmente espaçados, e para facilidade dos cálculos, como o intervalo de integração é $[t_{n+2},t_{n+3}]$, translade t_{n+2} para a origem tal que $[t_n,t^{(n+1)},\ldots,t_{n+3}] = [-2h,-h,0,h]$.

Considere a base $[\phi_0(t),\ldots,\phi_3(t)] = [1,t,t^2,t^3]$ e substitua $f(t)$ por $\phi_k(t)$ obtendo

$$\int_0^h 1\,dt = h = h(b_0(1) + b_1(1) + b_2(1) + b_3(1))$$

$$\int_0^h t\,dt = \frac{h^2}{2} = h(b_0(h) + b_1(0) + b_2(-h) + b_3(-2h))$$

$$\int_0^h t^2\,dt = \frac{h^3}{3} = h(b_0(h)^2 + b_1(0)^2 + b_2(-h)^2 + b_3(-2h)^2)$$

$$\int_0^h t^3\,dt = \frac{h^4}{4} = h(b_0(h)^3 + b_1(0)^3 + b_2(-h)^3 + b_3(-2h)^3)$$

que pode ser escrito na forma matricial

$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & -1 & -2 \\ 1 & 0 & 1 & 4 \\ 1 & 0 & -1 & -8 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1/2 \\ 1/3 \\ 1/4 \end{pmatrix} \tag{2.74}$$

Resolvendo o sistema obtemos

$$[b_0,b_1,b_2,b_3] = [\frac{1}{24}, -\frac{5}{24}, \frac{19}{24}, \frac{9}{24},]$$

fornecendo a regra

$$u_{n+3} \quad = u_{n+2} + \tfrac{h}{24}[9f_{n+3} + 19f_{n+2} - 5f_{n+1} + f_n] \tag{2.75}$$

Exercícios resolvidos

Em construção ... Gostaria de participar na escrita deste livro? Veja como em:

http://www.ufrgs.br/numerico/participe.html

Exercícios

E 2.10.1. Encontre o método de Adams-Moulton para $s = 2$.

E 2.10.2. Encontre o método de Adams-Moulton para $s = 3$.

2.11 Método BDF

Um método de ordem s com s estágios é chamado de **método BDF-Backward Differentiation Formula** se $\sigma(w) = b_s w^s$, onde $b_s \in \mathbb{R}$, ou seja,

$$a_s u_{n+s} + ... + a_1 u_{n+1} + a_0 u_n \quad = h b_s f_{n+s} \tag{2.76}$$

Exemplo 2.11.1. Mostre que o método BDF com $s = 3$ é

$$u_{n+3} - \frac{18}{11}u_{n+2} + \frac{9}{11}u_{n+1} - \frac{2}{11}u_n \quad = \tfrac{6}{11}hf_{n+3} \tag{2.77}$$

Exercícios

E 2.11.1. Mostre que o método BDF com $s = 1$ é o método de Euler implícito.

E 2.11.2. Mostre que o método BDF com $s = 2$ é

$$u_{n+2} - \frac{4}{3}u_{n+1} + \frac{1}{3}u_n \quad = \tfrac{2}{3}hf_{n+2} \tag{2.78}$$

2.12 Ordem e convergência de métodos de passo múltiplo

Mais geralmente, um método de passo múltiplo será da forma

$$a_s u_{n+s} + ... + a_1 u_{n+1} + a_0 u_n \quad = h[b_s f_{n+s} + ... + b_1 f_{n+1} + b_0 f_n] \tag{2.79}$$

Por convenção normalizamos a equação acima tomando $a_s = 1$. Quando $b_s = 0$ temos um método explícito e quando $b_s \neq 0$ temos um método implícito.

O método será de ordem p se o $ETL = \mathcal{O}(h^{p+1})$.

Dois polinômios são usados para estudar o método (2.79):

$$\rho(w) = a_s w^s + ... + a_1 w + a_0, \qquad \sigma(w) = b_s w^s + ... + b_1 w + b_0, \tag{2.80}$$

Exemplo 2.12.1. O método (2.69) de Adams-Bashforth para $s = 3$ estágios é de ordem 3 de convergência, ou seja, $ETL = \mathcal{O}(h^4)$. Ele é construído de tal maneira que seja exato para os polinômios $1, t, t^2, t^3$.

2.12.1 Consistência, estabilidade e convergência

Teorema 2.12.1. *Um método de passo múltiplo é* ***consistente*** *se* $\rho(1) = 0$ *e* $\rho'(1) = \sigma(1)$.

Teorema 2.12.2. *Um método de passo múltiplo é* ***estável*** *se todas as raízes de* $\rho(z)$ *estão em* $|z| \leq 1$ *e as raízes com* $|z| = 1$ *são simples.*

Teorema 2.12.3. *Se um método numérico é* ***consistente*** *e* ***estável*** *em* $[a,b]$ *então ele é* ***convergente****.*

Exemplo 2.12.2. Prove que o método de passo 3

$$u_{n+3} + \frac{27}{11}u_{n+2} - \frac{27}{11}u_{n+1} - u_n = \tag{2.81}$$

$$= \frac{h}{11}[3f_{n+3} + 27f_{n+2} + 27f_{n+1} + 3f_n] \tag{2.82}$$

não é estável.

Solução. O polinômio

$$\rho(w) = w^3 + \frac{27}{11}w^2 - \frac{27}{11}w - 1 \tag{2.83}$$

$$= (w-1)\left(w + \frac{19+4\sqrt{15}}{11}\right)\left(w + \frac{19-4\sqrt{15}}{11}\right) \tag{2.84}$$

falha na condição da raiz. ◊

Exercícios resolvidos

Em construção ... Gostaria de participar na escrita deste livro? Veja como em:

http://www.ufrgs.br/numerico/participe.html

Exercícios

E 2.12.1. Prove que todos os métodos de Adams-Bashforth satisfazem a condição da raiz.

Teorema 2.12.4. *O polinômio $\rho(w)$ em (2.76) satisfaz a condição da raiz e o método BDF é convergente se e somente se $1 \leq s \leq 6$.*

E 2.12.2. Mostre que os métodos BDF com $s = 2$ e $s = 3$ são convergentes.

2.12.2 As barreiras de Dahlquist

Um método de passo múltiplo possui $2s + 1$ coeficientes a_m, b_m. Poderíamos definir tais coeficientes de tal forma a obter ordem máxima.

Conclusão? Poderíamos obter métodos com s estágios e ordem $2s$.

Entretanto tal método (implícito de passo s e ordem $2s$) não é convergente para $s \geq 3$.

É possível provar que a ordem máxima de convergência para um método de passo múltiplo s é no máximo $2\lfloor (s+2)/2 \rfloor$ para métodos implícitos e s para métodos explícitos. Esta é a **primeira barreira de Dahlquist**.

2.13 Estabilidade dos métodos de passo múltiplo

Teorema 2.13.1. *O método BDF de 2 estágios é A-estável.*

Teorema 2.13.2 (A segunda barreira de Dahlquist)**.** *A ordem máxima de um método de passo múltiplo A-estável é dois.*

2.14 Métodos de Runge-Kutta

2.14.1 Método de Runge-Kutta implícito (IRK)

No conjunto de equações (1.64)-(**??**), U_k depende em valores conhecidos $F_1,\ldots,F_{k-1}$ tornando o método explícito.

Entretanto se U_k depender de $F_1,\ldots,F_\nu$ temos um método implícito como

$$U_j = u_n + h\sum_{i=1}^{\nu} a_{ji}F_i, \quad j=1,\ldots,\nu \tag{2.85}$$

$$u_{n+1} = u_n + h\sum_{i=1}^{\nu} b_i F_i \tag{2.86}$$

onde $A=(a_{ij})$ é a matriz de RK. É necessário que

$$\sum_{i=1}^{\nu} a_{ji} = c_j, \qquad j=1,\ldots,\nu \tag{2.87}$$

para que o método possua ordem $p \geq 1$.

Exemplo 2.14.1. Um método de Runge-Kutta Implícito (IRK) de dois estágios é dado por

$$U_1 = u_n + h/4[f(t_n,U_1) - f(t_n+\frac{2}{3}h,U_2)] \tag{2.88}$$

$$U_2 = u_n + h/12[3f(t_n,U_1) + 5f(t_n+\frac{2}{3}h,U_2)] \tag{2.89}$$

$$u_{n+1} = u_n + h/4[f(t_n,U_1) + 3f(t_n+\frac{2}{3}h,U_2)] \tag{2.90}$$

que possui uma tabela como

$$\begin{array}{c|cc} 0 & \frac{1}{4} & -\frac{1}{4} \\ \frac{2}{3} & \frac{1}{4} & \frac{5}{12} \\ \hline & \frac{1}{4} & \frac{3}{4} \end{array}$$

2.15 Estimativa da ordem de convergência

Raramente temos a solução exata $u(t)$ para calcular o erro obtido na solução numérica. Entretanto, se a solução é suave o suficiente e o espaçamento h é pequeno suficientemente, podemos usar o seguinte procedimento para estimar a ordem do método (ou ainda, o erro na solução).

Como visto nos exemplo numéricos anteriores, em gráficos na escala `loglog`, se h é grande não obtemos a ordem de convergência utilizada (por exemplo, encontramos que o método de Euler possui ordem $p \approx 0{,}7$ onde deveria ser 1). A medida que h decresce se aproximando de 0, a ordem de convergência tende a se aproximar de $p \approx 1$. (Entretanto h não pode ficar muito pequeno a ponto que as operações de ponto flutuante atrapalhem na convergência).

Portanto existe uma faixa $h_{min} < h < h_{max}$ onde o método apresenta a ordem desejada. Essa região depende do método e do problema de valor inicial estudado.

Mas se estivermos nessa região podemos aproximar a ordem do método da seguinte forma: Considere a solução para um determinado $t = T^*$ fixo, $u(T^*)$. Considere também as aproximações das soluções obtidas com espaçamento h, denotada por u^h; a aproximação obtida com espaçamento dividido por 2, $h/2$, denotada por $u^{h/2}$; a aproximação obtida com espaçamento $h/4$, denotada por $u^{h/4}$, ... e assim por diante, todas calculadas em $t = T^*$.

2.15.1 Método 1

Podemos utilizar uma solução bem refinada, por exemplo, $u^{h/16}$ como sendo uma boa aproximação da solução exata e supormos que $u^* = u^{h/16}$. Desta forma podemos aproximar o erro por $e^h = \|u^{(h)} - u^*\|$ e a ordem do método é estimada como

$$p \approx \frac{\log(e^h) - \log(e^{h/2})}{\log(h) - \log(h/2)} \tag{2.91}$$

$$\approx \frac{\log\left(\frac{e^h}{e^{h/2}}\right)}{\log(h/(h/2))} \tag{2.92}$$

$$\approx \frac{\log\left(\frac{e^h}{e^{h/2}}\right)}{\log(2)} \tag{2.93}$$

$$\approx \frac{\log\left(\frac{\|u^h - u^*\|}{\|u^{h/2} - u^*\|}\right)}{\log(2)} \tag{2.94}$$

$$\tag{2.95}$$

2.15.2 Método 2

Segundo Ferziger/Peric/Roache, podemos também estimar p diretamente de

$$p \approx \frac{\log\left(\frac{\|u^{h/2} - u^h\|}{\|u^{h/4} - u^{h/2}\|}\right)}{\log(2)} \tag{2.96}$$

$$\tag{2.97}$$

Exercícios

E 2.15.1. Resolva o problema 1 pelos diversos métodos e verifique heuristicamente a estabilidade para diversos valores de h.

2.16 Exercícios finais

E 2.16.1. Considere o seguinte modelo para o crescimento de uma colônia de bactérias:

$$\frac{du}{dt} = \alpha u(A - u)$$

onde u indica a densidade de bactérias em unidades arbitrárias na colônia e α e A são constantes positivas. Pergunta-se:

a) Qual a solução quando a condição inicial $u(0)$ é igual a 0 ou A?

b) O que acontece quando a condição inicial $u(0)$ é um número entre 0 e A?

c) O que acontece quando a condição inicial $u(0)$ é um número negativo?

d) O que acontece quando a condição inicial $u(0)$ é um número positivo maior que A?

e) Se $A = 10$ e $\alpha = 1$ e $u(0) = 1$, use métodos numéricos para obter tempo necessário para que a população dobre?

f) Se $A = 10$ e $\alpha = 1$ e $u(0) = 4$, use métodos numéricos para obter tempo necessário para que a população dobre?

E 2.16.1.
Os valores exatos para os itens e e f são: $\frac{1}{10}\ln\left(\frac{9}{4}\right)$ e $\frac{1}{10}\ln(6)$

E 2.16.2. Considere o seguinte modelo para a evolução da velocidade de um objeto em queda (unidades no SI):

$$v' = g - \alpha v^2$$

Sabendo que $g = 9{,}8$ e $\alpha = 10^{-2}$ e $v(0) = 0$. Pede-se a velocidade ao tocar o solo, sabendo que a altura inicial era 100.

E 2.16.2.
O valor exato é $\sqrt{\frac{g}{\alpha}\left[1 - e^{-200\alpha}\right]}$ em $t = \frac{1}{\sqrt{g\alpha}}\tanh^{-1}\left(\sqrt{1 - e^{-200\alpha}}\right)$

E 2.16.3. Considere o seguinte modelo para o oscilador não linear de Van der Pol:

$$u''(t) - \alpha(A - u(t)^2)u'(t) + w_0^2 u(t) = 0$$

onde A, α e w_0 são constantes positivas.

- Encontre a frequência e a amplitude de oscilações quando $w_0 = 1$, $\alpha = .1$ e $A = 10$. (Teste diversas condições iniciais)
- Estude a dependência da frequência e da amplitude com os parâmetros A, α e w_0. (Teste diversas condições iniciais)
- Que diferenças existem entre esse oscilador não linear e o oscilador linear?

E 2.16.4. Considere o seguinte modelo para um oscilador não linear:

$$\begin{aligned} u''(t) - \alpha(A - z(t))u'(t) + w_0^2 u(t) &= 0 \\ Cz'(t) + z(t) &= u(t)^2 \end{aligned}$$

onde A, α, w_0 e C são constantes positivas.

- Encontre a frequência e a amplitude de oscilações quando $w_0 = 1$, $\alpha = .1$, $A = 10$ e $C = 10$. (Teste diversas condições iniciais)
- Estude a dependência da frequência e da amplitude com os parâmetros A, α, w_0 e C. (Teste diversas condições iniciais)

E 2.16.5. Considere o seguinte modelo para o controle de temperatura em um processo químico:

$$\begin{aligned} CT'(t) + T(t) &= \kappa P(t) + T_{ext} \\ P'(t) &= \alpha(T_{set} - T(t)) \end{aligned}$$

onde C, α e κ são constantes positivas e $P(t)$ indica o potência do aquecedor. Sabendo que T_{set} é a temperatura desejada, interprete o funcionamento esse sistema de controle.

- Calcule a solução quando a temperatura externa $T_{ext} = 0$, $T_{set} = 1000$, $C = 10$, $\kappa = .1$ e $\alpha = .1$. Considere condições iniciais nulas.
- Quanto tempo demora o sistema para atingir a temperatura 900K?
- Refaça os dois primeiros itens com $\alpha = 0{,}2$ e $\alpha = 1$
- Faça testes para verificar a influência de T_{ext}, α e κ na temperatura final.

E 2.16.6. Considere a equação do pêndulo dada por:

$$\frac{d^2\theta(t)}{dt^2} + \frac{g}{l}\operatorname{sen}(\theta(t)) = 0$$

onde g é o módulo da aceleração da gravidade e l é o comprimento da haste.

- Mostre analiticamente que a energia total do sistema dada por

$$\frac{1}{2}\left(\frac{d\theta(t)}{dt}\right)^2 - \frac{g}{l}\cos(\theta(t))$$

é mantida constante.

- Resolva numericamente esta equação para $g = 9{,}8m/s^2$ e $l = 1m$ e as seguintes condições iniciais:

$\theta(0) = 0{,}5$ e $\theta'(0) = 0$.

$\theta(0) = 1{,}0$ e $\theta'(0) = 0$.

$\theta(0) = 1{,}5$ e $\theta'(0) = 0$.

$\theta(0) = 2{,}0$ e $\theta'(0) = 0$.

$\theta(0) = 2{,}5$ e $\theta'(0) = 0$.

$\theta(0) = 3{,}0$ e $\theta'(0) = 0$.

Em todos os casos, verifique se o método numérico reproduz a lei de conservação de energia e calcule período e amplitude.

E 2.16.7. Considere o modelo simplificado de FitzHugh-Nagumo para o potencial elétrico sobre a membrana de um neurônio:

$$\begin{aligned}\frac{dV}{dt} &= V - V^3/3 - W + I \\ \frac{dW}{dt} &= 0{,}08(V + 0{,}7 - 0{,}8W)\end{aligned}$$

onde I é a corrente de excitação.

- Encontre o único estado estacionário (V_0,W_0) com $I = 0$.

- Resolva numericamente o sistema com condições iniciais dadas por (V_0,W_0) e

$I = 0$

$I = 0{,}2$

$I = 0{,}4$

$I = 0{,}8$

$I = e^{-t/200}$

E 2.16.8. Considere o problema de valor inicial dado por

$$\begin{aligned} \frac{du(t)}{dt} &= -u(t) + e^{-t} \\ u(0) &= 0 \end{aligned}$$

Resolva analiticamente este problema usando as técnicas elementares de equações diferenciais ordinárias. A seguir encontre aproximações numéricas usando os métodos de Euler, Euler modificado, Runge-Kutta clássico e Adams-Bashforth de ordem 4 conforme pedido nos itens.

a) Construa uma tabela apresentando valores com 7 algarismos significativos para comparar a solução analítica com as aproximações numéricas produzidas pelos métodos sugeridos. Construa também uma tabela para o erro absoluto obtido por cada método numérico em relação à solução analítica. Nesta última tabela, expresse o erro com 2 algarismos significativos em formato científico. Dica: $format('e',8)$ para a segunda tabela.

	0,5	1,0	1,5	2,0	2,5
Analítico					
Euler					
Euler modificado					
Runge-Kutta clássico					
Adams-Bashforth ordem 4					

	0,5	1,0	1,5	2,0	2,5
Euler					
Euler modificado					
Runge-Kutta clássico					
Adams-Bashforth ordem 4					

b) Calcule o valor produzido por cada um desses método para $u(1)$ com passo $h = 0{,}1$, $h = 0{,}05$, $h = 0{,}01$, $h = 0{,}005$ e $h = 0{,}001$. Complete a tabela com os valores para o erro absoluto encontrado.

	0,1	0,05	0,01	0,005	0,001
Euler					
Euler modificado					
Runge-Kutta clássico					
Adams-Bashforth ordem 4					

E 2.16.8.

	0,5	1,0	1,5	2,0	2,5
Analítico	0,3032653	0,3678794	0,3346952	0,2706706	0,2052125
Euler	0,3315955	0,3969266	0,3563684	0,2844209	0,2128243
Euler modificado	0,3025634	0,3671929	0,3342207	0,2704083	0,2051058
Runge-Kutta clássico	0,3032649	0,3678790	0,3346949	0,2706703	0,2052124
Adams-Bashforth ordem 4	0,3032421	0,3678319	0,3346486	0,2706329	0,2051848

	0,5	1,0	1,5	2,0	2,5
Euler	2,8e-2	2,9e-2	2,2e-2	1,4e-2	7,6e-3
Euler modificado	7,0e-4	6,9e-4	4,7e-4	2,6e-4	1,1e-4
Runge-Kutta clássico	4,6e-7	4,7e-7	3,5e-7	2,2e-7	1,2e-7
Adams-Bashforth ordem 4	2,3e-5	4,8e-5	4,7e-5	3,8e-5	2,8e-5

	0,1	0,05	0,01	0,005	0,001
Euler	2,9e-2	5,6e-3	2,8e-3	5,5e-4	2,8e-4
Euler modificado	6,9e-4	2,5e-5	6,2e-6	2,5e-7	6,1e-8
Runge-Kutta clássico	4,7e-7	6,9e-10	4,3e-11	6,8e-14	4,4e-15
Adams-Bashforth ordem 4	4,8e-5	9,0e-8	5,7e-9	9,2e-12	5,8e-13

Capítulo 3

Limbo

3.1 Teoria de equações diferenciais

Uma questão fundamental é analisar se um dado problema de valor inicial é um problema **bem posto**. Ou seja,

- Existe uma solução para o problema de valor inicial?
- A solução é única?
- A solução do problema de valor inicial é pouco sensível a pequenas perturbações nas condições iniciais?

Definição 3.1.1. *A função $f(t,u)$ é Lipschitz em u se existe uma constante L, tal que $\forall t \in [a,b]$ e $u, v \in \mathbb{R}$,*

$$|f(t,u) - f(t,v)| \leq L|u(t) - v(t)|.$$

Teorema 3.1.1. *Seja $f(t,u)$ contínua em t e Lipschitz em u. Então existe uma única solução para o problema de valor inicial*

$$u'(t) = f(t,u(t)) \tag{3.1}$$
$$u(t^{(1)}) = a. \tag{3.2}$$

Definição 3.1.2. ***Estabilidade dinâmica*** *refere-se a propriedade de pequenas perturbações sobre o estado inicial de um sistema gerarem pequenas variações no estado final deste sistema (haverá decaimento nas variações, ou pelo menos não crescimento, quanto t cresce).*

Teorema 3.1.2 (Dependência na condição inicial)**.** *Se $u(t)$ e $v(t)$ são soluções do problema de valor inicial com f Lipschitz com $u(t^{(1)}) = u^{(1)}$, $v(t^{(1)}) = v_1$, então*

$$|u(t) - v(t)| \leq e^{L(t-t^{(1)})}|u^{(1)} - v_1|.$$

Exercícios resolvidos

ER 3.1.1. A função $f(t,u) = \sqrt{u},\ \ u \geq 0$ não é uma função Lipschitz em u, pois

$$\lim_{u\to 0+} \frac{|f(t,u) - f(t,0)|}{|u-0|} = \lim_{u\to 0+} \frac{\sqrt{u}}{u} = \lim_{u\to 0+} \frac{1}{\sqrt{u}} = \infty$$

Mostre que o seguinte problema de valor inicial não admite solução única:

$$\frac{du}{dt} = \sqrt{u},\ \ u > 0, \tag{3.3}$$

$$u(0) = 0. \tag{3.4}$$

Solução. A função identicamente nula, $u(t) = 0$, satisfaz a equação diferencial e a condição de contorno, logo é uma solução do problema de valor inicial. No entanto, a função[1] $u(t) = \frac{t^2}{4}$ satisfaz a condição inicial, pois $u(0) = 0$ e a equação diferencial pois $\frac{du}{dt} = \frac{t}{2} = \sqrt{\frac{t^2}{4}}$.

De fato, qualquer função do tipo

$$u(t) = \begin{cases} 0, & 0 \leq t \leq t_0 \\ \frac{(t-t_0)^2}{4}, & t > t_0 \end{cases}$$

é solução do problema de valor inicial dado. ◊

Em construção ... Gostaria de participar na escrita deste livro? Veja como em:

http://www.ufrgs.br/numerico/participe.html

Exercícios

Em construção ... Gostaria de participar na escrita deste livro? Veja como em:

http://www.ufrgs.br/numerico/participe.html

[1] Esta solução pode ser obtida por separação de variáveis.

Apêndice A

Rápida introdução ao Scilab

A.1 Sobre o Scilab

Scilab é uma linguagem de programação associada com uma rica coleção de algoritmos numéricos que cobrem muitos aspectos de problemas de computação científica. Do ponto de vista de *software*, Scilab é uma linguagem interpretada. A linguagem Scilab permite a compilação dinâmica e lincagem com outras linguagens como Fortran e C. Do ponto de vista de licença, Scilab é um software gratuito no sentido que o usuário não paga por ele. Além disso, Scilab é um software de código aberto disponível sobre a licença Cecill [1]. Scilab esta disponível para Linux, Mac Os e Windows. Ajuda *online* esta disponível em português e muitas outras línguas. Do ponto de vista científico, Scilab começou focado em soluções computacionais para problemas de álgebra linear, mas, rapidamente, o número de aplicações se estendeu para muitas áreas da computação científica.

As informações deste apêndice foram adaptadas do tutorial "Introduction to Scilab" [2], veja-o para maiores informações. Além disso, recomendamos visitar o sítio oficial do Scilab:

http://www.scilab.org/

O manual oficial do Scilab em português pode ser obtido em:

http://help.scilab.org/docs/5.5.2/pt_BR/index.html

A.1.1 Instalação e execução

O Scilab pode ser executado normalmente nos sistemas operacionais Linux, Mac Os e Windows. Muitas distribuições de Linux (Linux Mint, Ubuntu, etc.) têm o Scilab no seu sistema de pacotes (incluindo binário e documentação em várias línguas). Alternativamente, no sítio de internet oficial do Scilab pode-se

obter mais versões de binários e documentação para instalação em sistemas Linux. Para a instalação em sistemas Mac Os e Windows, visite sítio de internet oficial do Scilab.

A.1.2 Usando o Scilab

O uso do Scilab pode ser feito de três formas básicas:

- usando o **console** de modo iterativo;
- usando a função `exec` para executar um código Scilab digitado em um arquivo externo;
- usando processamento *bash*.

Exemplo A.1.1. Considere o seguinte pseudocódigo:

```
s = "Olá, mundo!". (Sem imprimir na tela o resultado.)
saída(s). (Imprime na tela.)
```

Implemente este pseudocódigo no Scilab: a) usando somente o console do Scilab; b) usando o editor do Scilab e executando o código com a função `exec`; c) usando processamento *bash*.

Solução. Seguem as soluções de cada item:

a) No console temos:

```
-->s = "Olá, mundo!";
-->disp(s)
```

b) Para abrir o editor do Scilab pode-se digitar no `prompt`:

```
-->editor()
```

ou, alternativamente:

```
-->scinotes
```

Então, digita-se no editor o código:

```
s = "Olá, mundo!"
disp(s)
```

salva-se em um arquivo de sua preferência (por exemplo, `~/foo.sce`) e executa-se o código clicando no botão "*play*" disponível na barra de botões do Scinotes.

c) Para executar o código em processamento *bash*, digita-se em um editor o código:

```
s = "Olá, mundo!"
disp(s)
```

salva-se em um arquivo de sua preferência (por exemplo, `~/foo.sce`) e executa-se em um console do sistema usando a linha de comando:

```
$ scilab -nw -f ~/foo.sce
```

Digite, então, `quit` para voltar ao prompt do sistema.

◊

A.2 Elementos da linguagem

Scilab é uma linguagem interpretada em que todas as variáveis são matrizes. Uma variável é criada quando um valor é atribuído a ela. Por exemplo:

```
-->x=1
 x  =
    1.
-->y = x * 2
 y  =
    2.
```

a variável `x` recebe o valor `double` 1 e, logo após, na segunda linha de comando, a variável `y` recebe o valor `double` 2. Observamos que o símbolo `=` significa o operador de atribuição não o de igualdade. O operador lógico de igualdade no Scilab é `==`.

Comentários e continuação de linha de comando são usados como no seguinte exemplo:

```
-->//Isto é um comentário
-->x = 1 ..
-->+ 2
 x  =
    3.
```

A.2.1 Operações matemáticas elementares

No Scilab, os operadores matemáticos elementares são os seguintes:

```
+ adição
- subtração
* multiplicação
/ divisão
^ potenciação (igual a **)
' transposto conjugado
```

A.2.2 Funções e constantes elementares

Várias funções e constantes elementares já estão pré-definidas no Scilab. Por exemplo:

```
-->cos(%pi) //cosseno de pi
 ans  =
  - 1.

-->exp(1) == %e //número de Euler
 ans  =
  T

-->log(1) //logarítmo natural de 1
 ans  =
    0.
```

Para mais informações sobre quais as funções e constantes pré-definidas no Scilab, consulte o manual, seções “Funções elementares” e o carácter especial “%".

A.2.3 Operadores lógicos

No Scilab, o valor lógico verdadeiro é escrito como `%T` e o valor lógico falso como `%F`. Temos os seguintes operadores lógicos disponíveis:

```
&  e lógico
|  ou lógico
~  negação
== igualdade
~= diferente
<  menor que
>  maior que
```

```
<= menor ou igual que
>= maior ou igual que
```

Exemplo A.2.1. Se $x = 2$, então x é maior ou igual a 1 e menor que 3?

Solução. No Scilab, temos:

```
-->x=2;

-->(x >= 1) & (x < 3)
 ans  =

  T
```

◊

A.3 Matrizes

No Scilab, matriz é o tipo básico de dados, a qual é definida por seu número de linhas, colunas e tipo de dado (real, inteiro, lógico, etc.). Uma matriz $A = [a_{i,j}]_{i,j=1}^{m,n}$ no Scilab é definida usando-se a seguinte sintaxe:

```
A = [ a11 , a12 , ... , a1n ; ...; am1 , am2 , ... , amn ]
```

Exemplo A.3.1. Defina a matriz:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

Solução. No Scilab, digitamos:

```
-->A = [1 , 2 , 3 ; 4 , 5 , 6]
 A  =

    1.    2.    3.
    4.    5.    6.
```

◊

A seguinte lista contém uma série de funções que geram matrizes particulares:

```
eye      matriz identidade
linspace vetor de elementos linearmente espaçados
ones     matriz cheia de uns
zeros    matriz nula
```

A.3.1 O operador ":"

O operador ":" cria um vetor linha de elementos. A sintaxe:

```
v = i:s:j
```

cria um vetor linha:

$$v = [i,\ i+s,\ i+2s, \ldots, i+ns]$$

onde n é o maior inteiro tal que $i + ns \leq j$.

Exemplo A.3.2. Veja as seguintes linhas de comando:

```
-->v = 10:-2:3
 v  =

    10.    8.    6.    4.

-->u = 2:6
 u  =
    2.    3.    4.    5.    6.
```

A.3.2 Obtendo dados de uma matriz

A função `size` retorna as dimensões de uma matriz, por exemplo:

```
-->A = ones(3,2)
 A  =

    1.    1.
    1.    1.
    1.    1.

-->[nl, nc] = size(A)
 nc  =

    2.
 nl  =

    3.
```

informando que a matriz `A` tem três linhas e duas colunas.

Existem vários métodos para acessar os elementos de uma matriz dada `A`:

- a matriz inteira acessa-se com a sintaxe:

```
A
```

- o elemento da i-ésima linha e j-ésima coluna acessa-se usando a sintaxe:

  ```
  A(i,j)
  ```

- o bloco formado pelas linhas i_1, i_2 e pelas colunas j_1, j_2 obtém-se usando a sintaxe:

  ```
  A(i1:i2, j1:j2)
  ```

Exemplo A.3.3. Veja as seguintes linhas de comando:

```
-->A = rand(3,4) //gera uma matriz randômica
 A  =

    0.2113249    0.3303271    0.8497452    0.0683740
    0.7560439    0.6653811    0.6857310    0.5608486
    0.0002211    0.6283918    0.8782165    0.6623569

-->A //mostra toda a matriz A
 ans  =

    0.2113249    0.3303271    0.8497452    0.0683740
    0.7560439    0.6653811    0.6857310    0.5608486
    0.0002211    0.6283918    0.8782165    0.6623569

-->A(2,3) //acessa o elemento a23
 ans  =

    0.6857310

-->A(2:3,2:4) //acessa um bloco de A
 ans  =

    0.6653811    0.6857310    0.5608486
    0.6283918    0.8782165    0.6623569
```

Definida uma matriz A no Scilab, as seguintes sintaxes são bastante úteis:

```
A(:,:)   toda a matriz
A(i:j,k) os elementos das linhas i até j (inclusive) da k-ésima coluna
```

```
A(i,j:k) os elementos da i-ésima linha das colunas j até k (inclusive)
A(i,:)   a i-ésima linha da matriz
A(:,j)   a j-ésima coluna da matriz
A(i,$)   o elemento da i-ésima linha e da última coluna
A($,j)   o elemento da última linha e da j-ésima coluna
```

Exemplo A.3.4. Veja as seguintes linhas de comando:

```
-->B = rand(4,4)
 B  =

    0.2113249    0.6653811    0.8782165    0.7263507
    0.7560439    0.6283918    0.0683740    0.1985144
    0.0002211    0.8497452    0.5608486    0.5442573
    0.3303271    0.6857310    0.6623569    0.2320748

-->aux = B(:,2); B(:,2) = B(:,3); B(:,3) = aux
 B  =

    0.2113249    0.8782165    0.6653811    0.7263507
    0.7560439    0.0683740    0.6283918    0.1985144
    0.0002211    0.5608486    0.8497452    0.5442573
    0.3303271    0.6623569    0.6857310    0.2320748
```

A.3.3 Operações matriciais e elemento-a-elemento

As operações matriciais elementares seguem a mesma sintaxe que as operações elementares de números. Agora, no Scilab, também podemos fazer operações elemento-a-elemento colocando um ponto "." antes da operação desejada.

Aqui, temos as sintaxes análogas entre operações matriciais e operações elemento-a-elemento:

```
+ adição                .+ adição elemento-a-elemento
- subtração             .- subtração elemento-a-elemento
* multiplicação         .* multiplicação elemento-a-elemento
                        ./ divisão elemento-a-elemento
^ potenciação           .^ potenciação elemento-a-elemento
' transposta conjugada .' transposta (não conjugada)
```

Exemplo A.3.5. Veja as seguintes linhas de comando:

```
-->A = ones (2 ,2)
 A  =
```

```
    1.    1.
    1.    1.

-->B = 2 * ones (2 ,2)
 B  =

    2.    2.
    2.    2.

-->A * B
 ans  =

    4.    4.
    4.    4.

-->A .* B
 ans  =

    2.    2.
    2.    2.
```

A.4 Estruturas de ramificação e repetição

O Scilab contém estruturas de repetição e ramificação padrões de linguagens estruturadas.

A.4.1 A instrução de ramificação "if"

A instrução "if" permite executar um pedaço do código somente se uma dada condição for satisfeita.

Exemplo A.4.1. Veja o seguinte código Scilab:

```
i = 2
if ( i == 1 ) then
    disp ( " Hello ! " )
elseif ( i == 2 ) then
    disp ( " Goodbye ! " )
elseif ( i == 3 ) then
    disp ( " Tchau ! " )
```

```
else
    disp ( " Au Revoir ! " )
end
```

Qual é a saída apresentada no console do Scilab? Por quê?

A.4.2 A instrução de repetição "for"

A instrução `for` permite que um pedaço de código seja executado repetidamente.

Exemplo A.4.2. Veja o seguinte código:

```
for i = 1:5
    disp(i)
end
```

O que é mostrado no console do Scilab?

Exemplo A.4.3. Veja o seguinte código:

```
for j = 1:2:8
    disp(j)
end
```

O que é mostrado no console do Scilab?

Exemplo A.4.4. Veja o seguinte código:

```
for k = 10:-3:1
    disp(k)
end
```

O que é mostrado no console do Scilab?

Exemplo A.4.5. Veja o seguinte código:

```
for i = 1:3
    for j = 1:3
        disp([i,j])
    end
end
```

O que é mostrado no console do Scilab?

A.4.3 A instrução de repetição "while"

A instrução `while` permite que um pedaço de código seja executado repetidamente até que uma dada condição seja satisfeita.

Exemplo A.4.6. Veja o seguinte código Scilab:

```
s = 0
i = 1
while ( i <= 10 )
   s = s + i
   i = i + 1
end
```

Qual é o valor de `s` ao final da execução? Por quê?

A.5 Funções

Além das muitas funções já pré-definidas no Scilab, podemos definir nossas próprias funções. Para tanto, existem duas instruções no Scilab:

- `deff`
- `function`

A instrução `deff` é apropriada para definirmos funções com poucas computações. Quando a função exige um grande quantidade de código para ser definida, a melhor opção é usar a instrução `function`. Veja os seguintes exemplos:

Exemplo A.5.1. O seguinte código:

```
-->deff('y = f(x)', 'y = x + sin(x)')
```

define, no Scilab, a função $f(x) = x + \text{sen } x$.

Observe que $f(\pi) = \pi$. Confirme isso computando:

```
-->f(%pi)
```

no Scilab.

Alternativamente, definimos a mesma função com o código:

```
function [y] = f(x)
   y = x + sin(x)
endfunction
```

Verifique!

Exemplo A.5.2. O seguinte código Scilab:

```
function [z] = h(x,y)
   if (x < y) then
      z = y - x
   else
      z = x - y
   end
endfunction
```

define a função:

$$h(x,y) = \begin{cases} y - x & , x < y \\ x - y & , x \geq y \end{cases}$$

Exemplo A.5.3. O seguinte código:

```
function [y] = J(x)
   y(1,1) = 2*x(1)
   y(1,2) = 2*x(2)

   y(2,1) = -x(2)*sin(x(1)*x(2))
   y(2,2) = -x(1)*sin(x(1)*x(2))
endfunction
```

define a matriz jacobiana $J(x_1,x_2) := \frac{\partial(f_1,f_2)}{\partial(x_1,x_2)}$ da função:

$$\boldsymbol{f}(x_1,x_2) = (x_1^2 + x_2^2,\ \cos(x_1 x_2)).$$

A.6 Gráficos

Para criar um esboço do gráfico de uma função de uma variável real $y = f(x)$, podemos usar a função `plot`. Esta função faz uma representação gráfica de pontos (x_i, y_i) fornecidos. O Scilab oferece uma série de opções para esta função de forma que o usuário pode ajustar várias questões de visualização. Consulte sobre a função `plot` no manual do Scilab.

Exemplo A.6.1. Veja as seguintes linhas de código:

```
-->deff('y = f(x)','y = x .^ 3 + 1')
-->x = linspace(-2, 2, 100);
-->plot(x, f(x)); xgrid
```

Resposta dos Exercícios

Recomendamos ao leitor o uso criterioso das respostas aqui apresentadas. Devido a ainda muito constante atualização do livro, as respostas podem conter imprecisões e erros.

Referências Bibliográficas

[1] Cecill and free sofware. http://www.cecill.info. Acessado em 30 de julho de 2015.

[2] M. Baudin. Introduction to scilab. http://forge.scilab.org/index.php/p/docintrotoscilab/. Acessado em 30 de julho de 2015.

[3] R.L. Burden and J.D. Faires. *Análise Numérica*. Cengage Learning, 8 edition, 2013.

[4] J. P. Demailly. *Analyse Numérique et Équations Differentielles*. EDP Sciences, Grenoble, nouvelle Édition edition, 2006.

[5] W Gautschi. Numerical analysis: An introduction birkhauser. *Barton, Mass, USA*, 1997.

[6] Walter Gautschi and Gabriele Inglese. Lower bounds for the condition number of vandermonde matrices. *Numerische Mathematik*, 52(3):241–250, 1987/1988.

[7] L.F. Guidi. Notas da disciplina cálculo numérico. http://www.mat.ufrgs.br/~guidi/grad/MAT01169/calculo_numerico.pdf. Acessado em julho de 2016.

[8] E. Isaacson and H.B. Keller. *Analysis of numerical methods*. Dover, Ontário, 1994.

[9] Arieh Iserles. *A first course in the numerical analysis of differential equations*. Cambridge university press, 2009.

[10] W.H. Press. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007.

[11] R. Rannacher. Einführung in die numerische mathematik (numerik 0). http://numerik.uni-hd.de/~lehre/notes/num0/numerik0.pdf. Acessado em 10.08.2014.

[12] Todos os Colaboradores. Cálculo numérico - um livro colaborativo - versão com scilab. disponível em https://www.ufrgs.br/numerico/livro/main.html, Novembro 2016.

Colaboradores

Aqui você encontra a lista de colaboradores do livro. Esta lista contém somente aqueles que explicitamente se manifestaram a favor de terem seus nomes registrados aqui. A lista completa de colaborações pode ser obtida no repositório GitHub do livro:

https://github.com/livroscolaborativos/CalculoNumerico

Além das colaborações via GitHub, o livro também recebe colaborações via discussões, sugestões e avisos deixados em nossa lista de e-mails:

livro_colaborativo@googlegroups.com

Estas colaborações não estão listadas aqui, mas podem ser vistas no site do grupo de e-mails.

Caso encontre algum equívoco ou veja seu nome listado aqui por engano, por favor, entre em contato conosco por e-mail:

livroscolaborativos@gmail.com

ou via o repositório GitHub.

Tabela A.1: Lista de colaboradores

Nome	Afiliação	E-Mail	1ª Cont
Rafael Sachetto Oliveira	Universidade Federal de São João del-Rei	sachetto@ufsj.edu.br	#158
Debora Lidia Gisch	-x-	-x-	#63

Índice Remissivo