UNIVERSIDADE DE SÃO PAULO
FACULDADE DE FILOSOFIA, CIÊNCIAS E LETRAS DE RIBEIRÃO PRETO
DEPARTAMENTO DE COMPUTAÇÃO E MATEMÁTICA

GABRIEL FONSECA AMENT

Método Runge-Kutta Contínuo para equações diferenciais com retardo e aplicações em dinâmica populacional

Ribeirão Preto–SP

2022

GABRIEL FONSECA AMENT

Método Runge-Kutta Contínuo para equações diferenciais com retardo e aplicações em dinâmica populacional

Versão Corrigida

Dissertação apresentada à Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto (FFCLRP) da Universidade de São Paulo (USP), como parte das exigências para a obtenção do título de Mestre em Ciências.

Área de Concentração: Computação Aplicada.

Orientador: Profª Drª Vanessa Rolnik Artioli
Coorientador: Prof Dr Olavo Henrique Menin

Ribeirão Preto–SP

2022

Gabriel Fonseca Ament

Método Runge-Kutta Contínuo para equações diferenciais com retardo e aplicações em dinâmica populacional. Ribeirão Preto–SP, 2022.

90p. : il.; 30 cm.

Dissertação apresentada à Faculdade de Filosofia, Ciências e Letras
de Ribeirão Preto da USP, como parte das exigências para
a obtenção do título de Mestre em Ciências,
Área: Computação Aplicada.

Orientador: Profª Drª Vanessa Rolnik Artioli
Coorientador: Prof Dr Olavo Henrique Menin

1. Métodos Numéricos. 2. Runge-Kutta Contínuo. 3. Modelos de Dinâmica Populacional.

Agradecimentos

Agradeço, primeiramente, à Deus, por sempre me guiar, apoiar e ajudar a vencer as dificuldades e obstáculos da vida, proporcionando sempre meu crescimento espiritual.

Agradeço à minha noiva, Gabriela, por todos esses anos de um eterno e infinito amor, carinho e companheirismo, e em especial nesse anos de mestrado pela compreensão e apoio em todos os momentos que precisei de alguém para conversar e desabafar.

Agradeço à minha mãe, Daniela, por desde pequeno me incentivar à estudar e encarar os desafios da vida, sempre me dando um apoio e amor maternal muito além daquilo que um filho poderia imaginar se possível.

Agradeço à minha avó, Regina, minha segunda mãe, por sempre me acolher em seu lar não só físico, mas também em seu lar de amor e cuidado, que sempre me ajudou em todas as etapas de minha vida.

Agradeço ao meu pai, Paulo, por se fazer presente em minha vida como um grande amigo e companhia para um café, sempre me ajudando com conselhos para a vida.

Agradeço à minha orientadora, Vanessa, por desde antes do meu ingresso no mestrado já ter me acolhido e incentivado iniciar essa nova etapa, sempre me ajudando muito e, principalmente, sendo sempre compreensiva com minhas dificuldades na conciliação da vida acadêmica com a vida profissional e pessoal.

Agradeço ao meu orientador, Olavo, um "pai" na minha vida acadêmica, por, em todos esses anos que desenvolvemos esse e outros projetos científicos, também ter sido sempre compreensivo, me ajudando e incentivando a não desistir.

Agradeço também aos meus antigos e atuais gestores, Roger, Walter e Henrique, em nome das empresas Colorado Máquinas e Sylvamo, por terem sido muito compreensivos e flexíveis, me proporcionando a possibilidade de concluir esse Mestrado enquanto me apoiavam em meu crescimento profissional.

Por fim, agradeço aos demais membros da minha família, amigos e companheiros de trabalho que estiveram presentes em minha vida ao longo desses anos e que, direta ou indiretamente, me ajudaram nessa etapa da minha vida.

"Any new realizations would have to wait
Till he had more time
More time
A time to dream
to himself
He waves goodbye to himself
I'll see you on the other side
Another man moved by sleight of hand"
(Vedder, Eddie; Ament, Jeff. "Sleight of Hand". Pearl Jam, Binaural, 2000.)

Resumo

AMENT, G. F. **Método Runge-Kutta Contínuo para equações diferenciais com retardo e aplicações em dinâmica populacional**. 2022. Dissertação (Mestrado em Computação Aplicada) - Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto, Universidade de São Paulo, São Paulo, 2022.

Os métodos de Runge-Kutta (RK) são técnicas bastante conhecidas e amplamente utilizadas para resolver numericamente problemas de valor inicial (PVIs) de equações diferenciais ordinárias. Derivado do RK, o método Runge-Kutta Contínuo (RKC) acrescenta ao anterior uma técnica de interpolação polinomial e produz uma função contínua para aproximar a solução do PVI. Dessa forma, o método RKC pode ser naturalmente estendido para as equações diferenciais com retardo (EDRs), que têm como característica a necessidade da avaliação da solução em momentos anteriores ao atual e que, em geral, não coincidem com um ponto da malha. O método RKC simplifica o processo de obtenção da solução numérica enquanto preserva a precisão e demais qualidades dos métodos de RK. Do ponto de vista das aplicações, as EDRs modelam fenômenos das mais diversas áreas do conhecimento, desde as ciências básicas como Biologia, Física e Química, quanto fenômenos econômicos e sociais. Na área de dinâmica populacional, destacam-se, por exemplo, variações dos modelos clássicos de crescimento malthusiano e logístico e os modelos epidêmicos compartimentais, como o modelo SIR (suscetível-infeccioso-recuperado). Nesse contexto, apresentamos nesse trabalho um estudo do método RKC para solução numérica de EDRs, seus aspectos teóricos, sua implementação computacional e aplicações em exemplos tanto puramente matemáticos quanto relacionados a modelos de dinâmica populacional. O código, desenvolvido em linguagem MATLAB, contempla uma ampla gama de problemas, incluindo as EDRs com retardo constante, dependente do tempo e dependente do estado, bem como sistemas de EDRs. Os resultados mostram que as soluções numéricas obtidas são bastante precisas o que torna o programa desenvolvido promissor para ser aplicado em problemas reais das ciências e engenharias.

Palavras-chave: Métodos Numéricos; Runge-Kutta Contínuo; Modelos de Dinâmica Populacional; Equações Diferenciais com Retardo.

Abstract

AMENT, G. F. **Continuous Runge-Kutta Method for delay differential equations and applications in population dynamics**. 2022. Dissertação (Mestrado em Computação Aplicada) - Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto, Universidade de São Paulo, São Paulo, 2022.

The Runge-Kutta (RK) methods are well-known techniques and widely used to numerically solve initial value problems (IVP) of ordinary differential equations. Derived from RK, the Continuous Runge-Kutta (CRK) method adds to the previous one a polynomial interpolation technique and generates a continuous function to approximate the IVP solution. Therefore, the CRK method can be naturally extended to delay differential equations (DDEs), which have as a feature the necessity to evaluate the solution in previous moments to the current one and that, in general, does not coincide with a mesh point. The CRK method simplifies the process to obtain the numerical solution while maintaining the precision and further qualities of the RK methods. From the application point of view, the DDEs models phenomena of the more diverse knoledgement areas, from the basics sciences, such as Biology, Physics and Chemistry, to economic and social phenomena. In the population dynamics area, stands out, for example, variations of the classic Malthusian and Logistic growth models and the compartmental epidemic model, such as the SIR (susceptible-infectious-recovered) model. In this context, we present in this work a study of the CRK method to numerically solve DDEs, its theoretical aspects, computational implementation and application both in examples that are purely mathematical and examples related to population dynamics models. The code, developed in MATLAB language, covers a wide range of problems, including DDEs with constant, time and state dependent delay, as well as systems of DDEs. The results show that the obtained numerical solutions are quite accurate, making the developed program promising to be applied in real problems of science and engineering.

Keywords: Numerical Methods; Continuous Runge-Kutta; Population Dynamics Models; Delay Differential Equations.

Sumário

1

Introdução

Os métodos de Runge-Kutta (RK) são bastante conhecidos e amplamente utilizados para resolver numericamente problemas de valor inicial (PVIs) de equações diferenciais ordinárias (EDOs). Derivado do RK, o método de Runge-Kutta Contínuo (RKC) combina a mesma técnica de integração numérica do RK, que fornece um conjunto de pontos discretos como aproximação da solução, a uma técnica de interpolação polinomial (HAYASHI, 1996). Assim, o RKC apresenta como resultado uma função contínua que aproxima a solução exata de um PVI.

O método RKC pode ser naturalmente estendido para as equações diferenciais com retardo (EDRs) (BELLEN; VERMIGLIO, 1996) e tem se mostrado adequado para esse propósito, pois para resolver tais equações é necessário o conhecimento da solução em momentos anteriores ao atual, os quais, em geral, não coincidem com os pontos discretos do RK. Já no RKC, conhece-se uma aproximação para a solução em um intervalo anterior ao momento atual.

As EDRs são atualmente objeto de ampla gama de pesquisas por abrangerem problemas mais complexos do que os modelados por EDOs, visto que, as EDOs modelam fenômenos nos quais a taxa de variação da solução no tempo atual depende da própria solução nesse mesmo tempo, enquanto que as EDRs permitem o estudo de fenômenos em que a taxa de variação da solução também dependa da solução calculada em tempos anteriores ao atual. As EDRs, assim como as EDOs, modelam fenômenos nas mais diversas áreas do conhecimento, das ciências básicas como Biologia, Física e Química, a fenômenos econômicos e sociais.

Os modelos de dinâmica populacional permeiam a maioria das áreas de aplicação e englobam modelos clássicos, tais como o modelo de Malthus, o modelo logístico e o modelo epidemiológico SIR (suscetível-infeccioso-recuperado). Os primeiros regem o crescimento ou decrescimento de uma dada população levando ou não em consideração a capacidade do ambiente em sustentar os indivíduos (BOYCE; DIPRIMA, 2015). Neles, é possível incluir termos com retardo que exprimam, por exemplo, um período de gestação e/ou período

para os indivíduos alcançarem a maturidade reprodutiva. Já o último rege a evolução de uma epidemia em uma dada população, sendo possível incluir termos com retardo que exprimam, por exemplo, o período de incubação da doença (KADDAR; ABTA; ALAOUI, 2011).

Nesse contexto, o objetivo desta dissertação é relatar o estudo realizado no sentido de investigar o método RKC para solução numérica de EDRs, com a aplicação em modelos de dinâmica populacional. Especificamente em relação aos métodos numéricos, buscamos reunir alguns resultados teóricos sobre consistência e convergência dos exemplos, desenvolver um algoritmo próprio para o RKC e realizar sua implementação computacional utilizando o ambiente de programação do software MATLAB. Em relação à aplicação, buscamos discutir como os retardos podem ser inseridos nos modelos clássicos de dinâmica populacional e utilizamos o programa computacional desenvolvido para obtenção de solução numérica para esses modelos com o propósito de comparar e entender de que forma o retardo afeta a dinâmica quando comparados aos modelos tradicionais.

Este estudo pretende contribuir com a difusão do conhecimento de métodos numéricos para EDRs. É notório que o método RKC ainda não está devidamente difundido, apesar de seu potencial de aplicação. De fato, segundo nossas pesquisas, ele ainda não aparece em livros didáticos, se restringindo a materiais de nível elevado como livros de pós-graduação e artigos científicos, nos quais não é apresentado com os detalhes necessários para alcançar um público maior. Além disso, materiais que tratam das aplicações das EDRs, em geral, não apresentam a parte numérica, da mesma forma que materiais de métodos numéricos não apresentam a teoria das EDRs ou as aplicações. A coletânea dos assuntos, abarcando os aspectos de cunho teórico, prático e das aplicações, é um dos resultados deste trabalho.

Quanto ao programa computacional desenvolvido, o qual denominamos `rkc4` por se tratar de um RKC de quarta ordem, realizamos testes utilizando problemas com soluções analíticas conhecidas, para efeito de comparação. Também comparamos as soluções aproximadas com a função `ddesd`, nativa do MATLAB. As soluções aproximadas obtidas com o `rkc4` foram bastante precisas em todos os testes. As simulações com os modelos de dinâmica populacional também geraram resultados compatíveis com os relatados na literatura.

Destacamos ainda, que o programa `rkc4` é bastante versátil. Ele pode ser utilizado para uma EDR ou sistemas de EDRs, problemas com retardos constante, dependendo do tempo ou dependendo do estado, bem como problemas que envolvam múltiplos retardos. Além disso, o programa está escrito em forma de função do MATLAB, o que permite que ele seja chamado por outros programas e seja apenas uma parte de um programa maior, sem a necessidade de ajustes e alterações no código.

Na sequência, esse trabalho apresenta, no segundo capítulo, um estudo teórico das

EDOs e das EDRs sob o ponto de vista da Análise Numérica, tratando das condições de existência e unicidade de soluções de PVIs, e, em seguida, apresenta exemplos de EDRs e suas soluções analíticas. Em seu terceiro capítulo, o texto aborda os métodos numéricos de um passo de Euler, Taylor e RK para solução de EDOs, sob os aspectos teóricos, e apresenta com detalhes o RK de quarta ordem e seu algoritmo, bem como discute técnicas de interpolação polinomial . No quarto capítulo, temos o estudo teórico acerca do método RKC4, apresentando e explicando, na sequência, a implementação do algoritmo em linguagem MATLAB e encerrando com os testes e resultado numéricos obtidos com a aplicação do programa implementado. O quinto capítulo apresenta o estudo e aplicação do método RKC em modelos de dinâmica populacional com retardo, sendo primeiramente apresentado o modelo de crescimento logístico e, por fim, o modelo epidemiológico SIR. No último capítulo fazemos as considerações finais e conclusões do projeto.

2

Equações diferenciais com retardo

Neste capítulo apresentamos uma introdução à teoria das equações diferenciais ordinárias (EDOs) do ponto de vista da Análise Numérica. Em particular, destacamos características dos problemas de valor inicial (PVIs) que são desejáveis quando se pretende aplicar métodos numéricos para aproximação de solução. Em seguida, introduzimos o conceito de equações diferenciais com retardo (EDRs) e sua teoria, também com vista à aplicação de métodos numéricos. Por fim, discutimos uma técnica de resolução analítica para EDRs, conhecida por método dos passos, e exemplos.

2.1 Equações diferenciais ordinárias

Uma EDO de primeira ordem é escrita, de forma geral, como

$$y' = f(t, y), \qquad t \in [t_0, t_f], \tag{2.1}$$

onde t é a variável independente que evolui continuamente de t_0 até t_f e em geral está associada ao tempo, f é uma função dada que depende de t e de y, sendo definida de $[t_0, t_f] \times \mathbb{R}$ em $\mathbb{R}$, y' é a derivada de y em relação a t e y é a incógnita da equação.

Uma solução da EDO (2.1) é uma função $y = \phi(t)$ contínua, diferenciável e que satisfaz a equação para todo $t \in [t_0, t_f]$. Apesar de a EDO (2.1) possuir uma família de soluções, em geral estamos interessados em obter uma solução específica que satisfaça determinada condição em algum ponto do intervalo $[t_0, t_f]$, chamada condição inicial, por exemplo,

$$y(t_0) = \alpha, \tag{2.2}$$

onde α é um número real dado.

O conjunto das Eqs. (2.1)-(2.2) é chamado de problema de valor inicial (PVI), cuja solução é uma função $y = \phi(t)$ que, além de ser solução da EDO (2.1) em $[t_0, t_f]$, também satisfaz a condição inicial (2.2) (BOYCE; DIPRIMA, 2015).

Podemos definir, de forma análoga, um sistema de n EDOs de primeira ordem como

$$\begin{cases} y_1' = f_1(t, y_1, y_2, \ldots, y_n), \\ y_2' = f_2(t, y_1, y_2, \ldots, y_n), \\ \vdots \\ y_n' = f_n(t, y_1, y_2, \ldots, y_n), \end{cases} \qquad t \in [t_0, t_f], \tag{2.3}$$

onde $f_1, f_2, \ldots, f_n$ são n funções definidas em $[t_0, t_f] \times \mathbb{R}^n$ com valores em $\mathbb{R}$ e $y_1, y_2, \ldots, y_n$ são as n incógnitas. Além disso, para particularizar a solução, impomos n condições iniciais em um mesmo ponto de $[t_0, t_f]$, por exemplo,

$$y_1(t_0) = \alpha_1, \quad y_2(t_0) = \alpha_2, \quad \ldots, \quad y_n(t_0) = \alpha_n, \tag{2.4}$$

onde $\alpha_1, \alpha_2, \ldots, \alpha_n$ são números reais dados.

O conjunto das Eqs. (2.3)-(2.4) é um PVI, cuja solução é dada por n funções $y_1 = \phi_1(t), \ldots, y_n = \phi_n(t)$ contínuas e diferenciáveis em $[t_0, t_f]$ e que satisfazem ao mesmo tempo as n equações e as n condições iniciais.

O PVI (2.3)-(2.4) pode ser representado na forma vetorial por

$$\begin{cases} y' = f(t, y), \qquad t \in [t_0, t_f], \\ y(t_0) = \alpha, \end{cases} \tag{2.5}$$

onde $y = (y_1, y_2, \ldots, y_n)$, $y' = (y_1', y_2', \ldots, y_n')$, $f_j(t, y) = f_j(t, y_1, y_2, \ldots, y_n)$, $f(t, y) = (f_1(t, y), \ldots, f_n(t, y))$ e $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, para $j = 1, \ldots, n$.

Para resolver um PVI do tipo (2.5), com $n = 1$, existem técnicas analíticas como integração, expansão em séries e transformada de Laplace. Tais técnicas encontram a resposta exata e exibem uma expressão (explícita ou implícita) para a solução. No entanto, essas técnicas podem resolver apenas algumas classes de equações, tais como as lineares, com variáveis separáveis, exatas, entre outras, ou seja, não há uma técnica de alcance geral. Para $n > 1$, as técnicas analíticas são mais escassas e costuma-se fazer análise qualitativa, observando regiões de crescimento e decrescimento da solução e construindo retratos de fase (BOYCE; DIPRIMA, 2015).

Uma alternativa é utilizar métodos numéricos, que podem ser aplicados a uma gama bem maior de equações e sistemas de equações diferenciais e são o foco desse trabalho. Mesmo para esses métodos, no entanto, há restrições. Antes de buscar por uma solução numérica de um PVI, é importante saber se a solução do problema matemático existe e se ela é única. Também é importante verificar se o problema é estável, para que pequenos erros de arredondamento que são cometidos no valor da condição inicial e ao longo dos cálculos subsequentes não levem a solução aproximada para longe da solução exata. Essas três propriedades, existência, unicidade e estabilidade, são agrupadas no conceito de problema bem posto.

Segundo Hadamard (1902), um problema bem posto é aquele em que a solução existe, é única e dependente continuamente dos dados de entrada. Especificamente para o PVI (2.5), Burden e Faires (2008), por exemplo, definem um problema perturbado por um δ_0 e por uma função contínua $\delta(t)$ e mostram sob quais condições a solução do problema original e do perturbado estarão próximas o suficiente. Considerando que a notação $\|\cdot\|$ significa a norma no espaço $\mathbb{R}^n$, pode-se definir um problema bem posto da seguinte forma.

Definição 2.1. O problema de valor inicial (2.5) é dito ser **bem posto** se:

- Existir uma única solução, $y(t)$, para o problema; e
- Existirem constantes $\varepsilon_0 > 0$ e $k > 0$ tais que, para qualquer ε, com $0 < \varepsilon < \varepsilon_0$, sempre que $\delta : [t_0, t_f] \to \mathbb{R}^n$ for contínua com $\| \delta(t) \| < \varepsilon$ para todo t em $[t_0, t_f]$ e quando $\| \delta_0 \| < \varepsilon$, o problema de valor inicial

$$\begin{cases} z' = f(t,z) + \delta(t), & t \in [t_0, t_f], \\ z(t_0) = \alpha + \delta_0, \end{cases}$$

tem uma única solução $z(t)$ que satisfaz $\| z(t) - y(t) \| < k\varepsilon$ para todo t em $[t_0, t_f]$.

A Definição 2.2, a seguir, trata de uma função satisfazer uma condição de Lipschitz na segunda variável (variável associada ao estado). Esta é uma característica desejável para a função f do PVI (2.5), pois está relacionada com a garantia de que o PVI seja bem posto, conforme mostra o Teorema 2.1 (BURDEN; FAIRES, 2008).

Definição 2.2. Uma função $f : D \subset \mathbb{R}^{n+1} \to \mathbb{R}^n$ satisfaz uma **condição de Lipschitz** na segunda variável no conjunto D se existir uma constante $L > 0$ tal que

$$\| f(t, y_1) - f(t, y_2) \| \leq L \| y_1 - y_2 \| , \tag{2.6}$$

para todo (t, y_1) e $(t, y_2) \in D$. A constante L é chamada de **constante de Lipschitz** da f.

Teorema 2.1. Suponha que $D = [t_0, t_f] \times \mathbb{R}^n$. Se f for contínua e satisfizer a uma condição de Lipschitz na segunda variável no conjunto D, então o PVI (2.5) é bem-posto.

Após essa breve introdução às EDOs, seguimos com o estudo de definições e resultados análogos para o contexto das EDRs.

2.2 Introdução às EDRs

As equações diferenciais funcionais compõem uma grande classe de equações diferenciais em que a variável de estado, e possivelmente suas derivadas, possui argumentos

calculados em instantes distintos. Contidas nessa grande classe, estão as EDRs, que são definidas como sendo equações diferenciais funcionais que contém argumentos em instantes passados apenas na própria variável de estado.

Assim, a principal diferença entre EDOs e EDRs é que nas primeiras, tanto a variável de estado, y, quanto sua derivada, y', estão calculadas no instante atual t. Já nas EDRs, a derivada, y', está calculada no tempo t, enquanto que a variável de estado, y, aparece calculada em um ou mais tempos passados. Por exemplo, podemos escrever uma EDR de primeira ordem como

$$y'(t) = f(t, y(t), y(t-\sigma_1), y(t-\sigma_2), \ldots, y(t-\sigma_d)), \quad t \in [t_0, t_f], \tag{2.7}$$

em que a função f dada depende não somente de t e y calculada em t, mas também de y calculada em instantes distintos $t-\sigma_j$, com $j = 1, \ldots, d$. Para simplificar o estudo teórico de EDRs, consideramos $d = 1$, porém, os resultados apresentados podem ser generalizados para $d > 1$ de forma bastante direta.

Primeiramente, estudamos o caso de uma única equação,

$$y'(t) = f(t, y(t), y(t-\sigma)), \quad t \in [t_0, t_f], \tag{2.8}$$

em que $f : [t_0, t_f] \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$.

O termo $t - \sigma$ é chamado de argumento de retardo (do inglês, *delay argument*) e σ é a função de retardo (em inglês, *delay function*). Podemos classificar as EDRs de acordo com a função σ. O retardo pode ser discreto, se σ for constante; dependente do tempo, se $\sigma = \sigma(t)$; ou dependente do estado, se $\sigma = \sigma(t, y(t))$ (HAYASHI, 1996). Em particular, para as EDRs impõe-se a condição $t - \sigma < t$ pois, do contrário, teríamos uma equação com avanço.

Diferentemente das EDOs, para uma EDR é necessário conhecer a solução em um intervalo anterior a t_0. Assim, descrevemos a condição inicial associada à EDR (2.8) por

$$y(t) = \varphi(t), \quad t \in [t_0 - p, t_0], \tag{2.9}$$

chamada de história, em que φ é uma função dada, definida de $[t_0 - p, t_0]$ em $\mathbb{R}$. Em relação a p, temos que $p = \sigma$ no caso de retardo discreto ou $p = \max\{|t - \sigma| : t_0 \leq t \leq t_f\}$ para os demais tipos de retardo, ou ainda, o intervalo pode ser infinito se φ estiver definida em $t \leq t_0$.

Outra diferença entre EDOs e EDRs é que a imposição de uma condição inicial qualquer pode não garantir a particularização da solução, entre outras consequências, como a propagação de descontinuidade nas derivadas da solução e bifurcações (BELLEN;

ZENNARO, 2003). A regularidade da função história é importante. Por isso, neste trabalho, consideramos φ contínua.

Uma solução do problema (2.8)-(2.9) em $[t_0 - p, t_f]$ é uma função $y = \phi(t)$ contínua em $[t_0 - p, t_f]$, diferenciável em (t_0, t_f) e que satisfaz tanto a condição inicial $y(t) = \varphi(t)$ em $[t_0 - p, t_0]$ quanto a equação diferencial em $[t_0, t_f]$.

A Eq. (2.8) pode ser estendida para um sistema com n equações diferenciais da seguinte forma

$$\begin{cases} y_1' = f_1(t, y_1(t), \ldots, y_n(t), y_1(t-\sigma), \ldots, y_n(t-\sigma)), \\ y_2' = f_2(t, y_1(t), \ldots, y_n(t), y_1(t-\sigma), \ldots, y_n(t-\sigma)), \\ \vdots \\ y_n' = f_n(t, y_1(t), \ldots, y_n(t), y_1(t-\sigma), \ldots, y_n(t-\sigma)), \end{cases} \quad t \in [t_0, t_f], \tag{2.10}$$

e um conjunto de condições iniciais,

$$\begin{cases} y_1(t) = \varphi_1(t), \\ y_2(t) = \varphi_2(t), \\ \vdots \\ y_n(t) = \varphi_n(t), \end{cases} \quad t \in [t_0 - p, t_0]. \tag{2.11}$$

Neste trabalho, estudamos a teoria e os métodos numéricos para os sistemas de EDRs com retardo dependendo do estado. Assim, para n equações e tomando $\sigma = \sigma(t, y(t))$, o sistema (2.10)-(2.11) estudado é representado na forma vetorial por

$$\begin{cases} y'(t) = f(t, y(t), y(t - \sigma(t, y(t)))), & t \in [t_0, t_f], \\ y(t) = \varphi(t), & t \in [t_0 - p, t_0], \end{cases} \tag{2.12}$$

em que $f : [t_0, t_f] \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$, $\sigma : [t_0, t_f] \times \mathbb{R}^n \to [0, p]$ e $\varphi : [t_0 - p, t_0] \to \mathbb{R}^n$.

Como nas EDOs, é desejável que o PVI (2.12) seja bem posto. Para isso, estendemos o conceito da f satisfazer a condição de Lipschitz para as duas últimas variáveis em um conjunto $D = [t_0, t_f] \times \mathbb{R}^n \times \mathbb{R}^n$ se existir uma constante $L_f > 0$ tal que

$$\| f(t, y_1, z_1) - f(t, y_2, z_2) \| \leq L_f(\| y_1 - y_2 \| + \| z_1 - z_2 \|), \tag{2.13}$$

para todo (t, y_1, z_1) e (t, y_2, z_2) em D. Também assumimos que as funções φ e σ satisfazem condições de Lipschitz no seguinte sentido

$$\| \varphi(t) - \varphi(s) \| \leq L_\varphi \| t - s \| , \tag{2.14}$$

para todo $t, s \in [t_0 - p, t_0]$, e

$$\| \sigma(t, y_1) - \sigma(t, y_2) \| \leq L_\sigma \| y_1 - y_2 \| , \tag{2.15}$$

para todo $(t, y_1), (t, y_2) \in [t_0, t_f] \times \mathbb{R}^n$.

Assim, segue um teorema com condições suficientes para o PVI (2.12) ser bem posto (HARTUNG et al., 2006).

Teorema 2.2. Suponha que f, σ e φ sejam funções continuas e satisfazem as condições de Lipschitz (2.13)-(2.15). Então o PVI (2.12) é bem posto.

Do ponto de vista da Análise Numérica, o Teorema 2.2 nos dá suporte para buscarmos por uma solução numérica, sabendo que a solução analítica existe e é única, e sabendo que pequenos erros causados por arredondamento nas contas efetuadas pelo computador não afetará a solução numérica de forma a afastá-la da solução analítica.

Antes de iniciarmos o estudo dos métodos numéricos para EDRs, vamos entender como elas são resolvidas analiticamente. A técnica disponível, chamada de método dos passos, consiste em dividir o intervalo $[t_0, t_f]$ em subintervalos, chamados de passos, e em cada passo resolvemos um PVI local da forma

$$\begin{cases} [y^{(j)}]'(t) = f(t, y^{(j)}(t), y(t - \sigma(t, y^{(j)}(t)))), & t \in [\xi_j, \xi_{j+1}], \\ y^{(j)}(\xi_j) = y(\xi_j), \end{cases} \tag{2.16}$$

em que $\xi_0 = t_0$ e os demais ξ_j são calculados de modo que o argumento de retardo $t - \sigma(t, y^{(j)}(t))$ caia para trás de ξ_j, onde a solução y já é conhecida. Especificamente, queremos resolver o PVI local (2.16) enquanto $t - \sigma(t, y^{(j)}(t)) < \xi_j$ e, para isso, escolhemos ξ_{j+1} como sendo um valor que satisfaça a equação

$$\xi_{j+1} - \sigma(\xi_{j+1}\ , y^{(j)}(\xi_{j+1})) = \xi_j. \tag{2.17}$$

Procedendo assim, o terceiro argumento da f em (2.16) é conhecido dos passos anteriores, a EDR se torna uma EDO e obtemos um PVI como o (2.5), cuja incógnita é $y^{(j)}$.

Ao final de cada passo, acrescentamos uma parte à solução do problema original (2.12), obtendo

$$y(t) = \begin{cases} \varphi(t), & t \in [-p, t_0], \\ y^{(1)}(t), & t \in [\xi_0, \xi_1], \\ \vdots & \\ y^{(j)}(t), & t \in [\xi_j, \xi_{j+1}] \\ \vdots & \end{cases} \tag{2.18}$$

O método dos passos é utilizado na Seção 2.3 para resolver alguns casos bem particulares de EDRs em que os PVIs locais são de resolução simples.

2.3 Exemplos de EDRs

Nesta seção apresentamos alguns exemplos de como resolver EDRs pelo método dos passos. O primeiro exemplo foi retirado de Bellen e Zennaro (2003) e os demais de Paul (1994). Para o primeiro exemplo, apresentamos a resolução com detalhes. Para os demais, apresentamos o problema, a solução e o gráfico da solução. O objetivo é, além de explicar o método dos passos, estabelecer um conjunto de problemas com diferentes características que, mais adiante, serão usados como exemplos para obtenção da solução numérica e comparação entre as soluções exata e numérica.

Exemplo 2.1. Considere o problema (2.8) com retardo constante $\sigma = 1$

$$\begin{cases} y'(t) = -y(t-1), & t \in [0,3], \\ y(t) = 1, & t \in [-1,0]. \end{cases} \tag{2.19}$$

Observe que $f(t,y) = -y$, $\varphi(t) = 1$ e $\sigma(t,y) = 1$ satisfazem as hipóteses do Teorema 2.2. Logo, esse problema tem uma única solução em $[-1,3]$. Iniciamos o métodos dos passos pela construção da solução no intervalo $[-1,0]$, conforme Eq. (2.18), obtendo

$$y(t) = 1, \qquad t \in [-1,0].$$

Seguindo para o próximo intervalo, $t \in [\xi_0, \xi_1]$, em que $\xi_0 = 0$ e ξ_1 é determinado resolvendo a equação (2.17), isto é, $\xi_1 - 1 = 0$, de onde obtemos que $\xi_1 = 1$.

Para $t \in [0,1]$, o argumento de retardo $t - 1 \in [-1,0]$, intervalo onde y já é conhecida e, portanto, $[y^{(1)}]'(t) = -y(t-1) = -1$. Além disso, a condição inicial é dada por $y^{(1)}(0) = y(0) = 1$. Dessa forma, o primeiro PVI local é definido por

$$\begin{cases} [y^{(1)}]' = -1, & t \in [0,1], \\ y^{(1)}(0) = 1. \end{cases}$$

Integrando a equação diferencial, temos que $y^{(1)}(t) = -t + c$. Impondo a condição inicial, $y^{(1)}(0) = -0 + c = 1$, logo $c = 1$. Portanto, obtemos que $y^{(1)}(t) = -t + 1$, para $t \in [0,1]$, e a solução torna-se

$$y(t) = \begin{cases} 1, & t \in [-1,0], \\ -t+1, & t \in [0,1]. \end{cases}$$

Realizando o mesmo processo para o próximo intervalo $t \in [\xi_1, \xi_2]$, obtemos $\xi_2 = 2$. Para $t \in [1,2]$, o argumento de retardo $t - 1 \in [0,1]$, onde y já é conhecida. Portanto, $[y^{(2)}]'(t) = -y(t-1) = -((-t+1)+1) = t - 2$ e a condição inicial é dada por $y^{(2)}(1) = y(1) = 0$. Assim, obtemos o segundo PVI,

$$\begin{cases} [y^{(2)}]' = t-2, & t \in [1,2], \\ y^{(2)}(1) = 0. \end{cases}$$

Integrando a EDO, obtemos que $y^{(2)}(t) = t^2/2 - 2t + c$. Impondo a condição inicial, $y^{(2)}(1) = 1^2/2 - 2(1) + c = 0$, logo, $c = 3/2$. Acrescentando esse trecho à solução, obtemos

$$y(t) = \begin{cases} 1, & t \in [-1,0], \\ -t+1, & t \in [0,1], \\ \dfrac{t^2}{2} - 2t + \dfrac{3}{2}, & t \in [1,2]. \end{cases}$$

Por fim, para o intervalo em $t \in [\xi_2, \xi_3]$, temos $\xi_3 = 3$, e o PVI local

$$\begin{cases} y'^{(3)} = -y(t-1) = -t^2/2 + 3t - 4, & t \in [2,3], \\ y^{(3)}(2) = y(2) = -1/2. \end{cases}$$

Integrando a EDO, temos que $y^{(3)}(t) = -t^3/6 + 3t^2/2 - 4t + c$. Impondo a condição inicial, $y^{(3)}(2) = -2^3/6 + 3(2^2)/2 - 4(2) + c = -1/2$, obtemos $c = 17/6$. Com isso, completamos a solução no intervalo em que o problema (2.19) está definido,

$$y(t) = \begin{cases} 1, & t \in [-1,0], \\ -t+1, & t \in [0,1], \\ \dfrac{t^2}{2} - 2t + \dfrac{3}{2}, & t \in [1,2], \\ -\dfrac{t^3}{6} + \dfrac{3t^2}{2} - 4t + \dfrac{17}{6}, & t \in [2,3], \end{cases} \tag{2.20}$$

cujo gráfico pode ser visto na Fig. 1.

Exemplo 2.2. Considere o sistema de duas equações diferenciais e um retardo constante $\sigma = 1$

$$\begin{cases} y_1'(t) = y_2(t) & t \geq 0, \\ y_2'(t) = 1 - y_2(t-1) - y_1(t) & t \geq 0, \\ y_1(t) = 0 & t \leq 0, \\ y_2(t) = 0 & t \leq 0. \end{cases} \tag{2.21}$$

Nesse exemplo, temos que $\varphi_1(t) = \varphi_2(t) = 0$ para qualquer $t \leq 0$. Além disso, $f_1(t, x_1, x_2, x_3, x_4) = x_2$ e $f_2(t, x_1, x_2, x_3, x_4) = 1 - x_4 - x_1$ são contínuas e Lipschitz em $\mathbb{R}^5$, o que nos permite afirmar que este problema tem solução única para qualquer $t \in \mathbb{R}$.

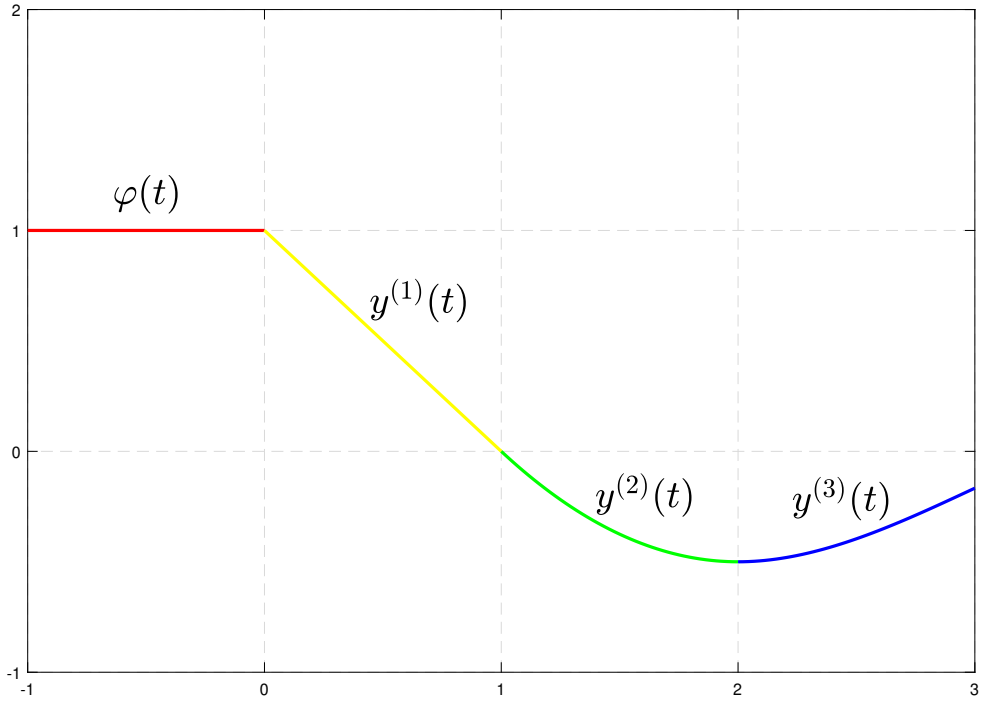


Figura 1 – Gráfico da solução do Exemplo 2.1, dada pela Eq. (2.20), obtida pelo método dos passos, sendo cada parte representado por uma cor.

A solução desse problema pode ser obtida pelo método dos passos aplicado em intervalos de comprimento 1 e é dada por

$$\begin{aligned} y_1(t) = &\begin{cases} 0, & t \in (-\infty, 0], \\ 1 - \cos(t), & t \in [0,1], \\ 1 - \cos(t) + \frac{1}{2}(t-1)\cos(t-1) - \frac{1}{2}\operatorname{sen}(t-1), & t \in [1,2], \end{cases} \\ y_2(t) = &\begin{cases} 0, & t \in (-\infty, 0], \\ \operatorname{sen}(t), & t \in [0,1], \\ \operatorname{sen}(t) + \frac{1}{2}(1-t)\operatorname{sen}(t-1), & t \in [1,2], \end{cases} \end{aligned} \tag{2.22}$$

cujo gráfico, para $t \in [0,2]$, pode ser visto na Fig. 2.

Exemplo 2.3. Considere o o sistema com retardo dependendo do tempo

$$\begin{cases} y_1'(t) = y_2(t), & t \geq 2, \\ y_2'(t) = -\frac{1}{2}y_1(t) - \frac{1}{2} + y_1\left(\frac{1}{2}t - \frac{\pi}{4}\right)^2, & t \geq 2, \\ y_1(t) = \operatorname{sen}(t), & t \leq 2, \\ y_2(t) = \cos(t), & t \leq 2. \end{cases} \tag{2.23}$$

Para esse problema, temos $t - \sigma = (1/2)t - \pi/4$ e, portanto, $\sigma(t) = (1/2)t + \pi/4$. Além disso, as expressões para as funções do lado direito das equações diferenciais são

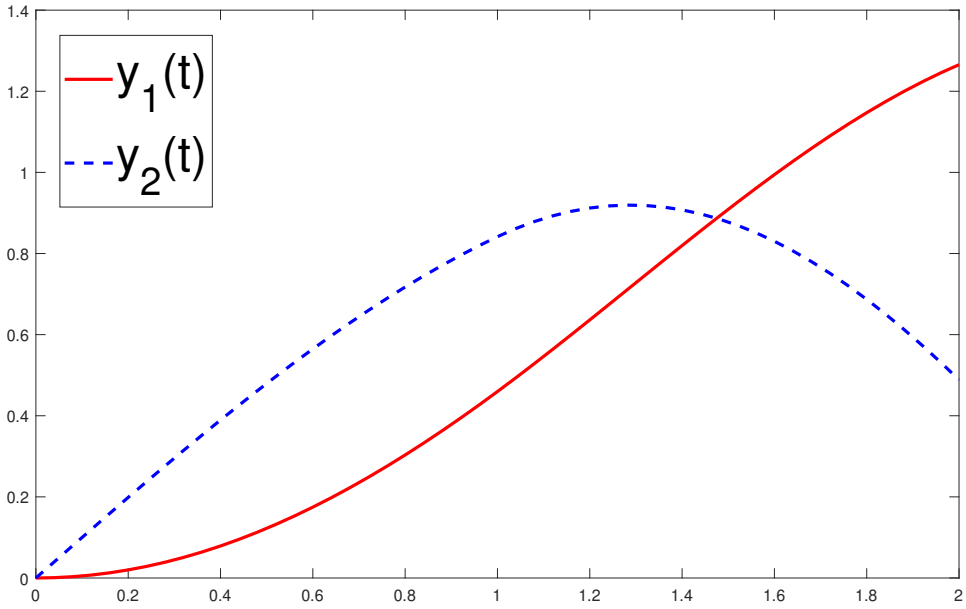


Figura 2 – Solução do Exemplo 2.2, conforme Eq. (2.22), para o intervalo $[0, 2]$.

$f_1(t, x_1, x_2, x_3, x_4) = x_2$, $f_2(t, x_1, x_2, x_3, x_4) = -\frac{1}{2}x_1 - \frac{1}{2} + x_3^2$ e as condições iniciais são $\varphi_1(t) = \text{sen}(t)$ e $\varphi_1(t) = \cos(t)$. Todas as funções envolvidas satisfazem as condições do Teorema 2.2 (para mostrar que f_2 é Lipschitz precisamos de uma análise mais detalhada do comportamento dessa função), portanto, o Problema (2.23) tem uma única solução.

A solução analítica pode ser obtida pelo método dos passos, sendo o prolongamento das funções seno e cosseno definidas como condição inicial, isto é,

$$\begin{aligned} y_1(t) &= \text{sen}(t) \quad t \in \mathbb{R}, \\ y_2(t) &= \cos(t) \quad t \in \mathbb{R}, \end{aligned} \tag{2.24}$$

cujo gráfico, no intervalo $t \in [0, 10]$, é mostrado na Fig. 3.

Exemplo 2.4. Considere o problema com retardo dependendo do estado

$$\begin{cases} y'(t) = \dfrac{y(t)y(\ln(y(t)))}{t}, & t \geq 1, \\ y(t) = 1, & t \leq 1. \end{cases} \tag{2.25}$$

Neste problema, $f(t, x_1, x_2) = x_1x_2/t$, $\varphi(t) = 1$ e $\sigma(t, y) = t - \ln(y(t))$. É possível mostrar que todas satisfazem as hipóteses do Teorema (2.2) e concluir que o Problema (2.25) tem uma única solução.

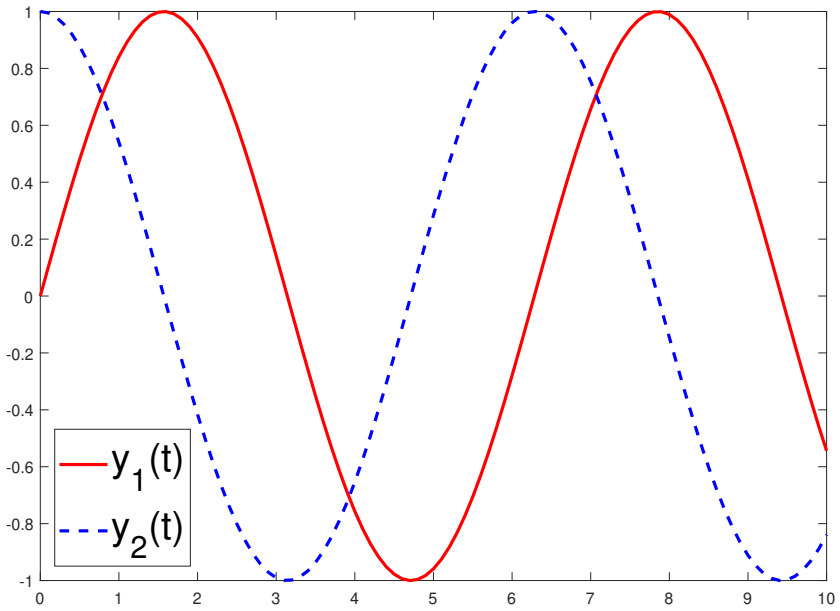


Figura 3 – Solução do Exemplo 2.3, dada pela Eq. (2.24), para o período $t \in [0, 10]$.

Neste problema, a solução analítica é dada por,

$$y(t) = \begin{cases} t, & t \in [1, e], \\ \exp\left(\dfrac{t}{e}\right), & t \in [e, e^2], \end{cases} \tag{2.26}$$

cujo gráfico da solução analítica está representado na Fig. 4.

Os exemplos acima nos mostram que, em geral, não é simples obter soluções analíticas de EDRs. Em particular, conseguimos resolver uma EDR quando a EDO associada a cada trecho é passível de ser resolvida. Mesmo nesse caso, efetuar os cálculos à mão pode ser viável apenas em curtos intervalos de t.

Resumidamente, quando o retardo é discreto, caminha-se de σ em σ unidades e para cada passo, busca-se pela solução. Quando o retardo é dependente do tempo, o tamanho dos passos não é constante. No passo j, conhecido ξ_j, determina-se ξ_{j+1} resolvendo a equação $t - \sigma(t) = \xi_j$, que depende de t.

No entanto, quando o retardo é dependente do estado, a Eq. (2.17) depende da solução, que deve ser monitorada, de forma que o j-ésimo PVI local (2.16) seja resolvido enquanto $t - \sigma(t, y(t)) < \xi_j$. Quando a solução for tal que a igualdade $t - \sigma(t, y(t)) = \xi_j$ é satisfeita, $t = \xi_{j+1}$ e se encerra o passo j. Esse monitoramento torna o processo lento e trabalhoso, podendo inclusive impedir que se obtenha a solução analítica.

Assim, os métodos numéricos representam uma técnica alternativa viável para estudar a solução das EDRs. Este é o assunto dos dois próximos capítulos.

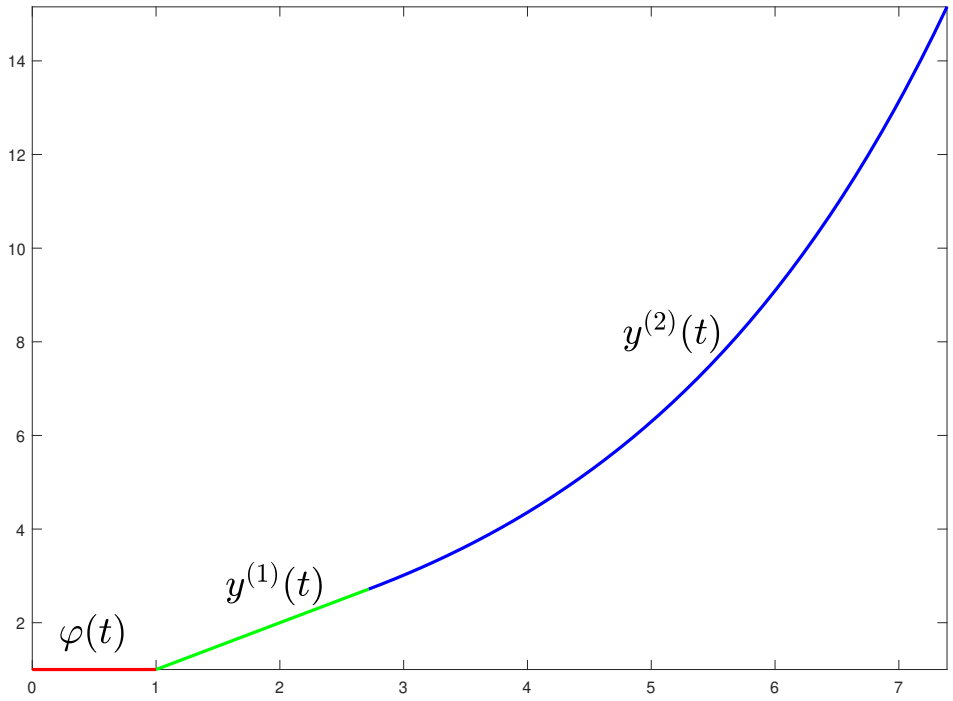


Figura 4 – Solução do Exemplo 2.4, conforme Eq. (2.26), para $t \in [0, e^2]$.

3

Métodos numéricos para EDOs

Neste capítulo apresentamos um estudo dos métodos numéricos de um passo (ou de passo simples) para aproximação de soluções de EDOs. Em específico, abordamos os métodos de Euler, Taylor e Runge-Kutta (RK) e seus aspectos teóricos no que diz respeito aos conceitos de consistência, convergência e estabilidade. Focamos com mais detalhes no Runge-Kutta de quarta ordem (RK4) e trazemos um exemplo resolvido por esse método. Por fim, discutimos a técnica de interpolação polinomial, que neste trabalho é utilizada para obtenção de aproximação da solução de EDOs em qualquer ponto do intervalo considerado ou para obtenção de uma solução numérica densa. A abordagem dos tópicos desse capítulo tem o objetivo de fornecer uma base teórica ao método de Runge-Kutta Contínuo (RKC), que está descrito no Capítulo 4.

3.1 Introdução aos métodos numéricos

Retomamos aqui o PVI (2.5),

$$\begin{cases} y'(t) = f(t, y(t)), \quad t \in [t_0, t_f], \\ y(t_0) = \alpha, \end{cases} \tag{3.1}$$

o qual queremos, agora, resolver numericamente.

A primeira etapa de aplicação de um método numérico é definir uma malha, ou seja, um conjunto de pontos discretos no intervalo $[t_0, t_f]$. Para isso, escolhemos um número natural $N \geq 2$ e definimos os pontos $t_i = t_0 + ih_i$, com $i = 0, 1, \ldots, N$, tal que $h_i = t_{i+1} - t_i > 0$. No caso específico de uma malha igualmente espaçada, tem-se $h_i = h = (t_f - t_0)/N$, a qual, por simplicidade, adotamos neste trabalho. Dessa forma, a malha utilizada nos métodos numéricos deste trabalho é definida como

$$\Delta = \{t_i = t_0 + ih, \quad i = 0, 1, ..., N \mid N \in \mathbb{N},\ N \geq 2,\ h = (t_f - t_0)/N\}. \tag{3.2}$$

Em seguida, definimos um processo iterativo para calcular uma aproximação para a solução em cada ponto da malha, começando da condição inicial, $w^{(0)} = y(t_0) = \alpha$ e calculando sucessivamente $w^{(1)}, w^{(2)}, \ldots, w^{(N)}$ por meio de uma equação algébrica. O conjunto de pontos $(t_i, w^{(i)})$, $i = 0, 1, \ldots, N$, é chamado de solução numérica, tal que $w^{(i)} \approx y(t_i)$, onde $y(t)$ é a solução exata do PVI (3.1) em $[t_0, t_f]$.

Os métodos de um passo, entre eles Euler, Taylor e RK, são assim classificados pois, para calcular uma aproximação para a solução em t_{i+1}, $w^{(i+1)} \approx y(t_{i+1})$, utilizam informação apenas do ponto anterior, $(t_i, w^{(i)})$. Assim, as sucessivas aproximações por meio do processo iterativo podem ser descritas por

$$\begin{cases} w^{(0)} = \alpha, \\ w^{(i+1)} = w^{(i)} + h\phi(t_i, w^{(i)}, h), \quad i = 0, 1, \ldots, N-1, \end{cases} \tag{3.3}$$

em que $\phi : [t_0, t_f] \times \mathbb{R}^n \times (0, \infty) \to \mathbb{R}^n$ é uma função que caracteriza cada método numérico.

Para obter a expressão de ϕ em um método de um passo, expandimos a solução $y(t)$ em série de Taylor em torno de t_i, calculamos em $t = t_{i+1}$ e truncamos após o termo com derivada de ordem p de $y(t)$, deixando o termo seguinte para o erro, ou seja,

$$y(t_{i+1}) = y(t_i) + hy'(t_i) + \cdots + \frac{h^p}{p!}y^{(p)}(t_i) + \frac{h^{p+1}}{(p+1)!}y^{(p+1)}(\xi_i), \tag{3.4}$$

para algum $\xi_i \in (t_i, t_{i+1})$.

O método de Euler, ou da reta tangente, é o método de um passo mais simples, obtido tomando $p = 1$ em (3.4) e substituindo $y'(t_i)$ por $f(t_i, y(t_i))$, resultando em

$$y(t_{i+1}) = y(t_i) + hf(t_i, y(t_i)) + \frac{h^2}{2}y''(\xi_i). \tag{3.5}$$

Na Eq. (3.5), descartando o termo do erro e usando a notação aproximada $w^{(i)} \approx y(t_i)$, obtemos a fórmula do método de Euler,

$$w^{(i+1)} = w^{(i)} + hf(t_i, w^{(i)}). \tag{3.6}$$

A Eq. (3.6) é a equação algébrica que gera uma solução numérica para o PVI (3.1) quando variamos i de 0 a $N-1$, uma vez que $w^{(i)}$ já é conhecido do passo anterior, h é dado e f tem expressão conhecida e pode ser avaliada em $(t_i, w^{(i)})$. Comparando (3.6) com a fórmula geral dos métodos de um passo (3.3), concluímos que, para o método de Euler, $\phi(t, y, h) = f(t, y)$.

Os métodos de Taylor de ordem p são obtidos considerando $p \geq 1$, sendo que, como acabamos de ver, $p = 1$ corresponde ao método de Euler. Para obter a função ϕ, observamos que as derivadas da expressão (3.4) não são conhecidas explicitamente. Usando o fato de que $y(t)$ satisfaz a EDO em (3.1) podemos obter as derivadas de ordem

superior diferenciando a EDO sucessivamente. No entanto, os cálculos dessas derivadas são complexos e as expressões ficam extensas, pois envolvem derivadas parciais da f e regra do produto. Para simplificar, adota-se a notação

$$y''(t) = \frac{d}{dt}f(t, y(t)) = f'(t, y(t)),$$

$$y^{(3)}(t) = \frac{d}{dt}f'(t, y(t)) = f''(t, y(t)),$$

e assim por diante.

Assim, para o método de Taylor de ordem p, temos o processo iterativo

$$w^{(i+1)} = w^{(i)} + hT^{(p)}(t_i, w^{(i)}, h), \tag{3.7}$$

onde

$$\phi(t, y, h) = T^{(p)}(t, y, h) = f(t, y) + \frac{h}{2!}f'(t, y) + ... + \frac{h^{p-1}}{p!}f^{(p-1)}(t, y). \tag{3.8}$$

O desejado é que a solução numérica nos pontos da malha, $w^{(0)}, w^{(1)}, \ldots, w^{(N)}$, esteja suficientemente próxima da solução exata nesses mesmos pontos. Para alcançar esse objetivo, reforçamos que o PVI deve ser bem posto. Além disso, precisamos também de algumas condições sobre o método numérico. Especificamente, o método escolhido deve ser consistente, convergente e estável (BURDEN; FAIRES, 2008; FUKUSHIMA, 2018). Tais conceitos são definidos a seguir, seguido de um teorema que trata de condições suficientes para alcançarmos uma solução numérica confiável.

O erro de truncamento local (ETL) de um método mede a quantidade, em um passo específico, pela qual a solução do método usado na aproximação deixa de satisfazer a solução exata da equação diferencial.

Definição 3.1. Os métodos de um passo, dados por (3.3), têm **erro de truncamento local** dado por

$$E_{i+1}(h) = \frac{y(t_{i+1}) - (y(t_i) + h\phi(t_i, y(t_i), h))}{h}, \tag{3.9}$$

para cada $i = 0, 1, ..., N - 1$.

Na Definição 3.1, $y(t_{i+1})$ é a solução exata em t_{i+1} enquanto que $y(t_i)+h\phi(t_i, y(t_i), h)$ é a solução aproximada pelo método (3.3), considerando que a solução anterior $y(t_i)$ é exata.

Utilizando a Eq. (3.5) no método de Euler, por exemplo, é possível ver que $y(t_{i+1}) - (y(t_i) + hf(t_i, y(t_i))) = \frac{h^2}{2}y''(\xi_i)$. Supondo $y''(t)$ limitada em $[t_0, t_f]$, digamos por M, temos que

$$\| E_{i+1}(h) \| \leq \frac{h}{2}M,$$

de onde concluímos que o método de Euler tem ETL da ordem de h. Em outras palavras, o método de Euler é $O(h)$.

Já no caso do método de Taylor, temos que o erro de truncamento da série é

$$\frac{h^{p+1}y^{(p+1)}(\xi_i)}{(p+1)!},$$

enquanto que o ETL é

$$E_{i+1}(h) = \frac{y(t_{i+1}) - (y(t_i) - hT^{(p)}(t_i, y(t_i), h))}{h} = \frac{h^p}{(p+1)!}y^{(p+1)}(\xi_i), \tag{3.10}$$

ou seja, se $y^{(p+1)}$ for limitada em $[t_0, t_f]$, digamos por M, então

$$\| E_{i+1}(h) \| \leq \frac{h^p}{(p+1)!}M,$$

para todo $i = 0, 1, \ldots, N$. Logo, o ETL do método de Taylor é $O(h^p)$.

A definição de consistência, a seguir, diz que um método de um passo é consistente se a equação algébrica do método se aproxima da equação diferencial, quando o tamanho do passo h tende a zero.

Definição 3.2. Os métodos de um passo, dados por (3.3) e com erro de truncamento local $E_i(h)$ no i-ésimo passo, são ditos **consistentes** com a equação diferencial que aproximam se

$$\lim_{h\to 0} \max_{0\leq i\leq N} \| E_i(h) \| = 0. \tag{3.11}$$

Podemos assim constatar que ambos os métodos que abordamos nessa seção são consistentes, visto que no método de Euler $\lim_{h\to 0} \frac{h}{2}M = 0$, enquanto que no método de Taylor $\lim_{h\to 0} \frac{h^p}{(p+1)!}M = 0$.

A respeito da convergência, um método é convergente se a solução da equação algébrica se aproxima da solução da equação diferencial quando o tamanho do passo h tende a zero.

Definição 3.3. Os métodos de um passo, dados por (3.3), são ditos **convergentes** com relação à equação diferencial que aproximam se

$$\lim_{h\to 0} \max_{i=0,1,\ldots,N} \| w^{(i)} - y(t_i) \| = 0, \tag{3.12}$$

em que $y(t_i)$ denota o valor exato da solução da equação diferencial e $w^{(i)}$ é a aproximação obtida a partir do método no i-ésimo passo.

Em geral, não é possível calcular o limite na Eq. (3.12), pois não conhecemos a solução exata nos pontos. Porém, podemos fazer uma estimativa utilizando o Teorema 3.1, apresentado abaixo, após o conceito de estabilidade.

Por último, a **estabilidade** de um método numérico é tomada no sentido de que pequenas variações ou perturbações nas condições iniciais e nas aritméticas subsequentes produzem mudanças correspondentemente pequenas na solução numérica. Na prática, resolvemos, ao invés de (3.3), um problema do tipo

$$\begin{cases} \hat{w}^{(0)} = \alpha + \delta^{(0)}, \\ \hat{w}^{(i+1)} = \hat{w}^{(i)} + h\phi(t_i, \hat{w}^{(i)}, h) + \delta^{(i+1)}, \quad i = 0, 1, ..., N-1, \end{cases} \tag{3.13}$$

onde $\delta^{(i)}$ representa um pequeno vetor de erro de arredondamento ocorrido no passo i do processo iterativo e cada $\delta^{(i)}$ é limitado por uma única constante positiva δ, ou seja, $\| \delta^{(i)} \| < \delta$.

O conceito de estabilidade de um método de um passo é de certa forma análogo à condição de uma equação diferencial ser bem posta, por isso é natural que a condição de Lipschitz apareça no teorema seguinte.

Teorema 3.1. Suponha que o problema de valor inicial (3.1) seja aproximado por um método de um passo na forma (3.3). Suponha também que exista um número $h_0 > 0$ e que $\phi(t, y, h)$ seja contínua e satisfaça uma condição de Lipschitz na variável y com a constante de Lipschitz L no conjunto $D = [t_0, t_f] \times \mathbb{R}^n \times (0, h_0)$. Então

(i) o método é estável;

(ii) se o método for consistente, o que é equivalente a $\phi(t, y, 0) = f(t, y), \quad t \in [t_0, t_f]$, então o método é convergente;

(iii) se o método for consistente e o ETL for $O(h^p)$ então a ordem global do erro é p, isto é,

$$\max_{i=1,...,N} \| y(t_i) - w^{(i)} \| = O(h^p).$$

A demonstração deste teorema pode ser encontrada em Gear (1971, p.57). Observe que a parte (i) diz respeito à estabilidade e a parte (ii) refere-se às condições suficientes para que um método consistente seja convergente. Quanto à parte (iii), ela diz que o erro global possui a mesma ordem de grandeza do erro de truncamento local.

Observamos que as funções ϕ dos métodos de Euler e Taylor são ambas contínuas e Lipschitz desde que a função f também seja e $\phi(t, y, 0) = f(t, y)$ para todo $t \in [t_0, t_f]$. Com isso, concluímos, pelo Teorema 3.1, que ambos os métodos são consistentes, convergentes e estáveis, desde que o problema (3.1) seja bem posto. Além disso, o método de Euler possui erro global $O(h)$, desde que a derivada segunda da solução seja limita, enquanto que o método de Taylor possui erro global $O(h^p)$, desde que a derivada de ordem $p+1$ seja limitada.

Diante dos resultados acima, é importante observar que o ETL do método de Euler é de baixa ordem, pois ele é proporcional a h. Na prática, isso significa que para o método

de Euler atingir uma precisão razoável, precisamos de h pequeno, implicando em uma grande quantidade de iterações do processo (3.6), o que demanda tempo computacional e propicia a propagação de erros de arredondamento. Já os métodos de Taylor são raramente utilizados pela necessidade de calcular as derivadas de alta ordem da função f. A importância desses últimos métodos está no fato deles darem origem aos métodos de Runge-Kutta, abordados a seguir.

3.2 Métodos de Runge-Kutta

Os métodos de RK são amplamente conhecidos e utilizados, pois conseguem alcançar a precisão dos métodos de Taylor sem a necessidade de calcular as derivadas de alta ordem da função f (CHAPRA; CANALE, 2015). São obtidos pela substituição da função $T^{(p)}(t, y, h)$ por uma expressão que não dependa das derivadas da f, mas do cálculo da própria f em diferentes pontos, ganhando assim mais eficiência enquanto tendem a manter a mesma precisão e o mesmo ETL dos métodos de Taylor. Como exemplo, no caso do método de Taylor de ordem 2, uma aproximação pode ser

$$f(t, y) + \frac{h}{2!} f'(t, y) \approx b_1 f(t, y) + b_2 f(t + \alpha h \ , \ y + \beta h f(t, y)),$$

gerando um RK de 2 estágios, onde $b_1 = b_2 = 1/2$ e $\alpha = \beta = 1$ seriam alguns dos possíveis valores para os parâmetros a fim de mantermos o ETL em $O(h^2)$.

Os métodos de RK constituem uma classe de métodos. Primeiramente, eles são divididos em estágios, sendo que um método de RK de s estágios tem a função iterativa dada por

$$\phi(t, y, h) = \phi_{RK^{(s)}}(t, y, h) = \sum_{j=1}^{s} b_j k_{i,j}, \tag{3.14}$$

ou seja, o processo parte de $w^{(0)} = \alpha$ e calcula as aproximações para a solução $y(t_{i+1})$ por

$$w^{(i+1)} = w^{(i)} + h \sum_{j=1}^{s} b_j k_{i,j}, \quad i = 0, 1, \ldots, N-1, \tag{3.15}$$

onde os valores de $k_{i,1}, \ldots, k_{i,s}$ são determinados por

$$k_{i,j} = f\left(t_i + h c_j, \, w^{(i)} + h \sum_{n=1}^{s} a_{jn} k_{i,n}\right), \quad j = 1, 2, \ldots, s, \tag{3.16}$$

com h, N e t_i definidos pela malha dada em (3.2).

Basicamente, os $k_{i,j}$ são as avaliações da f em diferentes pontos dentro do intervalo $[t_i, t_{i+1}]$, em substituição ao cálculo das derivadas de alta ordem da f no método de Taylor. Já os coeficientes a_{jn}, b_j e c_j são constantes que podem ser determinadas aproximando o método RK de s estágios com o Taylor de ordem p (em geral, $p = s$), ou seja,

$$\phi_{RK^{(s)}}(t, y, h) \approx T^{(p)}(t, y, h).$$

A dedução para essa aproximação é bastante extensa e requer ferramentas matemáticas mais sofisticadas. Alguns casos específicos podem ser encontrados em livros de Análise Numérica, como Atkinson, Han e Stewart (2009) e Burden e Faires (2008). Já a generalização é feita em Butcher (1985).

Para os métodos de RK mais utilizados, os coeficientes estão disponíveis em tabelas, chamadas *tableau* de Butcher (BUTCHER, 1985). Na forma matricial, o *tableau* de Butcher é assim representado

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b} \end{array}$$

onde $\mathbf{b} = (b_1, ..., b_s)$, $\mathbf{c} = (c_1, c_2, ..., c_s)$, com $c_i \in [0, 1]$, e $\mathbf{A} = (a_{ij})$, $i, j = 1, ..., s$, satisfazendo $\sum_{j=1}^{s} a_{ij} = c_i$, para cada $i = 1, \ldots, s$.

Dentre os métodos de RK, o mais conhecido e utilizado é o de 4ª Ordem (RK4), que é deduzido a partir da aproximação do método de Taylor de ordem 4 pelo RK de 4 estágios. Para compreender esse método, fazemos $s = 4$ em (3.15)-(3.16), obtendo

$$w^{(i+1)} = w^{(i)} + h(b_1 k_{i,1} + b_2 k_{i,2} + b_3 k_{i,3} + b_4 k_{i,4}), \qquad i = 0, 1, \ldots, N-1, \tag{3.17}$$

onde

$$\begin{aligned} k_{i,1} &= f(t_i + hc_1\ ,\ w^{(i)} + h(a_{11}k_{i,1} + a_{12}k_{i,2} + a_{13}k_{i,3} + a_{14}k_{i,4})), \\ k_{i,2} &= f(t_i + hc_2\ ,\ w^{(i)} + h(a_{21}k_{i,1} + a_{22}k_{i,2} + a_{23}k_{i,3} + a_{24}k_{i,4})), \\ k_{i,3} &= f(t_i + hc_3\ ,\ w^{(i)} + h(a_{31}k_{i,1} + a_{32}k_{i,2} + a_{33}k_{i,3} + a_{34}k_{i,4})), \\ k_{i,4} &= f(t_i + hc_4\ ,\ w^{(i)} + h(a_{41}k_{i,1} + a_{42}k_{i,2} + a_{43}k_{i,3} + a_{44}k_{i,4})). \end{aligned} \tag{3.18}$$

Neste caso, o *tableau* de Butcher fica da seguinte forma

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & a_{13} & a_{14} \\ c_2 & a_{21} & a_{22} & a_{23} & a_{24} \\ c_3 & a_{31} & a_{32} & a_{33} & a_{34} \\ c_4 & a_{41} & a_{42} & a_{43} & a_{44} \\ \hline & b_1 & b_2 & b_3 & b_4 \end{array}$$

Em particular, quando $a_{ij} = 0$ para todo $i < j$, o método se torna explícito, uma vez que $k_{i,1}$ não depende dos demais $k_{i,j}$, ou seja, $k_{i,2}$ depende apenas do $k_{i,1}$, que já está calculado, e assim por diante. O RK4 explícito é uma combinação de coeficientes que preserva a ordem do ETL do método de Taylor em $O(h^4)$ e é dado por (BUTCHER, 1985)

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}$$

Substituindo, por fim, os valores dos coeficientes acima nas Eq. (3.17)-(3.18) e simplificando, obtemos o processo RK4

$$\begin{cases} k_{i,1} = f(t_i,\, w^{(i)}), \\ k_{i,2} = f(t_i + h/2,\, w^{(i)} + (h/2)k_{i,1}), \\ k_{i,3} = f(t_i + h/2,\, w^{(i)} + (h/2)k_{i,2}), \\ k_{i,4} = f(t_i + h,\, w^{(i)} + hk_{i,3}), \\ w^{(i+1)} = w^{(i)} + \dfrac{h}{6}(k_{i,1} + 2k_{i,2} + 2k_{i,3} + k_{i,4}), \end{cases} \tag{3.19}$$

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

O método RK4 (3.19) satisfaz as condições de consistência, convergência e estabilidade, desde que o PVI seja bem posto e que a função f possua até a derivada de ordem 4, todas contínuas no intervalo considerado, ou seja, $f \in C^4\left([t_0, t_f] \times \mathbb{R}^n\right)$. A demonstração para isso pode ser vista em Burden e Faires (2008) e Fukushima (2018).

O esforço computacional dos métodos de RK é devido principalmente à quantidade de cálculos da f por passo. Para o RK4, são 4 avaliações da f por passo, uma em cada $k_{i.j}$. No entanto, a ordem do ETL dos métodos não segue a quantidade de estágios, que determina a quantidade de avaliações da função f por passo. De fato, Butcher (1985) prova que não existe método de Runge-Kutta com 5 avaliações da f que tenha ETL $O(h^5)$, e assim por diante, e apresenta uma tabela (reproduzida na Tabela 1) com a relação entre o número mínimo de avaliações da f e o melhor ETL possível. Por esse motivo o método de RK4 é o mais conhecido e utilizando dentre a classe de métodos RK.

Tabela 1 – Relação entre o número n de avaliações da função f por passo e o melhor erro de truncamento local possível para os métodos de Runge-Kutta.

Avaliações de f por passo	2	3	4	$5 \leq n \leq 7$	$8 \leq n \leq 9$	$10 \leq n$
Ordem do ETL	$O(h^2)$	$O(h^3)$	$O(h^4)$	$O(h^{n-1})$	$O(h^{n-2})$	$O(h^{n-3})$

Exemplo 3.1. Considere o PVI formado por duas EDOs e condições iniciais (BOYCE; DIPRIMA, 2015)

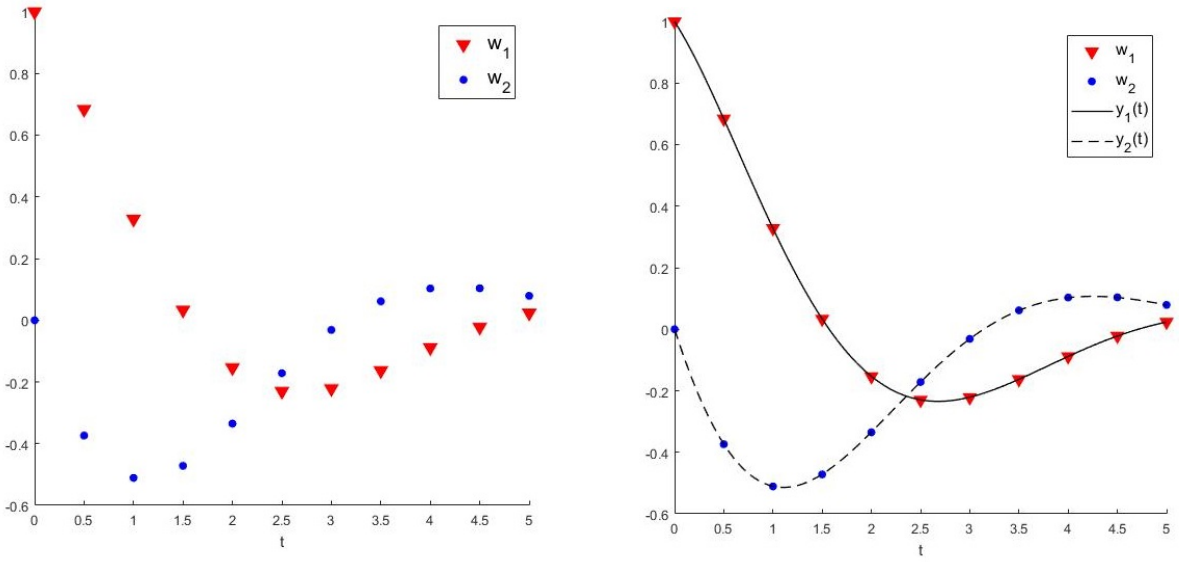
$$\begin{cases} y_1'(t) = -\dfrac{1}{2}y_1 + y_2, \\ y_2'(t) = -y_1 - \dfrac{1}{2}y_2, \\ y_1(0) = 1, \\ y_2(0) = 0. \end{cases} \tag{3.20}$$

Observe que as funções envolvidas são $f_1(t, y_1, y_2) = -y_1/2 + y_2$ e $f_2(t, y_1, y_2) = -y_1 - y_2/2$, ambas contínuas e Lipschitz na segunda variável, de forma que o PVI é bem posto. Além disso, ambas são de classe C^4 e, portanto, o Teorema 3.1 assegura a consistência, a convergência e a ordem global do erro em $O(h^4)$.

Considerando $t \in [0, 5]$ e $N = 10$, temos que $h = 0,5$. A Fig. 5 (a) mostra as soluções numéricas encontradas. Em particular, neste exemplo, é possível obter a solução de forma analítica, que é dada por

$$\begin{cases} y_1(t) = e^{-t/2}\cos(t), \\ y_2(t) = -e^{-t/2}\,\mathrm{sen}(t). \end{cases} \tag{3.21}$$

Assim, a Fig. 5 (b) mostra as soluções numérica e exata sobrepostas, de onde observa-se que ambas estão visivelmente bastante próximas.



(a) (b)

Figura 5 – Gráfico da solução do PVI (3.20), sendo (a) solução numérica pelo método de RK4 e (b) solução numérica (RK4) e exata sobrepostas.

Já a Tabela 2 quantifica os erros cometidos pelo método numérico. Em particular, para a solução y_1, o maior erro ocorreu em $w_1^{(5)} \approx y(2.5)$, ou seja, $|y_1(2.5) - w_1^{(5)}| = 0.00076$. Para a solução y_2, o maior erro ocorreu em $w_2^{(3)} \approx y_2(1.5)$, ou seja, $|y_2(1.5) - w_2^{(3)}| = 0.00077$. Em ambos os casos, os erros estão na terceira casa decimal, o que está de acordo com a teoria, pois no exemplo, a precisão é $O(0.25^4) = O(0.0039062)$, ou seja, na terceira casa decimal.

É importante notar que o método RK4 obtém aproximações da solução apenas em valores discretos, ou seja, nos pontos da malha. Por esse motivo, neste trabalho ele é também chamado de RK discreto, em contraste com o RKC que gera uma aproximação contínua para a solução e será abordado no Capítulo 4. Para gerar uma aproximação contínua, a técnica mais utilizada é a interpolação polinomial da solução numérica. A Seção 3.3 trata desse assunto.

Tabela 2 – Valores das soluções analíticas, y_1 e y_2, nos pontos t_i, soluções aproximadas,w_1 e w_2, e o erro entre as soluções para o problema (3.20)

t	y_1	w_1	$\lvert y_1 - w_1\rvert$	y_2	w_2	$\lvert y_2 - w_2\rvert$
0	1	1	0	0	0	0
0,5	0,68346	0,68376	0,00029	-0,37338	-0,37370	0,00032
1,0	0,32771	0,32787	0,00016	-0,51038	-0,51104	0,00066
1,5	0,03341	0,03321	0,00020	-0,47118	-0,47195	0,00077
2,0	-0,15309	-0,15366	0,00057	-0,33451	-0,33511	0,00059
2,5	-0,22953	-0,23029	0,00076	-0,17147	-0,17171	0,00025
3,0	-0,22090	-0,22163	0,00074	-0,03149	-0,03135	0,00014
3,5	-0,16273	-0,16326	0,00053	0,06096	0,06139	0,00043
4,0	-0,08846	-0,08869	0,00023	0,10242	0,10298	0,00056
4,5	-0,02222	-0,02216	0,00006	0,10303	0,10356	0,00053
5,0	0,02328	0,02355	0,00027	0,07871	0,07909	0,00038

3.3 Interpolação polinomial para solução numérica de EDOs

Conforme vimos na seção anterior, a solução numérica do PVI (3.20), gerada pelo RK4, consiste de um conjunto de pontos discretos $(t_i, w^{(i)})$, $i = 0, 1, \ldots, N$, em que as abscissas t_i são os pontos da malha e as ordenadas $w^{(i)}$ aproximam a solução exata $y(t_i)$. É comum, no entanto, que se queira obter uma aproximação em algum ponto que não pertença à malha ou, ainda, que se queira uma aproximação para solução que seja contínua em todo o intervalo estudado. Nestes casos, pode-se utilizar uma interpolação polinomial para aproximar pontos discretos por um polinômio. Nesta seção, nos concentramos na interpolação linear e na interpolação cúbica de Hermite, por serem técnicas adequadas à solução numérica de EDOs.

Deve-se destacar que ao longo do texto estamos tratando de sistemas de EDOs e, portanto, a solução numérica $(t_i, w^{(i)})$ é de fato $(t_i, w_1^{(i)}, w_2^{(i)}, \ldots, w_n^{(i)})$. Do ponto de vista da interpolação polinomial, a aproximação é realizada para cada coordenada w_j de w separadamente. Portanto, para aproximar a solução numérica w_j, tomamos os nós de interpolação $(t_i, w_j^{(i)})$, $i = 0, 1, \ldots, N$.

Na interpolação linear, precisamos de apenas dois nós de interpolação, ou seja, dois pontos discretos $(t_k, w_j^{(k)})$ e $(t_{k+1}, w_j^{(k+1)})$, e o polinômio obtido é de grau menor ou igual a um. A condição primordial da interpolação é que o polinômio interpolador $P_1(t)$ coincida com os valores de $w_j^{(k)}$ e $w_j^{(k+1)}$ quando calculado em t_k e t_{k+1}, ou seja,

$$P_1(t_k) = w_j^{(k)} \quad \text{e} \quad P_1(t_{k+1}) = w_j^{(k+1)}. \tag{3.22}$$

Impondo as condições (3.22) e chamando $t_{k+1} - t_k = h$, chegamos à seguinte

expressão para o polinômio interpolador linear

$$P_1(t) = w_j^{(k)} + \frac{\left(w_j^{(k+1)} - w_j^{(k)}\right)}{h}(t - t_k), \quad t \in (t_k, t_{k+1}). \tag{3.23}$$

Para obter uma função contínua em todo o intervalo $[t_0, t_f]$ que interpola os $N+1$ pontos discretos da solução numérica $(t_i, w_j^{(i)})$ sobre a malha Δ (ver Seção 3.1), podemos calcular um polinômio do tipo (3.23) para cada intervalo $[t_k, t_{k+1}]$, $k = 0, \ldots, N-1$, ou seja, fazer uma interpolação por partes (*splines*). Geometricamente, a interpolação linear por partes corresponde a obter segmentos de retas unindo os pontos discretos das soluções numéricas, conforme mostrado na Fig. 6.

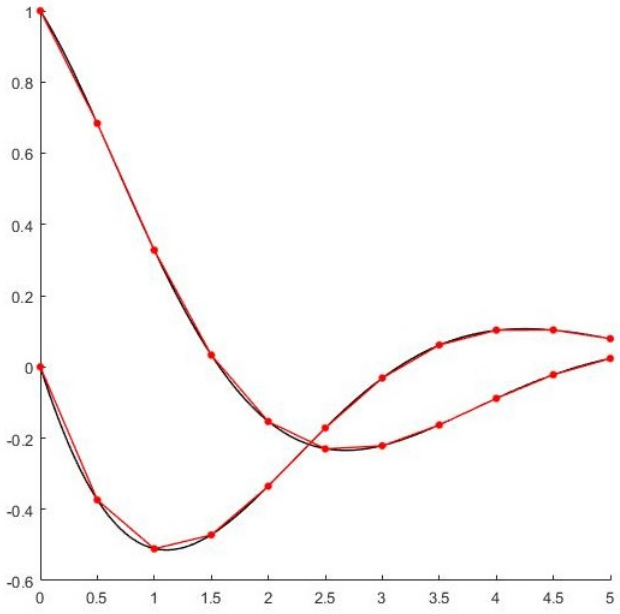


Figura 6 – Interpolação linear por partes da solução numérica do PVI (3.20).

Uma desvantagem na escolha da interpolação linear é que a ordem do erro de truncamento é somente $O(h)$ (BURDEN; FAIRES, 2008). Assim, para um método de precisão mais alta, como o RK4, os pontos da malha terão precisão de $O(h^4)$ enquanto que os pontos fora da malha terão precisão de $O(h)$. Também observamos pela Fig. 6 a não diferenciabilidade nas extremidades dos subintervalos, o que em um contexto geométrico significa que a função interpoladora não é suave. No caso de equações diferenciais, a solução é diferenciável e é desejável que a aproximação também seja.

Uma técnica bastante adequada para equações diferenciais de primeira ordem é a interpolação cúbica de Hermite que, com apenas dois nós, apresenta precisão de $O(h^3)$. Em contrapartida, esta técnica necessita de informação sobre a derivada de $w^{(k)}$. Essa informação pode ser retirada da própria EDO, uma vez que $y'(t_k) = f(t_k, y(t_k))$ e podemos chamar $(w^{(k)})' = f(t_k, w^{(k)})$, ou mais especificamente, para a coordenada j, $(w_j^{(k)})' = f_j(t_k, w_1^{(k)}, \cdots, w_m^{(k)})$.

As condições que se impõem são de que o polinômio interpolador $H_3(t)$ coincida com a solução numérica nos nós de interpolação e que, além disso, sua derivada coincida com a derivada da solução nesses nós. Com isso, impomos quatro condições

$$\begin{array}{rclcrcl} H_3(t_k) & = & w_j^{(k)}, & & H_3'(t_k) & = & f_j(t_k, w_1^{(k)}, \ldots, w_m^{(k)}), \\ H_3(t_{k+1}) & = & w_j^{(k+1)}, & & H_3'(t_{k+1}) & = & f_j(t_{k+1}, w_1^{(k+1)}, \cdots, w_m^{(k+1)}). \end{array} \tag{3.24}$$

O polinômio interpolador cúbico de Hermite para a j-ésima componente da w e para $t \in (t_k, t_{k+1})$ é escrito como

$$H_3(t) = H_3(t_k)\, h_k(t) + H_3(t_{k+1})\, h_{k+1}(t) + H_3'(t_k)\, \hat{h}_k(t) + H_3'(t_{k+1})\, \hat{h}_{k+1}(t), \tag{3.25}$$

em que

$$\begin{array}{rclcrcl} h_k(t) & = & \left(1 + 2\dfrac{t - t_k}{h}\right)\left(\dfrac{t_{k+1} - t}{h}\right)^2, & & \hat{h}_k(t) & = & (t - t_k)\left(\dfrac{t_{k+1} - t}{h}\right)^2, \\ h_{k+1}(t) & = & \left(1 - 2\dfrac{t - t_{k+1}}{h}\right)\left(\dfrac{t - t_k}{h}\right)^2, & & \hat{h}_{k+1}(t) & = & (t - t_{k+1})\left(\dfrac{t - t_k}{h}\right)^2. \end{array} \tag{3.26}$$

Para mais detalhes sobre dedução dos polinômios interpoladores de Hermite e ordem do erro de truncamento, veja Burden e Faires (2008).

De forma análoga à realizada para interpolação linear, também é possível calcular as interpolações cúbicas de Hermite por partes, sobre os intervalos $[t_0, t_1]$, $[t_1, t_2]$, ..., $[t_{N-1}, t_N]$, com a diferença de que, no segundo caso, o resultado é uma função com derivada contínua em todo o intervalo $[t_0, t_f]$.

Finalizamos o capítulo retomando o PVI (3.20). Tomando $t_a = 1,25$ (ponto escolhido porque, visualmente pela Fig. 6, apresenta certa discrepância entre a solução exata e a aproximada), observamos que $t_a \in (1,0; 1,5)$ e retiramos as seguintes informações da Tabela 2 e do cálculo das funções f do próprio PVI (3.20).

Tabela 3 – Valores das soluções aproximadas pelo RK4, w_1 e w_2, do PVI (3.20) e das funções, f_1 e f_2, calculadas com essas soluções para $t = 1,0$ e $t = 1,5$.

t	w_1	$f_1(t, w_1, w_2)$	w_2	$f_2(t, w_1, w_2)$
1,0	0,32787	-0,67497	-0,51104	-0,07235
1,5	0,03321	-0,48856	-0,47195	0,202763

Para a primeira componente da solução, w_1, substituímos os valores em (3.26) e

calculamos

$$h_k(1,25) = \left(1 + 2\frac{1,25 - 1,0}{0,5}\right)\left(\frac{1,5 - 1,25}{0,5}\right)^2 = 0,5$$

$$h_{k+1}(1,25) = \left(1 - 2\frac{1,25 - 1,5}{0,5}\right)\left(\frac{1,25 - 1,0}{0,5}\right)^2 = 0,5$$

$$\hat{h}_k(1,25) = (1,25 - 1,0)\left(\frac{1,5 - 1,25}{0,5}\right)^2 = 0,0625$$

$$\hat{h}_{k+1}(1,25) = (1,25 - 1,5)\left(\frac{1,25 - 1,0}{0,5}\right)^2 = -0,0625$$

Em seguida, substituímos os valores acima em (3.25), para finalmente obtermos a aproximação para $w_1(1,25)$ por

$$\begin{aligned} H_3(1,25) &= w_1(1,0)h_k(1,25) + w_1(1,5)h_{k+1}(1,25) \\ &\quad + f_1(1,0;w_1;w_2)\hat{h}_k(1,25) + f_1(1,5;w_1;w_2)\hat{h}_{k+1}(1,25) \\ &= 0,32787 \times 0,5 + (0,03321) \times 0,5 \\ &\quad + (-0,67497) \times 0,0625 + (-0,48856) \times (-0,0625) \\ &= 0,16889. \end{aligned}$$

Portando, aplicando a interpolação cúbica de Hermite, obtemos $w_1(1,25) = 0,16889$ que, comparando com a solução exata, Eq. (3.21), $y_1(1,25) = 0,16878$, nos fornece um erro absoluto de aproximadamente 1×10^{-4}.

Para a segunda componente, w_2, utilizamos os mesmos valores de $h_k(1,25)$, $h_{k+1}(1,25)$, $\hat{h}_k(1,25)$ e $\hat{h}_{k+1}(1,25)$ e obtemos a aproximação para $w_2(1,25)$ por

$$\begin{aligned} H_3(1,25) &= w_2(1,0)h_k(1,25) + w_2(1,5)h_{k+1}(1,25) \\ &\quad + f_2(1,0;w_1;w_2)\hat{h}_k(1,25) + f_2(1,5;w_1;w_2)\hat{h}_{k+1}(1,25) \\ &= (-0,51104) \times 0,5 + (-0,47195) \times 0,5 \\ &\quad + (-0,07235) \times 0,0625 + 0,202763 \times (-0,0625) \\ &= -0,50869. \end{aligned}$$

Assim, temos $w_2(1,25) = -0,50869$ que, comparado com a solução exata $y_2(1,25) = -0,50795$, fornece um erro absoluto de aproximadamente 7×10^{-4}.

O capítulo seguinte trata do RKC, um método que embute um processo de interpolação ao RK, não sendo necessário obter as soluções discretas para depois fazer a interpolação. Esse processo único é particularmente interessante quando resolvemos EDRs, pois quando o argumento do retardo não cai sobre um ponto da malha, ele é automaticamente calculado pela interpolação.

4

Runge-Kutta Contínuo e resultados numéricos

Por volta do final da década de 1980 e início da década de 1990 houve crescente interesse no estudo de métodos numéricos para a solução de PVIs do tipo (3.1) que gerassem soluções contínuas ao invés de discretas. Esse estudo decorreu da necessidade, por exemplo, de obter uma saída densa de dados, plotar a solução ou encontrar as raízes da solução (HARTUNG et al., 2006). No fim dos anos 1990, os métodos contínuos começaram a ser aplicados na solução numérica de EDRs, o que se mostrou bastante apropriado, pois, como já mencionado, o retardo presente na equação torna necessário avaliar a solução em momentos anteriores ao atual, que podem estar entre pontos da malha.

Neste capítulo apresentamos um método de Runge-Kutta Contínuo de quarta ordem (RKC4), composto pelo RK4 e pela interpolação polinomial cúbica de Hermite. Primeiramente, o RKC4 é tratado no âmbito das EDOs e, em seguida, das EDRs. Seguindo a linha do capítulo anterior, o texto traz resultados teóricos e exemplos. Por fim, descrevemos com detalhes o algoritmo desenvolvido neste projeto e implementado em linguagem MATLAB para, enfim, retomarmos os exemplos da Seção 2.3, agora resolvidos numericamente.

4.1 Método de Runge-Kutta Contínuo para EDOs

Nesta seção, estamos interessados na obtenção de uma aproximação contínua para o PVI (3.1). Uma alternativa seria proceder como no Capítulo 3, ou seja, aplicar um processo discreto e em seguida realizar interpolação por partes. No entanto, o objetivo dos métodos contínuos é unir as duas técnicas em um único processo, o que, além de agilizar, permite a aplicação em outros tipos de equações, como as com retardo.

A sistematização desse processo se dá por meio das extensões contínuas. Segundo Bellen e Zennaro (2003), uma extensão contínua, ou interpolante, de um método numérico discreto é uma função polinomial por partes $u(t)$ obtida a cada intervalo $[t_i, t_{i+1}]$ da malha, de forma que as condições de continuidade $u(t_i) = w^{(i)}$ e $u(t_{i+1}) = w^{(i+1)}$ sejam satisfeitas. Desta forma, a solução numérica de um método contínuo é a união das extensões contínuas em cada intervalo $[t_i, t_{i+1}]$, isto é, é uma função $u(t)$ contínua em $[t_0, t_f]$ que aproxima $y(t)$ nesse intervalo.

Para aplicar o RKC ao PVI (3.1), iniciamos construindo uma malha de pontos discretos. Aqui vamos considerar a mesma usada no capítulo anterior, a malha estabelecida em (3.2). Em seguida, atribuímos a condição inicial para início da solução numérica, isto é,

$$u(t_0) = w^{(0)} = y(t_0) = \alpha,$$

e utilizamos o processo abaixo para construir a solução por partes

$$u(t_i + \theta h) = w^{(i)} + h \sum_{j=1}^{s} \tilde{b}_j(\theta) k_{i,j}, \qquad \theta \in [0,1], \tag{4.1}$$

$$k_{i,j} = f\left(t_i + hc_j \ , \ w^{(i)} + h \sum_{n=1}^{s} a_{jn} k_{i,n}\right), \quad j = 1, 2, \ldots, s, \tag{4.2}$$

para $i = 0, 1, \ldots, N-1$. A variável θ faz com que o intervalo $[t_i, t_{i+1}]$ seja percorrido continuamente de tal forma que $u(t_i + \theta h)$ seja uma função contínua nesse intervalo (extensão contínua).

Comparando o processo do RCK, dado pelas Eqs. (4.1)-(4.2), com o do RK, dado pelas Eqs. (3.15)-(3.16), vemos que os coeficientes a_{jn} e c_j são os mesmos, enquanto que os b_j foram substituídos pelas funções $\tilde{b}_j(\theta)$, com $\theta \in [0,1]$. As funções $\tilde{b}_j(\theta)$, por sua vez, exercem papel fundamental no RKC. Por meio delas que se constrói a interpolação polinomial impondo as seguintes condições de continuidade

$$\tilde{b}_j(0) = 0, \qquad \tilde{b}_j(1) = b_j, \qquad j = 1, \ldots, s.$$

Quando $\theta = 0$, $\tilde{b}_j(0) = 0$ e a Eq. (4.1) mostra que $u(t_i) = w^{(i)}$ e quando $\theta = 1$, $\tilde{b}_j(1) = b_j$ e a Eq. (4.1) coincide com a Eq. (3.15), ou seja, $u(t_i + h) = w^{(i+1)}$.

O grau δ dos polinômios $\tilde{b}_j(\theta)$ é escolhido, em geral, de forma a manter a ordem do erro do método discreto. Veremos a seguir que a escolha da interpolação polinomial de Hermite, cujo grau é $\delta = 3$ e o erro de truncamento é $O(h^4)$, faz com que o RKC seja $O(h^4)$, que chamaremos de RKC4, de onde segue a escolha dos métodos utilizados neste trabalho.

Apenas como observação, analogamente aos métodos discretos, a teoria dos métodos contínuos é construída sobre a função iteradora ϕ que, para o RKC, é

$$\begin{aligned} &\phi(t, y, h) = \phi_{RKC^{(s)}}(t, y, h) = \textstyle\sum_{j=1}^{s} \tilde{b}_j(\theta) k_{i,j}, \\ &k_{i,j} = f\left(t_i + hc_j \ , \ y + h \textstyle\sum_{n=1}^{s} a_{jn} k_{i,n}\right), \quad j = 1, 2, \ldots, s \ . \end{aligned} \tag{4.3}$$

Semelhante ao conceito de ordem do ETL visto na Definição 3.1, para as extensões contínuas temos o conceito de ordem uniforme local (HARTUNG et al., 2006), como podemos ver abaixo.

Definição 4.1. Seja z a solução do problema local

$$\begin{cases} z'(t) = f(t, z(t)), \quad t \in [t_i, t_{i+1}], \\ z(t_i) = u(t_i). \end{cases}$$

Dizemos que o método de RKC (4.1)-(4.2) tem ordem uniforme local q se $q \geq 1$ for o maior inteiro tal que

$$\max_{t_i \leq t \leq t_{i+1}} ||z(t) - u(t)|| = O(h^{q+1}).$$

Da Definição 4.1 e do Teorema (3.1) item (iii), sobre um método discreto ter ordem global p, segue o importante resultado que determina a ordem uniforme global do método RKC (HARTUNG et al., 2006).

Teorema 4.1. Se o método RK discreto, Eqs. (3.15)-(3.16), tiver ordem global p e o método RKC associado, (4.1)-(4.2), tiver ordem uniforme local q, então esse método possui ordem uniforme global $q' = \min\{p, q+1\}$, ou seja,

$$\max_{t_0 \leq t \leq t_N} ||y(t) - u(t)|| = O(h^{q'}),$$

em que y é a solução exata e u é a solução obtida pelo RKC.

Dessa forma, conclui-se que um par ótimo compreende um método discreto de ordem global p e um método de interpolação também de ordem p, o qual gera um método contínuo de ordem uniforme local $q = p - 1$. Dessa conclusão, segue que para o método RK, que é $O(h^4)$, podemos utilizar polinômios interpoladores cúbicos de Hermite, que possuem erro de truncamento $O(h^4)$, de forma que o método contínuo resultante, RKC4 será $O(h^4)$.

Os coeficientes do RKC também podem ser dispostos no *tableau* de Butcher, que agora passa a ter o formato

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \tilde{\mathbf{b}}(\theta) \end{array}$$

onde $\tilde{\mathbf{b}}(\theta) = (\tilde{b}_1(\theta) \ldots, \tilde{b}_s(\theta))$, $\mathbf{c} = (c_1, c_2, ..., c_s)$ e $\mathbf{A} = (a_{ij})$, $i, j = 1, ..., s$.

Em particular, para o RKC4, o *tableau* de Butcher fica

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \tilde{b}_1(\theta) & \tilde{b}_2(\theta) & \tilde{b}_3(\theta) & \tilde{b}_4(\theta) \end{array}$$

com

$$\begin{aligned} \tilde{b}_1(\theta) &= \frac{2}{3}\theta^3 - \frac{3}{2}\theta^2 + \theta, \qquad & \tilde{b}_2(\theta) &= -\frac{2}{3}\theta^3 + \theta^2, \\ \tilde{b}_3(\theta) &= -\frac{2}{3}\theta^3 + \theta^2, & \tilde{b}_4(\theta) &= \frac{2}{3}\theta^3 - \frac{1}{2}\theta^2. \end{aligned} \tag{4.4}$$

Tomando $s = 4$ em (4.1)-(4.2), substituindo os coeficientes a_{jn} e c_j conforme o *tableau* de Butcher acima e fazendo as simplificações necessárias, obtemos o processo do método RKC4

$$\begin{cases} k_{i,1} = f(t_i \ , \ w^{(i)}), \\ k_{i,2} = f(t_i + h/2 \ , \ w^{(i)} + (h/2)k_{i,1}), \\ k_{i,3} = f(t_i + h/2 \ , \ w^{(i)} + (h/2)k_{i,2}), \\ k_{i,4} = f(t_i + h \ , \ w^{(i)} + hk_{i,3}), \\ u(t_i + \theta h) = w^{(i)} + h[\tilde{b}_1(\theta)k_{i,1} + \tilde{b}_2(\theta)k_{i,2} + \tilde{b}_3(\theta)k_{i,3} + \tilde{b}_4(\theta)k_{i,4}], \end{cases} \tag{4.5}$$

com $\theta \in [0, 1]$, para cada $i = 0, 1, \ldots, N - 1$.

Apresentamos a seguir um exemplo de resolução de EDO com o método RKC4.

Exemplo 4.1. Considere o PVI

$$\begin{cases} y' = y - t^2 + 1, \quad t \in [0, 3], \\ y(0) = 1/2. \end{cases} \tag{4.6}$$

Adotando o passo $h = 1$ e observando que $y(0) = w^{(0)} = 1/2$, resolvemos o PVI primeiramente para $t \in [0, 1]$ tomando $i = 0$. Calculamos $k_{0,n}$, $j = 1, 2, 3, 4$, conforme Eq. (4.5), que nos dá

$$k_{0,1} = f\left(0, \frac{1}{2}\right) = \frac{2}{3},$$

$$k_{0,2} = f\left(0 + \frac{1}{2}, \frac{1}{2} + \frac{1}{2} \times \frac{3}{2}\right) = 2,$$

$$k_{0,3} = f\left(0 + \frac{1}{2}, \frac{1}{2} + \frac{1}{2} \times 2\right) = \frac{9}{4},$$

$$k_{0,4} = f\left(0 + 1, \frac{1}{2} + \frac{9}{4}\right) = \frac{11}{4}.$$

Em seguida, efetuamos os produtos $\tilde{b}_j(\theta)k_{0,n}$, sendo $\tilde{b}_j(\theta)$ dados em (4.4),

$$\tilde{b}_1(\theta)k_{0,1} = \theta^3 - \frac{9}{4}\theta^2 + \frac{3}{2}\theta,$$

$$\tilde{b}_2(\theta)k_{0,2} = -\frac{4}{3}\theta^3 + 2\theta^2,$$

$$\tilde{b}_3(\theta)k_{0,3} = -\frac{3}{2}\theta^3 + \frac{9}{4}\theta^2,$$

$$\tilde{b}_4(\theta)k_{0,4} = \frac{11}{6}\theta^3 - \frac{11}{8}\theta^2,$$

e obtemos a solução no primeiro intervalo

$$u(t_0 + \theta h) = \frac{1}{2} + \left(\frac{5}{8}\theta^2 + \frac{3}{2}\theta\right), \quad \theta \in [0, 1].$$

Como a solução está expressa na variável θ e precisamos que ela e esteja em t, fazemos uma mudança de variável. Observando que $u(t) = u(t_i + \theta h)$, temos $\theta = (t - t_i)/h$. Assim, para o primeiro intervalo, $t_i = t_0 = 0$ e, portanto, $\theta = t$. Logo, a função contínua que aproxima a solução do PVI (4.6) para $t \in [0, 1]$ é

$$u(t) = \frac{1}{2} + \frac{5}{8}t^2 + \frac{3}{2}t, \quad t \in [0, 1]. \tag{4.7}$$

Repetimos o processo para o segundo intervalo $t \in [1, 2]$. Primeiramente calculamos a solução (4.7) em $t = 1$, de onde obtemos $u(1) = w^{(1)} = 21/8$. Calculamos os $k_{1,n}$, os produtos $\tilde{b}_j(\theta)k_{1,n}$ e obtemos

$$u(t_1 + \theta h) = \frac{21}{8} + \left(-\frac{7}{24}\theta^3 + \frac{19}{64}\theta^2 + \frac{21}{8}\theta\right), \quad \theta \in [0, 1].$$

As variáveis t e θ estão relacionadas agora por $\theta = (t - t_i)/h = t - 1$, e assim, fazendo a mudança de variáveis, a função contínua que aproxima a solução do PVI (4.6) para $t \in [1, 2]$ é

$$u(t) = \frac{21}{8} + \left[-\frac{7}{24}(t-1)^3 + \frac{19}{64}(t-1)^2 + \frac{21}{8}(t-1)\right], \quad t \in [1, 2]. \tag{4.8}$$

Finalmente, para o último intervalo $t \in [2, 3]$, calculamos a solução (4.8) em $t = 2$, obtendo $u(2) = w^{(2)} = 1009/192$. De forma análoga, calculando os $k_{2,n}$, os produtos $\tilde{b}_j(\theta)k_{2,n}$, e a $u(t_2 + \theta h)$, de onde obtemos

$$u(t_2 + \theta h) = \frac{1009}{192} + \left(-\frac{2492}{2304}\theta^3 - \frac{909}{1536}\theta^2 + \frac{433}{192}\theta\right), \quad \theta \in [0, 1].$$

As variáveis t e θ estão relacionadas agora por $\theta = (t - t_i)/h = t - 2$, e assim, fazendo a mudança de variáveis, a função contínua que aproxima a solução do PVI (4.6) para $t \in [2, 3]$ é

$$u(t) = \frac{1009}{192} + \left[-\frac{2492}{2304}(t-2)^3 - \frac{909}{1536}(t-2)^2 + \frac{433}{192}(t-2)\right], \quad t \in [2,3]. \tag{4.9}$$

A aproximação contínua de (4.6) é, portanto, formada pelas partes (4.7), (4.8) e (4.9), ou seja,

$$u(t) = \begin{cases} \dfrac{1}{2} + \dfrac{5}{8}t^2 + \dfrac{3}{2}t, & t \in [0,1], \\ \dfrac{21}{8} + \left[-\dfrac{7}{24}(t-1)^3 + \dfrac{19}{64}(t-1)^2 + \dfrac{21}{8}(t-1)\right], & t \in [1,2], \\ \dfrac{1009}{192} + \left[-\dfrac{2492}{2304}(t-2)^3 - \dfrac{909}{1536}(t-2)^2 + \dfrac{433}{192}(t-2)\right], & t \in [2,3]. \end{cases}$$

4.2 Método de Runge-Kutta Contínuo para EDRs

Nesta seção estudamos a aplicação do método RKC4 na solução numérica do Problema (2.12), o qual repetimos aqui

$$\begin{cases} y'(t) = f(t, y(t), y(t - \sigma(t, y(t)))), & t \in [t_0, t_f], \\ y(t) = \varphi(t), & t \in [t_0 - p, t_0]. \end{cases} \tag{4.10}$$

Ao tentarmos adaptar um método numérico discreto para EDOs, como o RK4, para o PVI (4.10), percebemos que para obter $w^{(i+1)}$, o processo requer conhecimento da solução aproximada no argumento de retardo $t_d = t_i - \sigma(t_i, w^{(i)})$. Essa aproximação possivelmente não está disponível, uma vez que em geral t_d não pertence à malha. Para ilustrar essa situação, observe a Fig. 7. A parte superior mostra que no caso das EDOs, a aproximação no ponto anterior sempre está disponível. Já no centro, vemos que no caso de uma EDR com retardo constante é possível construir uma malha de tal forma que $t_d = t_k$, onde $t_k \in \Delta$. Já a parte inferior mostra que quando os retardos de uma EDR são dependentes do tempo ou do estado, t_d pode cair em qualquer lugar anterior à t_i, inclusive no intervalo da condição inicial, isto é $t_d \in [t_0 - p, t_i]$.

Assim, as interpolações polinomiais são necessárias para resolver EDRs numericamente. É importante observar, no entanto, que para isso, as interpolações polinomiais não

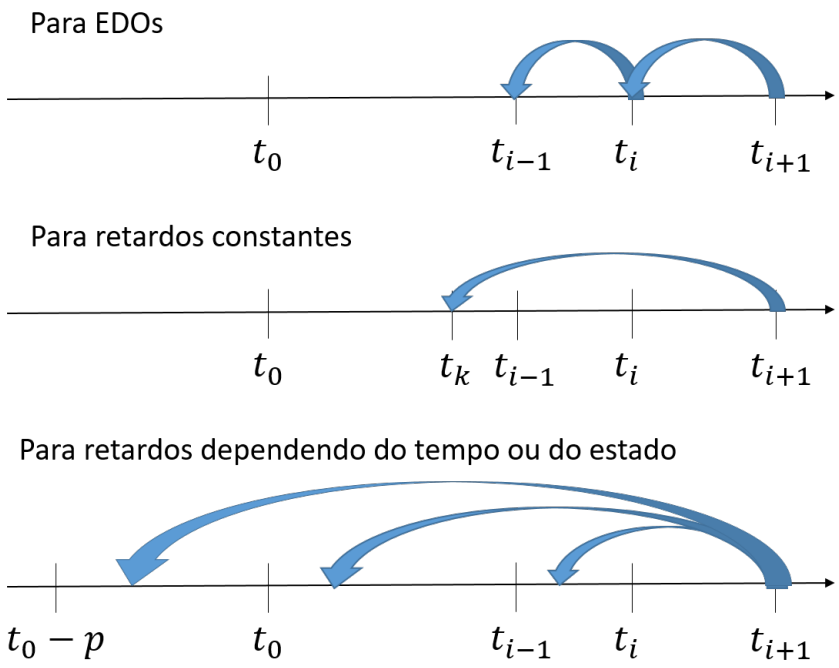


Figura 7 – Ilustração mostrando que nas EDRs dependendo do tempo e do estado, o argumento de retardo pode cair em qualquer lugar do intervalo $[t_0 - p, t_i]$ e, em geral, não caem sobre os pontos da malha.

podem ser feitas após os cálculos de todos os $w^{(i)}$ (como visto na Seção 3.3). Com isso, a melhor opção é de fato a utilização de um método contínuo (como visto na Seção 4.1).

Uma abordagem para resolver o PVI (4.10) combina o método dos passos (descrito na Seção 2.2) e um método numérico contínuo para EDOs, por exemplo o RKC4. Dessa forma, para cada PVI local

$$\begin{cases} [y^{(j)}]'(t) = f(t, y^{(j)}(t), y(t - \sigma(t, y^{(j)}(t)))), \quad t \in [\xi_j, \xi_{j+1}], \\ y^{(j)}(\xi_j) = y(\xi_j), \end{cases} \tag{4.11}$$

aplica-se o método numérico, adaptando a malha de forma a dividir o intervalo $[\xi_j, \xi_{j+1}]$ em subintervalos de tamanho $h_j = (\xi_{j+1} - \xi_j)/N_j$, onde N_j é a quantidade de pontos discretos determinada para o atual intervalo.

Essa abordagem também está relacionada ao resultado de ordem uniforme do RKC quando aplicado ao PVI (4.10). Neste resultado, para o método RK ter ordem global p, uma condição suficiente é que a função f seja de classe $C^p[t_0, t_f]$. Essa condição é agora expandida para f, σ e φ serem funções de classe C^p em seus respectivos domínios.

Por outro lado, vimos na Seção 2.2 que para as EDRs, a descontinuidade da derivada da solução em t_0 pode propagar descontinuidades do tipo salto nas derivadas de ordem superior nos pontos ξ_j (BELLEN; ZENNARO, 2003). Mais precisamente, uma descontinuidade de ordem k do tipo salto em um ponto ξ significa que $y^{(k-1)}$ existe e é contínua em ξ e $y^{(k)}$ tem descontinuidade da forma $y^{(k)}(\xi^+) \neq y^{(k)}(\xi^-)$. Nestes pontos de

descontinuidade não há como garantir que as funções f e σ sejam de classe C^p. Dessas considerações, o Teorema 4.1 é modificado para o seguinte (HARTUNG et al., 2006).

Teorema 4.2. Suponha que f, σ e φ sejam funções de classe C^p e que

(i) a malha Δ inclui todas os pontos de descontinuidade $\xi_1 < \xi_2 < \cdots < \xi_m$ da solução de ordem $\leq p$, e;

(ii) o método de RK discreto, Eqs. (3.15)-(3.16), tenha ordem global p e o método RKC associado, Eqs. (4.1)-(4.2), tenha ordem uniforme local q.

Então o método de RKC para resolver o PVI (4.10) possui ordem uniforme global $q' = \min\{p, q+1\}$, ou seja,

$$\max_{t\in[t_0-p,t_f]} \| y(t) - u(t) \| = O(q'),$$

em que y é a solução exata e u é a solução obtida pelo RKC.

O Teorema 4.2 indica que um método de alta ordem deve localizar todos os pontos de descontinuidade das soluções até ordem p e adicioná-los à malha. Para equações com retardo discreto ou dependendo do tempo essa é uma tarefa relativamente simples, pois podemos obter os ξ_j resolvendo a Eq. (2.17). Porém, para retardos dependendo do estado, a localização dos pontos de descontinuidade não podem ser calculados a priori porque eles dependem da solução. Além disso, outra dificuldade para a localização dos pontos de descontinuidade é que para os casos de retardos dependendo do tempo ou do estado, a Eq. (2.17), em geral, é não linear e conseguimos resolvê-las apenas aproximadamente.

Diante do exposto, optamos por uma abordagem em que usamos a malha Δ de pontos igualmente espaçados e com passo h pequeno, de forma que ela contenha os pontos de descontinuidade das derivadas da solução ou aproximações deles.

Para construirmos o processo iterativo do método RKC4 para EDRs, observamos que as extensões contínuas (descritas na Seção 4.1) garantem que o terceiro argumento da equação diferencial $y'(t) = f(t, y(t), y(t - \sigma(t, y(t))))$ seja conhecido e, portanto, a EDR se torna uma EDO. Esse fato se reflete no cálculo dos coeficientes $k_{i,j}$ na Eq. (4.2), que agora passam a ser

$$k_{i,j} = f\left(t_i + hc_j \ , \ w^{(i)} + h\sum_{n=1}^{s} a_{jn}k_{i,n} \ , \ u(t_i + hc_j - \sigma(t_i + hc_j, w^{(i)} + h\sum_{n=1}^{s} a_{jn}k_{i,n}))\right),$$

para $j = 1, 2, \ldots, s$.

Para facilitar a escrita, adotamos a notação $t_{i,j}$ e $w_{i,j}$ conforme definidos abaixo, e obtemos o processo do método RKC4 para EDRs

$$\begin{cases} t_{i,j} &= t_i + hc_j, \\ w_{i,j} &= w^{(i)} + h\sum_{j=1}^{4} a_{ij}k_{i,j}, \\ k_{i,j} &= f\left(t_{i,j},\, w_{i,j},\, u(t_{i,j} - \sigma(t_{i,j}, w_{i,j}))\right), \end{cases} \tag{4.12}$$

$$u(t_i + \theta h) = w^{(i)} + h\sum_{j=1}^{4} \tilde{b}_j(\theta)k_{i,j}, \quad \theta \in [0,1], \quad i = 0, 1, \ldots, N-1. \tag{4.13}$$

Na próxima seção, apresentamos a implementação do RKC4 e exemplos numéricos de sua aplicação.

4.3 Algoritmo RKC4 e implementação

Nesta seção apresentamos o algoritmo do método RKC4 desenvolvido neste projeto bem como a implementação em linguagem computacional MATLAB. Vale destacar que a compreensão do método (4.12), a construção do algoritmo e sua implementação computacional representaram a maior parte dos esforços dispendidos durante esse projeto e a motivação para isso surgiu da falta de material didático sobre o assunto. Apesar do RK discreto ser amplamente difundido e utilizado, o RKC não é.

O algoritmo foi desenvolvido não só para EDRs do tipo (4.10), em que $f : [t_0, t_f] \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$, $\sigma : [t_0, t_f] \times \mathbb{R}^n \to [0, p]$ e $\varphi : [t_0, t_f] \to \mathbb{R}^n$, mas também com múltiplos retardos, do tipo

$$\begin{cases} y'(t) = f(t, y(t), y(t - \sigma_1(t, y(t))), y(t - \sigma_2(t, y(t))), \ldots, y(t - \sigma_d(t, y(t)))),\ t \in [t_0, t_f], \\ y(t) = \varphi(t), \quad t \in [t_0 - p, t_0]. \end{cases} \tag{4.14}$$

em que $f : [t_0, t_f] \times \mathbb{R}^n \times \mathbb{R}^{nd} \to \mathbb{R}^n$, $\sigma_i : [t_0, t_f] \times \mathbb{R}^n \to [0, p]$, $i = 1, \ldots, d$, e $\varphi : [t_0 - p, t_0] \to \mathbb{R}^n$.

O algoritmo segue basicamente o processo (4.12) para EDRs do tipo (4.10), sendo generalizado para problemas com d retardos modificando-se o cálculo de $k_{i,j}$, pois f será uma função com mais variáveis, sendo que o terceiro argumento se repete trocando a função σ. Dessa forma, o programa aceita um sistema de n equações e d retardos, que podem ser de qualquer tipo (constante, dependendo do tempo ou do estado). Assim, temos um código bastante abrangente, que fornece solução numérica a uma grande variedade de problemas. Como veremos nos exemplos da Seção 4.4, os resultados tiveram sucesso,

apresentando boas performances em seus resultados computacionais quando comparado à solução analítica e também quando comparados à função nativa do MATLAB para resolver EDRs, chamada `ddesd`.

Sobre a implementação em MATLAB, criamos duas funções, uma para a entrada dos dados do problema e outra para o método propriamente dito. As funções são independentes de forma que não seja necessário fazer qualquer modificação ou configuração no código principal do método para resolver um dado problema. Além disso, o código em formato de função permite que ele seja utilizado como parte de outro programa maior, quando o método RKC4 constituir apenas uma parte de um processo.

Começamos explicando a função que armazena as informações do problema. Os dados que devem ser fornecidos estão apresentados na Tabela 4.

Tabela 4 – Variáveis referentes ao problema que devem ser entradas pelo usuário.

Parâmetro	Descrição
t_0	tempo inicial
t_f	tempo final
n	quantidade de equações
d	quantidade de retardos
f	função da equação diferencial
phi	funções história
r	funções de retardos

As quatro primeiras variáveis, t_0, t_f, n e d, são escalares. Já f, phi e r são funções simbólicas, aceitas pelo pelo MATLAB, que contém variáveis que são substituídas durante as iterações do programa. No caso da f entram como argumentos o tempo t, a solução no tempo atual y e a solução no tempo com retardo yd. Já phi depende apenas do tempo t e o retardo r, considerando o caso mais completo de um retardo dependendo do estado, depende do tempo t e da solução no tempo atual y.

As três funções, f, phi e r, são escritas como vetores, sendo f e phi vetores de n posições e r um vetor de d posições. Quanto aos seus argumentos, apenas t é um escalar, enquanto que y é definida como um vetor de n posições, e yd como uma matriz de n linhas e d colunas. Ao escrever esses dois últimos argumentos nas funções f e r, a posição certa do vetor ou matriz deve ser indicada, para coincidir corretamente com os valores sendo calculados no programa do algoritmo. Dessa forma, temos

$$y = \begin{bmatrix} y_1 & y_2 & y_3 & \cdots & y_n \end{bmatrix}, \tag{4.15}$$

$$yd = \begin{bmatrix} y_1(t-\sigma_1) & y_1(t-\sigma_2) & \cdots & y_1(t-\sigma_d) \\ y_2(t-\sigma_1) & y_2(t-\sigma_2) & \cdots & y_2(t-\sigma_d) \\ \vdots & & & \\ y_n(t-\sigma_1) & y_n(t-\sigma_2) & \cdots & y_n(t-\sigma_d) \end{bmatrix}. \tag{4.16}$$

Para ilustrar como deve ser escrita a função com o problema a ser resolvido, considere o PVI (PAUL, 1994)

$$\begin{cases} y_1'(t) = -y_1(t)y_2(t-1) + y_2(t-10), & t \geq 0, \\ y_2'(t) = y_1(t)y_2(t-1) - y_2(t), & t \geq 0, \\ y_3'(t) = y_2(t) - y_2(t-10), & t \geq 0, \\ y_1(t) = 5, & t \leq 0, \\ y_2(t) = 0,1, & t \leq 0, \\ y_3(t) = 1, & t \leq 0. \end{cases} \tag{4.17}$$

Considerando $t \in [0, 40]$ e notando que o PVI é composto de 3 equações e 2 retardos, o código da função do MATLAB que o define, que chamamos `exemplo_PVI`, é apresentado no Quadro 4.1. Na Seção 4.4, onde apresentamos mais alguns exemplos de PVIs, apresentamos também os códigos das funções em MATLAB que os define.

Quadro 4.1 – Código da função do MATLAB que definie PVI (4.17).

```
function [t0,tf,f,phi,r,n,d] = exemplo_PVI

t0 = 0;  % Tempo inicial
tf = 40; % Tempo final
n = 3;   % Quantidade de equacoes diferenciais no problema
d = 2;   % Quantidade de retardos no problema

f = @(t,y,yd) [-y(1)*yd(2,1)+yd(2,2),
                y(1)*yd(2,1)-y(2),
                y(2)-yd(2,2)];
                % funcoes do lado direito da equacao diferencial
phi = @(t) [5,0.1,1]; % funcoes historia
r = @(t,y) [1,10];    % funcoes retardo

end
```

Passamos agora para o programa principal, o qual chamamos de `rkc4`, cuja estrutura geral é dividida em quatro partes:

i. inicialização de variáveis;

ii. cálculos do método;

iii. geração de dados de saída;

iv. interpolação.

Para descrevê-las, primeiro apresentamos o código em MATLAB referente a cada parte e, em seguida, explicamos como ele funciona.

A primeira parte é responsável pela inicialização do algoritmo e das variáveis que serão utilizadas ao longo de todo o programa. O código dessa primeira parte é apresentada no Quadro 4.2. Inicialmente são definidas as variáveis que serão de saída e solicitando uma variável de entrada, no caso a variável N, que é a quantidade de iterações do método. Na sequência, temos a inicialização das variáveis vindas do PVI, conforme já mostrado e explicado acima.

Quadro 4.2 – Primeira parte do algoritmo

```
function [t,w,coef] = rkc4(N)

%%%%%%%%%%%%%%%%% Chama o problema a ser resolvido %%%%%%%%%%%%%%%%%
[t0,tf,f,phi,r,n,d] = exemplo_PVI;
%%%%%%%%%% Variaveis do metodo de Runge-Kutta de 4a ordem %%%%%%%%%%

h = (tf - t0)/N;           % Define o tamanho do passo
t = t0:h:tf;               % Armazena os pontos da malha em um vetor
w = zeros(length(t),n);    % Cria vetor p/ armazenar solucao numerica
w(1,:) = phi(t0);          % Armazena o valor inicial
k = zeros(4,n);            % Cria vetor p/ armazenar os k's
coef = zeros(length(t),3,n); % Cria vetor p/ armazenar coeficientes
                              % do polinomio interpolador de Hermite

%%%%%%%%%%%%%%%%%%%%%%%% Tableau de Butcher %%%%%%%%%%%%%%%%%%%%%%%%%

a = [0,0,0,0;1/2,0,0,0;0,1/2,0,0;0,0,1,0]; % Matriz A p/ RKC4
c = [0,1/2,1/2,1];      % Vetor c p/ RKC4
b1 = [2/3, -3/2, 1];    % b1 = (2/3)*theta^3 -(3/2)*theta^2 +theta
b2 = [-2/3, 1, 0];      % b2 = -(2/3)*theta^3 +theta^2
b3 = [-2/3, 1, 0];      % b3 = -(2/3)*theta^3 +theta^2
b4 = [2/3, -1/2, 0];    % b4 = (2/3)*theta^3 -(1/2)*theta^2
```

Com a entrada desses primeiros dados, o algoritmo consegue calcular e inicializar algumas variáveis próprias que apresentamos na Tabela 5.

Como o vetor w é um dado de saída, nesse primeiro momento ele é iniciado como uma matriz de $(N + 1)$ linhas e n colunas. Desse modo, ao longo do algoritmo, são

Tabela 5 – Variáveis calculadas e inicializadas na primeira parte do algoritmo para o processo iterativo do método.

Parâmetro	Descrição
h	tamanho do passo
t	vetor com os pontos da malha
w	vetor que armazenará a solução numérica
k	etor que armazenará os valores dos k's

concatenadas no vetor as soluções numéricas calculadas, sendo cada linha para um passo da iteração e cada coluna para uma equação do PVI. Além disso, como $w^{(1)}$ coincide com a função história em t_0, já fazemos o cálculo e preenchemos a primeira linha dessa variável.

A variável k é também iniciada como uma matriz de 4 linhas e n colunas, pois, sendo o algoritmo do método RKC de 4ª Ordem, teremos 4 k's para calcular, conforme Eq. (4.5), em que cada k deve avaliar todas as equações do PVI. Essa variável não é um dado de saída e, por isso, as informações nela serão sobrescritas à cada iteração.

Nessa primeira parte também são iniciadas o conjunto de variáveis referentes ao *tableau* de Butcher para o RKC4, apresentado na Seção. 4.1, conforme apresentamos na Tabela 6.

Tabela 6 – Variáveis calculadas e inicializadas na primeira parte do algoritmo para o processo iterativo do método.

Parâmetro	Descrição
a	matriz $a_{i,j}$
c	vvetor c_j
$b1$, $b2$, $b3$ e $b4$	vetores das funções $\tilde{b}_1(\theta)$, $\tilde{b}_2(\theta)$, $\tilde{b}_3(\theta)$ e $\tilde{b}_4(\theta)$

Com isso, temos as variáveis b's como vetores que armazenam os coeficientes dos polinômios interpoladores para o método RKC4, conforme Eq. (4.4). Essas variáveis são vetores com três posições, sendo a primeira posição o coeficiente de θ^3, a segunda posição o coeficiente de θ^2 e a terceira posição o coeficiente de θ.

Por fim, para finalizarmos a primeira parte do algoritmo, inicializamos uma última variável, chamada $coef$, onde serão concatenados os valores dos coeficientes dos polinômios interpoladores já multiplicado pelos k's do método, ou seja, o valor do somatório $\sum_{j=1}^{4} \tilde{b}_j(\theta)k_{i,j}$, conforme Eq. (4.12). Essa variável, assim como w, é um dado de saída do algoritmo. Porém a variável $coef$ é iniciada como uma matriz tridimensional, pois para cada passo e para cada equação do PVI, são gravados três coeficientes, um para cada potência de θ. Essa variável é de extrema importância para o processo de interpolação, pois através dos coeficientes que são gravados nela temos a possibilidade de realizar a interpolação em qualquer espaço entre dois pontos da malha, como mostraremos na quarta

parte do algoritmo.

Vamos agora para a segunda parte do programa principal, cujo código em MATLAB está apresentado no Quadro 4.3.

Quadro 4.3 – Segunda parte do algoritmo

```
%%%%%%%%%%%%%%%% Inicio do Metodo de Runge-Kutta Continuo %%%%%%%%%%%%
for i = 1:N     % Looping Principal, percorre os pontos da malha

    %%%%%%% Obtem os k's do metodo de Runge-Kutta no passo i %%%%%%
    for j = 1:4

    T = t(i)+h*c(j);                  % Calcula primeiro argumento da f
    W = w(i,:)+h*sum(a(j,:).'.*k,1); % Calcula segundo argumento da f
    Td = T - r(T,W);     % Calcula os argumentos de retardo
    Wd = rkc4interp(t,h,w,Td,phi,coef,n,d); % Calcula terceiro
                                            % argumento da f
    k(j,:) = f(T, W, Wd); % Calcula k
```

Nesta parte temos os cálculos dos k's, conforme Eq. (4.12), cujos resultados são utilizados na terceira parte para o cálculo da solução aproximada do passo. Sendo assim, iniciamos nessa parte do algoritmo um *looping* principal para realizar a iteração de cada passo do método. O cálculo dos k's é também realizado dentro de outro *looping*, para que os valores dos 4 k's do método RKC4 sejam obtidos. Assim, a cada iteração, calculamos primeiramente $t_{i,j}$ e $w_{i,j}$, armazenando os resultados nas variáveis T e W, respectivamente, onde W é um vetor, visto que o PVI pode conter mais de uma equação. Com isso, temos os valores para os dois primeiros argumentos da f para o cálculo do k.

Quanto ao terceiro argumento da f, calculamos primeiramente o argumento de retardo para cada um dos retardos do PVI, ou seja, $t_{i,j} - \sigma_d(t_{i,j}, w_{i,j})$, e armazenamos o valor na variável Td, sendo ela um vetor, já que o PVI pode conter vários retardos. Dessa forma, é possível finalmente calcular $u(t_{i,j} - \sigma_d)$, o qual armazenamos na variável Wd. Esse calculo da variável Wd é feito usando a parte de interpolação do algoritmo, que explicamos na quarta parte, sendo Wd uma matriz de n linhas e d colunas, com os valores da avaliação de cada solução aproximada para cada retardo sendo concatenadas de forma a ficarem no mesmo padrão da Eq. (4.16).

Com os três argumentos da f, calculamos por fim o valor de k. O processo então se repete novamente, dentro do *looping* de cálculo dos k's, para que tenhamos todos os 4 k's do método RKC4 calculados.

Na terceira parte do algoritmo, cujo código é mostrado no Quadro 4.4, concatenamos nas variáveis $coef$ e w, que são os dados de saída, os seus respectivos valores calculados

para o passo atual. Essa parte ainda está dentro do *looping* principal, de iteração do passo, que iniciamos na segunda parte do algoritmo. Porém, iniciamos mais um *looping* para calcularmos os valores de cada equação do PVI.

Quadro 4.4 – Terceira parte do algoritmo

```
        end

        %%%%%%%%%%%%%%%% Concatena os Coeficiente %%%%%%%%%%%%%%%%%%%%%%
         for g = 1:n
        coef(i,:,g)=b1.*k(1,g)+b2.*k(2,g)+b3.*k(3,g)+b4.*k(4,g);

        %%%%%% Calcula a aproximacao p/ a solucao y(t(i+1)) %%%%%%%%%%
        w(i+1,g) = w(i,g) + ...
            h*(coef(i,1,g)*(1^3)+coef(i,2,g)*(1^2)+coef(i,3,g)*(1));
        end
```

Com os valores obtidos dos 4 k's, realizamos o somatório da multiplicação dos k's com os b's, conforme Eq. (4.13). Porém, ainda mantemos esse resultado na forma de um vetor de três posições, cada posição sendo o coeficiente de cada potência de θ. Esse vetor é concatenado na matriz tridimensional $coef$, para aquele passo i e para aquela função g do PVI.

Por fim, calculamos a solução aproximada para o próximo passo, $w^{(i+1)}$, conforme Eq. (4.13). Para isso, utilizamos as 3 posições do vetor que acabamos de concatenar em $coef$, multiplicando cada posição por $\theta = 1$, elevado à potência daquele θ. Isso é feito pois, como $\theta \in [0, 1]$, ao adotar $\theta = 1$, estamos calculando a solução aproximada w no próximo ponto da malha. Ao finalizar esse processo, ele se repete para o calculo de $coef$ e w das demais equações do PVI.

Finalmente, ao encerrarmos a iteração desse passo, o algoritmo retorna ao inicio do *looping*, indo para o próximo passo da iteração, refazendo toda a segunda e terceira parte do algoritmo novamente. Ao finalizar todos os passos, o algoritmo irá se encerrar, passando os valores das matrizes w e $coef$, além do vetor t calculado no início do algoritmo, como dados de saída para o usuário. É importante notar que a variável w é a solução discreta do método RKC4, nos pontos da malha determinados pelo vetor t, enquanto que a variável $coef$ nos fornece a solução contínua obtidas pelos polinômios interpoladores. Logo, somente com a união dessas duas variáveis é que temos a solução contínua $u(t_i + h\theta)$ em cada intervalo $[t_i, t_{i+1}]$.

A quarta e última parte do algoritmo é onde temos a interpolação do método, cujo código é mostrado no Quadro 4.5. Essa parte consiste de uma nova função, a qual denominamos de `rkc4interp`, e que é chamada a cada calculo dos k's, em cada iteração

do passo, para se obter o valor de Wd. Esse processo é realizado pois, como sabemos, ao avaliarmos a solução aproximada com um retardo, essa avaliação pode resultar em um valor entre dois pontos da malha, como ocorre frequentemente.

Quadro 4.5 – Quarta parte do algoritmo

```
end                    % Fim do Looping Principal
end                        %%% Fim da Funcao %%%

%%%%%%%%%%%%%%%%%%%%%% Funcao rck4interp %%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%% Calcula a solucao aproximada no ponto com retardo  %%%%%%%%
function [Wd] = rkc4interp(t,h,w,Td,phi,coef,n,d)
Wd = zeros(n,d);
for q = 1:d
  if Td(q) ≤ t(1)  % Verifica se argumento de retardo < t_0
     Wd(:,q) = phi(Td(q)); % Sim = calcula usando a funcao historia
  else                      % Nao = faz Interpolacao cubica de Hermite
      k = find(t≤Td(q),1,'last'); % Procura [t_k,t_{k+1}] c/ o ponto Td
      theta = (Td(q)-t(k))/h;  %  Mapea Td em [0,1]
    for p = 1:n
        Wd(p,q) = w(k,p) + h*(coef(k,1,p)*(theta^3) + ...
            coef(k,2,p)*(theta^2) + coef(k,3,p)*theta);
    end
  end
end
```

Assim, passamos para essa função `rkc4interp` as variáveis necessárias para o cálculo da interpolação, sendo as principais a variável $coef$ e o valor do tempo calculado com o retardo Td. Ao final, essa função retornará o valor calculado de Wd.

Iniciamos a variável Wd como uma matriz com n linhas e d colunas para termos concatenados nessa variável os valores das aproximações de todas as equações do PVI para todos os retardos. Em seguida iniciamos um primeiro *looping*, de forma que todos os cálculos sejam feitos para todos os retardos do PVI.

Fazemos, inicialmente, uma verificação de condição para o primeiro retardo que estamos tratando. Nela, verificamos se o argumento de retardo $Td = t - \sigma_d \leq t_0$. Em caso positivo, não é necessário realizar a interpolação, sendo a solução aproximada que estamos procurando igual à função história φ, no tempo Td. Mas, em caso negativo, devemos seguir para a interpolação do método. Assim, identificamos primeiramente o intervalo $[t_k, t_{k+1}]$ em que o ponto Td está inserido e realizamos a mudança de variável para mapear esse ponto Td no intervalo $[0, 1]$.

Seguindo, iniciamos mais um *looping* para auxiliar agora no cálculo e concatenação do Wd para cada uma das equações do PVI para aquele retardo. Dentro desse *looping*

calculamos o Wd da mesma forma que calculamos $w^{(i+1)}$ na terceira parte do algoritmo, mas assumindo θ como o valor encontrado na mudança de variável de Td, ou seja, um valor específico no intervalo $[0, 1]$ que trará a solução aproximada em um ponto fora da malha.

Ao concluirmos esse último cálculo para as n equações do PVI para aquele retardo, retornamos para o início do primeiro *looping* e realizamos todo o processo novamente para o próximo retardo do PVI. Ao final de todas as iterações, é retornada a matriz Wd, que é o terceiro argumento da f para cálculo do k.

Concluímos a apresentação e explicação do algoritmo do método RKC4, cujo código foi implementado em MATLAB. O código completo está também apresentado no Apêndice A. Na próxima seção apresentamos os resultados obtidos com o algoritmo implementado na solução de exemplos de EDRs. A fim de confirmarmos que nosso código está com o método implementado corretamente, e também para termos resultados da eficiência do método, comparamos em cada exemplo o resultado obtido pelo algoritmo com a solução analítica e também com a solução obtida pelo `ddesd`, que é uma função nativa do MATLAB.

É valido reforçar que, apesar de algumas comparações quantitativas que fizemos entre o algoritmo e o `ddesd`, elas não foram feitas com o propósito de indicar qual função está com uma melhor ou pior eficiência. Isso porque, apesar de ambas serem destinadas para solução de EDR's com retardos dependendo do estado, elas são constituídas de métodos numéricos diferentes, visto que a função `ddesd` utiliza o método de RK4 convencional, com um interpolante para a extensão contínua, verificação de erro em cada passo para determinar um passo h variado, entre outros aspectos matemáticos, conforme pode ser verificado em Shampine (2005).

4.4 Exemplos de EDRs com Runge-Kutta Contínuo

Retomamos aqui os exemplos de EDRs da Seção 2.3, os quais resolvemos novamente utilizando agora o método RKC4. Dessa forma, iniciamos com o exemplo de Bellen e Zennaro (2003), e na sequência abordamos os exemplos de Paul (1994).

O método foi implementado em linguagem computacional do MATLAB, conforme apresentamos na seção anterior, e por isso mostraremos em cada exemplo a eficiência do algoritmo, comparando a solução aproximada obtida com a solução exata e também com a solução dada por uma função nativa do MATLAB, a função `ddesd`, que resolve EDR's com qualquer tipo de retardo.

Exemplo 4.2. Considere o PVI,

$$\begin{cases} y'(t) = -y(t-1), & t \in [0,3], \\ y(t) = 1, & t \in [-1,0]. \end{cases} \tag{4.18}$$

Os parâmetros de entrada para o algoritmo foram o tempo inicial, $t_0 = 0$, o tempo final, $t_f = 3$, e a condição inicial, $y^{(0)} = 1$. Vamos considerar ainda $N = 50$ iterações, o que resulta em um passo $h = 0,06$. O retardo nesse exemplo é $\sigma = 1$ constante. O Quadro 4.6 apresenta o código desse PVI.

Quadro 4.6 – Código do Exemplo 4.2

```
f   = @(t,y,yd) [-yd(1,1)];
phi = @(t) [1];
r   = @(t,y) [1];
```

Sendo a solução analítica dada pela Eq. (2.20), que encontramos resolvendo esse mesmo exemplo na Seção 2.3, apresentamos na Fig. 8 o gráfico da solução exata e das soluções aproximadas pelo nosso algoritmo do RKC4 e pela função `ddesd`.

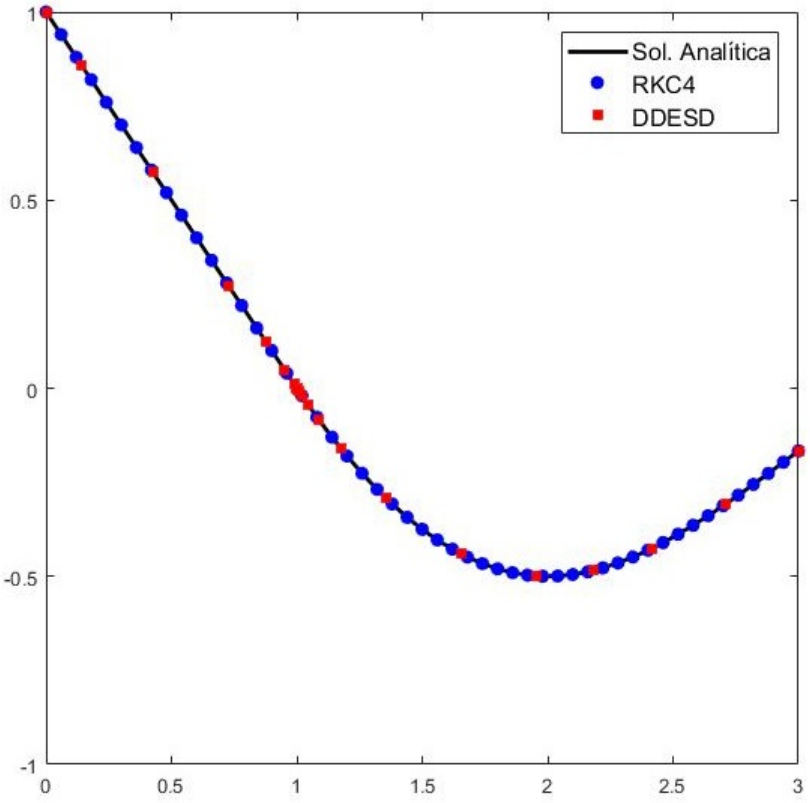


Figura 8 – Solução exata e aproximações obtidas pelo RKC4 e pelo `ddesd` do Exemplo 4.2, conforme Eq. (4.18), para o intervalo $[0,3]$.

Na Tabela 7 apresentamos também as soluções aproximadas obtidas por cada programa, comparando-os com o valor da solução exata para o primeiro e último ponto, além de mais três pontos aleatórios dentro do intervalo $[0, 3]$. Esses pontos aleatórios são, na verdade, pontos discretos utilizados pela função `ddesd` do MATLAB na resolução do PVI, pois ela utiliza um passo h variável ao invés de um passo constante, como nosso algoritmo do RKC4. Dessa forma, escolhemos três pontos dentre os determinados pela função `ddesd` para termos o valor da solução aproximada que a função gerou. Mesmo esses três pontos não coincidindo com pontos que o passo h de nosso algoritmo do RKC4 tenha gerado, foi possível determinar a solução aproximada pelo RKC4 por ele ser um método contínuo.

Tabela 7 – Valores da solução exata, y, das soluções aproximadas, w, pelo RKC4 e pelo `ddesd`, e erro entre as soluções aproximadas e a solução exata, para cinco pontos no intervalo $[0, 3]$.

		RKC4		DDESD	
t	y	w	$\lvert y - w\rvert$	w	$\lvert y - w\rvert$
0	1,00	1,00	0	1,00	0
0,98	0,05	0,05	0	0,05	0
1,01	-0,01	-0,01	$3{,}89 \times 10^{-6}$	-0,01	$1{,}19 \times 10^{-8}$
2,19	-0,48	-0,48	$1{,}56 \times 10^{-6}$	-0,48	$2{,}27 \times 10^{-2}$
3,00	-0,17	-0,17	$1{,}54 \times 10^{-6}$	-0,17	$2{,}19 \times 10^{-5}$

É possível notar pela Tabela 7 que as soluções do RKC4 e do `ddesd` coincidem até a segunda casa decimal, de onde concluímos que o RKC4 apresenta um bom desempenho. Comparando o tempo computacional dos dois programas, o RKC4 apresentou melhor peformance, (0,042 segundo para o RKC4 contra 0,087 segundo para o `ddesd`), mesmo realizando uma maior quantidade de passos ($N = 50$ para o RKC4 contra $N = 22$ para o `ddesd`).

Exemplo 4.3. Considere o sistema de duas equações abaixo

$$\begin{cases} y_1'(t) = y_2(t) & t \in [0, 2], \\ y_2'(t) = 1 - y_2(t-1) - y_1(t) & t \in [0, 2], \\ y_1(t) = 0 & t \in [-1, 0], \\ y_2(t) = 0 & t \in [-1, 0], \end{cases} \tag{4.19}$$

Como valores de entrada para esse exemplo temos $t_0 = 0$, $t_f = 2$ e $y_1^{(0)} = y_2^{(0)} = 0$. Vamos adotar novamente $N = 50$, o que nos dará um passo $h = 0,04$. Nesse exemplo, trabalhamos com duas equações/variáveis, mas somente com um retardo, novamente constante, $\sigma = 1$. O código para esse exemplo é mostrado no Quadro 4.7.

Quadro 4.7 – Código do Exemplo 4.3

```
f   = @(t,y,yd) [y(2),
                 1-yd(2,1)-y(1)];
phi = @(t) [0,0];
r   = @(t,y) [1];
```

A solução analítica para esse exemplo é dada em (2.22), para o intervalo $t \in [0, 2]$, e a Fig. 9 apresenta o gráfico das soluções numéricas pelo nosso algoritmo de RKC4 e pela função `ddesd`, em comparação com a solução exata. Já a Tabela 8 apresenta os valores do erro entre as soluções numérica e a exata, em 5 pontos distintos do intervalo $[0, 2]$.

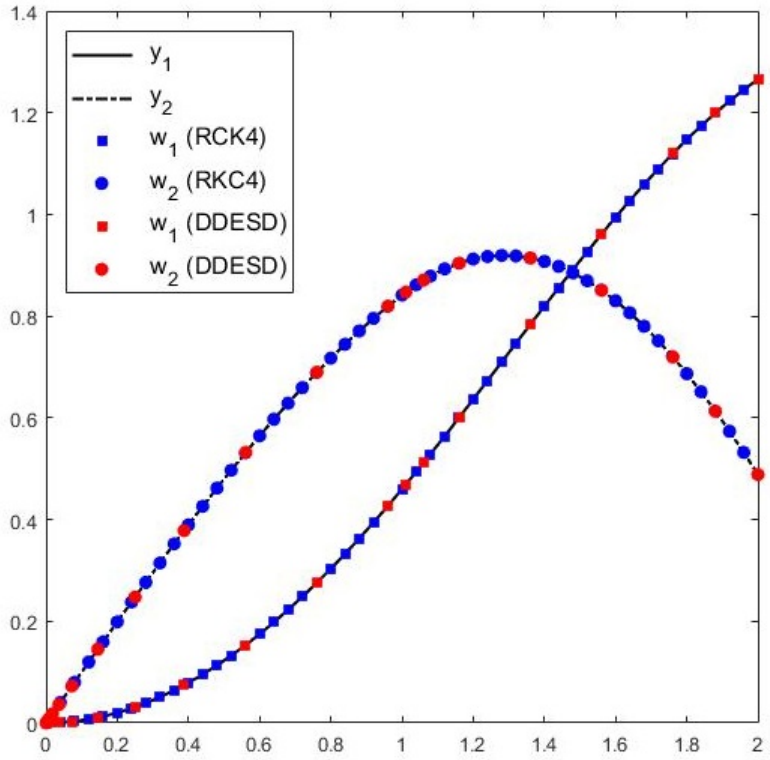


Figura 9 – Solução exata e aproximações obtidas pelo RKC4 e pelo `ddesd` do Exemplo 4.3, conforme Eq. (4.19), para o intervalo $[0, 2]$.

Tabela 8 – Valores do erro entre a solução exata e as soluções aproximadas, w_n, obtidas pelo RKC4 e pelo `ddesd`, para ambas as variáveis/equações.

	RKC4		DDESD	
t	$\lvert y_1 - w_1\rvert$	$\lvert y_2 - w_2\rvert$	$\lvert y_1 - w_1\rvert$	$\lvert y_2 - w_2\rvert$
0	0	0	0	0
0,004	$3{,}33 \times 10^{-9}$	$1{,}39 \times 10^{-14}$	$4{,}83 \times 10^{-17}$	$5{,}26 \times 10^{-16}$
0,25	$1{,}25 \times 10^{-8}$	$8{,}33 \times 10^{-9}$	$2{,}03 \times 10^{-8}$	$1{,}20 \times 10^{-7}$
1,36	$3{,}63 \times 10^{-8}$	$2{,}62 \times 10^{-9}$	$2{,}28 \times 10^{-5}$	$3{,}25 \times 10^{-5}$
2,00	$4{,}60 \times 10^{-8}$	$4{,}78 \times 10^{-8}$	$4{,}67 \times 10^{-5}$	$4{,}96 \times 10^{-6}$

Exemplo 4.4. Considere o seguinte sistema

$$\begin{cases} y_1'(t) = y_2(t) & t \geq 2, \\ y_2'(t) = -\frac{1}{2}y_1(t) - \frac{1}{2} + y_1\left(\frac{1}{2}t - \frac{\pi}{4}\right)^2 & t \geq 2, \\ y_1(t) = \text{sen}(t) & t \leq 2, \\ y_2(t) = \cos(t) & t \leq 2. \end{cases} \tag{4.20}$$

No programa, colocamos como valores de entrada $t_0 = 2$, $t_f = 10$, $y_1^{(0)} = \text{sen}(t_0)$, $y_2^{(0)} = \cos(t_0)$, e $N = 100$, gerando um passo $h = 0,08$. Neste exemplo temos um retardo dependendo do tempo, visto que $t - \sigma = (1/2)t - \pi/4$, logo, $\sigma(t) = (1/2)t + \pi/4$. O código em MATLAB que define esse problema é mostrado no Quadro 4.8.

Quadro 4.8 – Código do Exemplo 4.4

```
f   = @(t,y,yd) [y(2),
                 -0.5*y(1)-0.5+yd(1,1)^2];
phi = @(t) [sin(t),cos(t)];
r   = @(t,y) [0.5*t+pi/4];
```

A solução analítica é dada na Eq. (2.24), para $t \leq 2$, sendo que nesse exemplo vamos adotar o intervalo $t \in [2, 10]$. Dessa forma, a Fig. 10 apresenta o gráfico da solução analítica, em comparação com as soluções numéricas pelo algoritmo de RKC4 e pela função `ddesd`. E a Tabela 9 mostra os valores do erro entre as soluções numéricas e a analítica, em 5 pontos do intervalo.

Tabela 9 – Valores do erro entre a solução exata e as soluções aproximadas, w_n, obtidas pelo RKC4 e pelo `ddesd`, para ambas as variáveis/equações.

	RKC4		DDESD	
t	$\lvert y_1 - w_1\rvert$	$\lvert y_2 - w_2\rvert$	$\lvert y_1 - w_1\rvert$	$\lvert y_2 - w_2\rvert$
2,00	0	0	0	0
3,12	$1{,}79 \times 10^{-7}$	$3{,}37 \times 10^{-8}$	$3{,}91 \times 10^{-5}$	$2{,}34 \times 10^{-6}$
5,06	$2{,}18 \times 10^{-7}$	$3{,}68 \times 10^{-7}$	$4{,}02 \times 10^{-5}$	$7{,}36 \times 10^{-5}$
7,69	$8{,}50 \times 10^{-7}$	$2{,}46 \times 10^{-7}$	$1{,}93 \times 10^{-4}$	$5{,}82 \times 10^{-5}$
10,00	$2{,}96 \times 10^{-7}$	$1{,}21 \times 10^{-7}$	$9{,}21 \times 10^{-5}$	$2{,}01 \times 10^{-5}$

Exemplo 4.5. Considere o seguinte PVI,

$$\begin{cases} y'(t) = \dfrac{y(t)y(\ln(y(t)))}{t}, & t \geq 1, \\ y(t) = 1, & t \leq 1, \end{cases} \tag{4.21}$$

Para esse exemplo, utilizamos como parâmetros de entrada para o programa $t_0 = 1$, $t_f = e^2$, $y^{(0)} = 1$ e $N = 25$, sendo o passo $h = 0,2556$. O PVI apresenta um retardo

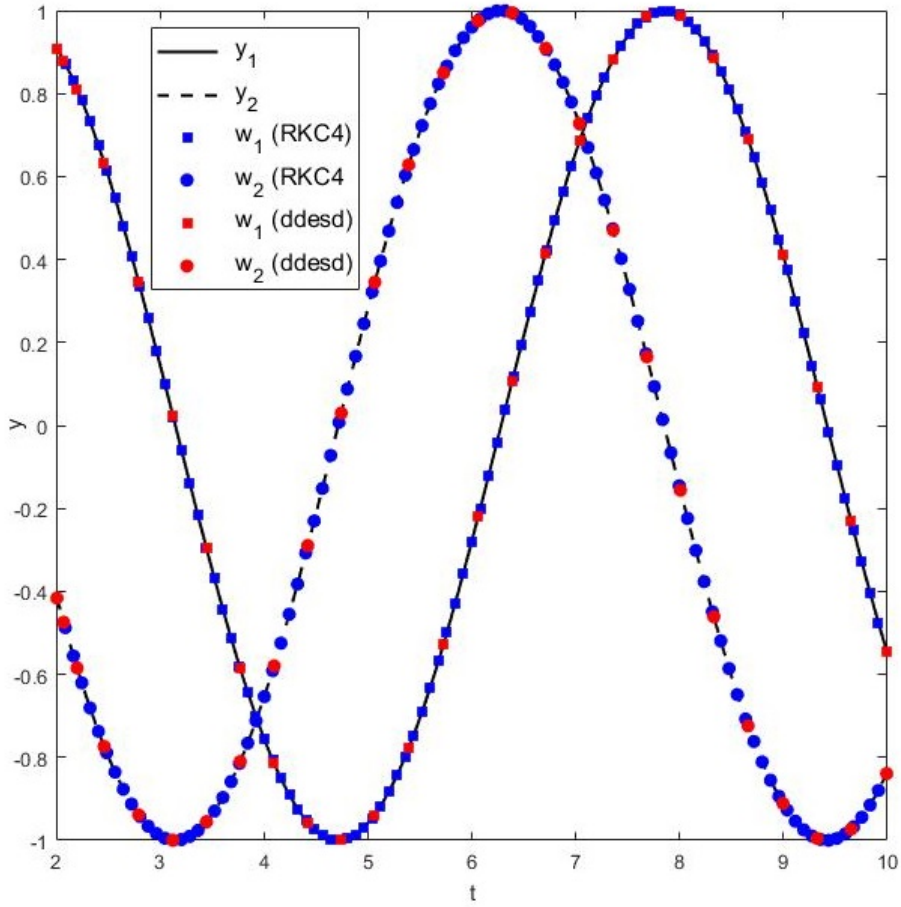


Figura 10 – Solução exata e aproximações obtidas pelo RKC4 e pelo `ddesd` do Exemplo 4.4, conforme Eq. (4.20), para o intervalo $[2, 10]$.

dependendo do estado, visto que, $t - \sigma = \ln(y(t))$, e logo, $\sigma(t, y) = t - \ln(y(t))$. O Quadro 4.9 apresenta o código para esse exemplo.

Quadro 4.9 – Código do Exemplo 4.5

```
f   = @(t,y,yd) [(y(1)*yd(1,1))/t];
phi = @(t) [1];
r   = @(t,y) [t-log(y)];
```

A Eq. (2.26) é a solução analítica para o exemplo, no intervalo $t \in [1, e^2]$. E assim, temos apresentado na Fig. 11 o gráfico da solução analítica e das duas soluções numéricas, do RKC4 e do `ddesd`, e na Tabela 10 temos a comparação dos erros entre essas soluções numéricas e a solução exata.

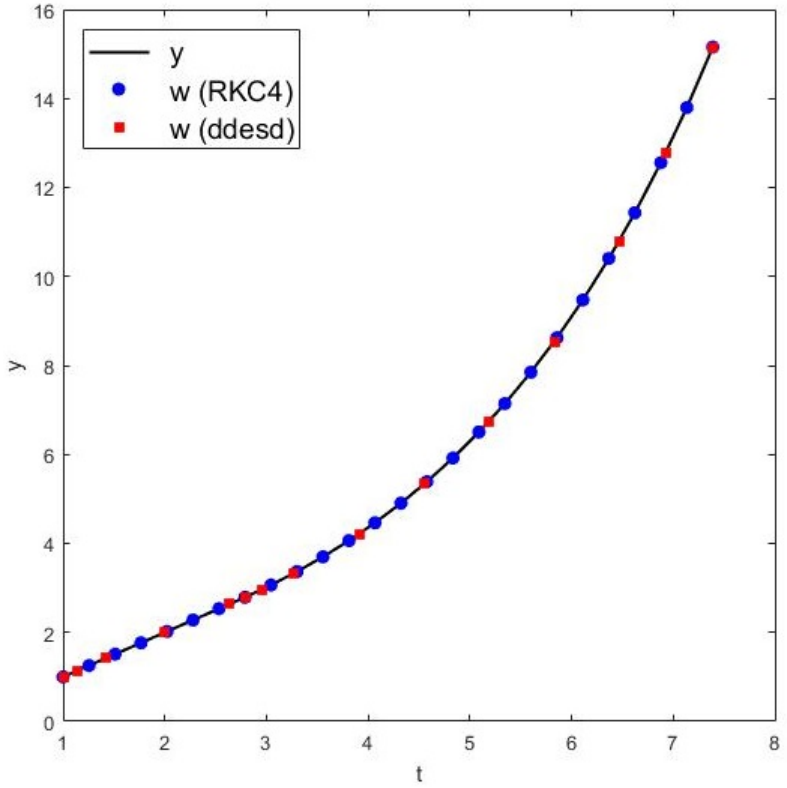


Figura 11 – Solução exata a aproximações obtidas pelo RKC4 e pelo `ddesd` do Exemplo 4.5, conforme Eq. (4.21), para o intervalo $[1, e^2]$.

Tabela 10 – Valores da solução exata, y, e as soluções aproximadas, w, obtidas pelo RKC4 e pelo `ddesd`, e o erro entre a solução exata e essas mesmas soluções numéricas.

		RKC4		DDESD	
t	y	w	$\lvert y-w\rvert$	w	$\lvert y-w\rvert$
1,00	1,00	1,00	0	1,00	0
2,63	2,63	2,63	2,43 $\times 10^{-4}$	2,63	0
3,91	4,21	4,22	3,36 $\times 10^{-4}$	4,21	8,55 $\times 10^{-4}$
5,83	8,54	8,54	1,01 $\times 10^{-3}$	8,53	3,17 $\times 10^{-3}$
7,39	15,15	15,16	2,11 $\times 10^{-3}$	15,14	7,51 $\times 10^{-3}$

5

Aplicações em dinâmica populacional

No Capitulo 2 foi apresentado um estudo quanto à teoria de EDOs e EDRs, que utilizamos nos capítulos seguintes como base para o objeto principal de estudo desse projeto, o método numérico de Runge-Kutta Contínuo (RKC). Porém, até o momento não discutimos a aplicação de equações diferenciais em problemas reais, mas apenas em problemas matemáticos. Sendo assim, nesse capítulo apresentamos um estudo voltado para a modelagem e aplicação de EDOs e EDRs em dinâmica populacional. Mais especificamente, focamos nos modelos de crescimento malthusiano e logístico, este último caracterizado pelo existência da capacidade de suporte do ambiente. Também abordaremos o clássico modelo compartimental SIR (suscetível-infeccioso-recuperado), amplamente utilizado para o estudo de propagação de doenças infecciosas em uma dada população. Assim, apresentamos um estudo teórico desses dois modelos, com exemplos de modelagem deles por EDRs e utilizando o programa implementado do algoritmo do método RKC para resolver esses exemplos.

5.1 Modelos de crescimento malthusiano e logístico

O crescimento dos estudos sobre modelos matemáticos para descrever a dinâmica de populações biológicas, seja tratando de uma população humana, de espécies em risco de extinção ou crescimento viral ou bacteriano, é reflexo da eficiência e, consequentemente, da confiabilidade desses modelos quando aplicados para retratar fenômenos do mundo real (MURRAY, 2002). Dentre os modelos de crescimento populacional, um dos mais simples e introdutórios ao tema é o proposto pelo economista britânico Thomas Malthus, em 1798. De acordo com esse modelo, a taxa y' com que o tamanho y de uma população varia é

proporcional ao próprio tamanho da população, ou seja,

$$y' = ry, \tag{5.1}$$

onde taxa r é uma constante denominada de taxa de crescimento/decaimento. Naturalmente que a população cresce se $r > 0$ ou decresce para $r < 0$.

Considerando que a Eq. (5.1) esteja sujeita a uma condição inicial $y(0) = y_0$, temos assim um PVI, cuja solução analítica é

$$y = y_0 e^{rt}. \tag{5.2}$$

O modelo de Malthus é capaz de descrever o crescimento de muitas populações reais, pelo menos em condições ideais e por um período de tempo limitado. Porém, é certo de que tais condições ideais não são permanentes e, em algum momento, certas limitações, como a falta de suprimento e recursos ou falta de espaço, irá inibir esse crescimento exponencial (BOYCE; DIPRIMA, 2015).

Já o modelo logístico, proposto por Pierre F. Verhulst em 1838, apresenta uma forma de modelar inibições ao crescimento da população. Para isso, a taxa de crescimento/decaimento, que no modelo malthusiano é uma constante, passa a ser uma função $h(y)$ que depende do tamanho da população y. Neste caso, a Eq. (5.1) torna-se

$$y' = h(y)y, \tag{5.3}$$

Segundo Verhulst, a função $h(y)$ segue algumas condições, sendo que $h(y) > 0$ quando y for suficientemente pequeno, $h(y) < 0$ quando y for suficientemente grande e $h(y)$ decresce quando y crescer. Assim, existem diferentes maneiras de modelar a função $h(y)$ de forma a satisfazer essas condições propostas. A mais simples é considerar que $h(y)$ é uma função linear decrescente de y, ou seja, $h(y) = r - ay$, com a sendo uma constante positiva (BOYCE; DIPRIMA, 2015), e assim temos que

$$y' = h(y)y = (r - ay)y = ry - ay^2 = ry\left(1 - \frac{ay}{r}\right),$$

logo,

$$y' = ry\left(1 - \frac{y}{K}\right), \tag{5.4}$$

onde r é a taxa de crescimento intrínseco, e $K = r/a$ é o nível de saturação, ou capacidade ambiental de sustentação, que representa as limitações imposta ao crescimento populacional. Considerando que a Eq. (5.4) esteja sujeita a uma condição inicial $y(0) = y_0$, a solução

analítica é dada por

$$y = \frac{y_0 K}{y_0 + (K - y_0)e^{-rt}}. \tag{5.5}$$

Nesse modelo, o K (capacidade ambiental de sustentação) representa o nível ideal da população, tal que populações abaixo dele tendem ao crescimento até alcançar esse limitante, enquanto que populações acima dele tendem ao decrescimento até voltarem ao nível de saturação. Podemos considerar o K, por exemplo, como a disponibilidade de alimento para uma determinada espécie. Assim, se temos uma população abaixo desse limitante, essa população tende a se reproduzir e crescer até o limite da disponibilidade de comida. Enquanto que se a população estiver acima do limitante, ela tenderá ao decrescimento, visto que não haveria disponibilidade de comida para todos.

Apenas para ilustrar, apresentamos na Fig. 12 o comportamento geral das soluções para modelos malthusiano e logístico, dadas pelas Eqs. (5.2) e (5.5), respectivamente.

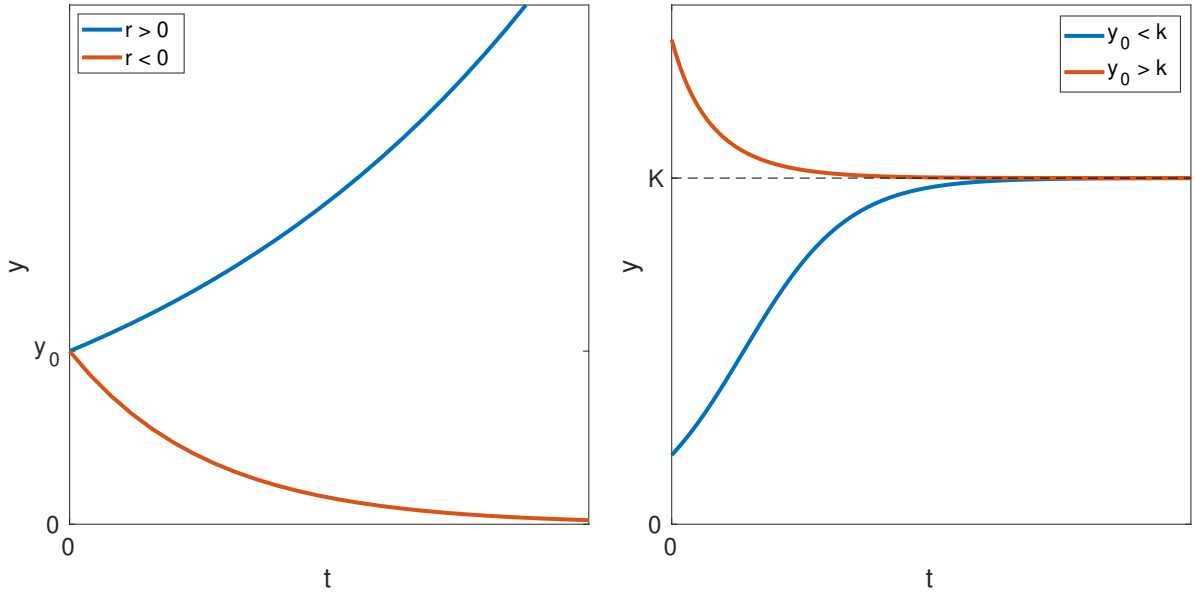


Figura 12 – Comportamento da solução y em função de t para os modelos de crescimento/decaimento (a) malthusiano, dada pela Eq. (5.2), e (b) logístico, dada pela Eq. (5.5).

Tanto o modelo malthusiano quanto o logístico, conforme apresentados nas Eqs. (5.1) e (5.4), respectivamente, consideram que os indivíduos nascem já maduros para poderem se reproduzir. Sabemos, no entanto, que essa modelagem não é realista para muitas populações. Por exemplo, no caso de seres humanos, além do período de gestação de 9 meses para o indivíduo nascer, ainda existe um período de anos até que esse indivíduo atinja a maturidade e possa se reproduzir.

Esse período para adquirir maturidade de reprodução pode ser incluído nos modelos (5.1) e (5.4) transformando as EDOs em EDRs. Para o modelo malthusiano, por exemplo,

obtém-se

$$y' = r(y - \sigma), \tag{5.6}$$

onde σ representa o período de maturação, ou seja, o tempo entre nascer e estar apto para a reprodução. Como exemplo, utilizamos o programa RKC4, com $N = 100$ passos, para resolver esse modelo considerando $\varphi = 1$, $r = 0,01$ e diferentes valores de σ (unidades arbitrárias) e os resultados estão mostrados na Fig. 14.

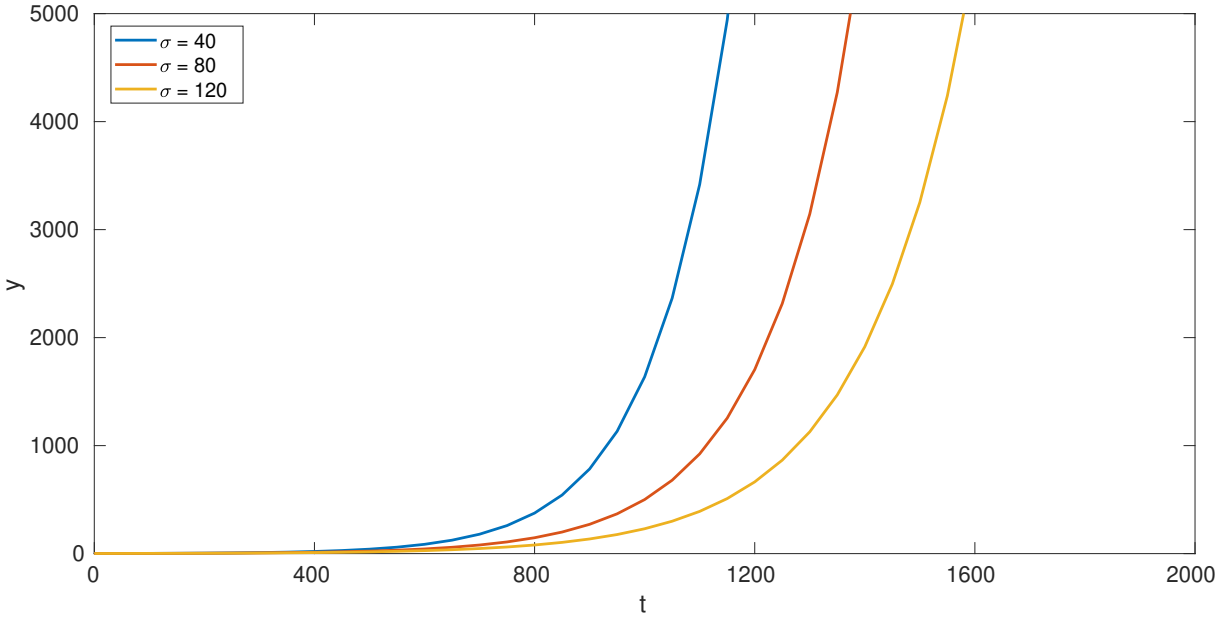


Figura 13 – Solução numérica obtida pelo RKC4, com $N = 100$ passos, para o modelos de Malthus com retardo (5.6), considerando $\varphi = 1$, $r = 0,01$ e diferentes valores de σ (unidades arbitrárias).

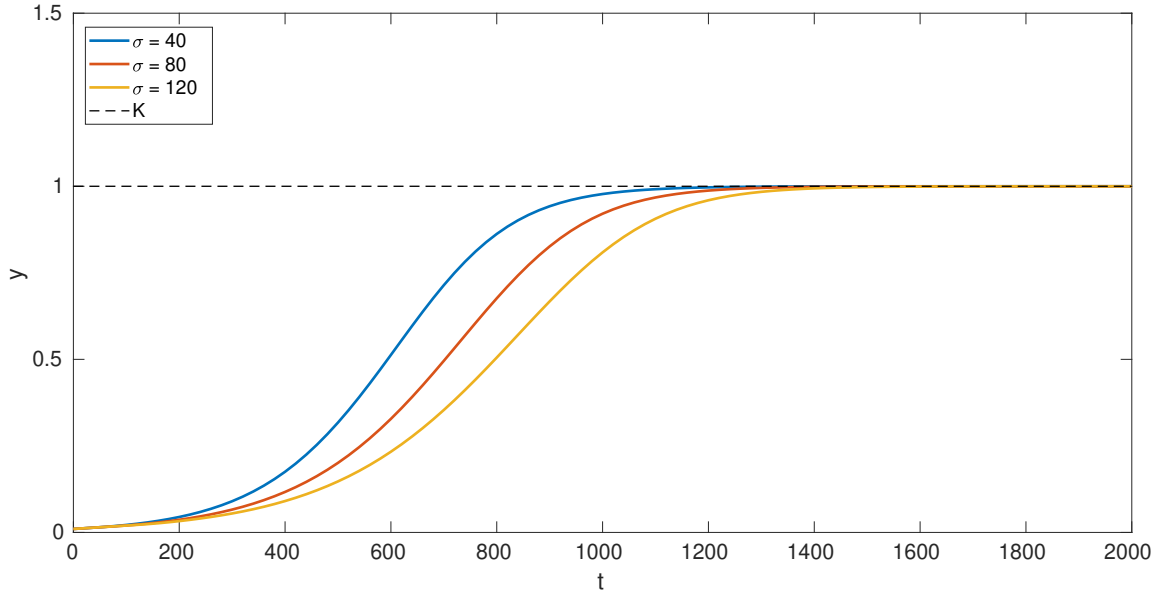
Já para o modelo logístico, o retardo σ pode ser inserido em dois locais diferentes, ou seja,

$$y'(t) = ry(t - \sigma)\left(1 - \frac{y(t)}{K}\right), \tag{5.7}$$

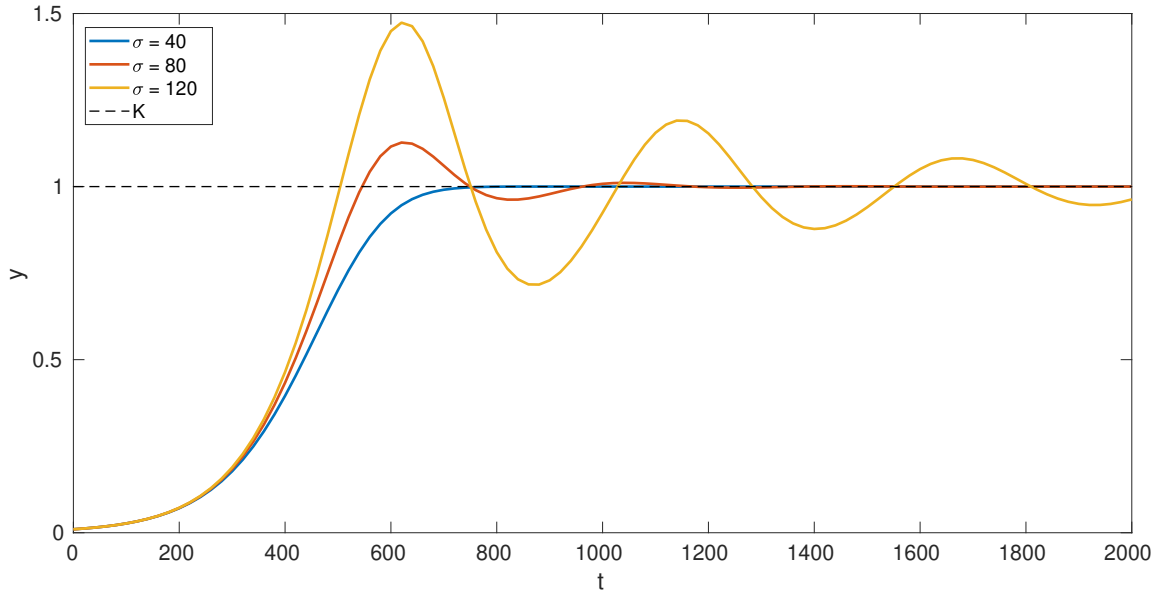
ou

$$y'(t) = ry(t)\left(1 - \frac{y(t - \sigma)}{K}\right). \tag{5.8}$$

Ambos modelos, (5.7) e (5.8) foram resolvidos numericamente pelo programa RKC4, com $N = 100$ passos, considerando $r = 0,01$, $K = 1$ e $\varphi = K/100$, com diferentes valores de σ (unidades arbitrárias). Os resultados são mostrados na Fig. 14.



(a)



(b)

Figura 14 – Solução numérica obtida pelo RKC4, com $N = 100$ passos, para o modelos logísticos com retardo dados pelas Eqs. (a) (5.7) e (b) (5.8), considerando $r = 0,01$, $K = 1$ e $\varphi = K/100$, com diferentes valores de σ (unidades arbitrárias).

5.2 Modelo epidemiológico SIR com retardo

Os modelos epidemiológicos investigam como as epidemias se propagam em uma dada população e se tornaram importantes ferramentas de estudo e controle de doenças infecciosas. A utilização de métodos matemáticos para estudar a disseminação de doenças contagiosas vem desde a década de 1760, pelo menos, quando Daniel Bernoulli

fez um trabalho relativo à varíola (BOYCE; DIPRIMA, 2015). Em anos mais recentes, muitos modelos matemáticos foram propostos e estudados para diversas doenças diferentes (Edelstein-Keshet, 2005; KEELING; ROHANI, 2008), tornando a modelagem matemática de doenças infecciosas uma das mais importantes áreas de pesquisa, por contribuir com um melhor entendimento da dinâmica dessas doenças, seus impactos e possíveis previsões de suas disseminações (ELHIA; RACHIK; BENLAHMAR, 2013).

Uma das maneiras mais simples de modelar a propagação de uma epidemia é dividir a população em diferentes classes, também chamados de compartimentos. Dentre os modelos que adotam essa abordagem, destaca-se o modelo SIR, no qual a população é dividida em três compartimentos, suscetíveis (S), infecciosos (I) e recuperados (R). Um indivíduo, inicialmente suscetível, pode adquirir a doença ao entrar em contato com um indivíduo infeccioso e assim passar do compartimento S para o compartimento I. Essa transição, S $\rightarrow$ I, é definida pelo parâmetro β, representando a taxa de infecção. Estando no compartimento I, o indivíduo passa a infectar outros indivíduos do compartimento S. No entanto, após um tempo, esse indivíduo se recupera e passa para o estado R. Essa transição, I $\rightarrow$ R, é definida pelo parâmetro γ, o qual chamamos de taxa de recuperação.

De forma a complementar esse modelo, podemos ainda destacar a ação de demografia nessa população. Com isso, destacamos no modelo a taxa de nascimento, ν, e a taxa de mortalidade, μ. Em geral, supomos que todo individuo nasce no compartimento dos suscetíveis, enquanto que que a taxa de mortalidade age sobre toda a população. A Fig. 15 ilustra os compartimentos do modelo SIR e as transições entre eles.

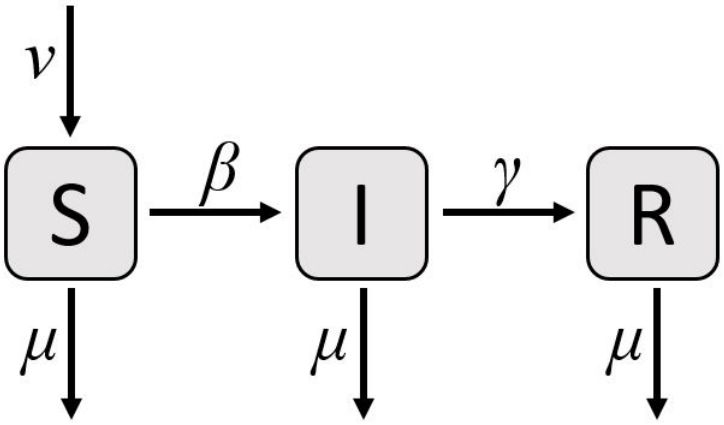


Figura 15 – Modelo compartimental SIR (suscetível-infeccioso-recuperado). As setas horizontais representam a movimentação da população entre os compartimentos, enquanto que as setas verticais representam a demografia.

A partir das suposições apresentadas, pode-se modelar a epidemia a partir de um sistema de equações diferenciais. Sendo $S(t)$, $I(t)$ e $R(t)$ as populações nos estados S, I e

R no tempo t, respectivamente, temos

$$\begin{cases} \dfrac{dS(t)}{dt} = \nu - \beta S(t)\dfrac{I(t)}{P(t)} - \mu S(t), \\ \dfrac{dI(t)}{dt} = \beta S(t)\dfrac{I(t)}{P(t)} - \gamma I(t) - \mu I(t), \\ \dfrac{dR(t)}{dt} = \gamma I(t) - \mu R(t), \end{cases} \tag{5.9}$$

com $P(t) = S(t) + I(t) + R(t)$ sendo o tamanho total da população.

O modelo (5.9) é não linear e não admite solução analítica, sendo necessários métodos numéricos para resolvê-lo. Vamos, então, utilizar o programa RKC4 para obter as soluções aproximadas para esse modelo. Para isso, adotamos os parâmetros e condições iniciais mostrados na Tabela 11, obtidos na Ref. (MENIN; BAUCH, 2018). Resolvemos o exemplo para um período de $t = 100$ dias, com $N = 1000$.

Quadro 5.1 – Código do exemplo com o modelo SIR conforme Eq. (5.9).

```
t0 = 0; tf = 100; m = 3; d = 1;
b = 0.5; g = 0.1; v = 4e-5; u = v;

f   = @(t,y,yd) [v-b*y(1)*(y(2)/(y(1)+y(2)+y(3)))-u*y(1),
                 b*y(1)*(y(2)/(y(1)+y(2)+y(3)))-g*y(2)-u*y(2),
                 g*y(2)-u*y(3)];
phi = @(t) [1e5,1,0];
r   = @(t,y) [0];
```

Tabela 11 – Valores base adotados para os parâmetros e condições iniciais para o modelo SIR (5.9), de acodo com Menin e Bauch (2018).

Parâmetro	Descrição	Valor
$S(0)$	Tamanho inicial da população de Suscetíveis	10^5
$I(0)$	Tamanho inicial da população de Infectados	1
$R(0)$	Tamanho inicial da população de Recuperados	0
β	Taxa de infecção pela doença	$0,5\,[\text{dia}^{-1}]$
γ	Taxa de recuperação da doença	$0,1\,[\text{dia}^{-1}]$
ν	Taxa de natalidade	4×10^{-5}
μ	Taxa de mortalidade	4×10^{-5}

A Figura 16 apresenta a solução numérica desse exemplo, enquanto que o Quadro 5.1 mostra o código para ser feito o teste em MATLAB, junto o programa RKC4.

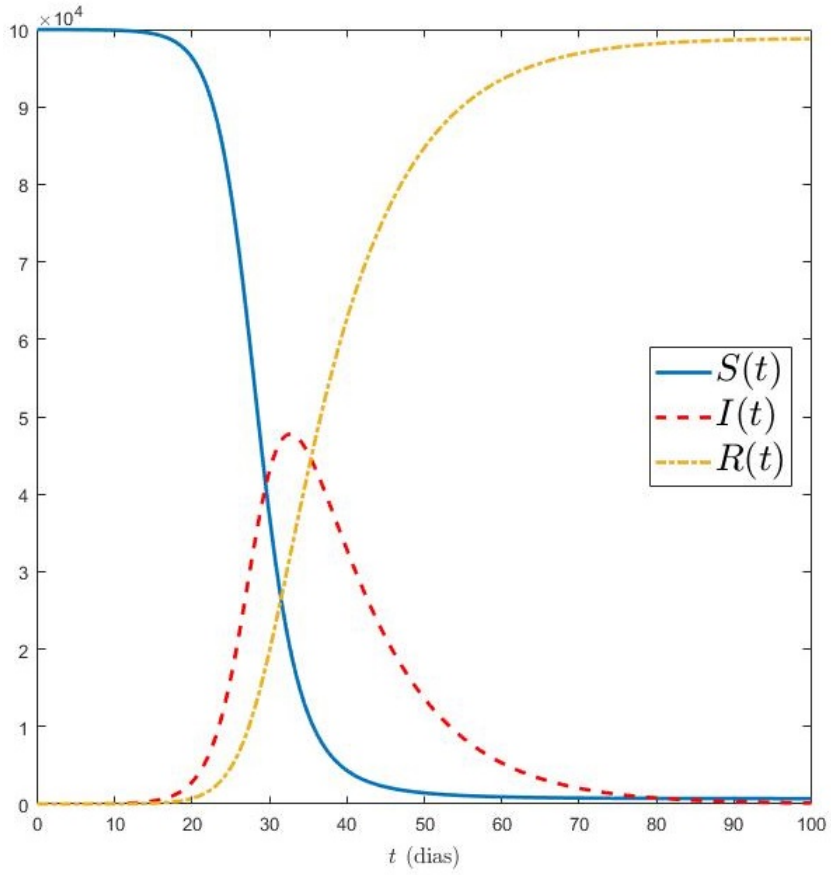


Figura 16 – Gráfico do exemplo do modelo SIR, conforme Eq. (5.9), sendo os valores dos parâmetros conforme a Tabela 11, para o intervalo $[0, 100]$.

Vemos nesse exemplo que a população de suscetíveis (linha contínua) começa a diminuir próximo ao 15º dia, praticamente no mesmo instante que a população de infectados (linha tracejada) começa a crescer, representando a transição S $\rightarrow$ I, que ocorre devido à interação entre a população de suscetíveis com a população de infectados e á modelada pelo termo de infecção $\beta S(t)I(t)/P(t)$. Observamos também que pouco após o início do crescimento da população de infectados, começa a crescer a população de recuperados. Essa a transição I $\rightarrow$ R, por consequência, faz com que a epidemia termine $(I(t_f) \approx 0)$ sem que toda a população tenha sido infectada $(S(t_f) > 0)$.

Esse modelo SIR que apresentamos pode ser utilizado na modelagem de doenças mais comuns em que, uma vez que o indivíduo infectado se recupera, ele se torna imune à doença e não pode mais ser infectado por ela. Naturalmente que, dependendo do tipo de doença e de outros fatores, pode-se adotar diferentes modelos compartimentais. O modelo SI (suscetível-infeccioso), por exemplo, modela doenças infecciosas para as quais não há recuperação, ou seja, uma vez infectado, o indivíduo permanece infeccioso por toda a vida. Também temos o modelo SIRS (suscetível-infeccioso-recuperado-suscetível), no qual o indivíduo, mesmo após contrair a infecção e se recuperar, pode retornar à condição

de suscetível.

Outra possibilidade de variação nos modelos SIR é a inclusão ou não da demografia. No exemplo que mostramos, incluímos taxas de nascimento e mortalidade para melhor exemplificar essa parte. Porém, é possível também desconsiderar completamente a demografia ($\nu = \mu = 0$) ou então considerá-la de tal forma a manter o tamanho da população constante ($\nu = \mu$), e assim modelar casos de epidemias relativamente curtas ou em populações controladas. Por outro lado, pode-se também complementar ainda mais a modelo, incluindo taxas de imigração e emigração, por exemplo, ou mesmo a mortalidade induzida em decorrência da própria doença.

Pensando nessas possibilidades de variação do modelo SIR tradicional, vamos incluir agora no modelo apresentado na Eq. (5.9) uma nova condição no nosso modelo epidemiológico: o período de incubação da doença.

A inclusão de incubação da doença coloca em nosso modelo uma condição/atraso para a transição S $\rightarrow$ I. Isso ocorre, pois o período de incubação da doença representa o tempo necessário para que aquele vírus se desenvolva e se multiplique em seu hospedeiro, até que a carga viral seja suficientemente grande para que o hospedeiro passe a infectar outros indivíduos. Em outras palavras, o indivíduo, em um primeiro momento se torna infectado pela doença, porém apenas após o período de incubação que ele se torna infeccioso, passando para o compartimento I.

Podemos representar essa incubação da doença inserindo um retardo na função $I(t)$ quando na interação com a população de suscetíveis, tal que o termo de infecção torna-se $\beta S(t)I(t-\sigma)/P(t)$, onde σ é o período de incubação. Assim, temos que a infecção de indivíduos suscetíveis no instante t se dá pela interação desses com indivíduos infectados em um instante $t-\sigma$, ou seja, no termo de infeção considera-se apenas indivíduos que foram infectados em um tempo passado e cujo tempo de incubação da doença já foi concluído. Ilustramos na Fig. 17 o modelo SIR com essa nova condição, onde a seta tracejada representa a transição entre compartimentos que não é imediata, ou seja, que possui um retardo.

Esse modelo epidemiológico com tempo de incubação pode ser escrito em um sistema de equações diferenciais com retardo, de forma similar à Eq. (5.9), tal que,

$$\begin{cases} \dfrac{dS(t)}{dt} = \nu - \beta S(t)\dfrac{I(t-\sigma)}{P(t)} - \mu S(t), \\ \dfrac{dI(t)}{dt} = \beta S(t)\dfrac{I(t-\sigma)}{P(t)} - \gamma I(t) - \mu I(t), \\ \dfrac{dR(t)}{dt} = \gamma I(t) - \mu R(t), \end{cases} \tag{5.10}$$

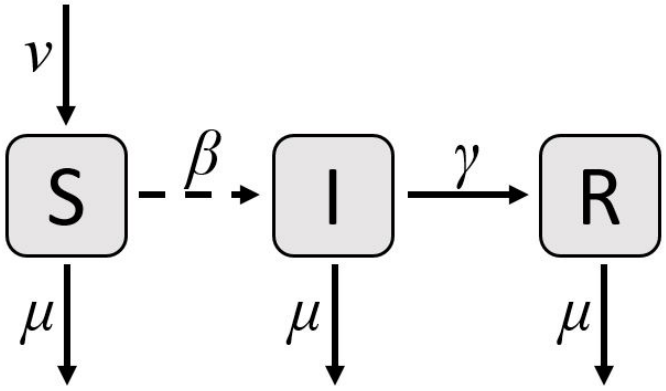


Figura 17 – Modelo compartimental SIR (suscetível-infeccioso-recuperado) com retardo. As setas na horizontal representam a movimentação da população entre os compartimentos, sendo a seta tracejada representando a transição que não é imediata, enquanto que as setas na horizontal representam a demografia.

com $P(t) = S(t) + I(t) + R(t)$.

Aplicamos, então, o programa RKC4 para resolver numericamente modelo (5.10), adotando os mesmos parâmetros do exemplo anterior, conforme Tabela 11, para um período de $t = 100$ dias e $N = 100$. Quanto ao período de incubação da doença, vamos, inicialmente, adotar como valor base $\sigma = 1$. Os resultados são mostrados na Figura 18, enquanto que o Quadro 5.2 mostra o código para ser reproduzido esse mesmo exemplo em MATLAB junto com o algoritmo do método RKC4.

Quadro 5.2 – Código do exemplo com o modelo SIR com retardo conforme Eq. (5.10)

```
t0 = 0; tf = 100; m = 3; d = 1;
b = 0.5; g = 0.1; v = 4e-5; u = v;

f   = @(t,y,yd) [v-b*y(1)*(yd(2)/(y(1)+y(2)+y(3)))-u*y(1),
                 b*y(1)*(yd(2)/(y(1)+y(2)+y(3)))-g*y(2)-u*y(2),
                 g*y(2)-u*y(3)];
phi = @(t) [1e5,1,0];
r   = @(t,y) [1];
```

Vemos nesse exemplo, de forma análoga ao exemplo anterior, as transições $S \to I \to R$ dos indivíduos dessa população. Porém, nesse modelo, observamos algumas diferenças do modelo anterior, sendo a principal a curva da população de infectados (linha tracejada), que além de iniciar mais próxima do 30º dia também apresenta um pico menor, ou seja, esse modelo tem uma menor quantidade de infectados.

Para melhor entendermos os impactos do retardo no nosso modelo vamos realizar agora um comparativo entre os dois modelos. Assim, a Figura 19 apresenta as curvas da

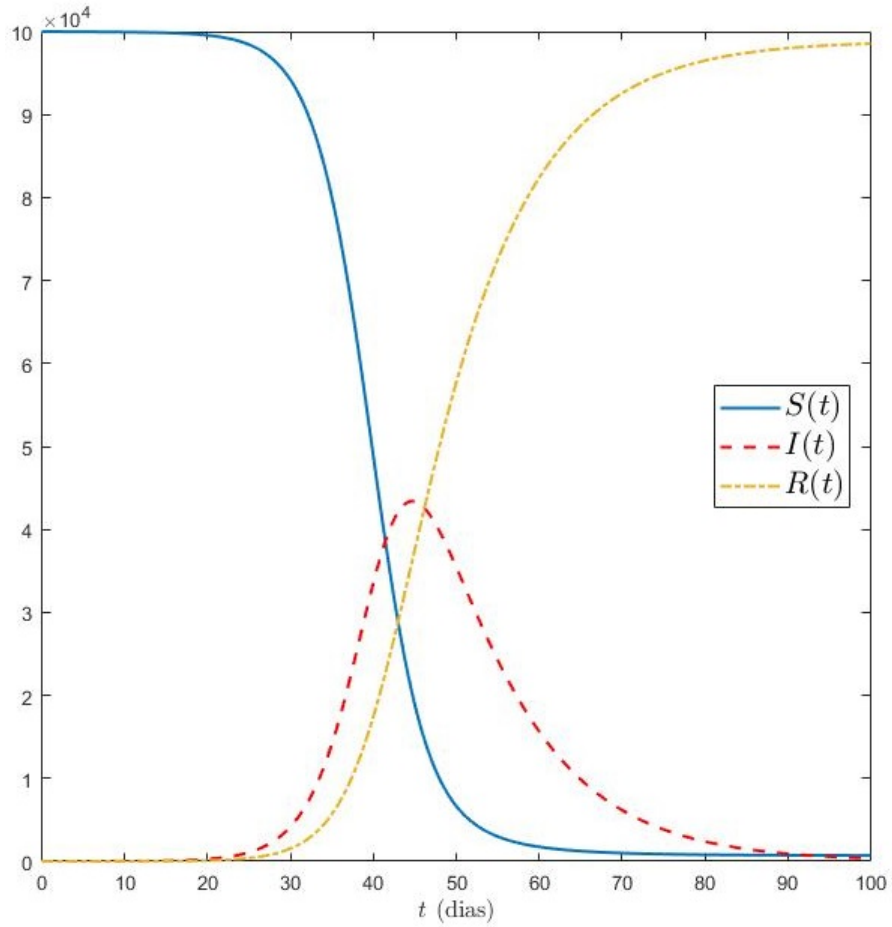


Figura 18 – Gráfico do exemplo do modelo SIR com retardo, conforme Eq. (5.10), sendo os valores dos parâmetros conforme Tabelas 11 e o período de incubação da doença $\sigma = 1$, para o intervalo $[0, 100]$.

população de Infectados para o primeiro modelo SIR, sem os retardos, e para o modelo SIR com o retardo, variando o período de incubação,σ, entre 1 (valor base), 2 e 3 dias. De forma complementar, a Tabela 12 apresenta os valores desse teste para a quantidade máxima de indivíduos infectados , $I(t)_{max}$, e o dia de ocorrência desse pico, t_{pico}.

Tabela 12 – Resultados da comparação da evolução da população de infectados, $I(t)$, para o modelo SIR sem retardo e modelo SIR com retardo para diferentes valores do período de incubação da doença, σ

Incubação	$I(t)_{max}$	t_{pico}
Sem	$4,7749 \times 10^4$	$32,7$ [dias]
$\sigma = 1$ [dias]	$4,3401 \times 10^4$	$44,7$ [dias]
$\sigma = 2$ [dias]	$3,9821 \times 10^4$	$54,8$ [dias]
$\sigma = 3$ [dias]	$3,6829 \times 10^4$	$63,9$ [dias]

Observamos, portanto, que o tempo de incubação da doença gera uma considerável influencia no pico de infectados. Nota-se, por exemplo, uma queda do número máximo de

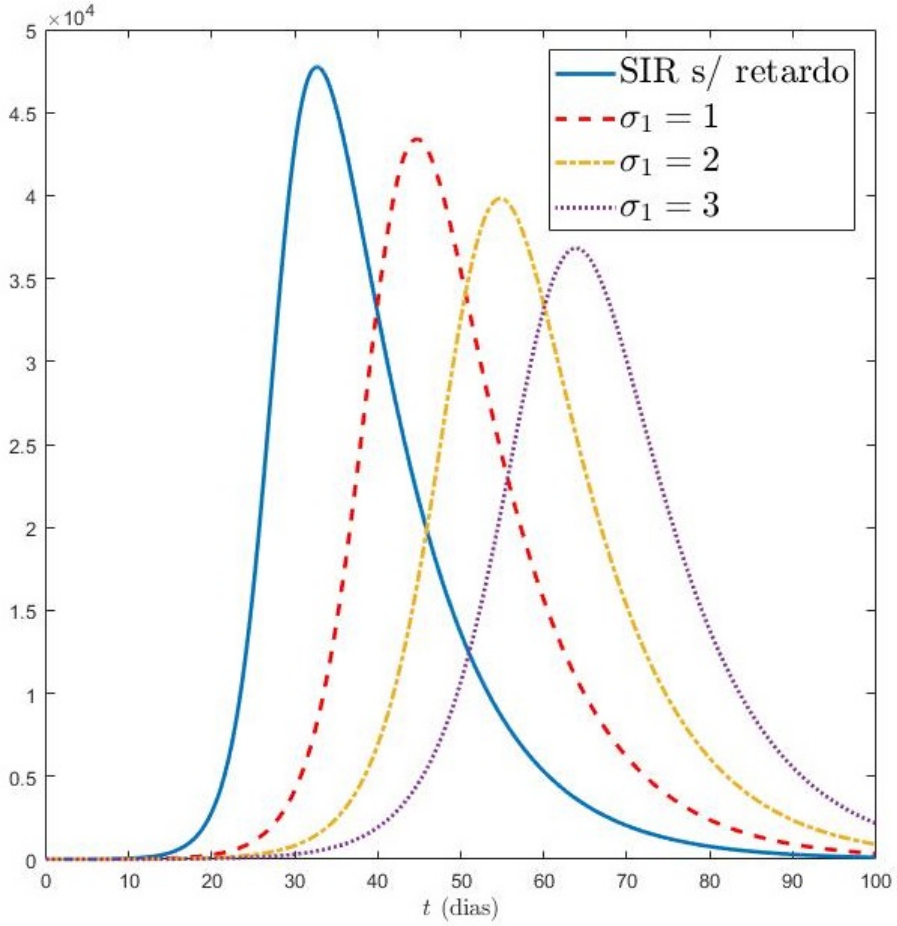


Figura 19 – Comparativo da evolução da população de infectados, $I(t)$, para o modelo SIR sem retardo (linha contínua) e para diferentes períodos de incubação da doença, σ, no modelo SIR com retardo. Os demais parâmetros seguiram as Tabelas 11 e 12.

indivíduos infectados, com uma média de 3.640 indivíduos a menos no número máximo de infectados conforme o aumentamos em um dia o período de incubação da doença. Além disso, nota-se que o aumento do tempo de incubação produz um atraso no pico de infectados, além de uma curva menos acentuada. Ou seja, quanto maior o período de incubação da doença, mais tempo levará para termos o crescimento da população de infectados, e menos acentuado será esse crescimento.

6

Conclusão

Com os estudos teóricos realizados nesse trabalho, foi possível aumentar nossos conhecimentos acerca das equações diferenciais com retardo (EDRs) e, principalmente, dos métodos de Runge-Kutta (RK) e do Runge-Kutta Contínuo (RKC), entendendo que esse último método é bastante adequado à resolução de EDRs.

Como principal resultado dos estudos realizados, tivemos a implementação de um código em MATLAB do método RKC4 que, através de testes numéricos e aplicações em modelos de dinâmica populacional, se mostrou bastante eficiente, pois alcançou aproximações da solução bem precisas em comparação com as respectivas soluções analíticas ou em comparação com a função `ddesd`, nativa do MATLAB. Como vimos, o programa do método RKC4 pode ser aplicado em uma ampla gama de problemas e modelos matemáticos por resolver EDRs dos mais variados tipos, seja com retardo constante, dependendo do tempo ou do estado, ou mesmo sistemas com mais de uma equação e/ou mais de um retardo.

O código implementado ainda foi otimizado e escrito no mesmo formato das funções nativas do MATLAB, onde todas as informações do problema a ser resolvido devem ser dadas por outra função ou programa, não sendo necessário fazer qualquer adaptação ou alteração no algoritmo principal em si. Tivemos a preocupação da escrita do programa nesse formato com o objetivo de que outros pesquisadores possam utilizar o algoritmo aqui implementado em suas pesquisas, onde o método RKC seja apenas uma parte de um outro processo.

Concluímos, dessa forma, que esse trabalho proporcionou o estudo teórico e prático de dois tópicos muito importantes e atuais, EDRs e método RKC, com uma linguagem mais fácil e simples, para entendimento de um maior número de leitores. Além disso, esperamos que o programa desenvolvido possa ser de grande ajuda em outras pesquisas e trabalhos científicos.

Referências

ATKINSON, K.; HAN, W.; STEWART, D. E. *Numerical solution of ordinary differential equations*. [S.l.]: John Wiley & Sons, 2009.

BELLEN, A.; VERMIGLIO, R. Some applicatons of continuous runge-kutta methods. *Elsevier - Applied Numerical Mathematics*, n. 22, p. 63–80, 1996.

BELLEN, A.; ZENNARO, M. *Numerical Methods for Delay Differential Equations*. [S.l.]: Clarendon Press, Oxford, 2003.

BOYCE, W. E.; DIPRIMA, R. C. *Equações Diferenciais Elementares e Problemas de Valor de Contorno*. [S.l.]: Rio de Janeiro: LTC., 2015.

BURDEN, R. L.; FAIRES, J. D. *Análise numérica*. [S.l.]: Cengage Learning, 2008.

BUTCHER, J. C. The non-existence of ten stage eighth order explicit runge-kutta methods. *BIT Numerical Mathematics*, Springer, v. 25, n. 3, p. 521–540, 1985.

CHAPRA, S. C.; CANALE, R. P. *Numerical Methods for Engineers*. [S.l.]: McGraw-Hill Education; 7ª Edição, 2015.

Edelstein-Keshet, L. *Mathematical models in Biology*. [S.l.]: Society for Industrial and Applied Mathematics, 2005.

ELHIA, M.; RACHIK, M.; BENLAHMAR, E. Optimal control of an sir model with delay in state and control variables. *Biomathematics*, p. 1–7, 2013.

FUKUSHIMA, P. K. *Problemas inversos associados a equações diferenciais impulsivas*. 2018.

GEAR, C. W. *Numerical initial value problems in ordinary differential equations*. [S.l.]: Prentice Hall PTR, 1971.

HADAMARD, J. Sur les problèmes aux dérivées partielles et leur signification physique. *Princeton University Bulletin*, p. 49–52, 1902.

HARTUNG, F. et al. *Chapter 5 Functional Differential Equations with State-Dependent Delays: Theory and Applications*. [S.l.]: Handbook of Differential Equations: Ordinary Differential Equations, Vol. 3, p. 435-545., 2006.

HAYASHI, H. *Numerical Solution of Retarded and Neutral Delay Differential Equations using Continuous Runge-Kutta Methods*. Tese (Doutorado) — Universidade de Toronto, 1996.

Referências

KADDAR, A.; ABTA, A.; ALAOUI, H. T. A comparison of delayed sir and seir epidemic models. *Nonlinear Analysis: Modeling and Control*, n. 16, p. 181–190, 2011.

KEELING, M. J.; ROHANI, P. *Modeling Infectious Diseases in Humans and Animals*. [S.l.]: Princeton University Press, 2008.

MENIN, O. H.; BAUCH, C. T. Solving the patient zero inverse problem by using generalized simulated annealing. *Elsevier - Physica A*, v. 490, p. 1513–1521, 2018.

MURRAY, J. D. *Mathematical Biology: I. An Introduction*. [S.l.]: Springer-Verlag; 3ª Edição, 2002.

PAUL, C. A. H. A test set of functional differential equations. *Numerical Analysis Report*, Manchester Centre for Computational Mathematics - University of Manchester/UMIST, v. 243, 1994.

SHAMPINE, L. F. Solving odes and ddes with residual control. *Applied Numerical Mathematics*, v. 52, p. 113–127, 2005.

Apêndices

A

Código do algoritmo do método RKC4

```
function [t,w,coef] = rkc4(N)

%%%%%%%%%%%%%%%%%% Chama o problema a ser resolvido %%%%%%%%%%%%%%%%%%
[t0,tf,f,phi,r,n,d] = exemplo_PVI;
%%%%%%%%%% Variaveis do metodo de Runge-Kutta de 4a ordem %%%%%%%%%%

h = (tf - t0)/N;           % Define o tamanho do passo
t = t0:h:tf;               % Armazena os pontos da malha em um vetor
w = zeros(length(t),n);    % Cria vetor p/ armazenar solucao numerica
w(1,:) = phi(t0);          % Armazena o valor inicial
k = zeros(4,n);            % Cria vetor p/ armazenar os k's
coef = zeros(length(t),3,n); % Cria vetor p/ armazenar coeficientes
                             % do polinomio interpolador de Hermite

%%%%%%%%%%%%%%%%%%%%%%%% Tableau de Butcher %%%%%%%%%%%%%%%%%%%%%%%%%

a = [0,0,0,0;1/2,0,0,0;0,1/2,0,0;0,0,1,0]; % Matriz A p/ RKC4
c = [0,1/2,1/2,1];      % Vetor c p/ RKC4
b1 = [2/3, -3/2, 1];    % b1 = (2/3)*theta^3 -(3/2)*theta^2 +theta
b2 = [-2/3, 1, 0];      % b2 = -(2/3)*theta^3 +theta^2
b3 = [-2/3, 1, 0];      % b3 = -(2/3)*theta^3 +theta^2
b4 = [2/3, -1/2, 0];    % b4 = (2/3)*theta^3 -(1/2)*theta^2

%%%%%%%%%%%%%%%% Inicio do Metodo de Runge-Kutta Continuo %%%%%%%%%%%%
for i = 1:N     % Looping Principal, percorre os pontos da malha

    %%%%%%% Obtem os k's do metodo de Runge-Kutta no passo i %%%%%%
    for j = 1:4

    T = t(i)+h*c(j);                  % Calcula primeiro argumento da f
    W = w(i,:)+h*sum(a(j,:).'.*k,1); % Calcula segundo argumento da f
    Td = T - r(T,W);     % Calcula os argumentos de retardo
```

```
    Wd = rkc4interp(t,h,w,Td,phi,coef,n,d); % Calcula terceiro
                                            % argumento da f
    k(j,:) = f(T, W, Wd); % Calcula k
    end

    %%%%%%%%%%%%%%%% Concatena os Coeficiente %%%%%%%%%%%%%%%%%%%%%%
     for g = 1:n
    coef(i,:,g)=b1.*k(1,g)+b2.*k(2,g)+b3.*k(3,g)+b4.*k(4,g);

    %%%%%% Calcula a aproximacao p/ a solucao y(t(i+1)) %%%%%%%%%%
    w(i+1,g) = w(i,g) + ...
       h*(coef(i,1,g)*(1^3)+coef(i,2,g)*(1^2)+coef(i,3,g)*(1));
    end
end                  % Fim do Looping Principal
end                      %%% Fim da Funcao %%%

%%%%%%%%%%%%%%%%%%%%%% Funcao rck4interp %%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%% Calcula a solucao aproximada no ponto com retardo  %%%%%%%%
function [Wd] = rkc4interp(t,h,w,Td,phi,coef,n,d)
Wd = zeros(n,d);
for q = 1:d
  if Td(q) ≤ t(1)  % Verifica se argumento de retardo < t_0
     Wd(:,q) = phi(Td(q)); % Sim = calcula usando a funcao historia
  else                     % Nao = faz Interpolacao cubica de Hermite
      k = find(t≤Td(q),1,'last'); % Procura [t_k,t_{k+1}] c/ o ponto Td
      theta = (Td(q)-t(k))/h;  %  Mapea Td em [0,1]
    for p = 1:n
        Wd(p,q) = w(k,p) + h*(coef(k,1,p)*(theta^3) + ...
            coef(k,2,p)*(theta^2) + coef(k,3,p)*theta);
    end
  end
end
end
```