Universidade Federal Rural do Semi-árido
Revista Eletrônica de Engenharia Elétrica e Engenharia Mecânica
ISSN 2763-5325 | DOI: https://doi.org/10.21708/issn27635325

Artigo

Análise comparativa entre métodos numéricos e analíticos em circuitos transitórios

Luiz Henrique Rodrigues Farias[1], Paulo Cesar Linhares Da Silva[2], Antônia Jocivania Pinheiro[3]

[1] Universidade Federal Rural do Semi-Árido; luizhenrique272725@gmail.com
[2] Universidade Federal Rural do Semi-Árido; linhares@ufersa.edu.br
[3] Universidade Federal Rural do Semi-Árido; vaniamat@ufersa.edu.br

Recebido: 24/03/2022;
Aceito: 12/06/2022;
Publicado: 30/06/2022.

Resumo: Os métodos numéricos e analíticos são de suma importância na análise e modelagem de problemas na engenharia. Diante deste contexto, este trabalho visa realizar uma metodologia de estudo e ensino de circuitos elétricos clássicos. Tal metodologia baseia-se em um estudo analítico e numérico das equações diferenciais que envolvem os circuitos. A programação utilizada é feita na linguagem C/C++. A equação diferencial é resolvida de maneira analítica e comparada com as soluções numéricas do método de Runge Kutta de 4ª ordem (RK 4ª) e o método das séries de potência. Foi constatado ao final das soluções um erro percentual pequeno, o que mostra a precisão e aplicabilidade dos métodos numéricos empregados.

Palavras-chave: Runge Kutta; Série de Potência; Circuitos Elétricos.

Abstract: Numerical and analytical methods are of paramount importance in the analysis and modeling of engineering problems. Given this context, this work aims to carry out a methodology for the study and teaching of classical electrical circuits. Such methodology is based on an analytical and numerical study of the differential equations that involve the circuits. The programming used is done in the C/C++ language. The differential equation is solved analytically and compared with the numerical solutions of the 4th order Runge Kutta method (RK 4th) and the power series method. A small percentage error was found at the end of the solutions, which shows the precision and applicability of the numerical methods used.

Key-words: Runge Kutta; Power Series; Electric circuits.

1. INTRODUÇÃO

No intuito de solucionar problemas que surgem no campo científico e tecnológico, foram desenvolvidos métodos matemáticos que utilizam operações matemáticas, lógicas e relacionais, além de teoremas formulados por grandes nomes da ciência. Como por exemplo, o Teorema de Pitágoras elaborado por Pitágoras e utilizado atualmente em diversas aplicações, Teorema de Pascal usado em números binomiais, elaborado por Blaise Pascal, o Teorema Fundamental do Cálculo elaborado por Isaac Barrow sendo base para as operações diferenciais e integrais do cálculo, dentre vários outros. Porém, com o desenvolvimento das novas tecnologias e a possibilidade de execução de uma quantidade enorme de operações em um pequeno intervalo de tempo, nunca visto anteriormente, verificou-se que em alguns casos a utilização de métodos e algoritmos matemáticos complexos podem ser substituídos a fim de utilizar das ferramentas tecnológicas que alavancam as análises enquanto continuam a suprir os requerimentos das soluções, sendo uma delas a rapidez da disponibilização dos dados em tempo real, no qual pequenas variações podem ser explicitadas dependendo do grau de suas relevâncias.

Logo, adaptando as necessidades dos problemas e auxiliando para uma maior eficiência, surgem os métodos numéricos que utilizam as ferramentas tecnológicas que estão a se desenvolver, dispersando a inovação tecnológica nas mais diversas áreas (elétrica, mecânica, química, entre outros) [1]. Generalizando os benefícios da utilização dos métodos numéricos podemos destacar a rapidez da disponibilização dos dados em tempo real, como mencionado anteriormente, e no alto nível de precisão, no qual é necessário dar ênfase, pois esta característica permite a adaptação das soluções para o caso abordado e suas finalidades de uso, minimizando os erros através do aumento da quantidade de iterações que o procedimento numérico executa, e/ou por uma modelagem mais adequada do problema.

O estudo da eletricidade e magnetismo, ou eletromagnetismo, possui um enorme peso na construção

tecnológica e social na atualidade [2], proporcionando o desenvolvimento de redes de transmissão de energia entre longas distancias que possibilitaram a produção de energia elétrica e abastecimento para a população, o desenvolvimento de eletroeletrônicos que auxiliam no dia-a-dia da população e o desenvolvimento da tecnologia de informação (T.I.) para a disponibilização de informação e comunicação. Porém, o conhecimento necessário para a disponibilização dessas funcionalidades empregadas no nosso cotidiano é complexo, proporcionado por pesquisas extensas que percorreram várias gerações. No entanto, estudos de circuitos transitórios fundamentais, que modificam de um estado inicial para o final em um período de transição possuindo um arranjo simples, como os circuitos resistor-capacitor (RC), resistor-indutor (RL), resistor-capacitor-indutor (RLC), facilitaram o entendimento do comportamento elétrico e magnético nas variações de tensões, correntes, cargas, auxiliando na compreensão do eletromagnetismo – sendo utilizados também na docência em livros didáticos [2].

Considerando os pontos mencionados anteriormente, este trabalho abordará as aplicações dos métodos numéricos de Runge-Kutta de 4ª ordem [1] e séries de potências [3], que será explicado posteriormente, no intuito de analisar as soluções obtidas para estabelecer o comportamento de alguns componentes nos circuitos transitórios, como corrente, tensões, e comparar, relacionar, com soluções analíticas vindas de métodos matemáticos, verificando a confiabilidade destes métodos numéricos para tais casos.

2. DESENVOLVIMENTO

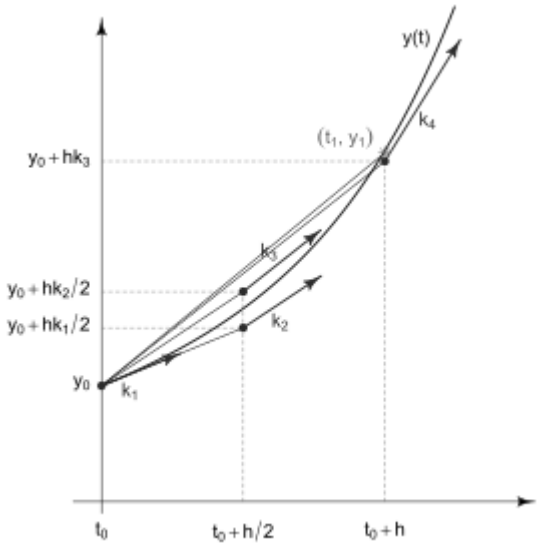
Nesta seção é abordado os métodos analíticos e numéricos utilizados para o desenvolvimento do trabalho. Destaca-se o uso do método de Runge Kutta de 4ª ordem com melhores resultados numéricos obtidos e também o método das séries de potência que foi aplicado para o estudo comportamental da busca das soluções das equações diferenciais que envolvem os circuitos RC, RLC e LC.

As soluções alcançadas visam representar o comportamento de alguns elementos de circuitos no período **transitório** – ou seja, em que o tempo de duração é limitado ou pouco; que é passageiro; que só dura um certo tempo -, no qual os valores de correntes, tensões, variam de condições iniciais para finais ao alcançar o chamado estado estacionário, deixando o sistema em equilíbrio. Uma das principais motivações para aplicação de métodos numéricos se dá devido ao curto intervalo de tempo, na maioria das vezes, em que ocorre o período transitório, trazendo complicações para medições em tempo real e criando a necessidade de estimativas precisas.

2.1 Método de runge-kutta de 4ª ordem

A ideia dos métodos de Runge-Kutta, é obter numericamente a solução de Equações Diferenciais Ordinárias (EDO) levando em consideração o cálculo das derivadas (inclinação da reta tangente) de uma função $\frac{dy}{dt} = f(t, y)$, a partir de um ponto inicial $P_0 = (t_0, y_0)$.

Figura 1 – Visão Geométrica do Método de Runge Kutta de 4ª Ordem.



Fonte: Adaptado de Thaline G. Evangelista, 2018.

Esse método também se baseia na ideia da aproximação linear utilizando a primeira derivada, de acordo com a Equação 1:

$$y_{k+1} = y_k + \frac{dy}{dt}h\,, \tag{1}$$

em que $h = t_{k+1} - t_k$ é o passo e $\frac{dy}{dt}$ é a função que representa a primeira derivada.

O método de Runge-Kutta de 4ªordem é feito de maneira recursiva no intervalo em que a função está definida para uma quantidade limitada de iterações, segundo a equação

$$y_{k+1} = y_k \; + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4)\,, \tag{2}$$

em que K_1, K_2, K_3 e K_4 serão calculados segundo as Equações (3)-(6) para cada ponto projetado $P_k = (t_k, y_k)$, para $k = (0, 1, 2, 3, \dots, n)$:

$$\mathrm{K}_1 = f(t_0, y_0), \tag{3}$$

$$\mathrm{K}_2 = f(t_0 \; + \; h/2\,, y_0 \; + K_1 h/2), \tag{4}$$

$$\mathrm{K}_3 = f(t_0 \; + \; h/2\,, y_0 \; + K_2 h/2), \tag{5}$$

$$\mathrm{K}_4 = f(t_0 \; + \; h\,, y_0 \; + K_3 h). \tag{6}$$

Nesta proposta de trabalho foi utilizado o método de Runge-Kutta de 4ª ordem com as devidas condições de contorno apropriadas para o analisar o comportamento da solução das equações diferenciais ordinárias que regem os circuitos RC, RL e RLC.

2.2 Método de séries de potências

O método das séries de potência baseia-se na hipótese de que as soluções de uma equação diferencial são analíticas em alguma vizinhança de um ponto ordinário $P_0 = (t_0, y_0)$, onde são dadas as condições de contorno. Deste modo, a função possui derivadas de todas as ordens neste ponto, portanto, podem ser expressas como uma série de potências. Este método busca a solução de uma equação diferencial no formato de série de potências (Eq. 7) com os auxílios de suas derivadas dado pelas Equações (8)-(10).

$$y = \sum_{n=0}^{\infty} a_n t^n, \tag{7}$$

$$y' = \sum_{n=1}^{\infty} \mathrm{n} a_n t^{n-1}\,, \tag{8}$$

$$y'' = \sum_{n=2}^{\infty} \mathrm{n(n-1)} a_n t^{n-2}, \tag{9}$$

$$y^{(k)} = \sum_{n=k}^{\infty} \mathrm{n(n-1)\dots(n-k+1)} a_n t^{n-k}. \tag{10}$$

Ao computar os primeiros termos da série e utilizando as condições de contorno Eq. (11) – (12),

$$y(0) \; = y_0 = a_0, \tag{11}$$

$$\frac{dy(0)}{dt} = y'_0 \; = \mathrm{a}_1, \tag{12}$$

é possível encontrar os coeficientes (a_n) para uma EDO fazendo as condições de contorno, dadas em cada problema específico.

Após uma aplicação do método de séries de potências [3] nas EDOs que representam alguns dos comportamentos dos circuitos RC, RL e RLC, foram obtidas as soluções numéricas referentes à tensão no capacitor (Vc) no circuito RC em carga (Eq. 13), a corrente no indutor (i_L) no circuito RL em carga (Eq. 14) e a corrente (i_{RLC}) no circuito RLC em descarga (Eq. 15). A resistência, indutância e capacitância foram representadas nas equações a seguir, respectivamente, como R, L e C.

 DOI: https://doi.org/10.21708/issn27635325.v4n1.a11183.2022

$$\begin{cases} a_{n+1} = \frac{1}{n+1}\left(\frac{\varepsilon}{R} - \frac{a_n}{RC}\right) \\ V_C = \sum_{n=0}^{\infty} \frac{a_n t^n}{C} \end{cases}, \tag{13}$$

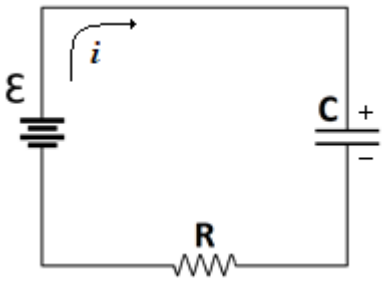
$$\begin{cases} a_{n+1} = \frac{1}{n+1}\left(\frac{\varepsilon - a_n R}{L}\right) \\ i_L = \sum_{n=0}^{\infty} a_n t^n \end{cases}, \tag{14}$$

$$\begin{cases} a_{n+2} = -\frac{1}{n+2}\left(\frac{(n+1)R}{L} - \frac{a_n}{LC(n+1)}\right) \\ i_{RLC} = \sum_{n=0}^{\infty} a_n t^n \end{cases}. \tag{15}$$

2.3 Circuito rc

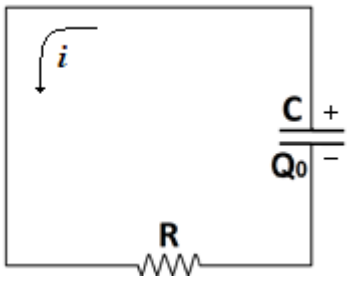
O circuito RC possui um arranjo simples composto por um resistor (R) e um capacitor (C) conectados em série. O capacitor é um componente composto por duas placas condutoras separadas por um isolante, ou imerso no vácuo [2] no qual a carga líquida ao estar em equilíbrio é igual a zero, devido a possuir cargas de módulos iguais em cada placa, no entanto, sinais opostos. No período transitório, ao transferir elétrons, as cargas são acumuladas, se o estado transitório estiver em carga (Figura 2), ou são dispersadas, se estiver em descarga (Figura 3), até a condição de equilíbrio (carga líquida igual a zero). A funcionalidade mais comum aplicada ao capacitor é acumular cargas elétricas, devido a isso, varia a corrente elétrica (fluxo de cargas) e, consequentemente, as diferenças de potencias (ddp's).

Figura 2 – Circuito RC e fonte (ε).



Fonte: Autor, 2021.

Figura 3 – Circuito RC.



Fonte: Autor, 2021.

Para obter a equação diferencial que demonstra o comportamento dos elementos é necessário utilizar da Lei de Kirchhof (16), a relação entre *carga* (Q), diferença de *potencial do capacitor* (V) e sua *capacitância* (C) se dá através da Equação 17 e sua energia elétrica armazenada (U_E) pode ser obtida pela Equação 18.

$$\sum V = 0, \tag{16}$$

$$Q = CV, \tag{17}$$

$$U_E = \frac{1}{2} CV^2. \tag{18}$$

Aplicando a Lei de Kirchhoff na Figura 2, sendo a ddp do capacitor em carga V_{CC}, a ddp do resistor V_R, é obtida a Equação 18.

$$\varepsilon - V_{CC} - V_R = 0. \tag{19}$$

Substituindo V_{CC} por $\frac{Q}{C}$, V_R por $R\frac{dQ}{dt}$ e isolando $\frac{dQ}{dt}$, tem-se

$$\frac{dQ}{dt} = \frac{\varepsilon}{R} - \frac{Q}{RC}. \tag{20}$$

 DOI: https://doi.org/10.21708/issn27635325.v4n1.a11183.2022

A Equação 20 pode ser resolvida pelo método de separação de variáveis, ou aplicando método numérico. A solução analítica provinda da separação de variáveis é

$$Q(t) = \mathcal{E}C\left(1 - e^{-\frac{t}{RC}}\right). \tag{21}$$

Sendo $i = \frac{dQ}{dt}$, tem-se

$$i(t) = \frac{\mathcal{E}}{R}e^{-\frac{t}{RC}}. \tag{22}$$

Substituindo $V_R = iR = \mathcal{E}e^{-\frac{t}{RC}}$ na Equação 19 e isolando V_{CC}, é obtida a Equação 23,

$$V_{CC}(t) = \mathcal{E}\left(1 - e^{-\frac{t}{RC}}\right). \tag{23}$$

Deste modo, as soluções analíticas para o circuito RC em carga, são as Equações 24, 25, 26:

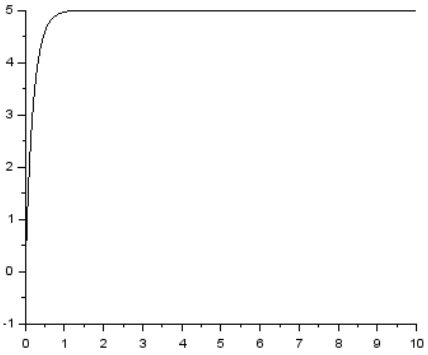
$$Q(t) = Q_F\left(1 - e^{-\frac{t}{RC}}\right) \quad .: \; Q_F = \mathcal{E}C, \tag{24}$$

$$i(t) = I_0 e^{-\frac{t}{RC}} .: I_0 = \frac{\mathcal{E}}{R}, \tag{25}$$

$$V_{CC}(t) = \mathcal{E}\left(1 - e^{-\frac{t}{RC}}\right). \tag{26}$$

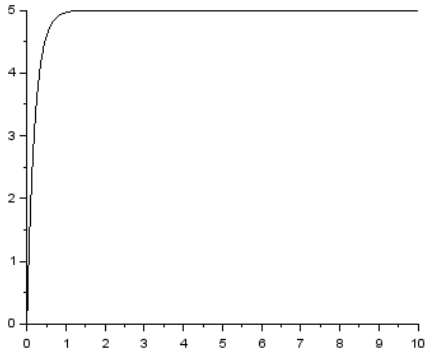
Para visualização e comparação dos resultados obtidos pelo método analítico e os métodos numéricos empregados, foram plotados nas Figuras 4 (a), 4 (b) e 4 (c).

Figura 4 (a) – Solução analítica para Eq. 26.



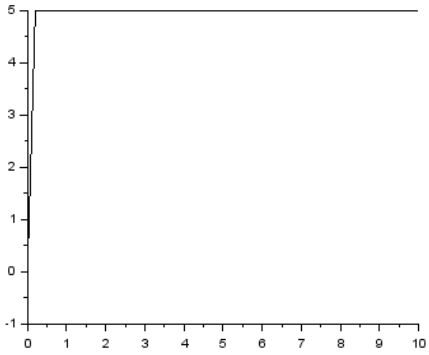
Fonte: Scilab 6.0.2, 2021.

Figura 4 (b) – Solução com Runge-Kutta para Eq. 20.



Fonte: Scilab 6.0.2, 2021.

Figura 4 (c) – Solução com séries de potências para Eq. 20.



Fonte: Scilab 6.0.2, 2021.

Parâmetros utilizados: fonte (ℰ) de 5V; capacitância (C) de 2mF; resistência (R) de 100Ω; e foram calculados 5000 pontos no código em C++ para ambos os métodos.

Tabela 1 – Tensão no capacitor no circuito RC em carga.

	Tensão (V)				
Tempo (s)	**Medida Analítica**	**Runge-Kutta 4ª ord.**	**Séries de Potências**	**Erro Percentual (%) – RK 4ª**	**Erro Percentual (%) – Séries de Potências**
0,02	0,475813	0,475813	0,500000	0,000000	5,083387
0,04	0,906346	0,906346	1,000000	0,000000	10,333178
0,06	1,295908	1,295909	1,500000	7,716597e-7	15,748944
0,08	1,648399	1,648400	2,000000	6,066492e-7	21,329858
0,10	1,967346	1,967347	2,500000	5,082990e-7	27,074770

Fonte: Autor, 2021.

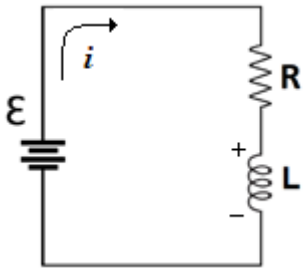
 DOI: https://doi.org/10.21708/issn27635325.v4n1.a11183.2022

De acordo com a Tabela 1, o método de Runge Kutta de 4ª ordem apresentou soluções com melhores aproximações em relação as soluções analíticas se comparado com o método de série de potências com erros percentuais inferiores a 10^{-6}.

2.4 Circuito RL

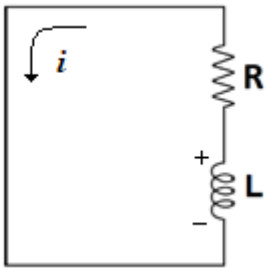
De forma semelhante, o arranjo do circuito RL é composto por um resistor (R) e um indutor (L) conectados em série. O objetivo de um indutor é fornecer uma corrente que contrarie a variação de corrente no circuito [2], logo, sua aplicação em um circuito auxilia a manter a corrente constante. Diferente do capacitor, o indutor armazena carga magnética, porém, também possui períodos transitórios de carga e descarga, demonstrados respectivamente nas Figuras 5 e 6.

Figura 5 – Circuito RL e fonte (ε).



Fonte: Autor, 2021.

Figura 6 – Circuito RL.



Fonte: Autor, 2021.

A relação entre a *taxa de variação da corrente* (di/dt), diferença de *potencial do indutor* (V_L) e *indutância* (L) se dá através da Equação 27, e a energia magnética armazenada (U_M) pelo indutor (solenoide) com a corrente (I) pode ser obtida pela Equação 28.

$$V_L = L\frac{di}{dt}, \tag{27}$$

$$U_M = \frac{1}{2}LI^2. \tag{28}$$

Aplicando a Lei de Kirchhoff (Eq. 16) para a Figura 5, circuito RL em carga, podemos obter a equação 29:

$$\varepsilon - L\frac{di}{dt} - iR = 0. \tag{29}$$

Isolando a taxa de variação da corrente, tem-se

$$\frac{di}{dt} = \frac{\varepsilon - iR}{L}. \tag{30}$$

Utilizando o método de separação de variáveis na Equação 30, é obtida a solução analítica a seguir:

$$i(t) = \frac{\varepsilon}{R}\left(1 - e^{-\left(\frac{R}{L}\right)t}\right). \tag{31}$$

A taxa de variação da corrente pode ser dada desta forma por $\frac{d}{dt}[i(t)]$, derivada da Equação 31, sendo

$$\frac{di}{dt} = \frac{\varepsilon}{L}e^{-\left(\frac{R}{L}\right)t}. \tag{32}$$

Ao substituir na Equação 27, será obtida a ddp do indutor, Equação 33.

$$V_L = \varepsilon e^{-\left(\frac{R}{L}\right)t}. \tag{33}$$

Deste modo, as soluções analíticas para o circuito RL em carga, são as Equações 34, 35, 36:

 DOI: https://doi.org/10.21708/issn27635325.v4n1.a11183.2022

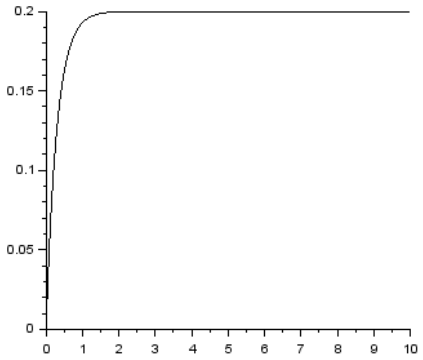
$$i(t) = \frac{\varepsilon}{R}\left(1 - e^{-\left(\frac{R}{L}\right)t}\right), \tag{34}$$

$$\frac{di}{dt} = \frac{\varepsilon}{L} e^{-\left(\frac{R}{L}\right)t}, \tag{35}$$

$$V_L = \varepsilon e^{-\left(\frac{R}{L}\right)t}. \tag{36}$$

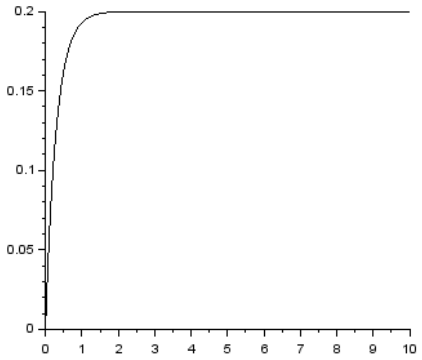
Para visualização e comparação dos resultados obtidos pelo método analítico e os métodos numéricos, foram plotados nas Figuras 7 (a), 7 (b) e 7 (c).

Figura 7 (a) – Solução analítica para Eq. 34.



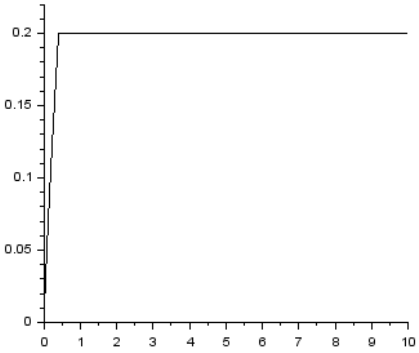
Fonte: Scilab 6.0.2, 2021.

Figura 7 (b) – Solução com Runge-Kutta para Eq. 30.



Fonte: Scilab 6.0.2, 2021.

Figura 7 (c) - Solução com série de potências para Eq. 30.



Fonte: Scilab 6.0.2, 2021.

Parâmetros utilizados: fonte (ε) de 20V; indutância (L) de 30H; resistência (R) de 100Ω; e foram calculados 5000 pontos no código em C++ para ambos os métodos.

Tabela 2 – Corrente no circuito RL em carga.

	Corrente (A)				
Tempo (s)	**Medida Analítica**	**Runge-Kutta 4ª ord.**	**Séries de Potências**	**Erro Percentual (%) – RK 4ª**	**Erro Percentual (%) – Séries de Potências**
0,02	0,012899	0,012899	0,013031	0,000000	1,024939
0,04	0,024965	0,024965	0,025490	0,000000	2,101334
0,06	0,036254	0,036254	0,037426	0,000000	3,232051
0,08	0,046814	0,046814	0,048884	0,000000	4,420089
0,10	0,056694	0,056694	0,059907	0,000000	5,668666

Fonte: Autor, 2021.

De acordo com a Tabela 2 e semelhante aos dados apresentados na Tabela 1 referente ao circuito RC, o método de Runge Kutta de 4ª ordem apresentou soluções com melhores aproximações em relação as soluções analíticas se comparado com o método de série de potências com erros percentuais inferiores a 10^{-6}.

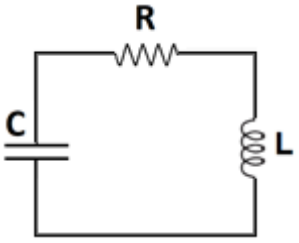
2.5 Circuito RLC amortecido

O arranjo do circuito RLC amortecido é composto por um resistor (R), um indutor (L) e um capacitor (C) conectados em série, como mostrado na Figura 8. O capacitor será responsável por acumular energia elétrica e transferir pelo circuito em forma de corrente, tendo seu comportamento de variação restringido pelo indutor, não ocorrendo de forma abrupta, além de ser convertida em parte a energia magnética e a restante dispersada em forma de calor pelo resistor. Sendo este processo repetido de forma análoga do indutor para o capacitor, e repetido continuamente infinitas vezes, convergindo os elementos do circuito a um estado estacionário, ou seja, a valores constantes.

Em diversas ocasiões, o estudo do circuito RLC tem como objetivo demonstrar seu comportamento de

amortecimento, trazendo uma maior suavidade as inclinações da função *Corrente x Tempo*.

Figura 8 – Circuito RLC.



Fonte: Autor, 2021.

Como ferramentas para auxiliar as deduções das equações diferenciais que representam o comportamento dos elementos do circuito RLC, temos a Lei de Kirchhoff (Eq. 37), baseada na Lei de Conservação de Energia, e as equações que representam a diferença de potencial no indutor (Eq. 38), do resistor (Eq. 39) e a do capacitor (Eq. 40).

$$\sum V = 0, \tag{37}$$

$$V_L = L\frac{di(t)}{dt}, \tag{38}$$

$$V_R = i(t)R, \tag{39}$$

$$V_C = \frac{1}{C}\int i(t)dt. \tag{40}$$

Aplicando a Lei de Kirchhoff para a Figura 8, circuito RLC em descarga, pode-se obter a Equação 41.

$$V_L + V_R + V_C = 0. \tag{41}$$

Substituindo as tensões pelas Equações 38, 39 e 40.

$$L\frac{di(t)}{dt} + i(t)R + \frac{1}{C}\int i(t)dt = 0. \tag{42}$$

Derivando novamente por t e depois multiplicando por $\frac{1}{L}$:

$$\frac{d^2i}{d^2t} + \frac{R}{L}\frac{di}{dt} + \frac{1}{LC}i = 0. \tag{43}$$

Abordando através do método de EDO homogênea com coeficientes constantes [3], é possível adquirir as soluções associando funções do tipo e^{st}. Com,

$$\begin{cases} s_1 = -\alpha + \sqrt{\alpha^2 - {\omega_o}^2} \\ s_2 = -\alpha - \sqrt{\alpha^2 - {\omega_o}^2} \end{cases}, \; onde\; \alpha = \frac{R}{2L}\; e\; \omega_o = \frac{1}{\sqrt{LC}}. \tag{44}$$

Logo, existem três casos diferentes, sendo

- Super amortecido ($\alpha^2 > {\omega_o}^2$)

$$i(t) = C_1 e^{s_1 t} + C_2 e^{s_2 t}. \tag{45}$$

- Criticamente amortecido ($\alpha^2 = {\omega_o}^2$)

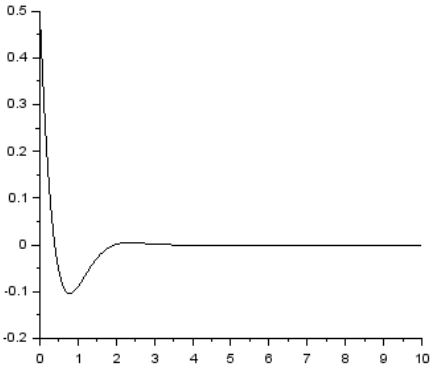
$$i(t) = C_1 e^{-\alpha t} + C_2 t e^{-\alpha t}. \tag{46}$$

- Subamortecido ($\alpha^2 < {\omega_o}^2$)

$$i(t) = C_1 e^{-\alpha t}\cos(\omega t) + C_2 e^{-\alpha t}\sin(\omega t) .: \omega = \sqrt{{\omega_o}^2 - \alpha^2}. \tag{47}$$

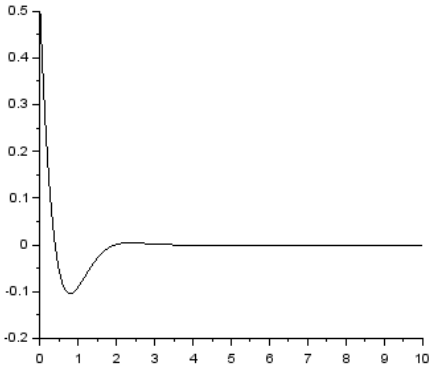
Para visualizar a comparação entre os métodos numéricos e analíticos aplicados observe as Figuras 9 (a), 9 (b) e 9 (c).

Figura 9 (a) – Solução analítica para Eq. 43.



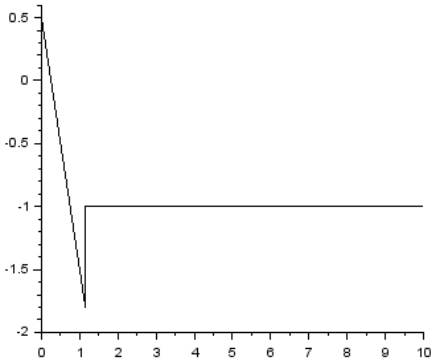
Fonte: Scilab 6.0.2, 2021.

Figura 9 (b) – Solução com Runge-kutta para Eq. 43.



Fonte: Scilab 6.0.2, 2021.

Figura 9 (c) – Solução com série de potências para Eq. 43.



Fonte: Scilab 6.0.2, 2021.

Parâmetros utilizados: corrente inicial (I) de 0,5A; indutância (L) de 2,5H; resistência (R) de 10Ω; capacitância 50mF; e foram calculados 5000 pontos no código em C++ para ambos os métodos.

Tabela 3 – Corrente no circuito RLC sub-amortecido.

	Corrente (A)				
Tempo (s)	**Medida Analítica**	**Runge-Kutta 4ª ord.**	**Séries de Potências**	**Erro Percentual (%) – RK 4ª**	**Erro Percentual (%) – Séries de Potências**
0,02	0,460800	0,460800	0,460000	0,000003	0,173568
0,04	0,423197	0,423197	0,420000	0,000005	0,755374
0,06	0,387184	0,387184	0,380000	0,000008	1,855338
0,08	0,352749	0,352749	0,340000	0,000011	3,614137
0,10	0,319877	0,319877	0,300000	0,000013	6,213957

Fonte: Autor, 2021.

De acordo com a Tabela 3, nota-se que houve um aumento do erro percentual das soluções alcançadas pelo método de Runge Kutta de 4ª ordem com os resultados anteriores, porém, este método ainda assim apresentou soluções mais próximas as analíticas em comparação ao método de Séries de Potências.

3. CONCLUSÃO

Mediante as simulações feitas, notou-se que o método Runge-Kutta de 4ª ordem apresentou eficiência superior ao método das series de potências na simulação de elementos dos circuitos RC e RL devido ao erro percentual inferior a 10^{-6}, como demostrado nas Tabela 1 e 2, obtendo assim uma maior credibilidade na sua utilização para análise de casos semelhantes destes circuitos. Lembrando, que de forma análoga, este método também pode ser aplicado na análise destes circuitos em estados transitórios de descarga.

Os resultados obtidos a partir das aplicações dos métodos numéricos no circuito RLC amortecido apresentaram maiores erros percentuais, se comparado as aplicações nos circuitos anteriores, em relação a solução analítica, no entanto, o método numérico RK 4ª ainda se torna viável, visto que seu erro percentual máximo foi inferior a 10^{-3}.

O método de RK 4ª obteve os melhores resultados em todos os casos apresentados neste estudo, entretanto, o método de séries de potências demonstrou soluções que condizem com o que é proposto, lembrando que este método tem como objetivo apresentar soluções em torno de um ponto ordinário.

REFERÊNCIAS

[1] RUGGIERO, Márcia A. Gomes; LOPES, Vera Lucia da Rocha. Cálculo Numérico: aspectos teóricos e computacionais. 2. ed. São Paulo: Makron Books, 1996.

[2] Zemansky, Sears e Freedman, Young E. Física III Eletromagnetismo, Ed. Addisson Wesley 2009.

[3] BOYCE, Willian E. Equações Diferenciais Elementares e Problemas de Valores de Contorno. 10. ed. Grafton, Nova York: LTC, 2015. v. 1.

ANEXOS

ANEXO A – MÉTODO DE RUNGE-KUTTA DE 4ª ORDEM PARA CIRCUITO RC, CARGA E DESCARGA, EM C++

```
#include <stdio.h>
#include <stdlib.h>
#include<math.h>

double f(double t, double vc);
double k1(double tk, double vck);
double k2(double tk, double vck, double h);
double k3(double tk, double vck, double h);
double k4(double tk, double vck, double h);
double fonte, resis, cap;
double e=2.71828;
int CargaOuDescarga;

int main(void){
   int n;
   double h;

    printf("O circuito Resistor-Capacitor esta em carga, digite 1, ou descarga, digite 2:\n1 ou 2? ");
    scanf("%d", &CargaOuDescarga);

   printf("Digite a quantidade de pontos: ");
   scanf("%d", &n);

   double * t = (double *) calloc(n, sizeof(double));
   double * vc = (double *) calloc(n, sizeof(double));
    double * solAnal = (double *) calloc(n, sizeof(double));

   while(n<=0){
   printf("Digite a quantidade de pontos: ");
   scanf("%d", &n);
   }

   printf("Digite o valor inicial de t: ");
   scanf("%lf", &t[0]);

   printf("Digite o valor final de t: ");
   scanf("%lf", &t[n]);

   printf("Digite o valor inicial de vc (ddp do capacitor): ");
   scanf("%lf", &vc[0]);
   solAnal[0] = vc[0];

   if(CargaOuDescarga==1){
    printf("Digite o valor da fonte: ");
   scanf("%lf", &fonte);
   }

   printf("Digite o valor da resistencia: ");
   scanf("%lf", &resis);

    printf("Digite o valor da capacitancia: ");
   scanf("%lf", &cap);

   h = (t[n]-t[0])/n;

   for(int i=0; i<n-1;i++){
```

 DOI: https://doi.org/10.21708/issn27635325.v4n1.a11183.2022

```
        vc[i+1] = vc[i] + (h/6)*(k1(t[i], vc[i]) + 2*k2(t[i], vc[i], h) + 2*k3(t[i], vc[i], h) + k4(t[i], vc[i], h));

        t[i+1] = t[i] + h;

        if(CargaOuDescarga==1){
              solAnal[i+1] = fonte*(1 - pow(e, (-t[i+1])/(resis*cap)));
           }
           if(CargaOuDescarga==2){
              solAnal[i+1] = vc[0]*(pow(e, (-t[i+1])/(resis*cap)));
           }
    }

    FILE * arquivo;

    arquivo = fopen("ddpCapacitor-RK.dat", "w");

    if(arquivo == NULL){
       printf("ERRO: Não foi possível abrir o arquivo\n");
       return 1;
    }

    printf("t\tvc\n");
    fprintf(arquivo,"t\t\tvc\n");
    for(int i=0;i<n;i++){
       fprintf(arquivo,"%lf\t%lf\t%lf\n", t[i], vc[i], solAnal[i]); // aqui colocar o valor da solução analitica
       printf("%lf\t%lf\t%lf\n", t[i], vc[i], solAnal[i]); // printar  solução analitica
    }
    printf("Arquivo salvo com sucesso!");
    free(t);
    free(vc);

    //system("pause");
    return 0;
}

double f(double t, double vc){
     if(CargaOuDescarga==1){
     vc = (fonte-vc)/(resis*cap);
     }
     if(CargaOuDescarga==2){
     vc = (-vc)/(resis*cap);
     }
    return vc;
}

double k1(double tk, double vck){
    return f(tk, vck);
}

double k2(double tk, double vck, double h){
    return f(tk + (h/2), vck + k1(tk, vck)*(h/2));
}

double k3(double tk, double vck, double h){
    return f(tk + (h/2), vck + k2(tk, vck, h)*(h/2));
}

double k4(double tk, double vck, double h){
    return f(tk + h, vck + k3(tk, vck, h)*h);
```

 DOI: https://doi.org/10.21708/issn27635325.v4n1.a11183.2022

```
}
```

ANEXO B – MÉTODO DE SÉRIES DE POTÊNCIAS PARA CIRCUITO RC EM CARGA EM C++

```
#include <stdio.h>
#include <stdlib.h>
#include<math.h>

double fonte, resis, cap;
double e=2.71828;

int main(void){
   int n;
   double h, errorel, sumerro;

   printf("Digite a quantidade de pontos: ");
   scanf("%d", &n);

   double * t = (double *) calloc(n, sizeof(double));
   double * vc = (double *) calloc(n, sizeof(double));
   double * a = (double *) calloc(n, sizeof(double));
   double * solAnal = (double *) calloc(n, sizeof(double));

   while(n<=0){
   printf("Digite a quantidade de pontos: ");
   scanf("%d", &n);
   }

   printf("Digite o valor inicial de t: ");
   scanf("%lf", &t[0]);

   printf("Digite o valor final de t: ");
   scanf("%lf", &t[n]);

   printf("Digite o valor inicial da tensao no capacitor: ");
   scanf("%lf", &a[0]);

   vc[0] = a[0]/cap;

   printf("Digite o valor da fonte: ");
   scanf("%lf", &fonte);

   printf("Digite o valor da resistencia: ");
   scanf("%lf", &resis);

    printf("Digite o valor da capacitancia: ");
   scanf("%lf", &cap);

   h = (t[n]-t[0])/n;

   solAnal[0] = vc[0];

   for(int i=0; i<n-1;i++){

      t[i+1] = t[i] + h;

      vc[i+1] = a[0]/cap;

         for(int j=0; j<n-1;j++){
```

 DOI: https://doi.org/10.21708/issn27635325.v4n1.a11183.2022

```
            if(vc[i]<fonte){

                    a[j+1]=(1/(j+1))*(fonte/resis-a[j]/(resis*cap));


                vc[i+1]+=(a[j+1]*pow(t[i+1], (j+1)))/cap;
            }
                if(vc[i]>=fonte){
                vc[i+1]=fonte;
                }
        }
        solAnal[i+1] = fonte*(1 - pow(e, (-t[i+1])/(resis*cap)));
     }
     FILE * arquivo;

   arquivo = fopen("ddpCapacitor-Sdp.dat", "w");

   if(arquivo == NULL){
      printf("ERRO: Não foi possível abrir o arquivo\n");
      return 1;
   }

   printf("\nt\t\tNumerico\tAnalitico\tErro Per.\n");
   fprintf(arquivo,"t\t\tNumerico\tAnalitico\tErro Per.\n");
   for(int i=0;i<n;i++){
      errorel = fabs(((vc[i] - solAnal[i])/solAnal[i])*100);
      sumerro+=errorel;
      fprintf(arquivo,"%lf\t%lf\t%lf\t%lf\n", t[i], vc[i], solAnal[i], errorel);
      printf("%lf\t%lf\t%lf\t%lf\n", t[i], vc[i], solAnal[i], errorel);
   }
   printf("O erro percentual medio foi de: %lf\n", (sumerro/n));
   printf("Arquivo salvo com sucesso!");
   free(t);
   free(vc);

   //system("pause");
   return 0;
}
```

ANEXO C – MÉTODO DE RUNGE-KUTTA DE 4ª ORDEM PARA CIRCUITO RL, CARGA E DESCARGA, EM C++

```
#include <stdio.h>
#include <stdlib.h>
#include<math.h>

double f(double t, double CorrenteIndutor);
double k1(double tk, double CorrenteIndutork);
double k2(double tk, double CorrenteIndutork, double h);
double k3(double tk, double CorrenteIndutork, double h);
double k4(double tk, double CorrenteIndutork, double h);
double fonte, resis, indut;
double e=2.71828;
int CargaOuDescarga;

int main(void){
   int n;
   double h;

     printf("O circuito Resistor-Indutor esta em carga, digite 1, ou descarga, digite 2:\n1 ou 2? ");
```

 DOI: https://doi.org/10.21708/issn27635325.v4n1.a11183.2022

```
 scanf("%d", &CargaOuDescarga);

printf("Digite a quantidade de pontos: ");
scanf("%d", &n);

double * t = (double *) calloc(n, sizeof(double));

double * CorrenteIndutor = (double *) calloc(n, sizeof(double));
double * solAnal = (double *) calloc(n, sizeof(double));

while(n<=0){
printf("Digite a quantidade de pontos: ");
scanf("%d", &n);
}

printf("Digite o valor inicial de t: ");
scanf("%lf", &t[0]);

printf("Digite o valor final de t: ");
scanf("%lf", &t[n]);

printf("Digite o valor inicial da corrente do indutor: ");
scanf("%lf", &CorrenteIndutor[0]);
solAnal[0] = CorrenteIndutor[0];

if(CargaOuDescarga==1){
 printf("Digite o valor da fonte: ");
scanf("%lf", &fonte);
}

printf("Digite o valor da resistencia: ");
scanf("%lf", &resis);

 printf("Digite o valor da indutancia: ");
scanf("%lf", &indut);

h = (t[n]-t[0])/n;

for(int i=0; i<n-1;i++){

   CorrenteIndutor[i+1] = CorrenteIndutor[i] + (h/6)*(k1(t[i], CorrenteIndutor[i]) +
2*k2(t[i],CorrenteIndutor[i], h) + 2*k3(t[i], CorrenteIndutor[i], h) + k4(t[i], CorrenteIndutor[i], h));

   t[i+1] = t[i] + h;

   if(CargaOuDescarga==1){
        solAnal[i+1] = (fonte/resis)*(1 - pow(e, (-t[i+1]*resis)/(indut)));
     }
     if(CargaOuDescarga==2){
        solAnal[i+1] = CorrenteIndutor[0]*(1 - pow(e, (-t[i+1]*resis)/(indut)));
     }
}

FILE * arquivo;

arquivo = fopen("CorrenteIndutor.dat", "w");

if(arquivo == NULL){
   printf("ERRO: Não foi possível abrir o arquivo\n");
   return 1;
```

```
    }

    printf("t\tCorrenteIndutor\n");
    fprintf(arquivo,"t\t\tCorrenteIndutor\n");
    for(int i=0;i<n;i++){
        fprintf(arquivo,"%lf\t%lf\t%lf\n", t[i], CorrenteIndutor[i], solAnal[i]);
        printf("%lf\t%lf\t%lf\n", t[i], CorrenteIndutor[i], solAnal[i]);
    }

    printf("Arquivo salvo com sucesso!");
    free(t);
    free(CorrenteIndutor);

    //system("pause");
    return 0;
}

double f(double t, double CorrenteIndutor){
     if(CargaOuDescarga==1){
     CorrenteIndutor = (fonte-CorrenteIndutor*resis)/(indut);
     }
     if(CargaOuDescarga==2){
     CorrenteIndutor = (-CorrenteIndutor*resis)/(indut);
     }
    return CorrenteIndutor;
}

double k1(double tk, double CorrenteIndutork){
    return f(tk, CorrenteIndutork);
}

double k2(double tk, double CorrenteIndutork, double h){
    return f(tk + (h/2), CorrenteIndutork + k1(tk, CorrenteIndutork)*(h/2));
}

double k3(double tk, double CorrenteIndutork, double h){
    return f(tk + (h/2), CorrenteIndutork + k2(tk, CorrenteIndutork, h)*(h/2));
}

double k4(double tk, double CorrenteIndutork, double h){
    return f(tk + h, CorrenteIndutork + k3(tk, CorrenteIndutork, h)*h);
}
```

ANEXO D – MÉTODO DE SÉRIES DE POTÊNCIAS PARA CIRCUITO RL EM CARGA EM C++

```
#include <stdio.h>
#include <stdlib.h>
#include<math.h>

double fonte, resis, indut;
double e=2.71828;

int main(void){
    int n;
    double h, errorel, sumerro;

    printf("Digite a quantidade de pontos: ");
    scanf("%d", &n);

    double * t = (double *) calloc(n, sizeof(double));
```

```
double * corrente = (double *) calloc(n, sizeof(double));
double * a = (double *) calloc(n, sizeof(double));
double * solAnal = (double *) calloc(n, sizeof(double));

while(n<=0){
printf("Digite a quantidade de pontos: ");
scanf("%d", &n);
}



printf("Digite o valor inicial de t: ");
scanf("%lf", &t[0]);

printf("Digite o valor final de t: ");
scanf("%lf", &t[n]);

printf("Digite o valor inicial da corrente no circuito RL: ");
scanf("%lf", &a[0]);

corrente[0] = a[0];

printf("Digite o valor da fonte: ");
scanf("%lf", &fonte);

printf("Digite o valor da resistencia: ");
scanf("%lf", &resis);

 printf("Digite o valor da indutancia: ");
scanf("%lf", &indut);

h = (t[n]-t[0])/n;

solAnal[0] = corrente[0];

for(int i=0; i<n-1;i++){

   t[i+1] = t[i] + h;

   corrente[i+1] = a[0];

      for(int j=0; j<n-1;j++){
      if(corrente[i]<(fonte/resis)){
               a[j+1]=(fonte-a[j]*resis)/(indut*(j+1));

          corrente[i+1]+=(a[j+1]*pow(t[i+1], (j+1)));
      }
      if(corrente[i]>=(fonte/resis)){
               corrente[i+1]=fonte/resis;
          }
      }
      solAnal[i+1] = (fonte/resis)*(1 - pow(e, (-t[i+1]*resis)/(indut)));
 }
 FILE * arquivo;

arquivo = fopen("CorrenteIndutor-Sdp.dat", "w");

if(arquivo == NULL){
   printf("ERRO: Não foi possível abrir o arquivo\n");
   return 1;
```

 DOI: https://doi.org/10.21708/issn27635325.v4n1.a11183.2022

```
    }

    printf("\nt\t\tNumerico\tAnalitico\tErro Per.\n");
    fprintf(arquivo,"t\t\tNumerico\tAnalitico\tErro Per.\n");
    for(int i=0;i<n;i++){
        errorel = fabs(((corrente[i] - solAnal[i])/solAnal[i])*100);
        sumerro+=errorel;
        fprintf(arquivo,"%lf\t%lf\t%lf\t%lf\n", t[i], corrente[i], solAnal[i], errorel);
        printf("%lf\t%lf\t%lf\t%lf\n", t[i], corrente[i], solAnal[i], errorel);
    }
    printf("O erro percentual medio foi de: %lf\n", (sumerro/n));
    printf("Arquivo salvo com sucesso!");

    free(t);
    free(corrente);

    //system("pause");
    return 0;
}
```

ANEXO E – MÉTODO DE RUNGE-KUTTA DE 4ª ORDEM PARA CIRCUITO RLC AMORTECIDO EM C++

```
#include <stdio.h>
#include <stdlib.h>
#include<math.h>

double fi(double t, double corrente, double correntek);
double fk(double t, double corrente, double correntek);
double k1(double tk, double corrente, double correntek);
double k2(double tk, double corrente, double correntek, double h);
double k3(double tk, double corrente, double correntek, double h);
double k4(double tk, double corrente, double correntek, double h);
double k1k(double tk, double corrente, double correntek);
double k2k(double tk, double corrente, double correntek, double h);
double k3k(double tk, double corrente, double correntek, double h);
double k4k(double tk, double corrente, double correntek, double h);
double indut, resis, cap;
double e=2.71828;
int rlcCase;

int main(void){
    int n;
    double h, errorel, sumerro;

    printf("Digite a quantidade de pontos: ");
    scanf("%d", &n);

    double * t = (double *) calloc(n, sizeof(double));
    double * corrente = (double *) calloc(n, sizeof(double));
    double * correntek = (double *) calloc(n, sizeof(double));
    double * solAnal = (double *) calloc(n, sizeof(double));

    while(n<=0){
    printf("Digite a quantidade de pontos: ");
    scanf("%d", &n);
    }

    printf("Digite o valor inicial de t: ");
    scanf("%lf", &t[0]);
```

```
    printf("Digite o valor final de t: ");
    scanf("%lf", &t[n]);

    printf("Digite o valor inicial da corrente: ");
    scanf("%lf", &corrente[0]);

    printf("Digite o valor inicial da primeira derivada da corrente: ");
    scanf("%lf", &correntek[0]);

     printf("Digite o valor da indutância: ");
    scanf("%lf", &indut);

    printf("Digite o valor da resistencia: ");
    scanf("%lf", &resis);

     printf("Digite o valor da capacitancia: ");
    scanf("%lf", &cap);

    h = (t[n]-t[0])/n;

     solAnal[0] = corrente[0];

     if((resis/(2*indut))>(1/(pow(indut*cap, 0.5)))){
         rlcCase = 1;
     }
     if((resis/(2*indut))==(1/(pow(indut*cap, 0.5)))){
         rlcCase = 2;
     }
     if((resis/(2*indut))<(1/(pow(indut*cap, 0.5)))){
         rlcCase = 3;
     }

    for(int i=0; i<n-1;i++){

        corrente[i+1] = corrente[i] + (h/6)*(k1(t[i], corrente[i], correntek[i]) + 2*k2(t[i], corrente[i], correntek[i], h)
+ 2*k3(t[i], corrente[i], correntek[i], h) + k4(t[i], corrente[i], correntek[i], h));

          correntek[i+1] = correntek[i] + (h/6)*(k1k(t[i], corrente[i], correntek[i]) + 2*k2k(t[i], corrente[i],
correntek[i], h) + 2*k3k(t[i], corrente[i], correntek[i], h) + k4k(t[i], corrente[i], correntek[i], h));

        t[i+1] = t[i] + h;

        if(rlcCase == 1){ // Teste para parâmetros específicos para plotagem de graficos conforme o artigo.
              solAnal[i+1]      =      -0.016500895*pow(e,      -1.031947467*t[i+1])+0.516500895*pow(e,      -
32.30138587*t[i+1]);
          }
          if(rlcCase == 2){
              solAnal[i+1] = 0.5*pow(e, -2*t[i+1])+-1*t[i+1]*pow(e, -2*t[i+1]);
          }
          if(rlcCase == 3){
              solAnal[i+1] = 0.5*pow(e, -2*t[i+1])*cos(2*t[i+1]) - 0.5*pow(e, -2*t[i+1])*sin(2*t[i+1]);
          }
    }

    FILE * arquivo;

    arquivo = fopen("CorrenteCircuitoRLC-RK.dat", "w");

    if(arquivo == NULL){
```

```
        printf("ERRO: Não foi possível abrir o arquivo\n");
        return 1;
    }

    printf("\nt\t\tNumerico\tAnalitico\tErro Per.\n");
    fprintf(arquivo,"t\t\tNumerico\tAnalitico\tErro Per.\n");
    for(int i=0;i<n;i++){
        errorel = fabs(((corrente[i] - solAnal[i])/solAnal[i])*100);
        sumerro+=errorel;
        fprintf(arquivo,"%lf\t%lf\t%lf\t%lf\n", t[i], corrente[i], solAnal[i], errorel);
        printf("%lf\t%lf\t%lf\t%lf\n", t[i], corrente[i], solAnal[i], errorel);
    }
    printf("O erro percentual medio foi de: %lf\n", (sumerro/n));
    printf("Arquivo salvo com sucesso!");
    free(t);
    free(corrente);


    //system("pause");
    return 0;
}



double fi(double tk, double corrente, double correntek){
    return correntek; // Retorna a inclinação do ponto (derivada)
}

double fk(double tk, double corrente, double correntek){
     correntek = -((resis/indut)*correntek + corrente/(indut*cap));
    return correntek; // Retorna a inclinação do ponto (derivada)
}

double k1(double tk, double corrente, double correntek){
     return fi(tk, corrente, correntek);
}

double k1k(double tk, double corrente, double correntek){
     return fk(tk, corrente, correntek);
}

double k2(double tk, double corrente, double correntek, double h){
     return fi(tk + (h/2), corrente + k1(tk, corrente, correntek)*(h/2), correntek + k1k(tk, corrente, correntek)*(h/2));

}

double k2k(double tk, double corrente, double correntek, double h){
     return fk(tk + (h/2), corrente + k1(tk, corrente, correntek)*(h/2), correntek + k1k(tk, corrente,
correntek)*(h/2));
}

double k3(double tk, double corrente, double correntek, double h){
    return fi(tk + (h/2), corrente + k2(tk, corrente, correntek, h)*(h/2), correntek + k2k(tk, corrente, correntek,
h)*(h/2));
}

double k3k(double tk, double corrente, double correntek, double h){
    return fk(tk + (h/2), corrente + k2(tk, corrente, correntek, h)*(h/2), correntek + k2k(tk, corrente, correntek,
h)*(h/2));
}
```

```
double k4(double tk, double corrente, double correntek, double h){
   return fi(tk + h, corrente + k3(tk, corrente, correntek, h)*h, correntek + k3k(tk, corrente, correntek, h)*h);
}

double k4k(double tk, double corrente, double correntek, double h){
     return fk(tk + h, corrente + k3(tk, corrente, correntek, h)*h, correntek + k3k(tk, corrente, correntek, h)*h);
}
```

ANEXO F – MÉTODO DE SÉRIES DE POTÊNCIAS PARA CIRCUITO RLC AMORTECIDO EM C++

```
#include <stdio.h>
#include <stdlib.h>
#include<math.h>

double indut, resis, cap;
double e=2.71828;
int rlcCase;


int main(void){
   int n;
   double h, errorel, sumerro;

   printf("Digite a quantidade de pontos: ");
   scanf("%d", &n);

   double * t = (double *) calloc(n, sizeof(double));
   double * corrente = (double *) calloc(n, sizeof(double));
   double * a = (double *) calloc(n, sizeof(double));
   double * solAnal = (double *) calloc(n, sizeof(double));

   while(n<=0){
   printf("Digite a quantidade de pontos: ");
   scanf("%d", &n);
   }

   printf("Digite o valor inicial de t: ");
   scanf("%lf", &t[0]);

   printf("Digite o valor final de t: ");
   scanf("%lf", &t[n]);

   printf("Digite o valor inicial da corrente: ");
   scanf("%lf", &a[0]);

   corrente[0] = a[0];

   printf("Digite o valor inicial da primeira derivada da corrente: ");
   scanf("%lf", &a[1]);

    printf("Digite o valor da indutância: ");
   scanf("%lf", &indut);

   printf("Digite o valor da resistencia: ");
   scanf("%lf", &resis);

    printf("Digite o valor da capacitancia: ");
   scanf("%lf", &cap);
```

 DOI: https://doi.org/10.21708/issn27635325.v4n1.a11183.2022

```
    h = (t[n]-t[0])/n;

    solAnal[0] = corrente[0];

     if((resis/(2*indut))>(1/(pow(indut*cap, 0.5)))){
         rlcCase = 1;
     }
     if((resis/(2*indut))==(1/(pow(indut*cap, 0.5)))){
         rlcCase = 2;
     }
     if((resis/(2*indut))<(1/(pow(indut*cap, 0.5)))){
         rlcCase = 3;
     }

    for(int i=0; i<n-1;i++){

       t[i+1] = t[i] + h;

       corrente[i+1] = a[0] + a[1]*t[i+1];

          for(int j=0; j<n-2;j++){

              a[j+2]=-(1/(j+2))*((resis*a[j+1])/indut - a[j]/(indut*cap*(j+1)));

          corrente[i+1]+=a[j+2]*pow(t[i+1], (j+2));
          }
       if(rlcCase == 1){ // Teste para parâmetros específicos para plotagem de graficos conforme o artigo.
              solAnal[i+1]      =      -0.016500895*pow(e,      -1.031947467*t[i+1])+0.516500895*pow(e,      -
32.30138587*t[i+1]);
          }
          if(rlcCase == 2){
              solAnal[i+1] = 0.5*pow(e, -2*t[i+1])+-1*t[i+1]*pow(e, -2*t[i+1]);
          }
          if(rlcCase == 3){
              solAnal[i+1] = 0.5*pow(e, -2*t[i+1])*cos(2*t[i+1]) - 0.5*pow(e, -2*t[i+1])*sin(2*t[i+1]);
          }
    }

    FILE * arquivo;

    arquivo = fopen("CorrenteCircuitoRLC-SdP.dat", "w");

    if(arquivo == NULL){
       printf("ERRO: Não foi possível abrir o arquivo\n");
       return 1;
    }

    printf("\nt\t\tNumerico\tAnalitico\tErro Per.\n");
    fprintf(arquivo,"t\t\tNumerico\tAnalitico\tErro Per.\n");
    for(int i=0;i<n;i++){
       errorel = fabs(((corrente[i] - solAnal[i])/solAnal[i])*100);
       sumerro+=errorel;
       fprintf(arquivo,"%lf\t%lf\t%lf\t%lf\n", t[i], corrente[i], solAnal[i], errorel);
       printf("%lf\t%lf\t%lf\t%lf\n", t[i], corrente[i], solAnal[i], errorel);
    }
    printf("O erro percentual medio foi de: %lf\n", (sumerro/n));
    printf("Arquivo salvo com sucesso!");
    free(t);
    free(corrente);
```

 DOI: https://doi.org/10.21708/issn27635325.v4n1.a11183.2022

```
    //system("pause");
    return 0;
}
```

 DOI: https://doi.org/10.21708/issn27635325.v4n1.a11183.2022