Revista Brasileira de Ensino de Física, vol. 45, e20230347 (2023)
www.scielo.br/rbef
DOI: https://doi.org/10.1590/1806-9126-RBEF-2023-0347

Produtos e Materiais Didáticos

Uma introdução aos métodos computacionais em física a partir do problema do decaimento radioativo

An introduction to computational methods in physics from the radioactive decay problem

M.J.B. Ferreira[*1], B.F. de Oliveira[2], A.M. Gonçalves[2]

[1]Universidade Estadual de Maringá, Departamento de Matemática, 87020-900, Maringá, PR, Brasil.
[2]Universidade Estadual de Maringá, Departamento de Física, 87020-900, Maringá, PR, Brasil.

Recebido em 11 de novembro de 2023. Aceito em 20 de novembro de 2023.

Neste artigo apresentamos uma introdução à física computacional e à programação científica a partir da aplicação em um problema de decaimento radioativo. Esse problema é resolvido computacionalmente a partir de duas abordagens distintas: integração numérica e simulação estocástica. A integração numérica é feita utilizando-se o método de Runge-Kutta, para a obtenção das soluções numéricas das equações diferenciais que regem a dinâmica do modelo, enquanto as simulações estocásticas são implementadas com o uso de simulações de Monte Carlo, em que números aleatórios são utilizados para tomadas de decisão. Uma comparação entre as vantagens e desvantagens de cada método e da conveniência de cada um deles é feita e alguns exemplos de implementação dos algoritmos aqui discutidos são também apresentados em diferentes linguagens de programação.
Palavras-chave: Decaimento radioativo, integração numérica, simulações estocásticas, Runge-Kutta, Monte Carlo.

This article brings up an introduction to computational physics and scientific programming through an application, the radioactive decay problem. This problem is solved computationally through two different approaches: numerical integration and stochastic simulation. The numerical integration is realised via the Runge-Kutta method, used to obtain numerical solutions of the ordinary differential equation describing the model dynamics, while the stochastic simulations are implemented via Monte Carlo simulations, where random numbers are used to guide decision making. A comparicion of the advantages of each method and its convenience is made and some examples of implementation of the algorithms presented in this work are also exhibited in different programming languages.
Keywords: Radioactive decay, numerical calculus, stochastic simulations, Runge-Kutta, Monte Carlo.

1. Introdução

Não é novidade que a física computacional tem crescido bastante e conquistado cada vez mais espaço no meio científico nas últimas décadas [1, 2], sendo utilizada principalmente para fins de pesquisas. Atualmente, até mesmo os computadores e celulares pessoais possuem uma capacidade de processamento razoável, permitindo que simulações relativamente complexas possam ser realizadas em poucos minutos. Em particular, o ensino de física ao longo dos anos tem se utilizado cada vez mais da física computacional. Por um lado, buscando a formação de profissionais capacitados a desenvolver atividades nessa área e, por outro, como ferramenta para promover uma aprendizagem mais significativa [3–6]. Os próprios currículos de física mais recentes têm procurado incorporar cada vez mais a física computacional [4, 7–9]. Diante dessa crescente incorporação, torna-se então cada vez mais necessário colocar o estudante de física em contato com a física computacional, tendo em vista que essa possui diferentes abordagens em física [10–19]. Neste trabalho apresentamos uma introdução a duas importantes ferramentas da física computacional: o método de Runge-Kutta – para a resolução numérica de equações diferenciais de primeira ordem; e o método de Monte Carlo – para simulações estocásticas. Estes dois são excelentes exemplos didáticos, dados à sua simplicidade e abrangência. Apresentaremos esses métodos da maneira mais autocontida possível, tendo como foco principal os algoritmos que irão desempenhar cada etapa do processo. Deste modo, para a compreensão deste trabalho não é necessário nenhum conhecimento prévio de nenhuma linguagem de programação específica. Mesmo assim, mostraremos alguns exemplos de implementação para esses algoritmos utilizando algumas linguagens de programação: Mathematica, C e Python.

O exemplo escolhido, como inspiração para a utilização dos métodos computacionais, é o problema do decaimento radioativo (PDR). O motivo para isso é que esse exemplo, além de ser bastante interessante e rico do ponto de vista computacional, também pode ser resolvido analiticamente, tornando-o um excelente exemplo didático.

*Endereço de correspondência: migueljbf@gmail.com

O fenômeno da radioatividade surge do fato de que o núcleo de átomos muito grandes são mais instáveis que o núcleo de átomos menores. Essa instabilidade eventualmente acaba causando uma quebra do núcleo atômico, transformando-o em um núcleo menor e mais estável. Usualmente, na quebra de um núcleo atômico alguns tipos de partículas podem ser liberados: partículas α, β ou γ [20]. Esse fenômeno é conhecido como decaimento radioativo e transforma um elemento químico X em outro elemento químico Y. Escrevemos $X \to Y$. Se inicialmente existe uma certa quantidade de X, a medida que os átomos decaem, essa quantidade diminui, enquanto a quantidade de Y aumenta. Sendo assim, resolver esse problema significa conhecer as quantidades de cada átomo como uma função do tempo.

Considere ainda que o elemento Y também seja radioativo e decaia em Z. Dessa forma, podemos nos perguntar como seriam as funções que descrevem as quantidades de cada átomo como uma função do tempo. Um exemplo real deste tipo de sistema é o decaimento $^{210}_{83}\text{Bi} \to ^{210}_{84}\text{Po} \to ^{206}_{82}\text{Pb}$, onde um átomo de Bismuto ($^{210}_{83}$Bi) decai em um átomo de Polônio ($^{210}_{84}$Po), o qual por sua vez decai em um átomo de chumbo ($^{206}_{82}$Pb) [21].

Conforme veremos na próxima seção, esse problema pode ser formulado a partir de um sistema de equações diferenciais ordinárias (EDOs) acopladas de primeira ordem, que pode ser resolvido analiticamente, isto é, é possível obter analiticamente as soluções para esse sistema de equações e, consequentemente, obter as quantidades de cada átomo como uma função do tempo. Em geral, somente uma pequena parcela de exemplos como esse, envolvendo resolver EDOs acopladas, podem ser resolvidos analiticamente. Nos outros casos, recorrer a métodos computacionais é quase sempre a única alternativa viável.

Computacionalmente esse problema pode ser abordado de duas maneiras distintas: integração numérica e a partir de simulações estocásticas. Para integração numérica, optamos por utilizar o método de Runge-Kutta, que permite resolver EDOs de primeira ordem utilizando um algoritmo de iteração. Nas simulações estocásticas, utilizamos o método de Monte Carlo, no qual simulações da dinâmica do modelo são realizadas baseadas em sorteios de números aleatórios (estocásticos), que nesse caso representam a decisão sobre se cada átomo decai ou não num dado intervalo de tempo. Para efetuar simulações deste tipo não é necessário conhecer o sistema de EDOs que regem o modelo.

Este trabalho está organizado da seguinte maneira: na seção 2 apresentamos o PDR e sua solução analítica. É discutido também a noção de tempo de meia-vida e probabilidade de decaimento dos átomos. Apresentamos o método de Runge-Kutta para resolução do sistema de EDOs na seção 3 e o algoritmo utilizado para a execução desse método na seção 4. Na seção 5 discutimos o método de Monte Carlo e na seção 6 o algoritmo utilizado para a realização da simulação. Os resultados obtidos em cada abordagem são apresentados na seção 7 e comparados. Na seção 8 apresentamos alguns exemplos de linguagens de programação que podem ser utilizadas para a implementação dos algoritmos discutidos nas seção anteriores. Finalizamos esse trabalho na seção 9 com algumas considerações finais.

2. Problema do Decaimento Radioativo

Nesta seção, estudaremos o problema do decaimento radioativo PDR. Considere primeiramente um decaimento do tipo $X \to Y$. Separaremos o estudo apresentado nesta seção em duas partes: na primeira parte analisamos somente a quantidade de átomos X presentes no sistema como uma função do tempo e na segunda parte iremos analisar também as quantidades dos outros elementos químicos que possam vir a existir.

2.1. Decaimento simples

Considere um sistema contendo uma quantidade inicial N_0 de átomos de um determinado elemento químico radioativo X. À medida que o tempo passa X decai em outro elemento químico Y, isto é, $X \to Y$, de modo que a quantidade de X é uma função do tempo, representada por $N_X(t)$. A equação diferencial que modela esse decaimento é

$$\frac{d}{dt}N_X = -\lambda_X N_X, \tag{1}$$

sendo λ_X uma constante positiva chamada de constante de decaimento. Essa constante está relacionada à rapidez do decaimento e é uma característica do elemento químico em questão, conforme discutiremos com mais detalhes a seguir. A equação (1) está sujeita à condição inicial $N_X(0) = N_0$ e considera que a taxa instantânea de decaimento é proporcional à quantidade de átomos X presentes no sistema no dado instante. A solução para essa equação pode ser obtida a partir de uma integração direta e é dada por:

$$N_X(t) = N_0 e^{-\lambda_X t}. \tag{2}$$

A constante de decaimento λ_X está relacionada ao tempo de meia-vida $t^X_{1/2}$ do elemento X. Sendo o tempo de meia vida definido como sendo o tempo necessário para a quantidade de átomos cair pela metade [22]. Podemos obter tal relação notando que

$$N_X(t^X_{1/2}) = \frac{N_0}{2} \;\Rightarrow\; e^{-\lambda_X t^X_{1/2}} = 2^{-1}, \tag{3}$$

e, portanto,

$$\lambda_X = \frac{\ln(2)}{t^X_{1/2}}. \tag{4}$$

Assim conseguimos relacionar diretamente a constante de decaimento com o tempo de meia-vida do elemento químico em consideração.

 DOI: https://doi.org/10.1590/1806-9126-RBEF-2023-0347

Mais adiante, a fim de implementar a dinâmica da simulação estocástica, será necessário conhecer a probabilidade P_X de que um único átomo decaia em um intervalo de tempo Δt. Podemos definir essa probabilidade como a razão entre a quantidade de átomos decaídos nesse intervalo e a quantidade de átomos no instante que marca o início do intervalo, isto é

$$P_X = \frac{|N_X(t+\Delta t) - N_X(t)|}{N_X(t)}. \tag{5}$$

Da equação (2) obtemos que $N_X(t+\Delta t) = e^{-\lambda_X \Delta t} N_X(t)$ e com isso podemos escrever

$$P_X = 1 - e^{-\lambda_X \Delta t}. \tag{6}$$

A equação (6) fornece a probabilidade de decaimento de um único átomo em um intervalo de tempo arbitrário Δt. Utilizando a equação (3) é possível mostrar que se $\Delta t = t^X_{1/2}$ a probabilidade de decaimento será de 50%. No entanto, nas simulações estocásticas o intervalo de tempo é tipicamente muito pequeno comparado ao tempo de meia-vida, isto é, $\Delta t \ll t^X_{1/2}$. Nesse caso, usando a equação (4) podemos notar que $\lambda_X \Delta t \ll 1$ e, portanto, $e^{-\lambda_X \Delta t} \approx 1 - \lambda_X \Delta t$. A probabilidade de decaimento de um único átomo, para um intervalo de tempo curto, fica então

$$P_X = \lambda_X \Delta t = \frac{\ln(2)}{t^X_{1/2}} \Delta t. \tag{7}$$

A quantidade inicial de átomos N_0 não desempenha um papel muito importante nesse modelo, representando apenas um fator de escala que pode ser absorvido se considerarmos a densidade de átomos ao invés do número de átomos. A densidade de átomos $\rho_X(t)$ é definida pela razão

$$\rho_X(t) = \frac{N_X(t)}{N_0}. \tag{8}$$

Dividindo ambos os lados da equação (1) por N_0, obtemos a seguinte equação diferencial para a densidade de átomos ρ_X

$$\frac{d}{dt}\rho_X = -\lambda_X \rho_X, \tag{9}$$

sujeita à condição inicial $\rho_X(0) = 1$. A interpretação dada à equação (9) é mesma dada à equação (1), isto é, a taxa de crescimento da densidade de átomos é proporcional à densidade de átomos, sendo λ_X a constante de proporcionalidade que está associada ao tempo de meia-vida $t^X_{1/2}$ do elemento X pela equação (4). Visto que a equação (1) é linear, a solução da equação (9) pode ser obtida dividindo-se ambos os lados da equação (2) por N_0, desta forma obtemos a solução

$$\rho_X(t) = e^{-\lambda_X t}. \tag{10}$$

Consideremos, sem perda de generalidade que no instante inicial só existem átomos do elemento X, isto é, $N_X(0) = N_0$ e $N_Y(0) = 0$. Para conhecer a densidade de átomos Y presentes no sistema, precisamos levar em consideração a lei de conservação dos átomos, isto é, em um sistema isolado como esse a quantidade total de átomos deve ser constante. Em outras palavras,

$$N_X(t) + N_Y(t) = N_0. \tag{11}$$

Dessa forma a densidade total de átomos deve também se manter constante. Dividindo ambos os lados da equação acima por N_0 obtemos a lei de conservação da densidade

$$\rho_X(t) + \rho_Y(t) = 1. \tag{12}$$

No caso em que o elemento Y não decai em nenhum outro elemento, podemos usar a lei de conservação da densidade para obter $\rho_Y(t)$,

$$\rho_Y(t) = 1 - \rho_X(t) = 1 - e^{-\lambda_X t}. \tag{13}$$

Entretanto, no caso em que o elemento Y possa decair em outro átomo Z, será analisado com mais detalhes a seguir.

2.2. Sequência de decaimentos radioativos

Considere agora que o elemento Y decai em outro elemento Z. Assim temos um decaimento do tipo $X \to Y \to Z$. Este é o caso, por exemplo, do decaimento ${}^{210}_{83}\mathrm{Bi} \to {}^{210}_{84}\mathrm{Po} \to {}^{206}_{82}\mathrm{Pb}$, onde os átomos de Bismuto (${}^{210}_{83}\mathrm{Bi}$) decaem em átomos de Polônio (${}^{210}_{84}\mathrm{Po}$), os quais por sua vez decaem em átomos de Chumbo (${}^{206}_{82}\mathrm{Pb}$).

As densidades de cada elemento químico como função do tempo são representadas pelas funções $\rho_X(t)$, $\rho_Y(t)$ e $\rho_Z(t)$. Dessa forma, usando a lei de conservação da quantidade total de átomos é sempre válido que

$$\rho_X(t) + \rho_Y(t) + \rho_Z(t) = 1\ . \tag{14}$$

Considere por simplicidade, mas sem perda de generalidade, que inicialmente só existem átomos do elemento X no sistema, isto é

$$\rho_X(0) = 1, \tag{15}$$

$$\rho_Y(0) = \rho_Z(0) = 0. \tag{16}$$

A equação diferencial que modela a densidade de átomos X é a equação (9), cuja a solução é dada pela equação (10) com a seguinte condição inicial: $\rho_X(0) = 1$.

A equação diferencial que modela a densidade de átomos Y é similar à equação (9), porém com a adição de um termo extra, relacionado ao decaimento $X \to Y$. Em outras palavras

$$\frac{d}{dt}\rho_Y = -\lambda_Y \rho_Y + \lambda_X \rho_X. \tag{17}$$

A constante λ_Y está relacionada com o tempo de meia-vida do elemento Y pela expressão

$$\lambda_Y = \frac{\ln(2)}{t^Y_{1/2}}. \tag{18}$$

Uma vez que a função $\rho_X(t)$ é conhecida da equação (10), a solução da equação (17), sujeita à condição inicial $\rho_Y(0) = 0$, pode ser obtida utilizando-se o método do fator integrante [23]. Assim, a equação diferencial se torna uma diferencial exata cuja solução é dada por

$$\rho_Y(t) = \frac{\lambda_X}{\lambda_Y - \lambda_X}\left(e^{-\lambda_X t} - e^{-\lambda_Y t}\right), \tag{19}$$

Por fim, a densidade de átomos Z pode ser modelada pela equação diferencial (20), sujeita à condição inicial $\rho_Z(0) = 0$. Note que nessa equação só existe um termo relacionado ao decaimento $Y \to Z$.

$$\dot{\rho}_Z = \lambda_Y \rho_Y. \tag{20}$$

Usando a lei de conservação do número total de átomos podemos encontrar a função $\rho_Z(t)$ sem precisar necessariamente resolver a equação (20). Para isso utilizamos a equação (14) e, uma vez conhecidas $\rho_X(t)$ e $\rho_Y(t)$, escrevemos

$$\rho_Z(t) = 1 - \rho_X(t) - \rho_Y(t), \tag{21}$$

que simplificando fica

$$\rho_Z(t) = \frac{\lambda_Y}{\lambda_Y - \lambda_X}\left(1 - e^{-\lambda_X t}\right) - \frac{\lambda_X}{\lambda_Y - \lambda_X}\left(1 - e^{-\lambda_Y t}\right). \tag{22}$$

Na Figura 1 podemos ver os gráficos dessas densidades em função do tempo para o decaimento $^{210}_{83}$Bi $\to^{210}_{84}$ Po $\to^{206}_{82}$ Pb. Os tempos de meia-vida utilizados para a confecção desse gráfico são: $t^X_{1/2} = 120$ horas e $t^Y_{1/2} = 3321,12$ horas [24].

Inicialmente, só existem átomos de Bi no sistema, por isso $\rho_X(0) = 1$ e $\rho_Y(0) = \rho_Z(0) = 0$. À medida que o tempo passa o Bi decai exponencialmente até se extinguir quase por completo, $\rho_X \to 0$. A densidade de Po aumenta rapidamente no início devido ao decaimento do Bi, mas à medida que a densidade de Bi começa a ficar muito baixa e o Po a decair, a taxa de variação do Po começa a diminuir até chegar a zero, no ponto onde a densidade do Po atinge seu valor máximo (entre $0,8$ e $0,9$). Daí em diante a densidade de Po só diminui até se extinguir quase por completo, $\rho_Y \to 0$. A densidade de Pb aumenta durante todo o processo, mais lentamente no início, devido à escassez de Po, e mais rapidamente no final. À medida que a densidade de Bi e Po se aproximam de zero a densidade de Pb tende a 1, o que representa que ao final desse processo todos os átomos acabam decaindo em Pb.

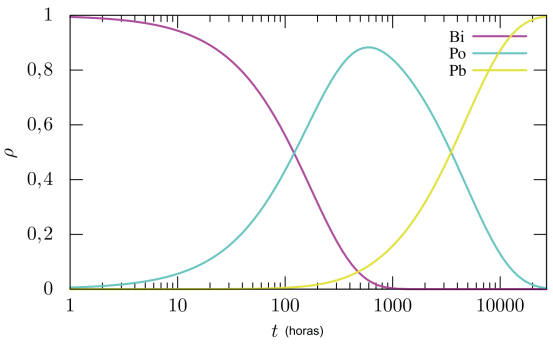

Figura 1: Densidades de átomos de Bi, Po e Pb em função do tempo. Observe que inicialmente só há átomos de Bi e ao fim somente átomos de Pb.

Conforme pudemos observar nesta seção, o PDR pode ser resolvido analiticamente, isto é, resolvendo o sistema de equações acopladas dadas pelas equações (9), (17) e (20) foi possível obter as soluções analíticas representadas pelas equações (10), (19) e (22) as quais são também representadas graficamente na Figura 1 para o exemplo do decaimento $^{210}_{83}$Bi $\to^{210}_{84}$ Po $\to^{206}_{82}$ Pb. A seguir apresentaremos duas maneiras de resolver este problema computacionalmente, a partir do método de Runge-Kutta (integração numérica) e a partir das simulações de Monte Carlo (simulações estocásticas).

3. Método de Runge-Kutta

Nesta seção, apresentaremos o método de Runge-Kutta (RK) utilizado para resolver equações diferencias ordinárias (EDOs) numericamente. Este método pode ser aplicado a qualquer EDO de primeira ordem da forma

$$\frac{d}{dt}\rho = f(t, \rho), \tag{23}$$

sujeita à condição inicial $\rho(t_0) = \rho_0$ e sendo f qualquer função das variáveis ρ e t. A solução obtida a partir do método de RK é uma solução numérica aproximada, que fornece os valores da função $\rho(t)$ para alguns valores discretizados de t pertencentes a um intervalo $t \in [t_0 : t_f]$. Entretanto, antes de prosseguir com o estudo do método de RK, para fins didáticos, apresentaremos primeiramente o método de Euler para resolver equações diferenciais do mesmo tipo.

3.1. Euler

O primeiro passo é discretizar o intervalo de tempo em consideração, dividindo-o em intervalos menores de mesma duração Δt. Dessa forma, os instantes de tempo serão dados por

$$t_n = t_0 + n\Delta t, \tag{24}$$

sendo $n = 0,\ 1,\ 2,\ 3, \ldots, N$. A quantidade de intervalos considerados com essa discretização é representada pela variável N, que pode ser obtida observando-se que $t_N = t_f$, isto é,

$$N = \frac{t_f - t_0}{\Delta t}. \tag{25}$$

Em geral, não há garantia que o resultado do cálculo acima seja um número natural, então, para contornar esse problema, teremos o cuidado de escolher os parâmetros de modo a garantir que N seja, de fato, um número natural.

 DOI: https://doi.org/10.1590/1806-9126-RBEF-2023-0347

Para Δt suficientemente pequeno, pode-se escrever a aproximação

$$\frac{d}{dt}\rho(t_n) \approx \frac{\rho(t_{n+1}) - \rho(t_n)}{\Delta t}, \tag{26}$$

a qual, utilizando a equação (23), pode ser reescrita como

$$f(t_n, \rho(t_n)) = \frac{\rho(t_{n+1}) - \rho(t_n)}{\Delta t}. \tag{27}$$

Dessa forma, isolando $\rho(t_{n+1})$ na equação (27), obtemos

$$\rho(t_{n+1}) = \rho(t_n) + f(t_n, \rho_n)\Delta t, \tag{28}$$

A equação (28) possibilita estimar o valor $\rho(t_{n+1})$, no instante t_{n+1}, uma vez conhecido o valor $\rho(t_n)$, no instante de tempo anterior t_n. Sabe-se, da condição inicial, que $\rho(t_0) = \rho_0$. Aplicando-se a equação (28) recursivamente a partir do instante inicial, é possível gerar uma tabela de valores para os pontos da forma $(t_n, \rho(t_n))$, com $n = 0,\ 1,\ 2,\ 3, \ldots, N$.

Esse método é conhecido como método de Euler e pode ser visualizado geometricamente. A equação (28) mostra ser possível estimar o valor de $\rho(t_{n+1})$ uma vez conhecido o ponto $(t_n, \rho(t_n))$. O motivo para isso é que, segundo a equação (23), é possível calcular a inclinação da reta tangente à função $\rho(t)$ no ponto $(t, \rho(t))$, uma vez que $\frac{d}{dt}\rho(t) = f(t, \rho(t))$. Com isso, uma vez conhecido o valor $\rho(t_n)$, pode-se calcular a inclinação α no ponto $(t_n, \rho(t_n))$ usando $\alpha = f(t_n, \rho(t_n))$. Dessa forma, conforme mostra a Figura 2, podemos utilizar a inclinação da reta que passa pelo ponto $(t_n, \rho(t_n))$ (reta em vermelho na Figura 2) para estimar o valor de $\rho(t_{n+1})$. É importante destacar que as coordenadas do ponto $(t_n, \rho(t_n))$, utilizado para estimar o novo ponto (t_{n+1}, ρ_{n+1}), são oriundas de uma aproximação.

3.2. Runge-Kutta de segunda ordem

O método de Euler pode ser otimizado de várias maneiras distintas. Uma otimização bastante conhecida e

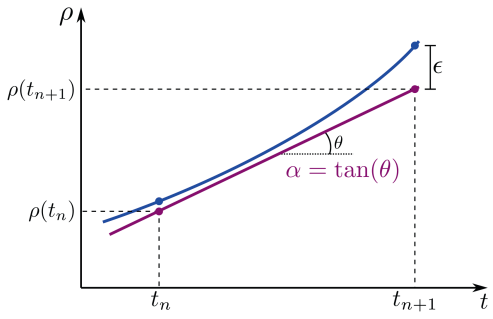

Figura 2: Visualização geométrica do método de Euler. A curva em azul representa a solução exata do problema, a qual não se conhece. Observe que a inclinação da reta não é exatamente a inclinação da reta tangente à curva azul no ponto $(t_n, \rho(t_n))$, uma vez que esse não está necessariamente sobre a curva e já é oriundo de uma aproximação. De forma ilustrativa pode-se perceber o erro avaliado nesse processo.

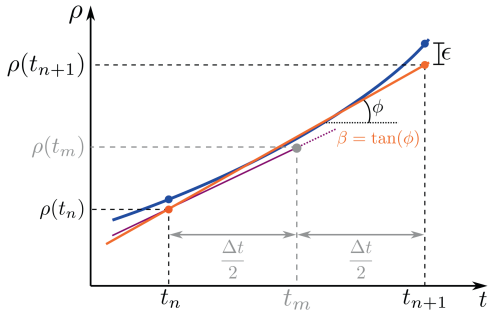

Figura 3: Visualização geométrica do método de Runge-Kutta de segunda ordem. A inclinação α é utilizada para estimar o ponto $(t_m, \rho(t_m))$, que encontra-se no meio do intervalo. Com esse novo ponto calcula-se uma nova inclinação β, a qual é finalmente utilizada para estimar o valor $\rho(t_{n+1})$. Nessa imagem é possível perceber também uma ilustração do erro avaliado nesse processo.

poderosa é o método de RK, que consiste em encontrar mais do que uma inclinação por intervalo para estimar os novos pontos da função. A Figura 3 ilustra o método de RK de segunda ordem, onde são calculadas duas inclinações por intervalo a fim de se obter uma estimativa mais precisa.

A primeira inclinação é aquela obtida a partir do ponto $(t_n, \rho(t_n))$, isto é

$$\alpha = f(t_n, \rho(t_n)). \tag{29}$$

Com essa inclinação e utilizando o método de Euler podemos fazer uma estimativa do valor da função no meio do intervalo, isto é, no ponto $t_m = t_n + \Delta t/2$. Dessa forma obtemos

$$\rho(t_m) = \rho(t_n) + \alpha\frac{\Delta t}{2}. \tag{30}$$

Uma segunda inclinação β pode ser calculada a partir do ponto médio, calculado anteriormente, isto é

$$\beta = f(t_m, \rho(t_m)). \tag{31}$$

O método de RK de segunda ordem consiste em obter o valor $\rho(t_{n+1})$ a partir da inclinação β, calculada no meio do intervalo, ou seja

$$\rho(t_{n+1}) = \rho(t_n) + \beta\Delta t. \tag{32}$$

Neste método também é possível utilizar ordens maiores, calculando-se mais inclinações ao longo do intervalo Δt e considerando uma média ponderada entre elas. Dessa forma obtém-se resultados mais precisos. No entanto, para os propósitos deste trabalho o método de segunda ordem fornece resultados bastante satisfatórios.

Queremos agora aplicar o método de RK especificamente para resolver numericamente o sistema de equações diferenciais acopladas (dadas pelas equações (9), (17) e (20)) que modelam o PDR. Sistematicamente podemos escrever o sistema de equações

diferenciais da seguinte maneira

$$\frac{d}{dt}\rho_X(t) = -\lambda_X \rho_X(t) \quad (33)$$

$$\frac{d}{dt}\rho_Y(t) = -\lambda_Y \rho_Y(t) + \lambda_X \rho_X(t) \quad (34)$$

$$\frac{d}{dt}\rho_Z(t) = \lambda_Y \rho_Y(t). \quad (35)$$

Por simplicidade de notação, note que todas essas equações são da forma

$$\frac{d}{dt}\rho_i = f_i(t, \rho_X, \rho_Y, \rho_Z), \quad (36)$$

com $i = X$, Y e Z.

Na equação (33) a função $f_X(t, \rho_X, \rho_Y, \rho_Z)$ depende explicitamente somente de ρ_X e assume a forma $f_X(t, \rho_X) = -\lambda_X \rho_X(t)$. Aplicar o método de RK para resolver a equação (33) significa calcular as quantidades abaixo para todos os valores de n, a partir da condição inicial $n = 0$.

$$\alpha_X = f_X(t_n, \rho_X) = -\lambda_X \rho_X(t_n) \quad (37)$$

$$\rho_X(t_m) = \rho_X(t_n) + \frac{\Delta t}{2}\alpha_X \quad (38)$$

$$\beta_X = f_X(t_m, \rho_X) = -\lambda_X \rho_X(t_m) \quad (39)$$

$$\rho_X(t_{n+1}) = \rho_X(t_n) + \Delta t \beta_X. \quad (40)$$

Aplicamos o método de RK na equação (34). Nesse caso, a função $f_Y(t, \rho_X, \rho_Y, \rho_Z)$ assume a forma $f_Y(t_n, \rho_X, \rho_Y) = -\lambda_Y \rho_Y(t) + \lambda_X \rho_X(t)$. Com isso, obtemos

$$\alpha_Y = f_Y(t_n, \rho_X, \rho_Y) = -\lambda_Y \rho_Y(t_n) + \lambda_X \rho_X(t_n) \quad (41)$$

$$\rho_Y(t_m) = \rho_Y(t_n) + \frac{\Delta t}{2}\alpha_Y \quad (42)$$

$$\beta_Y = f_Y(t_m, \rho_X, \rho_Y) = -\lambda_Y \rho_Y(t_m) + \lambda_X \rho_X(t_m) \quad (43)$$

$$\rho_Y(t_{n+1}) = \rho_Y(t_n) + \Delta t \beta_Y. \quad (44)$$

Note que para calcular α_Y e β_Y, nas equações (41) e (43), respectivamente, é necessário conhecer os valores de $\rho_X(t)$, calculados anteriormente. Isso sugere que o método de RK deve ser aplicado simultaneamente para todas as densidades.

Finalmente, aplicamos o método de RK na equação (35). Nesse caso, temos que $f_Z(t, \rho_Y) = \lambda_Y \rho_Y(t)$. Com isso, obtemos

$$\alpha_Z = f_Z(t_n, \rho_Y) = \lambda_Y \rho_Y(t_n) \quad (45)$$

$$\rho_Z(t_m) = \rho_Z(t_n) + \frac{\Delta t}{2}\alpha_Z \quad (46)$$

$$\beta_Z = f_Z(t_m, \rho_Y) = \lambda_Y \rho_Y(t_m) \quad (47)$$

$$\rho_Z(t_{n+1}) = \rho_Z(t_n) + \Delta t \beta_Z. \quad (48)$$

Novamente, observe que para calcular α_Z e β_Z, nas equações (45) e (47), respectivamente, é necessário antes calcular os valores $\rho_Y(t)$.

Resumindo, para aplicar o método de RK de segunda ordem no sistema de EDOs acopladas que regem esse modelo, é preciso computar, para cada intervalo de tempo, os valores intermediários $\rho_X(t_m)$, $\rho_Y(t_m)$ e $\rho_Z(t_m)$, os quais, por sua vez, são obtidos a partir dos valores dessas funções no início do intervalo considerado, isto é, $\rho_X(t_n)$, $\rho_Y(t_n)$ e $\rho_Z(t_n)$. A seguir apresentamos o algoritmo utilizado para se obter as soluções descritas nas equações anteriores.

4. Algoritmo Runge-Kutta

Nesta seção, apresentamos o algoritmo utilizado para o método de RK. Os detalhes de programação e de implementação desse algoritmo são bastante particulares da linguagem de programação utilizada e não são importantes nesse momento. Na seção 8 fazemos uma discussão sobre diferentes formas de se implementar esse algoritmo.

Os algoritmos serão apresentados na forma de fluxograma de programação. Nesse ponto é importante mencionar uma convenção básica para a leitura desses fluxogramas. Usualmente, utiliza-se formas diferentes para representar a função de cada trecho do algoritmo conectadas por setas. As funções são, basicamente do tipo: início/fim; entrada de dados; saída de dados; processamento; tomada de decisão. A Figura 4 ilustra as formas utilizadas de acordo com a função de cada trecho do algoritmo.

Neste problema, as variáveis de entrada, que precisam ser especificadas, são listadas abaixo:

- t_0 representa o instante inicial
- t_f representa o instante final
- dt representa o intervalo de tempo considerado

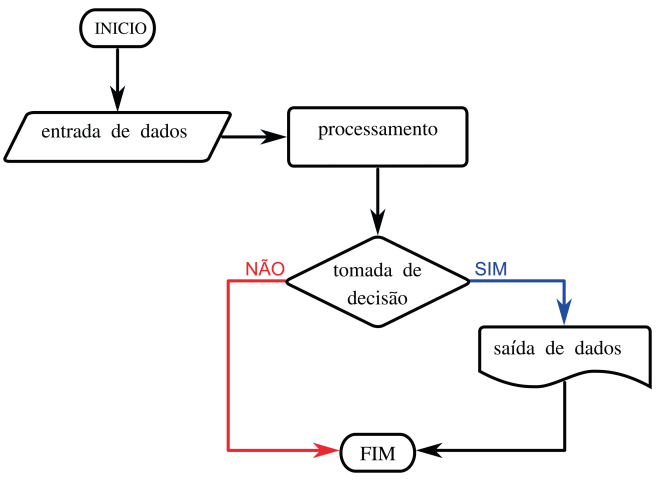

Figura 4: uso das formas para melhor representar um fluxograma de programação.

 DOI: https://doi.org/10.1590/1806-9126-RBEF-2023-0347

- $t_{1/2}^X$ e $t_{1/2}^Y$ são os tempos de meia-vida dos átomos X e Y, respectivamente

Todas as outras variáveis nesse algoritmo são obtidas a partir desses parâmetros de entrada. Ao final desse algoritmo teremos uma tabela de dados contendo N linhas de informações, uma para cada instante de tempo dt. Cada linha de dados, por sua vez, carrega quatro valores: t_n, $\rho_X(t_n)$, $\rho_Y(t_n)$ e $\rho_Z(t_n)$. Com essa tabela de dados podemos confeccionar um gráfico das densidades como função do tempo que poderá ser comparado com o gráfico da Figura 1. A Figura 5 apresenta o fluxograma do algoritmo utilizado no método de RK de segunda ordem.

É válido mencionar que todos os métodos de integração numérica apresentados até aqui são métodos aproximados, cujos erros seguem leis de potência em relação ao tamanho do passo Δt. No caso do método de Euler, o erro local (erro por passo) é quadrático, enquanto o erro global é da ordem do tamanho do passo. No método de RK de segunda ordem o erro local é cúbico, enquanto o global é de ordem quadrática. Para uma discussão mais detalhada sobre a quantificação de erros nesses métodos de integração numérica veja a referência [25].

5. Simulação de Monte Carlo

Simulações estocásticas são meios estatísticos que permitem modelar a evolução temporal de um sistema a partir de um conjunto de variáveis aleatórias. Em particular,

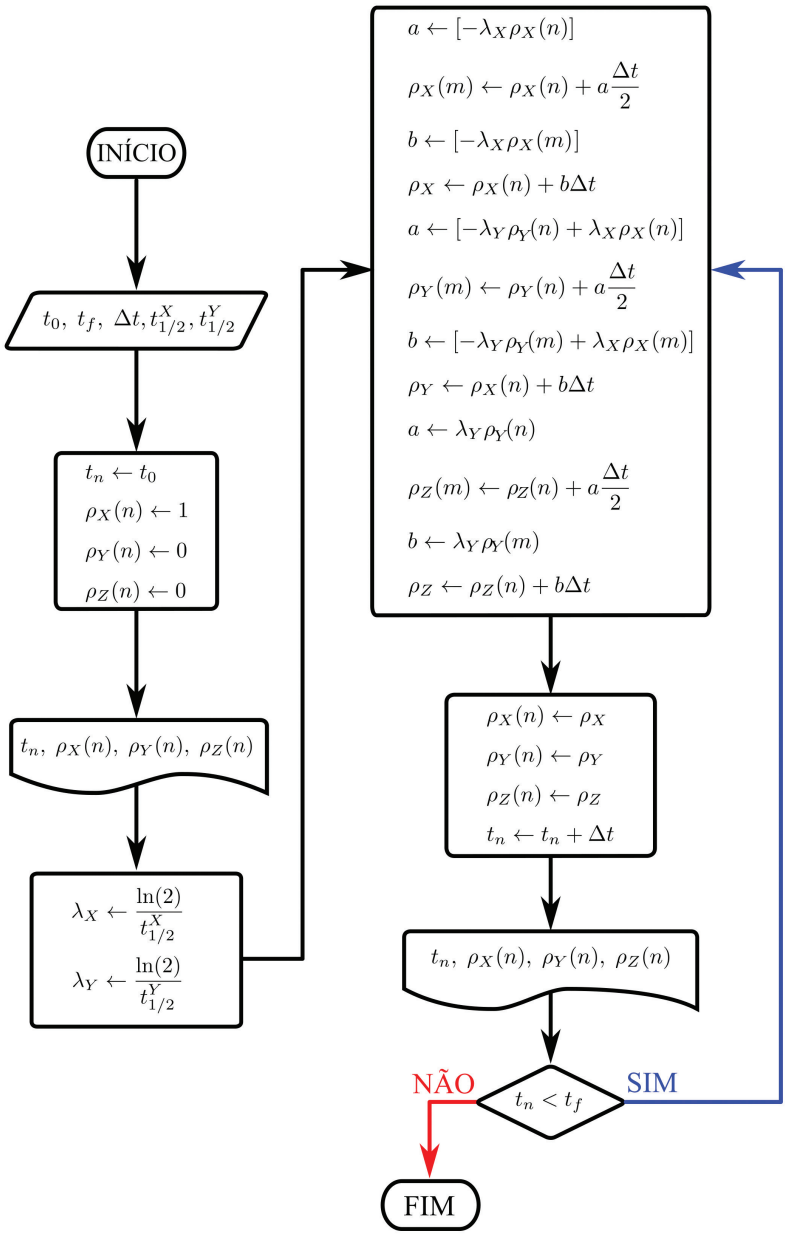

Figura 5: Fluxograma de programação do método de Runge-Kutta de segunda ordem.

o método computacional mais utilizado é o chamado método de Monte Carlo (MC), que consiste em gerar cenários a partir de distribuições de probabilidade com o intuito de se obter resultados numéricos [26].

Nesta seção, abordaremos o problema do decaimento radioativo a partir de uma simulação da evolução temporal do modelo utilizando o método de MC. Este método consiste em simular a evolução das densidades como uma função do tempo para alguns valores discretizados da variável t. A discretização que usamos aqui é a mesma apresentada na seção 3 para o método de RK. Assim, cada passo da simulação de MC representa um instante de tempo t_n e são chamados de passos de MC.

O intervalo de tempo de um passo de MC é Δt. Nesse intervalo de tempo alguns átomos do sistema poderão decair, com probabilidade dada pela equação (7). No início de cada passo as quantidades de átomos X, Y e Z são, respectivamente, N_X, N_Y e N_Z. Um número aleatório (de distribuição uniforme no intervalo [0,1[) é sorteado para cada átomo que possa vir a decair. Esses números são comparados com as probabilidades de decaimento de cada elemento. Sendo o número aleatório menor que a probabilidade de decaimento, o elemento decai, caso contrário, nada ocorre. Ao final do passo de MC as novas quantidades de átomos serão $N_X \leftarrow N_X - c_X$, $N_Y \leftarrow N_Y + c_X - c_Y$ e $N_Z \leftarrow N_Z + c_Y$, sendo c_X e c_Y as quantidades de átomos X e Y, respectivamente, decaídos nesse intervalo de tempo.

Ao todo serão realizados N passos de MC, com N dado pela equação (25), o que significa que ao fim dessa simulação teremos uma tabela de dados semelhante àquela obtida no método de RK, isto é, uma tabela de dados contendo N linhas, com cada linha carregando os valores $t_n, \rho_X(t_n)$, $\rho_Y(t_n)$ e $\rho_Z(t_n)$.

Na próxima seção apresentamos o algoritmo utilizado a fim de implementar essa simulação.

6. Algoritmo Monte Carlo

Nesta seção discutiremos com mais detalhes o algoritmo utilizado para realizar a simulação de MC descrita na seção anterior. A forma como apresentaremos o algoritmo será também na forma de fluxograma de programação. As variáveis de entrada para essa simulação são praticamente as mesmas usadas no método de RK, além da quantidade total de átomos, conforme listadas a seguir.

- N_0 é a quantidade total de átomos no sistema
- t_0 representa o instante inicial
- t_f representa o instante final
- dt representa o intervalo de tempo considerado
- $t^X_{1/2}$ e $t^Y_{1/2}$ são os tempos de meia-vida dos átomos X e Y, respectivamente

O fluxograma de programação da simulação de MC é apresentado na Figura 6.

7. Resultados dos Métodos Computacionais

Nesta seção, apresentamos os resultados obtidos a partir da aplicação dos métodos computacionais discutidos nas seções anteriores. Nas abordagens computacionais, os valores numéricos que parametrizam os modelos precisam ser especificados. Fazemos isso atribuindo valores para as variáveis de entrada utilizadas em cada algoritmo. Nesse trabalho, esses parâmetros foram escolhidos de modo a representar o decaimento $^{210}_{83}\text{Bi} \rightarrow ^{210}_{84}\text{Po} \rightarrow ^{206}_{82}\text{Pb}$, por isso, os tempos de meia-vida $t^X_{1/2}$ e $t^Y_{1/2}$ são fixados nos valores $t^X_{1/2} = 120\text{h}$ e $t^Y_{1/2} = 3321,12\text{h}$, conforme a referência [24]. O instante inicial é tomado como sendo $t_0 = 0$, enquanto o instante final é escolhido como sendo $t_f = 30.000\text{h}$. Por fim, o intervalo de tempo Δt foi escolhido como sendo $\Delta t = 1\text{h}$. Note que esse intervalo é muito inferior ao tempo de meia-vida dos átomos envolvidos, e sendo assim todas as aproximações feitas anteriormente levarão a excelentes resultados. A quantidade total de átomos foi fixada em $N_0 = 10.000$.

Implementando o método de RK conforme descrito pelo algoritmo da Figura 5 e com os parâmetros especificados anteriormente, geramos um conjunto de dados que pode ser visualizado na Figura 7. É importante frisar que para essa escolha dos parâmetros o algoritmo do método de RK gera uma tabela de dados contendo 30.000 pontos para cada função densidade. Para garantir melhores resultados, é muito importante que os métodos computacionais possuam uma grande quantidade de iterações. No entanto, para fins de visualização, não há necessidade de gerar um gráfico contendo essa quantidade de pontos. Por esse motivo, escolhemos alguns pontos igualmente espaçados na escala logarítmica (da ordem de 100 pontos) dentro do conjunto de dados, para confeccionar o gráfico apresentado na Figura 7. Essa escolha de "descartar" alguns dados não interfere na visualização dos resultados, mas certamente interfere na análise estatística e no ajuste de curvas quando é o caso.

Analisando o gráfico da Figura 7 é possível perceber que o método de RK fornece um resultado em concordância com o método analítico do problema do decaimento radioativo. Na Figura 7 os dados obtidos a partir do método de RK são representados pelos pontos, enquanto as curvas sólidas representam as soluções analíticas apresentadas na Figura 1 para os parâmetros escolhidos.

Implementando a simulação de MC conforme descrito pelo algoritmo da Figura 6 e com os parâmetros especificados anteriormente, geramos um conjunto de dados que pode ser visualizado na Figura 8. Mais uma vez, assim como na confecção do gráfico da Figura 7 e pelo mesmo motivo, escolhemos somente alguns pontos do conjunto de dados para ilustrar o resultado mostrado na Figura 8.

 DOI: https://doi.org/10.1590/1806-9126-RBEF-2023-0347

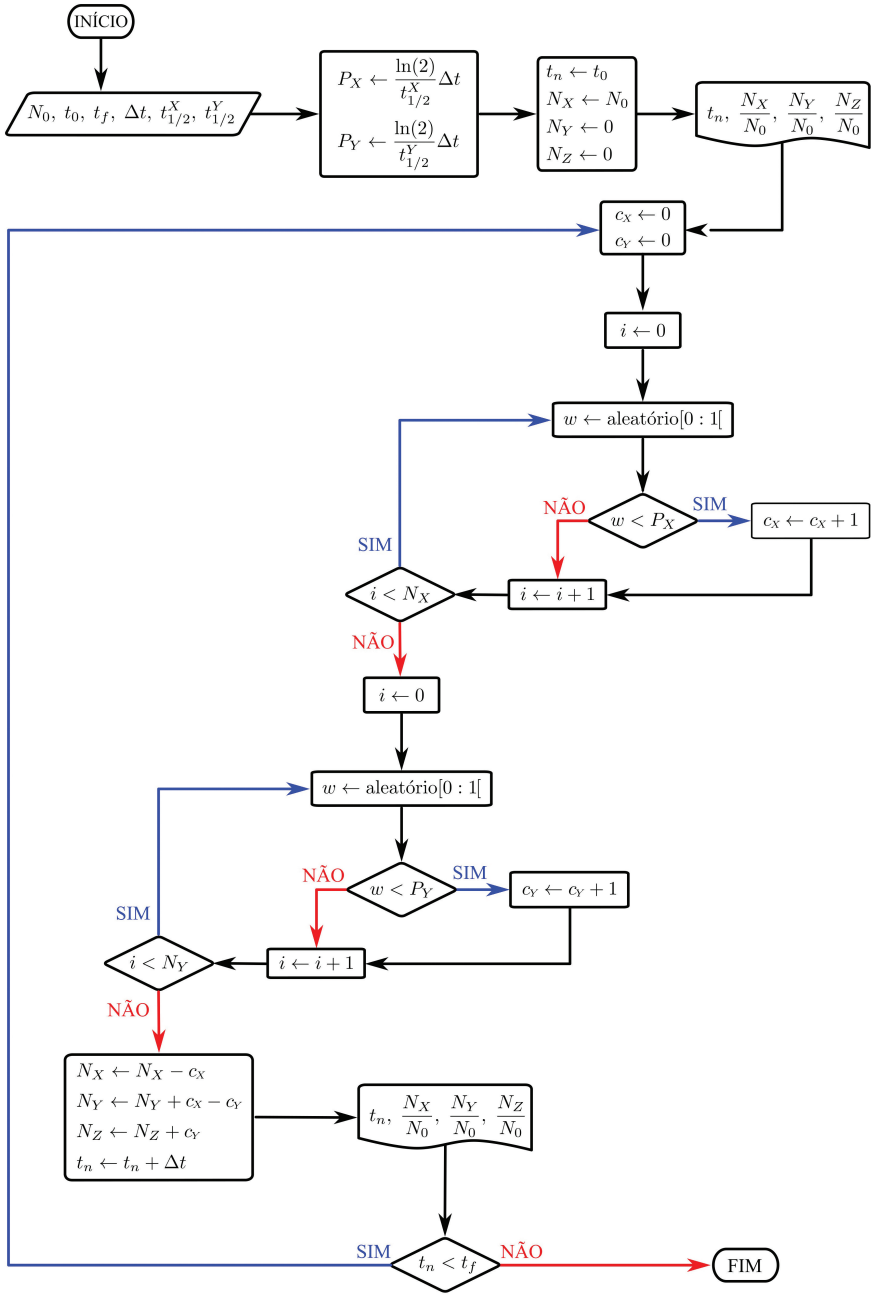

Figura 6: Fluxograma de programação da estrutura básica do algoritmo da simulação de Monte Carlo.

O método de RK é um método determinístico, uma vez que sempre gerará o mesmo resultado partindo-se dos mesmos parâmetros e das mesmas condições iniciais. O método de MC faz uso de números aleatórios, não sendo um método determinístico. É válido ressaltar que, como o computador é determinístico, é necessário partir de uma condição inicial (semente) para a geração de números denominados pseudoaleatórios. Apesar de não serem estritamente aleatórios, esses números passam por critérios estatísticos que validam a sua aleatoriedade. Neste sentido, fazendo uso da mesma semente no método de MC, um resultado idêntico será obtido. Entretanto, para diferentes sementes o resultado, na média, será próximo do obtido pelo método determinista de RK.

Comparando os gráficos apresentados nas Figuras 7 e 8 podemos perceber que ambos os métodos são muito eficazes na resolução do PDR. Em geral, o método de RK é mais rápido e requer menor poder computacional para ser executado, envolvendo um número de iteração igual a quantidade N de intervalos de tempo considerados ($N = 30.000$ nesse caso). Porém, esse método só pode ser utilizado quando as equações diferenciais que modelam o problema são conhecidas e de primeira ordem. Lembrando que, em alguns casos, é possível reduzir EDOs

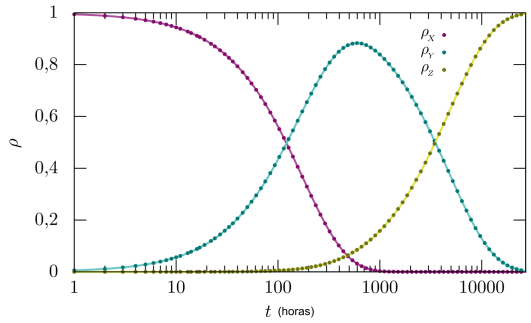

Figura 7: Densidades de átomos em função do tempo obtidas pelo método de Runge-Kutta de segunda ordem (pontos) contrastada com as soluções analíticas (linhas sólidas) mostradas na Figura 1.

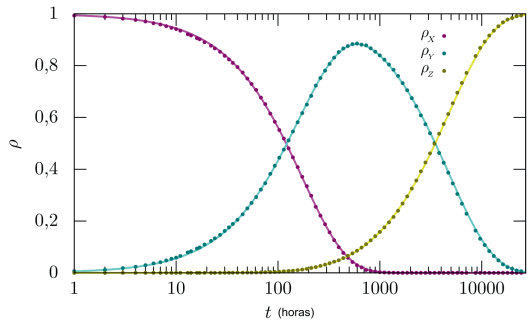

Figura 8: Densidades de átomos em função do tempo obtidas a partir do método de Monte Carlo (pontos) contrastada com as soluções analíticas (linhas sólidas) mostradas na Figura 1.

de ordem superiores a sistemas de EDOs acopladas de primeira ordem. O método de MC é, em geral, mais lento e requer maior poder computacional para ser executado. Dependendo da complexidade do problema, as estruturas dos algoritmos podem incluir um número elevado de laços de repetição, o que torna a computação mais lenta. No entanto, o método de MC pode ser utilizado mesmo quando as equações diferenciais que regem a dinâmica do modelo não são conhecidas. Por esse motivo, representa um método numérico versátil, podendo ser utilizado na resolução de uma vasta gama de problemas.

8. Implementação dos Algoritmos

Neste trabalho, optamos por dar foco aos algoritmos de cada método computacional apresentado sem nos preocuparmos muito com a linguagem de programação utilizada na implementação desses algoritmos. Tudo que foi apresentado até aqui é independente de linguagem de programação e pode, pelo menos, em princípio, ser implementado usando qualquer linguagem de programação. Nessa seção, apresentamos alguns exemplos de implementações possíveis para os algoritmos apresentados em algumas linguagens de programação específicas.

É importante frisar que os algoritmos apresentados podem ser implementados, inclusive, sem o uso de computadores. O método de RK, por exemplo, pode ser aplicado manualmente. Basta subdividir o intervalo em consideração em algumas poucas partes e proceder com os cálculos feitos à mão. Simulações de MC também podem ser feitas sem o uso de computadores. O único instrumento necessário para uma simulação de MC é um gerador de números aleatórios, que poderia ser um conjunto de números escritos em pequenos pedaços de papel, dentro de uma jarra, onde os números possam ser sorteados. É evidente que, feitos dessa maneira, esses métodos não seriam muito eficazes do ponto estatístico, pois se tornam processos muito inviáveis à medida que buscamos mais precisão dos resultados. Por outro lado, a implementação manual certamente é uma ótima maneira de aprender sobre algoritmos. Do ponto de vista da pesquisa científica, a precisão dos resultados é fundamental, sendo assim apresentamos a seguir algumas linguagens de programação que podem ser utilizadas na implementação dos algoritmos estudados neste trabalho. Todos os códigos mencionados a seguir estão disponíveis no repositório GitHub por meio do *link* em [27].

8.1. Mathematica

O Mathematica, além de uma linguagem de programação, é um *software* bastante poderoso de álgebra computacional. Suas utilizações são ilimitadas, e possui diversas bibliotecas prontas para serem utilizados em diferentes contextos e áreas da ciência.

A programação em Mathematica apresenta uma sintaxe bastante simples e flexível. Do ponto de vista didático, a transcrição do fluxograma para um código em Mathematica pode ser ensinada com relativa facilidade. O próprio *software* acusa e identifica os erros de sintaxe, permitindo a correção rápida de eventuais falhas do código. Outro ponto forte do Mathematica é a possibilidade de confeccionar os gráficos, com os resultados obtidos nos métodos computacionais, dentro do próprio *software*.

O algoritmo de Runge-Kutta, apresentado no fluxograma da Figura 5, pode ser implementado no Mathematica por meio do código no arquivo RK.nb. Esse código leva em torno de $4,3$ segundos para ser executado. É possível também, utilizando algumas funções próprias do Mathematica, ao invés de seguir o fluxograma da Figura 5, reescrever o código a fim de obter mais eficiência, conforme pode ser visto no arquivo RK2.nb. Esse código leva em torno de $0,1$ segundos para ser executado.

O algoritmo de Monte Carlo, apresentado na Figura 6, pode ser implementado por meio do arquivo MC.nb. Esse código é mais lento que os anteriores, e leva cerca de 80 segundos para ser executado. Mais uma vez, usando funções próprias do Mathemática é possível usar o código do arquivo MC2.nb para tornar o processo mais rápido. Nesse caso o tempo de execução cai pela metade.

 DOI: https://doi.org/10.1590/1806-9126-RBEF-2023-0347

Uma das desvantagens do Mathematica é o tempo de execução em simulações. Um código simples como o MC.nb, por exemplo, leva consideravelmente mais tempo para ser executado que o mesmo código escrito em linguagem C ou Python, conforme veremos a seguir.

8.2. Linguagem C

A linguagem C é uma linguagem de programação de nível intermediário, combinando elementos de linguagens de programação mais avançadas com a funcionalidade da linguagem a nível de máquina. Os códigos escritos em C precisam ser primeiramente compilados por um compilador para só depois serem executados.

O arquivo RK.c carrega o código que implementa o algoritmo de Runge-Kutta em linguagem C, o qual segue a estrutura do fluxograma da Figura 6. O tempo de execução desse código é de aproximadamente $0,2$ segundos. O arquivo MC.c, é o código do algoritmo de Monte Carlo, também implementado em linguagem C. O tempo de execução desse código é de aproximadamente $1,2$ segundos.

É importante destacar a rapidez dos códigos em C que envolvem simulações de sistemas grandes, comparada ao mesmo código feito em Mathematica, por exemplo. Porém, comparando os códigos, pode-se notar que o código escrito em linguagem C é um pouco mais complicado que o código em Mathematica. Na linguagem C todas as variáveis envolvidas precisam ser declaradas. O código precisa então ser compilado e executado. Outro detalhe é que não há um verificador de sintaxe, o que significa que em caso de falha no código o usuário precisa encontrá-lo e corrigi-lo manualmente. Do ponto de vista da pesquisa acadêmica em física computacional, muitas vezes a linguagem C é mais indicada por ser mais eficaz e permitir utilizar toda a capacidade de processamento do computador.

Por fim, o programa em linguagem C gera um conjunto de dados. Esses dados são importados para o *software* livre Gnuplot para gerar os gráficos.

8.3. Linguagem Python

A linguagem Python é uma linguagem de programação de alto nível, e por esse motivo mais amistoso ao usuário. Python tem se tornado uma das linguagens de programação mais populares nas últimas décadas devido a sua grande versatilidade.

O arquivo RK.py traz o código que implementa o algoritmo do método de Runge-Kutta em linguagem Python. Esse código segue a estrutura do fluxograma da Figura 5. O tempo de execução desse código é de aproximadamente $0,2$ segundos. O arquivo MC.py, que é o código do algoritmo de Monte Carlo, implementado em linguagem Python, leva cerca de $6,7$ segundos para ser executado.

Os códigos em Python são bastante parecidos com os códigos em C, mas com a vantagem que a sintaxe é mais fluida que na linguagem C. Além disso, Python é uma linguagem interpretada, ou seja, não é necessário a sua compilação.

A Tabela 1 mostra os tempos médios de execução de cada código nas diferentes linguagens de programação.

É evidente que o algoritmo de Runge-Kutta é muito mais rápido que o algoritmo de Monte Carlo, e isso pode ser notado em todas as implementações apresentadas. O algoritmo de Monte Carlo tem um maior custo computacional. Para que o método de MC reproduza os resultados analíticos de forma satisfatória, o número de partículas envolvidas na simulação dever ser suficientemente grande (acima de 10^4 partículas), elevando assim o custo computacional. O Mathematica não se mostrou tão eficiente para esse tipo de simulação, enquanto que as linguagens C e Python parecem mais indicadas para esse tipo de tarefa. É importante mencionar que os códigos disponibilizados [27] seguem as estruturas dos fluxogramas apresentados nas Figuras 5 e 6. Entretanto, nas linguagens Python e Mathematica é possível otimizar os códigos fazendo uso de sorteios de números pseudoaleatórios em lotes, diferentemente dos sorteios sequenciais que foram utilizados nesse trabalho.

9. Considerações Finais

Neste trabalho, apresentamos dois métodos computacionais bastante importantes: a resolução numérica e as simulações estocásticas. Conforme podemos notar nas Figuras 7 e 8, ambos os métodos fornecem resultados excelentes para modelar o problema do decaimento radioativo e é muito difícil decidir sobre a eficiência de cada método olhando somente para os resultados obtidos. As diferenças entre esses dois métodos são bastante relevantes de um ponto de vista mais técnico.

A resolução numérica, por exemplo, é muito mais rápido de ser executado que as simulações estocásticas (cerca de cinco vezes mais rápido para a escolha de parâmetros utilizada). Isso só não é válido, entanto, para todos os sistemas. O motivo para essa diferença é que o método de RK pode ser aplicado diretamente às EDOs que regem a dinâmica do modelo e assim obtemos estimativas sobre as funções de densidade de cada átomo. Para as simulações estocásticas, é necessário simular um sistema físico contendo uma quantidade suficientemente grande de átomos (10000 átomos nesse

Tabela 1: Tempo médio de execução dos códigos em diferentes linguagens de programação. Esses códigos foram executados em um computador pessoal com as seguintes especificações: processador Intel Core i5-8260U, 16Gb de memória RAM, 256Gb de armazenamento SSD.

código	tempo (s)
RK.nb	4,3
RK2.nb	0,1
RK.c	0,3
RK.py	0,2

código	tempo (s)
MC.nb	80,0
MC2.nb	40,0
MC.c	1,2
MC.py	6,7

caso) a fim de obter resultados razoáveis. Sendo assim, os laços de programação envolvidos nas simulações de MC precisam ser repetidos, para cada passo de MC, uma quantidade de vezes que é igual a quantidade de átomos presentes no sistema, tornando a simulação mais lenta.

Por outro lado, as simulações de MC possuem uma vantagem sobre a resolução numérica nos casos em que não existe (ou não se conhece) um sistema de EDOs que regem a dinâmica do modelo. Além disso, o método de MC pode ser utilizado para resolver problemas multidimensionais, como a resolução numérica de integrais que não podem ser resolvidas analiticamente ou quando as funções primitivas não são conhecidas, como é o caso da distribuição gaussiana no cálculo da função erro [28]

Agradecimentos

Este trabalho foi financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq, Grants 304544/2019-1 e 309835/2022-4 (BFO)), Fundação Araucária e INCT-FCx (CNPq/FAPESP).

Referências

[1] J.F. Boudreau e E.S. Swanson, *Applied Computational Physics* (Oxford University Press, Oxford, 2018), v. 1.

[2] R.H. Landau, M.J. Páez e C.C. Bordeianu, *A Survey of Computational Physics: Introductory Computational Physics* (Princeton University Press, Princeton, 2008).

[3] J. Weber e T. Wilhelm, European Journal of Physics **41**, 034003 (2020).

[4] R.F. Martin, Journal of Physics: Conference Series **759**, 012005, 2016.

[5] J.C.C. Palacio, L.V. Abad, E.C. Madera e J.A. Monsoriu, Revista Brasileira de Ensino de Física **33**, 3313 (2011).

[6] S. López, E.A. Veit e I.S. Araujo, Revista Brasileira de Ensino de Física **38**, e2401 (2016).

[7] R. Landau, Computing in Science and Engineering **8**, 22 (2006).

[8] R.H. Landau, Computer Physics Communications **177**, 191 (2007).

[9] K.R. Roos, Computing in Science and Engineering **8**, 44 (2006).

[10] F.J. Dellajustina e L.C. Martins, Revista Brasileira de Ensino de Física **36**, 3305 (2014).

[11] R.S.R. Neto, F.M.L. Souza, A.L.E. Fidelis, A.M. Rocha, L.G.O. Santana, L.A.R. Rosa e S.C. Cardoso, Revista Brasileira de Ensino de Física **45**, e20220231 (2023).

[12] M.A. Segura, J. Salamanca e E. Munevar, Revista Brasileira de Ensino de Física **39**, e1504 (2016).

[13] M.A. Reis e S.A. Vitiello, Revista Brasileira de Ensino de Física **28**, 45 (2006).

[14] M.C. Capizzo, R.M. Sperandeo-Mineo e M. Zarcone, European Journal of Physics **29**, 451 (2008).

[15] B. Coto, A. Arencibia e I. Suárez, Computer Applications in Engineering Education **24**, 765 (2016).

[16] D.R. Stump, American Journal of Physics **54**, 1096 (1986).

[17] C. Körber, I. Hammer, J.L. Wynen, J. Heuer, C. Müller e C. Hanhart, Physics Education **53**, 055007 (2018).

[18] K. Atkin, Physics Education **54**, 025012 (2019).

[19] R.D.S. Acosta, E.M.H. Cooper, A.H. Medina e R.G. Castillo, Journal of Physics: Conference Series **1936**, 012017 (2021).

[20] E. Okuno, *Radiação: efeitos, riscos e benefícios* (Oficina de Textos, São Paulo, 2018).

[21] L. Bueno, *Análise de radionuclídeos naturais e chumbo em produtos alimentícios e dietas*. Dissertação de Mestrado, Instituto de Pesquisas Energéticas e Nucleares, São Paulo (1999).

[22] A.A. Souza e M.H.S. Passos, *Quimica Nuclear E Radioatividade* (Átomo, Campinas, 2012).

[23] W.E. Boyce e R.C. Diprima, *Elementary Differential Equations* (John Wiley and Sons, Chichester, 2008), 9 ed.

[24] IFUSP, *Tabela periódica dos elementos*, disponível em: http://www.dfn.if.usp.br/gd_dfn/periodico/, acessado em 19/06/2023.

[25] M.E.J. Newman, *Computational Physics* (CreateSpace Independent Publishing Platform, Scotts Valley, 2013).

[26] D.P. Kroese, T. Brereton, T. Taimre e Z.I. Botev, WIREs Computational Statistics **6**, 386 (2014).

[27] GITHUB, *Decaimento radioativo (códigos)*, disponível em: https://github.com/migueljbf/decaimento_radioativo_codes, acessado em: 21/06/2023.

[28] L.C. Andrews, *Special Functions of Mathematics for Engineers* (SPIE Optical Engineering Press, Bellingham, 1998).