Farmaco-cinética

Relatório de Métodos Numéricos

Pedro Miguel Jesus da Silva (up201907523)
Joel Alexandre Vieira Fernandes (up201904977)
Francisco José Barbosa Marques Colino (up201905405)
Grupo 55

Dezembro 2020 - Janeiro 2021

Conteúdo

1 Introdução

Dado o fármaco Diclofenac na forma de tablete, foi-nos pedido para estudar a concentração do mesmo no corpo para um tratamento de 5 dias com uma dose de 75mg de 12 a 12 horas (150 mg por dia). Para isto recorreu-se ao modelo bi-compartimental, que considera dois grandes compartimentos: o compartimento central que representa o meio pelo qual o fármaco entra no organismo (incluindo as vias de administração e de incorporação gastro-intestinal, transdermal ou pulmonar) e, o segundo compartimento representado pelo plasma sanguíneo. O modelo segue portanto um sistema de equações diferenciais ordinárias:

$$\begin{cases} \frac{dm_i}{dt} = D(t) - K_a m_i \\ \frac{dm_p}{dt} = K_a m_i - K_{et} m_p \end{cases}$$

Em que m_p e m_i são as concentrações do fármaco no compartimento plasmático e central, respetivamente; k_{et} é a constante cinética de eliminação total; k_a é a constante cinética de absorção e $D(t)$ é a função de administração Outro dado relevante é o t_{max} que diz o tempo, depois da toma do medicamento, onde ocorre a máxima concentração sanguínea do fármaco. As constantes dadas foram $k_{et} = 0.17325$ h^{-1} *e* $\mathrm{t}_{max} = 4$ h. É preciso portanto obter o k_a e modelar o $D(t)$ para resolver o sistema.

Para resolver, numericamente, o problema na sua globalidade, usou-se os métodos apropriados para obter raízes de funções de variável real e métodos para resolver sistemas de equações diferenciais ordinárias, todos eles abordados em aula. A metodologia foi programar os métodos, eficientemente (com python3), estudar, discutir e explicar a diferença ao nível dos resultados que ofereceram e avaliar a precisão e exatidão dos mesmos.

2 Cálculo do coeficiente de absorção cinético K_a

2.1 Overview do problema

Dada a equação:

$$K_a e^{-K_a t_{max}} - K_e e^{-K_e t_{max}} = 0;$$

repara-se que resolvê-la é o mesmo que resolver a equação $f(k_a) = 0$, ou seja, é descobrir as raízes reais da função f. A solução analítica $k_a = k_{et}$ é imediata visto que substituindo k_a na equação por k_{et} faz com os dois termos se anulem. Ora, isto não exclui a existência de outras soluções, logo foi preciso representar graficamente a função f. Para isto, correu-se um script de Máxima que fez plot à função f. Numa primeira fase, visto o gráfico da função, pode-se concluir que a função diverge para $-\infty$ quando k_a tende para $-\infty$ ($\lim_{k_a \to -\infty} f(k_a) = -\infty$) e converge para um valor negativo quando tende para $+\infty$ ($\lim_{k_a \to +\infty} f(k_a) = -0.08663...$). Dito de outro modo, a partir de uns

dados valores, não é preciso procurar pela existência de zeros. Graficamente, em $x = 1$ e $x = 0$, a função já começa a divergir e a convergir, respetivamente, para os tais valores negativos. Assim sendo, podemos cortar o intervalo $(0, 1)$ para estudar a existência de zeros. Fazendo o isolamento das raízes através da análise do gráfico e recorrendo ao teorema de Bolzano, pode-se obter os intervalos $(0, 0.2)$ e $(0.2, 0.7)$ onde há uma e uma só raíz em cada intervalo, contendo o primeiro intervalo a raiz $k_a = k_{et}$.
Tendo exatamente uma raiz isolada num intervalo, pode-se recorrer a um método numérico para a descobrir visto que é uma equação difícil de resolver, analiticamente. Para isto, recorreu-se ao método da Bissecção, da Corda ou Falsa-Posição, da Corda modificado (método de Illinois), de Newton e de Picard-Peano, cada um oferecendo resultados diferentes dos outros. Para avaliar todos de maneira igualitária, usou-se como critério de paragem a anulação da função, $|f(x_n)| <= \epsilon$, que deu melhores resultados comparado com o critério da precisão absoluta, $|x_{n+1} - x_n| <= \epsilon$, isto é, fez em menos iterações e com mais precisão e exatidão, para o mesmo valor de ϵ. Foi testado para ϵ igual a $10^{-4}, 10^{-6}, 10^{-8}, 10^{-10}$ e 10^{-20} e a exatidão foi obtida com referência ao valor calculado pelo Máxima que é 0.3466320541629992.

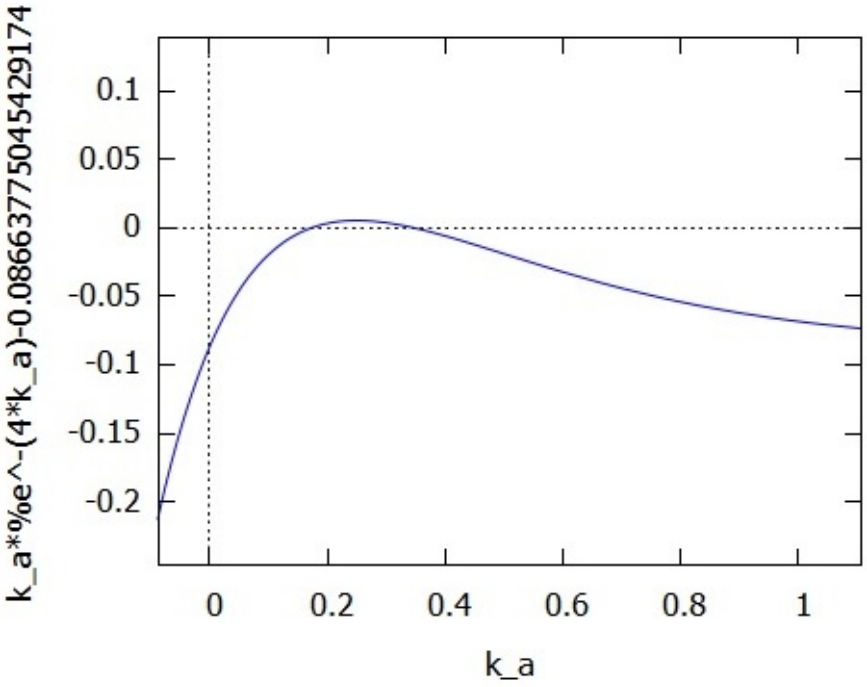

Fig 1 - Gráfico de $f(k_a)$ - visibilidade de duas raízes

2.2 Método de Picard-Peano

Tentou-se pôr $f(k_a) = 0$ na forma $k_a = g(k_a)$ para poder aplicar o algoritmo do método de Picard-Peano. A primeira função g obtida foi:

$$g_1(k_a) = k_{et}e^{(-k_{et}+k_a)t_{max}},$$

que verifica a condição de $|g'(k_a)| < 1$ apenas para o intervalo de (0, 0.265), o que faz não levar à desejada convergência. O que se verificou é que quando foi aplicado o método com um guess $x_0 = 0.3$ para tentar obter a raiz perto desse valor (a raiz diferente de k_{et}), o algoritmo convergiu para a outra raiz. Quando se tentou aumentar a primeira aproximação para x0 = 0.35, o algoritmo divergiu inteiramente como também estava previsto. Mudou-se então a função g para

$$g_2(k_a) = -log(\frac{k_{et}e^{-k_{et}t_{max}}}{k_a})/t_{max},$$

que já verificava a condição para aplicar o método de Picard-Peano sem problemas no intervalo (0,1).

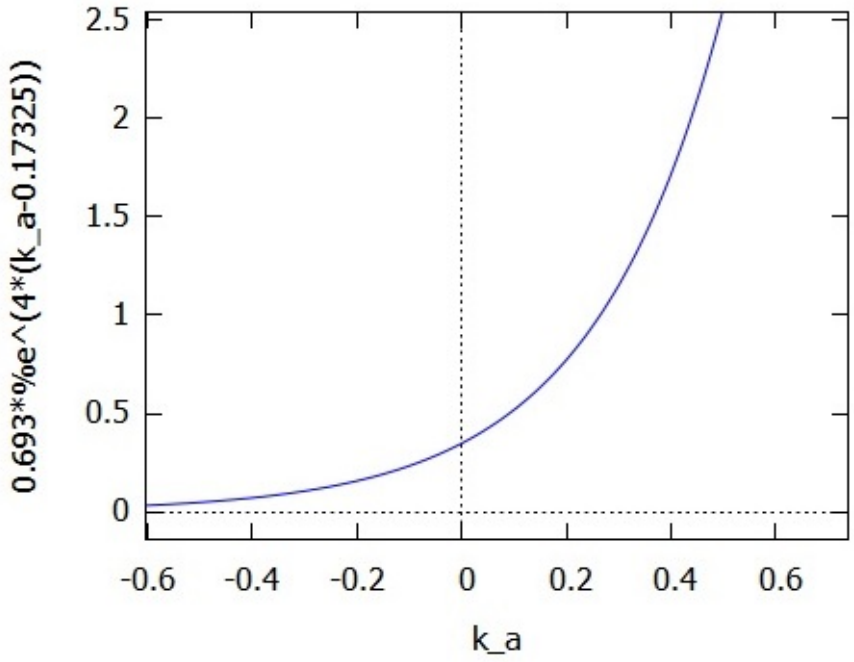

Fig 2 - Gráfico de $g_1'(k_a)$ - convergência comprometida

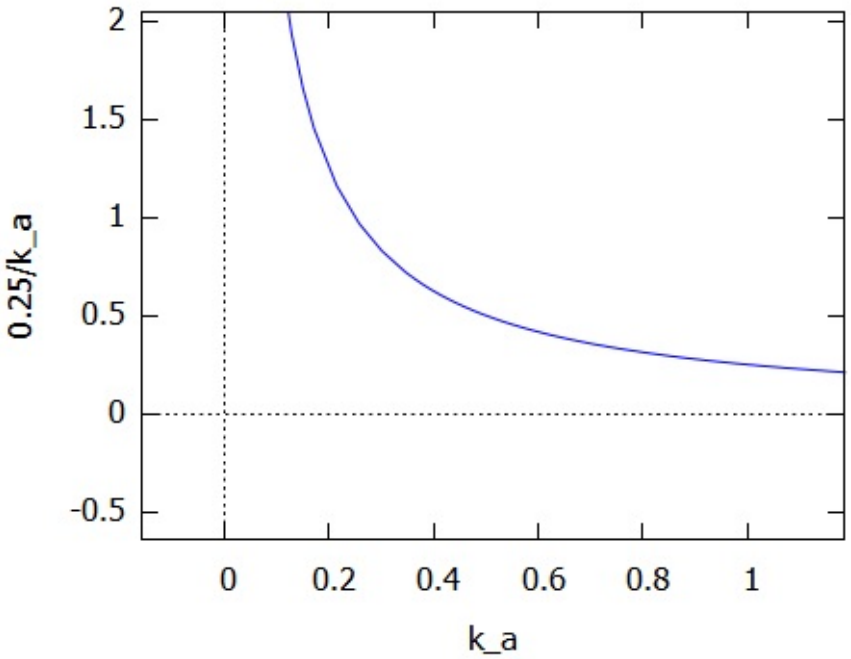

Fig 3 - Gráfico de $g_2'(k_a)$ - convergência "segura"

Executou-se então o algoritmo com o guess $x_0 = 0.5$ e obteve-se:

Picard-Peano				
precisão ϵ	iterações n	raiz k_a	erro absoluto $e(k_a)$	erro relativo $r(k_a)$
10^{-4}	14	0.34753008856150913	-8.98034399e-4	-2.59074251e-3
10^{-6}	28	0.34664126499484730	-9.21083185e-6	-2.65723603e-5
10^{-8}	42	0.34663214906522527	-9.49022261e-8	-2.73783757e-7
10^{-10}	56	0.34663205514085366	-9.77854464e-10	-2.82101569e-09
10^{-20}	105	0.34663205416299925	-5.55111512e-17	-1.60144310e-16

Dos 5 métodos testados, foi o que convergiu mais lentamente (em termos do número de iterações).

2.3 Método de Newton

Determinou-se , analiticamente, a derivada de f dado que o método necessita obtendo-se:

$$f'(k_a) = -k_a^2 t_{max} e^{-k_a t_{max}} + k_{et}^2 t_{max} e^{-k_{et} t_{max}},$$

que possui o zero $k_a = 0.25$, logo este não pode ser a primeira aproximação senão haverá divisão por zero no algoritmo. A função f e a derivada f' são ambas contínuas no seu domínio porque são somas de exponenciais, que são

contínuas, logo não há perigos. Formulou-se então a fórmula de recorrência $k_{an+1} = k_{an} + \frac{f(k_{an})}{f'(k_{an})}$ para aplicar o método de Newton, executou-se o algoritmo com o guess $x_0 = 0.5$ e obteve-se:

Newton				
precisão ϵ	iterações n	raiz k_a	erro absoluto $e(k_a)$	erro relativo $r(k_a)$
10^{-4}	11	0.34578990410255417	8.42150060e-4	2.42952159e-3
10^{-6}	20	0.34664160079468580	-9.54663169e-6	-2.75411104e-5
10^{-8}	30	0.34663211991158593	-6.57485867e-8	-1.89678323e-7
10^{-10}	39	0.34663205341805920	7.44939999e-10	2.14907995e-09
10^{-20}	71	0.3466320541629992	0.0	0.0

Surpreendentemente, o método de Newton foi o segundo a convergir mais lentamente, sendo que e o guess for 0.7 resulta na divergência do método para $-\infty$ mas para valores maiores que 0.75 já diverge para $+\infty$. Por análise do gráfico da função e da sua derivada percebe-se o porquê. Efetivamente, a derivada tem um ponto de inflexão em $k_a = 0.75$ e a função f é muito propensa a fazer os valores de k_a disparar para os dois sentidos porque para $k_a > 7$ a função é muito plana.

2.4 Método da Bissecção

Para o método da Bissecção usou-se o intervalo (0.2, 0.7) para obter a aproximação da raiz. Garante-se que $f(0.2)$ e $f(0.7)$ têm sinais opostos e como a função também é contínua, o método é aplicável. O intervalo foi excolhido de modo a que o guess do método de Picard-Peano e do método de Newton estivesse à mesma distância de cada um dos limites do intervalo. Obteve-se então:

Bissecção				
precisão ϵ	iterações n	raiz k_a	erro absoluto $e(k_a)$	erro relativo $r(k_a)$
10^{-4}	8	0.34578990410255417	8.42150060e-4	4.26040123e-4
10^{-6}	14	0.34663696289062496	-4.90872763e-6	-1.41612051e-5
10^{-8}	21	0.34663195610046380	9.80625354e-8	2.82900944e-7
10^{-10}	28	0.34663205482065673	-6.57657540e-10	-1.89727849e-9
10^{-20}	50	0.34663205416299925	-5.55111512e-17	-1.60144310e-16

2.5 Método da Corda (Regula-Falsi)

Para o método da Corda, fez-se nas mesmas condições que o método da Bissecção. Obteve-se então:

Corda				
precisão ϵ	iterações n	raiz k_a	erro absoluto $e(k_a)$	erro relativo $r(k_a)$
10^{-4}	6	0.34626343996849580	3.68614195e-4	1.06341635e-3
10^{-6}	9	0.34662781757174090	4.23659126e-6	1.22221566e-5
10^{-8}	12	0.34663200566767927	4.84953199e-8	1.39904315e-7
10^{-10}	15	0.34663205360790990	5.55089308e-10	1.60137904e-9
10^{-20}	26	0.3466320541629992	0.0	0.0

2.6 Método da Corda modificado (algoritmo Illinois)

Para o método da Corda modificado, fez-se nas mesmas condições que o método da Bissecção. Obteve-se então:

Corda modificado				
precisão ϵ	iterações n	raiz k_a	erro absoluto $e(k_a)$	erro relativo $r(k_a)$
10^{-4}	5	0.34656537998995240	6.66741730e-5	1.92348550e-4
10^{-6}	7	0.34663204096802486	1.31949743e-8	3.80662266e-8
10^{-8}	7	0.34663204096802486	1.31949743e-8	3.80662266e-8
10^{-10}	8	0.34663205416038910	2.61007882e-12	7.52982532e-12
10^{-20}	10	0.34663205416299925	-5.55111512e-17	-1.60144310e-16

2.7 Overview dos métodos

O método da Corda modificado produziu os melhores resultados em termos de iterações (convergiu sempre com menos iterações) e teve maioritariamente resultados mais precisos e exatos quando comparando com o valor do Máxima para a raiz. O método não estagnou como o Picard-Peano e o Newton por causa da forma da função nem ficou com convergência mais lenta como o método da Corda porque era sempre impedido que a corda ficasse muito vertical e, obviamente, como usa os valores da função nos extremos converge melhor que a Bissecção. Contudo, se o objetivo for arranjar o valor mais exato no menor número de iterações, deve-se usar o método da Corda porque, juntamente com o método de Newton (só que com mais iterações), consegue ter as 17 primeiras casas decimais corretas no python com $\epsilon = 10^{-20}$. Usa-se, portanto, esta aproximação da raiz, dada pelo método da Corda normal, para o resto do estudo.

3 Modelagem da função de administração $D(t)$

A função de administração $D(t)$ dá a quantidade de fármaco administrado por unidade do tempo como função do tempo. No estudo foi usado as unidades em mg/h e h. Dado que o fármaco é tomado na forma de comprimido, periodicamente, a função será semelhante aos dentes de uma serra, ou seja, terá que ser reproduzido um conjunto de triângulos, cujo período foi escolhido para 12 horas visto que o medicamento era tomado de 12 em 12 horas. Visto que após a ingestão do Diclofenar este só se encontra nos compartimentos 30 minutos após, é preciso acrescentar no início de cada cada triângulo uma linha horizontal com esse comprimento.
A análise dimensional diz-nos que o integral da função ou a área no gráfico num certo intervalo de tempo tem que ser igual à quantidade de fármaco tomado nesse intervalo de tempo, tal que, como se sabe a fórmula para área do triângulo, pode-se obter que:

$$m_{tomada} = \frac{(t_{toma} - t_{reação}) D(t)_{max}}{2}$$

Calculando o valor máximo de $D(t)$, sabendo que ocorre em t_{max} e tendo os outros dados é possível obter duas equacões de reta para os outros lados do triângulo. Fica-se portanto com a função definida por ramos:

$$D(t) = \begin{cases} 0 & 0 \leq t \leq t_{reação} \\ \frac{D(t)_{max}}{t_{max} - t_{reação}}(t - t_{reação}) & \mathrm{t}_{reação} < t \leq t_{max} \\ -\frac{D(t)_{max}}{t_{toma} - t_{max}}(t - t_{max}) + D(t)_{max} & t_{max} < t \leq t_{toma} \end{cases}$$

O python faz com que seja trivial a definição desta função através do uso do if-else statement. É de menção especial que em vez de se usar o operador módulo % para simular a periodicidade dos triângulos, recorreu-se a um simples ciclo while que decrementou 12 horas ao instante t até estar num intervalo de 0 a 12 horas. Efetivamente, exige menos poder computacional do que que estar a fazer em cada um dos 4 ifs a operação do módulo, que seria entre números decimais, ficando ainda mais exigente.

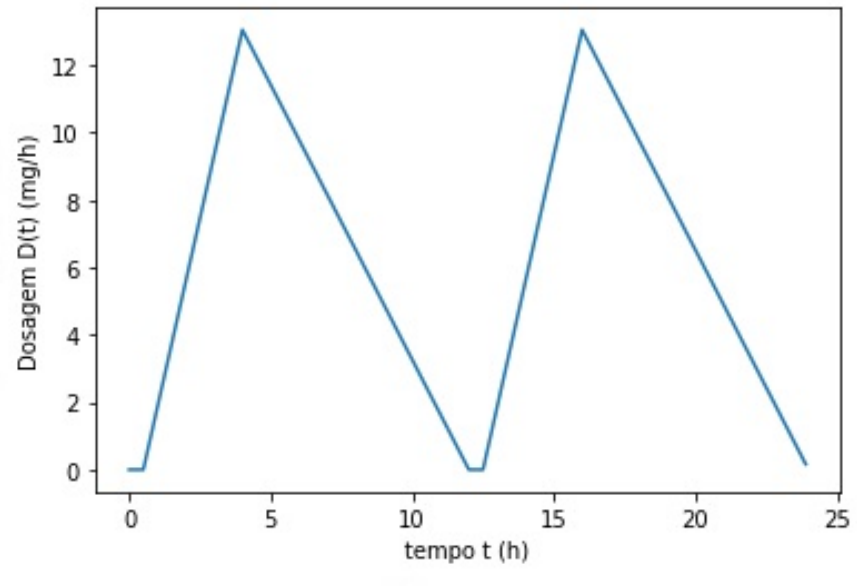

Fig 4 - Gráfico de $D(t)$ para um dia

4 Estudo das concentrações - Resolução do sistema de ODEs

4.1 Overview do problema

Tendo já a função de administração e a constante de absorção, é possível resolver o sistema de ODEs, numericamente, de modo a obter um conjunto de pontos na forma (t, m_i) e (t, m_p). Para isto recorreu-se ao método de Euler, Euler modificado, Runge-Kutta 2ª ordem e Runge-Kutta 4ª ordem com as condições iniciais $(t_0, m_{i0}, m_{p0}) = (0, 0, 0)$, com 200, 400 e 800 iterações para um tempo final de 7 dias. Para o Euler modificado, correu-se o método com o Runge-Kutta de 4ª ordem com n = 200 para obter outro ponto $(t_1, m_{i1}, m_{p1}) = (0.84, 0.17739, 0)$ que para $2n$ e $4n$ era $(0.42, 0, 0)$ e $(0.21, 0, 0)$, respetivamente.

4.2 Método de Euler

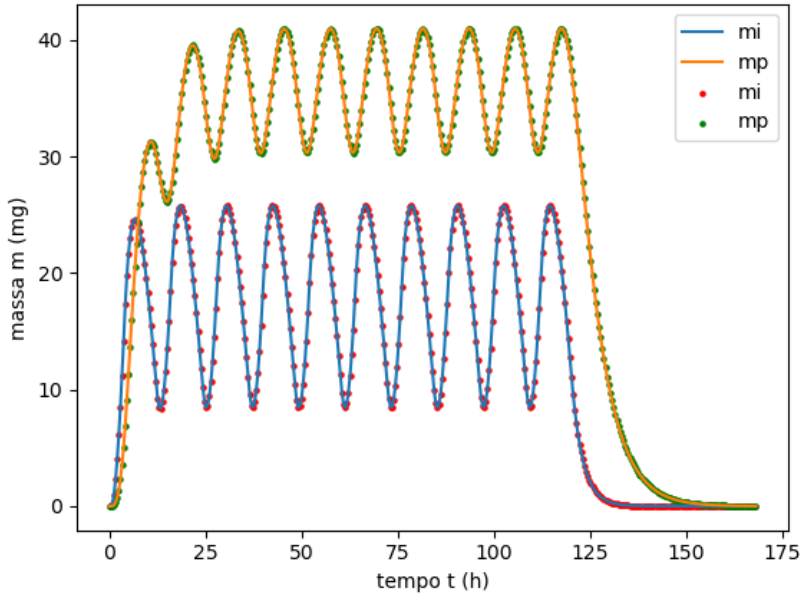

Fig 5 - Gráfico de m_i e m_p em função do tempo para 7 dias com $n = 200$

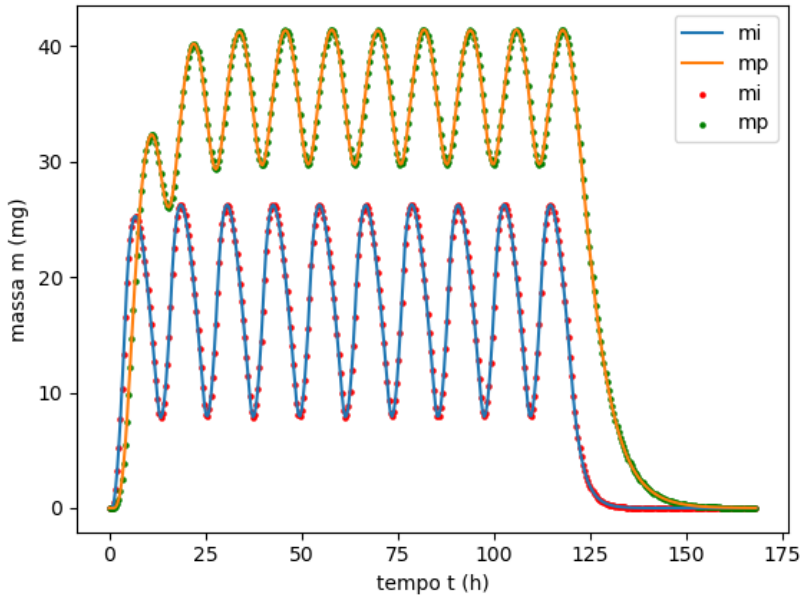

Fig 6 - Gráfico de m_i e m_p em função do tempo para 7 dias com $n = 400$

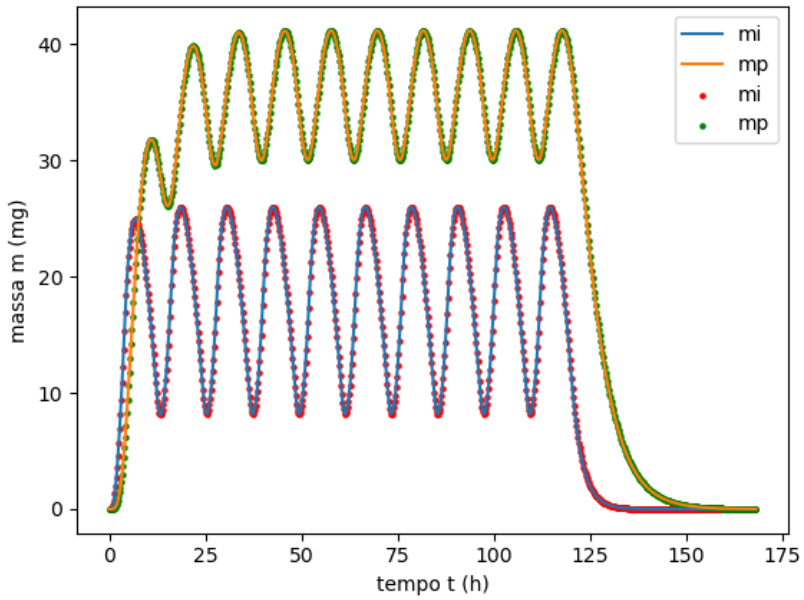

Fig 7 - Gráfico de m_i e m_p em função do tempo para 7 dias com $n = 800$

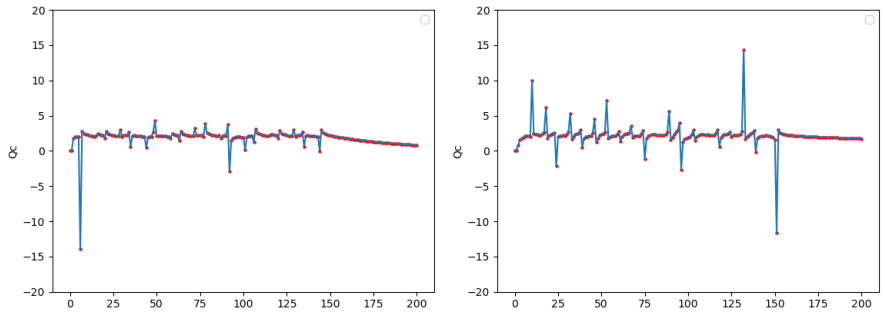

Fig 8 - Gráfico do quociente de convergência para m_i e m_p, respetivamente

4.3 Método de Euler modificado

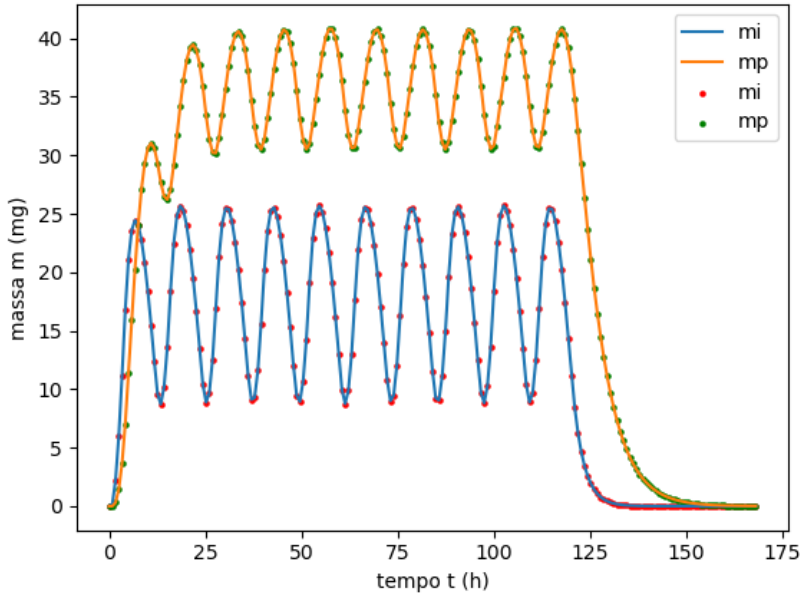

Fig 9 - Gráfico de m_i e m_p em função do tempo para 7 dias com $n = 200$

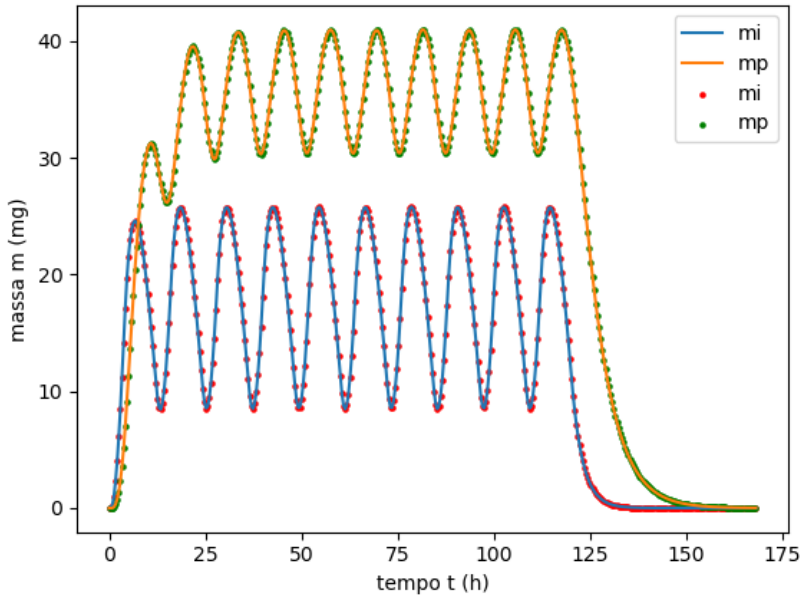

Fig 10 - Gráfico de m_i e m_p em função do tempo para 7 dias com $n = 400$

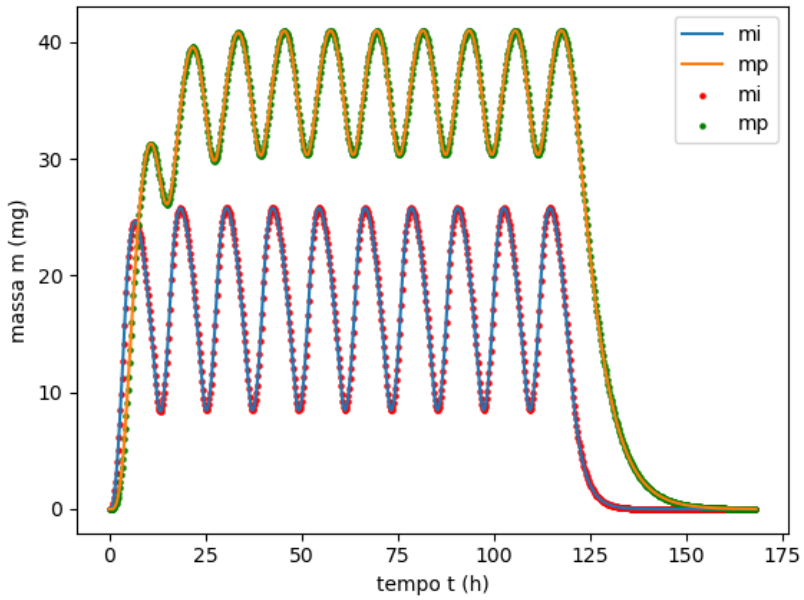

Fig 11 - Gráfico de m_i e m_p em função do tempo para 7 dias com $n = 800$

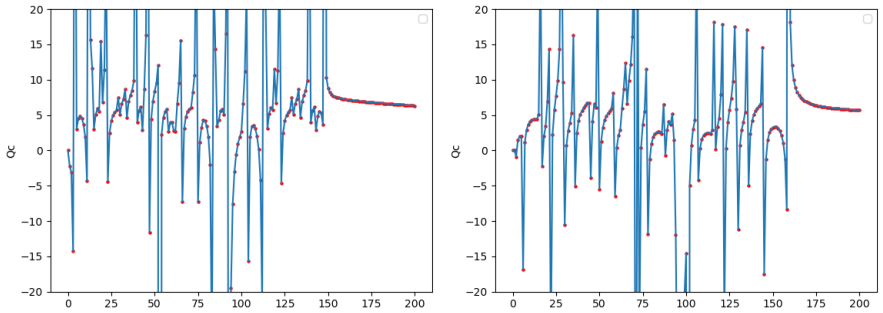

Fig 12 - Gráfico do quociente de convergência para m_i e m_p, respetivamente

4.4 Runge-Kutta 2ª ordem

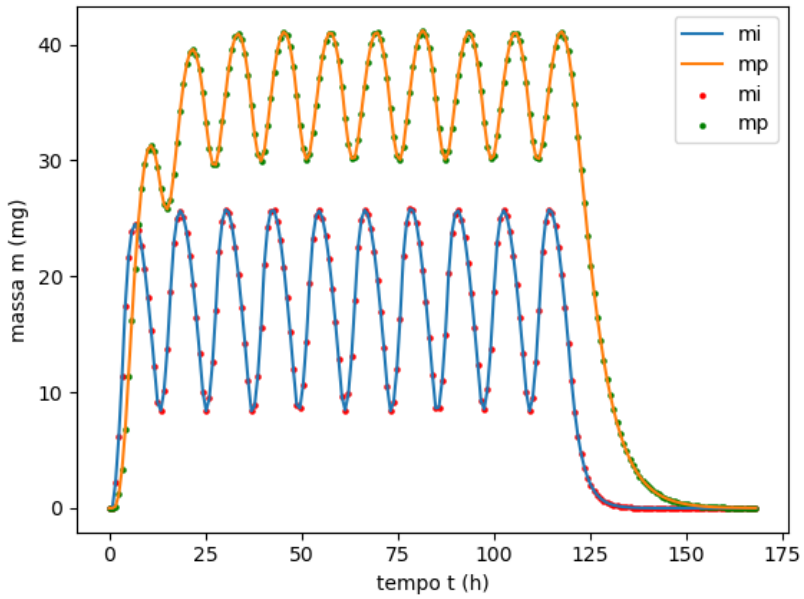

Fig 13 - Gráfico de m_i e m_p em função do tempo para 7 dias com $n = 200$

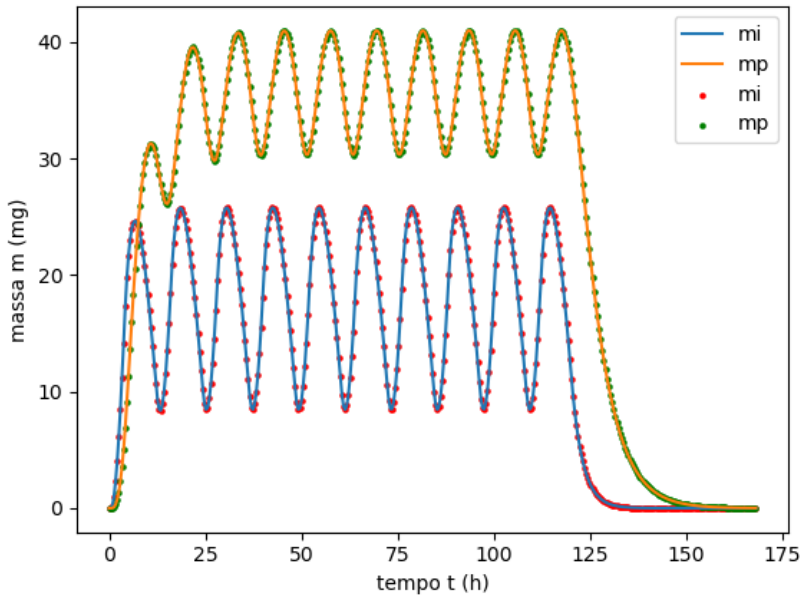

Fig 14 - Gráfico de m_i e m_p em função do tempo para 7 dias com $n = 400$

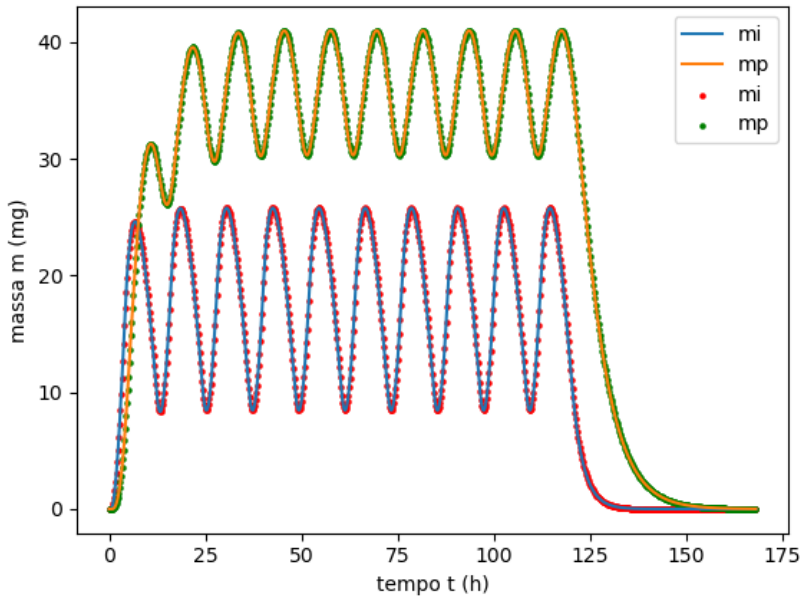

Fig 15 - Gráfico de m_i e m_p em função do tempo para 7 dias com $n = 800$

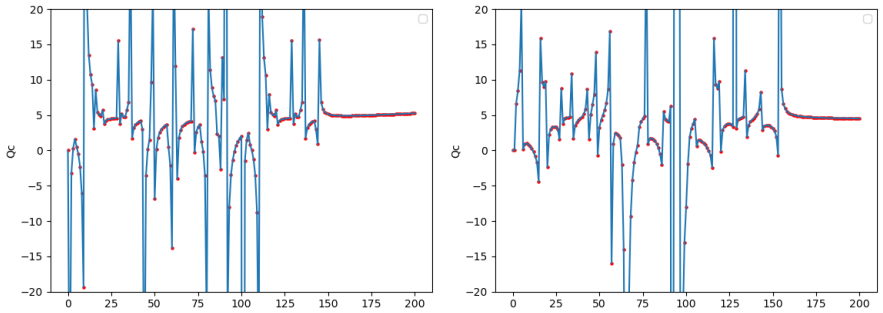

Fig 16 - Gráfico do quociente de convergência para m_i e m_p, respetivamente

4.5 Runge-Kutta 4ª ordem

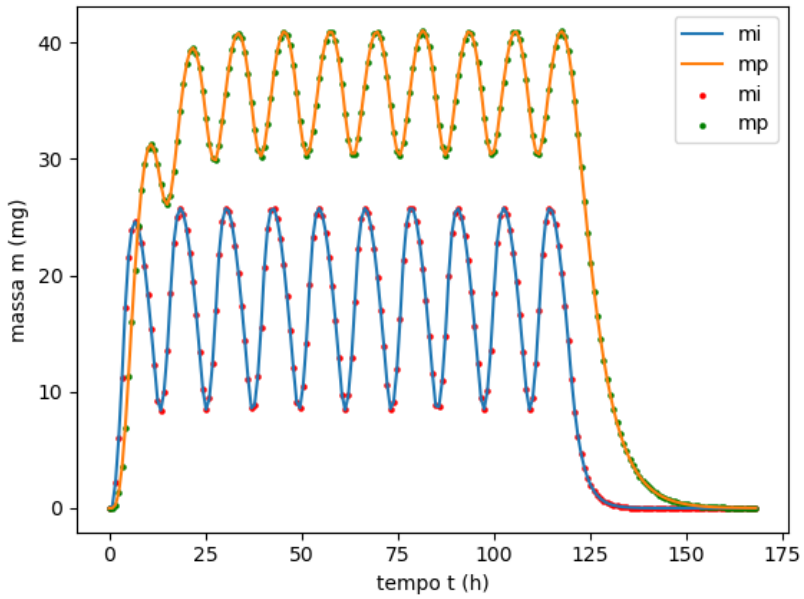

Fig 17 - Gráfico de m_i e m_p em função do tempo para 7 dias com $n = 200$

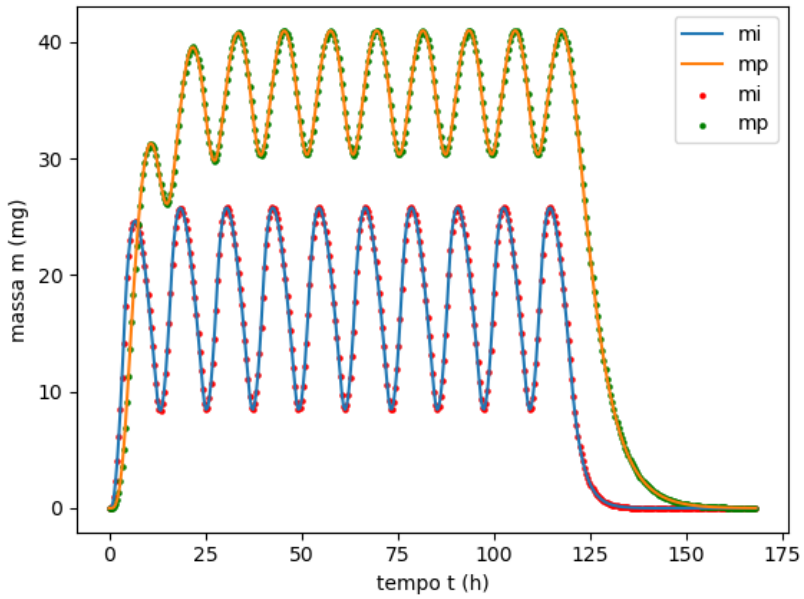

Fig 18 - Gráfico de m_i e m_p em função do tempo para 7 dias com $n = 400$

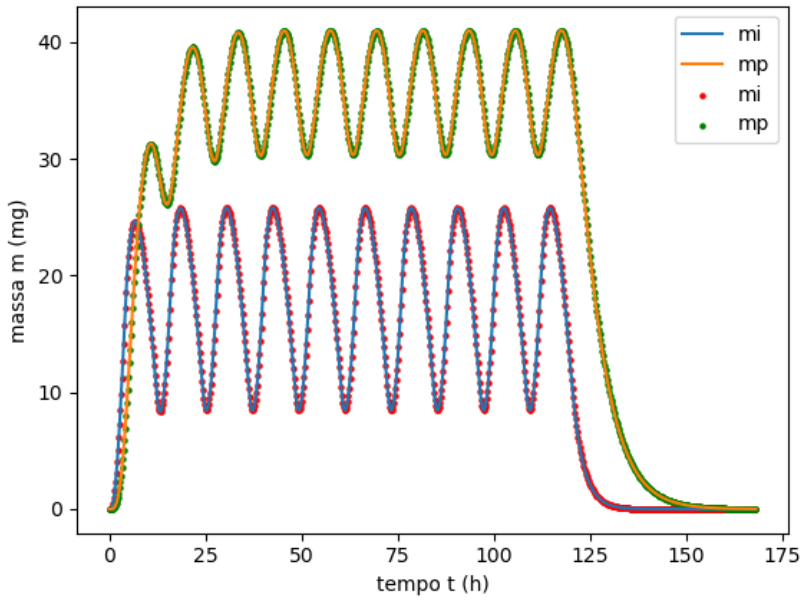

Fig 19 - Gráfico de m_i e m_p em função do tempo para 7 dias com $n = 800$

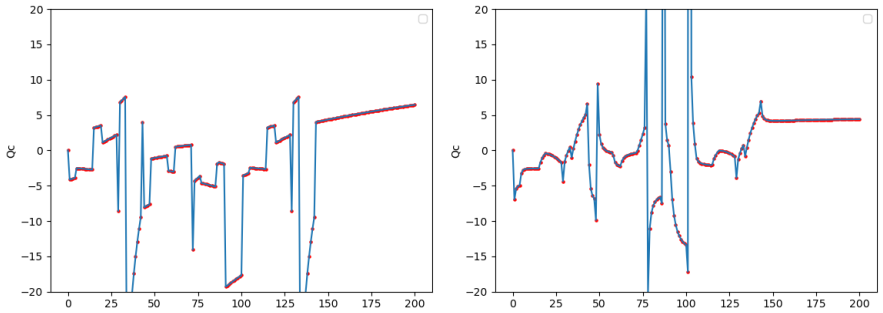

Fig 20 - Gráfico do quociente de convergência para m_i e m_p, respetivamente

4.6 Overview dos métodos

Efetivamente, a partir de n = 500 a diferença ao nível gráfico de todos os métodos é, virtualmente, não existente e para n = 200 a diferença também não é muita. Para o mesmo método, aumentar o n faz aumentar a distância entre os gráficos de m_i e m_p, sendo isto mais notável para o Euler e menos notável para o Runge-Kutta de 4ª ordem. Para o mesmo valor de n, a tal distância é também maior no Runge-Kutta de 4ª ordem.
Quanto à avaliação das estimativas dos erros ϵ'', dado que os valores de Q_c eram muito esporádicos e na sua totalidade quase nunca estavam perto dos valores necessários, não dá para concluir nada acerca da precisão e da exatidão dos métodos porque a "fórmula"do erro não é válida, isto para os métodos de Euler modificado, Runge-Kutta de 2ª ordem e Runge-Kutta de 4ª ordem. Contudo, no método de Euler, que para poder fazer a estimativa do erro teria que ter um $Q_c = 2$ já que é de primeira ordem, quase na totalidade dos seus pontos conseguiu atingir esse quociente de convergência. Aumentar o n não melhorou os resultados do Q_c nos métodos.
Dado que para amostras grandes de pontos a diferença de métodos não é notável e que é possível calcular o erro no método de Euler, este é o mais indicado.

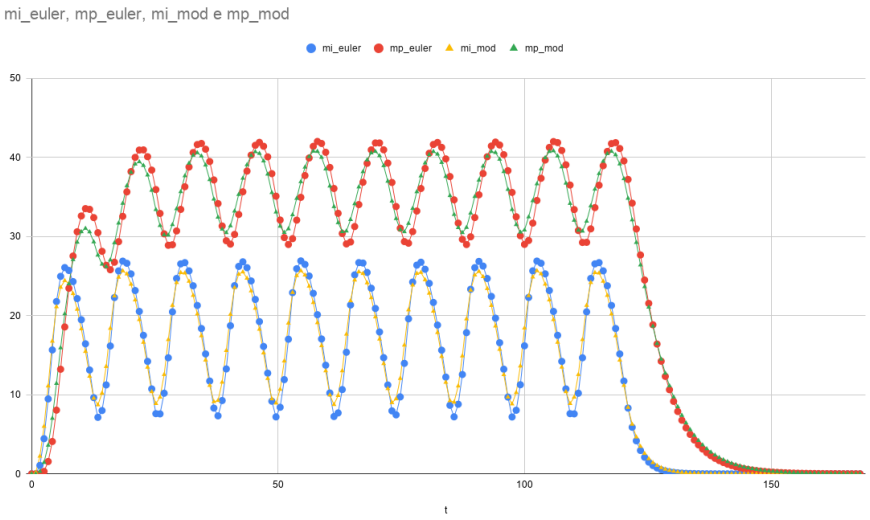

Fig 21 - Comparação do Euler e Euler modificado com n = 200

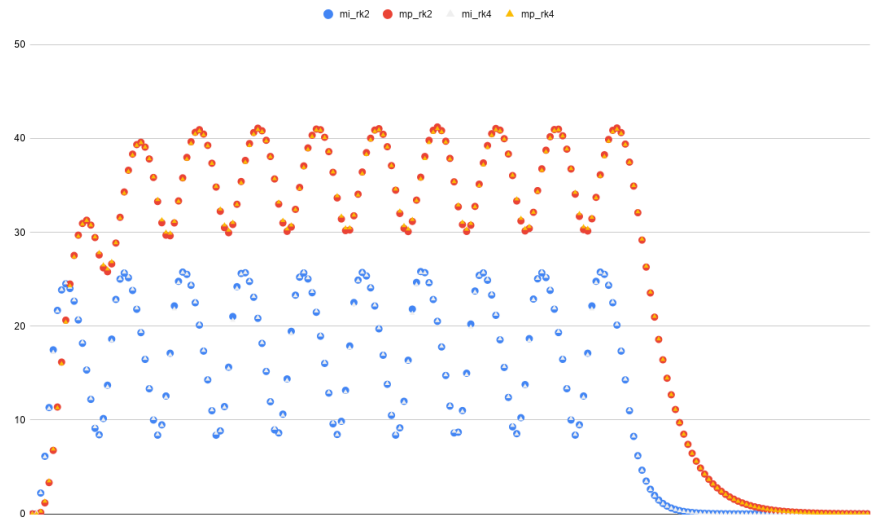

Fig 22 - Comparação do Runge-Kutta 2ª ordem e Runge-Kutta 4ª ordem com n = 200

5 Conclusão

Concluiu-se então que que a periodicidade nos triângulos reflete na periodicidade dos gráficos de m_i e m_p sendo que a solução para o sistema são duas funções que incluem uma parte logarítmica quando o medicamento começa a ser tomado, uma parte exponencial quando o tempo de tratamento acaba e uma parte oscilatória na interface das duas. Percebeu-se que os métodos tanto poderiam produzir resultados muito diferentes como também resultados muito iguais. Foram implementados todos os métodos abordados em aula em python da maneira mais eficiente possível.

6 Notas

Nos ficheiros do projeto foram incluídos todos os gráficos que o python produziu para m_i e m_p, quer sejam separados ou juntos, com o curve-fitting e os pontos, em conjunto ou não, sendo tudo isto para cada método e para cada valor de n. Envia-se também ficheiros de Excel com os pontos, plots nas mesmas condições, onde é mais fácil de comparar nos gráficos as diferenças dos métodos. As tabelas das iterações de todos os métodos usados para descubrir o k_a e gráficos das mesmas estão presentes também. Quanto ao código, envia-se na forma de ficheiros python (e em texto) que poderão ser corridos e modificados, sendo preferido ter o anaconda instalado pois usa-se os módulos de lá. O ficheiro a ser corrido é o "main.py". Está também presente o script de Máxima usado. O nosso repositório de Github do projeto está também disponível no link: "https://github.com/JFernandes2612/mnumproj2021".