João Pedro Dias Nunes 10705846

Equações Diferenciais

PME 3380 - Modelagem de Sistemas Dinâmicos - Lista 2

Brasil

2020

Sumário

1 Exemplos

1.1 Comparação entre a solução numérica e analítica com o método de Euler

1.1.1 Modelagem

Como motivação para exemplificar uma integração numérica, utilizaremos um material cujo calor específico a pressão constante c_p, que está inicialmente a uma temperatura T_0, sofre uma variação infinitesimal de temperatura, modelada por:

$$mc_p dT = Ak(T_f - T)dt = \text{calor trocado no intervalo de tempo } dt \tag{1.1}$$

$$\frac{dT}{dt} = \frac{Ak}{mc_p}(T_f - T) \tag{1.2}$$

ou, chamando

$$T = y \tag{1.3}$$

$$\frac{Ak}{mc_p} = C \tag{1.4}$$

obtemos

$$\dot{y} = C(T_F - y) \tag{1.5}$$

Ao adotarmos $C = 1/2$ e $T_F = 1$, tem-se

$$\dot{y}(t) = (1 - y(t))/2 \tag{1.6}$$

com parâmetros de tempo iniciais e finais, respectivamente dados por $t_i = 0$ e $t_f = 10$.

O problema acima também possui solução analítica (exata), dada por

$$y_e(t) = 1 - e^{-t/2} \tag{1.7}$$

1.1.2 Código

Utilizando o método de *Euler* no código abaixo, podemos simular a integração numérica, com passo 0,5, e a analítica com a equação 1.7.

```
1 clc(); clear(); close();
2
3 t(1) = 0; %instante inicial
```

```
tf = 10; %instante final

y(1) = 0; %condição inicial
ye(1) = 0; %valor inicial da solução exata

h = 0.5; %passo

n = round(tf/h); %Cálculo do número de passos

for i=1:n %método de Euler
    t(i+1)=t(i)+h; %vetor de tempo
    y(i+1)=y(i)+h*derivada(y(i)); %solução numérica
    ye(i+1) = 1-exp((-t(i+1)/2)); %solução exata
end


f1 = figure;
plot(t,y,t,ye)
title("Determinação da temperatura do corpo")
xlabel("Tempo (s)")
ylabel("Temperatura ( C )")
legend('Solução Numérica','Solução Analítica','Location','southeast')
grid on
ax = gca;
ax.XRuler.Exponent = 0;
trocar()
```

Com esse código para derivada:

```
function out = derivada(y)
out = (1-y)/2;
end
```

E esse para substituição de ponto por vírgula nos gráficos:

```
function trocar()
xl=get(gca,'XTickLabel');
new_xl=strrep(xl(:),'.',',');
set(gca,'XTickLabel',new_xl);

yl=get(gca,'YTickLabel');
new_yl=strrep(yl(:),'.',',');
set(gca,'YTickLabel',new_yl);
end
```

1.1.3 Resultados

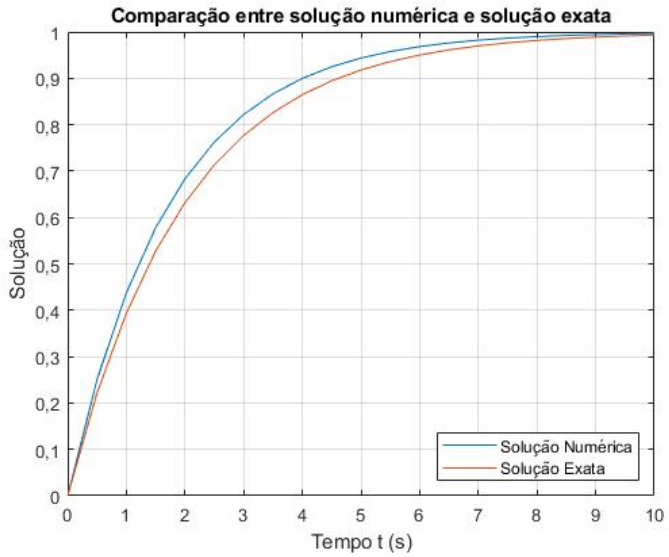

Figura 1 – Primeiro tipo de gráfico para os resultados da temperatura

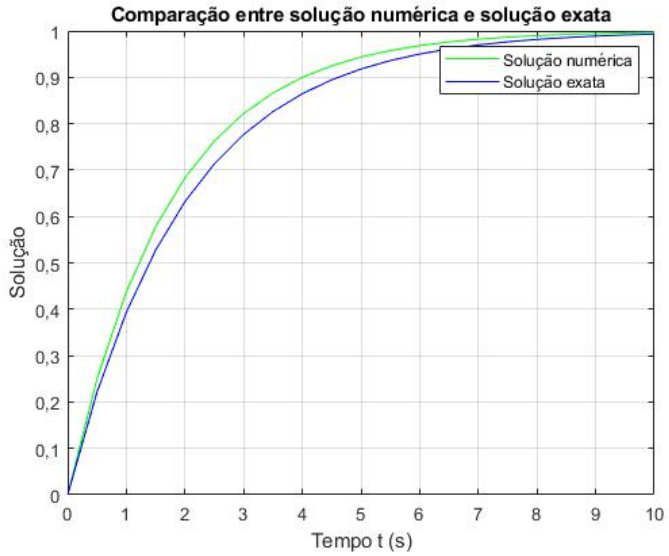

Figura 2 – Segundo tipo de gráfico para os resultados da temperatura

Observa-se que há uma disparidade entre as soluções, que é mitigada com o aumento do passo. Com um passo $h = 0.1$, obtemos:

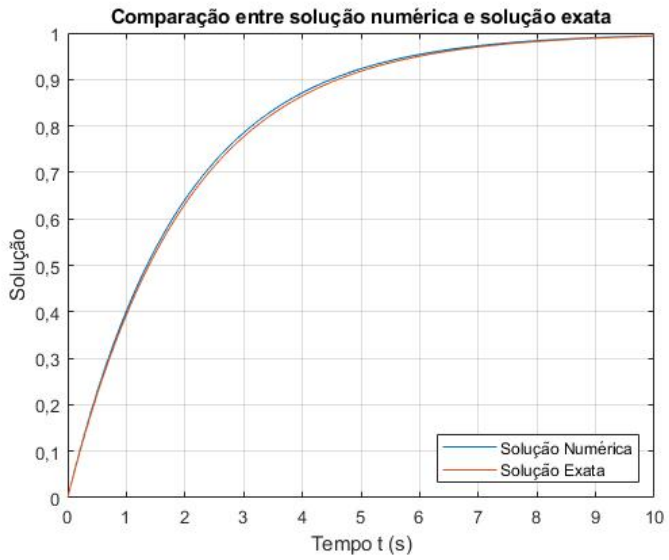

Figura 3 – Primeiro tipo de gráfico para os resultados da temperatura, com passo menor

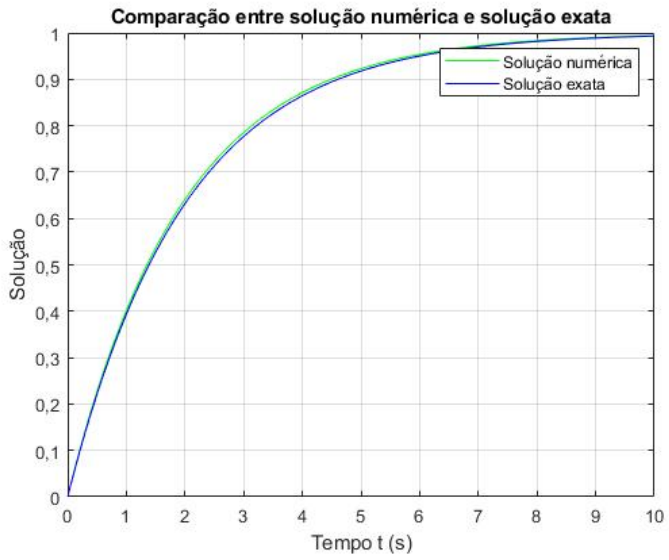

Figura 4 – Segundo tipo de gráfico para os resultados da temperatura, com passo menor

1.2 Comparação entre a solução numérica e analítica com o método de Runge-Kutta

1.2.1 Modelagem

A modelagem nesse exercício é a mesma do anterior, mas dessa vez, utilizaremos o método de *Runge-Kutta* para a integração numérica, e a analítica com a equação 1.7.

1.2.2 Código

Utilizando o método de *Runge-Kutta* no código abaixo, executamos a integração com passo 0,5.

```
clc(); clear(); close();

t(1) = 0; %instante inicial
tf = 10; %instante final

y(1) = 0; %condição inicial
ye(1) = 0; %valor inicial da solução exata

h = 0.5; %passo

n = round(tf/h); %Cálculo do número de passos

for i=1:n %método de Runge-Kutta
    t(i+1)=t(i)+h; %tempo
    k1 = h*derivada(y(i));
    k2 = h*derivada(y(i)+k1/2);
    k3 = h*derivada(y(i)+k2/2);
    k4 = h*derivada(y(i)+k3);

    y(i+1)=y(i)+(k1+2*k2+2*k3+k4)/6; %solução numérica
    ye(i+1)=1-exp(-t(i+1)/2); %solução analítica
end

f1 = figure;
plot(t,y,'g')
hold on
plot(t,ye,'b')
hold off

title("Comparação entre solução numérica e solução exata")
xlabel("Tempo t (s)")
ylabel("Solução")
legend('Solução Numérica','Solução Exata','Location','southeast')
ax = gca;
ax.XRuler.Exponent = 0;
trocar()




f2 = figure;
plot(t,y,'LineWidth',2)
```

```
hold on
plot(t,ye,'LineWidth',2)
hold off
title("Comparação entre solução numérica e solução exata")
xlabel("Tempo t (s)")
ylabel("Solução")
legend({'Solução Numérica','Solução Analítica'},'FontSize',10,'Location','
    southeast')
ax = gca;
ax.XRuler.Exponent = 0;
trocar()
```

```
f3 = figure;
plot(t,y,'LineWidth',1)
hold on
plot(t,ye,'LineWidth',1)
hold off
T=["Comparação entre solução numérica e solução exata","Tempo t (s)","Soluç
    ão","Solução numérica","Solução exata"];
title(T(1))
xlabel(T(2))
ylabel(T(3))
legend({T(4),T(5)},'FontSize',10,'Location','northwest')
ax = gca;
ax.XRuler.Exponent = 0;
trocar()
grid on
```

Com esse código para derivada:

```
function out = derivada(y)
out = (1-y)/2;
end
```

E esse para substituição de ponto por vírgula nos gráficos:

```
function trocar()
xl=get(gca,'XTickLabel');
new_xl=strrep(xl(:),'.',',');
set(gca,'XTickLabel',new_xl);

yl=get(gca,'YTickLabel');
new_yl=strrep(yl(:),'.',',');
set(gca,'YTickLabel',new_yl);
end
```

1.2.3 Resultados

A seguir, três tipos diferentes de gráficos, meramente para demonstração de recursos gráficos do *Matlab*.

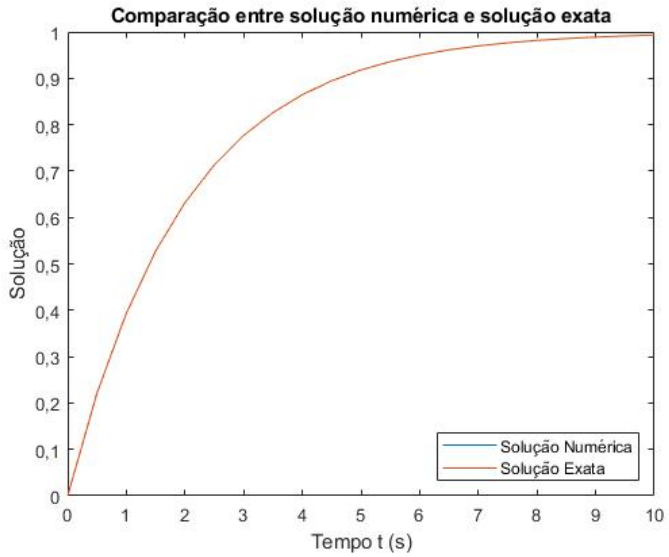

Figura 5 – Primeiro tipo de gráfico para os resultados da temperatura

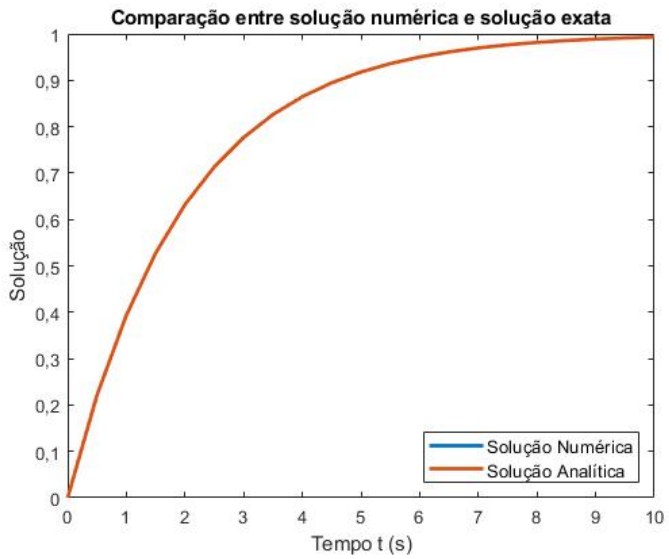

Figura 6 – Segundo tipo de gráfico para os resultados da temperatura

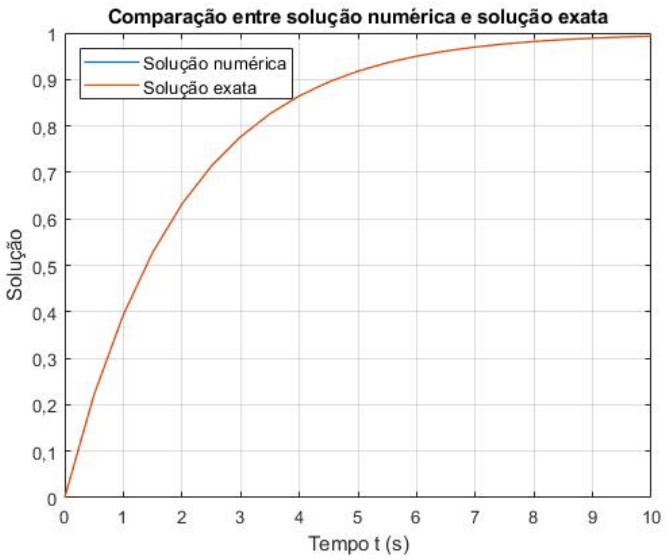

Figura 7 – Terceiro tipo de gráfico para os resultados da temperatura

Observa-se que nesse caso, mesmo com o passo $h = 0,5$, a solução numérica é bem mais próxima à analítica, pois o método de *Runge-Kutta* possui convergência maior que o método de *Euler*. Por isso, observando apenas o gráfico, há a impressão de que a solução numérica não está no gráfico, mas está embaixo da analítica.

2 Exercícios

2.1 Reservatório com uma entrada e uma saída

2.1.1 Modelagem

Esse problema consiste na determinação da altura de um reservatório que recebe e perde água, em função do tempo. A figura 8 mostra um esquema do problema.

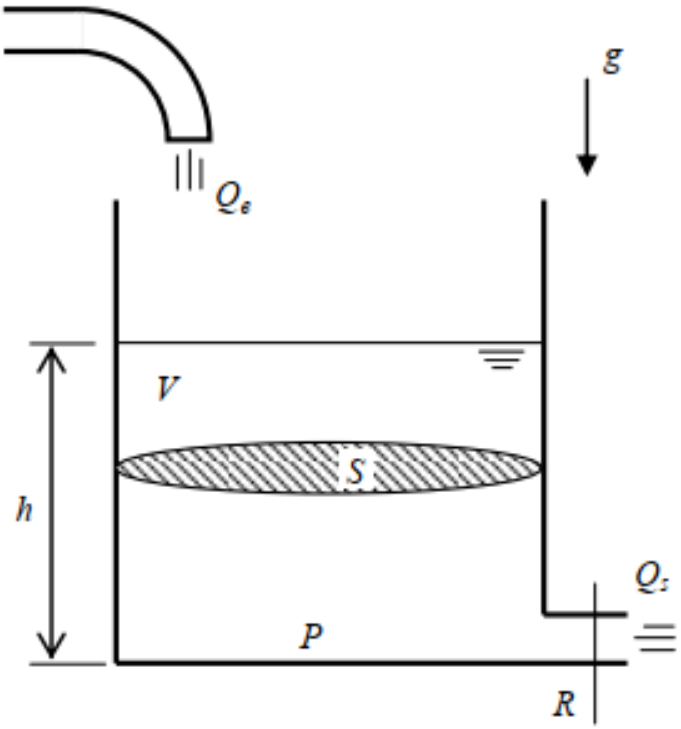

Figura 8 – Esquema do exercício 1

Os parâmetros da simulação são:

- $Q_e = 0,010247 m^3/s$ - vazão de entrada no reservatório, considerada constante;
- $S = 10 m^2$ - área da seção transversal do reservatório;
- $h_0 = 0 m$ - altura inicial do reservatório;
- $R = 2 \times 10^8 Pa/(m^3/s)^2$ - parâmetro que relaciona vazão com perda de pressão (perda de carga);
- $\rho = 1000 kg/m^3$ - massa específica da água.

e suas variáveis:

- h - altura do reservatório, em metros;
- V - volume de água no reservatório, em metros cúbicos;
- P - pressão relativa à atmosférica, no fundo do reservatório, em Pascal;
- Q_s - vazão de saída, em metros cúbicos por segundo.

Com a hipótese de que a água é um fluido incompressível, tem-se, pela equação da continuidade:

$$\frac{dV}{dt} = Q_e - Q_s \tag{2.1}$$

Supondo que a expressão abaixo modela a perda de carga na saída:

$$P = RQ_s^2 \longrightarrow Q_s = \sqrt{\frac{P}{R}} \tag{2.2}$$

com a pressão no fundo do reservatório dada por:

$$P = \rho g h \tag{2.3}$$

volume preenchido do reservatório dado por:

$$V = Sh \longrightarrow \dot{V} = S\dot{h} \tag{2.4}$$

Com as substiuições, a equação diferencial ordinária não linear é:

$$\dot{h} = \left(-\sqrt{\frac{\rho g h}{R}} + Q_e \right) \frac{1}{S} \tag{2.5}$$

2.1.2 Código

Assim, pode-se resolver esse problema no *Matlab* com o método de *Euler* e de *Runge-Kutta*, com o código abaixo.

```
clc(); clear(); close;

%Par metros
S = 10; %m2, área da seção transversal
R = 2e8; %Pa/(m3/s2) - par metro que relaciona vazão com queda de pressão
rho = 1000; %kg/m3 - massa específica da água
g = 10; %m/s2 - aceleração da gravidade na Terra
Qe = 0.010247; %m3/s - vazão de entrada
hint = 0; %altura inicial

ti = 0; %tempo inicial
tf = 30000; %tempo final
```

```
dt = 0.1; %passo

ndivs = (tf-ti)/dt;

t = linspace(ti,tf,ndivs); %vetor de tempos

he(1) = hint;

for i=1:ndivs-1
    he(i+1)= he(i)+ dt*derivada(he(i),S,R,rho,g,Qe);
end

hrk(1) = hint;

for i=1:ndivs-1
    k1 = dt*derivada(hrk(i),S,R,rho,g,Qe);
    k2 = dt*derivada(hrk(i)+k1/2,S,R,rho,g,Qe);
    k3 = dt*derivada(hrk(i)+k2/2,S,R,rho,g,Qe);
    k4 = dt*derivada(hrk(i)+k3,S,R,rho,g,Qe);

    hrk(i+1)=hrk(i)+(k1+2*k2+2*k3+k4)/6;
end

f1 = figure
plot(t,he)
title("Altura do reservatório para o Método de Euler")
xlabel("Tempo(s)")
ylabel("Altura do reservatório (m)")
grid on

f2 = figure
plot(t,hrk)
title("Altura do reservatório para o Método de Runge-Kutta")
xlabel("Tempo(s)")
ylabel("Altura do reservatório (m)")
grid on
```

e

```
function hpto = derivada(h,S,R,rho,g,Qe)
    hpto = (1/S)*(-(rho*g*h/R)^0.5+Qe);
```

2.1.3 Resultados

Com o código acima, foi possível obter os resultados para o método de *Euler* (figura 9) e *Runge-Kutta* (figura 10).

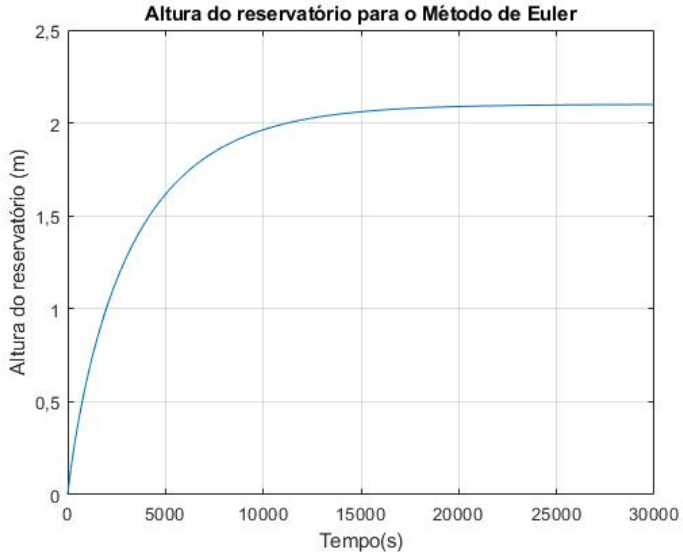

Figura 9 – Altura determinada pelo método de *Euler*

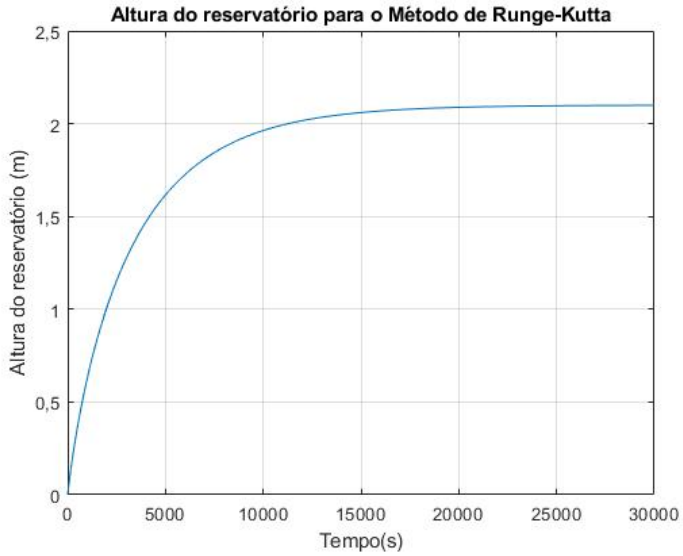

Figura 10 – Altura determinada pelo método de *Runge-Kutta*

Percebe-se que ambos métodos indicaram uma estabilização para a altura em $2.1m$.

2.2 Dois reservatórios com uma entrada e uma saída

2.2.1 Modelagem

Parecido com o problema anterior, aqui será determinada a altura em função do tempo de dois reservatórios. A figura 11 mostra um esboço desse problema.

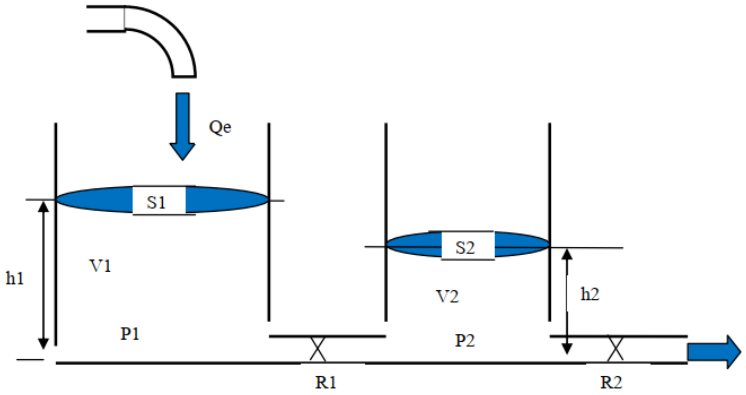

Figura 11 – Esquema do exercício 2

Sob as mesmas mesmas hipóteses adotadas no exercício de um reservatório, temos as seguintes equações diferenciais:

$$\dot{h_1} = \left[Q_e - \sqrt{\frac{\rho g}{R_a}(h_1 - h_2)} \right] \frac{1}{S_1} \tag{2.6}$$

$$\dot{h_2} = \left[\sqrt{\frac{\rho g}{R_a}(h_1 - h_2)} - \sqrt{\frac{\rho g}{R_s} h_2} \right] \frac{1}{S_2} \tag{2.7}$$

sendo os parâmetros:

- $Q_e = 0,010247 m^3/s$ - vazão de entrada no reservatório 1, considerada constante;
- $S_1 = 10 m^2$ - área da seção transversal do reservatório 1;
- $S_2 = 8 m^2$ - área da seção transversal do reservatório 2;
- $R_a = 2 \times 10^8 Pa/(m^3/s)^2$ - parâmetro que relaciona vazão com perda de pressão (perda de carga) na primeira saída;
- $R_b = 1 \times 10^8 Pa/(m^3/s)^2$ - parâmetro que relaciona vazão com perda de pressão (perda de carga) na segunda saída;
- $h_{10} = 0 m$ - altura inicial do reservatório 1;
- $h_{20} = 0 m$ - altura inicial do reservatório 2;
- $\rho = 1000 kg/m^3$ - massa específica da água.

e as variáveis:

- $\dot{h_1}$ - altura do reservatório 1, em metros;
- $\dot{h_2}$ - altura do reservatório 2, em metros.

2.2.2 Código

Assim, pode-se resolver esse problema no *Matlab* com o método de *Euler* e o de *Runge-Kutta*, com o código abaixo.

```
%%% Deixa os eixos em LaTeX
set(groot, 'defaultLegendInterpreter','latex');
clc();clear();close();

hint1 = 0; %altura inicial do reservatório 1
hint2 = 0; %altura inicial do reservatório 2

ti = 0; %tempo inicial
tf = 30000; %tempo final
dt = 0.5; %passo

ndivs = (tf-ti)/dt;

t = linspace(ti,tf,ndivs); %vetor de tempos
he=zeros(ndivs,2);

he(1,1:2) = [hint1 hint2];

for i=1:ndivs-1
   he(i+1,:) = he(i,:) + dt*derivadas(he(i,1),he(i,2));
end


f1 = figure;
plot(t,he)
title("Alturas dos reservatórios para o Método de Euler")
xlabel("Tempo (s)")
ylabel("Altura (m)")
legend('Reservatório 1','Reservatório 2','Location','southeast')
grid on
trocar()


hrk(1,1) = hint1;
hrk(1,2) = hint2;
k(1,1)=0;
k(1,2)=0;

for i=1:ndivs-1
    k(1,:) = dt*derivadas(he(i,1),he(i,2));
    k(2,:) = dt*derivadas(he(i,1)+k(1,1)/2,he(i,2)+k(1,2)/2);
    k(3,:) = dt*derivadas(he(i,1)+k(2,1)/2,he(i,2)+k(2,2)/2);
    k(4,:) = dt*derivadas(he(i,1)+k(3,1),he(i,2)+k(3,2));
```

```
44
45      hrk(i+1,:)=hrk(i,:)+(k(1,:)+2*k(2,:)+2*k(3,:)+k(4,:))/6;
46  end
47
48
49  f2 = figure;
50  plot(t,hrk)
51  title("Alturas dos reservatórios para o Método de Runge-Kutta")
52  xlabel("Tempo (s)")
53  ylabel("Altura (m)")
54  legend('Reservatório 1','Reservatório 2','Location','southeast')
55  grid on
56
57
58  trocar()
```

e

```
1  function out = derivadas(h1,h2)
2  Qe = 0.010247; %m3/s - vazão volumétrica
3  rho = 1000; %kg/m3 - massa específica da água
4  Ra = 2e8; %par metro da perda de carga 1
5  Rs = 0.5*Ra; %par metro da perda de carga 2
6  g = 10; %gravidade na terra
7  S1 = 10; %área do reservatório 1
8  S2 = 0.8*S1; %área do reservatório 2
9
10 hpt1 = (1/S1)*(Qe-(rho*g*(h1-h2)/Ra)^0.5);
11 hpt2 = (1/S2)*((rho*g*(h1-h2)/Ra)^0.5-(rho*g*h2/Rs)^0.5);
12 out = [hpt1 hpt2];
```

2.2.3 Resultados

Com o código acima, foi possível obter os resultados para o método de *Euler* (figura 12) e o método de *Runge-Kutta* (figura 13). Os métodos mostraram grande similaridade no resultado.

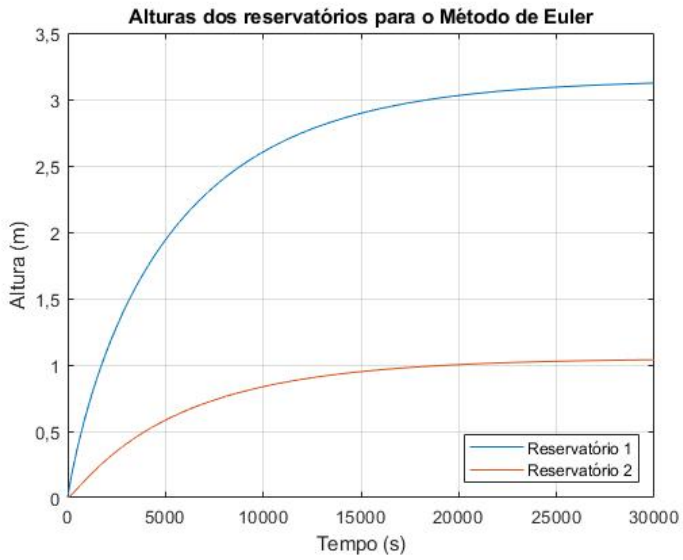

Figura 12 – Alturas determinadas pelo método de *Euler*

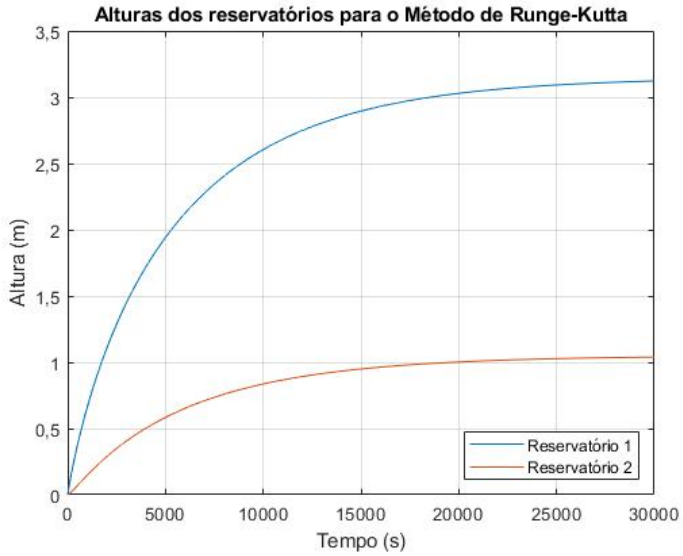

Figura 13 – Alturas determinadas pelo método de *Runge-Kutta*