ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

DEPARTAMENTO DE ENGENHARIA MECÂNICA

PME3380 – Modelagem de Sistemas Dinâmicos

Lista B

Nome	**NºUSP**
João Pedro Junqueira Seara de Morais	10774437

Professores Agenor T. Fleury e Décio C. Donha

São Paulo

Setembro, 2020

Sumário

1. Lista B

1.1 Exemplo 1 – Método de Euler

Implementação da função 'funcao.sci':

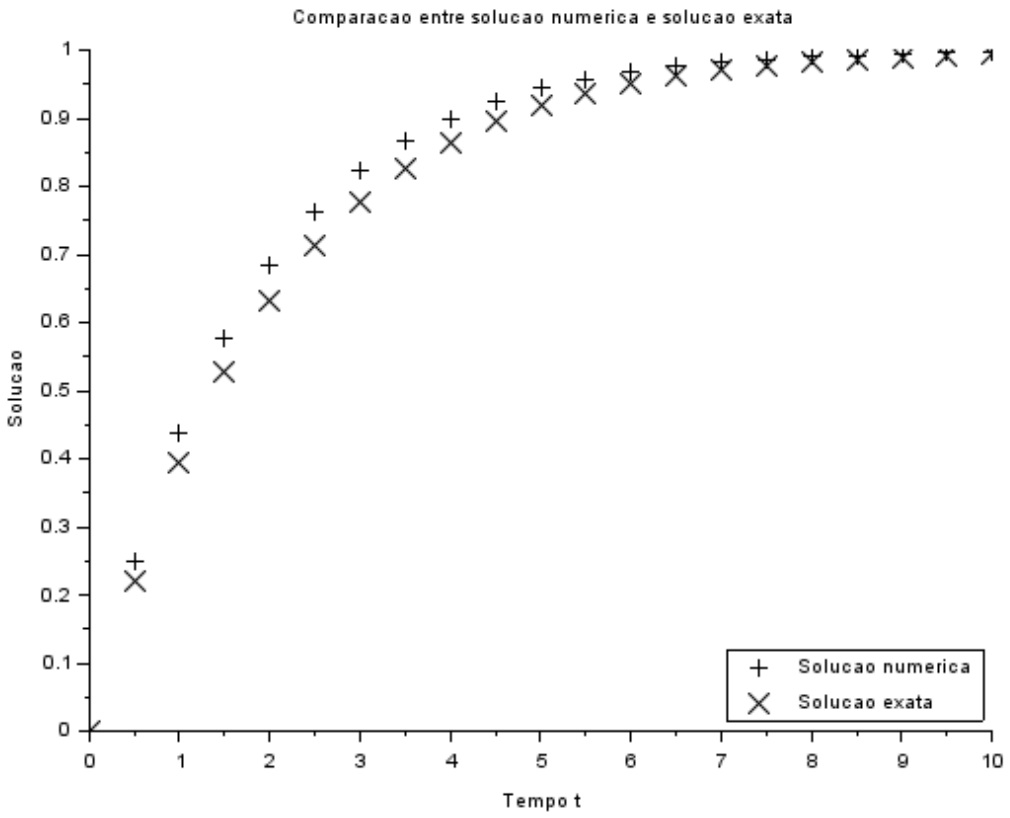
```
function [ydot]=funcao(y)
ydot=(1-y)/2;
endfunction
save ('funcao.sci','funcao');
```

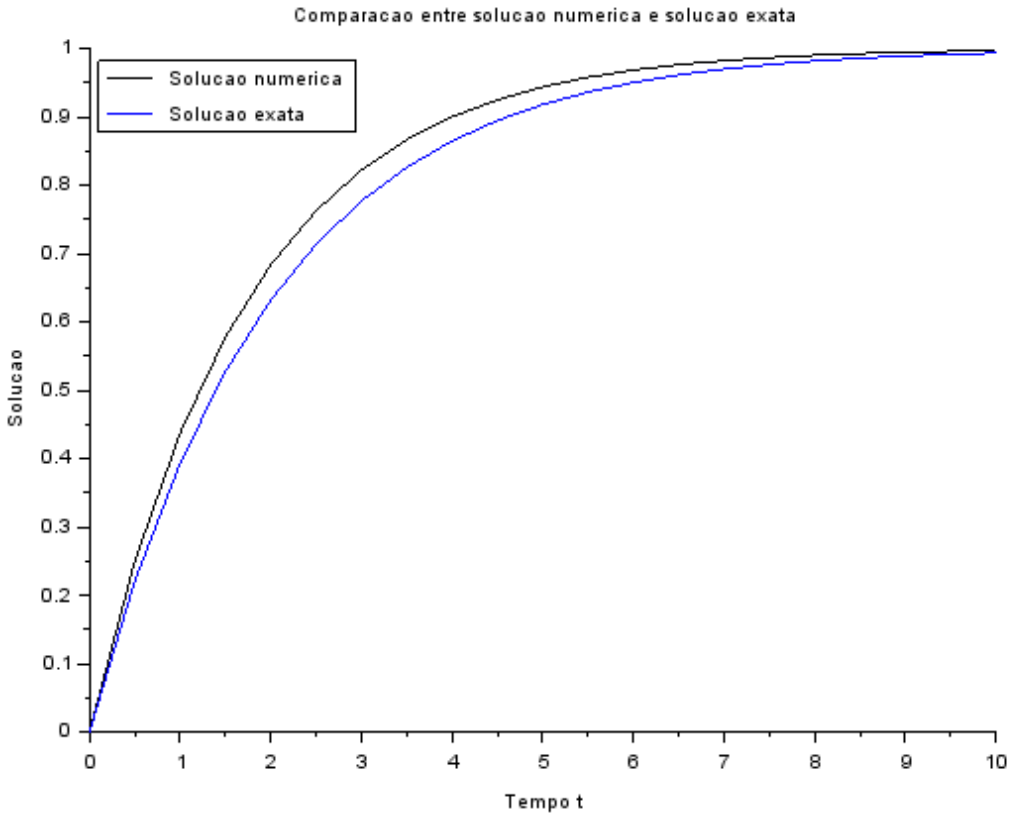
Implementação do arquivo 'numericoE.sce':

```
// Conjunto de comandos para solucao numerica de equacao diferencial dada pela
funcao funcao.sci
// Apagando dados anteriores:
clear
// Carregando a equacao diferencial:
// Carregue a função usando o comando Load do Scilab
load('funcao.sci','funcao');
// Instante inicial:
t(1)=0;
// Instante final:
tf=10;
// Condicao inicial:
y(1)=0;
// Valor inicial da solucao exata:
ye(1)=0;
// Passo de integracao (experimente alterar o passo):
h=0.5;
// Calculo de numero de passos):
n=round(tf/h);
// Integracao numerica usando o metodo de Euler:
   // Comando for:
   for i=1:n
   // Vetor de tempo:
   t(i+1)=t(i)+h;
   // Solucao numerica:
   y(i+1)=y(i)+h* funcao(y(i));
   // Solucao exata:
   ye(i+1)=1-%e^(-t(i+1)/2);
   // Termino do comando for:
   end
// Plotando solucao numerica y versus vetor de tempo t e solucao exata ye versus vetor
de tempo t:
plot2d([t,t],[y,ye],[-1 -2]);
```

```
// Colocando uma legenda na parte inferior direito da figura (parametro 4):
legends(["Solucao numerica","Solucao exata"],[-1,-2],4)
// Colocando um titulo na figura e nomeando os eixos:
xtitle("Comparacao entre solucao numerica e solucao exata","Tempo t","Solucao")
// Abrindo uma nova janela de graficos:
set("current_figure",1);
// Desenhando outro grafico com linhas diferentes:
plot2d([t,t],[y,ye],[1 2]);
// Usando a variavel do tipo 'lista':
T=list("Comparacao entre solucao numerica e solucao exata","Tempo
t","Solucao","Solucao numerica","Solucao exata");
// Colocando uma legenda na parte superior esquerda da figura (parametro 2):
legends([T(4),T(5)],[1,2],2);
// Colocando um titulo na figura e nomeando os eixos:
xtitle(T(1),T(2),T(3));
```

Resultados:



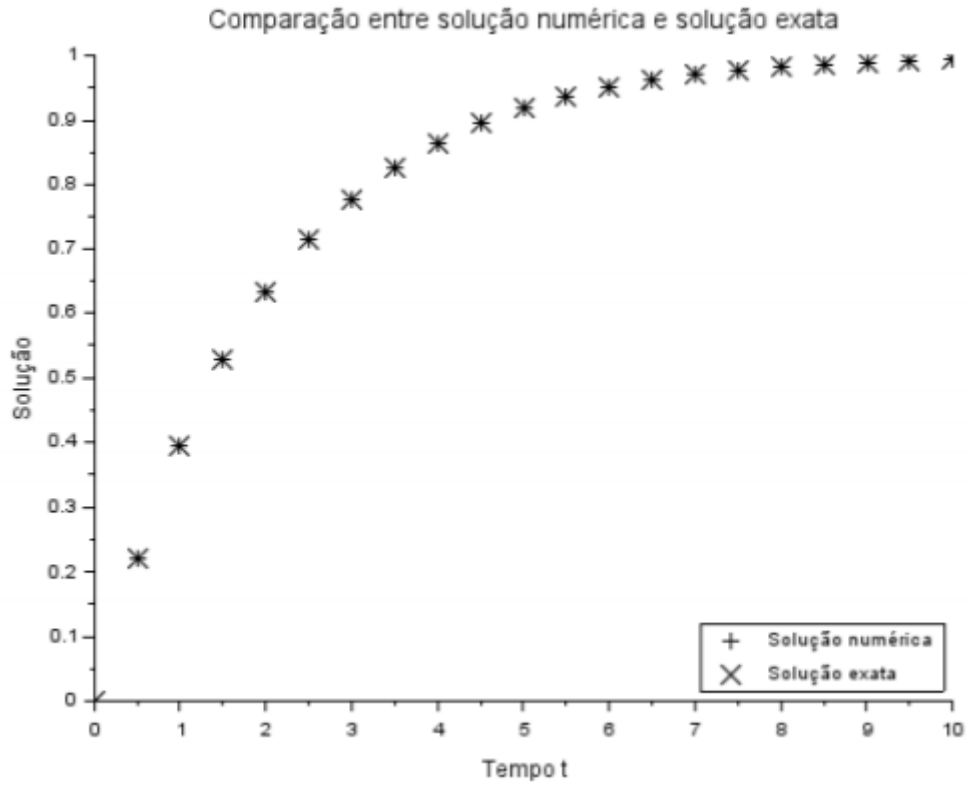

Comparacao entre solucao numerica e solucao exata
Solucao numerica
Solucao exata
Solucao
Tempo t
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
1
2
3
4
5
6
7
8
9
10

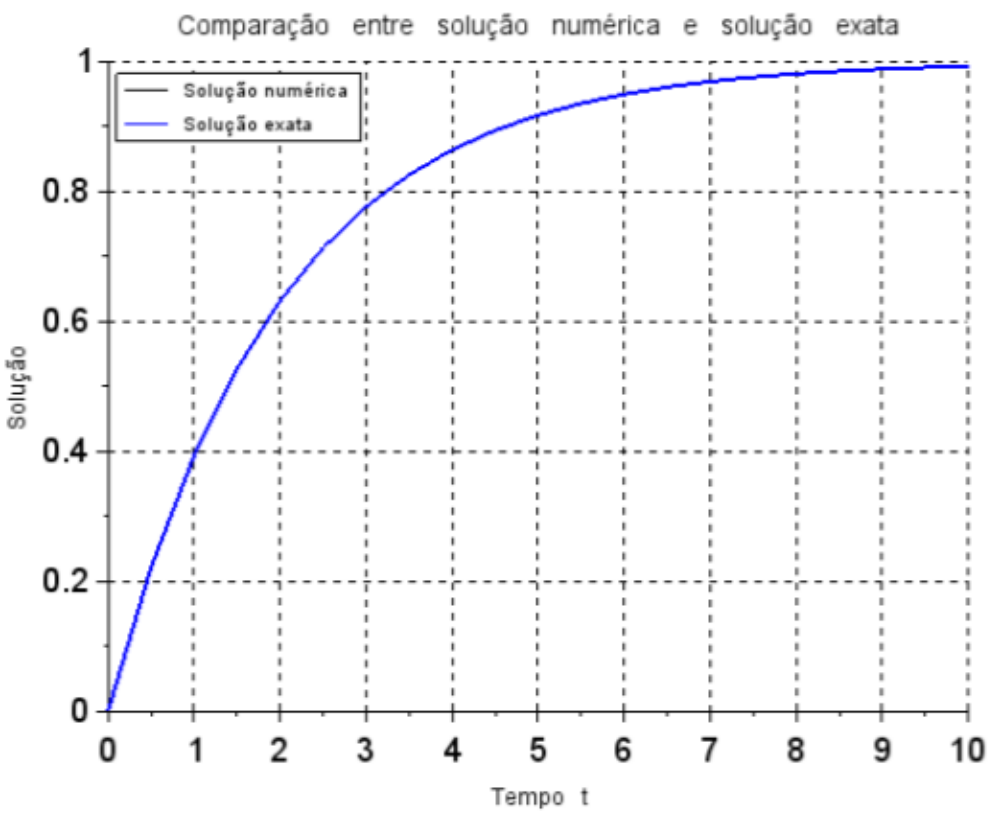
1.2 Exemplo 2 – Método de Runge-Kutta

```
// Conjunto de comandos para solucao numerica de equacao diferencial [1-y(i)]/2
// Apagando dados anteriores:
clear
// Instante inicial:
t(1) = 0;
// Instante final:
tf = 10;
// Condicao inicial:
y(1) = 0;
// Valor inicial da solucao exata:
ye(1) = 0;
// Passo de integracao (experimente alterar o passo):
h = 0.5;
// Calculo de numero de passos):
n = round((tf-t(1))/h);
// Integracao numerica usando o metodo de Runge Kutta:
// Comando for:
for i = 1:n
   //Vetor de tempo:
   t(i+1) = t(i)+h;
   // Solucao numerica:
   k1 = h * (1-(y(i)))/2;
   k2 = h * (1-(y(i)+k1/2))/2;
   k3 = h * (1-(y(i)+k2/2))/2;
   k4 = h * (1-(y(i)+k3))/2;
   y(i+1) = y(i) + ((k1+2*k2+2*k3+k4)/6);
   // Solucao exata:
   ye(i+1) = 1 - %e^(-t(i+1)/2);
   // Termino do comando for:
end
// Plotando solucao numerica y versus vetor de tempo t e solucao exata ye versus vetor
de tempo t:
plot2d([t,t],[y,ye],[-1 -2]);
// Colocando uma legenda na parte inferior direito da figura (parametro 4):
legends(["Solução numérica","Solução exata"],[-1,-2],4)
// Colocando um titulo na figura e nomeando os eixos:
xtitle("Comparação entre solução numérica e solução exata","Tempo t","Solução")
// Abrindo uma nova janela de graficos:
set("current_figure",1);
// Aumentando a espessura das linhas:
xset('thickness',2)
// Aumentando o tamanho da fonte:
xset('font size',4)
// Desenhando outro grafico com linhas diferentes:
plot2d([t,t],[y,ye],[1 2]);
```

```
// Usando a variavel do tipo 'lista':
T = list("Comparação entre solução numérica e solução exata","Tempo
t","Solução","Solução numérica","Solução exata");
// Diminuindo a espessura das linhas:
xset('thickness',1)
// Colocando uma legenda na parte superior esquerda da figura (parametro 2):
legends([T(4),T(5)],[1,2],2);
// Colocando um titulo na figura e nomeando oseixos:
xtitle(T(1),T(2),T(3));
// Colocando uma grade no grafico:
xgrid(1)
```

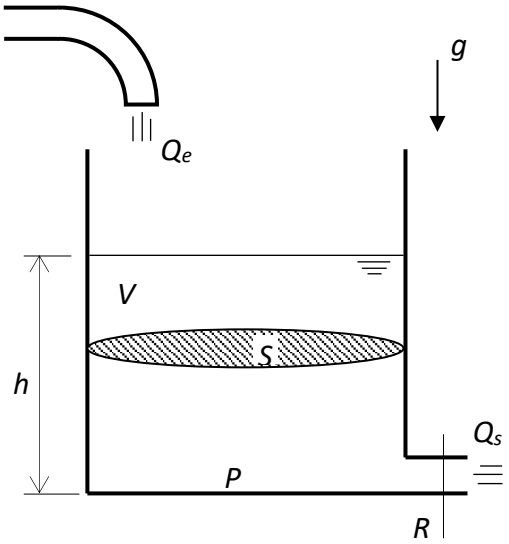
Resultados:




Comparação entre solução numérica e solução exata
Solução numérica
Solução exata
Solução
Tempo t
0
0.2
0.4
0.6
0.8
1
0
1
2
3
4
5
6
7
8
9
10

1.3 Exercício 1 – Reservatório Simples

Implementando um programa no Scilab que resolva numericamente a equação diferencial que modela o sistema abaixo, tanto pelo método de Euler como Runge Kutta.



Temos os parâmetros e variáveis do sistema:

Parâmetros:

S = 10 m^2 - área da seção transversal (constante)

R = 2×10^8 $Pa/(m^3/s)^2$ - parâmetro que relaciona vazão com queda de pressão (perda de carga)

ρ = 1000 kg/m^3 - massa específica da água

G = 10 m/s^2 - aceleração da gravidade na superfície da terra

Variáveis:

Q_e = 0,010247 m^3/s - vazão de entrada

h: nível do reservatório [m]

V: volume de água no reservatório [m^3]

P: pressão relativa à atmosférica, no fundo do reservatório [Pa]

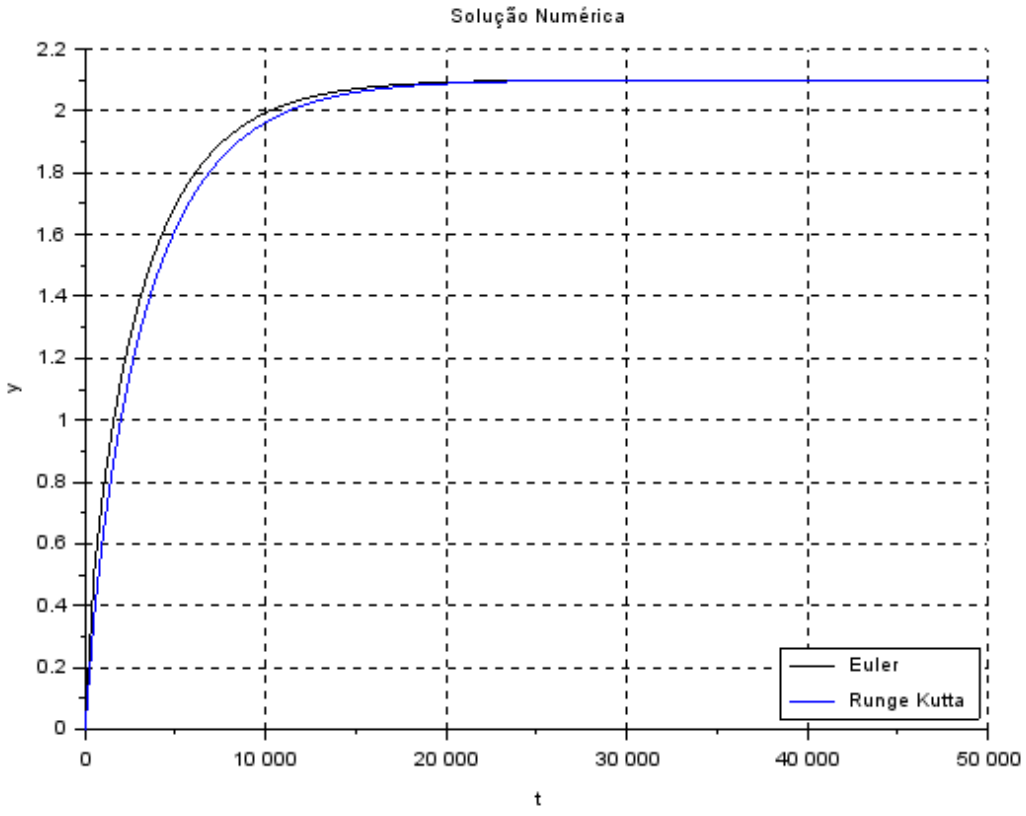
Q_s: vazão de saída [m^3/s]

Implementando o código, temos que:

```
clear();

//Definição dos parâmetros:
S = 10;
R = 2*10^8;
p = 1000;
g = 10;
Qe = 0.010247;

function dy=reservatorio(h)
   dy = (-sqrt(p*g*h/R)+Qe)*(1/S)
endfunction

t(1) = 0;
tf = 50000;

// Instante inicial do reservatório:
y(1) = 0;
y_e(1) = y(1);
y_rk(1) = y(1);

// Passo de integração:
h = 500;

// Número de passos:
n = round((tf-t(1))/h);

// Método de Euler:
for i=1:n
   // Vetor de tempo:
   t(i+1)=t(i)+ h;
   // Solucao numérica:
   y_e(i+1)=y_e(i) + h*reservatorio(y_e(i));
end

// Método de Runge Kutta:
for i=1:n
   // Vetor de tempo:
   t(i+1)=t(i)+ h;
   // Solucao Numérica:
   k1 = reservatorio(y_rk(i))
   k2 = reservatorio(y_rk(i) + (h/2)*k1)
   k3 = reservatorio(y_rk(i) + (h/2)*k2)
   k4 = reservatorio(y_rk(i) + h*k3)
   y_rk(i+1)=y_rk(i) + h*((k1+2*k2+2*k3+k4)/6);
```

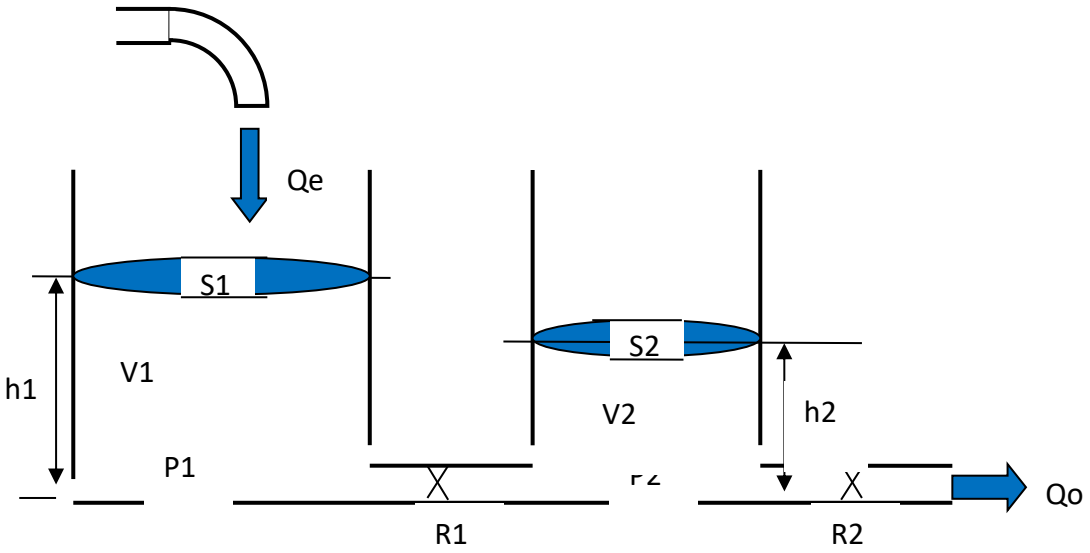
```
end

scf
plot2d([t,t],[y_e,y_rk],[1 2]);
legends(["Euler","Runge Kutta"],[1,2],4)
xset('thickness',1);
xtitle("Solução Numérica","t","y");
xgrid(1)
```

Resultado:



1.4 Exercício 2 – Sistema com Reservatórios Duplos

Desenvolveremos um programa em Scilab que resolva numericamente o sistema de equações diferencias que modela o sistema com dois reservatórios, usando tanto Euler como Runge Kutta.



Modelo do sistema de 2 reservatórios:

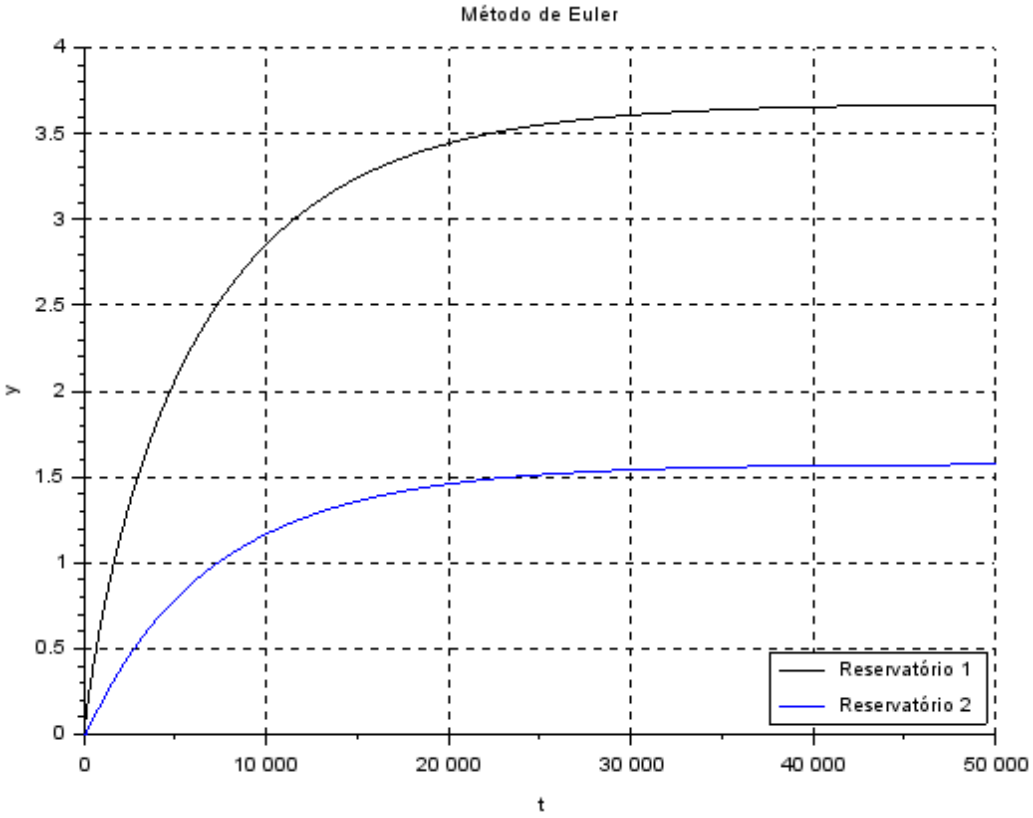
$$\begin{cases} \dot{h}_1 = \left[Q_e - \sqrt{\frac{\rho g}{R_a}(h_1 - h_2)} \right] \frac{1}{S_1} \\ \dot{h}_2 = \left[\sqrt{\frac{\rho g}{R_a}(h_1 - h_2)} - \sqrt{\frac{\rho g}{R_s} h_2} \right] \frac{1}{S_2} \end{cases}$$

Implementando o código, temos que primeiramente, para o método de Euler:

```
clear
// Definição dos parâmetros:

// Seção transversal do reservatório 1
S1 = 10;
// Seção transversal do reservatório 2
S2 = 7;
// Razão com perda de carga
Ra = 2*(10^8);
Rs = 1.5*(10^8);
// Massa específica da água
```

```
p = 1000;
// Gravidade
g = 10;
// Vazão de entrada
Qe = 0.010247;

function [H1h]=reservatorio1(h1, h2, S1, Ra, ro, G, Qe)
   H1h = (1/S1) * (- sqrt(ro*G*(h1-h2)/Ra) + Qe);
endfunction
function [H2h]=reservatorio2(h1, h2, S2, Ra, Rs, ro, G, Qe)
   H2h = (1/S2) * (sqrt(ro*G*(h1-h2)/Ra) - sqrt(ro*G*h2/Rs))
endfunction

t(1) = 0;
tf = 50000;

// Instante inicial dos reservatórios
H1(1) = 0;
H2(1) = 0;

// Passo de integração
h = 100;

// Número de passos:
n = round((tf-t(1))/h);

// Método de Euler:
for i = 1:n
   // Vetor de tempo
   t(i+1) = t(i) + h;
   H1(i+1) = H1(i) + h * reservatorio1(H1(i),H2(i),S1,Ra,p,g,Qe);
   H2(i+1) = H2(i) + h * reservatorio2(H1(i),H2(i),S2,Ra,Rs,p,g,Qe);
end

// Plotagem dos gráficos:
scf
plot2d([t,t],[H1,H2],[1 2]);
legends(["Reservatório 1","Reservatório 2"],[1,2],4)
xset('thickness',1);
xtitle("Método de Euler ","t","y");
xgrid(1)
```

Resultado:

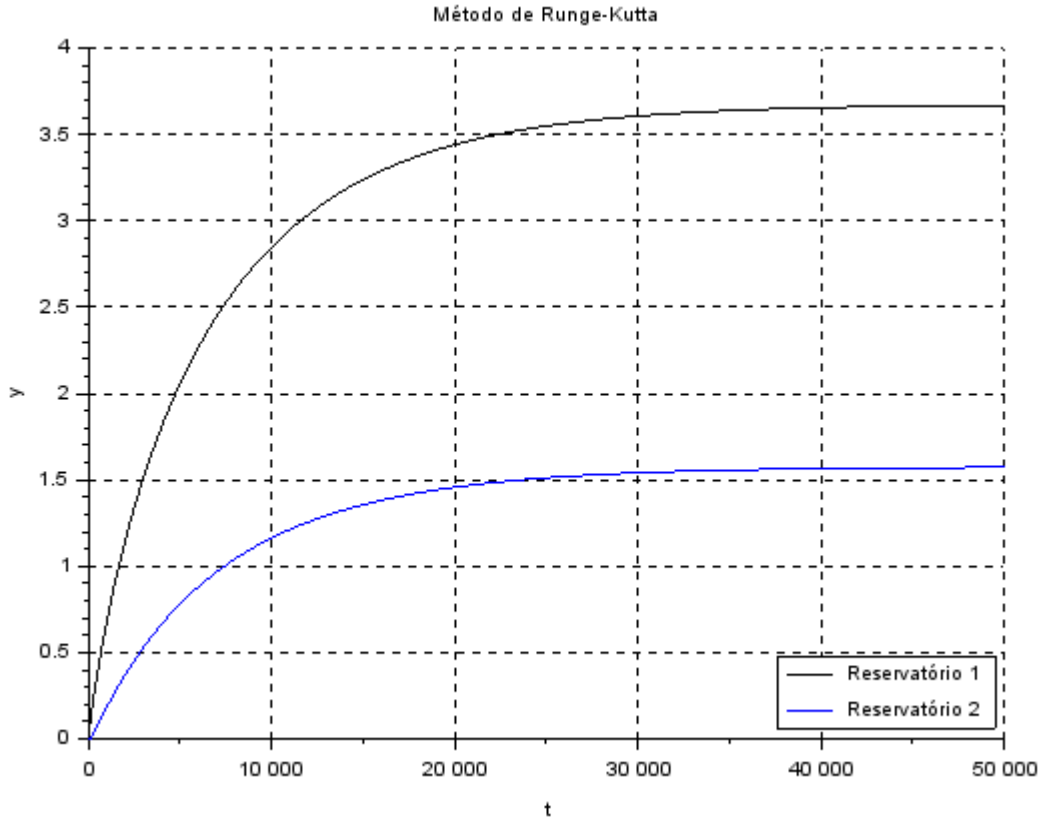


Por fim, implementaremos o código pelo método de Runge-Kutta:

```
clear
// Definição dos parâmetros:

// Seção transversal do reservatório 1
S1 = 10;
// Seção transversal do reservatório 2
S2 = 7;
// Razão com perda de carga
Ra = 2*(10^8);
Rs = 1.5*(10^8);
// Massa específica da água
p = 1000;
// Gravidade
g = 10;
// Vazão de entrada
Qe = 0.010247;

function [H1h]=reservatorio1(h1, h2, S1, Ra, ro, G, Qe)
   H1h = (1/S1) * (- sqrt(ro*G*(h1-h2)/Ra) + Qe);
endfunction
function [H2h]=reservatorio2(h1, h2, S2, Ra, Rs, ro, G, Qe)
```

```
   H2h = (1/S2) * (sqrt(ro*G*(h1-h2)/Ra) - sqrt(ro*G*h2/Rs))
endfunction

t(1) = 0;
tf = 50000;

// Instante inicial dos reservatórios
H1(1) = 0;
H2(1) = 0;

// Passo de integração
h = 100;

// Número de passos:
n = round((tf-t(1))/h);

// Método de Runge-Kutta:
for i = 1:n
   // Vetor de tempo
   t(i+1) = t(i) + h;
   k1 = reservatorio1(H1(i),H2(i),S1,Ra,p,g,Qe);
   k2 = reservatorio1(H1(i)+h*0.5*k1,H2(i)+h*0.5*k1,S1,Ra,p,g,Qe);
   k3 = reservatorio1(H1(i)+h*0.5*k2,H2(i)+h*0.5*k2,S1,Ra,p,g,Qe);
   k4 = reservatorio1(H1(i)+h*k3,H2(i)+h*k3,S1,Ra,p,g,Qe);
   H1(i+1) = H1(i)+((k1+2*k2+2*k3+k4)*(h/6));
   k1 = reservatorio2(H1(i),H2(i),S2,Ra,Rs,p,g,Qe);
   k2 = reservatorio2(H1(i)+h*0.5*k1,H2(i)+h*0.5*k1,S2,Ra,Rs,p,g,Qe);
   k3 = reservatorio2(H1(i)+h*0.5*k2,H2(i)+h*0.5*k2,S2,Ra,Rs,p,g,Qe);
   k4 = reservatorio2(H1(i)+h*k3,H2(i)+h*k3,S2,Ra,Rs,p,g,Qe);
   H2(i+1) = H2(i)+((k1+2*k2+2*k3+k4)*(h/6));
end

// Plotagem dos gráficos:
scf
plot2d([t,t],[H1,H2],[1 2]);
legends(["Reservatório 1","Reservatório 2"],[1,2],4)
xset('thickness',1);
xtitle("Método de Runge-Kutta","t","y");
xgrid(1)
```

Resultado:


Método de Runge-Kutta
y
t
0
0.5
1
1.5
2
2.5
3
3.5
4
0
10 000
20 000
30 000
40 000
50 000
Reservatório 1
Reservatório 2