Lista B de Modelagem PME3380

Nome: Yuri Lopes Pamplona

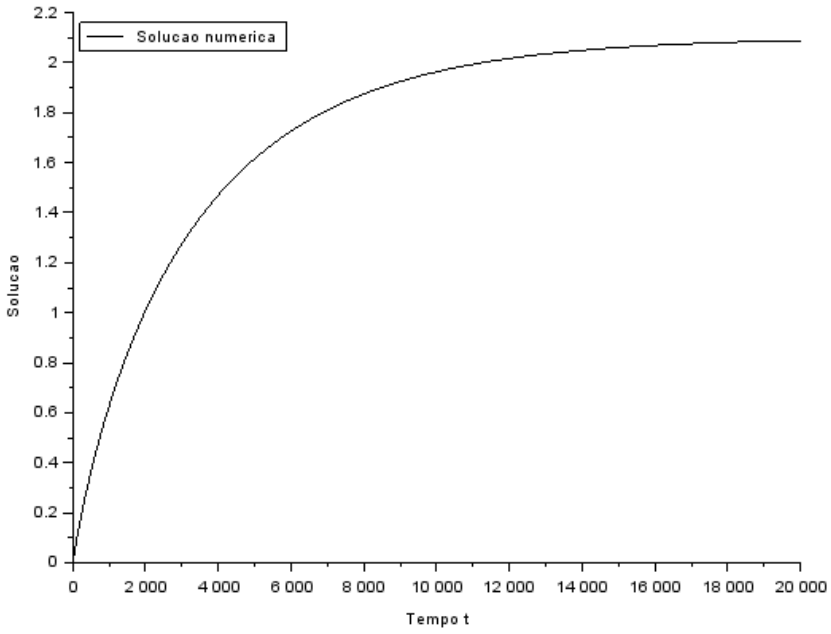
NºUSP: 10853498

Sumário

Exercício 1

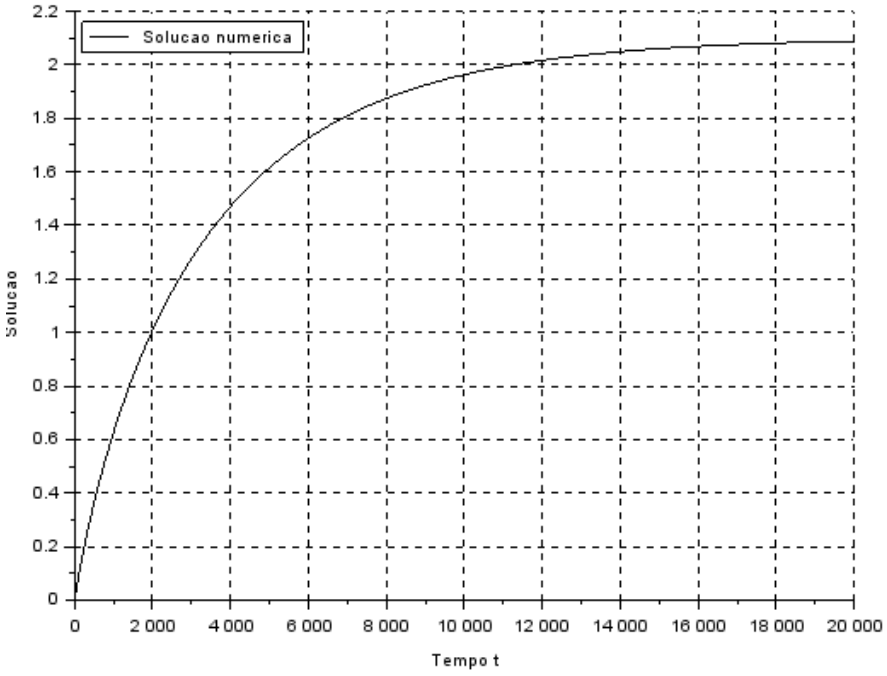
Para o exercício de um tanque único foi considerado que o tanque estava inicialmente vazio, uma modelagem tal que o tempo final fosse suficientemente longo para haver uma estabilização da altura e com passos de integração de 0.5. Observa-se para ambas as resoluções numéricas dado os dados do problema a altura estável foram de 2,1 metros.

Euler



1.Gráfico da Solução da altura pelo tempo pelo método de Euler.

Rugge-Kutta 4º Ordem

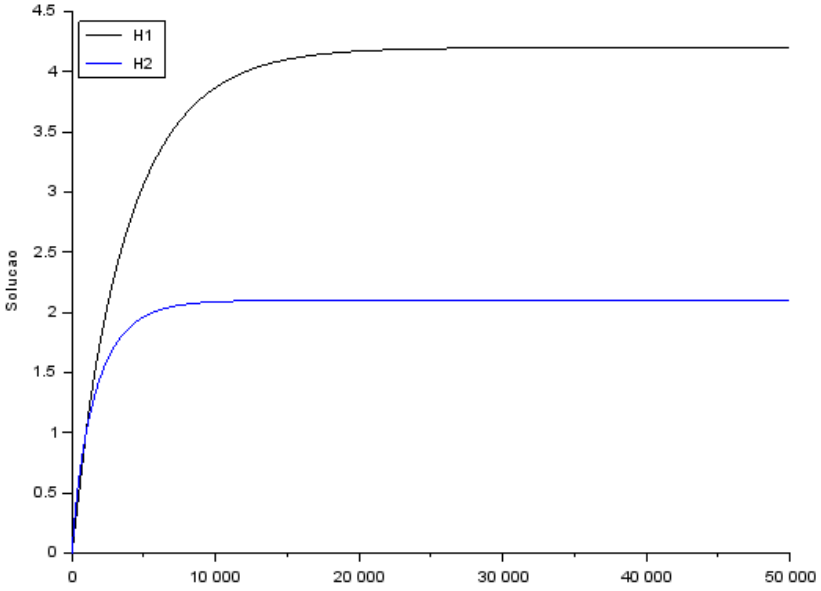


2.Gráfico da Solução da altura pelo tempo pelo método de Rugge-Kutta.

Exercício 2

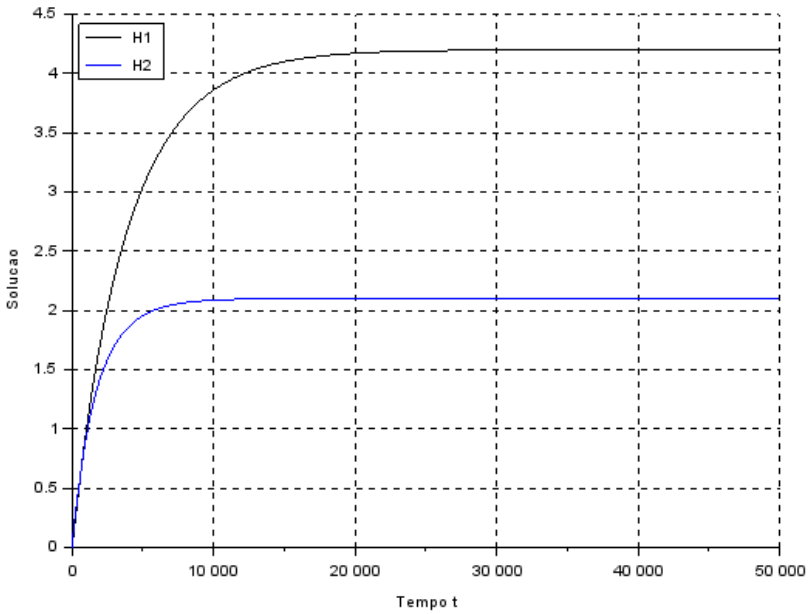
Para esse exercício foi implementado através de dois métodos de integração no Scilab a resolução do modelo para 2 tanques. Nesse modelo a altura inicial dos dois tanques é zero (inicialmente vazios). Os valores para Ra e Rs foram considerados iguais e foi considerado que o segundo tanque tem metade da área do primeiro tanque, onde o primeiro tanque e suas constantes são iguais aos do primeiro exercício. Observa-se que para o primeiro tanque há uma estabilização na altura = 4,25m e no segundo tanque uma estabilização em2,1 metros.

Euler



3.Gráfico da solução de dois reservatórios para o método de Euler

Rugge-Kutta 4º Ordem



4Gráfico para o método de Rugge-Kutta para dois reservatórios

Programa

Primeiro exercício

Função

```
function [ydot]=funcao(y)
   ydot = (-sqrt( rho*g*y/R) + Qe)/S ;
endfunction
```

Numérico Euler

```
// 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
// Variáveis
R = 2*10^8;
Qe = 0.010247;
rho = 1000;
S = 10;
g = 10;
// Instante inicial:
t(1)=0;
// Instante final:
tf=20000;
// 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:
load('funcao.sci')
y(i+1)=y(i)+h*funcao(y(i));
```

```
// Termino do comando for:
end

// Desenhando outro grafico com linhas diferentes:
plot2d([t],[y],[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)],[1,2],2);
// Colocando um titulo na figura e nomeando os eixos:
xtitle(T(1),T(2),T(3));
```

Numérico Rugge-Kutta:

```
// Conjunto de comandos para solucao numerica de equacao diferencial [1-
y(i)]/2
// Apagando dados anteriores:
clear
R = 2*10^8;
Qe = 0.010247;
rho = 1000;
S = 10;
g = 10;
// Instante inicial:
t(1)=0;
// Instante final:
tf=20000;
// 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*(-sqrt( rho*g*y(i)/R) + Qe)/S;
```

```
    k2=h*(-sqrt( rho*g*(y(i)+k1/2)/R) + Qe)/S;
    k3=h*(-sqrt( rho*g*(y(i)+k2/2)/R) + Qe)/S;
    k4=h*(-sqrt( rho*g*(y(i)+k3)/R) + Qe)/S;
    y(i+1)=y(i)+((k1+2*k2+2*k3+k4)/6);

// Termino do comando for:
end

// Desenhando outro grafico com linhas diferentes:
plot2d([t],[y],[1 2]);
// Usando a variavel do tipo 'lista':
T=list("Comparacao entre solucao numerica e solucao exata","Tempo
t","Solucao","Solucao numerica","Solucao exata");
// Diminuindo a espessura das linhas:
xset("thickness",1)
// Colocando uma legenda na parte superior esquerda da figura (parametro
2):
legends([T(4)],[1,2],2);
// Colocando um titulo na figura e nomeando os eixos:
xtitle(T(1),T(2),T(3));
// Colocando uma grade no grafico:
xgrid(1)
```

Segundo exercício

Numérico Euler

```
// 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
// Variáveis
Ra = 2*10^8;
Rs = 2*10^8;
Qe = 0.010247;
rho = 1000;
S1 = 10;
S2 = 5;
g = 10;
// Instante inicial:
t(1)=0;

// Instante final:
tf=50000;
```

```
// Condicao inicial:
h1(1)=0;
h2(1)=0;
// Passo de integracao (experimente alterar o passo):
p=0.5;
// Calculo de numero de passos):
n=round(tf/p);
// Integracao numerica usando o metodo de Euler:
// Comando for:
function [hdot]=funcao(h1, h2)
   hdot = (-sqrt(rho*g*(h1-h2)/Ra) + Qe)/S1 ;
endfunction
function [h2dot]=funcao2(h1, h2)
   h2dot = (-sqrt( rho*g*(h1-h2)/Ra) -sqrt( rho*g*h2/Rs) + Qe)/S2 ;
endfunction
for i=1:n
   // Vetor de tempo:
   t(i+1)=t(i)+p;
   // Solucao numerica:
   //load('funcao.sci');
   h1(i+1)=h1(i) + p*funcao(h1(i),h2(i));
   //load('funcao2.sci');
   h2(i+1)=h2(i)+p*funcao2(h2(i),h2(i));

// Termino do comando for:
end

// Desenhando outro grafico com linhas diferentes:
plot2d([t],[h1,h2],[1 2]);
// Usando a variavel do tipo 'lista':
T=list("Solucao numerica","Tempo t","Solucao","H1","H2");
// 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));
```

Numérico Rugge-Kutta:

```
// Conjunto de comandos para solucao numerica de equacao diferencial [1-
y(i)]/2
// Apagando dados anteriores:
clear
Ra = 2*10^8;
Rs = 2*10^8;
Qe = 0.010247;
rho = 1000;
```

```
S1 = 10;
S2 = 5;
g = 10;
function [hdot]=funcao1(h1, h2)
   hdot = (-sqrt(rho*g*(h1-h2)/Ra) + Qe)/S1 ;
endfunction
function [h2dot]=funcao2(h1, h2)
   h2dot = (-sqrt( rho*g*(h2-h1)/Ra) -sqrt( rho*g*h2/Rs) + Qe)/S2 ;
endfunction
// Instante inicial:
t(1)=0;
// Instante final:
tf=50000;
// Condicao inicial:
h1(1)=0;
h2(1)=0;

// Passo de integracao (experimente alterar o passo):
p=0.5;
// Calculo de numero de passos):
n=round((tf-t(1))/p);
// Integracao numerica usando o metodo de Runge Kutta:
// Comando for:
for i=1:n
   // Vetor de tempo:
   t(i+1)=t(i)+p;
   // Solucao numerica:
   k1=p*funcao2(h1(i),h2(i));
   k2=p*funcao2(h1(i)+k1/2,h2(i)+k1/2);
   k3=p*funcao2(h1(i)+k2/2,h2(i)+k2/2);
   k4=p*funcao2(h1(i)+k3,h2(i)+k3);
   h2(i+1)=h2(i)+((k1+2*k2+2*k3+k4)/6);

   a1=p*funcao1(h1(i),h2(i));
   a2=p*funcao1(h1(i)+a1/2,h2(i)+a1/2);
   a3=p*funcao1(h1(i)+a2/2,h2(i)+a2/2);
   a4=p*funcao1(h1(i)+a3,h2(i)+a3);
   h1(i+1)=h1(i)+((a1+2*a2+2*a3+a4)/6);

// Termino do comando for:
end

// Desenhando outro grafico com linhas diferentes:
plot2d([t,t],[h1,h2],[1 2]);
// Usando a variavel do tipo 'lista':
```

```
T=list("Comparacao entre solucao numerica e solucao exata","Tempo
t","Solucao","H1",”H2");
// 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 os eixos:
xtitle(T(1),T(2),T(3));
// Colocando uma grade no grafico:
xgrid(1)
```