Alessandra da Cruz Nunes de Moraes
10337209

PME3380 - Modelagens de Sistemas Dinâmicos
Lista B

São Paulo

2020

Lista de ilustrações

Sumário

1 Exemplos

1.1 Exemplo 1

O primeiro exemplo dado pede para implementar a função e o arquivo no software Scilab:

A função é:

```
function [ydot]=funcao(y)
ydot=(1-y)/2;
endfunction
```

E o arquivo:

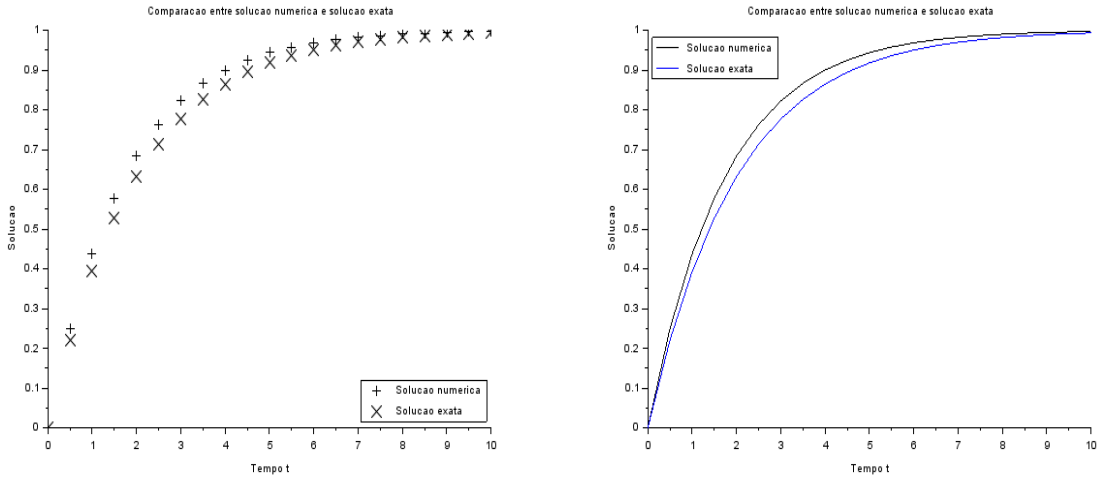
```
// Conjunto de comandos para solucao numerica de equacao diferencial dada
   pela funcao funcao.sci
// Apagando dados anteriores:
clear
// Carregando a equacao diferencial:
load('funcao.sci')
// Carregue a função usando o comando Load do Scilab
// Instante inicial:
t(1)=0;
// Instante final:
tf=10;
// 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
```

```
30 // Plotando solucao numerica y versus vetor de tempo t e solucao exata ye
       versus vetor de tempo t:
31 plot2d([t,t],[y,ye],[-1 -2]);
32 // Colocando uma legenda na parte inferior direito da figura (parametro 4):
33 legends(["Solucao numerica","Solucao exata"],[-1,-2],4)
34 // Colocando um titulo na figura e nomeando os eixos:
35 xtitle("Comparacao entre solucao numerica e solucao exata","Tempo t","
       Solucao")
36 // Abrindo uma nova janela de graficos:
37 set("current_figure",1);
38 // Desenhando outro grafico com linhas diferentes:
39 plot2d([t,t],[y,ye],[1 2]);
40 // Usando a variavel do tipo 'lista':
41 T=list("Comparacao entre solucao numerica e solucao exata","Tempo t","
       Solucao","Solucao
42 numerica","Solucao exata");
43 // Colocando uma legenda na parte superior esquerda da figura (parametro 2)
       :
44 legends([T(4),T(5)],[1,2],2);
45 // Colocando um titulo na figura e nomeando os eixos:
46 xtitle(T(1),T(2),T(3));
```

1.1.1 Resultados com passo 0.5

Testando o código com o passo 0.5, foram plotados os seguintes gráficos:

Figura 1 – Gráficos Gerados Com Passo 0.5

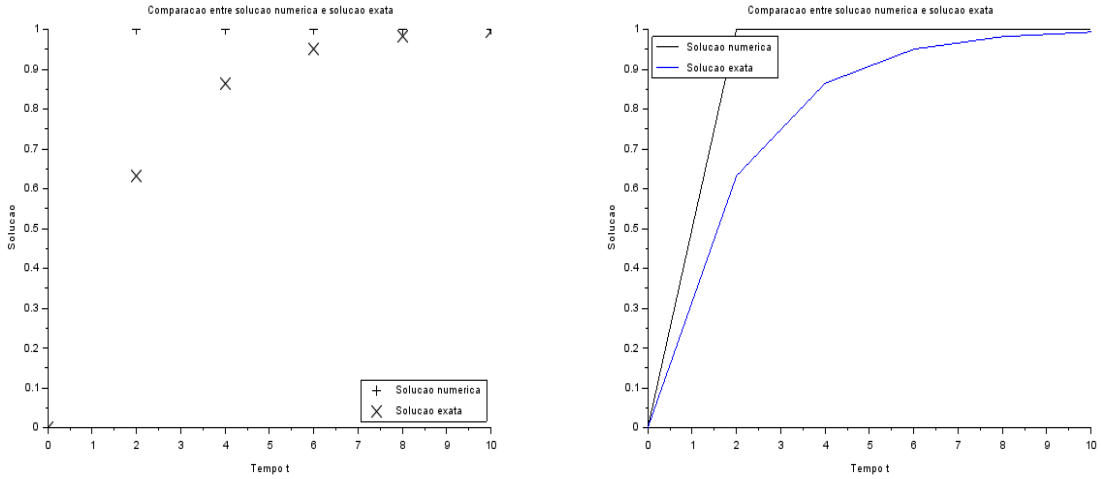


Fonte: "Scilab"

1.1.2 Passo 2

Para fins comparativos, o arquivo foi alterado e o passo foi aumentado para 2.

Figura 2 – Gráficos Gerados Com Passo 2

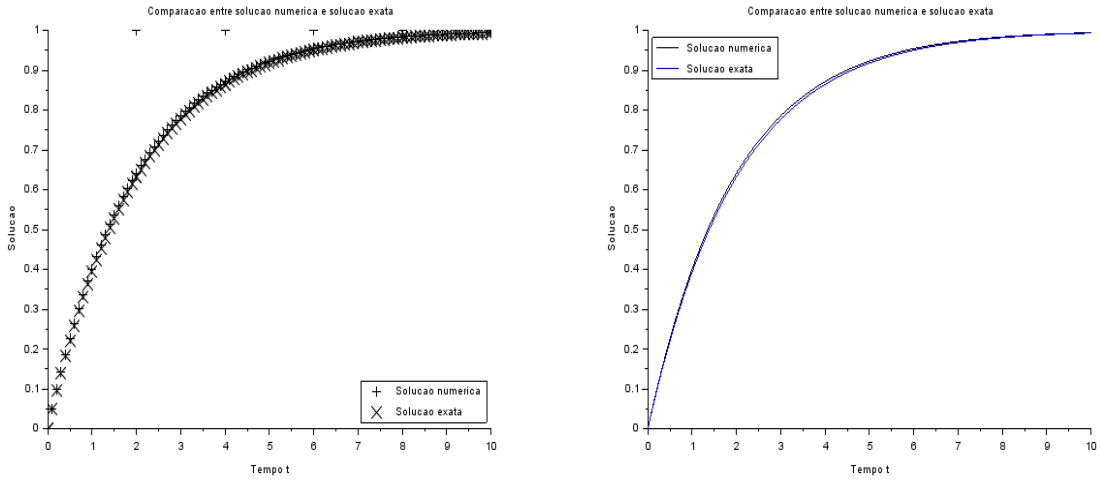


Fonte: "Scilab"

1.1.3 Resultados com passo 0.1

Para fins comparativos, o arquivo foi alterado e o passo foi diminuido para 0.1.

Figura 3 – Gráficos Gerados Com Passo 0.1



Fonte: "Scilab"

O que se pode perceber é que quanto menor o número de passos, mais pontos no gráfico e mais próximas as soluções numéricas e exatas.

1.2 Exemplo 2

No exemplo 2 resolve-se o problema em um código único, sem a necessidade de utilizar o comando "load"para carregar a função salva em outro arquivo.

O código fica:

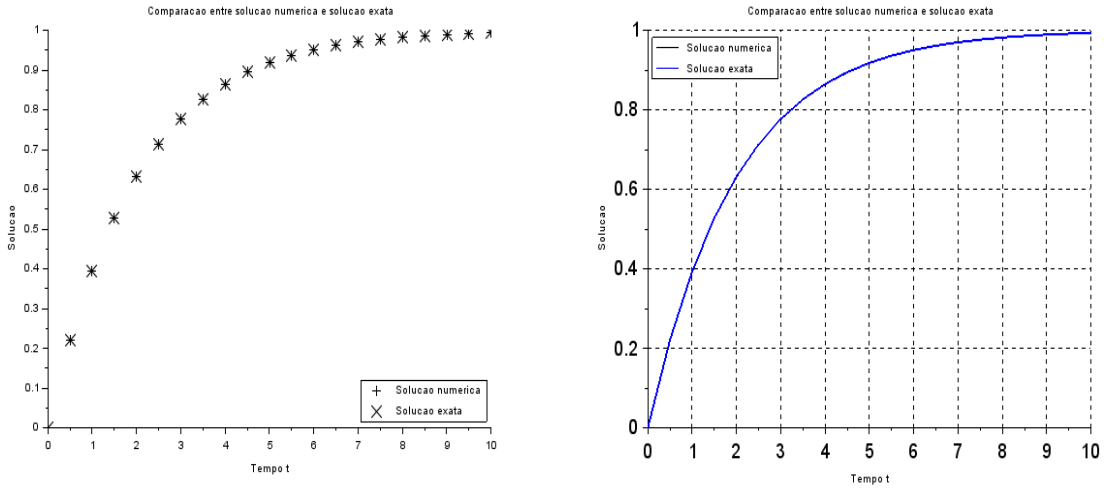
```
// Conjunto de comandos para solucao numerica de equacao diferencial [1-y(i)]/2
// Apagando dados anteriores:
clear
// Instante inicial:
t(1)=0;
// Instante final:
tf=10;
// Condicao inicial:
y(1)=0;
// Valor inicial da solucao exata:
ye(1)=0;
// Passo de integracao (experimente alterar o passo):
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(["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);
```

```
39 // Aumentando a espessura das linhas:
40 xset("thickness",2)
41 // Aumentando o tamanho da fonte:
42 xset("font size",4)
43 // Desenhando outro grafico com linhas diferentes:
44 plot2d([t,t],[y,ye],[1  2]);
45 // Usando a variavel do tipo 'lista':
46 T=list("Comparacao entre solucao numerica e solucao exata","Tempo t","
    Solucao","Solucao numerica","Solucao exata");
47 // Diminuindo a espessura das linhas:
48 xset("thickness",1)
49 // Colocando uma legenda na parte superior esquerda da figura (parametro 2)
    :
50 legends([T(4),T(5)],[1,2],2);
51 // Colocando um titulo na figura e nomeando os eixos:
52 xtitle(T(1),T(2),T(3));
53 // Colocando uma grade no grafico:
54 xgrid(1)
```

1.2.1 Resultados com passo 0.5

Testando o código com o passo 0.5, foram plotados os seguintes gráficos:

Figura 4 – Gráficos Gerados Com Passo 0.5

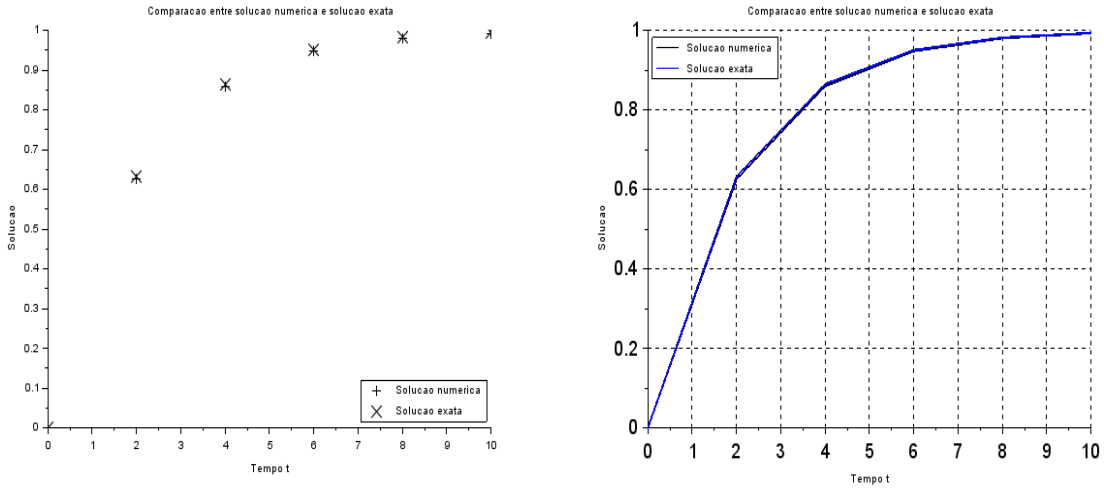


Fonte: "Scilab"

1.2.2 Resultados com passo 2

Ao alterar o passo para 2, os gráficos plotados foram diferentes:

Figura 5 – Gráficos Gerados Com Passo 2

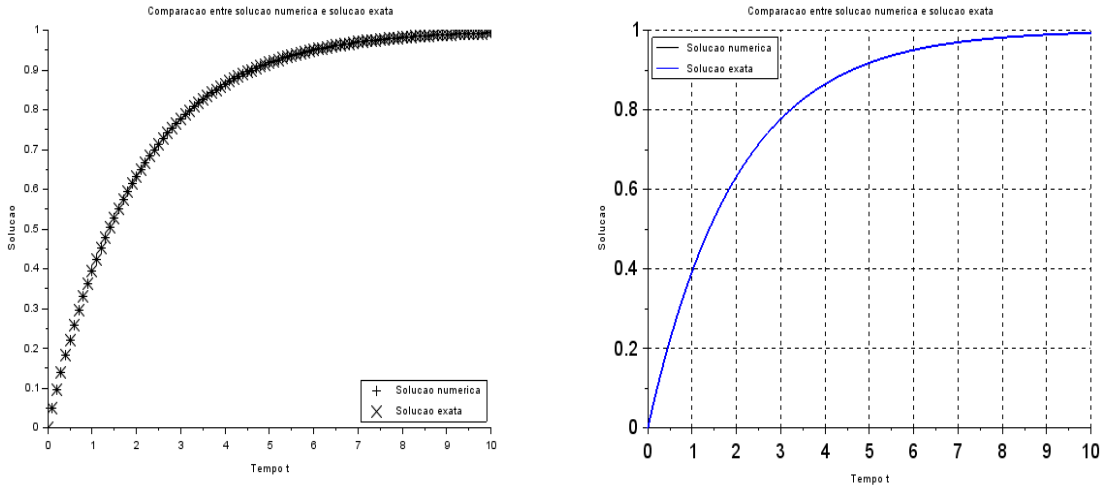


Fonte: "Scilab"

1.2.3 Resultados com passo 0.1

Ao alterar o passo para 0.1, os gráficos plotados foram diferentes:

Figura 6 – Gráficos Gerados Com Passo 0.1



Fonte: "Scilab"

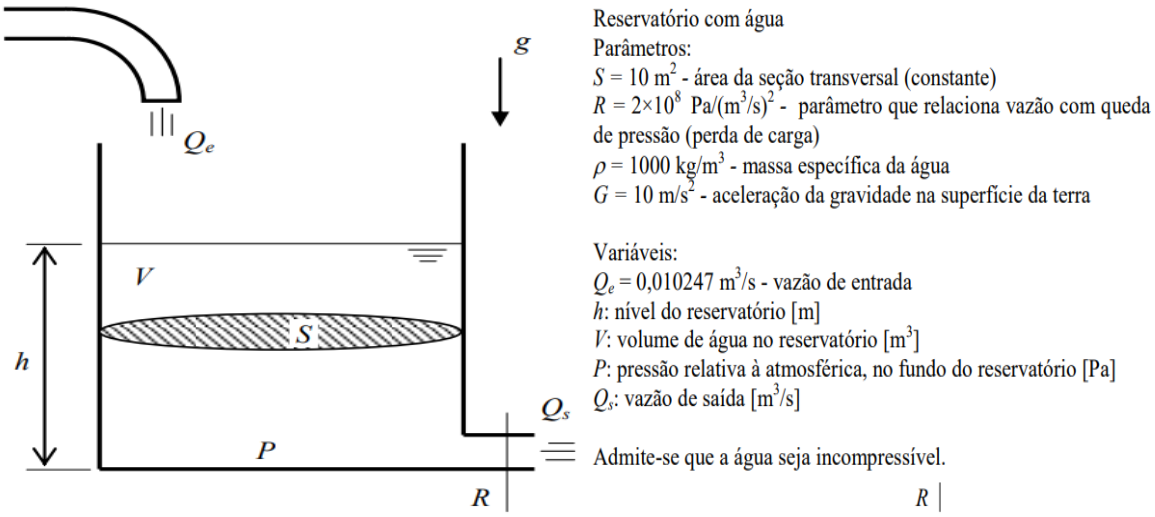
Novamente, observa-se que quando menor o passo, mais pontos no gráfico e mais as soluções se aproximam.

2 Exercícios

2.1 Reservatório Simples

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

Figura 7 – Esquema do Primeiro Problema



2.1.1 Por Euler

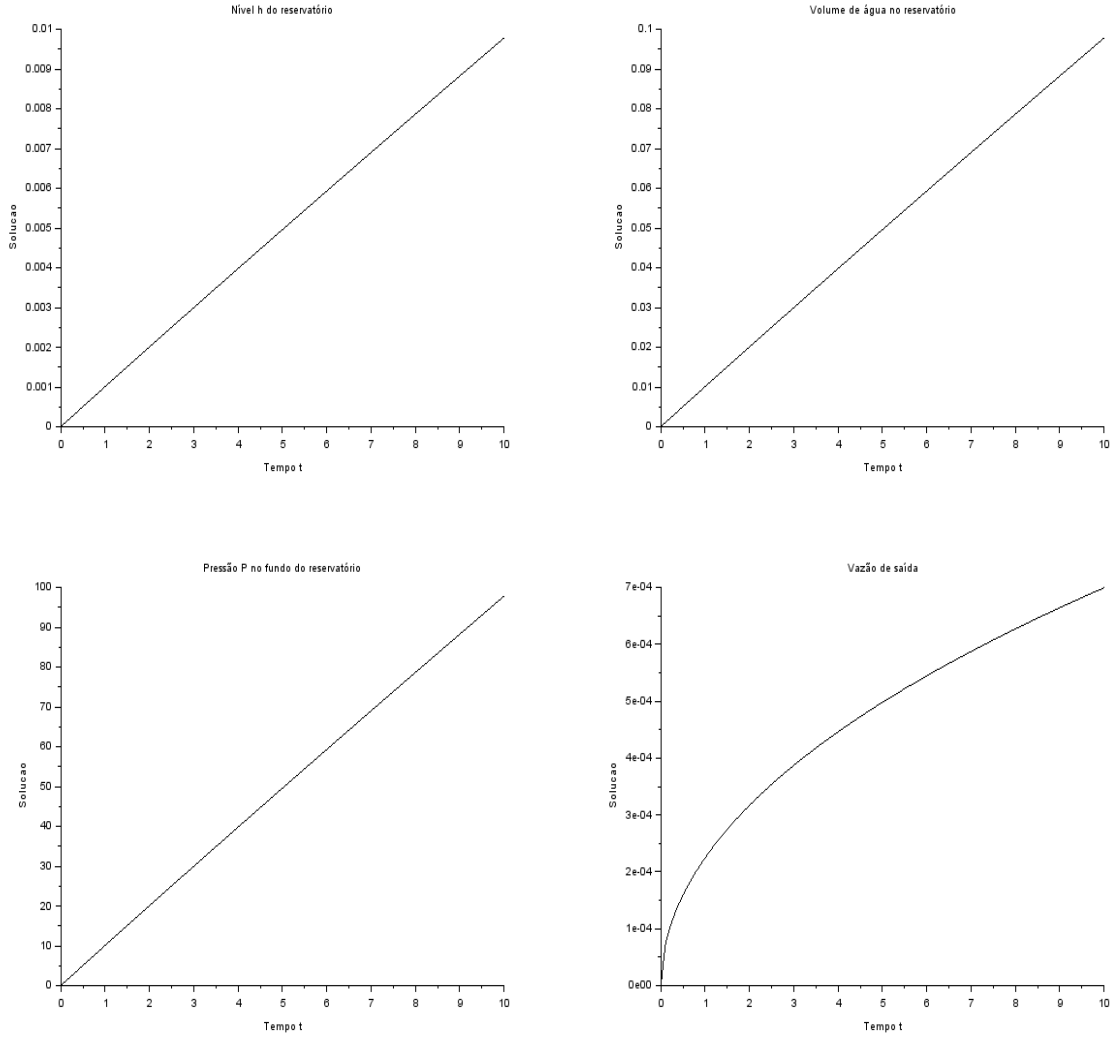
Pelo método de Euler, foi implementado o seguinte código:

```
// Apagando dados anteriores:
clear
S = 10 //m2 - área da seção transversal (constante)
R = 2e8 //Pa/(m3/s)2 - par metro que relaciona vazão com queda de pressão
    (perda de carga)
p = 1e3 //kg/m3 - massa específica da água
g = 10 //m/s2 - aceleração da gravidade na superfície da terra
Qe = 0.010247 //m3/s - vazão de entrada

//Achar
//h (nível do reservatório [m])
//V (volume de água no reservatório [m3])
//P (pressão relativa à atmosférica, no fundo do reservatório [Pa])
//Qs (vazão de saída [m3/s])
//Admite-se que a água seja incompressível.

```

```
16 function ydot=funcao(y)
17 ydot=(-sqrt(p*g*y/R)+Qe)/S;
18 endfunction
19
20 // Instante inicial:
21 t(1)=0;
22 // Instante final:
23 tf=10;
24 // Condicao inicial:
25 y(1)=0;
26 // Valor inicial da solucao exata:
27 ye(1)=0;
28 // Passo de integracao (experimente alterar o passo):
29 h=0.1;
30 // Calculo de numero de passos:
31 n=round(tf/h);
32 // Integracao numerica usando o metodo de Euler:
33 // Comando for:
34 for i=1:n
35 // Vetor de tempo:
36 t(i+1)=t(i)+h;
37 // Solucao numerica:
38 y(i+1)=y(i)+h*funcao(y(i));
39 // Termino do comando for:
40 end
41
42 V = S*y;
43 P = p*g*y;
44 Qs = sqrt(P/R);
45
46 // Plotando solucao numerica y versus vetor de tempo t:
47 f(1)=scf(1);
48 plot2d([t],[y]);
49 xtitle("Nível h do reservatório","Tempo t","Solucao")
50
51 f(2)=scf(2);
52 plot2d([t],[V]);
53 xtitle("Volume de água no reservatório","Tempo t","Solucao")
54
55 f(3)=scf(3);
56 plot2d([t],[P]);
57 xtitle("Pressão P no fundo do reservatório","Tempo t","Solucao")
58
59 f(4)=scf(4);
60 plot2d([t],[Qs]);
61 xtitle("Vazão de saída","Tempo t","Solucao")
```

A execução do código gerou os seguintes gráficos:

Figura 8 – Resultados Gerados pelo Método de Euler



2.1.2 Por Runge Kutta

Para resolver por Runge Kutta, foi implementado o seguinte código:

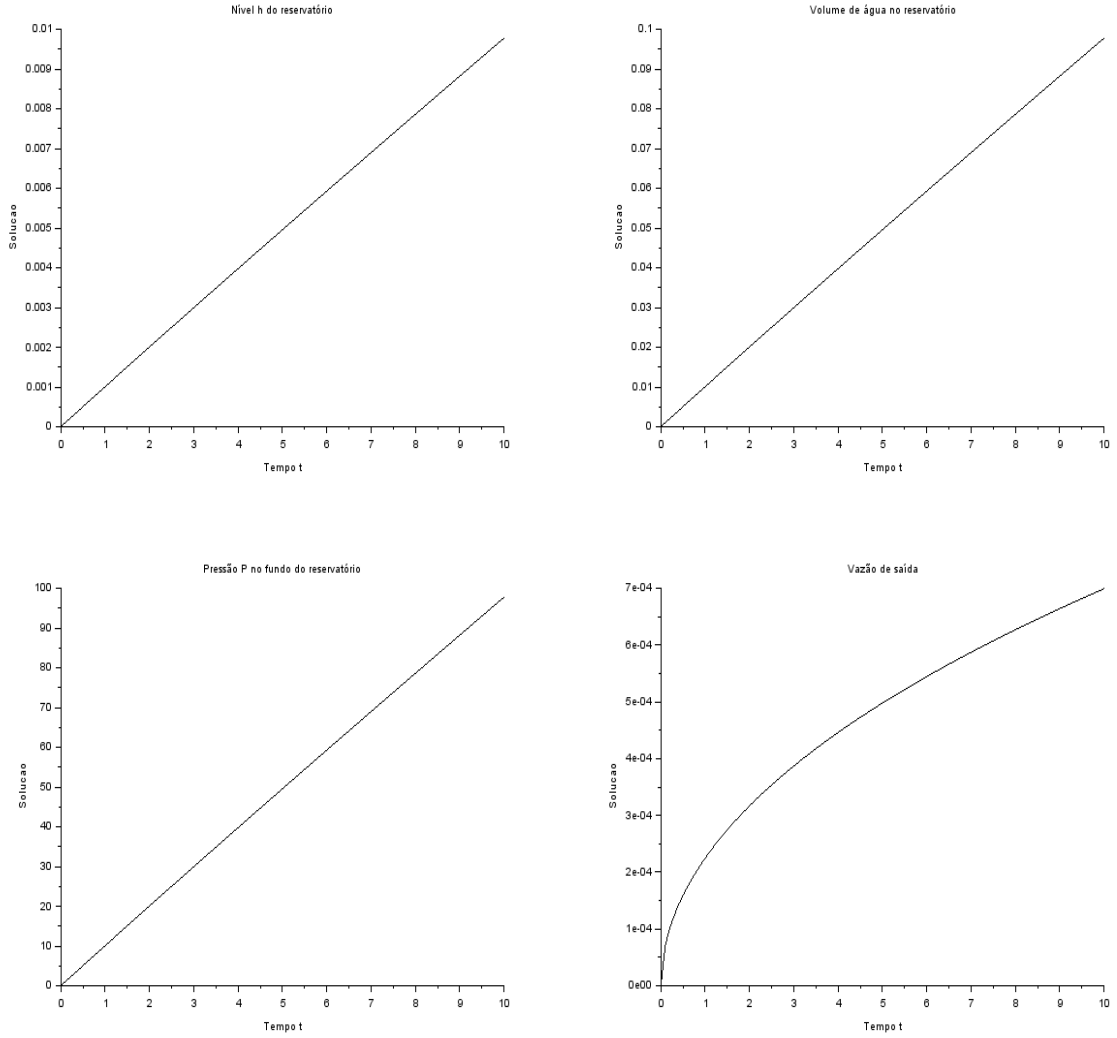
```
// Apagando dados anteriores:
clear
S = 10 //m2 - área da seção transversal (constante)
R = 2e8 //Pa/(m3/s)2 - par metro que relaciona vazão com queda de pressão
    (perda de carga)
p = 1e3 //kg/m3 - massa específica da água
g = 10 //m/s2 - aceleração da gravidade na superfície da terra
Qe = 0.010247 //m3/s - vazão de entrada

//Achar
//h (nível do reservatório [m])
```

```
//V (volume de água no reservatório [m3])
//P (pressão relativa à atmosférica, no fundo do reservatório [Pa])
//Qs (vazão de saída [m3/s])
//Admite-se que a água seja incompressível.

// 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=0.1;
// Calculo de numero de passos:
n=round(tf/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=(-sqrt(p*g*y(i)/R)+Qe)/S
k2=(-sqrt(p*g*(y(i)+h*k1/2)/R)+Qe)/S
k3=(-sqrt(p*g*(y(i)+h*k2/2)/R)+Qe)/S
k4=(-sqrt(p*g*(y(i)+h*k3)/R)+Qe)/S

//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)*h/6);

// Termino do comando for:
end

V = S*y
P = p*g*y
Qs = sqrt(P/R)

// Plotando solucao numerica y versus vetor de tempo t:
f(1)=scf(1);
plot2d([t],[y]);
xtitle("Nível h do reservatório","Tempo t","Solucao")
```

```
58
59 f(2)=scf(2);
60 plot2d([t],[V]);
61 xtitle("Volume de água no reservatório","Tempo t","Solucao")
62
63 f(3)=scf(3);
64 plot2d([t],[P]);
65 xtitle("Pressão P no fundo do reservatório","Tempo t","Solucao")
66
67 f(4)=scf(4);
68 plot2d([t],[Qs]);
69 xtitle("Vazão de saída","Tempo t","Solucao")
```

A execução do código gerou os seguintes gráficos:

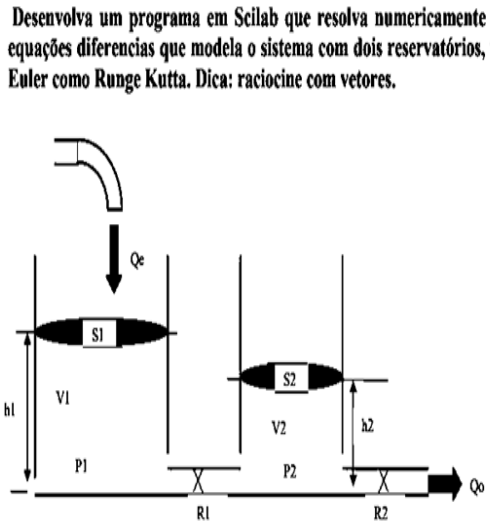
Figura 9 – Resultados Gerados pelo Método de Runge Kutta



Observa-se que os resultados são muito semelhantes.

2.2 Dois Reservatórios

Figura 10 – Problema com 2 Reservatórios



Implementando o seguinte código para Resolver por Euler:

```
// Apagando dados anteriores:
clear
S1 = 10 //m2 - área da seção transversal (constante)
S2 = 2 //m2 - área da seção transversal (constante)
R1 = 2e8 //Pa/(m3/s)2 - par metro que relaciona vazão com queda de pressão
    (perda de carga)
R2 = 1.6e8 //Pa/(m3/s)2 - par metro que relaciona vazão com queda de press
   ão (perda de carga)
p = 1e3 //kg/m3 - massa específica da água
g = 10 //m/s2 - aceleração da gravidade na superfície da terra
Qe = 0.010247 //m3/s - vazão de entrada

//Achar
//h (nível do reservatório [m])
//V (volume de água no reservatório [m3])
//P (pressão relativa à atmosférica, no fundo do reservatório [Pa])
//Qs (vazão de saída [m3/s])
//Admite-se que a água seja incompressível.

function hdot1=hp1(h1, h2)
hdot1=(Qe-sqrt(p*g*(h1-h2)/R1))/S1;
endfunction

function hdot2=hp2(h1, h2)
hdot2=(sqrt(p*g*(h1-h2)/R1)-sqrt(p*g*h2/R2))/S2;
endfunction

```

```
// Instante inicial:
t(1)=0;
// Instante final:
tf=10;
// Condicoes iniciais:
h1(1)=0;
h2(1)=0;
// Passo de integracao (experimente alterar o passo):
h=0.1;
// 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:
h1(i+1)=h1(i)+h*hp1(h1(i),h2(i));
h2(i+1)=h2(i)+h*hp2(h1(i+1),h2(i)); //colocando h1(i+1) para usar um dado
    mais atualizado
// Termino do comando for:
end

V1 = S1*h1;
P1 = p*g*h1;
Qs1 = sqrt(P1/R1);

V2 = S2*h2;
P2 = p*g*h2;
Qs2 = sqrt(P2/R2);

// Plotando solucao numerica y versus vetor de tempo t:
f(1)=scf(1);
plot2d([t,t],[h1,h2],[1 2]);
legends(["Altura h1","Altura h2"],[1,2],2)
xtitle("Comparacao das alturas dos reservatórios","Tempo t","Solucao")
f(2)=scf(2);
plot2d([t,t],[V1,V2], [1 2]);
xtitle("Comparacao dos Volumes dos reservatórios","Tempo t","Solucao")
f(3)=scf(3);
plot2d([t,t],[P1,P2], [1 2]);
xtitle("Comparacao das pressões dos reservatórios","Tempo t","Solucao")
f(4)=scf(4);
plot2d([t,t],[Qs1, Qs2], [1 2]);
xtitle("Comparacao das vazões dos reservatórios","Tempo t","Solucao")
```

Obtemos os seguintes gráficos:

Figura 11 – Resultados do problema 2 por Euler