

Alessandra da Cruz Nunes de Moraes  
10337209

**PME3380 - Modelagens de Sistemas Dinâmicos**  
**Lista B**

São Paulo

2020

# Lista de ilustrações

Figura 1 – Gráficos Gerados Com Passo 0.5 . . . . .	4
Figura 2 – Gráficos Gerados Com Passo 2 . . . . .	5
Figura 3 – Gráficos Gerados Com Passo 0.1 . . . . .	5
Figura 4 – Gráficos Gerados Com Passo 0.5 . . . . .	7
Figura 5 – Gráficos Gerados Com Passo 2 . . . . .	8
Figura 6 – Gráficos Gerados Com Passo 0.1 . . . . .	8
Figura 7 – Esquema do Primeiro Problema . . . . .	9
Figura 8 – Resultados Gerados pelo Método de Euler . . . . .	11
Figura 9 – Resultados Gerados pelo Método de Runge Kutta . . . . .	13
Figura 10 – Problema com 2 Reservatórios . . . . .	14
Figura 11 – Resultados do problema 2 por Euler . . . . .	16

# Sumário

<b>1</b>	<b>EXEMPLOS</b>	<b>3</b>
<b>1.1</b>	<b>Exemplo 1</b>	<b>3</b>
1.1.1	Resultados com passo 0.5	4
1.1.2	Passo 2	5
1.1.3	Resultados com passo 0.1	5
<b>1.2</b>	<b>Exemplo 2</b>	<b>6</b>
1.2.1	Resultados com passo 0.5	7
1.2.2	Resultados com passo 2	7
1.2.3	Resultados com passo 0.1	8
<b>2</b>	<b>EXERCÍCIOS</b>	<b>9</b>
<b>2.1</b>	<b>Reservatório Simples</b>	<b>9</b>
2.1.1	Por Euler	9
2.1.2	Por Runge Kutta	11
<b>2.2</b>	<b>Dois Reservatórios</b>	<b>14</b>

# 1 Exemplos

## 1.1 Exemplo 1

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

A função é:

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

E o arquivo:

```
1 // Conjunto de comandos para solucao numerica de equacao diferencial dada
   pela funcao funcao.sci
2 // Apagando dados anteriores:
3 clear
4 // Carregando a equacao diferencial:
5 load('funcao.sci')
6 // Carregue a função usando o comando Load do Scilab
7 // Instante inicial:
8 t(1)=0;
9 // Instante final:
10 tf=10;
11 // Condicao inicial:
12 y(1)=0;
13 // Valor inicial da solucao exata:
14 ye(1)=0;
15 // Passo de integracao (experimente alterar o passo):
16 h=0.5;
17 // Calculo de numero de passos):
18 n=round(tf/h);
19 // Integracao numerica usando o metodo de Euler:
20 // Comando for:
21 for i=1:n
22 // Vetor de tempo:
23 t(i+1)=t(i)+h;
24 // Solucao numerica:
25 y(i+1)=y(i)+h* funcao(y(i));
26 // Solucao exata:
27 ye(i+1)=1-exp(-t(i+1)/2);
28 // Termina do comando for:
29 end
```

```

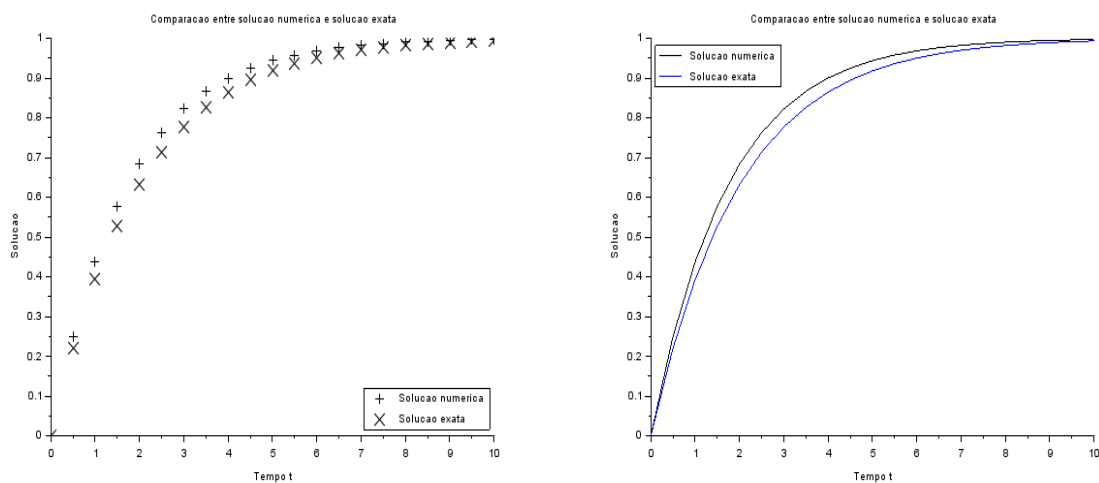
30 // Plotando solucao numerica y versus vetor de tempo t e solucao exata ye
    versus vetor de tempo t:
31 plot2d([t,t],[y,ye],[-1 -2]);
32 // Colocando uma legenda na parte inferior direito da figura (parametro 4):
33 legends(["Solucao numerica","Solucao exata"],[-1,-2],4)
34 // Colocando um titulo 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

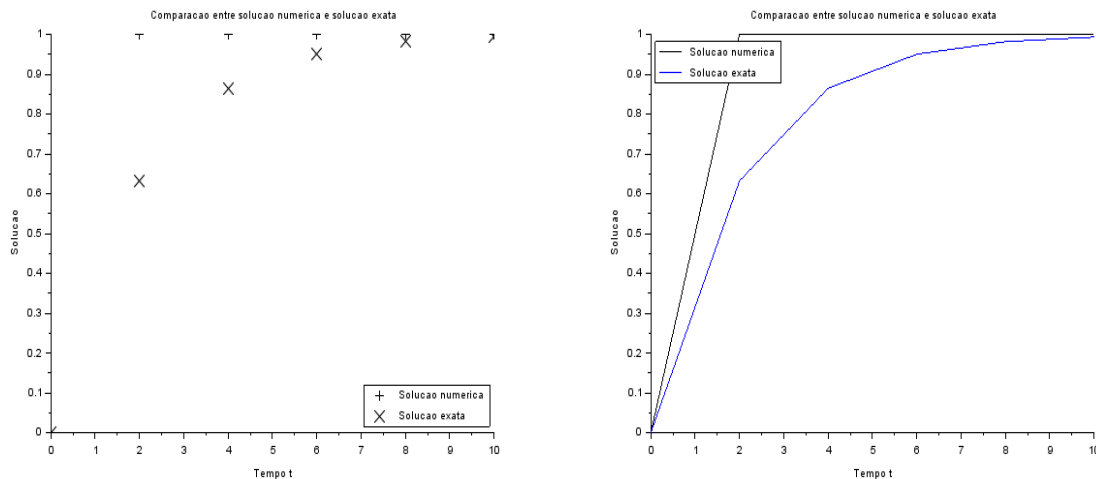


Fonte: "Scilab"

### 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

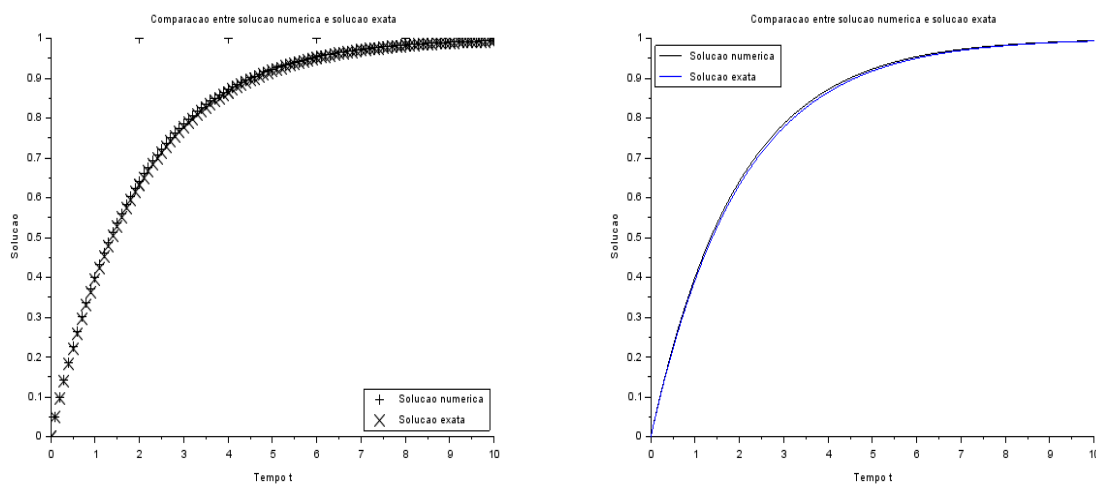


Fonte: "Scilab"

### 1.1.3 Resultados com passo 0.1

Para fins comparativos, o arquivo foi alterado e o passo foi diminuído 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:

```
1 // Conjunto de comandos para solucao numerica de equacao diferencial [1-y(i
  )]/2
2 // Apagando dados anteriores:
3 clear
4 // Instante inicial:
5 t(1)=0;
6 // Instante final:
7 tf=10;
8 // Condicao inicial:
9 y(1)=0;
10 // Valor inicial da solucao exata:
11 ye(1)=0;
12 // Passo de integracao (experimente alterar o passo):
13 h=0.5;
14 // Calculo de numero de passos):
15 n=round((tf-t(1))/h);
16 // Integracao numerica usando o metodo de Runge Kutta:
17 // Comando for:
18 for i=1:n
19 // Vetor de tempo:
20 t(i+1)=t(i)+h;
21 // Solucao numerica:
22 k1=h*(1-(y(i)))/2;
23 k2=h*(1-(y(i)+k1/2))/2;
24 k3=h*(1-(y(i)+k2/2))/2;
25 k4=h*(1-(y(i)+k3))/2;
26 y(i+1)=y(i)+((k1+2*k2+2*k3+k4)/6);
27 // Solucao exata:
28 ye(i+1)=1-exp(-t(i+1)/2);
29 // Termina do comando for:
30 end
31 // Plotando solucao numerica y versus vetor de tempo t e solucao exata ye
  versus vetor de tempo t:
32 plot2d([t,t],[y,ye],[-1 -2]);
33 // Colocando uma legenda na parte inferior direito da figura (parametro 4):
34 legends(["Solucao numerica","Solucao exata"],[-1,-2],4)
35 // Colocando um titulo na figura e nomeando os eixos:
36 xtitle("Comparacao entre solucao numerica e solucao exata","Tempo t","
  Solucao")
37 // Abrindo uma nova janela de graficos:
38 set("current_figure", 1);
```

```

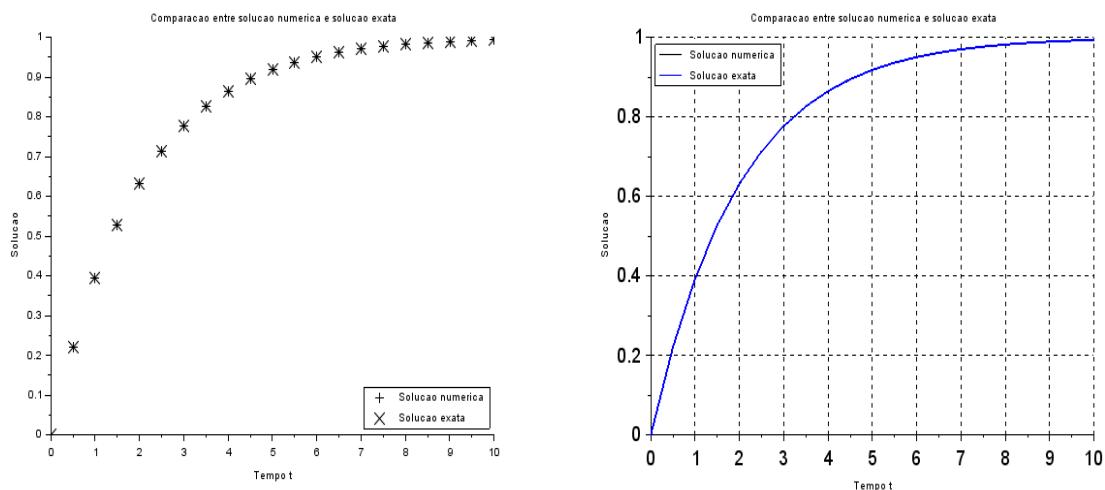
39 // Aumentando a espessura das linhas:
40 xset("thickness",2)
41 // Aumentando o tamanho da fonte:
42 xset("font size",4)
43 // Desenhando outro grafico com linhas diferentes:
44 plot2d([t,t],[y,ye],[1 2]);
45 // Usando a variavel do tipo 'lista':
46 T=list("Comparacao entre solucao numerica e solucao 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



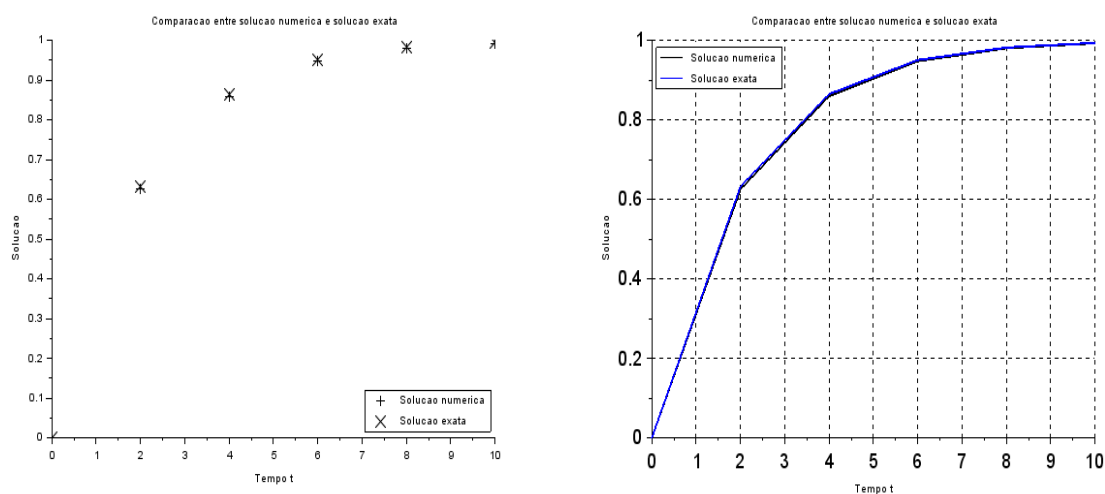
Fonte: "Scilab"

### 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

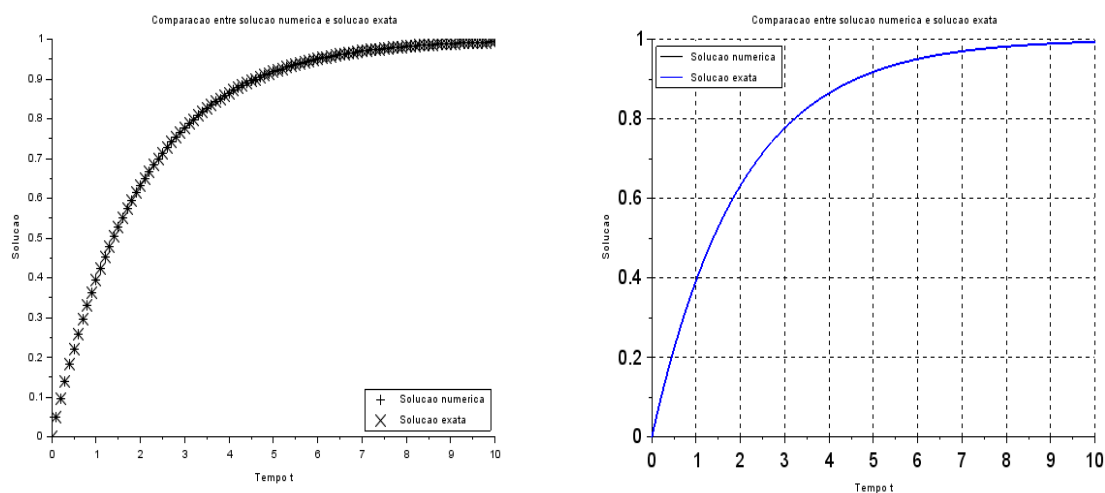


Fonte: "Scilab"

### 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"

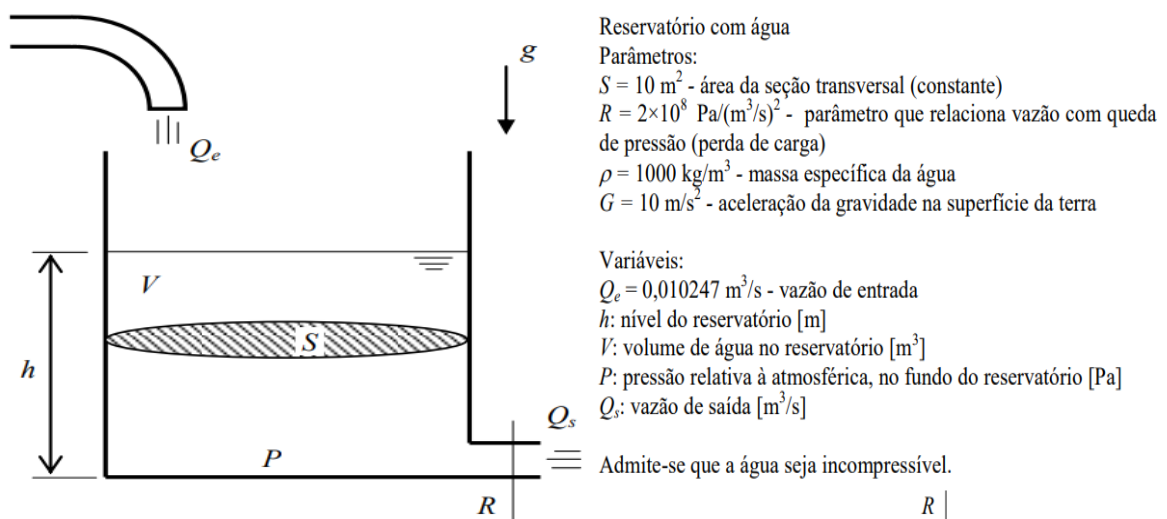
Novamente, observa-se que quando menor o passo, mais pontos no gráfico e mais as soluções se aproximam.

## 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:

```

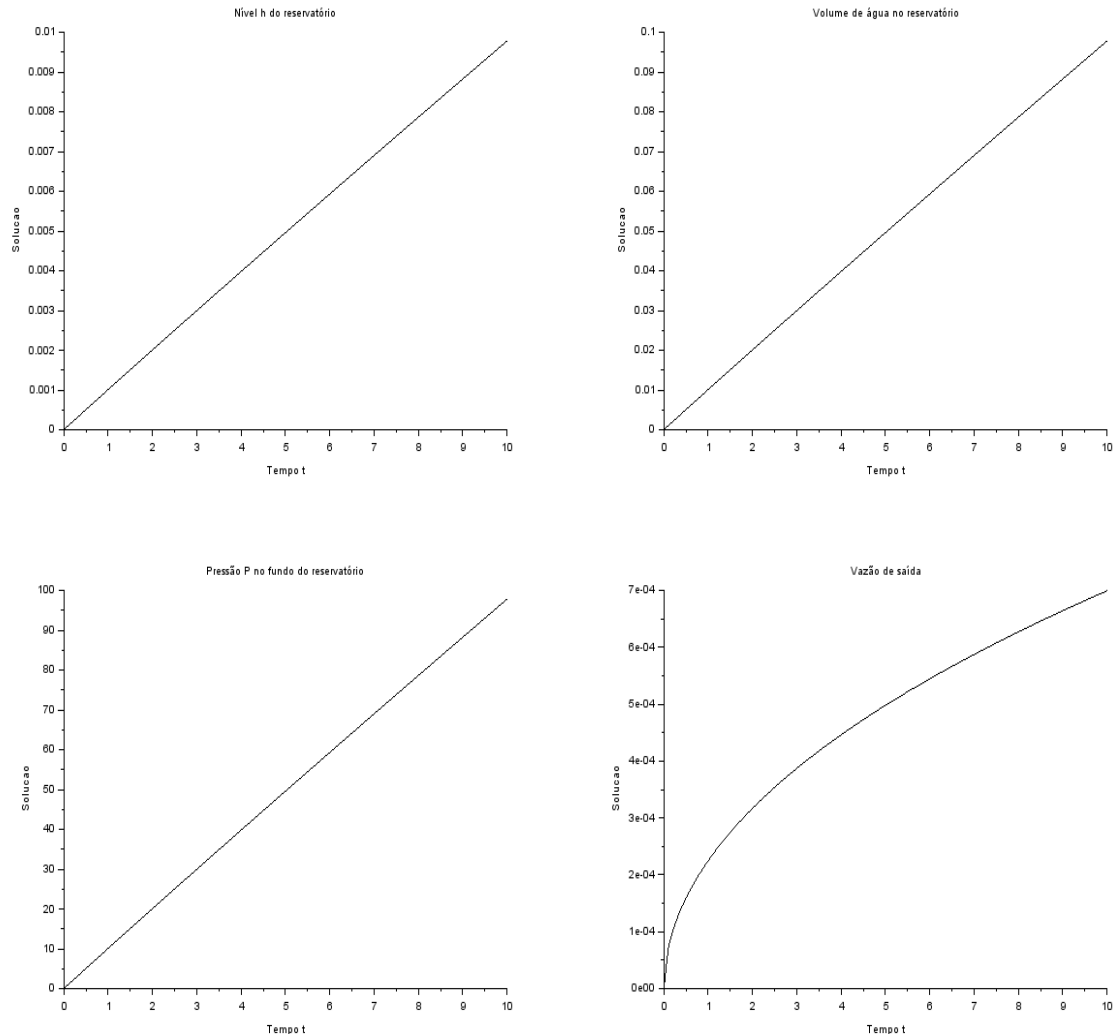
1 // Apagando dados anteriores:
2 clear
3 S = 10 //m2 - área da seção transversal (constante)
4 R = 2e8 //Pa/(m3/s)2 - par metro que relaciona vazão com queda de pressão
   (perda de carga)
5 p = 1e3 //kg/m3 - massa específica da água
6 g = 10 //m/s2 - aceleração da gravidade na superfície da terra
7 Qe = 0.010247 //m3/s - vazão de entrada
8
9 //Achar
10 //h (nível do reservatório [m])
11 //V (volume de água no reservatório [m3])
12 //P (pressão relativa à atmosférica, no fundo do reservatório [Pa])
13 //Qs (vazão de saída [m3/s])
14 //Admite-se que a água seja incompressível.
15

```

```
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 // Termina 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:

```

1 // Apagando dados anteriores:
2 clear
3 S = 10 //m2 – área da seção transversal (constante)
4 R = 2e8 //Pa/(m3/s)2 – par metro que relaciona vazão com queda de pressão
   (perda de carga)
5 p = 1e3 //kg/m3 – massa específica da água
6 g = 10 //m/s2 – aceleração da gravidade na superfície da terra
7 Qe = 0.010247 //m3/s – vazão de entrada
8
9 //Achar
10 //h (nível do reservatório [m])

```

```

11 //V (volume de água no reservatório [m3])
12 //P (pressão relativa à atmosférica, no fundo do reservatório [Pa])
13 //Qs (vazão de saída [m3/s])
14 //Admite-se que a água seja incompressível.
15
16 // Instante inicial:
17 t(1)=0;
18 // Instante final:
19 tf=10;
20 // Condicao inicial:
21 y(1)=0;
22 // Valor inicial da solucao exata:
23 ye(1)=0;
24 // Passo de integracao (experimente alterar o passo):
25 h=0.1;
26 // Calculo de numero de passos:
27 n=round(tf/h);
28
29 // Integracao numerica usando o metodo de Runge Kutta:
30 // Comando for:
31 for i=1:n
32 // Vetor de tempo:
33 t(i+1)=t(i)+h;
34 // Solucao numerica:
35 k1=(-sqrt(p*g*y(i)/R)+Qe)/S
36 k2=(-sqrt(p*g*(y(i)+h*k1/2)/R)+Qe)/S
37 k3=(-sqrt(p*g*(y(i)+h*k2/2)/R)+Qe)/S
38 k4=(-sqrt(p*g*(y(i)+h*k3)/R)+Qe)/S
39
40 //k1=h*(1-(y(i)))/2;
41 //k2=h*(1-(y(i)+k1/2))/2;
42 //k3=h*(1-(y(i)+k2/2))/2;
43 //k4=h*(1-(y(i)+k3))/2;
44
45 y(i+1)=y(i)+((k1+2*k2+2*k3+k4)*h/6);
46
47 // Termina do comando for:
48 end
49
50 V = S*y
51 P = p*g*y
52 Qs = sqrt(P/R)
53
54 // Plotando solucao numerica y versus vetor de tempo t:
55 f(1)=scf(1);
56 plot2d([t],[y]);
57 xtitle("Nível h do reservatório","Tempo t","Solucao")

```

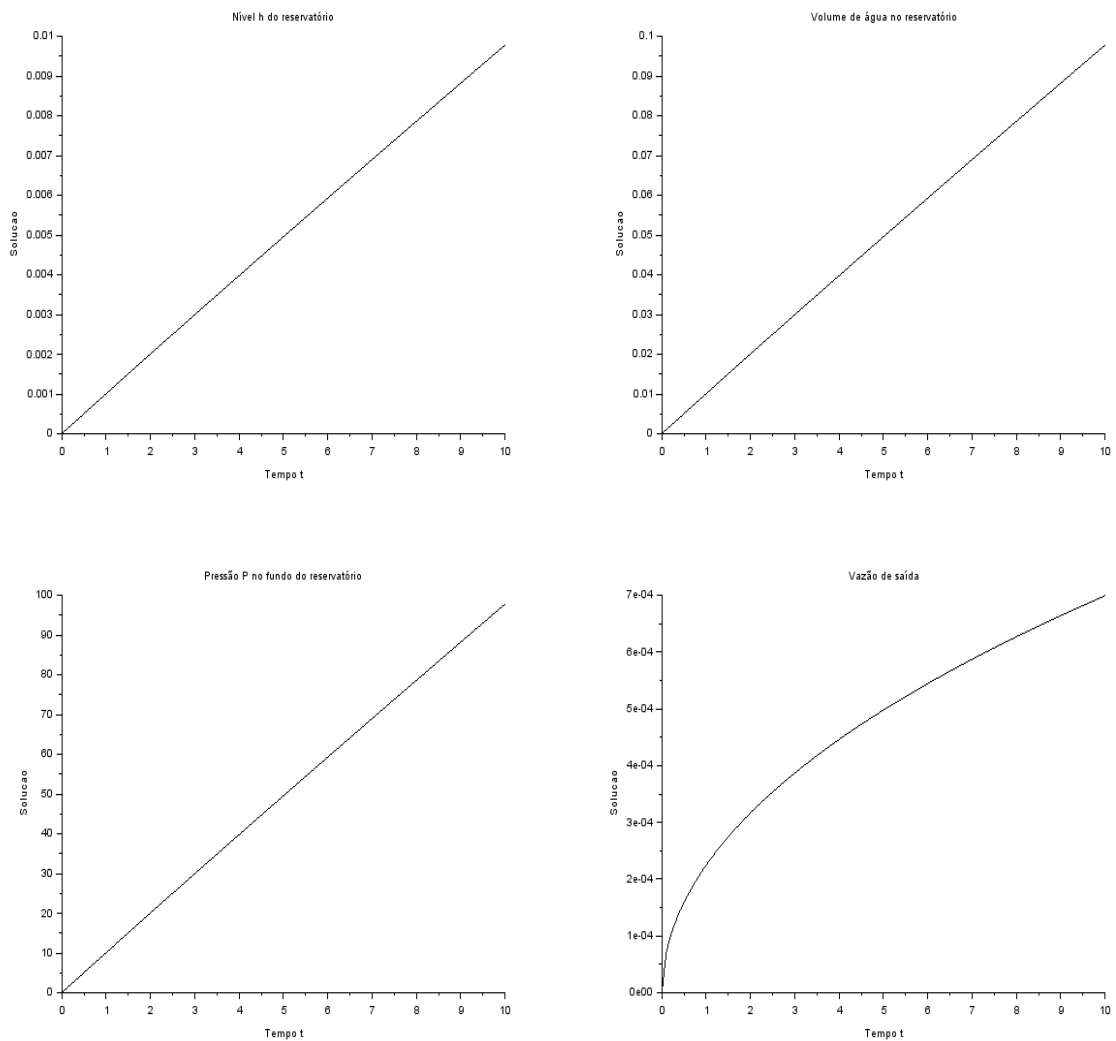
```

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

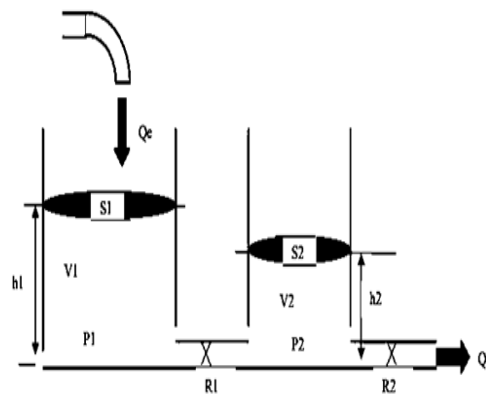


Observa-se que os resultados são muito semelhantes.

## 2.2 Dois Reservatórios

Figura 10 – Problema com 2 Reservatórios

Desenvolva um programa em Scilab que resolva numericamente o sistema de equações diferenciais que modela o sistema com dois reservatórios, usando tanto Euler como Runge Kutta. Dica: raciocine com vetores.



Implementando o seguinte código para Resolver por Euler:

```

1 // Apagando dados anteriores:
2 clear
3 S1 = 10 //m2 - área da seção transversal (constante)
4 S2 = 2 //m2 - área da seção transversal (constante)
5 R1 = 2e8 //Pa/(m3/s)2 - par metro que relaciona vazão com queda de pressão
   (perda de carga)
6 R2 = 1.6e8 //Pa/(m3/s)2 - par metro que relaciona vazão com queda de press
   ão (perda de carga)
7 p = 1e3 //kg/m3 - massa específica da água
8 g = 10 //m/s2 - aceleração da gravidade na superfície da terra
9 Qe = 0.010247 //m3/s - vazão de entrada
10
11 //Achar
12 //h (nível do reservatório [m])
13 //V (volume de água no reservatório [m3])
14 //P (pressão relativa à atmosférica, no fundo do reservatório [Pa])
15 //Qs (vazão de saída [m3/s])
16 //Admite-se que a água seja incompressível.
17
18 function hdot1=hp1(h1, h2)
19 hdot1=(Qe-sqrt(p*g*(h1-h2)/R1))/S1;
20 endfunction
21
22 function hdot2=hp2(h1, h2)
23 hdot2=(sqrt(p*g*(h1-h2)/R1)-sqrt(p*g*h2/R2))/S2;
24 endfunction
25

```

```

26 // Instante inicial:
27 t(1)=0;
28 // Instante final:
29 tf=10;
30 // Condições iniciais:
31 h1(1)=0;
32 h2(1)=0;
33 // Passo de integração (experimente alterar o passo):
34 h=0.1;
35 // Cálculo de número de passos:
36 n=round(tf/h);
37
38 // Integração numérica usando o método de Euler:
39 // Comando for:
40 for i=1:n
41 // Vetor de tempo:
42 t(i+1)=t(i)+h;
43 // Solução numérica:
44 h1(i+1)=h1(i)+h*hp1(h1(i),h2(i));
45 h2(i+1)=h2(i)+h*hp2(h1(i+1),h2(i)); //colocando h1(i+1) para usar um dado
    mais atualizado
46 // Término do comando for:
47 end
48
49 V1 = S1*h1;
50 P1 = p*g*h1;
51 Qs1 = sqrt(P1/R1);
52
53 V2 = S2*h2;
54 P2 = p*g*h2;
55 Qs2 = sqrt(P2/R2);
56
57 // Plotando solução numérica y versus vetor de tempo t:
58 f(1)=scf(1);
59 plot2d([t,t],[h1,h2],[1 2]);
60 legends(["Altura h1","Altura h2"],[1,2],2)
61 xtitle("Comparação das alturas dos reservatórios","Tempo t","Solução")
62 f(2)=scf(2);
63 plot2d([t,t],[V1,V2],[1 2]);
64 xtitle("Comparação dos Volumes dos reservatórios","Tempo t","Solução")
65 f(3)=scf(3);
66 plot2d([t,t],[P1,P2],[1 2]);
67 xtitle("Comparação das pressões dos reservatórios","Tempo t","Solução")
68 f(4)=scf(4);
69 plot2d([t,t],[Qs1,Qs2],[1 2]);
70 xtitle("Comparação das vazões dos reservatórios","Tempo t","Solução")

```



Obtemos os seguintes gráficos:

Figura 11 – Resultados do problema 2 por Euler

