Lista B - Aulas Práticas de Scilab

Cássio Murakami – 10773798

O seguinte código foi elaborado a fim de resolver numericamente as equações diferenciais que regem o comportamento da altura do reservatório de água:

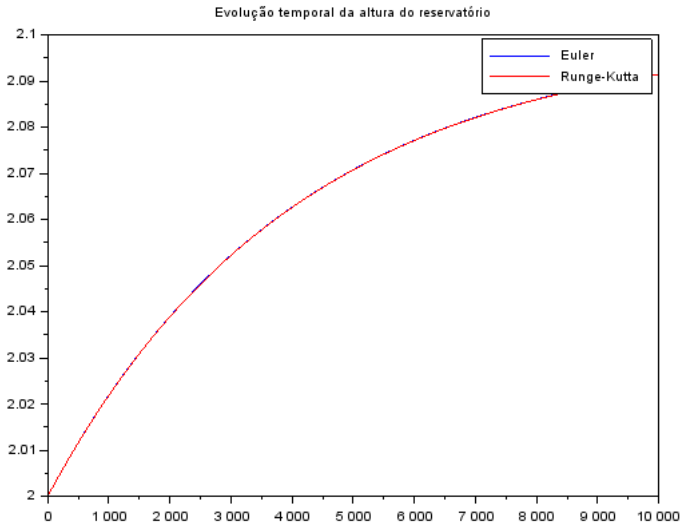
```
clear();
xdel(winsid());

//Definição das constantes :
S = 10;
R = 2*10^8;
rho = 1000;
g = 10;
Qe = 0.010247;

function fun=funcao(h)
   fun = (-sqrt(rho*g*h/R)+Qe)*(1/S)
endfunction

// Instante inicial:
t(1)=0;
// Instante final:
tf=10000;

// Condicao inicial da altura do reservatório:
y(1)=2;
y_e(1) = y(1);
y_rk(1) = y(1);
// Passo de integracao (experimente alterar o passo):
h=10;
// Calculo de numero de passos:
n=round((tf-t(1))/h);

// Integracao numerica usando o metodo de Euler:
for i=1:n
   // Vetor de tempo:
   t(i+1)=t(i)+ h;
   // Solucao numerica:
   y_e(i+1)=y_e(i)+h*funcao(y_e(i));
end

// Integracao numerica usando o metodo de Runge Kutta:
for i=1:n
// Vetor de tempo:
   t(i+1)=t(i)+ h;
// Solucao numerica:
   k1=funcao(y_rk(i))
```

```
   k2=funcao(y_rk(i) + (h/2)*k1)
   k3=funcao(y_rk(i) + (h/2)*k2)
   k4=funcao(y_rk(i) + h*k3)
   y_rk(i+1)=y_rk(i) + h*((k1+2*k2+2*k3+k4)/6);
end

scf(1)
xtitle("Evolução temporal da altura do reservatório")
plot(t,y_e);
plot(t,y_rk,'r');
legend(['Euler';'Runge-Kutta']);

//----------------------------Problema 2 : Dois reservatórios---------------------------------//

function fun=y1(h1, h2)
   fun = (Qe - sqrt(rho*g*(h1-h2)/R))*(1/S)
endfunction

function fun=y2(h1, h2)
   fun =(sqrt(rho*g*(h1-h2)/R) - sqrt(rho*g*h2/R))*(1/S)
endfunction

// Instante inicial:
t(1)=0;
// Instante final:
tf=10000;

// Condicao inicial da altura do reservatório:
y1_inicial = 10;
y2_inicial = 3;

y1_e(1) = y1_inicial;
y2_e(1) = y2_inicial;
y1_rk(1) = y1_inicial;
y2_rk(1) = y2_inicial;

// Passo de integracao (experimente alterar o passo):
h=10;
// Calculo de numero de passos:
n = round((tf-t(1))/h);

//Método de Euler:
for i = 1:n
   t(i+1) = t(i) + h
   y1_e(i+1) = y1_e(i) + h*y1(y1_e(i),y2_e(i));
   y2_e(i+1) = y2_e(i) + h*y2(y1_e(i),y2_e(i));
end

//Método de Runge-Kutta:
for i = 1:n
   k11 = y1(y1_rk(i),y2_rk(i));
```

```
    k21 = y2(y1_rk(i),y2_rk(i));

    k12 = y1(y1_rk(i) + (h/2)*k11 , y2_rk(i) + (h/2)*k21)
    k22 = y2(y1_rk(i) + (h/2)*k11 , y2_rk(i) + (h/2)*k21)

    k13 = y1(y1_rk(i) + (h/2)*k12 , y2_rk(i) + (h/2)*k22)
    k23 = y2(y1_rk(i) + (h/2)*k12 , y2_rk(i) + (h/2)*k22)

    k14 = y1(y1_rk(i) + h*k13 , y2_rk(i) + h*k23)
    k24 = y2(y1_rk(i) + h*k13 , y2_rk(i) + h*k23)

    y1_rk(i+1) = y1_rk(i) + (h/6)*(k11 + 2*k12 + 2*k13 + k14)
    y2_rk(i+1) = y2_rk(i) + (h/6)*(k21 + 2*k22 + 2*k23 + k24)
end

scf(2)
xtitle("Altura do reservatório 1")
plot(t,y1_e)
plot(t,y1_rk,'r')
legend(['Euler';'Runge-Kutta']);

scf(3)
xtitle("Altura do reservatório 2")
plot(t,y2_e)
plot(t,y2_rk,'r')
legend(['Euler';'Runge-Kutta']);
```

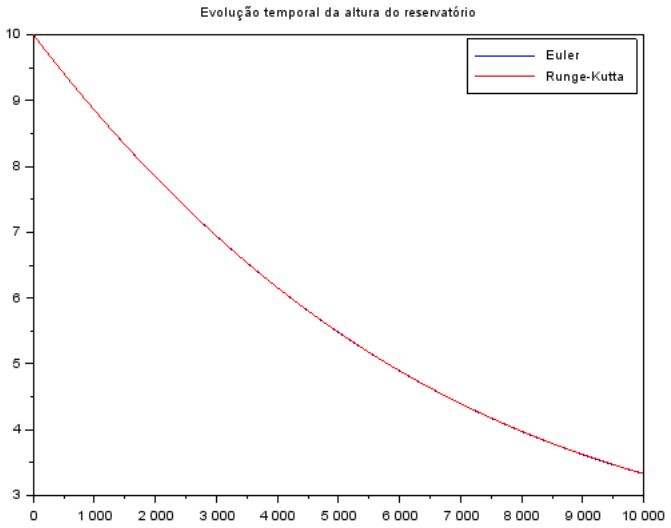
Exercício 1:

Como parâmetro inicial para o problema de um único reservatório, foi utilizado h(0) = 2 m. Além disso, o intervalo temporal escolhido foi de 10000 s com um passo de 10 s. A seguir será mostrado o gráfico obtido tanto da solução pelo método de Euler, quanto pelo método de Runge-Kutta:



O valor crescente da altura da água no reservatório é reflexo da vazão e água da torneira que é elevado em comparação aos demais parâmetros que definem o escoamento.

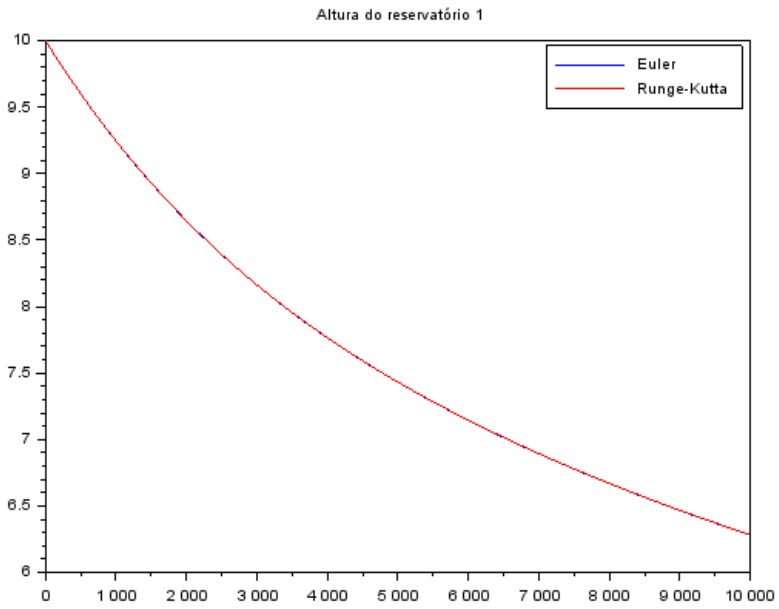
Outro possível caso de teste é com h(0) = 10 m :

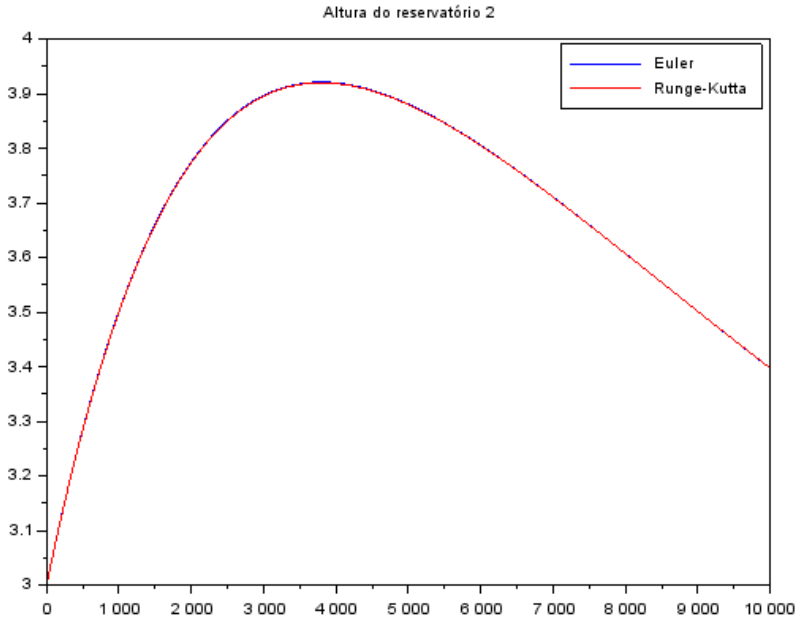


Note que nesse caso a função é decrescente, resultado da taxa de água que sai do sistema ser maior que a da torneira.

Exercício 2:

O intervalo de tempo considerado para a solução foi de 10000 s com um passo de 10 s. As alturas iniciais consideradas foram h1(0) = 10 m e h2(0) = 3 m. Os resultados das soluções numéricas obtidas são exprimidos nos seguintes gráficos:




Altura do reservatório 2
Euler
Runge-Kutta
4
3.9
3.8
3.7
3.6
3.5
3.4
3.3
3.2
3.1
3
0
1 000
2 000
3 000
4 000
5 000
6 000
7 000
8 000
9 000
10 000