Resolvendo EDO's no MATLAB

Prof. Doherty Andrade -- www.metodosnumericos.com.br

1. **Determinando soluções explícitas**

MatLab tem uma grande biblioteca de funções para resolver equações diferenciais ordinárias. Neste **Live Script**, vamos considerar apenas as mais básicas.

Embora MatLab seja primeiramente um pacote numérico ele pode também resolver simbolicamente EDO's para isto usamos o comando *dsolve( ).*

Consideremos como exemplo, a EDO dada por $y'(x) = -xy$.

```
dsolve('Dy=-y*x','x')
```

```
ans =
```

$$C_4\, e^{-\frac{x^2}{2}}$$

Note que MatLab usa a letra D maiúscula para indicar derivada mas também pode ser diff.

Podemos também entrar com a equação e os dados inciais. Veja o exemplo

```
eq1 = 'Dy = -y*x'
```

```
eq1 =
'Dy = -y*x'
```

```
inic = 'y(1)=1'
```

```
inic =
'y(1)=1'
```

```
dsolve(eq1,inic,'x')
```

```
ans =
```

$$\sqrt{e}\, e^{-\frac{x^2}{2}}$$

Para salvar a solução simbólica, devemos criar uma função simbólica *y(x)*

```
syms y(x)
```

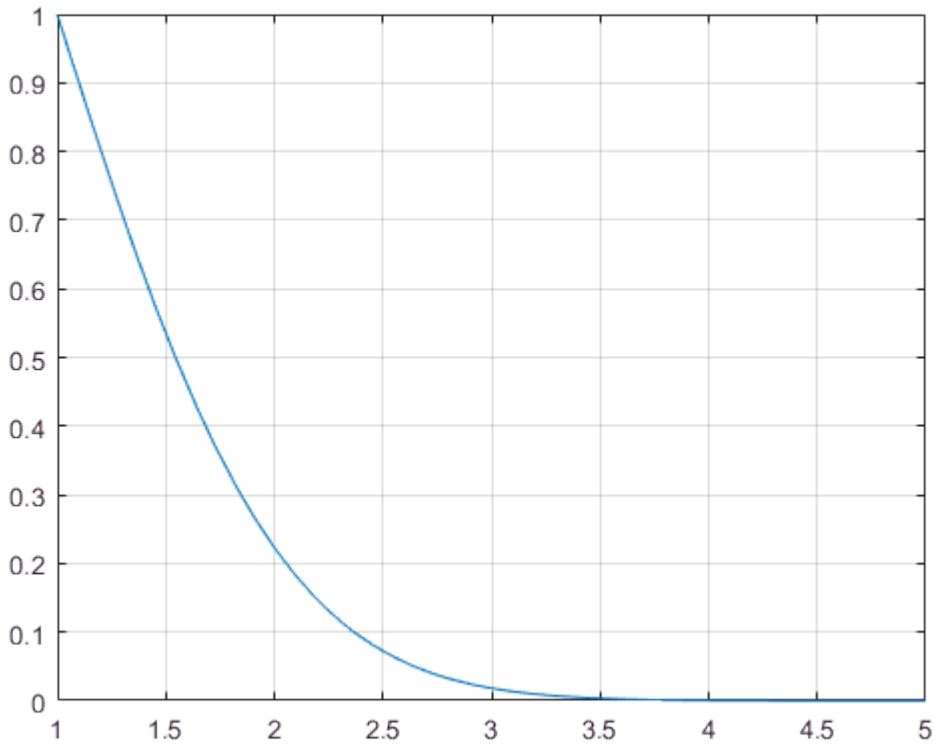
```
ysol(x) = dsolve(eq1,inic,'x')
```

```
ysol(x) =
```

$$\sqrt{e}\, e^{-\frac{x^2}{2}}$$

Uma vez que já temos a solução de uma EDO podemos querer plotar o gráfico da solução para termos uma ideia do seu comportamento.

```
fplot(@(x)[ysol(x)],[1,5]), grid
```



Se o comando dsolve não pode resolver sua EDO tente os métodos numéricos para EDO's do MatLab.

Vejamos um exemplo de EDO não linear.

$$\left(\frac{dy}{dx} + y\right)^2$$
$$y(0) = 0.$$

```
syms y(x)
ode = (diff(y,x)+y)^2 == 1;
cond = y(0) == 0;
ysol(x) = dsolve(ode,cond)
```

```
ysol(x) =
```

$$\begin{pmatrix} e^{-x} - 1 \\ 1 - e^{-x} \end{pmatrix}$$

Agora um exemplo de EDO de **segunda ordem**:

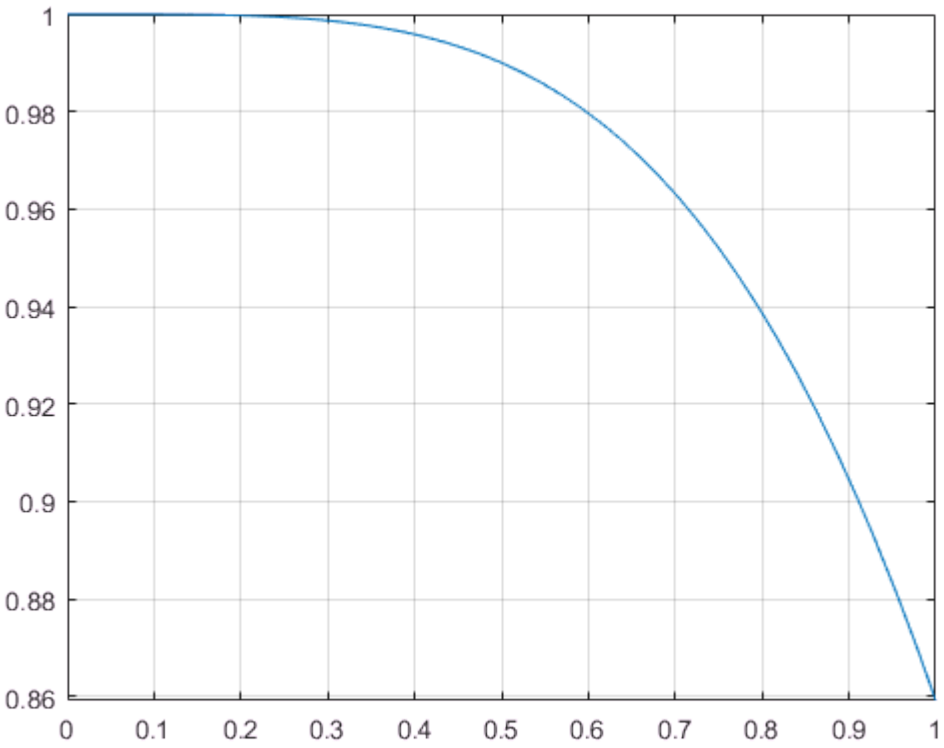
$$\frac{d^2y}{dx^2} = \cos(2x) - y$$
$$y(0) = 1$$
$$y'(0) = 0$$

```
syms y(x)
Dy = diff(y);
ode = diff(y,x,2) == cos(2*x)-y;
cond1 = y(0) == 1;
cond2 = Dy(0) == 0;
conds = [cond1 cond2];
ySol(x) = dsolve(ode,conds);
ySol = simplify(ySol)
```

```
ySol(x) =
```

$$1 - \frac{8\sin\left(\frac{x}{2}\right)^4}{3}$$

```
fplot(@(x)[ySol(x)],[0,1]), grid
```



Outro Exemplo de EDo de segunda ordem

Suponha que desejamos resolver a EDO de segunda ordem

$$y''(x) + 8y'(x) + 2y(x) = \cos(x)$$
$$y(0) = 0$$
$$y'(0) = 1$$

Vejamos como fazer isso no MatLab.

```
eq2 = 'D2y+8*Dy+2*y= cos(x)';
inics = 'y(0) =0, Dy(0)= 1';
y = dsolve(eq2,inics,'x')
```

```
y =
```

$$\frac{\sqrt{14}\ e^{4x-\sqrt{14}x}\sigma_2\ \left(\sin(x)-\cos(x)\ \left(\sqrt{14}-4\right)\right)}{28\ \left(\left(\sqrt{14}-4\right)^2+1\right)}-\frac{\sqrt{14}\ e^{4x+\sqrt{14}x}\sigma_1\ \left(\sin(x)+\cos(x)\ \left(\sqrt{14}+4\right)\right)}{28\ \left(\left(\sqrt{14}+4\right)^2+1\right)}-\frac{\sqrt{1}}{}$$

where

$$\sigma_1 = e^{-x\ (\sqrt{14}+4)}$$

$$\sigma_2 = e^{x\ (\sqrt{14}-4)}$$

$$\sigma_3 = 8\ \sqrt{14}+31$$

ou podemos definir uma função simbolica usando syms como anteriormente.

```
syms y(x)
Dy = diff(y);
ode2 = diff(y,x,2) +8*diff(y,x)+2*y== cos(x);
cond1 = y(0) == 0;
cond2 = Dy(0) == 1;
conds = [cond1 cond2];
ySol2(x) = dsolve(ode2,conds);
ySol2 = simplify(ySol2)
```

```
ySol2(x) =
```

$$\frac{\sqrt{65}\ \sin\left(x+\operatorname{atan}\left(\frac{1}{8}\right)\right)}{65}-\frac{e^{-x\ (\sqrt{14}+4)}}{130}-\frac{e^{x\ (\sqrt{14}-4)}}{130}+\frac{53\ \sqrt{14}\ e^{x\ (\sqrt{14}-4)}}{1820}-\frac{53\ \sqrt{14}\ e^{-x\ (\sqrt{14}+4)}}{1820}$$

3. **Sistema de EDOs**

É possível determinar explicitamente solução de um sistema de EDO's. Veja o exemplo.

Considere o sistema:

$$\frac{dy}{dx} = z$$
$$\frac{dz}{dx} = -y$$

```
syms y(t) z(t)
eqns = [diff(y,t)==z, diff(z,t)==-y];
[ySol(t),zSol(t)] = dsolve(eqns)
```

```
ySol(t) = C13*cos(t) + C12*sin(t)
zSol(t) = C12*cos(t) - C13*sin(t)
```

4. **Solução numérica**

MatLab possui vários métodos numéricos (chamados de solver) que permitem obter numericamente a solução de EDOs, método de Euler, métodos de Runge-Kutta e etc.

Vamos tomar como exemplo a EDO

$$y' = -2xy + 1$$
$$y(0) = 1$$

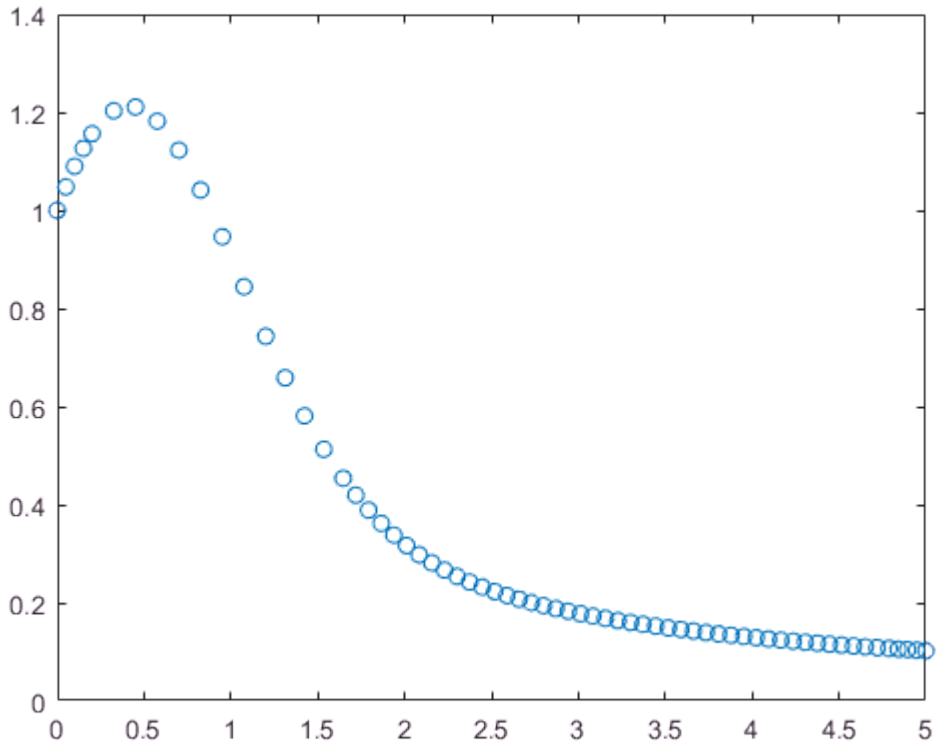
e vamos usar o método ***ode45*** (método de Runge -Kutta de ordem 4 melhorado) para determinar a solução numérica no intervalo [0,5].

```
xspan = [0 5];
y0 = 1;
[x,y] = ode45(@(x,y)-2*x*y+1,xspan, y0)
```

```
x =
         0
    0.0502
    0.1005
    0.1507
    0.2010
    0.3260
    0.4510
    0.5760
    0.7010
    0.8260
     ⋮
```

```
y =
    1.0000
    1.0476
    1.0898
    1.1260
    1.1561
    1.2030
    1.2105
    1.1817
    1.1226
    1.0413
     ⋮
```

```
plot(x,y,'o ')
```

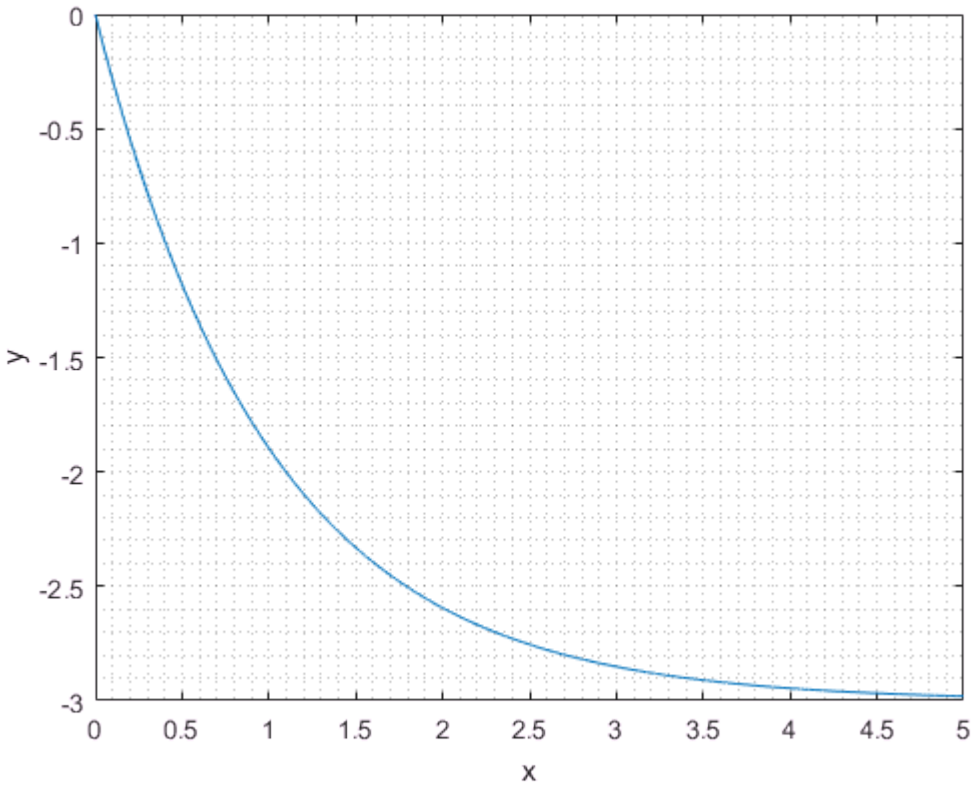


Vejamos outro exemplo. Resolver numericamente a EDO

$$y' = -3\exp(-x)$$
$$y(0) = 0$$

```
x=0:0.001:5;
inicial_y = 0;
[x,y]=ode45( @(x,y)-3*exp(-x), x, inicial_y);

plot(x,y);
xlabel('x'); ylabel('y');grid minor;
```



4. Problemas de valor de Fronteira (PVF)

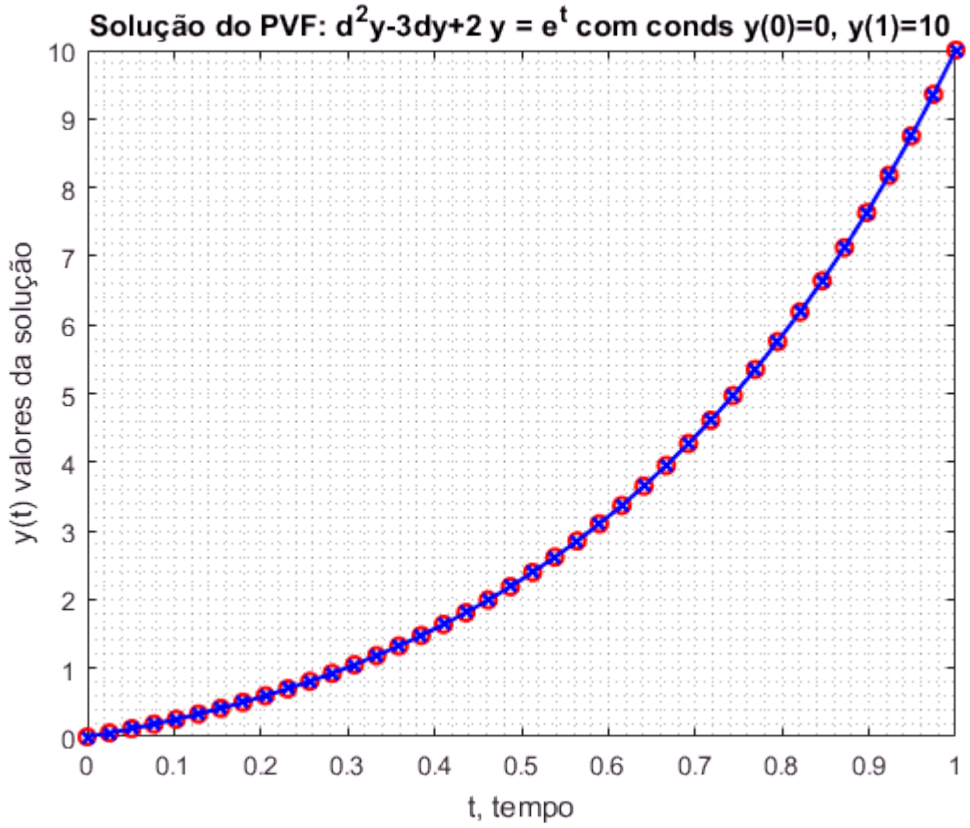
Baseado no trabalho de Sulaymon L. ESHKABILOV.

Exemplo:

$$
\begin{aligned}
&y'' - 3y' + 2y = 0 \\
&y(0) = 0 \\
&y(1) = 10
\end{aligned}
$$

```
% Defina a função inline dy:
dy=inline('[y(2); -2*y(1)+3*y(2)]', 't', 'y');
% Defina as condições de bordo via uma função inline:
Res=inline('[yl(1), yr(1)-10]', 'yl', 'yr');
% Criar a discretização do espaço temo para a solução problema:
t = linspace(0, 1, 40);
% Cria uma estrutura consistindo de uma malha inicial
% de 40 pontos igualmente espaços em  [0,1] e
% e valores constantes como candidato de  y1=0  e y2=1 com este comando:
SOLin1 = bvpinit(t,[0 1]);
% Outro valor constante como candidato: y1=2 e y2=10
SOLin2 = bvpinit(t,[2 10]);
% Obter a solução numericia do problema:
SOLs1GUESS = bvp4c(dy,Res,SOLin1);
```

```
SOLs2GUESS = bvp4c(dy,Res,SOLin2);
y1 = deval(SOLs1GUESS,t);
y2 = deval(SOLs2GUESS,t);
figure
plot(t, y1(1,1:end), 'ro-', t, y2(1, :),'bx-', 'linewidth', 1.5),
grid minor ,title('Solução do PVF: d^2y-3dy+2 y = e^t com conds y(0)=0, y(1)=10')
xlabel( 't, tempo'), ylabel( 'y(t) valores da solução')
```




```
```

Para ver os valores calculados da solução escrevea y2.

```
y2
```

```
y2 =
         0    0.0571    0.1186    0.1849    0.2562    0.3329    0.4153    0.5037 ···
    2.1410    2.3107    2.4908    2.6819    2.8847    3.0997    3.3276    3.5693
```

-

```
```

Exemplo 2

Suponha que desejamos determinar a solução do seguinte PVF:

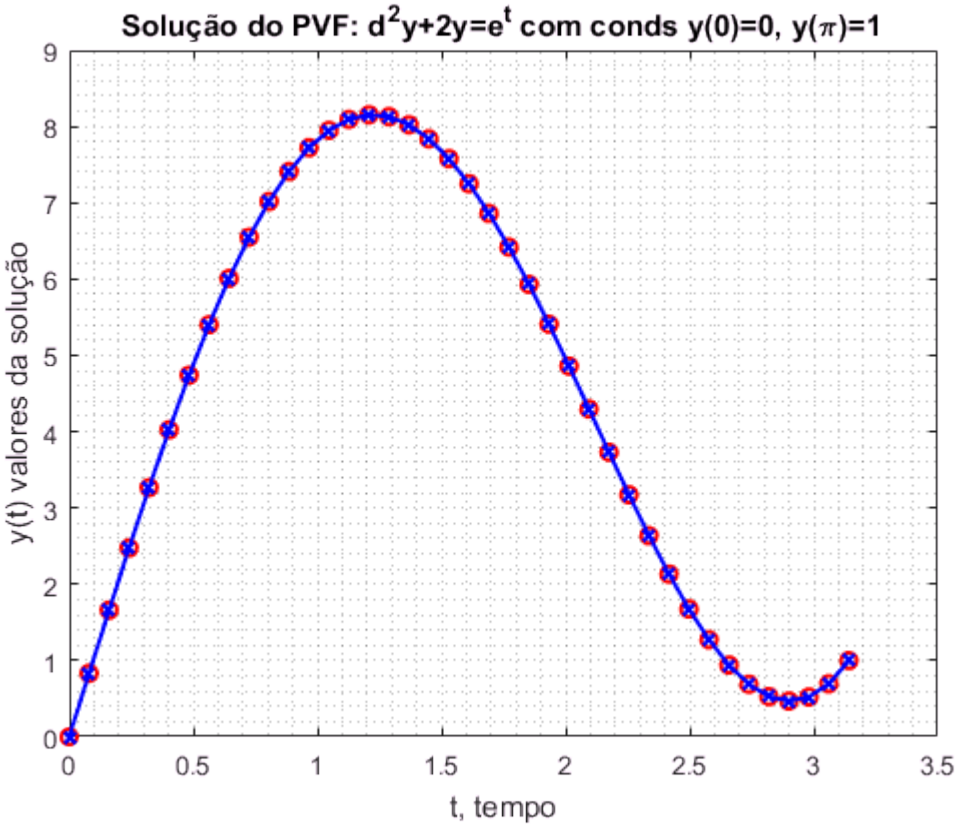
$$y'' + 2y = \exp(t)$$
$$y(0) = 0$$
$$y(\pi) = 1$$

Precisamos obrigatoriamente passar para um sistema de EDOs de primeira ordem. Faça $y_1 = y$ e portanto:

$$y_1' = y_2$$
$$y_2' = \exp(t) - 2y_1$$

Vamos entrar com os dados.

```
% Defina a função inline dy:
dy=inline('[y(2); exp(t)-2*y(1)]', 't', 'y');
% Defina as condições de bordo via uma função inline:
Res=inline('[yl(1), yr(1)-1]', 'yl', 'yr');
% Criar a discretização do espaço temo para a solução problema:
t = linspace(0, pi, 40);
% Cria uma estrutura consistindo de uma malha inicial
% de 40 pontos igualmente espaços em  [0,pi] e
% e valores constantes candidatos de  y1=0  e y2=1 com este comando:
SOLin1 = bvpinit(t,[0 1]);
% Outro valor constante candidato: y1=2 e y2=1
SOLin2 = bvpinit(t,[2 1]);
% Obter a solução numericia do problema:
SOLs1GUESS = bvp4c(dy,Res,SOLin1);
SOLs2GUESS = bvp4c(dy,Res,SOLin2);
y1 = deval(SOLs1GUESS,t);
y2 = deval(SOLs2GUESS,t);
figure
plot(t, y1(1,1:end), 'ro-', t, y2(1, :),'bx-', 'linewidth', 1.5),
grid minor, title('Solução do PVF: d^2y+2y=e^t com conds y(0)=0, y(\pi)=1')
xlabel( 't, tempo'), ylabel( 'y(t) valores da solução')
```


Solução do PVF: d^2y+2y=e^t com conds y(0)=0, y(π)=1
y(t) valores da solução
t, tempo
0
1
2
3
4
5
6
7
8
9
0
0.5
1
1.5
2
2.5
3
3.5