Main

```
%Параметры задачи
global a b k0 k1 n num h
k0 = 1;
k1 = 0.5;
n = 8;
a = 0;
b = 1;
num = 7;
h = (b - a)/n;
eps = 10^-12;

%Собственно рассчет (рншение + собственные значения)
u0 = zeros(1,n+2);
[x,u,lambda] = Newton(u0, eps);

figure(1)
hold on; grid on
title('Solution')
plot(x,u(1:end-1),'m')

figure(2)
hold on; grid on
title('\lambda')
plot(1:length(lambda), lambda,'.-b')

%Эффективный порядок метода
p = zeros(length(lambda)-2, 1);
for i = 1:length(lambda)-2
    p(i) = -log2(abs((lambda(i+2) - lambda(i+1))/(lambda(i+1) - lambda(i))));
end

figure(3)
hold on; grid on
title('Degree')
plot(p,'.-c')

% уточнение по Ричадрсону
RichardsonRefinement(lambda)
```

Newton.m

```
function [x,u,lambda] = Newton(u0, eps)
global a b k0 k1 n num h

%Начальные значения для решения
```

```
lambda0 = 40;
du = ones(1,n+2);
u0(end) = lambda0;
u0(end/2+1) = h;
u0(end/2-1) = -h;
x = a:h:b;
u = u0;
lambda = zeros(num + 1, 1);
lambda(1) = lambda0;

for j = 1 : num
    while(max(abs(du)) > eps)

        %Формирование новой матрицы Якоби
        dF = zeros(n+2);

        dF = dF + diag([(k0 + k1*(u(2:end-2).^2 + u(1:end-3).^2)/2) - (u(2:end-2) -
u(1:end-3)).*(k1*u(1:end-3)) 0 0], -1) + diag([1 (-(k0 + k1*(u(2:end-2).^2 + u(3:end-
1).^2)/2) + (u(3:end-1) - u(2:end-2)).*(k1*u(2:end-2))-(k0 + k1*(u(2:end-2).^2 +
u(1:end-3).^2)/2) - (u(2:end-2) - u(1:end-3)).*(k1*u(2:end-2)) + h^2*u(end)) 1 0], 0) +
diag([0 (k0 + k1*(u(2:end-2).^2 + u(3:end-1).^2)/2) + (u(3:end-1) - u(2:end-2)).*
(k1*u(3:end-1)) 0], 1);
        dF(2:end-2, end) = h^2*u(2:end-2);
        dF(end, end/2-1) = -1;
        dF(end, end/2+1) = 1;

        %Метод Ньютона
        F = zeros(n+2, 1);
        F(1) = u(1);    %ГУ
        F(2:end-2) = (u(3:end-1) - u(2:end-2)).*(k0 + k1*(u(3:end-1).^2 + u(2:end-
2).^2)/2)-(u(2:end-2) - u(1:end-3)).*(k0 + k1*(u(1:end-3).^2 + u(2:end-2).^2)/2)+
h^2*u(2:end-2)*u(end);
        F(end-1) = u(end-1); %ГУ
        F(end) = u(end/2+1)-u(end/2-1)-2*h; %ГУ на лямбду
        F = -F;

        G = dF\F;
        du = G(:, end);
        u = u + du';
    end

    lambda(j+1) = u(end);

    %Пересчет значений для вычисления слудующего приближения
    u0 = u;
    n = 2*n;
    h = (b-a)/n;
    x = a:h:b;
    u = zeros(1,n+2);
```

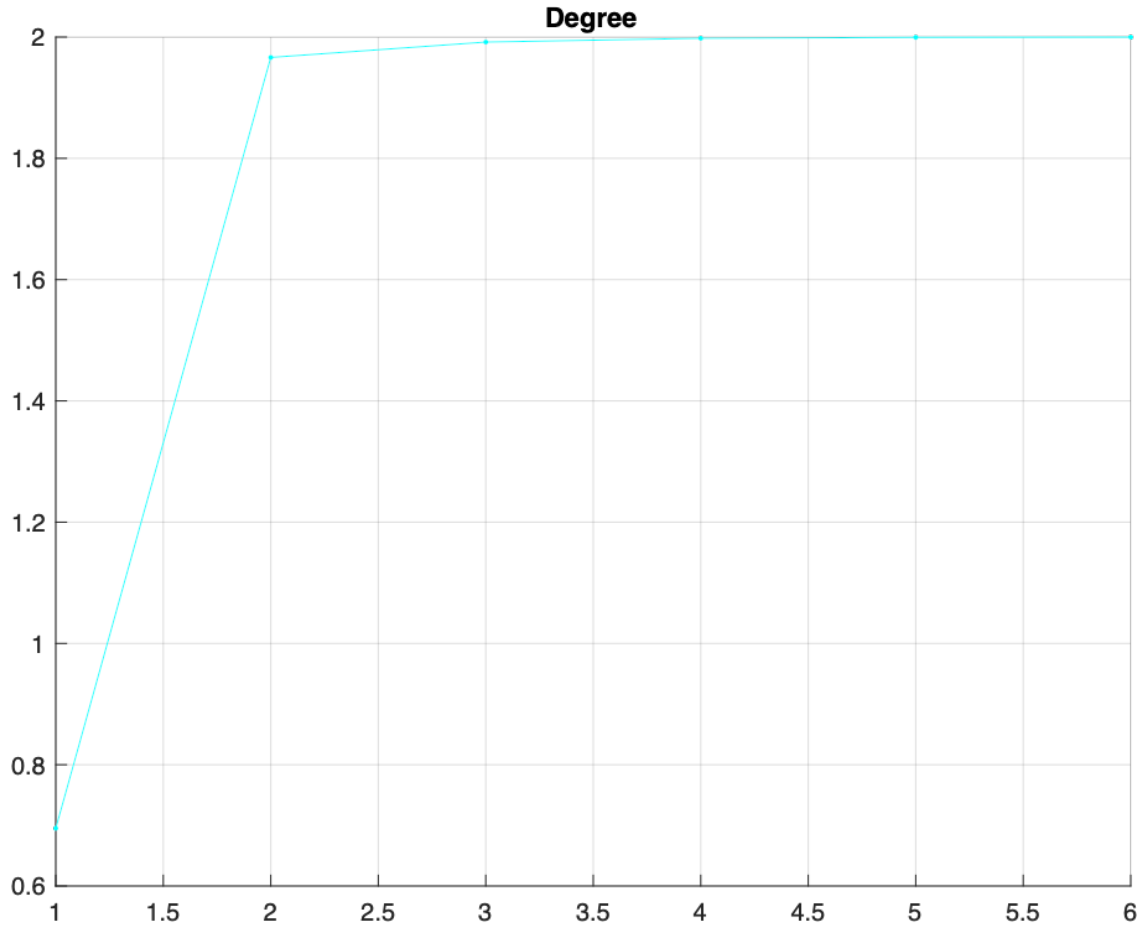
```
    du = ones(1,n+2);
    u(1:2:end-1) = u0(1:end-1);
    u(end) = u0(end);
    u(2:2:end-2) = (u0(1:end-2) + u0(2:end-1))/2;
end
end
```

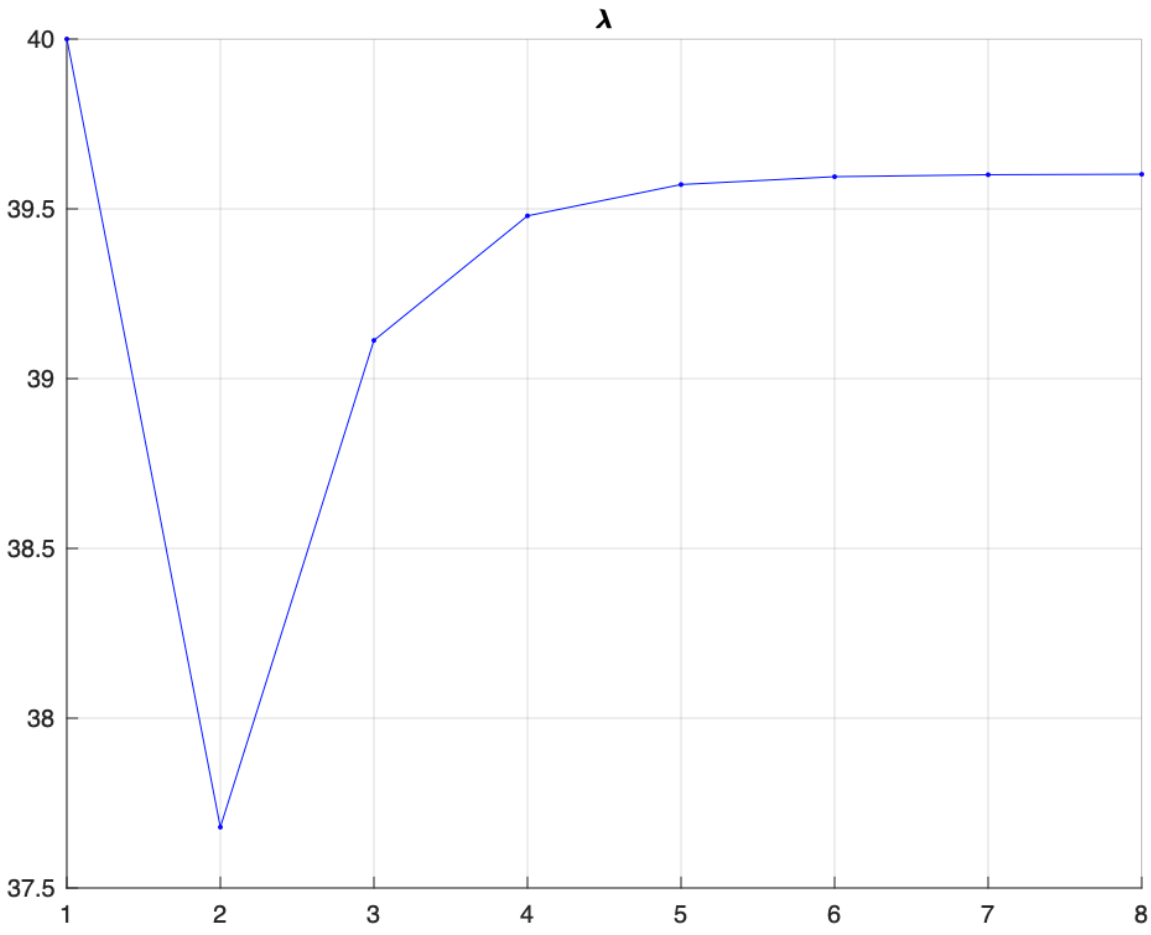
RichardsonRefinement.m

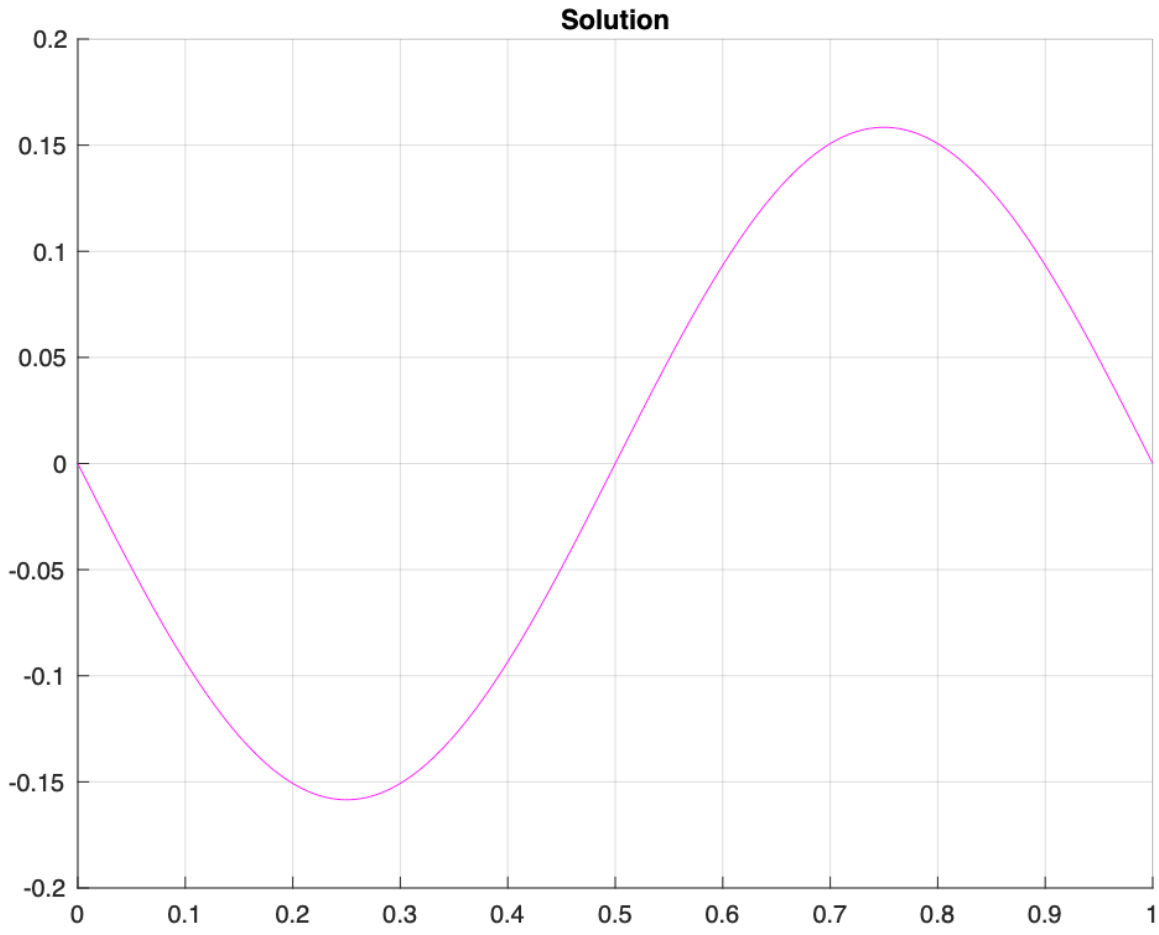
```
function lambda_out = RichardsonRefinement(lambda)
global num

lambda_pr = lambda(2:end); %промежуточное лямбда

for j = 1:floor(num-2)
    p = zeros(length(lambda_pr)-2, 1);

    for i = 1:length(lambda_pr)-2
        p(i) = -log2(abs((lambda_pr(i+2) - lambda_pr(i+1))/(lambda_pr(i+1) -
lambda_pr(i))));
    end

    R = zeros(length(lambda_pr)-1, 1);

    for i = 1:length(R)
        R(i) = (lambda_pr(i+1) - lambda_pr(i)) / (2^round(p(end)) - 1);
    end
    lambda_pr = lambda_pr(2:end) + R;
end
lambda_out = (2^round(p(end))*lambda_pr(2) - lambda_pr(1)) / (2^round(p(end)) - 1);
end
```

Result

```
ans = 39.602394042760189
```


Degree
2
1.8
1.6
1.4
1.2
1
0.8
0.6
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
6


λ
40
39.5
39
38.5
38
37.5
1
2
3
4
5
6
7
8


Solution
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1