Решение нелинейных уравнений и систем нелинейных уравнений второго порядка

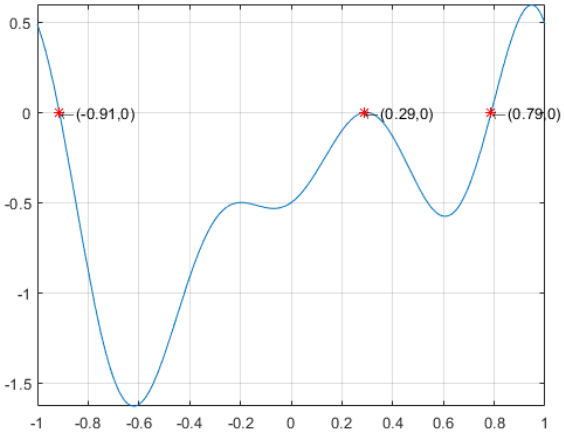
I) Функция fsolve

1й способ . Вместо решения системы двух нелинейных уравнений y=f(x) и y=g(x) будем искать корни уравнения F(x)= f(x)–g(x)=0 с помощью функции ***fsolve:*** froots=fsolve(func,x0):

1. строим график F(x)
2. Определяем визуально нулевое приближение x0. Заметим, что x0 – может быть скаляром или вектором, который аккумулирует все нулевые приближения в случае неединственного корня.
3. Ищем корни, используя fsolve
   Заметим, что функция `num2str(x,k),` оставляет в строке k цифр после десятичной точки

Пример 1.

```
Поиск корней: случай задания вектора нулевых приближений
требует векторизации F(x)=f(x) - g(x):
figure
fplot('x*sin(8*x)-(x^5-x+0.5)',[-1 1]), hold on, grid on; %
ищем x0
x0=fsolve('x.*sin(8*x)-(x.^5-x+0.5)',[-0.9 0.2 0.8])
y=0;   % пересечение с осью OX, y=0
for x=x0          % здесь вектор решений
    plot(x,y,'r*')
    xy=['(',num2str(x,2),',',num2str(y,2),')']
    text(x,y,['\leftarrow' xy])
end
```

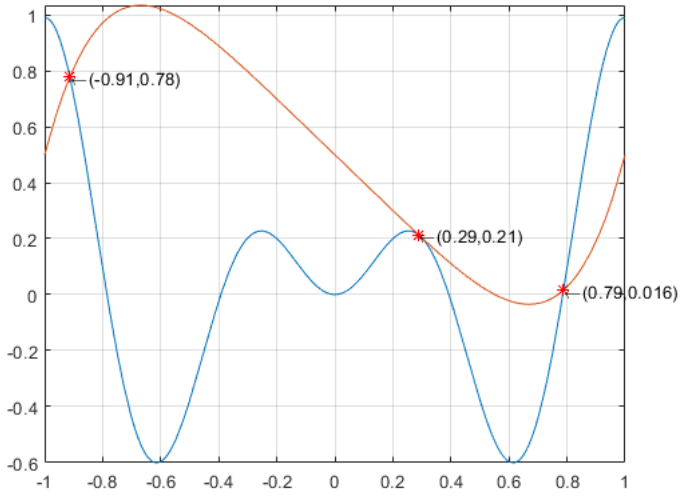


2й способ. Решения системы двух нелинейных уравнений – точки пересечения кривых y=f(x) и y=g(x) строим график F(x)

1. Строим графики.
2. Определяем визуально нулевое приближение x0.
3. Ищем корни, используя fsolve, а систему задаем как подфункцию

Пример 2.

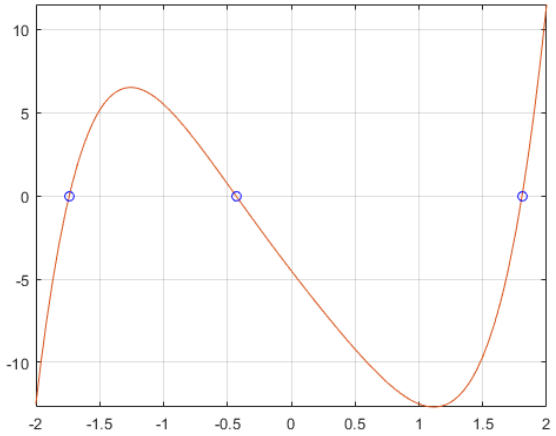
```
Clear, figure
%% строим графики
fplot('x*sin(8*x)',[-1 1]),hold on,
fplot('(x^5-x+0.5)',[-1 1]),grid on;
%% решаем, добавляем точки пересечения
x0=[-0.8,0.65;-0.2,0; 0.6,0]% нулевые приближения
        for i=1:length(x0')
                [x{i},Fx]=fsolve(@F,x0(i,:))
                plot(x{i}(1),x{i}(2),'r*')
                xy=['(',num2str(x{i}(1),2),','...
                        ,num2str(x{i}(2),2),')']
    text(x{i}(1),x{i}(2),['\leftarrow' xy])
       end % for
function f=F(x)  %% подфункция в этом же файле
f(1)=x(2)-x(1)*sin(8*x(1))
f(2)=x(2)-(x(1)^5-x(1)+0.5)
end
```



II) <u>Поиск корней, в т.ч. полиномов, функции</u> fzero, roots

а) Функция *roots* ищет корни полинома (все корни), аргументом процедуры является вектор коэффициентов полинома, начиная со старшей степени.

```
% Задание полиномов своими коэффициентами:
%[1 0 0 1 -10 -4.5] полином: x^5+x^2-10*x-4.5
x12345=roots([1 0 0 1 -10 -4.5])
%Ищем вещественные корни, отмечаем их на графике:
indexrealroots=find(imag(x12345)==0)
x12345(indexrealroots)
pts=x12345(indexrealroots)';
plot(pts,zeros(size(pts)),'bo')
% Построение полинома по его корням:
g=poly(x12345) % сравните с 'x^5+x^2-10*x-4.5'
Определение fzero вещественных корней функции:
xp1=fzero(@(x)x^5+x^2-10*x-4.5,-1.5)
xp2=fzero(@(x)x^5+x^2-10*x-4.5,0.5)
xp3=fzero(@(x)x^5+x^2-10*x-4.5,2)
```



б) Интерактивное задание `ginput` нулевого приближения:

```
figure
% y='sin(x.*exp(1./(x+0.1)))' %функция с   сильной
осцилляцией
fplot('sin(x.*exp(1./(x+0.1)))',[0,0.01]), hold on
% pointsX,pointsY - вектор координат
[pointsX,pointsY]=ginput(4) % отмечаем 4 точки для
нулевого приближения, на графике, они помещаются в
pointsX,pointsY
plot(pointsX,pointsY,'r^') % отмечаем нулевые
приближения
rootpts=fsolve('sin(x.*exp(1./(x+0.1)))',pointsX)
plot(rootpts,0,'mo')% нашли корни по заданным
приближениям
legend('y=sin(x.*exp(1./(x+0.1)))',...
    'initial points','roots')
```