```
func = cell(2,1);
func{1} = @(x,y) -x.^2 - y +x + 0.75;
func{2} = @(x,y) -x + y + 5*x*y;
x = [1.2 1.2]';

numIter = 50;


%func = handle of function returning [f1,f2,...,fn]
%x = guess solution vector [x1,x2,...,xn]
%tol = error toleance
%numIter = number of iteractions
tol = 1.0e8*eps;

%Ensure x is a column vector
% if size(x,1) == 1
%     x=x';
% end
```

Multiple newton raphson method

```
for i = 1:numIter
    [jac,f0]=findJacobian(func,x);
    if sqrt(dot(f0,f0)/length(x)) < tol
        fprintf('The roots are\n');
        roots = x
        return
    end
    dx = jac\(-f0);
    x = x + dx;
    if sqrt(dot(dx,dx)/length(x)) < tol
        fprintf('The roots are\n');
        roots=x
        return
    end
end
error('Maximum iterations exceeded')
```

function for Jacobian

```
function [jac, f0] = findJacobian(f,x)
h = 1.0e-06;
n = length(x);
jac = zeros(n);
f0 = [f{1}(x(1),x(2)) f{2}(x(1),x(2))]';
for i = 1:n
    tmp = x(i);
    x(i) = tmp + h;
    f1 = [f{1}(x(1),x(2)) f{2}(x(1),x(2))]';
    x(i) = tmp;
    jac(:,i)= (f1 - f0)/h;
```

```
end
end

The roots are

roots =

    1.4082
    0.1751
```

Published with MATLAB® R2018b