```
function [rootF,number_of_iteration] =
EXnewtonF(vector,initial,tolerance,maxiteration)
% ------------------------------------------------------------
% General usage: Newton's Method to find the roots of a polynomial
%
% Notes:
%      the ìrootî function in the MATLAB library can find all the
roots of
%      a polynomial with arbitrary order.
%      But this method, gives the one the roots based on the initial
guess
%      and it gives the number of iteration required to converge.
%
% Input:
%                   vector: The vector describing the polynomial
%                  initial: Initial guess
%               tolerance: The desired error
%            maxiteration: The maximum number of iteration
%
% Output:
%                     root: The detected root based on the initial
guess
%     number_of_iteration: number of iteraion to find the root
%
%
% Example:
%               f(x)=(x^3)-6(X^2)+72(x)-27=0
%               therefore
%               vector=[1 -6 -72 -27]
%               initial=300;
%               tolerance=10^-2;
%               maxiteration=10^4;
%               [root,number_of_iteration] =
newton(vector,initial,tolerance,maxiteration)
%               or
%               [root,number_of_iteration] = newton([1 -6 -72
-27],300,10^-2,10^4)
%               root=
%                   12.1229
%               number_of_iteration=
%                   13
%               This means that the detected root based on the initial
%               guess (300) is 12.1229 and it converge after 7
iterations.
%
% Author: Farhad Sedaghati
% Contact: <farhad_seda@yahoo.com>
% Written: 07/30/2015
%
% ------------------------------------------------------------
```

```
if nargin>4
    error('Too many input arguments');
elseif nargin==3
    maxiteration=10^4;
elseif nargin==2
    maxiteration=10^4;
    tolerance=10^-2;
elseif nargin<2
    error('Function needs more input arguments');
end

% switch the elements
vector(1:end)=vector(end:-1:1);
% Size of the vector
N=length(vector);
% initial guess
x=initial;
% dummy variable
sum=0;
for l=1:N
    % evaluate the polynomial using intiail value
    sum=sum+vector(l)*x.^(l-1);
end
number_of_iteration=0;
while abs(sum)>tolerance
    number_of_iteration=number_of_iteration+1;
    if number_of_iteration>maxiteration
        error('Failed to converge, the maximum number of iterations is
reached')
    end
    dif=0;
    sum=0;
    for k=2:N
        % finding the deravitive at a specific point
        dif=dif+(k-1)*vector(k)*x.^(k-2);
    end
    if dif==0
        error('The deravitve of the function is zero, peak another
initial point and run again')
    end
    for l=1:N
        % find the value of the polynomial at a specific point
        sum=sum+vector(l)*x.^(l-1);
    end
    % substituting in the newton's formula
    x=x-sum/dif;
end
rootF=x;
```