## Table of Contents



```
func =@(x) 0.5*x.^3 - 4*x.^2 + 5.5*x -1;
dfunc = @(x) 1.5*x.^2 - 8*x + 5.5;

% I find the range and derivate of the given function by using
%p = [0.5 -4 5.5 -1] and >> roots(p)  and then found the range by
 taking
%maximum and minimum values. derivate by using >>polyder(p)
```

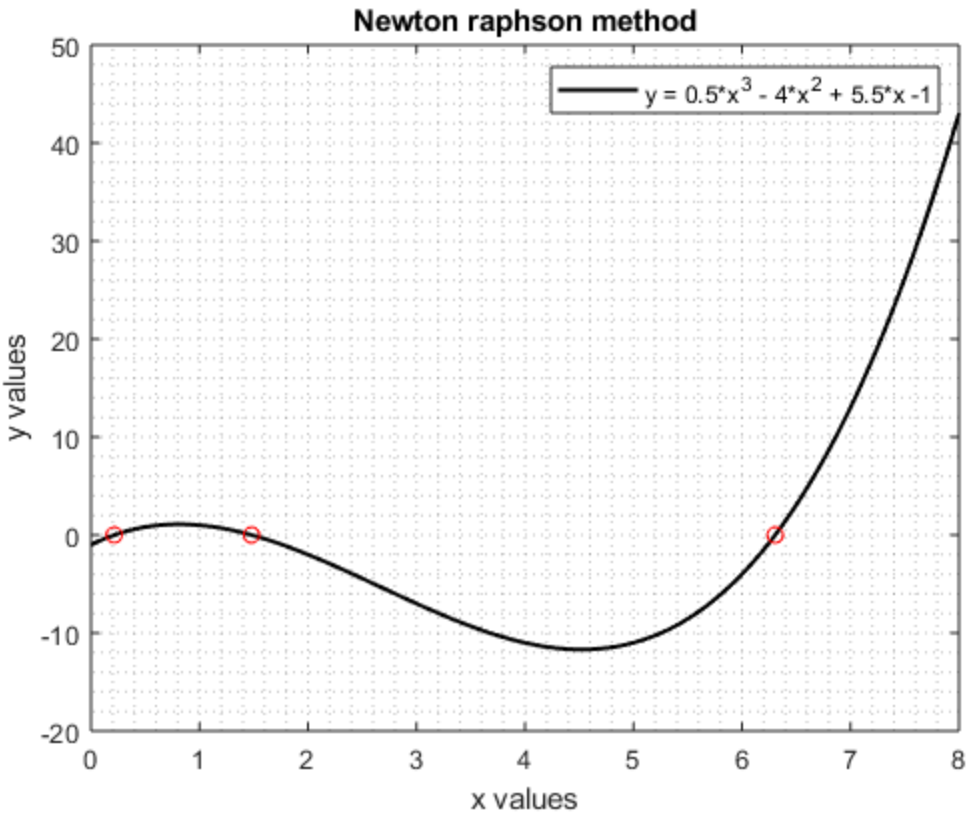
# values printing

```
fprintf('By taking the values of a =5 and b = 7');
x1 = newtRaph(func,dfunc,5,7,100)
fprintf('By taking the values of a = 1.2and b = 2');
x2 = newtRaph(func,dfunc,1.2,2,100)
fprintf('By taking the values of a =0 and b = 1');
x3 = newtRaph(func,dfunc,0,1,100)
```

```
By taking the values of a =5 and b = 7
```

# plots

```
x = linspace(0,8);
plot(x,func(x),'k','linewidth',1.5);
hold on;
grid minor;
xsol = [x1,x2,x3];
ysol = [0 0 0];
plot(xsol,ysol,'ro');
title('Newton raphson method');
xlabel('x values');
ylabel('y values');
legend('y = 0.5*x^3 - 4*x^2 + 5.5*x -1');
hold off;
```




# function declaration

```
function [sol,Iter] = newtRaph(func,dfunc,a,b,iter,tolerance)
%func - handle of the function returning f(x)
%dfunc - handle of the function returning f'(x)
% a,b - brackets of the solution
%tolerance - user defined error tolerance in solution
%iter - number of allowed iterations
% Iter - output iterations
% sol - output solution

if nargin < 6
    tolerance = 0.01;
end

fa = feval(func,a);
fb = feval(func,b);

if fa == 0
    sol = a;
    return;
end

if fb == 0
    sol = b;
```

```
    return;
end

if (fa * fb > 0.0)
    error('Solution does not lie within (a,b)')
end

x = (a + b)/2.0;

for i = 1:iter
    fx = feval(func,x);
    if abs(fx) < tolerance
        sol = x;
        return;
    end
    if fa * fx < 0.0
        b = x;
    else
        a = x;
    end

    % Newton-Raphson step
    dfx = feval(dfunc,x);
    if abs(dfx) == 0
        dx = b - a;
    else
        dx = -fx/dfx;
    end
    x = x + dx;

    %if x not in bracket, use bisection
    if (b - x) * (x - a) < 0.0
        dx = (b - a)/2.0;
        x = a + dx;
    end

    if abs(dx) < tolerance
        sol = x;
        Iter = i;
        return;
    end
end
sol = NaN

end
```

```
x1 =

    6.3065

By taking the values of a = 1.2and b = 2
x2 =
```

```
    1.4798

By taking the values of a =0 and b = 1
x3 =

    0.2141
```

*Published with MATLAB® R2018b*