Table of Contents

function handles

```
vol =@(h) pi*3*(h.^2) - (pi*(h.^3))/3;
dvol =@(h) -pi*(h.^2) + 18.8496*h;
```

Verifying roots by using inbuilt functions

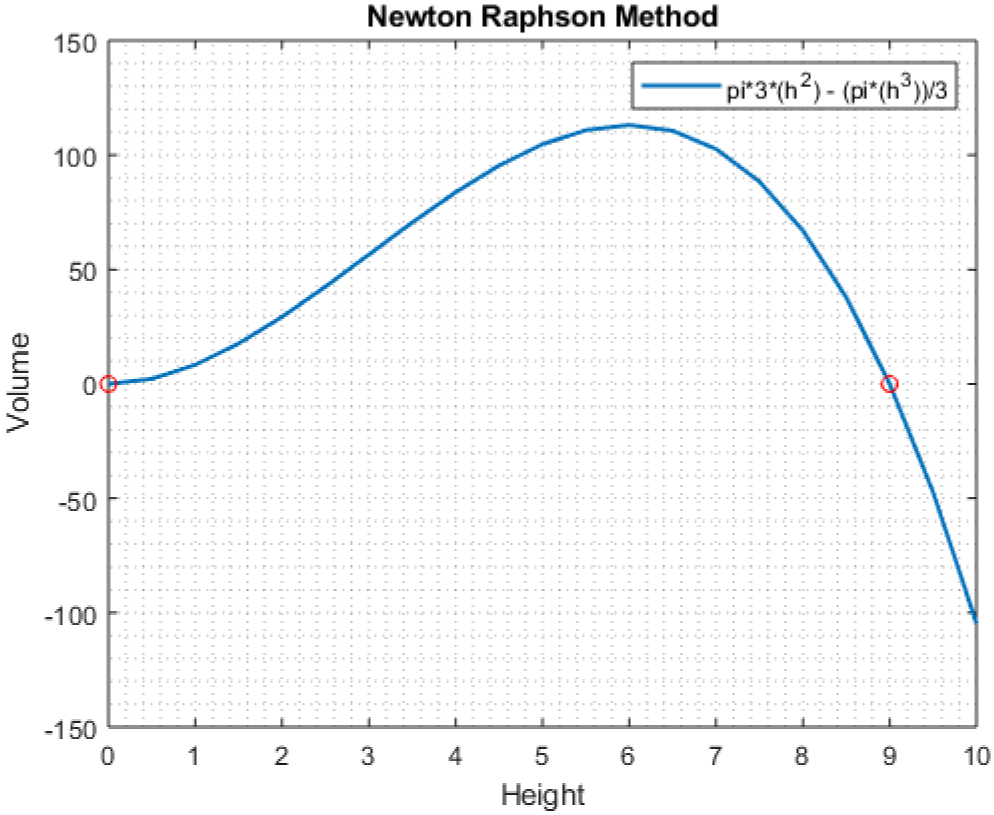
```
p = [pi/3 -3*pi  0 0];
H = roots(p);
```

function call

```
[hsol,iter] = newtRaph(vol,dvol,7,10,10)
```

Plots

```
yzeros = [0 0 0];
h = 0:0.5:10;
plot(h,vol(h),'linewidth',1.5);
hold on;
grid minor;
plot(H,yzeros,'ro')
title('Newton Raphson Method');
legend('pi*3*(h^2) - (pi*(h^3))/3')
xlabel('Height');
ylabel('Volume');
hold off;
```



function Newton Raphson Method

```
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
```

```
hsol =

    9.0000


iter =
3
```

Published with MATLAB® R2018b