

Newton-Raphson Method--Graphical Simulation of the Method.

© 2003 Nathan Collier, Autar Kaw, Jai Paul, Michael Keteltas, University of South Florida,
kaw@eng.usf.edu, <http://numericalmethods.eng.usf.edu/mws>

NOTE: This worksheet demonstrates the use of Maple to illustrate the Newton-Raphson method of finding roots of a nonlinear equation.

- Introduction

Newton-Raphson method [[text notes](#)][[PPT](#)] is based on the principle that if the initial guess of the root of $f(x) = 0$ is at x_i , then if one draws the tangent to the curve at $f(x_i)$, the point x_{i+1} where the tangent crosses the x-axis is an improved estimate of the root. Using the definition of the slope of a function,

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

The following simulation illustrates the Newton-Raphson method of finding roots of a nonlinear equation.

```
> restart;
```

- Section I : Data.

The following is the data that is used to solve the nonlinear equation which is obtained from the [floating ball problem](#) from the General Engineering to find the depth 'x' to which the ball is submerged under water

Function in $f(x)=0$

```
> f(x) := x^3 - 0.165*x^2 + 3.993*10^(-4) :
```

Initial guess

```
> x0 := 0.05 :
```

Upper bound of range of 'x' that is desired

```
> uxrange := 0.12 :
```

Lower bound of range of 'x' that is desired

```
> lxrange := -0.02 :
```

- Section II: Before you start.

Because the method uses a line tangent to the function at the initial guess, we must calculate the derivative of the function to find the slope of the line at this point. Here we will define the derivative of the function $f(x)$ as $g(x)$.

```
[ > g(x) :=diff(f(x),x);
```

$$g(x) := 3x^2 - 0.330x$$

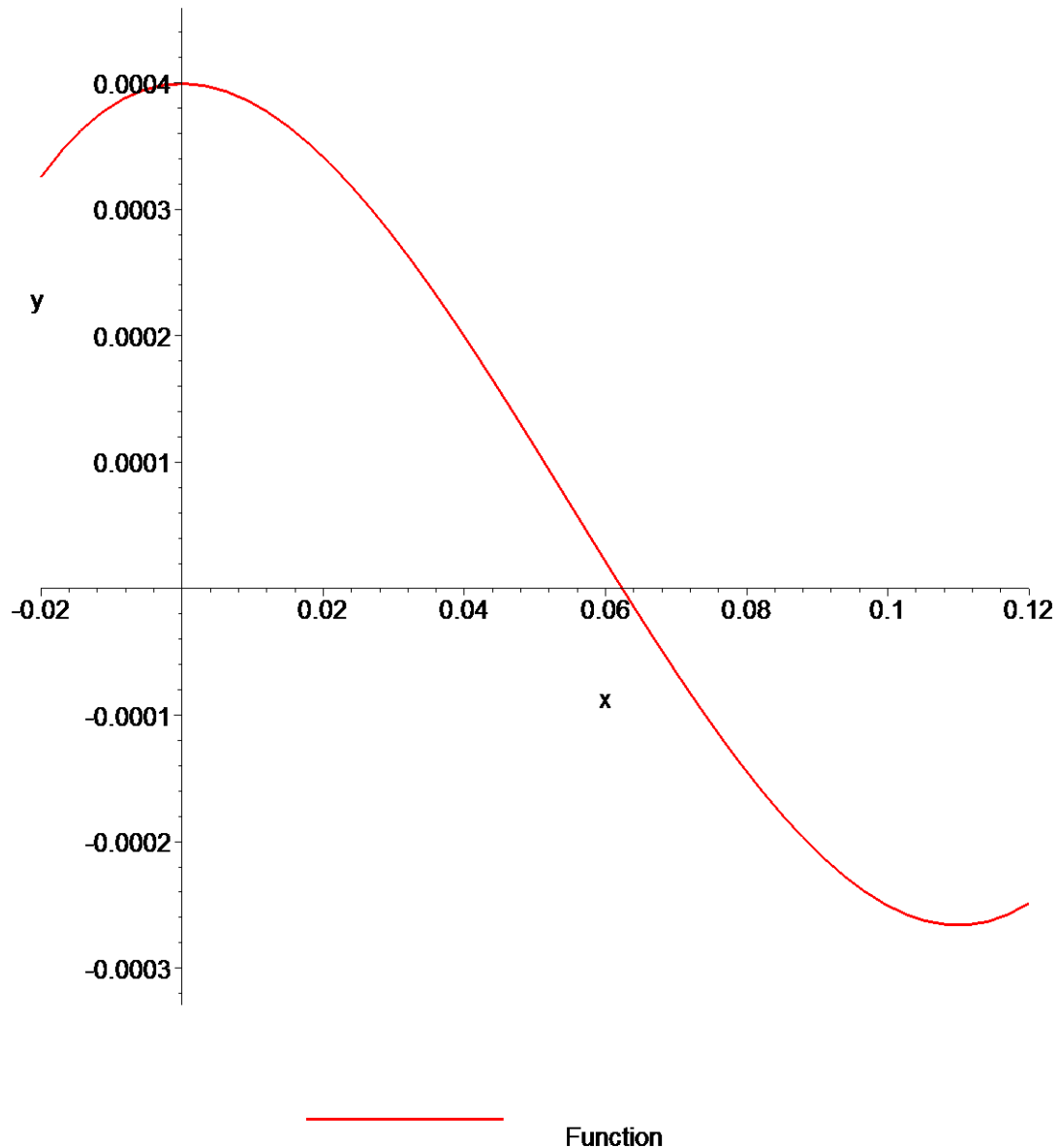
We now plot the data. The following function determines the upper and lower ranges on the Y-axis. This is done using the upper and lower ranges of the X-axis specified, and the value of the original functional at these values.

```
[ > yranger:=proc(uxrange,lxrange)
  local i,maxi,mini,tot;
  maxi:=eval(f(x),x=lxrange);
  mini:=eval(f(x),x=lxrange);
  for i from lxrange by (uxrange-lxrange)/10 to uxrange do
  if eval(f(x),x=i)<mini then mini:=eval(f(x),x=i) end if;
  if eval(f(x),x=i)>maxi then maxi:=eval(f(x),x=i) end if;
  end do;
  tot:=maxi-mini;
  -0.1*tot+mini..0.1*tot+maxi;
end proc:
[ > yrange:=yranger(uxrange,lxrange):
[ > xrange:=lxrange..uxrange:
```

The following calls are needed to use the plot function

```
[ > with(plots):
Warning, the name changecoords has been redefined
[ > with(plottools):
Warning, the name arrow has been redefined
[ > plot(f(x),x=xrange,y=yrange,title="Entered function on given
interval",legend=["Function"],thickness=3);
```

Entered function on given interval



- Section III: Iteration 1.

The Newton Raphson Method works by taking a tangent line at the value of the function at the initial guess, and seeing where that tangent line crosses the x-axis. This value will be the new estimate for the root and can be obtained by using the above formula.

The first estimate of the root is,

```
> x1:=x0-eval(f(x),x=x0)/eval(g(x),x=x0);  
x1 := 0.06242222222
```

How good is that answer? Find the absolute relative approximate error to judge.

```
> epsilon:=abs((x1-x0)/x1)*100;
```

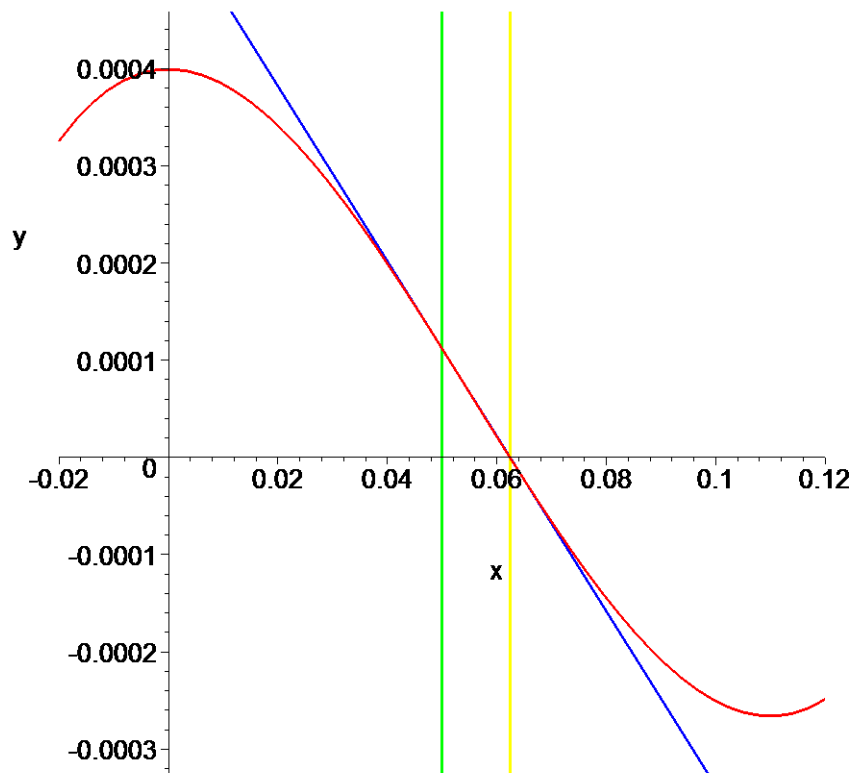
$$\varepsilon := 19.90032040$$





While not necessary to the method, for graphing purposes we define the equation of the tangent line touching x_0 .

```
> tanline(x) := eval(g(x), x=x0)*x + eval(f(x), x=x0) - eval(g(x), x=x0)*x0:  
> plot([f(x), [x0, t, t=yrange], [x1, t, t=yrange], tanline(x)], x=xrange, y=yrange, title="Entered function on given interval with current and next root\n and tangent line of the curve at the current root", legend=["Function", "x0, Current root", "x1, New root", "Tangent line"], thickness=3);
```

Entered function on given interval with current and next root

and tangent line of the curve at the current root



	Function
	x_0 , Current root
	x_1 , New root
	Tangent line

Section IV: Iteration 2.

The same formula is used in iterations to calculate the next estimation of the root. The second estimate of the root is,

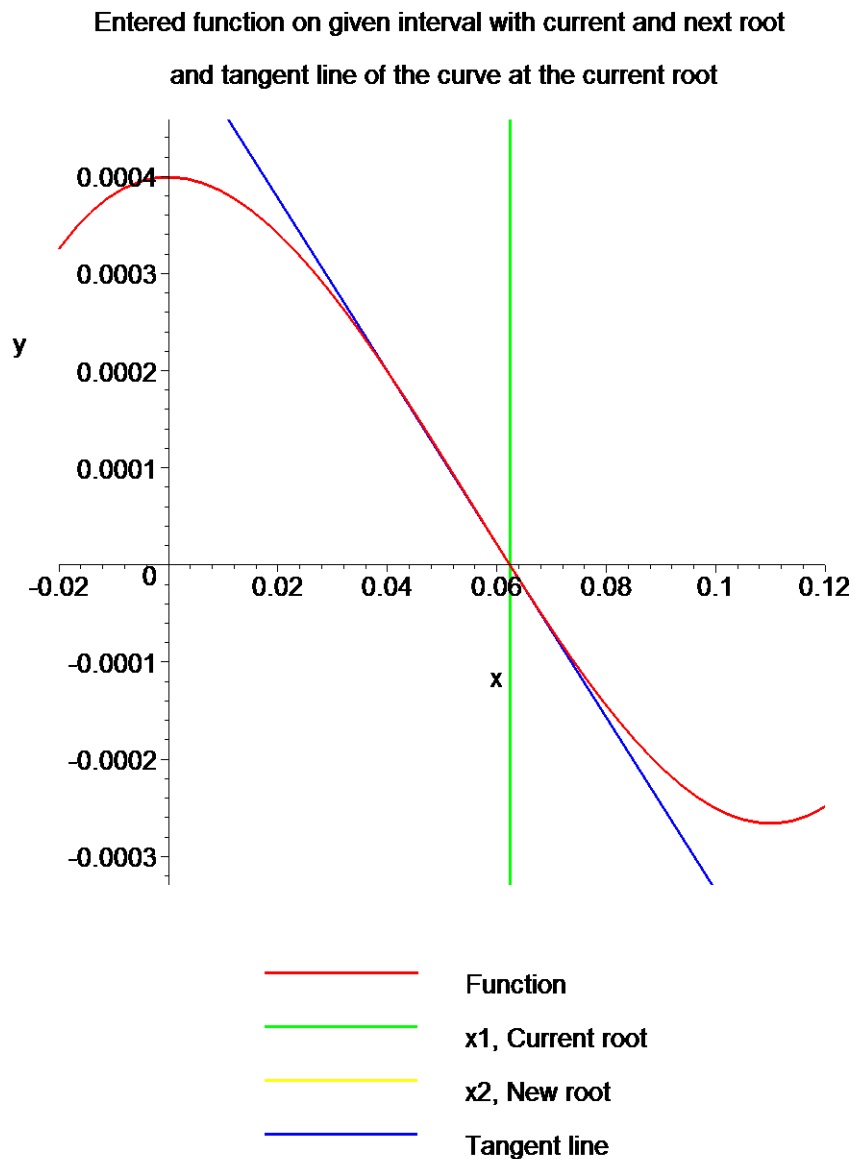
```
[ > x2:=x1-eval(f(x),x=x1)/eval(g(x),x=x1);  
                                     x2 := 0.06237757653
```

How good is that answer? Find the absolute relative approximate error to judge.

```
[ > epsilon:=abs((x2-x1)/x2)*100;  
                                     ε := 0.07157330003
```

While not necessary to the method, for graphing purposes we define the equation of the tangent line touching x_0 .

```
[ > tanline(x):=eval(g(x),x=x1)*x+eval(f(x),x=x1)-eval(g(x),x=x1)*x  
  1:  
  
[ > plot([f(x),[x1,t,t=yrange],[x2,t,t=yrange],tanline(x)],x=xrange  
  ,y=yrange,title="Entered function on given interval with  
  current and next root\n and tangent line of the curve at the  
  current root",legend=["Function", "x1, Current root", "x2, New  
  root", "Tangent line"],thickness=3);
```



Section V: Iteration 3.

The same formula is used in iterations to calculate the next estimation of the root. The third estimate of the root is,

```
> x3:=x2-eval ( f ( x ) , x=x2 ) /eval ( g ( x ) , x=x2 ) ;  

x3 := 0.06237758151
```

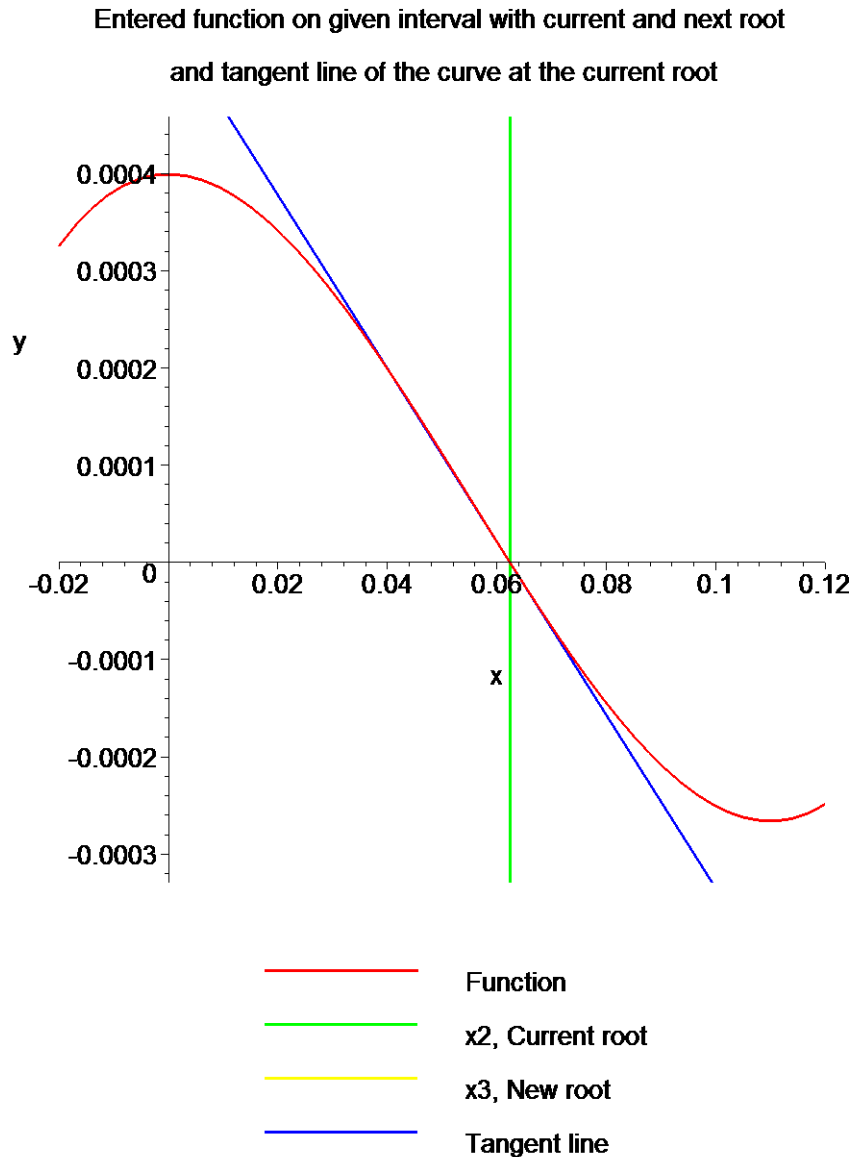
How good is that answer? Find the absolute relative approximate error to judge.

```
> epsilon:=abs ( ( x3-x2 ) /x3 ) *100 ;
```

$$\varepsilon := 0.7983637518 \cdot 10^{-5}$$

While not necessary to the method, for graphing purposes we define the equation of the tangent line touching x_0 .

```
> tanline(x) := eval(g(x), x=x2) * x + eval(f(x), x=x2) - eval(g(x), x=x2) * x  
2:  
  
> plot([f(x), [x2, t, t=yrange], [x3, t, t=yrange], tanline(x)], x=xrange  
, y=yrange, title="Entered function on given interval with  
current and next root\n and tangent line of the curve at the  
current root", legend=["Function", "x2, Current root", "x3, New  
root", "Tangent line"], thickness=3);
```



>

- Section VI: Conclusion.

Maple helped us to apply our knowledge of numerical methods of finding roots of a nonlinear equation to simulate the Newton-Raphson method and find the root of the given nonlinear equation.

References

[1] *Nathan Collier, Autar Kaw, Jai Paul, Michael Keteltas, Holistic Numerical Methods Institute, See http://numericalmethods.eng.usf.edu/mws/gen/03nle/mws_gen_nle_txt_newton.pdf*

Disclaimer: While every effort has been made to validate the solutions in this worksheet, University of South Florida and the contributors are not responsible for any errors contained and are not liable for any damages resulting from the use of this material.