Root jumping several roots away--Newton-Raphson Method.

kaw@eng.usf.edu , http://numericalmethods.eng.usf.edu/mws

NOTE: This worksheet demonstrates the use of Maple to illustrate how, in the Newton-Raphson method, an initial guess close to one root can jump to a location several roots away when a function is oscillatory in nature.

Introduction

Although, Newton-Raphson method [text notes][PPT] converges faster than any other method it still has a few drawbacks.One of them is Root jumping. In some cases where the function f(x) is oscillating and has a number of roots, one may choose an initial guess close to a root. However, the guesses may jump and converge to some other root. The following simulation illustrates how, in the Newton-Raphson method, an initial guess close to one root can jump to a location several roots away when a function is oscillatory in nature.

```
> restart;
```

Section I : Data.

Function in f(x)=0

```
> f(x):=sin(x):
```

Initial guess

```
> x0:=1.431:
```

Upper bound of range of 'x' that is desired

```
> uxrange:=10.0:
```

Lower bound of range of 'x' that is desired

```
> lxrange:=-10.0:
```

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);
```

$$g(x) := \cos(x)$$

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.

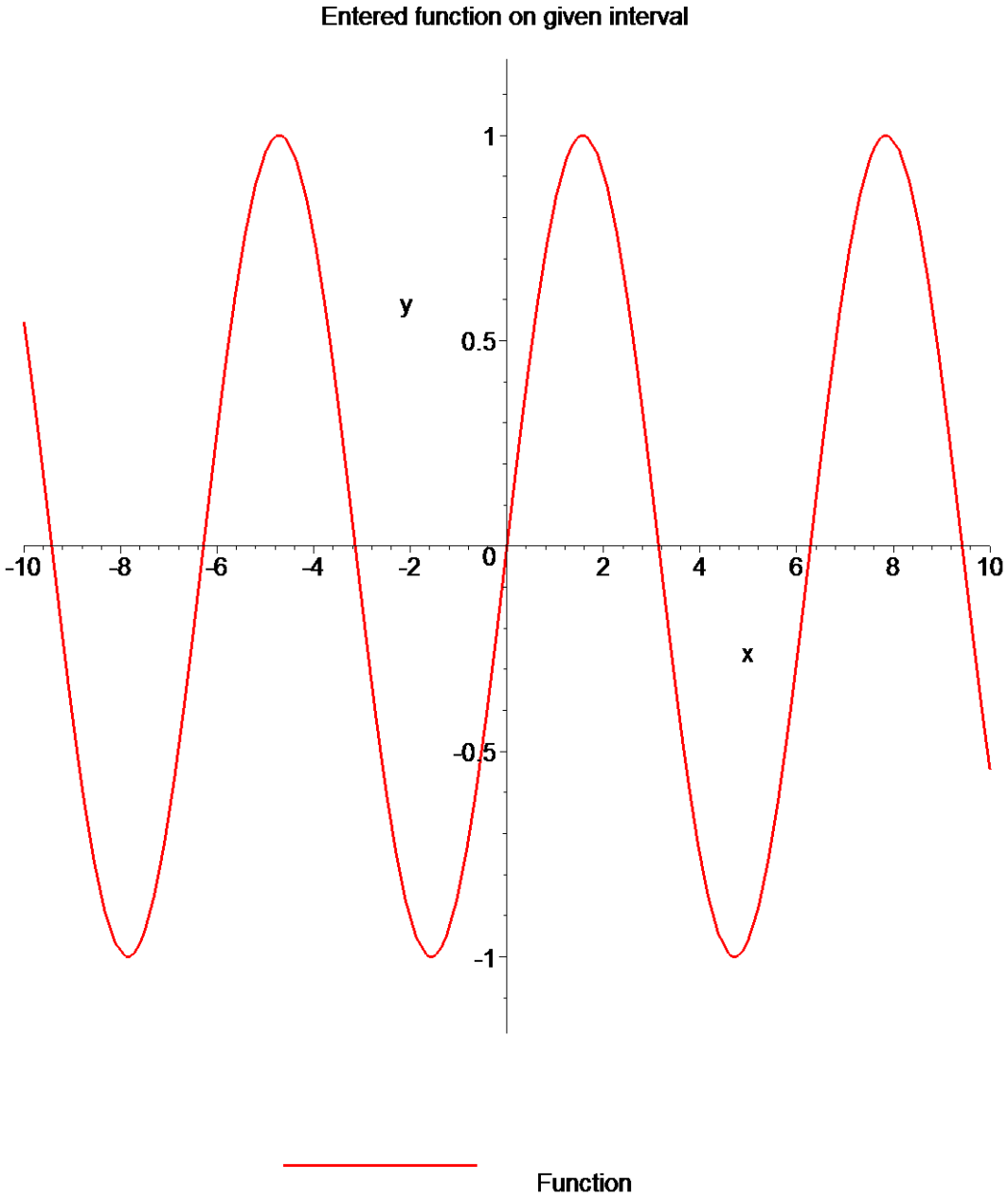
```
> yranger:=proc(uxrange,lxrange)
  local i,maxi,mini,tot;
  maxi:=eval(f(x),x=lxrange);
  mini:=eval(f(x),x=lxrange);
  for i from lxrange by (uxrange-lxrange)/10 to uxrange do
  if eval(f(x),x=i)<mini then mini:=eval(f(x),x=i) end if;
  if eval(f(x),x=i)>maxi then maxi:=eval(f(x),x=i) end if;
 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



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..

The first estimate of the root is,

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

$$x1 := -5.675604171$$

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

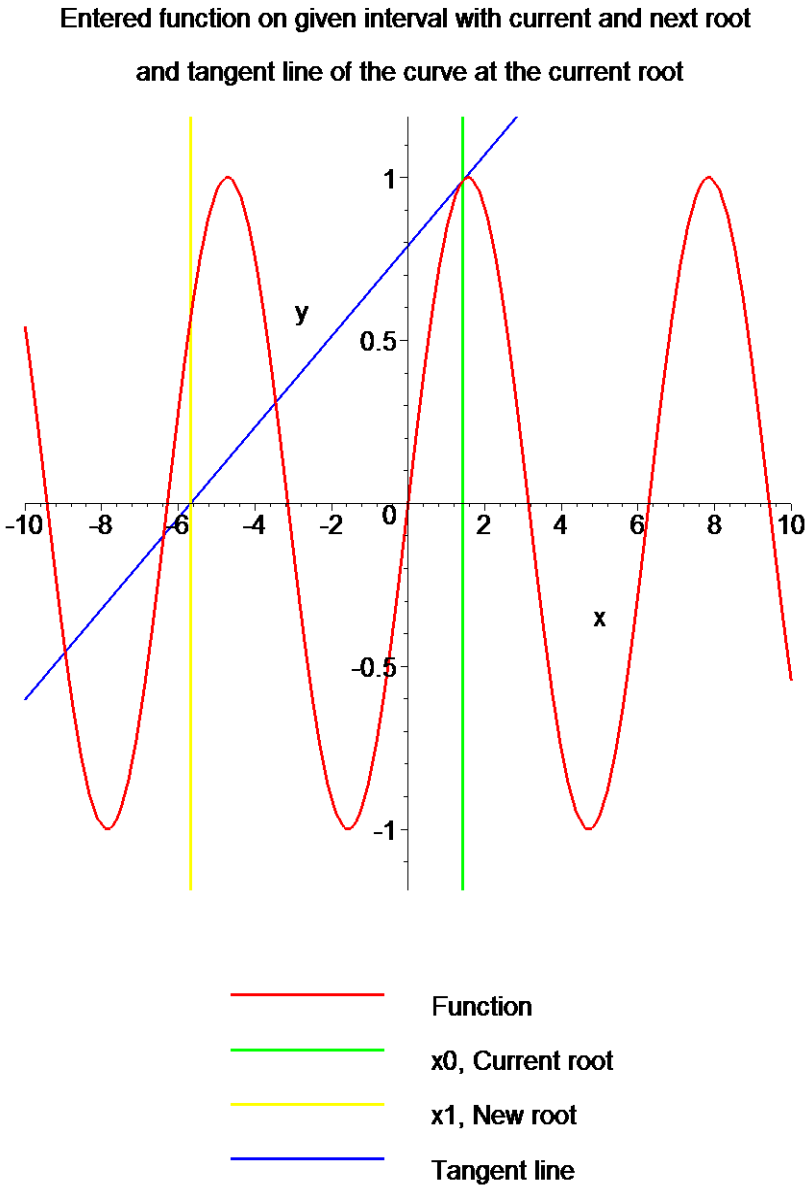
```
> epsilon:=abs((x1-x0)/x1)*100;
```

$$\varepsilon := 125.2131748$$

While not necesary to the method, for graphing purposes we define the equation of the tangent line touching x0.

```
> tanline(x):=eval(g(x),x=x0)*x+eval(f(x),x=x0)-eval(g(x),x=x0)*x
  0:
```

```
> 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);
```



Section IV: Iteration 2.

The same method 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);
```

$$x2 := -6.370928654$$

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

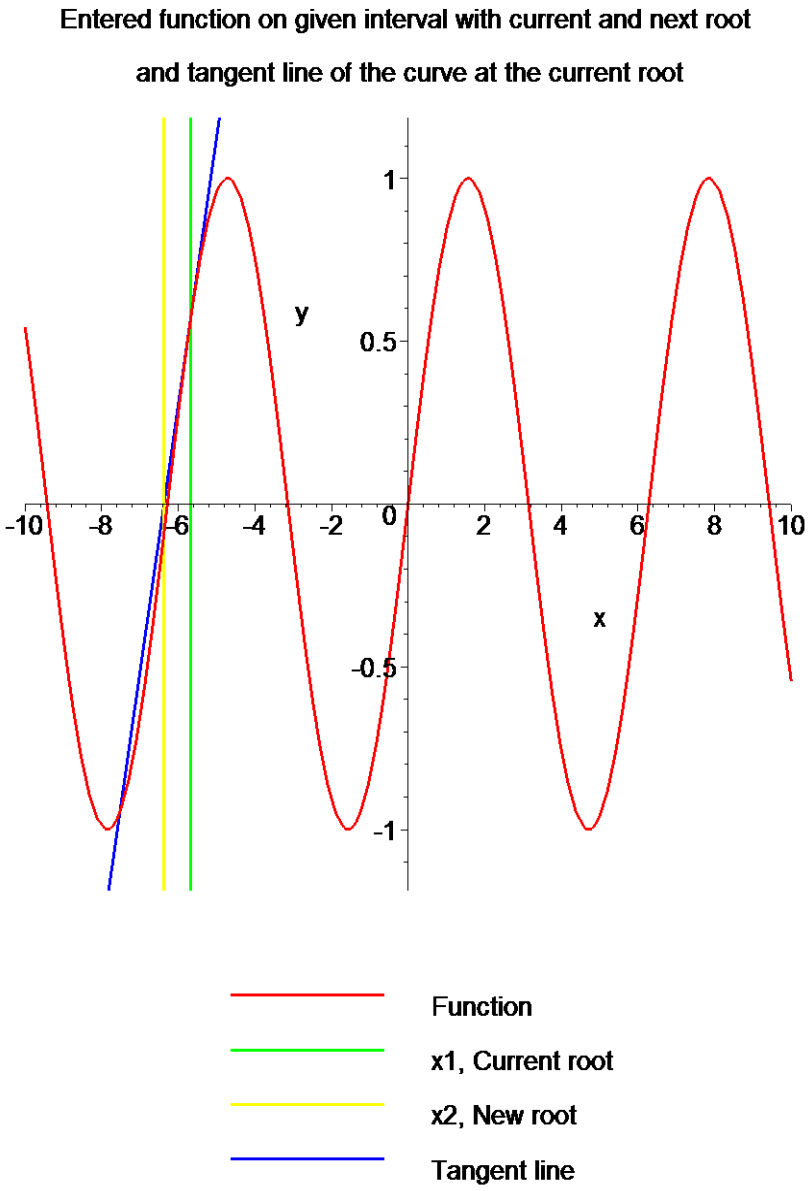
```
> epsilon:=abs((x2-x1)/x2)*100;
```

$$\varepsilon := 10.91402087$$

While not necesary to the method, for graphing purposes we define the equation of the tangent line touching x0.

```
> 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 method 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);
```

$$x3 := -6.282959436$$

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

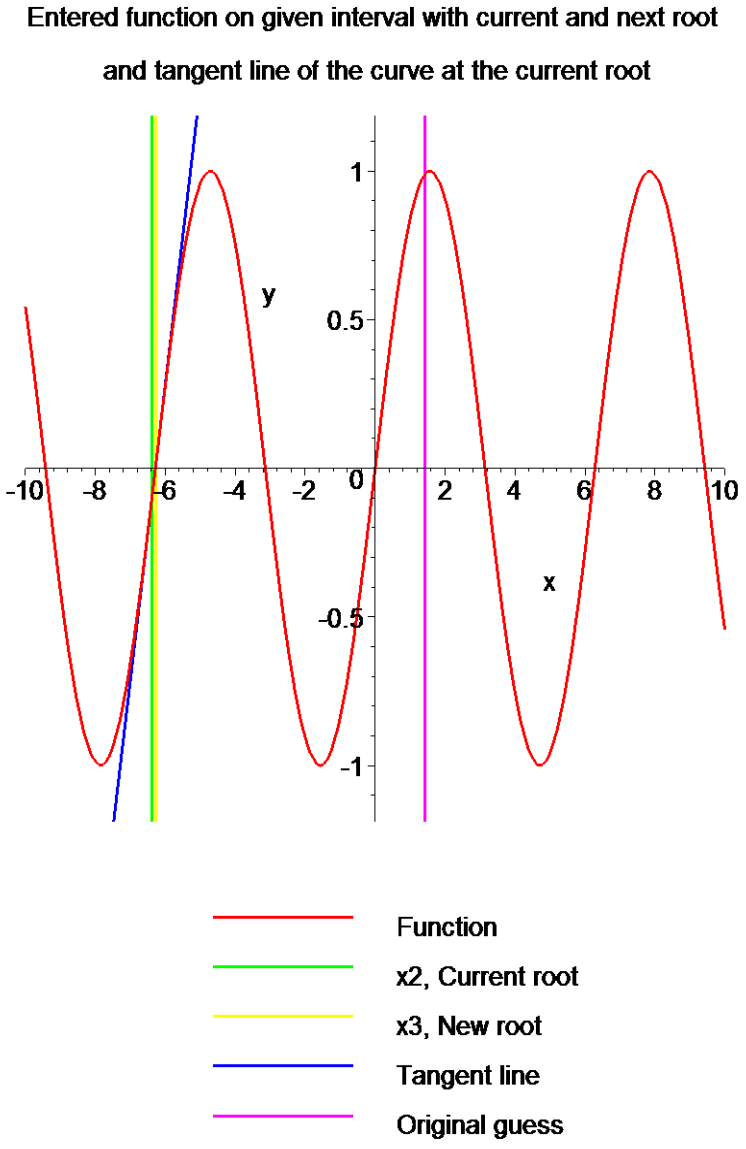
```
> epsilon:=abs((x3-x2)/x3)*100;
```

$$\varepsilon := 1.400123921$$

While not necesary to the method, for graphing purposes we define the equation of the tangent line touching x0.

```
> 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),[x0,t,t=y
  range]],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", "Original
  guess"],thickness=3);
```




```
>
```

Section VI: Conclusion.

Maple helped us to apply our knowledge of numerical methods of finding roots of a nonlinear equation using the Newton-Raphson method to understand how an initial guess close to one root can jump to a location several roots away when a function is oscillatory in nature.

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*