Slow convergence around inflection points--Newton-Raphson Method.

NOTE: This worksheet demonstrates the use of Maple to illustrate how the Newton-Raphson method converges slowly due to an inflection point occuring in the vicinity of the root.

Introduction

Although, Newton-Raphson method [text notes][PPT] converges faster than any other method it still has a few drawbacks.One of them is Divergence at inflection points. For a function f(x), the point where the concavity changes from up-to-down or down-to-up are called inflection points. Theorem for a function f(x) that states : "If f '(c) exists and f "(c) changes sign at x = c , then the point (c, f(c)) is an inflection point of the graph of f. The following simulation illustrates the Newton-Raphson method converges slowly due to an inflection point occuring in the vicinity of the root.

```
> restart;
```

Section I : Data.

Function in f(x)=0

```
> f(x):=(x-1)^3:
```

Initial guess

```
> x0:=-1.0:
```

Upper bound of range of 'x' that is desired

```
> uxrange:=3.0:
```

Lower bound of range of 'x' that is desired

```
> lxrange:=-3.0:
```

Maximum number of iterations

```
> nmax:=10:
```

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);
```

$$g(x) := 3\,(x-1)^2$$

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.

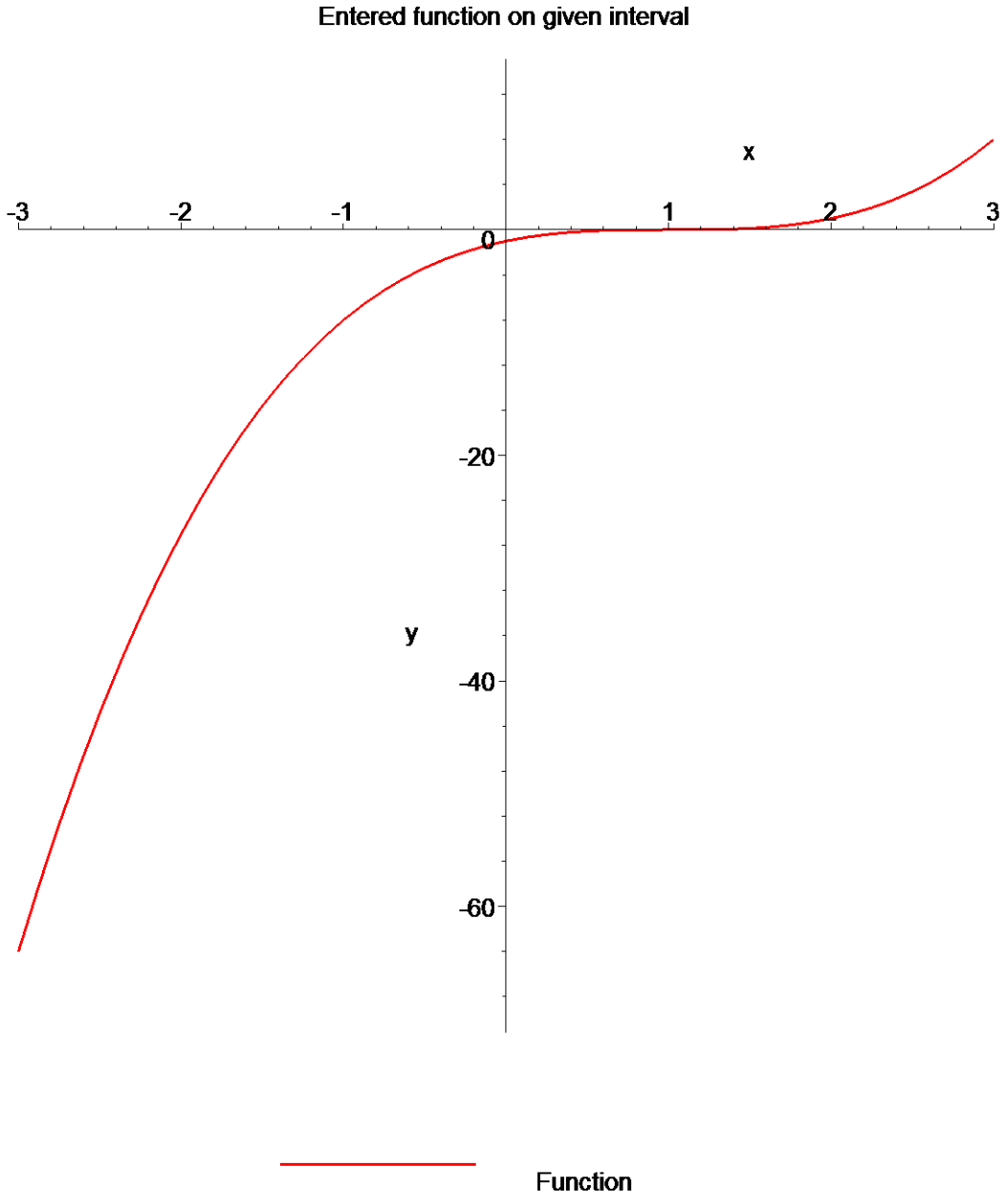
```
> yranger:=proc(uxrange,lxrange)
  local i,maxi,mini,tot;
  maxi:=eval(f(x),x=lxrange);
  mini:=eval(f(x),x=lxrange);
  for i from lxrange by (uxrange-lxrange)/10 to uxrange do
  if eval(f(x),x=i)<mini then mini:=eval(f(x),x=i) end if;
  if eval(f(x),x=i)>maxi then maxi:=eval(f(x),x=i) end if;
 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

$$x1 := -0.3333333333$$

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

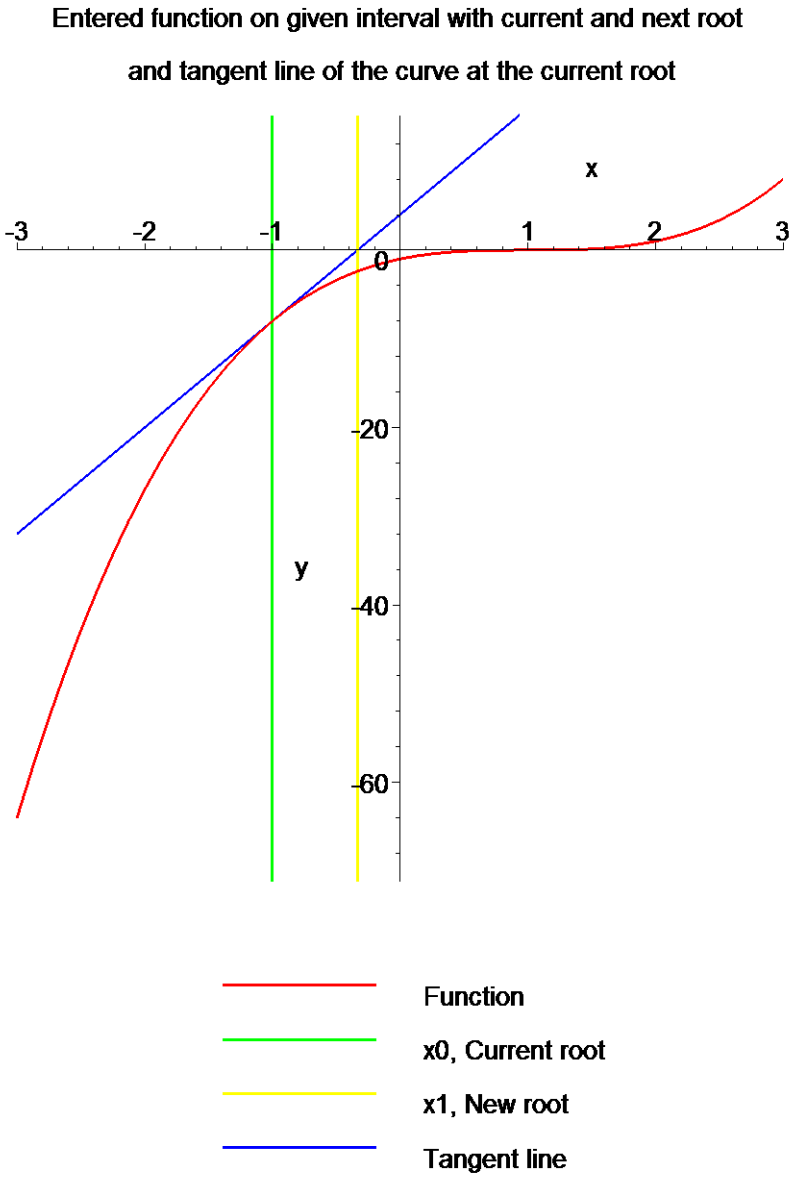
```
> epsilon:=abs((x1-x0)/x1)*100;
```

$$\varepsilon := 200.0000000$$

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);
```

$$x2 := 0.1111111111$$

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

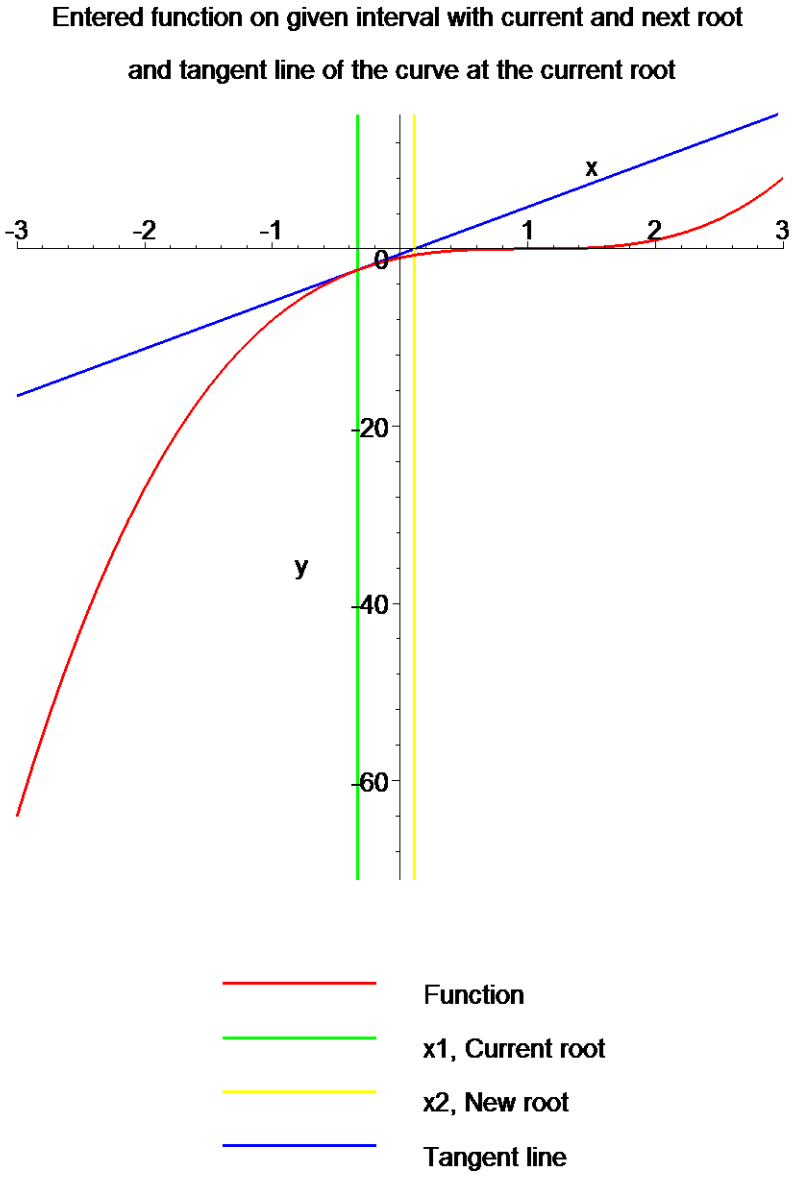
```
> epsilon:=abs((x2-x1)/x2)*100;
```

$$\varepsilon := 400.0000000$$

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);
```

$$x3 := 0.4074074074$$

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

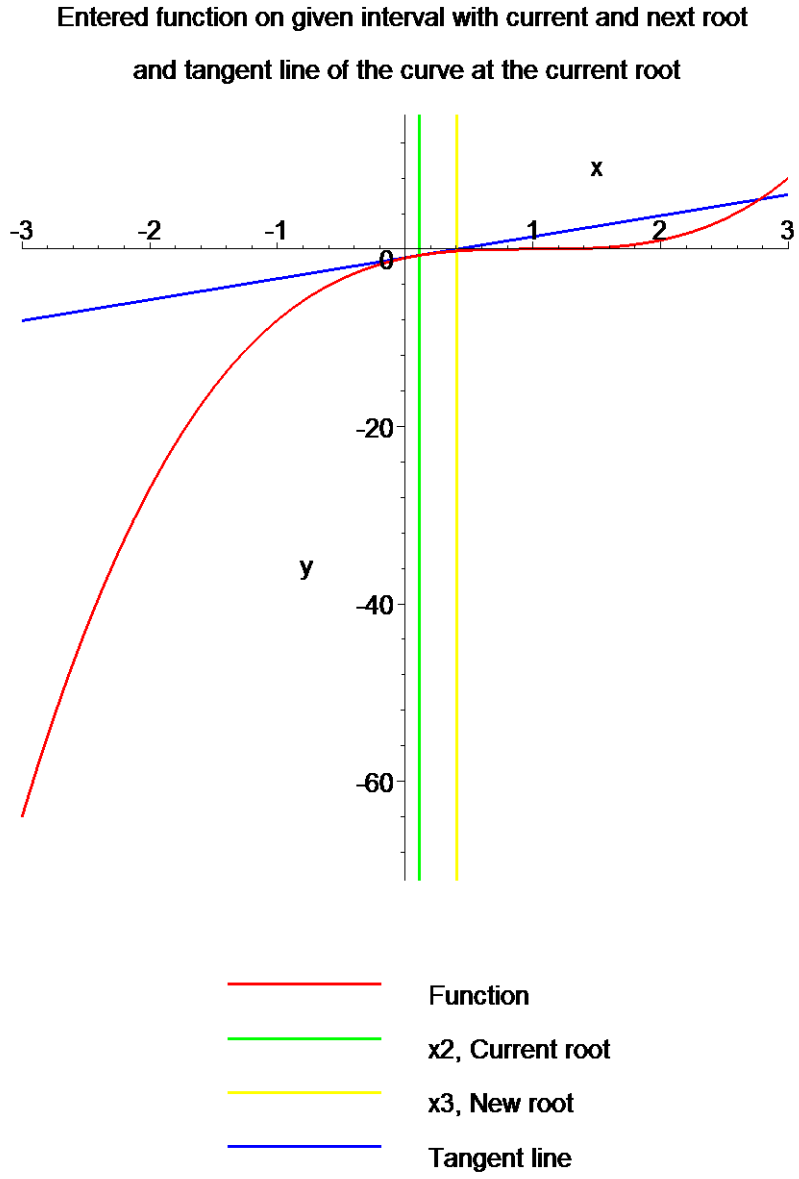
```
> epsilon:=abs((x3-x2)/x3)*100;
```

$$\varepsilon := 72.72727273$$

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)],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: Iteration 4.

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

```
> x4:=x3-eval(f(x),x=x3)/eval(g(x),x=x3);
```

$$x4 := 0.6049382717$$

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

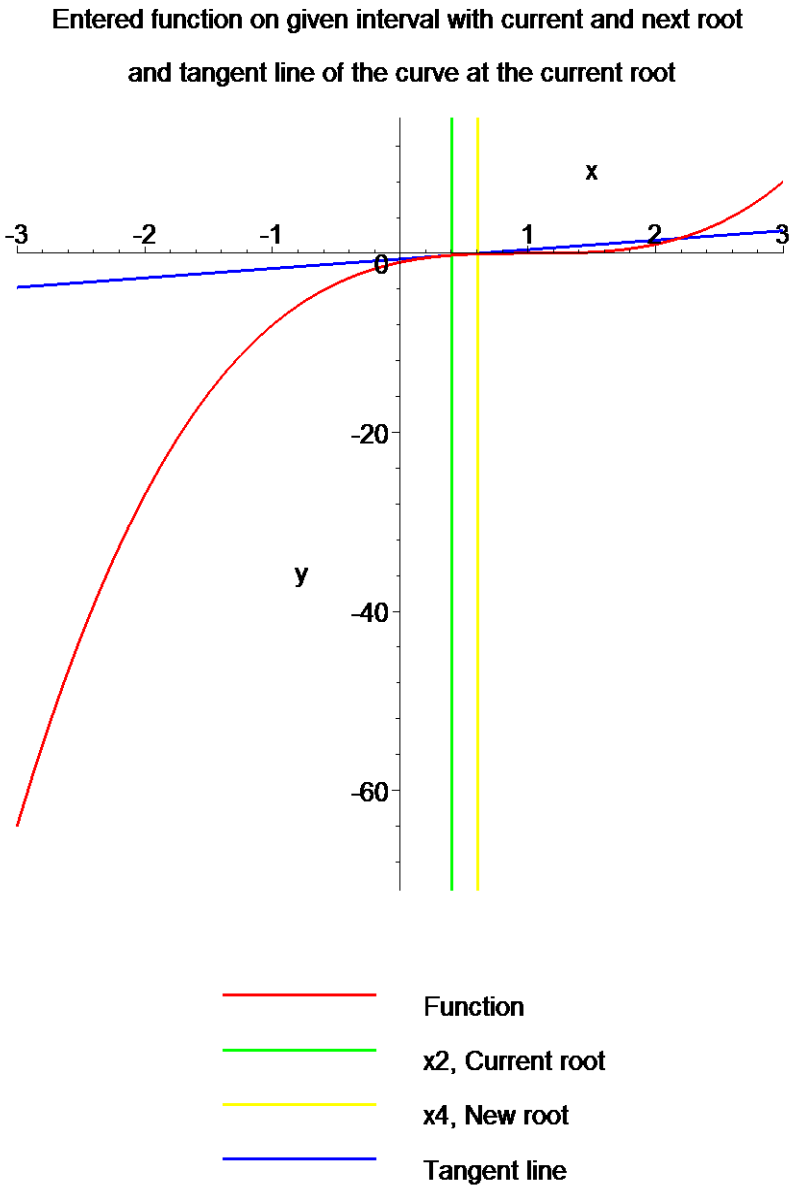
```
> epsilon:=abs((x4-x3)/x4)*100;
```

$$\varepsilon := 32.65306124$$

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

```
> tanline(x):=eval(f(x),x=x3)+(0-eval(f(x),x=x3))/(x4-x3)*(x-x3):
```

```
> plot([f(x),[x3,t,t=yrange],[x4,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", "x4, New
  root", "Tangent line"],thickness=3);
```



>

Section VII: Value of Root.

Here we will calculate the root and try to plot the absolute relative approximate error aginst the number of iterations to understand teh slow convergence.

Root Calculation

```
> xr:=proc(n)
```

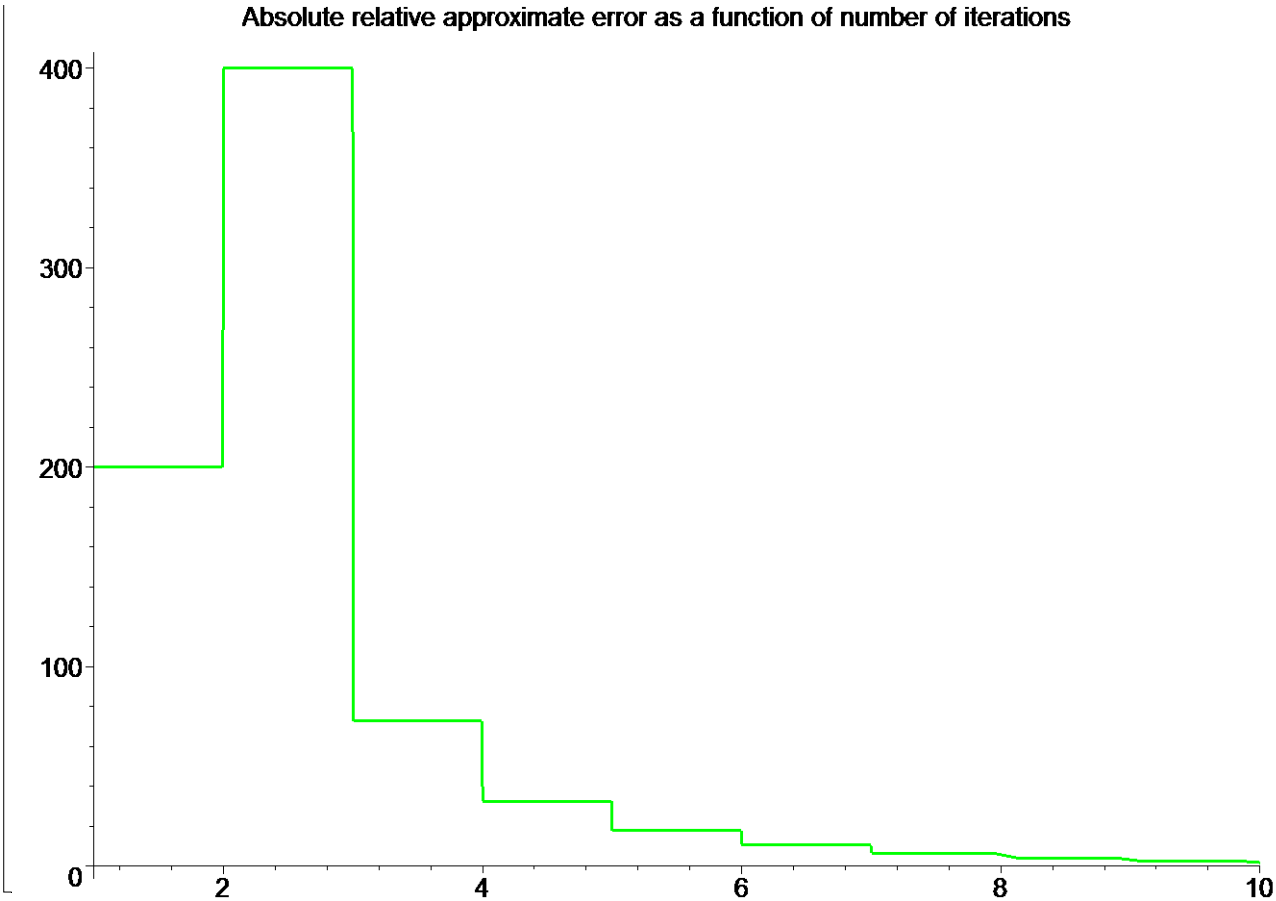
```
        local p, q, i;
        p:=x0;
        q:=x0;
        for i from 1 to n do
            p:=q-eval(f(x),x=q)/eval(g(x),x=q);
            q:=p;
        end do;

        p;
    end proc:
> nrange:=1..nmax:
```

Absolute approximate error

```
> Ea:=proc(n)
      if n=1 then
        abs(xr(n)-x0);
      else
        abs(xr(n)-xr(n-1));
      end if;
    end proc:
```

Absolute relative approximate error

```
> ea:=proc(n)
  abs(Ea(n)/xr(n))*100;
  end proc:
> plot(ea,nrange,title="Absolute relative approximate error as a
  function of number of iterations",thickness=3,color=green);
```



As the above graph depicts, convergence will still happen, but very slowly. In this example, after 10 iterations, only 1 significant digit is correct in the value of the root.

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 the method can have slow convergence due to an inflection point occuring in the vicinity of the root.

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*