Code for Müller's method to check the order of convergence

```
using Printf
setprecision(4096)
f(x)=5*x*exp(-x)-1
x0=big"2.0"; x1=big"2.1"; x2=big"1.9"

en=1.0
for n=1:9
    global en,x0,x1,x2
    d1=(f(x1)-f(x0))/(x1-x0)
    d2=(f(x2)-f(x1))/(x2-x1)
    c=f(x0)
    a=(d2-d1)/(x2-x0)
    b=d1-a*(x1-x0)
    en0=en
    m1=-b+sqrt(b^2-4*a*c)
    m2=-b-sqrt(b^2-4*a*c)
    r1=0.0; r2=0.0
    if abs(m1)>=abs(m2)
        r1=m1/(2*a); r2=2*c/m1
    else
        r1=2*c/m2; r2=m2/(2*a)
    end
    x2=x1; x1=x0
    if abs(r1)<=abs(r2)
        x0=x1+r1
    else
        x0=x1+r2
    end
    en=abs(x0-x1)
    @printf("x0=%g\n",x0)
    println("n=$(n-1) log(e(n+1))/log(e(n))=",
    Float64(log(en)/log(en0)))
end
```

```
julia> include("mullerbig.jl")
x0=2.52283
n=0 log(e(n+1))/log(e(n))=-Inf
x0=2.54217
n=1 log(e(n+1))/log(e(n))=6.084164846491169
x0=2.54264
n=2 log(e(n+1))/log(e(n))=1.9440095342086527
x0=2.54264
n=3 log(e(n+1))/log(e(n))=1.9266232320886907
x0=2.54264
n=4 log(e(n+1))/log(e(n))=1.9867995256419966
x0=2.54264
n=5 log(e(n+1))/log(e(n))=1.8669641827388768
x0=2.54264
n=6 log(e(n+1))/log(e(n))=1.8600909686417904
x0=2.54264
n=7 log(e(n+1))/log(e(n))=1.8550641708215654
x0=2.54264
n=8 log(e(n+1))/log(e(n))=1.8447717032264295
```

About 84 percent more significant digits per iteration

no

order is about $p \approx 1.84$

The order is greater than secant method where $p = 1.618$. This makes sense, since secants are linear interpolations and Muller's method uses quadratic interpolation, which should be more accurate.