We'll improve Newton's method and I'll give a somewhat open-ended problem to solve using it.

1 Review of last week

Last week, we looked for solutions to the equation \( f(x) = xe^{-x} - .5x + \cos(x) \). This required us to save the following two functions

\[
\begin{align*}
  f & := \text{function}(x) \{ x \cdot \exp(-x) -.5 \cdot x + \cos(x) \} \\
  f' & := \text{function}(x) \{ \exp(-x) - x \cdot \exp(-x) -.5 - \sin(x) \}
\end{align*}
\]

We also wrote the function "newton" that ran this command several times

\[
x := x - \frac{f(x)}{f'(x)}
\]

When using "newton" we specified how many iterations of newton's method we wanted to do.

2 Specifying the Error

Newton's method does not give an exact answer, no matter how many iterations that we do. Instead, we want to find \( x \) so that \( f(x) \approx 0 \). So we'll pick a small number and run newton's method until \( f(x) \) becomes smaller than it.

I've written the code that does this. It uses the while command, which means "run this command while the following is true".

\[
\text{newton} := \text{function}(x0, \text{error}) \{
  x := x0 \\
  \text{while} (\text{max} (\text{abs} (f(x))) > \text{error}) \{
    x := x - \frac{f(x)}{f'(x)}
  \} \\
  \text{print}(x)
\}
\]

Let's use the new function

\[
\begin{align*}
  \text{newton}(1, .0001) \\
  [1] 1.281694
\end{align*}
\]

The nice thing about this function is that we can run two guesses at once. Here I try guesses \( x0 = 0, 1, 2 \).

\[
\begin{align*}
  \text{newton}(c(0, 1, 2), .0001) \\
  [1] -0.6106643 \text{ 1.2816943 \text{ 1.2816943}}
\end{align*}
\]
3 The problem with Newton’s method

Newton’s method is great and is the preferred method for solving nonlinear problems. But sometimes it fails.

Try

>newton(10,.01)

The downfall of Newton’s method is that it sometimes "blows up". The problem is that if \( f'(x) \) is small, then you divide by a very small number.

4 Your tasks for this week

**Avoid Blowups:** Modify the ‘newton’ function so that it checks to make sure that \( f'(x) \) isn’t too small. Specifically, if \( \| f'(x) \| < .01 \) let \( x = x - .1 \). To do this you’ll need to use an if/else command. I’ve started it for you below.

```r
if(....){....}
else{x<-x-f(x)/fprime(x)
```

**Multinewton** Unfortunately, the function above can only take one value at a time. So the next step is to write a function that takes a list of \( x_0 \) values and runs newton on each of them. Again, I’ve written most of it below.

```r
multinewton<-function(x0list,error){
for(x0 in x0list){....}
}
```

**New Equation:** Consider the equation \( f(x) = .2 * x + 7 * \cos(x + \cos(x)) - 2 = 0 \). Find all 21 zeros using Newton’s method. Include a plot of the function that shows all of the zeros of this function. Describe the method that you used to find the zeros and be specific.