CSCI 2202 Computer Modelling for Scientists Lab 8b 25 Feb 2020
Root Finding 2: Newton's Method

(1)

- There are many ways to reach $f(x_r) = 0$, to find x_r. We can use iteration on $f(x_n)$ to decide on the next approximation:

$$x_{n+1} = F(x_n) = x_n - c \cdot f(x_n)$$

The choice of c is critical. Suppose x_n converges to x_r. Then, the limit of the above equation is:

$$x_r = x_r - c \cdot f(x_r)$$

This gives $f(x_r) = 0$! Just what we want. But, is there a *perfect* value for c?

Consider the *linear equation* $f(x) = ax - b$. It has a zero at $x_r = b/a$.
Use the iteration $x_{n+1} = x_n - c(ax_n - b)$. (Early computers could not divide. They used such an iteration).
Subtracting x_r from both sides:
$x_{n+1} - x_r = x_n - x_r - c(ax_n - b)$, Notice that $x_n - x_r = e_n$, the error in step n.
OR $e_{n+1} = (1 - c \cdot a)e_n$, So at every step the error is multiplied by:
$(1 - c \cdot a)$, which is F'.
The error goes to zero **IF** $|F'| < 1$. *i.e.* the absolute value $|1 - c \cdot a|$ decides everything.
The *Perfect Choice* for c is $1/a$.
Then in one iteration, we have the exact answer: $x_1 = x_0 - (1/a)(ax_0 - b)$ This is a linear equation - does not need calculus.
Now, for a more general f:
$x_{n+1} - x_r = x_n - x_r - c(f(x_n) - f(x_r))$. Here we are going to replace: $(f(x_n) - f(x_r)) = A(x_n - x_r)$. (here $A = \dfrac{df}{dx}|_{x=x_r}$)
$x_{n+1} - x_r \approx (1 - cA)(x_n - x_r)$. OR $e_{n+1} = (1 - cA)e_n$ The error equation.
Once again the error will go to zero in the limit, **IF** $|1 = cA| < 1$.
So now the *Perfect Choice* for c is

$$1/A = \frac{1}{\frac{df}{dx}}|_{x=x_r}$$

Problem We do not yet know x_r.
However, we overcome that using $c = 1/f'(x_n)$. Then:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

This is Newton's Method.

- **Ex. 1a** Solve $f(x) = 2x - \cos x = 0$ using iterations with different c's. Use $x_0 = 0.5$. Use $c = 1,\ c = 1/f'(x_0)$ and $c = 1/f'(x_n)$. Print $x_1, x_2 \ldots$. side-by-side columns (use 7 digits after the decimal) and use a **tolerance** of $1e-7$.

(Using $c = 1/f'(x_0)$ is called modified Newton's method).

- **Ex. 1b** Using the x_r obtained above, print three columns (for the three values of c (above)), of the error: $x_i - x_r$ for $i = 0, 1, 2 \ldots$

(2) Repeat 1(d) and 1(e) for Newton's method. Recall, for Newton's method you need the derivative of the function. For $f(x) = \ln(x+2) - \sqrt{x}$; $df/dx(x) = 1/(x+2) + 1/(2\sqrt{x})$. Pick x_0 carefully.

(3) Newton's method is fast, but prone to problems like division by zero. We examine how Newton's method fails: For this, solve $f(x) = \tanh(x) = 0$ (hyperbolic Tan) *(i)* Start with an initial guess $x_0 = 1.08$. Plot the tangent (and the function at each iteration of Newton's method. *(ii)* Repeat with an initial guess $x_0 = 1.09$.
Note: $\dfrac{d(\tanh(x))}{dx} = 1 - \tanh(x)^2$.
For this part, write a function to plot the curve and the tangent (like the one shown below), that you can call from your program, After each call to `plot_line`, you will need a command `input("Hit Enter to Continue")` for the program to move to the next iteration.

```
  def plot_line(f, xn, fxn, slope):
# Plot both f(x) and the tangent
     xf = linspace(-2,2,100)
     yf = f(xf)
     xt = linspace(xn-2, xn+2,10)
     yt = slope*xt + (fxn - slope*xn) # Straight line:  ax + b
     plt.figure()
     plt.plot(xt, yt, 'r-', xf, yf, 'b-')
     plt.grid('on')
     plt.xlabel('x')
     plt.ylabel('f(x)')
     plt.show()
```

Newton's Method: Set max number of iterations (around 40). Compute $f_0 = f(x_0)$
Compute $f^{'}(x_0)$, check that it is non-zero (if $f^{'}(x_k)$ at any stage is zero, print an error message and return None. This will terminate the execution of the function.
While $|f_0| >$ tolerance and max iterations not exceeded,
Calculate $x_1 = x_0 - \dfrac{f(x_0)}{f'(x_0)}$
Set $x_0 = x_1, \quad f_0 = f(x_0)$ increase iteration count.
when loop terminates, report x_1 as the approximation and the number of iteratio

Also print the number of function evaluations: (each time you evaluate f or $f^{'} = dfdx$