Homework 3
APPM/MATH 4650 Fall '20 Numerical Analysis

Due date: Friday, September 18, before 5 PM, via Gradescope. **Instructor**: Prof. Becker
Theme: Newton's method and rootfinding

Instructions Collaboration with your fellow students is OK and in fact recommended, although direct copying is not allowed. The internet is allowed for basic tasks (e.g., looking up definitions on wikipedia) but it is not permissible to search for proofs or to *post* requests for help on forums such as `http://math.stackexchange.com/` or to look at solution manuals. Please write down the names of the students that you worked with.

An arbitrary subset of these questions will be graded.

Turn in a PDF (either scanned handwritten work, or typed, or a combination of both) to **Gradescope**, using the link to Gradescope from our Canvas page. Gradescope recommends a few apps for scanning from your phone; see the Gradescope HW submission guide.

We will primarily grade your written work, and computer source code is *not* necessary except for when we *explicitly* ask for it (and you can use any language you want). If not specifically requested as part of a problem, you may include it at the end of your homework if you wish (sometimes the graders might look at it, but not always; it will be a bit easier to give partial credit if you include your code).

Problem 1: Which of the following iterations will converge to the indicated fixed point p (provided x_0 is sufficiently close to p)? If it does converge, give the order of convergence; for linear convergence, give the rate of linear convergence.

a) $x_{n+1} = -16 + 6x_n + \frac{12}{x_n}$, $p = 2$

b) $x_{n+1} = \frac{2}{3}x_n + \frac{1}{x_n^2}$, $p = 3^{1/3}$

c) $x_{n+1} = \frac{12}{1+x_n}$, $p = 3$

Problem 2: **Programming Newton's Method** Program Newton's method in a language of your choice, and make sure you can save the history of all the variables x_n and their values $f(x_n)$. You may include your code as an appendix if you want, but grading of this will be based on the following specific items:

a) We will test that our code is working by using a function with a known root. Using $f(x) = (x-3)\cdot(x-2)$, first program the derivative f'. This is an easy derivative, but for more complex functions, it's easy to either make a math mistake or a typing error when programming the derivative. Hence, we want to check that your derivative is implemented correctly. **Plot** your derivative on the interval $[1, 4]$, and then plot a finite difference approximation of the derivative (using only your function f, *not* f') on the same plot. Ensure that these plots are close enough. For finite difference approximations, using `verb` (Matlab) or `numpy.diff` (Python) are useful.

b) Still using $f(x) = (x-3)\cdot(x-2)$, give a **plot** for the error $|x_n - 3|$ for $x_0 = 10$ using your Newton's method code. **Discuss**: does this error decay to zero, and does it decay at the rate you expect?

c) Now use $f(x) = (x-3)^2$, and give a **plot** for the error $|x_n - 3|$ for $x_0 = 10$ using your Newton's method code. **Discuss**: does this error decay to zero, and does it decay at the rate you expect?

d) Misspecify the derivative for $f(x) = (x-3)^2$. The true derivative is $2(x-3)$, but misspecify it as (i) $3(x-3)$ and (ii) $2(x-3.1)$. For both cases, **describe** what happens to the error.

e) Now use $f(x) = (x-3)^2 + 1$ which does not have a real root. **Discuss**: what happens to Newton's method?

Problem 3: Modified Newton's Method Let $f(x) = (x - 1/3)^5$.

a) Run Newton's method, starting at $x_0 = 0.4$, and include either a list (up to 20 iterations) or a plot of the errors. Does the error decay as expected?

b) Now program Modified Newton's method, i.e., run Newton's method on $\mu(x) = f(x)/f'(x)$ since $\mu(p) = f(p) = 0$ but $\mu'(p) \neq 0$, still starting at $x_0 = 0.4$. Simplify the expression for $\mu(x)$ by hand. Include a list of the error. Is this expected?

c) Now program Modified Newton's method again, but implement $\mu(x)$ explicitly as the ratio of $f(x)$ over $f'(x)$ and use `polyval` (Matlab) or `numpy.polyval` (Python) to evaluate $f(x)$ and $f'(x)$. To find the coefficients of f, you can use `poly` (Matlab) or `numpy.poly` (Python) and specify that the root is at $1/3$, and to find the coefficients of the derivative, you can use `polyder` (Matlab) or `numpy.polyder` (Python). Include a list of the first 20 error terms. Is this expected?

d) Repeat (b) and (c) for the polynomial $f(x) = (x - 1/2)^5$. Comment on any differences, and speculate about why they may arise.

Problem 4: Word problem Salma is a dog lover and wants to own dogs. Her happiness level for having d dogs is given by the function

$$F(d) = \underbrace{d^2}_{\text{joy of having dogs}} - \underbrace{.5\left(e^d - 1\right)}_{\substack{\text{unpleasantness of} \\ \text{picking up dog waste}}} .$$

How many dogs should she own in order to maximize her happiness? (Assume that fractional amounts of dogs are allowed, e.g., either counting small dogs as part of a whole dog, or by sharing dog ownership with a friend). Specifically, show how to convert this into a root-finding problem, solve it with your own root-finding implementation (include some output from every iteration), and report the final answer.

Tips for exporting jupyter notebook code to a PDF: try `nbconvert` which requires `pandoc`. You can do this on Colab, following the instructions here (but note that you may need to add a backslash before any white space when you run commands, e.g., change a command like
`!cp drive/My Drive/Colab Notebooks/Untitled.ipynb ./` to
`!cp drive/My\ Drive/Colab\ Notebooks/Untitled.ipynb ./`).

Note that if you include latex in the jupyter notebook, when you run `nbconvert`, you cannot have any whitespace near the $ symbols for math due to a requirement of `pandoc` (see here). So, `$ f(x) = 3x^2 $` will not work, but `$f(x) = 3x^2$` will be OK.