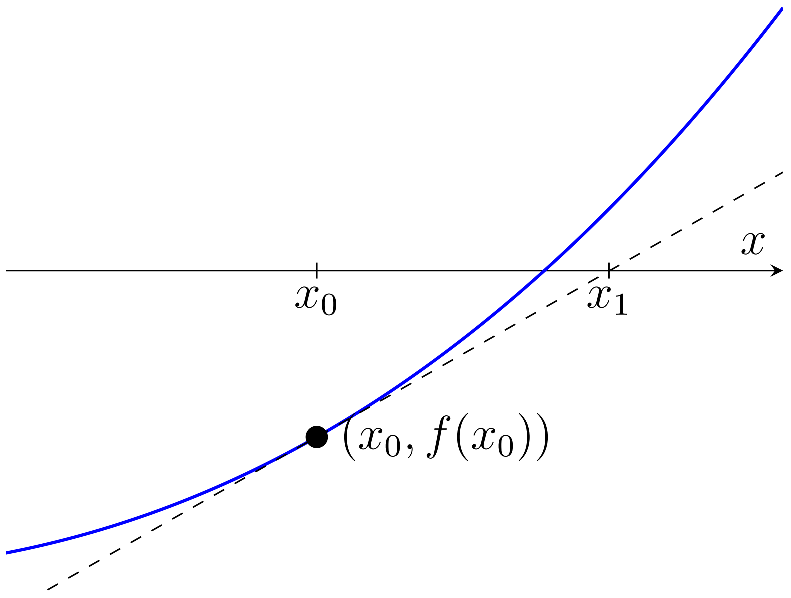
4330 Assignment 9 *

June 4, 2020

Write your code for the following problems in a single file named:

$$\boxed{\texttt{hw9-}\textit{lastname}\texttt{.py}}$$

Recall (from Calculus I) Newton's method for approximating a zero of 'nice' function. Suppose that $f(x)$ is differentiable on $\mathbb{R}$ and x_0 is an initial guess for a zero of f. Geometrically, the idea behind Newton's method is to use a linear approximation of f at the point $(x_0, f(x_0))$ and find the zero of that linear approximation. This gives a new approximation x_1 for a zero of f.



Under suitable conditions, x_1 will usually be a better approximation than x_0, and this process can be repeated until an approximation of desired accuracy is achieved (or until it becomes obvious that the produces sequence is diverging). Working out the resulting formulas gives a sequence of successive approximations determined by

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

Clearly, this is invalid if $f'(x_n) = 0$, and on a computer we actually need to be concerned with $f'(x_n)$ being too close to zero.

---

(1) (10 points) Write a function `Newton(F, dF, x0, epsilon)`, which implements Newton's method. The parameters to this function are as follows:
`F` : A Python function which takes a single floating point number `x` and returns $f(x)$, for some function f to be determined later.
`dF` : As above, except this function evaluates and returns the derivative $f'(x)$ of the same function.
`x0` : An initial point for Newton's method.
`epsilon`: A tolerance; the function should return the first number x_n in the sequence for which $|f(x_n)| < \epsilon$. Or, if the denominator ever satisfies $|f'(x_n)| < \epsilon$, the function should print an error message and return nothing.

Newton's method does not always converge; you should construct this function so that if it has not succeeded after 25 iterations, it will print a failure message and return nothing.

You can test your function with the following snippet; you should be able explicitly determine the zeros of this function, for confirmation that it's working as intended.

```python
def f1(x):
    return x**2 - 5

def df1(x):
    return 2*x

x = 2
xn = Newton(f1, df1, x, 1e-10)
if type(xn) is float:
    print "f1(%1.8f) = %1.8f" % (xn, f1(xn))
```

---

(2) (10 points) Use your function `Newton()` to find an approximate zero of the polynomial $h(x) = x^5 - x^4 + x^2 + x - 1$. Use a tolerance of `1e-10`.

---

(3) (10 points) Find an approximate solution to the equation $e^{-x^2} = \sin x$ by rewriting it in the form $g(x) = 0$, and creating an appropriate pair of Python functions `g(x)` and `dg(x)` which compute $g(x)$ and $g'(x)$ respectively. Use a tolerance of `1e-10`, and print the resulting approximate solution to the eighth decimal place.