Exercise 3: Answer guide

1 Newton Iteration: Algebraic and Geometric Motivation

1. (a) $f(x_{k+1}) = f(x_k) + f'(x_k)(x_{k+1} - x_k) + \frac{f''(x_k)}{2!}(x_{k+1} - x_k)^2 + \frac{f'''(x_k)}{3!}(x_{k+1} - x_k)^3 + \ldots$
   (b) $f(x_{k+1}) \approx f(x_k) + f'(x_k)(x_{k+1} - x_k)$
   (c) $0 = f(x_k) + f'(x_k)(x_{k+1} - x_k)$
   (d) $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$.

2. (a)

$$\begin{aligned} y &= mx + c \\ f(x_k) &= f'(x_k)x_k + c \\ c &= f(x_k) - f'(x_k)x_k \\ y &= f'(x_k)x_{k+1} + f(x_k) - f'(x_k)x_k = f'(x_k)(x_{k+1} - x_k) + f(x_k) \end{aligned}$$

   (b) $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$.

2 1d Examples using Newton Iteration

1. Any function along the lines of the below is acceptable:

```
def newton1d(f, df, x0, tol=1e-10, maxit=50):
    xk = x0
    k=0
    while k<maxit:
        k +=1
        dx = -f(x)/df(x)
        xk1 = xk+ dx
        if abs(dx) < tol or abs(f(x)) < tol:
            return x
        xk=xk1+1
    return xk1
```

2. (a) $x = 2$
   (b) It does not converge to the closest root $x = 2$. This is because the derivative

3. (a) Newton's method diverges from this point and moves further from the true solution. This is because the linear approximation that Newton's method uses (see question 1) is very poor at this point – see the slides for a full explanation.
   (b) For this function there is a region around the root that will result in Newton's method converging. (This is not true for any function! Try solve $x^{1/3} = 0$)

4. (a) The solution is $-1/2$, and Newton's method takes one iteration to find it.
   (b) Newton's method will always take one iteration to solve a linear system, as it uses a first order approximation of the Taylor series, which is exact for a linear system.

3 Differentiation as an Oracle

1. $e^{\sin(x)}\cos(x)$ and $e^{\sin(x)}(\cos(x)^2 - \sin(x)$

2. A small step size is necessary to accurately estimate the derivative, however the smaller the step size the large the effect of machine error on the calculations. Thus there is a trade off between using a small step size to get an accurate approximation, and using a large enough step size that the computer doesn't make large errors in calculating the fraction.

3. The formula you should use is: $h = 2\sqrt{\epsilon\frac{|f(x)|}{|f''(x)|}}$, where ϵ is the machine precision.

4. Read off the figure.

5. (Extra) Completing this example should give a plot more similar to the one shown in the lectures. Essentially, a central difference scheme gives a more accurate approximation than a forward (or backward) difference scheme. Thus you are able to (theoretically) use a larger step size and get the same approximation error. This means that you should find a 1) lower minimum error and 2) that this should occur with a larger h.