Real Analysis Final Project

John Harakas

May 1, 2017

(i) $f(a)f(b) < 0 \implies f(a) < 0$ or $f(b) > 0$

(ii) $f'(x)f''(x) > 0 \; \forall x \in [a, b] \implies f'(x) \neq 0$.

If $f''(x) = 0$, then $f'(x)$ changes sign it some point.

Since f is twice differentiable, $f''(x)$ is some constant $k \neq 0$. Then $kf'(x) > 0$.

If $k < 0$ then $f' < 0 \implies f$ is strictly decreasing

If $k > 0$ then $f' > 0 \implies f$ is strictly increasing

(iii) Intermediate Value Theorem: $f(a) < f(c) < f(b)$ or $f(b) < f(c) < f(a)$

If $f(a) < f(b)$ then $f(a) < 0 < f(b)$

If $f(b) < f(a)$ then $f(b) < 0 < f(a)$

Proposition 1. Let I be an open interval in $\mathbb{R}$ and $f : I \to \mathbb{R}$ be a twice differentiable function. Suppose $[a, b] \subset I$ and $f(a)f(b) < 0$. Suppose also that f' does not vanish on $[a, b]$ and $f'(x)f''(x) > 0$ for all $x \in [a, b]$. Then the following hold.

(a) f has a unique zero at some point $c \in (a, b)$.

(b) f is strictly increasing or decreasing on $[a, b]$.

Proof (a). Because f is twice differentiable, f must also be continuous on $[a, b]$. It is given that $f(a)f(b) < 0$ so the function must change signs somewhere on the interval. Since we know that f is continuous on $[a, b]$, by the Intermediate Value Theorem, there exists at least one point $c \in (a, b)$ such that $f(c) = 0$.

Now since f is twice differentiable, f' must be continuous. Since $f'(x)f''(x) > 0$ for all $x \in [a, b]$, f' cannot change signs, otherwise the Intermediate Value Theorem would imply that f' vanishes somewhere on (a, b), which would be a contradiction. Since f' does not change signs, it is always positive or always negative. Therefore f is strictly monotone, which implies that f is injective and c must be unique.

$\square$

Proof of (b). It follows from (a), that since c is unique, f is strictly increasing or decreasing. Because $f'(x) \neq 0$ for all $x \in [a, b]$, then by the Intermediate Value Theorem for Derivatives, it must be that $f'(x) > 0$ or $f'(x) < 0$ for all $x \in [a, b]$. Suppose that $f'(x) > 0$. Then by the Mean Value Theorem, there exists $d \in (a, b)$ such that

$$f'(d) = \frac{f(b) - f(a)}{b - a} > 0$$

Thus $f'(x) > 0$ and is strictly increasing for all $x \in [a, b]$ $\square$

Proposition 2. Consider the recursive sequence defined by:

$$x_0 = b, \qquad x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \qquad \text{for all } n \geq 0.$$

(a) The sequence is well-defined.

(b) The sequence is convergent, and its limit is c.

Proof (a) Suppose $x_n \in [c, b]$. Since f is twice differentiable on $[a, b]$, by Taylor's Theorem there exists some $\xi \in (c, x_n)$ such that

$$f(c) = f(x_n) + f'(x_n)(c - x_n) + \frac{f''(\xi)}{2}(c - x_n)^2$$

Since f' does not vanish and $f(c) = 0$ we have

$$c + \frac{f''(\xi)}{2}(c - x_n)^2 = x_n - \frac{f(x_n)}{f'(x_n)}$$

But $f'(x)f''(x) > 0$ for $x \in [a, b]$, so they have the same signs. This implies

$$0 < \frac{f''(\xi)}{2f'(x_n)}(c - x_n)^2.$$

Then we have

$$c < x_{n+1}$$

We have shown that f' is either positive or negative. Without loss of generality, suppose that f' is positive. Then f must be strictly increasing. Since $c < x_n$ we have

$$0 = f(c) < f(x_n).$$

From there it follows that

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

$\square$

Proof (b) Let $\lim_{n\to\infty} x_n = \alpha$ and $\lim_{n\to\infty} x_{n+1} \in [a, b]$.

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

Since f is twice differentiable, f and f' are both continuous on $[a, b]$, we have

$$\begin{aligned} \alpha &= \alpha - \frac{f(\alpha)}{f'(\alpha)} \\ 0 &= \frac{-f(\alpha)}{f'(\alpha)} \\ \therefore f(\alpha) &= 0 \end{aligned}$$

But then c is the only zero on $[a, b]$, it is unique and $\alpha = c$. $\square$

Example

Consider the function $f : [0, 50] \to \mathbb{R}$ given by

$$f(x) = x^2 + 2x - 4$$

Then we have The function is strictly increasing, $f(0)f(50) < 0$ and $f'(x)f''(x) > 0 \; \forall x \in [0, 50]$. The function converges to its root at $x = 1.2360679775$ in 8 iterations. The column E_r is the percent relative error given by

$$E_{r,n} = \frac{|x_{n+1} - x_n|}{x_n} \times 100$$

Additionally if the true error $\varepsilon_n = (c - x_n)$, the rate of convergence can be viewed as

$$|\varepsilon_{n+1}| = \frac{|f''(\xi)|}{|2f'(x_n)|}\varepsilon_n^2$$

n	x_n	$f(x_n)$	x_{n+1}	E_r
0	20.0000000000	436.0000000000	9.6190476190	107.9207920795
1	9.6190476190	107.7641723354	4.5449498185	111.6425483931
2	4.5449498185	25.7464684895	2.2233356171	104.4203215925
3	2.2233356171	5.3898925002	1.3872618816	60.2679095134
4	1.3872618816	0.6990192912	1.2408558051	11.7987985286
5	1.2408558051	0.0214347392	1.2360730924	0.3869279877
6	1.2360730924	0.0000228743	1.2360679775	0.0004137998
7	1.2360679775	0.0000000000	1.2360679775	0.0000000005
8	1.2360679775	0.0000000000	1.2360679775	0.0000000000

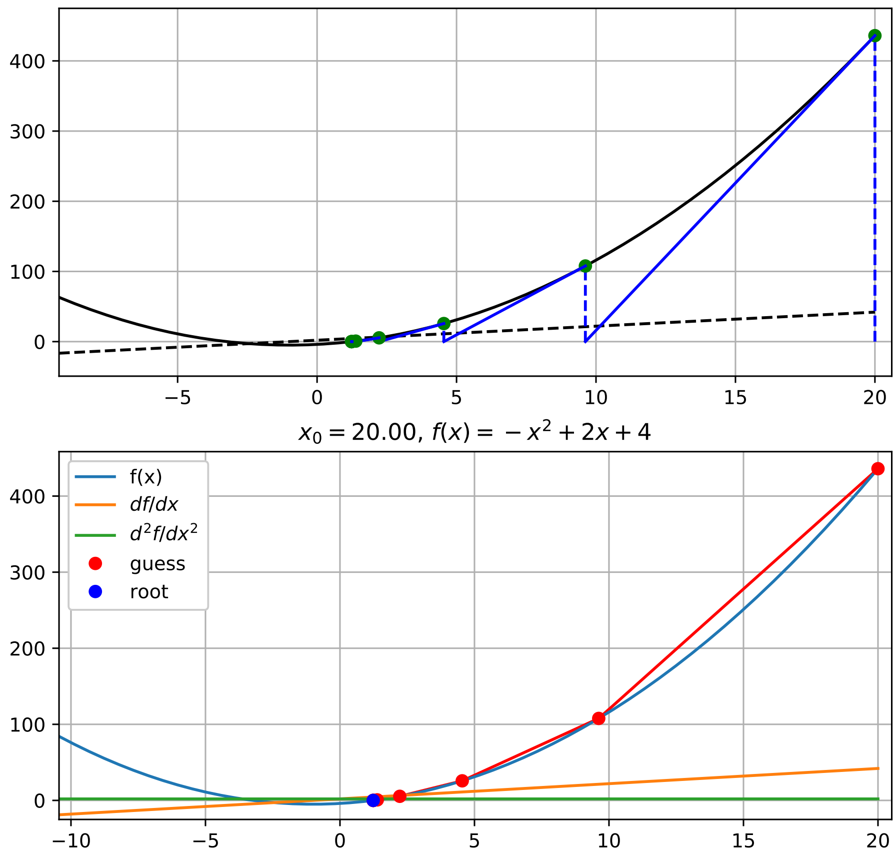

Figure 1

The function $f:[0,50]\to\mathbb{R}$ given by $f(x)=-x^2+2x-4$

n	x_n	$f(x_n)$	x_{n+1}	E_r
0	20.0000000000	−356.0000000000	10.6315789474	88.1188118814
1	10.6315789474	−87.7673130192	6.0753523152	74.9952660480
2	6.0753523152	−20.7592011235	4.0302527995	50.7437031226
3	4.0302527995	−4.1824320290	3.3401400753	20.6611911072
4	3.3401400753	−0.4762555721	3.2383821555	3.1422455687
5	3.2383821555	−0.0103546742	3.2360691738	0.0714750410
6	3.2360691738	−0.0000053499	3.2360679775	0.0000369668
7	3.2360679775	0.0000000000	3.2360679775	0.0000000000

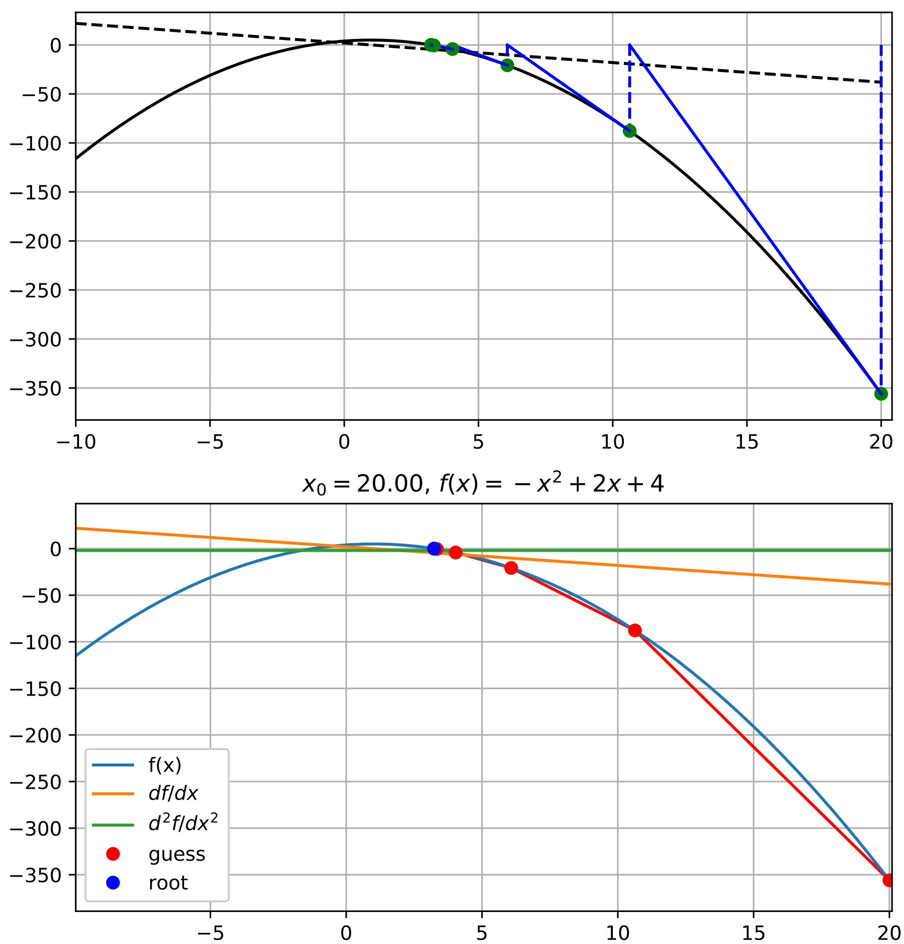

Figure 2

Code

```python
import matplotlib
matplotlib.use('Qt5Agg')
import matplotlib.pyplot as plt
# Numerical package
import numpy as np
# Gonna use quadruple precision to be safe
# Still, 2nd finite difference hits machine epsilon quickly
from numpy import float128


def newton(f,I):

    MAX = 100
    h = float128(1e-6)
    eps = float128(1e-10)
    d = float128(1e-6)

    # 100k points is overkill
    x = np.linspace(*I, 100000, dtype=float128)
    # Fourth order accurate 1st and 2nd finite differences
    df = lambda x: (-f(x + 2*h) + 8*f(x + h) - 8*f(x - h) + f(x-2*h)) / (12 * h)
    d2f = lambda x: (-f(x+2*h) + 16*f(x+h) - 30*f(x) + 16*f(x-h) - f(x-2*h)) / (12*h*h)

    t = np.zeros(MAX,dtype='float128')
    # initial guess
    x0 = 50
    # This array is for the secant method.
    # Should be using preallocated array for newton but I'm not.
    t[0] = x0
    # t[1] = 3
    guess = x0 # keep track, to print
    # next iteration
    xr = x0
    # Error function
    err = lambda x0, x1: (abs(x1-x0) / x1)*100

    # Terminate after 100 iterations
    for k in range(0, MAX-1):
        a = x0
        # This is actually taylor series, but gives same points
```

```
        g = lambda x: f(a) + df(a) * (x - a) + (d2f(a)/2)*(x - a) ** 2
        # Plot each iteration path
        plt.plot([xr, x0], [0, f(x0)], '--b')
        plt.plot(a, g(a), 'og')

        # Newton's method
        xr = x0 - f(x0) / df(x0)

        print("k={0:<10d} xn={1:<8.10f}  f({1:1.10f})={2:<8.10f}".format(k, xr,f(xr)))

        # Secant method
        t[k+1] = t[k] - (d*f(t[k])) /(f(t[k] + d) - f(t[k]))
        plt.plot([xr, x0],[0, f(x0)],'-b')

        # Termination criteria
        if err(x0,xr) < eps:
            t = t[0:k+1]
            break
        x0 = xr
        plt.grid(True)

else:
    print("seems divergent.")

print("initial guess =%1.4f" % guess)
print("f(%1.4f)= %1.4f" % (xr, f(xr)))
plt.plot(x, f(x), 'k')
plt.plot(x, df(x), '--k')
for n in range(1,len(t)-1):
    plt.plot([t[n], t[n-1]], [f(t[n]), f(t[n-1])], '-r')
plt.tight_layout()
plt.figure()
plt.plot(x, f(x), label="f(x)")
plt.plot(x, df(x), label="$df/dx$")
plt.plot(x, d2f(x), label="$d^2f/dx^2$")
plt.plot(t, f(np.asarray(t)), 'or', label="guess")
plt.plot(xr, f(xr), 'ob', label="root")
plt.grid(True)
plt.legend()
plt.title("$x_0=%1.2f$, $f(x)=x^2 + 2x-4$" % guess)
plt.show()
```

```python
def main():
    # f(x) using lambda expressions.
    f = lambda x: x**2 + 2*x - 4
    a = 0
    b = 50
    I = [a, b]
    newton(f, I)

if __name__ == '__main__':
    main()
```