Will Firmin

Math 151AH: HW1

Problem 1

a)

In [ ]:
```
import math
import matplotlib.pyplot as plt
print("Calculated:\n",1 - math.cos(10**-3))
print("True:\n","4.9999995833333472222219742063519620811078710385413014376488... ×10^-7")
```

```
Calculated:
 4.999999583255033e-07
True:
 4.9999995833333472222219742063519620811078710385413014376488... ×10^-7
```

Accurate to 10 digits.

b)

Identity: $1 - \cos(\theta) = 2\sin^2(\frac{\theta}{2})$

c)

In [ ]:
```
print(2*math.sin(10**-3 / 2)**2)
```

```
4.999999583333347e-07
```

Accurate to all 16 digits; this is much better than before.

Problem 2

a)

By Taylor's Theorem,

$$\sin(\frac{\pi}{2} + \delta) = \sin(\frac{\pi}{2}) - \frac{1}{2}\sin(\zeta)\delta^2$$

for $\zeta \in (\frac{\pi}{2}, \frac{\pi}{2} + \delta)$. Therefore

$$\begin{aligned}
|\sin(\frac{\pi}{2}) - \sin(\frac{\pi}{2} + \delta)| &< 10^{-\beta} \\
\Rightarrow \frac{1}{2}\sin(\zeta)\delta^2 &< 10^{-\beta} \\
\Rightarrow \delta^2 &< \frac{2 * 10^{-\beta}}{\sin(\zeta)} \\
\Rightarrow \delta^2 &< \frac{2 * 10^{-\beta}}{\sin(\frac{\pi}{2} + 1)} \\
\Rightarrow \delta &< \sqrt{\frac{2 * 10^{-\beta}}{\sin(\frac{\pi}{2} + 1)}}
\end{aligned}$$

b)

In [ ]:
```
fun = lambda x: math.sin(math.pi / 2 + x)

d = 0.1
lowest_not = 1
highest_is = 0
for i in range(1000):
  if fun(d) == 1:
    highest_is = d
  else:
    lowest_not = d
  d = (highest_is + lowest_not)/2

print(f"delta = {d}")
```

```
delta = 1.0536712280462268e-08
```

Problem 3

a)

$$\begin{aligned}
f(x) &= 1.01e^{4x} - 4.62e^{3x} - 3.11e^{2x} + 12.2e^{x} - 1.99 \\
f(1.53) &= 1.01(4.62^4) - 4.62(4.62^3) - 3.11(4.62^2) + 12.2(4.62) - 1.99 \\
&= 1.01(21.3^2) - (21.3^2) - 3.11(21.3) + 56.4 - 1.99 \\
&= 1.01(454) - 454 - 66.2 + 56.4 - 1.99 \\
&= 459 - 454 - 66.2 + 56.4 - 1.99 \\
&= 5 - 66.2 + 56.4 - 1.99 \\
&= -61.2 + 56.4 - 1.99 \\
&= -4.8 - 1.99 \\
&= -6.79
\end{aligned}$$

b)

$$f(x)=e^x(e^x(e^x(1.01e^x-4.62)-3.11)+12.2)-1.99$$

c)

$$\begin{aligned}f(1.53)&=4.62(4.62(4.62(1.01(4.62)-4.62)-3.11)+12.2)-1.99\\&=4.62(4.62(4.62(0.05)-3.11)+12.2)-1.99\\&=4.62(4.62(-2.88)+12.2)-1.99\\&=4.62(-1.1)-1.99\\&=-7.07\end{aligned}$$

d)

The nested form has greater accuracy. Its error is almost half that of the original form.

Problem 4

a)

$$\begin{aligned}e_k&=\mathrm{fl}(A_k)-A_k\\&=(\mathrm{fl}(A_{k-1})+a_k)(1+\delta_k)-A_k\\&=(1+\delta_k)\mathrm{fl}(A_{k-1})+(1+\delta_k)a_k-A_k\\&=(1+\delta_k)(e_{k-1}+A_{k-1})+(1+\delta_k)a_k-A_k\\&=(1+\delta_k)e_{k-1}+(1+\delta_k)(A_{k-1}+a_k)-A_k\\&=(1+\delta_k)e_{k-1}+(1+\delta_k)A_k-A_k\\&=(1+\delta_k)e_{k-1}+\delta_kA_k\end{aligned}$$

b)

$$\begin{aligned}e_k-e_{k-1}&=\delta_ke_{k-1}+\delta_kA_k\\\Rightarrow\sum_{k=2}^n(e_k-e_{k-1})&=\sum_{k=2}^n(\delta_ke_{k-1}+\delta_kA_k)\\\Rightarrow e_n-e_1&=\sum_{k=2}^n\delta_kA_k+\sum_{k=2}^n\delta_ke_{k-1}\\\Rightarrow e_n&=\sum_{k=2}^n\delta_kA_k+\sum_{k=2}^n\delta_ke_{k-1}+e_1\\&=\sum_{k=2}^n\delta_kA_k+\sum_{k=2}^n\delta_ke_{k-1}\end{aligned}$$

All that is left to show is $\sum_{k=2}^n\delta_ke_{k-1}=O(\epsilon_M^2)$:

$$\begin{aligned}|\sum_{k=2}^n\delta_ke_{k-1}|&\le\sum_{k=2}^n|\delta_ke_{k-1}|\\&\le\sum_{k=2}^n\epsilon_M|e_{k-1}|\\&\le\sum_{k=2}^n\epsilon_M^2|A_{k-1}|&&(\text{Since }\frac{|e_{k-1}|}{|A_{k-1}|}\le\epsilon_M^2)\\&\le\sum_{k=2}^n\epsilon_M^2A&&\text{for }A=\max_k\{|A_k|\}\\&=A(n-1)\epsilon_M^2\\&=O(\epsilon_M^2)\end{aligned}$$

It is preferable to start with the smallest numbers so A_k stays smaller longer. This reduces the effect of each δ_k on e_n. Starting with the larger numbers would keep A_k large and increase the effect of each δ_k.

Problem 5

By Taylor's Theorem we have that

$$\begin{aligned}f(x)&=\frac{1}{t!}f^{(t)}(\zeta_0)(x-x^*)^t&&\text{for }\zeta_0\in(x^*,x)\\\text{and }f'(x)&=\frac{1}{(t-1)!}f^{(t)}(\zeta_1)(x-x^*)^{t-1}&&\text{for }\zeta_1\in(x^*,x)\end{aligned}$$

Then for Newton's method with $x_n=x_{n-1}-\frac{f(x_{n-1})}{f'(x_{n-1})}$,

$$\begin{aligned}|x_n-x^*|&=|f(x_{n-1})-x^*|\\&=|x_{n-1}-\frac{1}{t}\frac{f^{(t)}(\zeta_{0n})}{f^{(t)}(\zeta_{1n})}(x_{n-1}-x^*)-x^*|&&\text{for }\zeta_{0n},\zeta_{1n}\in(x^*,x_{n-1})\\&=|x_{n-1}-x^*||1-\frac{1}{t}\frac{f^{(t)}(\zeta_{0n})}{f^{(t)}(\zeta_{1n})}|\\\Rightarrow\lim_{n\to\infty}\frac{|x_n-x^*|}{|x_{n-1}-x^*|}&=\lim_{n\to\infty}|1-\frac{1}{t}\frac{f^{(t)}(\zeta_{0n})}{f^{(t)}(\zeta_{1n})}|\\&=|1-\frac{1}{t}\frac{f^{(t)}(x^*)}{f^{(t)}(x^*)}|&&\text{since }\zeta_{0n},\zeta_{1n}\to x^*\text{ by the squeeze theorem}&&=1-\frac{1}{t}\end{aligned}$$

Problem 6

$$
\begin{aligned}
g(x) &= x - \frac{f(x)^2}{f(x+f(x)) - f(x)} \\
\Rightarrow g'(x) &= 1 - \frac{2f(x)f'(x)}{f(x+f(x)) - f(x)} + \frac{f(x)^2[f'(x+f(x))(1+f'(x)) - f'(x)]}{[f(x+f(x)) - f(x)]^2} \qquad \text{By L'Hopital's rule,} \\
\lim_{x\to x^*} g'(x) &= 1 - \lim_{x\to x^*} \frac{2f'(x)^2 + 2f(x)f''(x)}{f'(x+f(x))(1+f'(x)) - f'(x)} + \lim_{x\to x^*} \frac{2f(x)f'(x)[f'(x+f(x))(1+f'(x)) - f'(x)] + f(x)^2[f''(x+f(x))(1+f'(x))^2 + f''(x)f'(x+f(x)) - f''(x)]}{2[f(x+f(x)) - f(x)][f'(x+f(x))(1+f'(x)) - f'(x)]} \\
&= 1 - 2 + \lim_{x\to x^*} \frac{f(x)f'(x)}{f(x+f(x)) - f(x)} + \lim_{x\to x^*} \frac{f(x)^2}{f(x+f(x)) - f(x)} \lim_{x\to x^*} \frac{f''(x+f(x))(1+f'(x))^2 + f''(x)f'(x+f(x)) - f''(x)}{2f'(x+f(x))(1+f'(x)) - f'(x)} \\
&= -1 + \lim_{x\to x^*} \frac{f'(x)^2 + f(x)f''(x)}{f'(f(x)+x)(1+f'(x)) - f'(x)} + \lim_{x\to x^*} [\frac{2f(x)f'(x)}{f'(x+f(x))(1+f'(x)) - f'(x)}] \frac{f''(x^*)(1+f'(x^*))^2 + f''(x^*)f'(x^*) - f''(x^*)}{2f'(x^*)(1+f'(x^*)) - f'(x^*)} \\
&= -1 + 1 + 0[\frac{f''(x^*)(1+f'(x^*))^2 + f''(x^*)f'(x^*) - f''(x^*)}{2f'(x^*)(1+f'(x^*)) - f'(x^*)}] \\
&= 0 \\
\Rightarrow g'(x^*) &= 0
\end{aligned}
$$

Assuming $g(x) \not\equiv c$ for some $c \in \mathbb{R}$, some higher order derivative of g must be nonzero: there must exist $s > 1$ s.t. $f^{(s)}(x) \not\equiv 0$. Thus by the higher order convergence theorem, $g(x)$ must converge at least quadratically.

Problem 7

$$
\begin{aligned}
g'(x) &= 1 - \frac{(f'(x))^2 - f(x)f''(x)}{(f'(x))^2} - \frac{2f'(x)f'''(x) - 2f''(x)f''(x)}{4f''(x)f''(x)}[\frac{f(x)}{f'(x)}]^2 - \frac{f''(x)}{f'(x)}\frac{f(x)}{f'(x)}\frac{f(x)^2 - f(x)f''(x)}{f'(x)^2} \\
&= \frac{f(x)f''(x)}{f'(x)^2} + \frac{f(x)^2}{2f'(x)^2} - \frac{f'''(x)f(x)^2}{2f'(x)f''(x)^2} - \frac{f''(x)f(x)}{f'(x)^2} + \frac{f''(x)^2 f(x)^2}{f'(x)^4} \\
\Rightarrow g'(x^*) &= 0 \\
g'(x) &= f(x)[\frac{f(x)}{2f'(x)^2} - \frac{f'''(x)f(x)}{2f'(x)f''(x)^2} + \frac{f''(x)^2 f(x)}{f'(x)^4}] \\
\Rightarrow g''(x) &= f'(x)\phi(x) + f(x)\phi'(x) \qquad \text{for } \phi = \frac{f(x)}{2f'(x)^2} - \frac{f'''(x)f(x)}{2f'(x)f''(x)^2} + \frac{f''(x)^2 f(x)}{f'(x)^4} \\
\Rightarrow g''(x^*) &= f'(x^*)0 + 0\phi'(x^*) \\
&= 0
\end{aligned}
$$

Note that by the quotient rule, each term in $\phi'(x^*)$ will contain nonzero values.

By Taylor's Theorem,

$$
\begin{aligned}
g(x_n) &= g(x^*) + g'(x^*)(x_n - x^*) + \frac{1}{2}g''(x^*)(x_n - x^*)^2 + \frac{1}{6}g'''(\zeta)(x_n - x^*)^3 \qquad \text{for } \zeta \in (x^*, x_n) \\
\Rightarrow x_{n+1} &= x^* + \frac{1}{6}g'''(\zeta)(x_n - x^*)^3 \\
\Rightarrow |x_{n+1} - x^*| &= \frac{1}{6}|g'''(\zeta)||x_n - x^*|^3 \\
&\leq \frac{M}{6}|x_n - x^*|^3 \qquad \text{for bound } M \text{ of } g'''(x) \text{ on } [x^*, x_n] \\
\Rightarrow \lim_{n\to\infty} \frac{|x_{n+1} - x^*|}{|x_n - x^*|^3} &\leq \frac{M}{6}
\end{aligned}
$$

Problem 8

In [ ]:

```
x0 = 0.75
f = lambda x: (math.e**x / 3)**0.5
x = [x0]
for i in range(6):
  x.append(f(x[-1]))

aitken = lambda i: x[i] - (x[i+1] - x[i])**2 / (x[i+2] - 2*x[i+1] + x[i])
x2 = [aitken(i) for i in range(5)]
for a in x2:
  print(a)
```

```
0.9078585524534872
0.9095675068671815
0.9099168937459944
0.9099888384861006
0.910003697648647
```

Problem 9

In [ ]:

```
def FPI(f, x0, alpha, N=100):
  # need to figure out alpha
  g = lambda x: x - alpha*f(x)
  x = [x0]
  for i in range(N):
    try:
      x.append(g(x[-1]))
    except: pass
  return x

def newton(f, df, x0, N=100):
  x = [x0]
  for i in range(N):
    x.append(x[-1] - f(x[-1]) / df(x[-1]))
  return x

def secant(f, x0, N=100):
  x = [x0, f(x0)]
  for i in range(N):
    try:
      x.append(x[-1] - f(x[-1])*(x[-1]-x[-2]) / (f(x[-1]) - f(x[-2])))
```

```python
        except ZeroDivisionError: pass
    return x

# def steffensen(f, x0, N=100):
#   x = x0
#   for i in range(N):
#     try:
#       xi0 = x
#       xi1 = f(x)
#       xi2 = f(xi1)
#       #print(i,"\n",x,"\n",xi1,"\n",xi2)
#       x = xi0 - (xi1 - xi0)**2 / (xi2 - 2*xi1 + x)
#     except ZeroDivisionError: pass
#   return x

def steffensen(f, x0, N=100):
  x = [x0]
  for i in range(N):
    try:
      x.append(x[-1] - f(x[-1])**2 / (f(x[-1] + f(x[-1])) - f(x[-1])))
    except ZeroDivisionError: pass
  return x
```

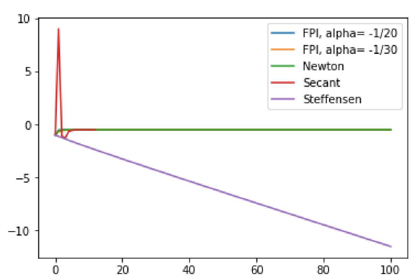
a)

In [ ]:

```python
F = lambda x: 10*x**2 - 4*x - 5
dF = lambda x: 20*x - 4
#1 - alpha*(20*x + 4)
#FPI:
fpi1 = FPI(F, -1, -1/20)
fpi2 = FPI(F, -1, -1/30)
#Newton:
newt = newton(F, dF, -1)
#secant:
sec = secant(F,-1)
#steffensen:
stef = steffensen(F,-1)
L = lambda X: list(range(len(X)))

for method, result in zip(["FPI, alpha=-1/20","FPI, alpha=-1/30","Newton's Method","Secant Method","Steffensen's Method"],[fpi1, fpi2, newt, sec, stef]):
    print(f"{method}: {result[-1]}")

plt.plot(L(fpi1),fpi1, label="FPI, alpha= -1/20")
plt.plot(L(fpi2),fpi2, label="FPI, alpha= -1/30")
plt.plot(L(newt),newt,label="Newton")
plt.plot(L(sec),sec, label="Secant")
plt.plot(L(stef),stef, label="Steffensen")
plt.legend()
plt.show
```

```
FPI, alpha=-1/20: -0.5348469228349535
FPI, alpha=-1/30: -0.5348469228349535
Newton's Method: -0.5348469228349534
Secant Method: -0.5348469228349534
Steffensen's Method: -11.517858534230458
```

Out[ ]: <function matplotlib.pyplot.show(*args, **kw)>



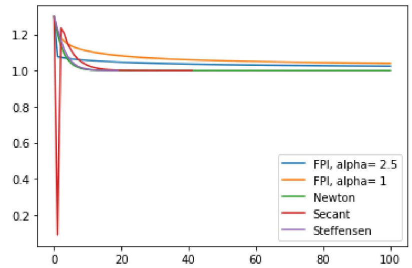
b)

In [ ]:

```python
F = lambda x: x**4 - x**3 - 3*x**2 + 5*x - 2
dF = lambda x: 4*x**3 - 3*x**2 -6*x + 5

#FPI:
fpi1 = FPI(F, 1.3, 2.5)
fpi2 = FPI(F, 1.3, 1)
#Newton:
newt = newton(F, dF, 1.3)
#secant:
sec = secant(F,1.3)
#steffensen:
stef = steffensen(F,1.3)
L = lambda X: list(range(len(X)))

for method, result in zip(["FPI, alpha=2.5","FPI, alpha=1","Newton's Method","Secant Method","Steffensen's Method"],[fpi1, fpi2, newt, sec, stef]):
    print(f"{method}: {result[-1]}")

plt.plot(L(fpi1),fpi1, label="FPI, alpha= 2.5")
plt.plot(L(fpi2),fpi2, label="FPI, alpha= 1")
plt.plot(L(newt),newt, label="Newton")
plt.plot(L(sec),sec, label="Secant")
plt.plot(L(stef),stef, label="Steffensen")
plt.legend()
plt.show
```

```
FPI, alpha=2.5: 1.0242696071414192
FPI, alpha=1: 1.03935015272538
Newton's Method: 1.0000060716137125
Secant Method: 1.0000035002296985
Steffensen's Method: 1.0003166415823173
```

Out[ ]: <function matplotlib.pyplot.show(*args, **kw)>

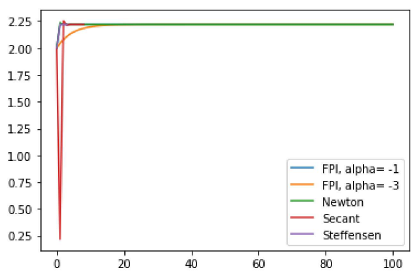


c)

In [ ]:
```python
F = lambda x: math.sin(x) - math.log(x)
dF = lambda x: math.cos(x) - 1/x

#FPI:
fpi1 = FPI(F, 2, -1)
fpi2 = FPI(F, 2, -0.2)
#Newton:
newt = newton(F, dF, 2)
#secant:
sec = secant(F,2)
#steffensen:
stef = steffensen(F,2)

for method, result in zip(["FPI, alpha=-1","FPI, alpha=-3","Newton's Method","Secant Method","Steffensen's Method"],[fpi1, fpi2, newt, sec, stef]):
    print(f"{method}: {result[-1]}")

plt.plot(L(fpi1),fpi1, label="FPI, alpha= -1")
plt.plot(L(fpi2),fpi2, label="FPI, alpha= -3")
plt.plot(L(newt),newt, label="Newton")
plt.plot(L(sec),sec, label="Secant")
plt.plot(L(stef),stef, label="Steffensen")
plt.legend()
plt.show
```

```
FPI, alpha=-1: 2.219107148913746
FPI, alpha=-3: 2.2191071489014393
Newton's Method: 2.219107148913746
Secant Method: 2.219107148913746
Steffensen's Method: 2.219107148913746
```

Out[ ]:
```
<function matplotlib.pyplot.show(*args, **kw)>
```



Problem 10

a)

In [ ]:
```python
F = lambda x: 10*x**2 - 4*x + 5
dF = lambda x: 20*x - 4
tol = 10**-5
a1, a2, a3 = 1/100, 1/20, 2

def FPI2(f, x0, alpha, tol=tol):
  g = lambda x: x - alpha*f(x)
  x = x0
  count = 0
  while abs(f(x)) >= tol and count < 100:
    x = g(x)
    count += 1
  print(f"Alpha = {alpha}\nx* = {x}\nCompleted in {count} iterations.\n")


FPI2(dF, 0, a1)
FPI2(dF, 0, a2)
FPI2(dF, 0, a3)
```

```
Alpha = 0.01
x* = 0.1999995210951435
Completed in 58 iterations.

Alpha = 0.05
x* = 0.2
Completed in 1 iterations.

Alpha = 2
x* = -255558716510693824674537449010434911861914569149854421106803114134089547153738625028717782550797070641885149999885800538671581040229223258427250623004869423200
Completed in 100 iterations.
```

b)

In [ ]:
```python
d2F = lambda x: 20
```

```python
def newton2(f, df, x0, tol=tol):
  x = x0
  count = 0
  while abs(f(x)) >= tol and count < 100:
    x = x - f(x)/df(x)
    count += 1
  return (x,count)

x, count = newton2(dF,d2F,0)
print(f"x* = {x}\nCompleted in {count} iterations.")
```

```
x* = 0.2
Completed in 1 iterations.
```