1Jozsef Morrisseyjtmorri2@illinois.edu

Problem 1:

a.)

$$f_0 = f_0$$

$$f_1 = f(x_0 + h) = f_0 + hf_0' + \frac{h^2}{2!}f_0'' + \frac{h^3}{3!}f_0'''$$

$$f_2 = f(x_0 + 2h) = f_0 + 2hf_0' + \frac{4h^2}{2!}f_0'' + \frac{8h^3}{3!}f_0'''$$

$$f_2 - 2f_1 = -f_0 + h^2 f_0'' + \frac{6h^3}{3!}f_0'''$$

$$f_0'' = \frac{f_2 - 2f_1 + f_0}{h^2}$$

b.)

$$h$$

c.)

$$\frac{4\epsilon}{h}$$

ed.)

```
from __future__ import division
import matplotlib.pyplot as plt
import math

# helper functions:

def f(h):
    return (math.sin(.7+2*h)-2*math.sin(0.7+h)+math.sin(0.7))/(h*h)
```

```
def g(x):
    return -math.sin(.7+x)

def gangster(x):
    return x

def prostitute(x):
    return 4*math.pow(2, -53)/(x*x)

# setup domain:
x0 = 0.7
xmax = 1
steps = 25
h = [math.pow(2, -x) for x in range(xmax, xmax*steps)]

# function evaluations:
f_vals = [gangster(x) for x in h]
g_vals = [prostitute(x) for x in h]
difference = [abs(f(x)-g(x)) for x in h]

# plot:
# (uncomment these for log scale)

sub = plt.subplot(111)
sub.set_xscale('log')
sub.set_yscale('log')

plt.plot(h, f_vals, h, g_vals, h, difference)
plt.show()
```

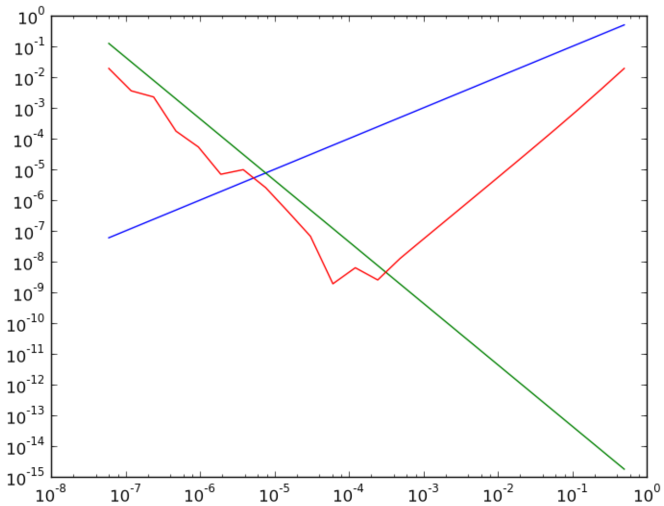

Figure 1: Error Plot

f.)

$$4^{1/3} * \epsilon^{1/3}$$

g.)

$$4^{1/3} * \epsilon^{1/3} = h$$

Problem 2:

a.)

Table 1: Finite-difference errors when using $f_0, ..., f_3$ for $f = sin(x)$

Derivative	Truncation	Round-Off	h_{min}	min.err.
f_0'	$\approx h^3$	$\approx \varepsilon_{mach}/h$	$E_{mach}^{1/4}$	$E_{mach}^{3/4}$
f_0''	$\approx h^2$	$\approx \varepsilon_{mach}/h^2$	$E_{mach}^{1/4}$	$E_{mach}^{1/2}$
f_0'''	$\approx h$	$\approx \varepsilon_{mach}/h^3$	$E_{mach}^{1/4}$	$E_{mach}^{1/4}$

bc.)

```
from __future__ import division
import matplotlib.pyplot as plt
import math

# helper functions:


def f(h):
    return (math.sin(.7+3*h)-3*math.sin(.7+2*h)+3*math.sin(.7+h)-math.sin(.7))/(h*h*h

def g(x):
    return -math.cos(.7+x)

def gangster(x):
return x

def prostitute(x):
return math.pow(2, -53)/(x*x*x)

# setup domain:
x0 = 0.7
xmax = 1
steps = 15
h = [math.pow(2, -x) for x in range(xmax, xmax*steps)]

# function evaluations:
f_vals = [gangster(x) for x in h]
g_vals = [prostitute(x) for x in h]
difference = [abs(f(x)-g(x)) for x in h]

# plot:
# (uncomment these for log scale)

sub = plt.subplot(111)
sub.set_xscale('log')
sub.set_yscale('log')

plt.plot(h, f_vals, h, g_vals, h, difference)
```

```
plt.show()
```

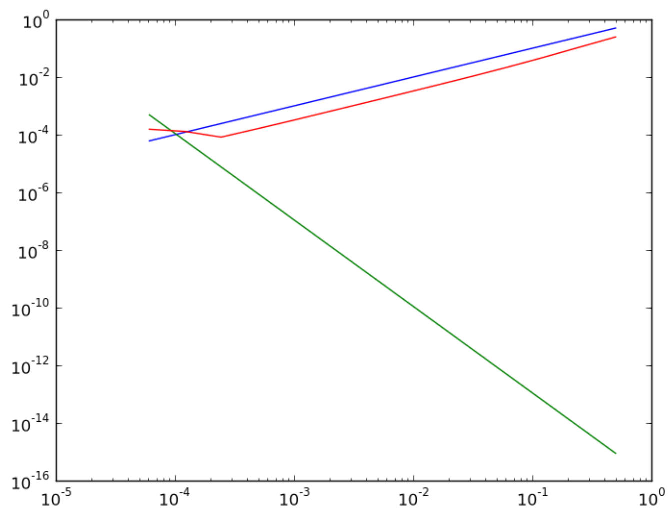

Figure 2: Error Plot

d.)

```
To evaluate high order derivatives using this method more points need to be
evaluated and an n*n matrix must be solved to determine the coefficients
of the formula that cancels out all other derivatives.

To improve the estamates use more points to cancel more terms.
```

Problem 3:

a.)

$$\varepsilon_{mach} = 2^{-52} \approx 1.1 * 10^{-16}$$

b.)

```
from __future__ import division
```

```
import matplotlib.pyplot as plt
import math


y=.1
nprev=0
#i chose to do 8 iterations of increasingly smaller division values to clearly
#illustrate that the error is approaching 2^(-53)
for x in range (0, 8):
n=1
while (1+n>1): #testing if the error has an impact on the value.
nprev=n
n=n/(1+y)  #decreasing n
y=y/10         #decreasing the divisor
print (nprev)

Output:
1.15828708536e-16 = $2^{-52.938856643719}$
1.11042183872e-16 = $2^{-52.999741671337}$
1.11040714506e-16 = $2^{-52.999760761929}$
1.11026452246e-16 = $2^{-52.999946076058}$
1.11022774654e-16 = $2^{-52.999993864054}$
1.11022406945e-16 = $2^{-52.999998642288}$
1.11022311907e-16 = $2^{-52.999999877272}$
1.11022303198e-16 = $2^{-52.999999990443}$
```

c.)

$$2^{10} = 1 * 10^3$$

$$2^{20} = 1 * 10^6$$

$$2^{30} = 1 * 10^9$$

$$2^{-53} = 1 * 10^{-16}$$

d.)

$$k = log_2(10^6) = 20$$