EGR 103L – Fall 2019

# Structured Programming I

***Marcus Deans (md374)***
***Lab Section 2, Tuesdays 11:45-2:35***
***29 September 2019***

I understand and have adhered to all the tenets of the Duke Community Standard in completing every part of this assignment. I understand that a violation of any part of the Standard on any part of this assignment can result in failure of this assignment, failure of this course, and/or suspension from Duke University.

## Contents

## List of Figures

# 1 Sinusoids

The $A$ value represents the amplitude of the cosine function. The $A$ value of 2 results in a amplitude twice that of the function with a $A$ value of 1; in this case, the amplitude is precisely two. Conversely, a $A$ value $< 1$ would result in a decreased amplitude. The $\omega$ value affects the period of graph by changing the horizontal extension/compression of the graph. As the $\omega$ value increases, the period will decrease as related by the equation $period = ((2\pi)/\omega)$. The $\phi$ value represents the phase shift, or horizontal translation, of the cosine function, thus shifting the function either in the $+x$ or $-x$ direction or ”left or right”. Notably, a positive $\phi$ value will result into a shift to the left, thus perceived as moving in the negative direction, unlike other function transformations.

# 2 P&E 4.48

```
(3, 7):
       1   2   3   4   5   6   7
   1   1   2   3   4   5   6   7
   2   2   4   6   8  10  12  14
   3   3   6   9  12  15  18  21

(7, 3):
       1   2   3
   1   1   2   3
   2   2   4   6
   3   3   6   9
   4   4   8  12
   5   5  10  15
   6   6  12  18
   7   7  14  21

(12, 12):
       1   2   3   4   5   6   7   8   9  10  11  12
   1   1   2   3   4   5   6   7   8   9  10  11  12
   2   2   4   6   8  10  12  14  16  18  20  22  24
   3   3   6   9  12  15  18  21  24  27  30  33  36
   4   4   8  12  16  20  24  28  32  36  40  44  48
   5   5  10  15  20  25  30  35  40  45  50  55  60
   6   6  12  18  24  30  36  42  48  54  60  66  72
   7   7  14  21  28  35  42  49  56  63  70  77  84
   8   8  16  24  32  40  48  56  64  72  80  88  96
   9   9  18  27  36  45  54  63  72  81  90  99 108
  10  10  20  30  40  50  60  70  80  90 100 110 120
  11  11  22  33  44  55  66  77  88  99 110 121 132
  12  12  24  36  48  60  72  84  96 108 120 132 144
```

# 3 P&E 2.38

```
Running tests for user md374
Test 1: passed: digit_sum(        1) returns 1
Test 2: passed: digit_sum(       15) returns 6
Test 3: passed: digit_sum(      354) returns 3
Test 4: passed: digit_sum(     5803) returns 7
Test 5: passed: digit_sum(    85291) returns 7
Test 6: passed: digit_sum(   293439) returns 3
Test 7: passed: digit_sum(  1378324) returns 1
Test 8: passed: digit_sum( 95007602) returns 2
Test 9: passed: digit_sum(960477756) returns 6
```

## 4 Chapra Problem 3.10

- max_pos_disp = 1.9532e+02
- max_pos_disp_loc = 5.7019 ft
- max_neg_disp = =3.2741e+01
- max_neg_disp_loc = 8.7076 ft

## 5 Geometric Progression

```
[1.0, 2.0, 4.0, 8.0, 16.0, 32.0, 64.0]
127.0
[10.0, 5.0, 2.5, 1.25, 0.625, 0.3125, 0.15625]
19.84375
[2.0, 6.0, 18.0, 54.0, 162.0, 486.0]
728.0
All arguments must be single numbers.
-1
-1
All arguments must be single numbers.
-1
-1
All arguments must be single numbers.
-1
-1
All arguments must be positive.
-2
-2
All arguments must be positive.
-2
-2
All arguments must be positive.
-2
-2
Invalid sequence.
-3
-3
Invalid sequence.
-3
-3
```

# A Codes

## A.1 Sinusoidal_.py

```python
# -*- coding: utf-8 -*-
"""
[Sinusoidal]
[Marcus Deans]
[18 September 2019]

I understand and have adhered to all the tenets of the Duke Community Standard
in creating this code.
Signed: [md374]
"""
# %% import libraries
import math as m
import matplotlib.pyplot as plt
import numpy as np
# %% get inputs
def y(t, A, omega, phi):
    answer = A*np.cos((omega*t)+phi)
    return answer

#%% create x-values
x_values = np.linspace((-2*(m.pi)),(2*(m.pi)), 101)
y_alpha = y(x_values, 1, 1, 0)
y_bravo = y(x_values, 2, 1, 0)
y_charlie = y(x_values, 1, 2, 0)
y_delta = y(x_values, 1, 1, ((m.pi)/4))
# %% draw functions

fig, ax = plt.subplots(num=1, clear=True)
ax.plot(x_values, y_alpha, '^-b', markevery=10, ms=10, mec='r', mfc='m', label='$y(t,1,1,0)$')
ax.plot(x_values, y_bravo, 's--y', markevery=10, ms=10, mec='r', mfc='m', label='$y(t,2,1,0)$')
ax.plot(x_values, y_charlie, 'p:b', markevery=10, ms=10, mec='c', mfc='m', label='$y(t,1,2,0)$')
ax.plot(x_values, y_delta, 'h-.b', markevery=10, ms=10, mec='r', mfc='g', label='$y(t,1,1,\pi/4)$')
#ax.plot(force_model, disp_model, 'k-')
ax.grid(True)
ax.legend(loc='best')
ax.set_xlabel('X-Values')
ax.set_ylabel('Y-Values')
ax.set_title('Graph of Different Sinusoidal Functions(md374)')

fig.tight_layout()
fig.savefig('sine_plot.eps')
fig.savefig('SinPlot.pdf')
```

### A.2 P&E 4.48_.py

```python
# -*- coding: utf-8 -*-
"""
[Multi Table]
[Marcus Deans]
[18 September 2019]

I understand and have adhered to all the tenets of the Duke Community Standard
in creating this code.
Signed: [md374]
"""

# %% table function
def mult_table(rows=12, columns=12):
    rows += 1
    columns += 1
    print("  ", end="")

    for a in range(1, columns):
        print("{:4}".format(a), end = "")
    print()

    for x in range(1,rows):
        print("{:2}".format(x), end="")
        for y in range(1,columns):
            print("{:4}".format(x*y), end="")
        print()
```

### A.3 P&E 2.38_.py

```python
# -*- coding: utf-8 -*-
"""
Created on Sat Sep 28 13:50:34 2019

@author: marcu
"""

# -*- coding: utf-8 -*-
"""
[Digit Sum]
[Marcus Deans]
[18 September 2019]

I understand and have adhered to all the tenets of the Duke Community Standard
in creating this code.
Signed: [md374]
"""
#%% DEFINE FUNCTION
def digit_sum(alpha):
    final=0
    while (alpha!=0):
        val = (alpha%10)
        final += val
        alpha = (alpha//10)
    if (final>=10):
        return digit_sum(final)
    return(final)
```

## A.4 Chapra Problem 3.10_.py

```python
# -*- coding: utf-8 -*-
"""
[Singularity]
[Marcus Deans]
[18 September 2019]
I understand and have adhered to all the tenets of the Duke Community Standard
in creating this code.
Signed: [md374]
"""
import numpy as np
import matplotlib.pyplot as plt

#%% previously established singularity function
def singularity(x, a, n):
    return (x > a)*((x - a)**n)
#%% equation from text
def calc(x):
    first = (-5/6)*((singularity(x,0,4))-(singularity(x,5,4)))
    second = (5/2)*(singularity(x,8,3))
    third = (325/2)*(singularity(x,7,2))
    fourth = ((79/12)*(x**3))
    fifth = ((76/3)*x)
    singu = (first + second + third + fourth - fifth)
    return singu

# %%
if __name__ == '__main__':
    x = np.linspace(0,10,100)
    y = calc(x)

    xcalc = np.linspace(0, 10, int(1e7))
    ycalc = calc(xcalc)
    ix = np.array(xcalc)
    iy = np.array(ycalc)

    yMax = max(iy)
    yMin = min(iy)
    xMax = float(ix[np.where(iy==yMax)])
    xMin = float(ix[np.where(iy==yMin)])

    print("Maximum Values: ", "x=", "{:.4e}".format(xMax),"y=", ("{:.4e}".format(yMax)), sep=' ')
    print("Minimum Values: ", "x=", "{:.4e}".format(xMin),"y=", ("{:.4e}".format(yMin)), sep=' ')

    fig, ax = plt.subplots(num=1, clear=True)

    ax.plot(x, y,'k-')
    ax.set_title('Beam Displacement vs. Distance along Beam (md374)')
    ax.set_xlabel('Distance Along Beam in Feet from Left Side')
    ax.set_ylabel('Beam Displacement')
    ax.grid(True)
    fig.tight_layout()
    fig.savefig('SingPlots.eps')
    fig.savefig('SingPlots.pdf')
```

### A.5 Geocode_.py

```
# -*- coding: utf-8 -*-
"""
[Geo Code]
[Marcus Deans]
[23 September 2019]

I understand and have adhered to all the tenets of the Duke Community Standard
in creating this code.
Signed: [md374]
"""

# %% function to check that inputs are numeric
def numeric(alpha):
    try:
        checked=float(alpha)
        alpha = 1
    except:
        alpha = -1
    return alpha
# %% check inputs
def prove_inputs(x_in, y_in, z_in):
    x, y, z = (1,1,1)
    proof = 1
    proof_x = numeric(x_in )
    proof_y = numeric(y_in)
    proof_z = numeric(z_in)
    if ((proof_x ==1) and (proof_y == 1) and (proof_z == 1)):
        x = float(x_in)
        y = float(y_in)
        z = float(z_in)
    else:
        print("All arguments must be single numbers.")
        proof=-1
    if ((x<=0) or (y<=0) or (z<=0)):
        print("All arguments must be positive.")
        proof=-2
    elif ((x>y) and (z>1)):
        print("Invalid sequence.")
        proof=-3
    elif ((x<y) and (z<1)):
        print("Invalid sequence.")
        proof=-3
    return x, y, z, proof

# %% create sequence
def geo_prog(romeo, sierra, tango):
    table = [0]
    start, last, ratio, valid= prove_inputs(romeo, sierra, tango)
    if (valid==1):
        table[0]=start
        final = start
        alter = 1
        trail = 0
        while(alter==1):
            trail+=1
```

```python
            prev = table[trail-1]
            new = ratio*prev
            if ((new<=last) and (ratio>1)):
                alter = 1
            elif ((new>=last) and (ratio<1)):
                alter = 1
            else:
                alter = 0
                break
            table.append(new)
            final += table[trail]
        print(table)
    else:
        final=valid
        print(final)
    return final
```

## B Figures

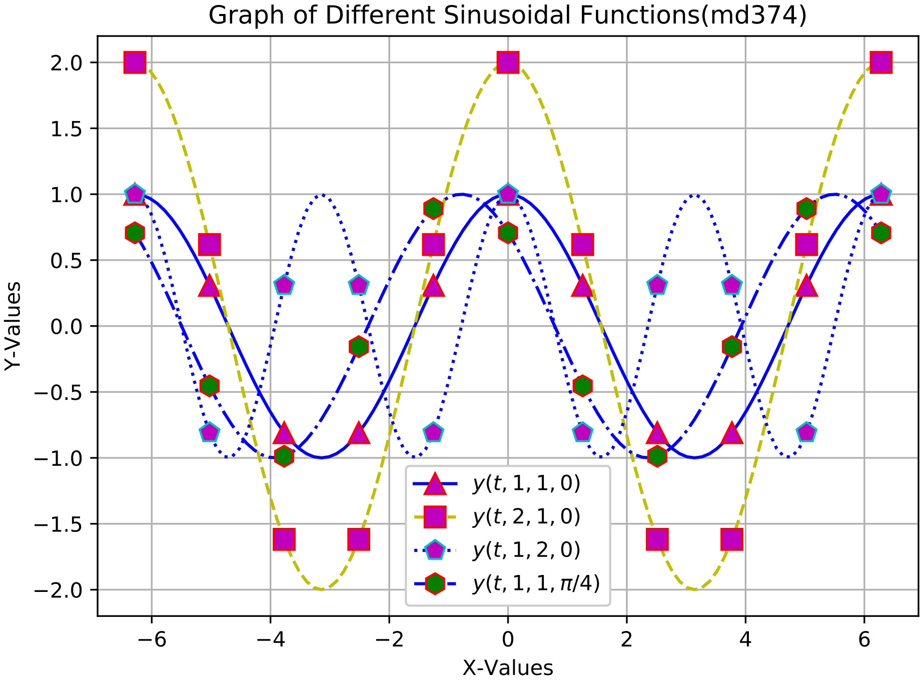

Figure 1: Four sinusoids.

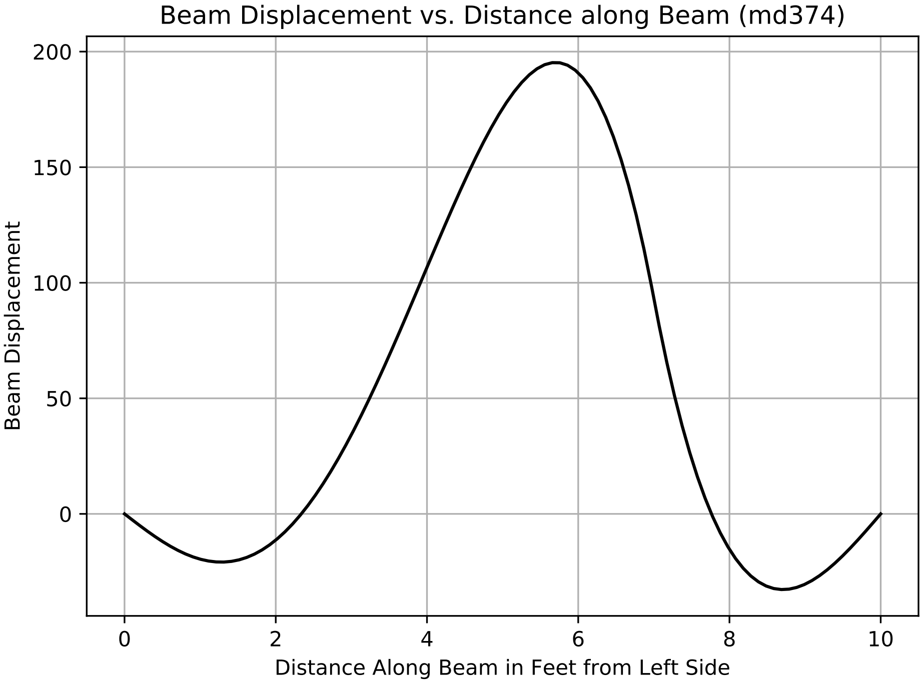

Figure 2: Displacement plot for a beam.