Test 1 – MANE 3351
Manufacturing Engineering Analysis
November 4, 2019

1. This examination is closed book and contains 6 pages,
2. You are provided a formula sheet needed to work this exam and this exam
3. You will need a calculator,
4. You have 75 minutes to complete this exam,
5. The points are clearly labelled,
6. Good Luck!

Name: _____

SID: _____

1. (20 points) **Gaussian Quadrature**

Use 3-point Gaussian Quadrature to evaluate

$$\int_1^3 e^{-sin^2 x}\, dx$$

. Use the truncated Gaussian points and weights shown below.

Table 25.4 ABSCISSAS AND WEIGHT FACTORS FOR GAUSSIAN INTEGRATION

$$\int_{-1}^{+1} f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i)$$

Abscissas $= \pm x_i$ (Zeros of Legendre Polynomials) | Weight Factors $= w_i$

$\pm x_i$	w_i
$n = 2$	
0.57735 02691 89626	1.00000 00000 00000
$n = 3$	
0.00000	0.88888
0.77459 66692 41483	0.55555 55555 55556
$n = 4$	
0.18343 46424 95650	0.36268 37833 78362
0.52553 24099 16329	0.31370 66458 77887
0.79666 64774 13627	0.22238 10344 53374
0.96028 98564 97536	0.10122 85362 90376
$n = 9$	
0.00000 00000 00000	0.33023 93550 01260
0.32425 34234 03809	0.31234 70770 40003

Note that $sin^2 x = \frac{1}{2}[1 - cos(2x)]$ and make sure to work in radians (not in degrees).

Question 1

Note: $a=1$, $b=3$

Points (x_i)	weights (w_i)	$y_i = \frac{(b-a)}{2} x_i + \frac{(b+a)}{2}$
0.0	0.88889	2
0.7746	0.55556	2.7746
-0.7746	0.55556	1.2254

$f(y_i)$	$w_i * f(y_i)$
$\frac{1}{2}(1-\cos(2\cdot 2)) = 0.82682$	$.82682(.88889) = .73495$
$\frac{1}{2}(1-\cos(2\cdot 2.7746)) = 0.12874$	$.12874(.55556) = .07153$
$\frac{1}{2}(1-\cos(2\cdot 1.2254)) = 0.88537$	$.88537(.55556) = .49188$

$$I = \frac{b-a}{2}\sum_{i=1}^{3} w_i f(y_i) = \frac{3-1}{2}(.73495+.07153+.49188) = 1.29836$$

Test

Question 2

In [10]:
```
import math
import numpy as np
def f(z):
    return 0.5*(1.0-math.cos(2.0*z))
#
a=1
b=3
#
# 3 point Gaussian Quadrature
x3=[0.0,0.7746,-0.7746]
w3=[0.88889,0.55556,0.55556]
#
# find y_i
y3=[]
for i in range(0,len(x3)):
    y3.append(((b-a)/2.0)*x3[i]+(b+a)/2.0)

i3=0.0
for i in range(0,len(y3)):
    #print(w3[i]*f(y3[i]))
    i3=i3+w3[i]*f(y3[i])
i3=i3*(b-a)/2.0
print("the area using 3-point gaussian quadrature is {}".format(i3))
```

```
the area using 3-point gaussian quadrature is 1.298355481786122
```

In [ ]:

2. (20 points) **Romberg Integration**

Answer the following questions regarding Romberg integration.

(a) (16 points) Compute two rows of Romberg's algorith to evaluate

$$\int_1^2 \left(x + \frac{1}{x}\right)^2 dx.$$

Clearly indicate your answer.

see Python code

$R(0,0) = 5.125$

$R(1,0) = 4.90972$

$R(1,1) = 4.83796$

(b) (2 points) If the exact answer is 2.19314718, calculate the absolute and percentage error for your best estimate from part a.

Absolute error

$|4.83796 - 2.19314718|$

2.64481

% error

$\frac{|4.83796 - 2.19314718|}{2.19314718} \times 100\%$

120.59%

(c) (2 points) Name the two numerical algorithms that Romberg Integration is based upon.

Trapezoid + Richardson's Extrapolation

Test - Question 2

In [3]:
```
import math
import numpy as np
def f(z):
    return (z+1.0/z)**2.0
#
a=1
b=2
n=1
#
# initialize matrix r
r=np.zeros(shape=(n+1,n+1))
h=b-a
#find R(0,0)
r[0][0]=(h/2.0)*(f(a)+f(b))
for i in range(1,n+1):
    h=h/2.0
    sum=0.0
    for k in range(1,2**i,2):
        sum=sum+f(a+k*h)
    r[i][0]=0.5*r[i-1][0]+sum*h
    for j in range(1,i+1):
        r[i][j]=r[i][j-1]+(r[i][j-1]-r[i-1][j-1])/(4**j-1)
print(r)
#h=(b-a)/n
#print("The area is {} after {} iterations".format(sum,n))
```

```
[[5.125      0.        ]
 [4.90972222 4.83796296]]
```

3. (20 points) **Newton-Raphson** $f'(x) = 6x^2 - 23.4x + 17.7$

(a) (16 points) Calculate three iterations of the Newton-Raphson algorithm to find the root of $f(x) = 2x^3 - 11.7x^2 + 17.7X - 5$ starting at $x_0 = 3$.

1st iteration $X_1 = X_0 - \frac{f(x_0)}{f'(x_0)} = 3 - \frac{-3.2}{1.5} = 5.133$ $f(3) = -3.2$ $f'(3) = 1.50$

2nd iteration $X_2 = X_1 - \frac{f(x_1)}{f'(x_1)} = 5.133 - .86358 = 4.26975$ $f(5.133) = 48.0907$ $f'(5.133) = 55.6866$

3rd iteration $X_3 = X_2 - \frac{f(x_2)}{f'(x_2)} = 4.26975 - .47682 = 3.79293$ $f(4.26975) = 12.95624$ $f'(4.26975) = 27.17344$

(b) (2 points) Name the main advantage and main disadvantage of the Newton-Raphson algorithm.

Main advantage: fast (converges quickly)

Main Disadvantage: Requires derivative

(c) (2 points) Suggest another root finding algorithm with performance similar to the Newton-Raphson method that does not require knowledge of the derivative.

Secant

4. (20 points) **False Position**

Calculate four iterations of the false position method to find the root of

$$f(x) = x^{10} - 1$$

between $x = 0$ and 1.3.

Iteration	by hand	by Python
1	.0943	.0942996
2	.1818	.1875888
3	.26287	.262874
4	.34311	.338105

Correct formula +4

Correct iteration +4/iteration

Question 4

~~$x_u = 0$~~ , $x_l = 0.0$, $x_u = 1.3$, $f(x_l) = f(0) = -1$, $f(x_u) = f(1.3) =$
$= 12.7858$

Iteration 1

$$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)} = 1.3 - \frac{12.78585(0 - 1.3)}{-1 - 12.78585} = 0.0943$$

$$f(x_r) \doteq -1.0$$

replace x_l with x_r, $x_l = 0.0943$, $x_u = 1.3$

Iteration 2

$$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)} = 1.3 - \frac{12.78585(0.0943 - 1.3)}{-1 - 12.7875} = 0.1818$$

$$f(x_r) = -.99999966$$

replace x_l with x_r, $x_l = .1818$, $x_u = 1.3$

Iteration 3

$$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)} = 1.3 - \frac{12.78585(.1818 - 1.3)}{-.99999966 - 12.78585} = .26287$$

$$f(x_r) = -.99999842.$$

replace x_l with x_r, $x_l = .26287$, $x_u = 1.3$

Iteration 4

$$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)} = 1.3 - \frac{12.78585(.26287 - 1.3)}{-.99999842 - 12.78585} = .34311$$

$x_r = $ ~~.33~~ .34310 (by hand)
.33810 (by Python)

Test - Question 4

In [9]:
```
# Question 4
import math
def f(x):
    return x**10.0 - 1.0
# input a & b
while True:
    a=0.0
    b=1.3
    if f(a)*f(b)<0.0:
        print("root is bounded")
        break
counter=0
print("starting point: a={},f(a)={}".format(a,f(a)))
print("starting point: b={},f(b)={}".format(b,f(b)))
# process loop
while True:
    xr= b - (f(b)*(a-b))/(f(a)-f(b))
    print("iteration {}, xr is {},f(xr)={}".format(counter,xr,f(xr)))
    if f(a)*f(b)<0.0:
        a=xr
    else:
        b=xr
    counter=counter+1
    if counter>3:
        print("counter exceeded")
        break
print("the root is {}".format(xr))
print("the value at the root is {}".format(f(xr)))
```

```
root is bounded
starting point: a=0.0,f(a)=-1.0
starting point: b=1.3,f(b)=12.785849184900005
iteration 0, xr is 0.09429959537232735,f(xr)=-0.999999999944397
iteration 1, xr is 0.1817588725190793,f(xr)=-0.9999999606489614
iteration 2, xr is 0.2628740125203042,f(xr)=-0.9999984242890871
iteration 3, xr is 0.3381051033222693,f(xr)=-0.9999804783168278
counter exceeded
the root is 0.3381051033222693
the value at the root is -0.9999804783168278
```

In [ ]:

$x^4 - 8x^3 - 35x^2 + 450x - 1001$

5. (20 points) **Bisection Method**

(a) (16 points) Calculate four iterations of the bisection method find the root of

$$f(x) = x^4 - 8x^3 - 35x^2_4 50x - 1001$$

with starting values of $a = 4.5$ and $b = 6.0$.

Correct formula +4
each correct iteration +3/iteration

(b) (4 points) Name one advantage and one disadvantage of the bisection method.

Question 5

starting conditions $a = 4.5$, $b = 6.0$

$f(a) = f(4.5) = -3.6875$, $f(b) = f(6) = 7$, so bracketed

Iteration 1

$$M = \frac{a+b}{2} = \frac{4.5+6}{2} = 5.25$$

$f(5.25) = -1.12109$

$a = M = 5.25$, $b = 6.0$

Iteration 2

$$M = \frac{a+b}{2} = \frac{5.25+6}{2} = 5.625$$

$f(5.625) = .12915$

$a = 5.25$, $b = 5.625$

Iteration 3

$$M = \frac{a+b}{2} = \frac{5.25+5.625}{2} = 5.4375$$

$f(5.4375) = -0.91551$

~~$a = 5.4375$~~ $a = 5.4375$, $b = 5.625$

Iteration 4

$$M = \frac{a+b}{2} = \frac{5.4375+5.625}{2} = 5.53125$$

$f(5.53125) = -.53229$

Test, Question 5

In [2]:
```
# September 18
# Pseudo-code 1
from scipy import stats
import math
def f(x):
    return x**4-8*x**3-35*x**2+450*x-1001
# initialize code
a=4.5
b=6.0
tol=0.005
N=3
err=math.fabs(b-a)
L=f(a)
for i in range(0,N+1):
    m=(a+b)/2.0
    M=f(m)
    err=err/2.0
    if (err<tol):
        print("stopping inside for loop at iteration {}".format(i))
        break
    if (L*M<0):
        b=m
    else:
        a=m
        L=M
    print("iteration {}, a={}, b={}".format(i,a,b))
print("the root is {} with f({})={}".format(m,m,M))
```

```
iteration 0, a=5.25, b=6.0
iteration 1, a=5.25, b=5.625
iteration 2, a=5.4375, b=5.625
iteration 3, a=5.53125, b=5.625
the root is 5.53125 with f(5.53125)=-0.5322866439819336
```

In [ ]: