ISYE 6669 – HOMEWORK 4

1 – CONVEX OPTIMIZATION

ANS 1.1 -

$f(x) = e^x - x$

$f'(x) = e^x - 1$
$f''(x) = e^x$

$x^{k+1} = x^k + f'(x^k)/ f''(x^k)$
$x^0 = -1$

Let's convert this to a simple program to do this for us –

```
e = 2.71828183
x0 = -1
der1 = (e**x0)-1
der2 = (e**x0)
count = 1

while abs(der1) >= 1/(10**5):

    print("\nIteration ",count,': ')
    print('X value: ',x0)
    print('First Derivative: ',der1)
    print('Second Derivative: ',der2)
    x0 = x0 - (der1/der2)
    der1 = (e**x0)-1
    der2 = (e**x0)
    count += 1
```

Upon running a python program with a while loop conditioned as mentioned in the problem – the absolute value of first derivative must be lesser than 10^{-5}, we get –

Iteration 1 :
X value: -1
First Derivative: -0.6321205590371033
Second Derivative: 0.36787944096289676

Iteration 2 :
X value: 0.7182818300000002
First Derivative: 1.0509063766879514
Second Derivative: 2.0509063766879514

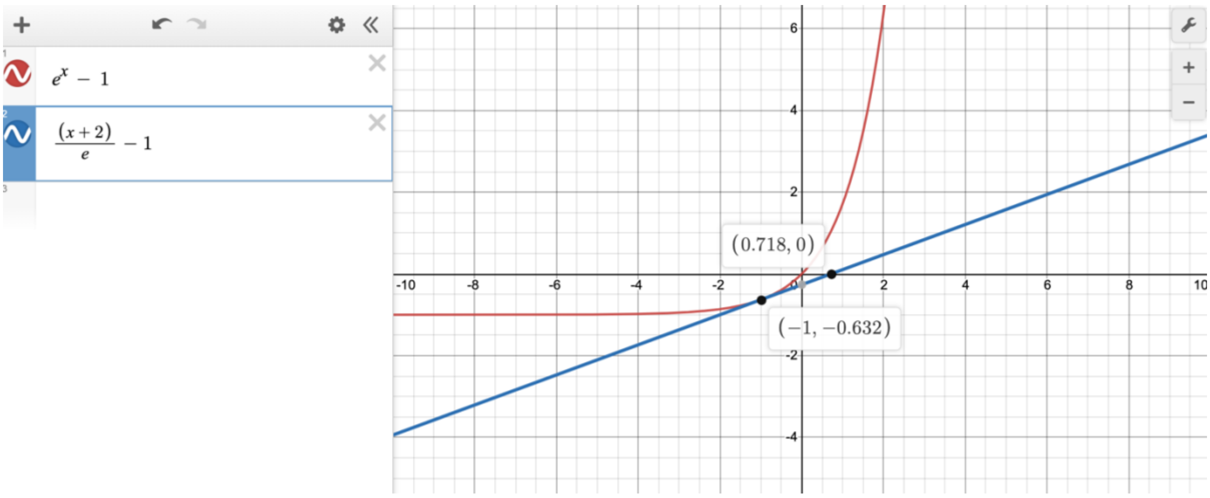
Iteration 3 :
X value: 0.20587112776936967
First Derivative: 0.22859486217556269
Second Derivative: 1.2285948621755627

Iteration 4 :
X value: 0.019809091199582074
First Derivative: 0.020006593214233348
Second Derivative: 1.0200065932142335

Iteration 5 :
X value: 0.00019491091160087
First Derivative: 0.00019492990807723487
Second Derivative: 1.0001949299080772

ANS 1.2 -

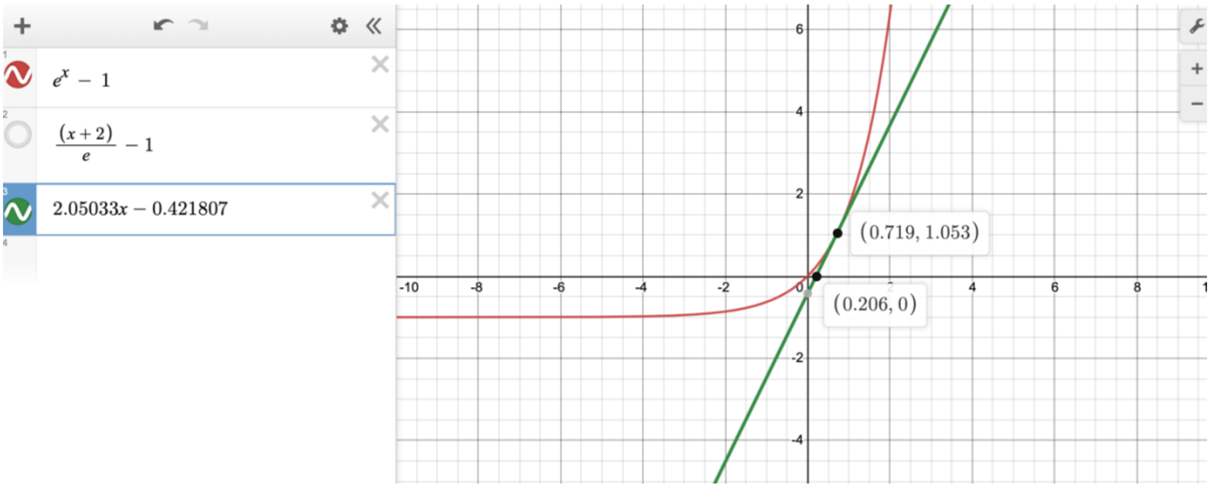
Tangent 1 at x^0 –



In the form k(x-a) + b = 0, the tangent would be –

(x – (e – 2)) = 0
k = 1
a = e-2
b = 0
x^1 = 0.718 (value from Newton's method = 0.2718)

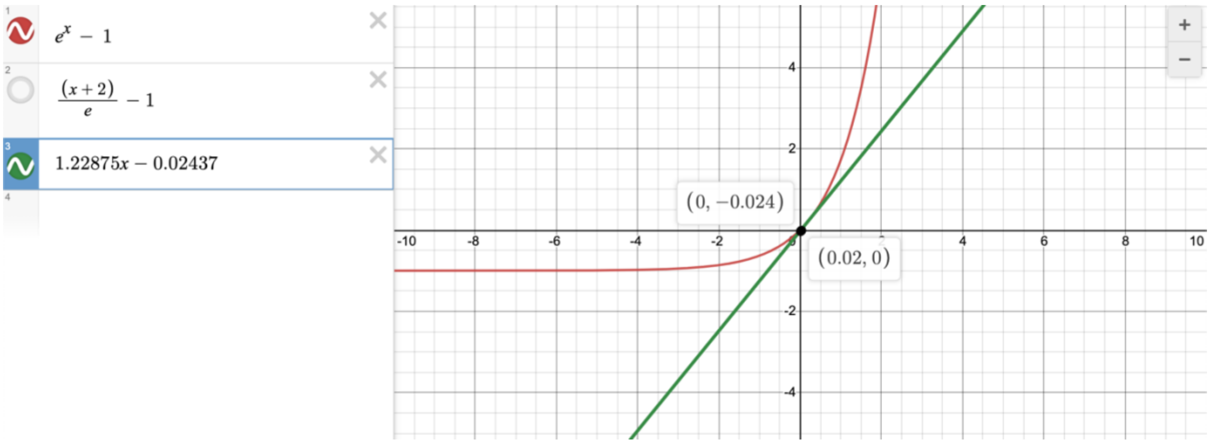
Tangent 1 at x^1 –



In the form k(x-a) + b = 0, the tangent would be –

2.05033(x) + (-0.421807) = 0
k = 2.05033
a = 0
b = -0.421807
x^2 = 0.206 (value from Newton's method = 0.2058)

Tangent 1 at x^2 –



In the form k(x-a) + b = 0, the tangent would be –

1.22875(x) + (-0.02437) = 0
k = 1.22875
a = 0
b = -0.02437
x^3 = 0.02 (value from Newton's method = 0.019)

The new values from these observations match the new values from part 1!

2 – NONCONVEX OPTIMIZATION

ANS 2.1 -

Minimum (x_1,x_2) = (1.47568569, -0.32234972)
Objective Value = 1.6129287968151138
Code for finding global minimum using the BFGS update method –

```
import scipy.optimize as optimize

def f(x):

    x1 = x[0]
    x2 = x[1]
    y1 = (1-x1+(x1*x2))**2
    y2 = (2-x1+((x1**2)*x2))**2
    y3 = (3-x1+((x1**2)*x2))**2
    return y1 + y2 + y3

bnds = ((-5, 5), (-5, 5))

res = optimize.minimize(fun=f,x0=[0,0],method='L-BFGS-
B',bounds=bnds)
res.x
```

Run in jupyter –

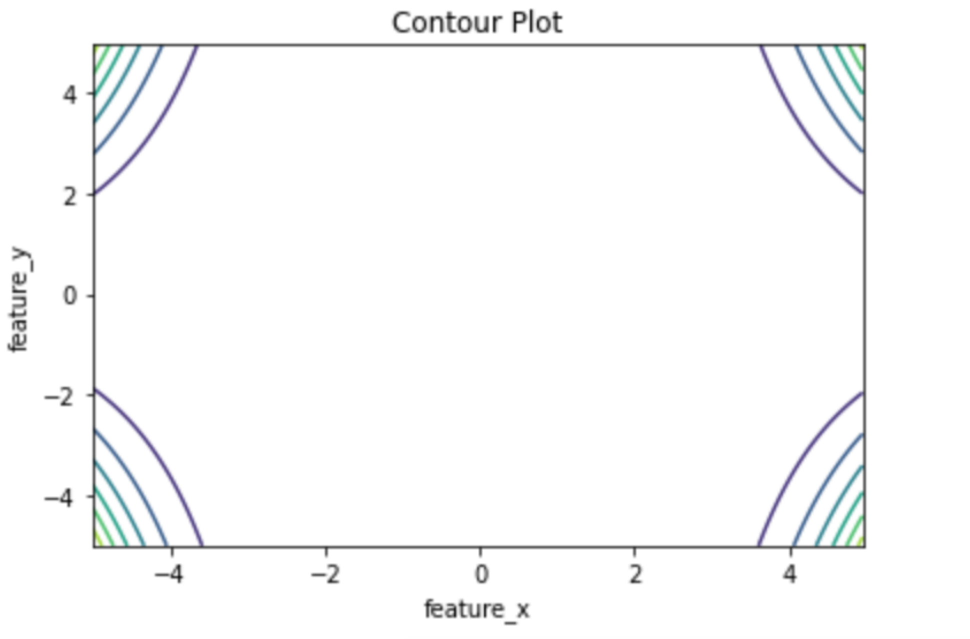
```
import scipy.optimize as optimize
```

```
def f(x):

    x1 = x[0]
    x2 = x[1]
    y1 = (1-x1+(x1*x2))**2
    y2 = (2-x1+((x1**2)*x2))**2
    y3 = (3-x1+((x1**2)*x2))**2
    return y1 + y2 + y3

bnds = ((-5, 5), (-5, 5))
```

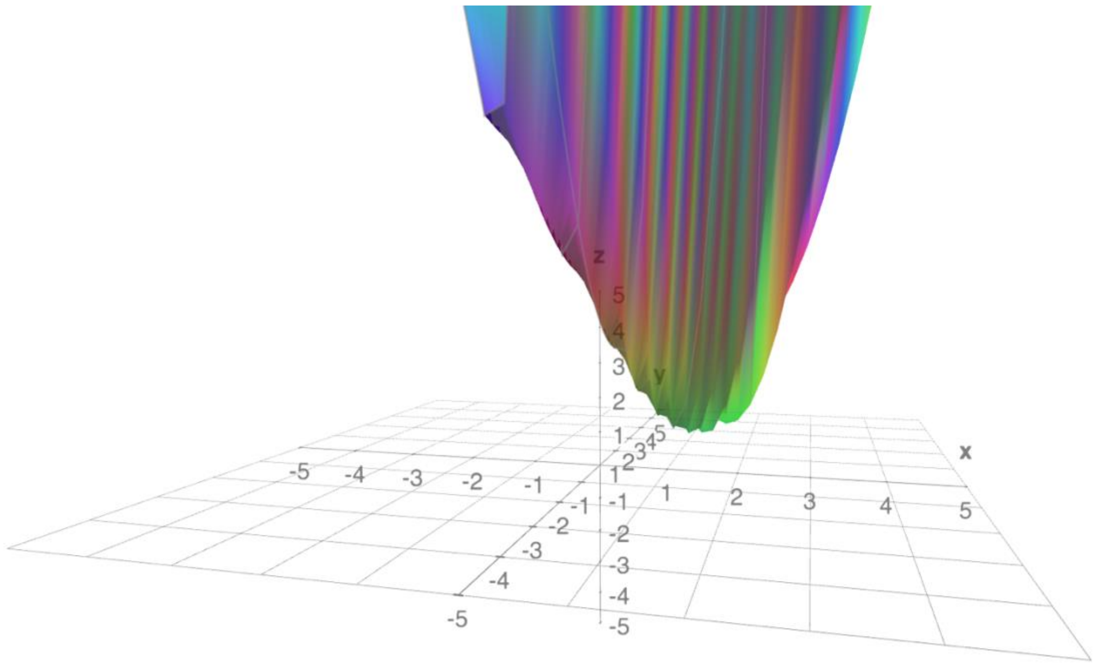
```
res = optimize.minimize(fun=f,x0=[0,0],method='L-BFGS-B',bounds=bnds)
res
```

```
      fun: 1.6129287968151138
 hess_inv: <2x2 LbfgsInvHessProduct with dtype=float64>
      jac: array([-6.86117833e-06,  6.15063556e-06])
  message: b'CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL'
     nfev: 42
      nit: 11
     njev: 14
   status: 0
  success: True
        x: array([ 1.47568569, -0.32234972])
```

2D Contour Plot –



3D Plot –



ANS 2.2 -

For global minima –

F1 = −2.062605516810367

X1, X2 = [-1.349385, -1.349385]
X1, X2 = [1.34938658, -1.3493853]
X1, X2 = [-1.3493853, 1.34938658]
X1, X2 = [-1.3493853, 1.34938658]

For local minima –

F = -1.554458311873118
X1, X2 = [-7.63259085 7.63259226]

Note – Both global and local minima were derived from looking at contour plots within the interval of [-10, 10] and taking starting points within likely regions.

This is the code for a local minimum –

```
import scipy.optimize as optimize

def f(x):

    x1 = x[0]
    x2 = x[1]
    pwr = abs(100 - (((x1**2 + x2**2)**0.5)/3.14159))
    inner = abs(np.sin(x1)*np.sin(x2)*(2.718281**pwr))
    outer = -0.0001*((inner-1)**0.1)
    return outer

bnds = ((-10, 10), (-10, 10))

res = optimize.minimize(fun=f,x0=[-7.5,7.5],method='L-BFGS-B',bounds=bnds)
res
```

Run in jupyter for global minima as well –

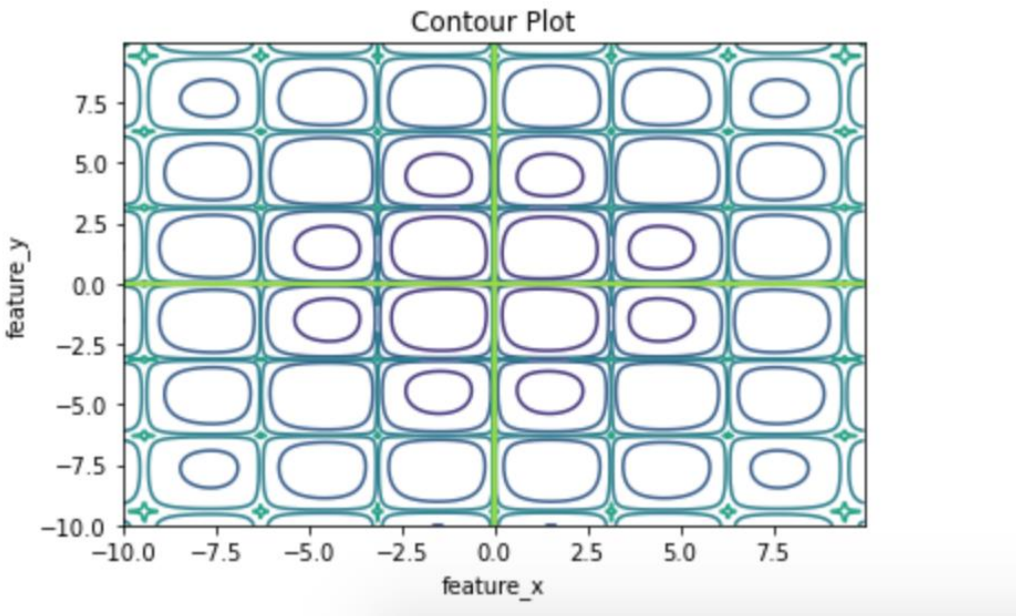
```
import scipy.optimize as optimize
```

```
def f(x):

    x1 = x[0]
    x2 = x[1]
    pwr = abs(100 - (((x1**2 + x2**2)**0.5)/3.14159))
    inner = abs(np.sin(x1)*np.sin(x2)*(2.718281**pwr))
    outer = -0.0001*((inner-1)**0.1)
    return outer

bnds = ((-10, 10), (-10, 10))
```

```
#res = optimize.minimize(fun=f,x0=[-7.5,7.5],method='L-BFGS-B',bounds=bnds)
res1 = optimize.minimize(fun=f,x0=[-1.0,-1.0],method='L-BFGS-B',bounds=bnds)
res2 = optimize.minimize(fun=f,x0=[1.0,-1.0],method='L-BFGS-B',bounds=bnds)
res3 = optimize.minimize(fun=f,x0=[-1.0,1.0],method='L-BFGS-B',bounds=bnds)
res4 = optimize.minimize(fun=f,x0=[1.0,1.0],method='L-BFGS-B',bounds=bnds)
print(res1.fun)
print(res2.fun)
print(res3.fun)
print(res4.fun)
```

```
-2.0626055168018405
-2.062605516810367
-2.062605516810367
-2.062605516816397
```

2D Contour Plot –



3D Plot –