Assignment 4_PHYB57

In [24]:
```
import numpy as np
```

Define derivatives (central differences) for 2-D situation

In [25]:
```
def cdf_2d(f,x,y,h=1e-5):

    df_dx = (f(x+h, y) - f(x-h, y)) / (2*h)
    ## Derivative for f with respect to x

    df_dy = (f(x, y+h) - f(x, y-h)) / (2*h)
    ## Derivative for f with respect to y

    return df_dx, df_dy
```

Define two functions

In [26]:
```
def u(x, y):

    return x**2 + x*y - 10

def v(x, y):

    return y + 3*x*y**2 - 57

def u_1(x, y):

    return x**2 + x*y - 30
    ## The third function is used to test the
    ##"Value Error" fuction, clearly
    ## u and u_1 have no intersection.
```

Define newton_2d function (partial derivative and Jacobian matrix are inside the function)

In [27]:
```
def newton_2d(f, g, x_0, y_0):

    max_iter = 1000
    ## Max times of interation accectable

    tol = 1e-10
    ## When difference is less than 1e-10, we say it converges

    h = 1e-5
    x, y = x_0, y_0
    ## Give value to x and y.

    for i in range(max_iter):
        df_dx, df_dy = cdf_2d(f, x, y)
        dg_dx, dg_dy = cdf_2d(g, x, y)

        J = abs(df_dx * dg_dy - df_dy * dg_dx)

        x_new = x - (f(x, y)*dg_dy - g(x, y)*df_dy) / J
        y_new = y - (g(x, y)*df_dx - f(x, y)*dg_dx) / J

        if abs(x_new - x) < tol and abs(y_new - y) < tol:
            return (x_new, y_new), i + 1

        x, y = x_new, y_new

    raise ValueError("No convergence after maximum number of iterations.")
    ## If out of 1000 interration,
    ## we say "No convergence after maximum number of iterations"
```

Test Functions

In [30]:
```
initial_guess = (u,v,1, 1)
(root, iterations) = newton_2d(*initial_guess)
print("Root:", root)
print("Iterations:", iterations)

## This is a test for u and v.
## The result is (2.0,2.9999999999999996)
## Result obtained within 7 interations.
```

```
Root: (2.0, 2.9999999999999996)
Iterations: 7
```

2D grid search system

In [35]:
```
def grid_search_newton():

    x_range = np.linspace(-10, 10, num=101)
    y_range = np.linspace(-10, 10, num=101)
    x_grid, y_grid = np.meshgrid(x_range, y_range)

    for xi, yi in zip(x_grid.ravel(), y_grid.ravel()):
    ## Iterate (loop) over each point in the grid. Here, ravel is used to f
    ##into 1D arrays so we can loop through each combination of (xi, yi) in

        x_root, y_root, num_iterations = newton_2d(u, v, xi, yi)
        ##For each grid point (xi, yi), try to find the roots of the functi
        ##the newton_2d method.

        if num_iterations < max_iter:  # This checks if it converged

            return x_root, y_root, num_iterations

    return None, None, max_iter
```

In [ ]:
```

```