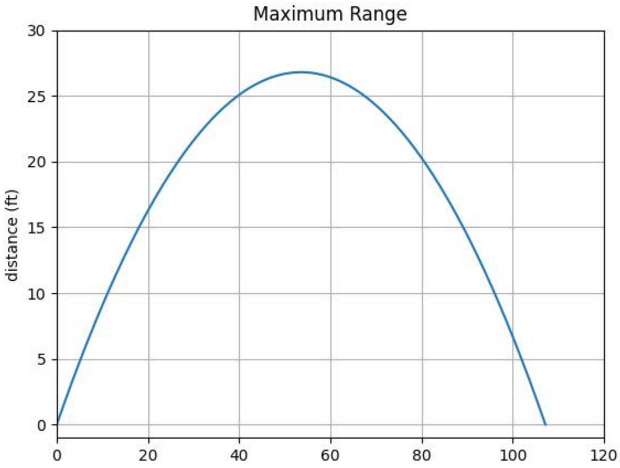
PYTHON CODE

```
from lib.Header import *

# Givens

g = 32.2 # ft/s/s
m =  5 / g # slug (lbs / ft/s/s)
k0 = 10 # lbs/ft
k1 = 10 # lbs/ft^2
h = 1 # ft
theta = 45*np.pi/180 # degrees

# Q1. Compute Total Potential Energy, PE, and "Report" the computed PE.

def func1(s):
    return 2*(k0 + k1 * np.sqrt(s**2 + h**2) * s)

def Gaussian (func, x1, x2, N):
    x = np.linspace(x1, x2, N + 1)
    zta1 = 1 / np.sqrt(3)
    zta2 = -1 / np.sqrt(3)
    area = 0
    for i in range(len(x)-1):
        a = x[i]
        b = x[i+1]
        bma2 = (b-a)/2
        apb2 = (b+a)/2
        x1 = apb2 + bma2 * zta1
        x2 = apb2 + bma2 * zta2
        area += bma2 * (func(x1) + func(x2))
    return area

Int = Gaussian(func1, 0, 1, 1000)
print2(f"Potential Energy = {Int}")

# Q2. Compute the shooting speed, v0, and "Report" the computed v0.

def func2(PE):
    return np.sqrt((2*PE) / m)

def centralDiff1(func, x, h):
    return (func(x+h) - func(x-h)) / (2*h)

v0 = centralDiff1(func2, .001, .0005)
print2(f"v0 = {v0}")

# Q3. Find Projectile Landing Distance (xmax)
#   -Use either MATLAB command "fzero" or your own root finding code
```

```
#   -Use the computed initial shooting speed, v0
#   -Compute x value that meets y=0 condition

def fn_secant(x0, x1, mypoly, counter):  # SECANT METHOD
    tol = 1e-10
    counter += 1

    y0 = mypoly(x0)
    y1 = mypoly(x1)

    x2 = x1 - (y1 * (x1 - x0)) / (y1 - y0)
    y2 = mypoly(x2)

    if abs(y2) < tol:
        xf = x2
        yf = y2
        return [xf, yf, counter]

    [xf, yf, counter] = fn_secant(x1, x2, mypoly, counter)

    return [xf, yf, counter]

def func3(x):
    return -.5 * g * ((x)/(v0*np.cos(theta)))**2 + x * np.tan(theta)

counter = 0
xmax = fn_secant(5,125, func3, counter)
print2(f"xmax = {xmax[0]}")


# Q4. Plot trajectory of the projectile
#   - X range is from 0 to xmax
#   - Use equation (3) to plot the trajectory

plt.figure()
# plt.plot(np.linspace(0,,1000), func3(np.linspace(0,200,1000)))
plt.plot(np.linspace(0,xmax[0],1000), func3(np.linspace(0,xmax[0],1000)))
plt.xlim(0, 120)
plt.ylim(-1, 30)
plt.xlabel("")
plt.ylabel("distance (ft)")
plt.title("Maximum Range")
plt.grid()
SAVE(1)
plt.show()



PDF()
```

OUTPUT

Plots



Prints

Potential Energy = 32.18951416497461
v0 = 58.746792240196925
xmax = 107.17967697244652