Lab 3

Name: (your name)

Some remarks before you start:

- Import the Python packages below each time you start a session.
- Before submitting your report, remove all ouputs by clicking on **Edit→Clear all ouputs**. Then download your report by clicking on **File→Download→.ipynb**. This is the file you will submit on Canvas.

In this lab, you will practice:

- define and graph functions
- use Bisection method and Newton-Raphson method to find roots
- use Gradient Descent/Ascent method to find local min/max
- solve problems of projectile motion and mortgage rate

Problems	Points
$1, 3, 5, 7$	3
$2, 4, 6, 8, 9$	4
Readability of your report	3
Total: 9	Total: 35

I. Import necessary Python packages

Execute the following code each time you start a session.

```
from matplotlib.pyplot import*
from autograd.numpy import*
from autograd import elementwise_grad as diff
```

II. Define and graph functions

To define a function on Python, you will use the keyword **def**. For example, the code below defines the function

$$f(x) = \cos^2(x) - \cos(x^2)$$

```
def f(x):
  return cos(x)**2-cos(x**2)
```

To graph the function, you first need to generate an array of x-values using the command **linspace**. For example,

```
x=linspace(0,6,16)  # Generates 16 evenly-spaced points between 0 and 6
print(x)
```

Note that 16 is the number of points, not the number of subintervals. The number of subintervals in this case is 15.

At these values of x, the command **plot** will connect the points $(x, f(x))$ together by straight lines.

```
plot(x,f(x))
```

You can label the axes and give a title to the plot as follows.

```
plot(x, f(x))
xlabel("x")
ylabel("f(x)")
title("Plot of $f(x) = \cos(x)^2 - \cos(x^2)$")
```

You can graph two or more functions on the same graph. For example

$$f(x) = x^2, \;\; g(x) = 1 - x$$

```
plot(x, x**2, label="graph of $f(x)=x^2$")
plot(x, 1-x, label="graph of $g(x)=1-x$")
xlabel("x")
legend()
title("two functions")
```

Exercise 1

Graph the function $g(x) = \cos(x) - \cos(\sqrt{2}x) + \frac{1}{3}\sin(3x)$ on the interval $[0, 12]$. In your **plot** command, use 100 equal *subintervals* (therefore, the number of equally-spaced points is 101). Give your plot a title.

Exercise 2

To find the derivative of f, use the command **diff**. For example,

```
df = diff(f)
x=linspace(0,6,16)
plot(x,df(x))
```

To evaluate the derivative of f at $x = 1$, namely $f'(1)$, just run the command **df(1.0)** or **df(1.)**. If you run **df(1)**, Python will give an error message because it wants a float-point number for its argument instead of an integer.

- With the function g in Exercise 1, evaluate $g'(0)$, $g'(\pi)$, and $g''(1)$. *Hint: g'' is the derivative of g'.*
- Graph g, g', g'' together on the same plot on the interval $[0, 12]$. In your **plot** command, use 100 equal subintervals.

III. Bisection method

Suppose a continuous function f has different signs at $x = a$ and at $x = b$. *Intermediate Value Theorem* guarantees that f must have a root between a and b. You can find approximately this root using the Bisection method, i.e. by the process of halfing the search interval until a certain number of divisions is reached or until a desirable precision is reached. If the original search interval $[a, b]$ is divided n times, the Python code is as follows.

```
def bisection(f,a,b,n):
  for i in range(n):
    c = (a+b)/2
    if f(c)*f(a)<0: b = c
    else: a = c
    print(i, c)
  return (a+b)/2
```

Exercise 3

- Explain what each line of the code above does.
- Use this code, with a and b of your choice, to find approximately $\sqrt{2}$ (root of $f(x) = x^2 - 2$). How large must n be to get 4 correct decimal places?

Exercise 4

- Graph two curves $y = xe^{-x}$ and $y = \frac{1}{x^2+1}$ on the same plot.
- From the graph, how many intersection points do you see? Give rough estimates for the coordinates of those intersection points.
- Use the Bisection method to find the coordinates of those points. The x-coordinates must be correct with an allowable error of 0.001.

IV. Newton-Raphson method

In the Newton-Raphson method to solve $f(x) = 0$, you start with an initial guess x_0. Then successively update this guess by $x_1, x_2, x_3, \ldots$ using the recursive formula

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

You stop when a certain number of iterations is reached or when a certain precision is reached, i.e. $|x_{n+1} - x_n|$ is less than some prescribed ϵ. The Python code for doing n iterations is as follows.

```
def newton(f,x0,n):
  df = diff(f)
  x = float(x0)
  for i in range(n):
    x = x - f(x)/df(x)
    print(i, x)
  return x
```

Exercise 5

- Explain what each line of the code does.
- Find $\sqrt{2}$ by finding the root of $f(x) = x^2 - 2$. Use initial guess $x_0 = 2$. How large is n in order to obtain a precision of 0.00001 ? In other words, how large is n such that $|x_{n+1} - x_n| < 0.00001$?

Exercise 6

In projectile motion, the horizontal x and the vertical y positions are functions of time. From Newton's Second Law of Motion:

$$x = v_0 \cos(\theta) t, \quad y = h + v_0 \sin(\theta) t - \frac{1}{2} g t^2$$

where:

x = horizontal displacement

y = vertical displacement

h = initial height

v_0 = initial velocity

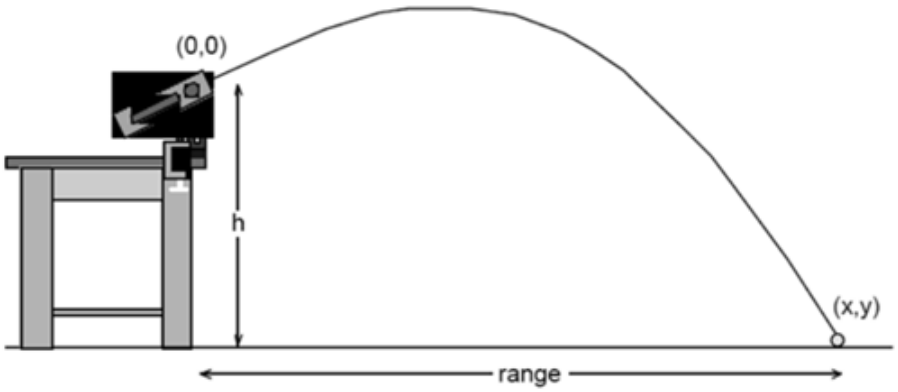
θ = angle of projection

g = 9.81 m/s^2 = acceleration due to gravity

From the first equation, one can extract t in terms of x and then substitute t into the second equation. One arrives at the formula:

$$y = h + x \tan(\theta) - \frac{g x^2}{2 v_0^2 \cos^2(\theta)}$$

The *horizontal range* is the value of x such that $y = 0$.



With an initial velocity of 50 m/s and an initial height of 10 meters, what is the angle θ to obtain a horizontal range of 225 meters? Give your answer in degree (not radian), correct to 4 decimal places.

V. Gradient Descent/Ascent methods

The *Gradient Descent method* is a simple algorithm to find a local minimum of a function f. You start with an initial guess x_0 and successively update it using the recursion

$$x_{n+1} = x_n - \alpha f'(x_n)$$

The *Gradient Ascent method* is to find a local maximum. It uses the same recursion formula above, except that the minus sign is changed to plus sign. The Python code to compute x_1, x_2,..., x_n in the Gradient Descent method is as follows.

```
def gradd(f,x0,alpha,n):
  df = diff(f)
  x = float(x0)
  for i in range(n):
    x = x - alpha*df(x)
    print(i, x, df(x))
  return x
```

Exercise 7

- Explain what each line in the code above does.
- Use the code to find the minimum value of $f(x) = x^2$ with initial guess $x_0 = 1$. Stop when $|f'(x_n)| < 0.01$. Give a value of the learning rate α for which the sequence (x_n) converges quickly to the minimum. Give a value of α for which the sequence (x_n) fails to converge.

Exercise 8

- Graph the function $2x^4 - 9x^3 + 18x - 4$ on the interval $[-2, 4]$.

- Use the Gradient Ascent method to find its local maximum. Choose your own initial guess and learning rate. Stop when $|f'(x_n)| < 0.005$.

Exercise 9

If you take a loan of P (dollars) at a mortgage rate APR$= r$ and if the loan term is n months, then your monthly payment is

$$q = \frac{Pr/12}{1-(1+r/12)^{-n}}$$

You might have seen this formula if you took Math 107. Imagine that you have found your dream house that is listed for $400,000. You can only afford $2,000 per month. Suppose the loan term is 30 years. How low must the mortgage rate be (in percentage) for you to afford this house?

Hint: you can model this problem as an optimization problem or as a root-finding problem.