


# MA 305: Final Project

 Choose one problem and work in the assigned group. Organize your python code using modules and functions. Write sufficient comments in your code for clarity. Use formatted output for nicer display. Give appropriate title, labels and legends in the plots. Prepare your final report using the guidelines posted at your course Canvas, inserting tables and figures in the text with appropriate captions.

---

## I. Computing $\pi$ .

**1. Numerical Integration.** The value of  $\pi$  can be computed by using the definite integrals.

$$(i) \int_0^1 4\sqrt{1-x^2} dx \qquad (ii) \int_{-1}^1 \frac{1}{\sqrt{1-x^2}} dx$$

(a) Implement the Trapezoid rule:

$$\int_a^b f(x) dx \approx \Delta x \left[ \frac{1}{2}f(x_0) + \sum_{i=1}^{N-1} f(x_i) + \frac{1}{2}f(x_N) \right], \quad \Delta x = \frac{b-a}{N}, \quad x_i = a + i\Delta x$$

and the Simpson's 1/3<sup>rd</sup> rule:

$$\int_a^b f(x) dx \approx \frac{\Delta x}{3} \left[ f(x_0) + 4 \sum_{i=1,3,5}^{N-1} f(x_i) + 2 \sum_{i=2,4,6}^{N-2} f(x_i) + f(x_N) \right], \quad \Delta x = \frac{b-a}{N}, \quad x_i = a + i\Delta x$$

to approximate the integral (i).

(b) Implement the Midpoint rule:

$$\int_a^b f(x) dx \approx \Delta x \sum_{i=1}^N f(x_i), \quad \Delta x = \frac{b-a}{N}, \quad x_i = a + \frac{2i-1}{2}\Delta x$$

to approximate both the integrals (i) and (ii).

Which method gives the best approximation to  $\pi$ ? How accurate did you get? Display your results (error of approximation for different values of  $N$ ) both graphically and in tabular form.

**2. Sum of Alternating Series.** From Calculus II we know that

$$\tan^{-1} x = x - \frac{x^3}{3} + \frac{x^5}{5} - \cdots \approx \sum_{n=1}^N (-1)^{n+1} \frac{x^{2n-1}}{2n-1} \quad (1)$$

This is an alternating series which converges for  $-1 < x \leq 1$ . Thus, for  $x > 0$  we know that if we only sum the first  $N$  terms then the error (the difference between  $\arctan x$  and the partial sum) is no larger than the absolute value of the next term, i.e  $|x^{2N+1}/(2N+1)|$ .

- a. Write a function `arctan(x,N)` that takes  $x$  and  $N$ , and returns the sum of the first  $N$  terms in the series. The value of  $\pi$  can be approximated by `4*arctan(1,N)`. Explore how large should  $N$  be to get a good approximation to  $\pi$ !

b. **Machin's Formula**

$$16 \tan^{-1}\left(\frac{1}{5}\right) - 4 \tan^{-1}\left(\frac{1}{239}\right) = \pi$$

uses a combination of the series of  $\arctan(x)$  to compute  $\pi$ . Write a function `machine(N)` that takes  $N$  and returns the value `16*arctan(1/5,N) - 4*arctan(1/239,N)` as an approximation to  $\pi$ . How big should  $N$  be to get a good approximation to  $\pi$  using the function `machine()`? Any surprise?

- c. Compare your results from (a) and (b) with the **Madhava's series**

$$\sqrt{12} \left( 1 - \frac{1}{3 \cdot 3} + \frac{1}{5 \cdot 3^2} - \frac{1}{7 \cdot 3^3} + \cdots \right) = \pi$$

**3. Monte Carlo Integration.** The value of  $\pi$  can also be calculated using the following procedure (Dart Board Algorithm): Inscribe a unit circle  $x^2 + y^2 = 1$  in a square ( $|x| \leq 1$ ,  $|y| \leq 1$ ). Randomly generate  $N$  points in the square. Determine the number of points in the square that are also in the circle. Let  $r$  be the number of points in the circle divided by the number of points in the square. Then  $\pi \approx 4r$ .

- Write a function `mc_area(N)` to approximate  $\pi$  using the above algorithm. To generate random number in  $(-1,1)$ , use `random.uniform(-1,1)` How many correct digits did you manage to find?
- Extending this idea write a function `mc_volume(N)` to approximate the volume of the solid ("Ice Cream Cone") that lies above the cone  $z = \sqrt{x^2 + y^2}$  and below the sphere  $x^2 + y^2 + (z - 1)^2 = 1$ . The exact volume is computed using the triple integral in spherical coordinates.

$$\int_0^{2\pi} \int_0^{\pi/4} \int_0^{2 \cos \phi} \rho^2 \sin \phi \, d\rho d\phi d\theta = \pi$$

[**Hint:** Generate  $N$  random numbers in the box  $-1 \leq x \leq 1$ ,  $-1 \leq y \leq 1$ ,  $0 \leq z \leq 2$  and count how many points  $(x, y, z)$  satisfy the conditions  $x^2 + y^2 < z^2$  and  $x^2 + y^2 + (z - 1)^2 < 1$ .]

## II. Root Finding using Newton's Method.

1. Write a function `newton()` to implement Newton's method for finding a root of a given function  $F(x)$ .

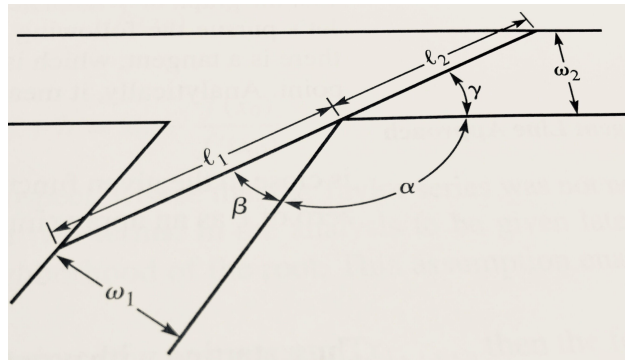
$$x_0 = \text{given}; \quad x_{n+1} = x_n - \frac{F(x_n)}{F'(x_n)} \quad n = 0, 1, 2, \dots, N$$

A good way to decide convergence is to test both the relative change and the residual:

$$|1 - x_{n+1}/x_n| < TOL, \quad |F(x_n)| < TOL$$

The main program should read in initial guess for the root  $x_0$ , and tolerance for testing convergence  $\epsilon$  (these could also be set in the code, if you prefer). Your program should output the input values ( $x_0$  and  $\epsilon$ ) at the beginning, then at each iteration print out:  $n$ ,  $x_n$ ,  $F(x_n)$  and, upon convergence, the number of iterations  $n$ , the root  $x_n$  and the residual  $|F(x_n)|$ . The values of  $F(x_n)$  and  $F'(x_n)$  may be computed in a function or two separate functions.

- Set  $TOL = 10^{-12}$  and find the real root of the “Newton's cubic”  $x^3 - 2x - 5 = 0$  (the only equation Newton ever bothered to solve with his method, in 1671 !)
- In a building, two intersecting halls with widths  $w_1 = 9$  feet and  $w_2 = 7$  feet meet at an angle  $\alpha = 125^\circ$ , as shown below. Assuming a two-dimensional situation (i.e., ignoring the thickness of the board), what is the longest board that can negotiate the turn? [**Hints:** Express length of the board  $l = l_1 + l_2$  in terms of the angle  $\gamma$ , then find the maximum of the function  $l(\gamma)$  by solving the nonlinear equation  $l'(\gamma) = 0$ . ]



- Commissioner Gordon has been found dead in his office. At 8:00pm, the county coroner determined the core temperature of the corpse to be  $90^\circ\text{F}$ . One hour later, the core temperature had dropped to  $85^\circ\text{F}$ . Maintenance reported that the building's air conditioning unit broke down at 4:00pm. The temperature in the commissioner's office was  $68^\circ\text{F}$  at that time. The computerized climate control system recorded that the office temperature rose at a rate of  $1^\circ\text{F}$  per hour after the air conditioning stopped working. Captain Furillo believes that

the infamous Doc B killed the commissioner. Doc B, however, claims that he has an alibi. Lois Lane was interviewing him at the Daily Planet Building, just across the street from the commissioner's office. The receptionist at the Daily Planet Building checked Doc B into the building at 6:35pm, and the interview tapes confirm that Doc B was occupied from 6:40pm to 7:15pm. Could Doc B have killed the commissioner?

---

**Hints:** Assume that the core temperature of the corpse was 98.6°F at the time of death and began decreasing immediately obeying Newton's Law of Cooling. Let  $T(t)$  denote the temperature (°F) of the corpse at time  $t$  (hr). Set  $t = 0$  to correspond to 8:00pm. Then

$$\frac{dT}{dt} = -k(T - 72 - t), \quad T(0) = 90, \quad T(1) = 85$$

Now, to answer the question you need to determine the exact time of death. That is, find  $t$  such that  $T(t) = 98.6^\circ\text{F}$ .

### III. Newton Method, Fractals and Julia Sets

1. Write a function `newton()` to implement Newton's method for finding a root of a given function  $F(x)$ .

$$x_0 = \text{given}; \quad x_{n+1} = x_n - \frac{F(x_n)}{F'(x_n)} \quad n = 0, 1, 2, \dots, N_{\max}$$

A good way to decide convergence is to test both the relative change and the residual:

$$|1 - x_{n+1}/x_n| < TOL, \quad |F(x_n)| < TOL$$

The main program should read in initial guess for the root  $x_0$ , and tolerance for testing convergence  $\epsilon$  (these could also be set in the code, if you prefer). Your program should output the input values ( $x_0$  and  $\epsilon$ ) at the beginning, then at each iteration print out:  $n$ ,  $x_n$ ,  $F(x_n)$  and, upon convergence, the number of iterations  $n$ , the root  $x_n$  and the residual  $|F(x_n)|$ . The values of  $F(x_n)$  and  $F'(x_n)$  may be computed in a function or two separate functions.

- a. Set  $TOL = 10^{-12}$  and find the real root of the "Newton's cubic"  $x^3 - 2x - 5 = 0$  (the only equation Newton ever bothered to solve with his method, in 1671 !)
- b. Use it to find the three roots of the cubic (from Lab 4)  $x^3 + x^2 - 3x - 3 = 0$ . Is Newton method more efficient than Bisection?

**2. Fractals from Newton Iterations.** It is interesting to examine the convergence of Newton method to find the roots for various starting points. Theory tells us that each root has a certain "region of attraction", such that if the initial guess lies in this region then the iteration will converge to the target root eventually. What happens near the borders of such regions belonging

to different roots? Is there a clear demarcation between regions? Let's explore it, for functions  $F(z)$  of a complex variable  $z = x + iy$ . Since  $z$  is representable as a pair  $(x, y)$  of real numbers, we can visualize the regions of attraction on the plane. Consider a function  $F(z)$  with several roots, e.g.  $F(z) = z^3 - 1$  (its three roots are 1 and  $\frac{1}{2}(-1 \pm i\sqrt{3})$  lying on the unit circle). Pick a window (rectangle  $[a, b] \times [c, d]$ ) containing all the roots inside it, e.g.  $W = \{(x, y) : -1.2 \leq x \leq 1.2, -1.2 \leq y \leq 1.2\}$ ; pick one of the roots as the “target root”, e.g.  $Z_{tgt} = 1 = 1 + i * 0$ ; pick an “escape radius”, e.g.  $R = 0.2$ ; pick a number of iteration to track, e.g.  $N_{\max} = 20$ ; and finally pick a number of points to be generated inside the window, e.g.  $M = 50$ . Subdivide  $[a, b]$  and  $[c, d]$  into  $M$  subintervals each, thus generating a 2-dimensional grid of  $(M + 1) \times (M + 1)$  points  $(x_i, y_j)$ ,  $i, j = 0, \dots, M$ . For each of these grid points in  $W$ , we construct the following **Escape-Time Algorithm** for Newton iterations:

Taking a grid point  $z$  as starting value, calculate the Newton iterates

$$z_0 = z, \quad z_{n+1} = z_n - F(z_n)/F'(z_n), \quad n = 0, \dots, N_{\max}.$$

If an iterate gets within  $R$  of  $Z_{tgt}$ , then we consider it converged, and we output the coordinates  $x_i, y_j$  of the starting point. If all  $N_{\max}$  iterates lie outside the disk of radius  $R$  centered at  $Z_{tgt}$ , then we say the iterates of this point escape up to “time”  $N_{\max}$ , and we don't output the starting point (or print in another file for plotting).

- a. Implement this “Escape-time Algorithm” for the function  $F(z) = z^3 - 1$ , in a Python code. It should accept as data:  $M, R, N_{\max}, a, b, c, d$  (you may hard code  $a, b, c, d$  and read the rest from a “data” file). Set the target root in the code (so can use formulas). e.g.,  
`Ztgt = complex( 1.0, 0.0 )`
- b. To debug, use  $Z_{tgt} = 1, R = 0.2, N_{\max} = 20, W$  as above, and  $M = 10$ , then  $M = 200$ . Output the “converging” points  $x_i, y_j$  into a file “out1”. Plot the file from  $M = 200$ . You should see “fractal” structures along the  $\pm 60$  and  $180$  degree rays!
- c. Play around with  $R, N_{\max}, M$ , describe what happens as you vary each. Record your observations.
- d. Now fix the parameters:  $M = 400, R = 0.2, N_{\max} = 20$ , and do it for each of the roots, outputting in separate files `'out1.txt'`, `'out2.txt'` and `'out3.txt'`.
- e. Plot all three files together, with points or with dots. See the regions of attraction of each root by color and observe how intricately interwinded their boundaries are! These are the **Fractal sets**.
- f. Finally, output the plot to a PNG file (equivalent to JPG) to be included in your report.

h. Repeat a) to f) with the function  $F(z) = (z+1)^2(z^2+0.25)$  and the relaxed Newton's method

$$z_0 = z, \quad z_{n+1} = z_n - 2F(z_n)/F'(z_n), \quad n = 0, \dots, N_{\max}.$$

(The roots are  $-1, \pm 0.5i$ )

**3. The Julia set** associated with the complex function  $f(z) = z^2 + c$  may be depicted using the following algorithm.

For each point,  $z_0$ , in the complex plane with  $-1.5 \leq \text{Re}(z_0) \leq 1.5$  and  $-1.5 \leq \text{Im}(z_0) \leq 1.5$ , iterate according to  $z_{n+1} = z_n^2 + c$ . Color the pixel in an image corresponding to this region of complex plane according to the number of iterations required for  $|z|$  to exceed some critical value,  $|z|_{\max}$  (or black if this does not happen before a certain maximum number of iterations  $n_{\max}$ ).

Plot the Julia set for  $c = -0.1 + 0.65j$  using  $|z|_{\max} = 10$  and  $n_{\max} = 500$ .

## IV. Least Squares Fitting

1. A civil engineer is monitoring a leaning tower by measuring the angle it makes with a vertical line. The engineer knows that the tower will collapse if the angle were to exceed  $23^\circ$ . Over the years he has recorded the following data:

Angle $x_i$ ( $^\circ$ )	5	6	9	11	11	13.5	14	16	17
Year $y_i$	1960	1965	1970	1975	1980	1985	1990	1995	2000

Use the least-squares regression to fit the data to a straight line  $y(x) \approx a_0x + a_1$  and compute the sum of squared error  $S_r$ , the standard error of estimate  $S_{y/x}$  and the coefficient of correlation  $r$ . Plot both the data points and the linear fit on the same axes. Use the line's equation so obtained to predict when the tower will collapse.

$$S_r = \sum_{k=0}^{n-1} (y_k - a_0x_k - a_1)^2, \quad S_{y/x} = \sqrt{\frac{S_r}{n-2}}, \quad r = \frac{n \sum x_k y_k - (\sum x_k)(\sum y_k)}{\sqrt{n \sum x_k^2 - (\sum x_k)^2} \sqrt{n \sum y_k^2 - (\sum y_k)^2}}$$

(i) Method 1. Compute the slope and intercept of the line  $y = a_0x + a_1$  using the following formulas.

$$a_0 = \frac{n \sum x_k y_k - \sum x_k \sum y_k}{n \sum x_k^2 - (\sum x_k)^2}, \quad \bar{x} = \frac{1}{n} \sum_{k=0}^{n-1} x_k, \quad \bar{y} = \frac{1}{n} \sum_{k=0}^{n-1} y_k, \quad a_1 = \bar{y} - a_0 \bar{x}$$

(ii) Method 2. Find the values of  $a_0$  and  $a_1$  by solving the linear system  $X\mathbf{a} = \mathbf{y}$  in the least squares sense. ( $X$ : design matrix,  $\mathbf{y}$ : observation vector)

$$\begin{bmatrix} x_0 & 1 \\ x_1 & 1 \\ \cdot & \cdot \\ \cdot & \cdot \\ x_{n-1} & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \cdot \\ \cdot \\ y_{n-1} \end{bmatrix}$$

The least squares approximation to this over-determined linear system is equivalent to solving the following system (normal equations).

$$(X^T X)\mathbf{a} = X^T \mathbf{y}$$

Define  $X^T X = M$  and  $X^T \mathbf{y} = \mathbf{b}$  and solve the  $2 \times 2$  linear system  $M\mathbf{a} = \mathbf{b}$  using `numpy.linalg.solve`.

2. A quick way of fitting a function of the form

$$f(x) \approx \frac{a + bx}{1 + cx}$$

is to apply the least-squares method to the problem  $(1 + cx)f(x) \approx a + bx$ . Use this technique to fit the **world population** data given here (in billions):

Year	Population
1000	0.340
1650	0.545
1800	0.907
1900	1.61
1950	2.56
1960	3.15
1970	3.65
1980	4.20
1990	5.30
2000	6.12
2010	6.98
2020	7.75

Plot the points and the approximated function  $f(x)$  on the same plot. Does it seem to fit well? Determine when the world population becomes infinite!

**Hint:** Write  $a + bx_i - cx_i y_i = y_i$ ,  $i = 0, 1, 2, \dots, n-1$  and obtain a  $n \times 3$  system of equations  $X\mathbf{a} = \mathbf{y}$  and solve it in the least square sense  $[(X^T X)\mathbf{a} = X^T \mathbf{y}]$

$$\begin{bmatrix} 1 & x_0 & -x_0 y_0 \\ 1 & x_1 & -x_1 y_1 \\ \vdots & \vdots & \vdots \\ 1 & x_{n-1} & -x_{n-1} y_{n-1} \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{bmatrix}$$

## V. Numerical Methods for Integration and ODEs

1. Implement the following numerical methods to approximate the definite integral  $\int_0^1 e^{-x^2} dx$ .

a. Midpoint rule:

$$\int_a^b f(x)dx \approx h \sum_{i=1}^N f(x_i), \quad h = \frac{b-a}{N}, \quad x_i = a + \frac{2i-1}{2}h.$$

b. Trapezoid rule:

$$\int_a^b f(x) dx \approx h \left[ \frac{1}{2}f(x_0) + \sum_{i=1}^{N-1} f(x_i) + \frac{1}{2}f(x_N) \right], \quad h = \frac{b-a}{N}, \quad x_i = a + ih$$

c. Simpson's rule:

$$\int_a^b f(x)dx \approx \frac{h}{3} \left[ f(x_0) + 4 \sum_{i=1,3,5}^{N-1} f(x_i) + 2 \sum_{i=2,4,6}^{N-2} f(x_i) + f(x_N) \right], \quad h = \frac{b-a}{N}, \quad x_i = a + ih.$$

d. 2-point Gauss quadrature rule:

$$\int_a^b f(x)dx \approx \frac{h}{2} \sum_{i=1}^N [f(x_l) + f(x_r)], \quad x_l = \frac{1}{2} \left( x_i + x_{i-1} - \frac{h}{\sqrt{3}} \right), \quad x_r = \frac{1}{2} \left( x_i + x_{i-1} + \frac{h}{\sqrt{3}} \right)$$

$$h = \frac{b-a}{N}, \quad x_i = a + ih$$

Comparing against the true solution  $\frac{1}{2}\sqrt{\pi}\text{erf}(1)$ , which method gives the best approximation to  $\pi$ ? How accurate did you get? Display your results (error of approximation for different values of  $N$ ) both graphically and in tabular form.

2. Length and width measurements (in feet) of an irregularly shaped land is shown in the table below. Suppose you want to lay down a uniform 3-inch covering of pine bark mulch over the entire plot. Home improvement store sells bags containing 3 cubic feet of bark mulch, how many bags do you need? Use the Simpson's rule to approximate area.

$x$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$y$	8	10	11	15	16	16	16	16	15.5	15.5	15.5	15	15	15	14.5	14.5	14

$x$	17	18	19	20	21	22	23	24	25	26
$y$	14	14	13.5	13	13	13	12	11	19	6

3. The logistic model is used to simulate population as in

$$\frac{dp}{dt} = k_{\text{rm}} \left( 1 - \frac{p}{p_{\text{max}}} \right) p$$



where  $p(t)$  is the population at time  $t$ ,  $k_{\text{rm}}$  is the maximum growth rate under unlimited conditions, and  $p_{\text{max}}$  is the carrying capacity. Simulate the world's population from 1950 to 2020 using the Euler method. Employ the following initial conditions and parameter values for your simulation:  $p_0$ (in 1950) = 2555 million people,  $k_{\text{rm}} = 0.026/\text{yr}$ , and  $p_{\text{max}} = 12,000$  million people. Have the function generate output corresponding to the dates for the following measured population data. Develop a plot of your simulation along with the data.

$t$	1950	1960	1970	1980	1990	2000	2010	2020
$p$	2555	3040	3708	4454	5276	6079	6922	7753

## VI. Numerical Methods for Linear Systems.

1. Write a code in Python to implement the **Jacobi** method

$$x_i^{(k)} = \frac{1}{a_{ii}} \left[ b_i - \sum_{j=1, j \neq i}^n a_{ij} x_j^{(k-1)} \right] \quad (1 \leq i \leq n)$$

and the **Gauss-Seidel** method

$$x_i^{(k)} = \frac{1}{a_{ii}} \left[ b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^n a_{ij} x_j^{(k-1)} \right] \quad (1 \leq i \leq n)$$

to solve a linear system  $Ax = b$ , and test your code on the following problem. Iterate it enough times to get an accuracy of 6 decimal places.

$$A = \begin{pmatrix} 7 & 1 & -1 & 2 \\ 1 & 8 & 0 & -2 \\ -1 & 0 & 4 & -1 \\ 2 & -2 & -1 & 6 \end{pmatrix}, \quad b = \begin{pmatrix} 3 \\ -5 \\ 4 \\ -3 \end{pmatrix}$$

Hint: The exact solution is  $x = (1, -1, 1, -1)^T$ .

2. Use your code (both **Jacobi** and **Gauss-Seidel** methods) to solve the following sparse system within six correct decimal places (forward error in infinity norm) for  $n = 100$  and  $n = 100,000$ . The correct solution is  $x = (1, \dots, 1)^T$ . Report the number of steps needed and the backward error.

$$A = (a_{ij}), \quad a_{ij} = \begin{cases} 3, & i = j, \quad j = 1, 2, \dots, n, \\ -1, & i = j + 1, \quad j = 1, 2, \dots, n - 1, \\ -1, & j = i + 1, \quad i = 1, 2, \dots, n - 1, \\ 0, & \text{otherwise} \end{cases} \quad b = \begin{pmatrix} 2 \\ 1 \\ \vdots \\ 1 \\ 2 \end{pmatrix}.$$

**VII. Special Topic.** If you want to work on your own project, please talk to me. I need a brief description of the problem you are planning to work on.