Math 248: Computers and Numerical Algorithms, Spring 2025
Project Two, Made Available 04/03, Due 05/11

The Projects: Since students in this class have a diverse range of interests, I am presenting to you a range of potential projects you can work on. If you wish to do an alternate project that seems appropriately numerical, then suggest it to me and I'll let you know if I will accept it. I am very happy to discuss the different projects and give help as necessary. The Lotka-Volterra problem would be particularly useful if you have a computational modeling requirement in your major.

Project Teams: You may work in teams of **up to three** people. Working in groups is not required, but if you do at most three in a group, and each group should submit one project.

Specifications: Your project will consist of four parts:

1. **A written description of the work.** This will constitute approximately forty percent of your grade, and can be written in any text editor you like. Word is fine, but if you know LaTeX, use it. The following outline should be followed for each part.
   (a) **A title page.** This should at minimum include a title, date, and the names of the group.
   (b) **A statement of the problem in your own words.**
   (c) **A description of the solution procedure.** You should state the methods used to solve the problem, identifying any special aspects of the implementation. An english outline of any algorithms used should be included here, including any relevant mathematical discussion. If any equations are required, include them on separate lines with equations numbers, which makes referencing them in the text particularly easy. Pictures, tables or graphs may be included as appropriate.
   (d) **Discussion of results and conclusions.** Include relevant results in an appropriate form, which may include tables, graphs or other relevant pictures. Explain your results and discuss what their relevance may be. Address any troubles or difficulties you encountered.
2. **Computer Programs.** Approximately forty percent your grade will be for the Matlab functions you produce to form your solutions. The text should be included in your submission, and I will also ask one member of each group to submit them electronically to me. The names of all members of your team should be included in each function, as well as the standard required elements in the homework assignments so far. Functions will be graded first on performance (do they work), and second on programming style.
3. **Honor Pledge.** The final part of the write-up will be a statement of the pledge that the material is your group's work. Any additional sources used should be cited. All members of each group are expected to contribute to the project, and I expect you to include a brief summary of the contributions of various group members to the project.

SUMS: Every fall, the department runs SUMS, the Shenandoah Undergraduate Mathematics and Statistics conference. I encourage any group who feels they have done a decent job to prepare a poster for presentation at the conference, and will be happy to guide them through the process. It looks great on the CV!

Getting help: Due to the deliberately open-ended nature of most of the projects, I would hope that the groups will spend plenty of time asking me questions about what you are doing and what your progress is. But remember, I am deliberately setting projects where I don't know all the answers.

Probing the unknown is what most mathematicians do, and I hope this will give you a taste of how it works.

Final thoughts: Projects are expected to be challenging, and should take you a large amount of time to complete. To minimize all-nighters close to the deadline, it is essential to get started as soon as possible. Note that as well as developing programming skills, technical writing skills are an important aspect of these projects. Please proofread your work, and read each other's contributions to make sure that the point is clearly made.

Project #1 – High Precision Arithmetic

We have seen how Matlab limits us to roughly sixteen digits of accuracy for real numbers. It would be nice to be able to have an arbitrary number of digits available. The aim of this project is to build a set of routines to do arithmetic in arbitrary precision with natural numbers.

The first requirement is a representation of a high precision natural number. I would suggest a vector, where the first element holds the units digit, the second element holds the tens digit, and so on. Storing digits in reverse order will make arithmetic much easier.

- To achieve a C grade, you should write functions that display high precision natural numbers, as well as perform subtraction of a smaller natural number from a larger. The solution should be exact, so the number of digits output may vary. You should include in your solution several examples showing that your code is working properly.

  Here is an example of a function to add natural numbers that is a useful starting point. You will have to think a bit about subtraction, and one easy approach would be to just subtract digits, and later if a digit is negative do a carry in reverse. For example, $342 - 277 = (1(-3)(-5))$. Then adding ten ones and subtracting one ten gives $(1(-4)(5)$. Then adding ten tens and subtracting one hundred gives (065) or 65. Note you have to remove any leading zeros.

  ```
  function c=sumd(a,b)
  % Function to add the numbers represented by individual digits in the
  % arrays a and b, least significant digit first, and output them in c.
  % Written by Stephen Lucas, October 2009
  % Variables are:
  % m,n = numbers of digits in a,b
  % o = biggest of m and n, initial length of c
  % i = loop counter

  m=length(a); n=length(b);
  % Put the largest number in c, then add the smaller
  if m>n
    c=a; o=m;
    for i=1:n, c(i)=c(i)+b(i); end
  else
    c=b; o=n;
    for i=1:m, c(i)=c(i)+a(i); end
  end
  % Now do the carry
  ```

```
for i=1:o-1
  if c(i)>9, c(i)=c(i)-10; c(i+1)=c(i+1)+1; end
end
% It's a special case if the last digit carries to a new digit
if c(o)>9, c(o)=c(o)-10; c(o+1)=1; end
```

- To achieve a B grade, you should also write a function that compares two numbers, and returns $1, 0, -1$ if the first number is greater than, equal to, or less than the second number. A first check of the number of digits may be enough, but if they are the same length, you will need to compare digits.

  Then write a function that performs natural number multiplication. You could take the product of every pair of digits, put them in the appropriate place, then do extensive carries. Show these functions work with some examples.

- To achieve an A grade, you should also write a function that performs natural number division. The function should return both the quotient q and remainder r: if `[q,r]=div(a,b)`, then $a = qb + r$ with $0 \leq r < b$. Show that it works with some examples. A greatest common divisor code for large numbers would be a lovely bonus.

  Note that the standard long division algorithm is quite challenging to implement. I would suggest as an easier option to try and build Egyptian division, which is all about doubling the divisor until it is too big, then subtracting (look it up!). So you will use the codes you have already written to manipulate the numbers.

Project #2 – The Integer Valued Logistic Equation

In class we saw how an equation modeling the population of a species with a carrying capacity could be rewritten as an equation where the variable lay between zero and one, and if we assumed it took a real value, then chaos was possible, where the period doubled as the reproduction rate increased. This is *the* classic examples of a simple chaotic system, and is the first example in nearly every book on chaos. However, true populations take integer values, not reals. Does this make a difference? This hasn't been looked at much before, and I only know some of what will happen! The aim of this project is to investigate what happens to the logistic equation when we assume only integer values are possible. The equation assuming a carrying capacity C and reproduction rate λ is $x_{n+1} = x + n + \lambda x_n(C - x_n)$ with $0 \leq \lambda \leq 3/C$ and $0 \leq x_n \leq C + 1/\lambda$.

- To achieve a C grade, you need to be able to reproduce the bifurcation diagram (where x_i lies between zero and one), that can be seen in every description of chaos on the web. One way of doing so would be to choose a large number of equally spaced r values between one and four, and for each r value run the logistic equation for five hundred steps, then save the next five hundred values and put them on a graph. Then repeat this for the equation where population lay between zero and the carrying capacity (still a real number), and you vary the reproduction rate λ. Choose some large C that works for you. Do you get the same sort of picture? Does the period double in the same way leading to chaos?

- To achieve a B grade, you should also choose a particular carrying capacity, and assume the population can only take integer values, so you will have to either take the floor, ceiling, or round the result. Actually, trying all three choices would be nice. Since the population can only take integer values, the solution will be eventually periodic regardless of the reproduction rate. But does period doubling and a solution that looks like chaos actually happen? And if period doubling happens, how close is it to the values that happen in the real case? A picture and discussion would be appropriate.

- To achieve an A grade, you should also vary the carrying capacity and see what effect it has – clearly the larger the carrying capacity, the closer the integer solution should be to the real one. Are there other questions you can address about the behavior of the integer valued solution? I know that funny things happen when your initial population is very close to zero or the carrying capacity. This is completely open territory.

Project #3 – Diffusion on a Grid

Brownian motion is the process by which a speck of dust in the sky or a speck of cream on the top of a cup of coffee appears to move at random. It is of incredible importance in the fields of physics, biology and financial mathematics, to name a few. The aim of this project will be to investigate some of the properties of Brownian motion. The Matlab function `rand`, which returns a random number between zero and one, will be incredibly useful here.

- To achieve a C grade, you need to write code that simulates one dimensional Brownian motion. Assume that your particle starts at the point zero, and at every step has a fifty percent chance of moving to the left and a fifty percent chance of moving to the right. Problems to be solved are:
  - It is known that the particle almost always returns to zero. Write a function that for a particular simulation calculates how many steps are required before the particle returns to zero for the first time as well as the maximum distance away from the origin achieved. After running the function a large number of times, what is the average number of steps and its distribution? What is the distribution of the maximum distance achieved? The Matlab function `bar` may be of use to you here.
  - Write a function that takes as input the number of steps to take, and returns the position at the end of that number of steps. After running the function a large number of times form a histogram of the position of the particle after one hundred steps. Do you recognize it? Does it change appreciably if you go two hundred or more steps?
- To achieve a B grade, you also need to write code that simulates two dimensional Brownian motion. Assume that your particle starts at the origin, and at every step has a twenty five percent chance of moving up, left, down, or right. Problems to be solved are:
  - It is known that the particle almost always returns to the origin. Write a function that for a particular simulation calculates how many steps are required before the particle returns to the origin for the first time, as well as how far from the origin it gets (as a distance). After running the function a large number of times, what is the average number of steps and its distribution (use a histogram)? What is the distribution of the maximum distance achieved?
  - Write a function that takes as input the number of steps to take, and returns the distance from the origin at the end of that number of steps. After running the function a large number of times form a histogram of the distance after one hundred steps. Do you recognize it? Does it change appreciably if you go two hundred or more steps?

- To achieve an A grade, there are a number of possible ways to extend what we have done so far, and I have no idea what the answers will be in the cases I list below. As well as anything interesting you can come up with on your own, possibilities are:
  - In the one-dimensional case, make your Brownian motion biased – p percent to the left, $100 - p$ percent to the right. I don't believe are still guaranteed to always be able to get back to the origin. For a few values of p, how many steps are required to return to the origin, given some maximum value beyond which we shall assume it never returns. The return fraction as a function of p would be interesting to see, where hopefully the return fraction is extremely close to one when $p = 50$.
  - There are other uniform grids on the plane apart from the square one we used for two dimensional Brownian motion. Two example are a triangular (equilateral) grid and a regular hexagon grid. Can you work out how to encode the location of each potential particle position in one or both of these grids, as well as where they can move? If you can, repeat the above. I have no idea if they are guaranteed to return to the origin or not!
  - Instead of simulating individual cases, you could try and do all of them at once, storing the probability of the particle being at a particular position at any particular moment. In one dimension, if at step j the probability of being at location i is $p_{i,j}$, then at step $j+1$, we add $p_{i,j}/2$ to each of $p_{i+1,j+1}$ and $p_{i-1,j+1}$. What sort of graphs and interesting results can you get? How about in two dimensions?
  - Not for the fainthearted, and probably shouldn't be included here, but what the hey! This would qualify as beyond A, and there would possibly be a research publication in it. There are a number of "random" regular tilings of the plane, including Penrose's darts and kites – see `http://en.wikipedia.org/wiki/Penrose_tiling`. Is it possible to randomly come up with a tiling and some way of recognizing which vertices a particle can move along. Then starting from the origin and moving at random, what happens to Brownian motion? No-one knows!

Project #4 – Chaos in Newton's Method

Newton's method converges to the nearest zero when the tangent at the initial guess points to the zero. But odd things can happen near points with horizontal tangent.

- To achieve a C grade, apply Newton's method to $f(x) = x(x-1)(x+1/2)$ for a very large number of initial guesses on $[-2, 2]$, and plot which of the three zeros it converges to as a function of initial point. You might notice some points where it jumps around a bit. Zoom in on those places (choose lots of points on a smaller range of initial guesses) and show that even as you make the interval smaller, the jumpiness of the solution looks similar. This means we have a self-similar, or chaotic behavior.

- To achieve a B grade, apply Newton's method to $f(x) = x^3 - 1$ with initial guess a selection of complex numbers with real and imaginary parts varying from -2 to 2. The solution will be one of the cube roots of one: 1, $-1/2 + i\sqrt{3}/2$, $-1/2 - i\sqrt{3}/2$. You don't need to do anything fancy in Matlab – it automatically deals with complex numbers ($i = \sqrt{-1}$). You can then make a plot where the x-axis is the real part of the initial guess, the y-axis is the imaginary part, and the plot is colored based upon which of the three roots you converge to. The picture should be a fractal that looks like the page on Newton's method on Wikipedia. The Matlab `patch` function might be useful for generating pictures.

- To achieve an A grade, make a selection of pictures zooming in where the picture has more detail, and verify that the detail continues as you zoom in – it is a fractal. Does the same thing happen with other polynomials on the complex plane? Can you find other function with this behavior?

Project #5 – Project Euler

Project Euler (`http://projecteuler.net/`) is a set of mathematical/computational problems of varying levels of difficulty, several of which you have already solved without realizing it! The first dozen problems are:

1. If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23. Find the sum of all the multiples of 3 or 5 below 1000.
2. Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be: $1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots$. Find the sum of all the even-valued terms in the sequence which do not exceed four million.
3. The prime factors of 13195 are 5, 7, 13 and 29. What is the largest prime factor of the number 600851475143?
4. A palindromic number reads the same both ways. The largest palindrome made from the product of two 2-digit numbers is $9009 = 91 \times 99$. Find the largest palindrome made from the product of two 3-digit numbers.
5. 2520 is the smallest number that can be divided by each of the numbers from 1 to 10 without any remainder. What is the smallest number that is evenly divisible by all of the numbers from 1 to 20?
6. The sum of the squares of the first ten natural numbers is $1^2 + 2^2 + \cdots + 10^2 = 385$. The square of the sum of the first ten natural numbers is $(1 + 2 + \cdots + 10)^2 = 55^2 = 3025$. Hence the difference between the sum of the squares of the first ten natural numbers and the square of the sum is $3025 - 385 = 2640$. Find the difference between the sum of the squares of the first one hundred natural numbers and the square of the sum.
7. By listing the first six prime numbers, 2, 3, 5, 7, 11, and 13, we can see that the 6th prime is 13. What is the 10001st prime number?
8. Find the greatest product of five consecutive digits in the 1000-digit number (see the website).
9. A Pythagorean triplet is a set of three natural numbers, $a < b < c$, for which, $a^2 + b^2 = c^2$. For example, $3^2 + 4^2 = 9 + 16 = 25 = 5^2$. There exists exactly one Pythagorean triplet for which $a + b + c = 1000$. Find the product abc.
10. The sum of the primes below 10 is $2 + 3 + 5 + 7 = 17$. Find the sum of all the primes below two million.
11. In the 20×20 grid (on the website), four numbers along a diagonal line have been marked in red. The product of these numbers is $26 \times 63 \times 78 \times 14 = 1788696$. What is the greatest product of four adjacent numbers in any direction (up, down, left, right, or diagonally) in the 20×20 grid?
12. The sequence of triangle numbers is generated by adding the natural numbers. So the 7th triangle number would be $1 + 2 + 3 + 4 + 5 + 6 + 7 = 28$. The first ten terms would be $1, 3, 6, 10, 15, 21, 28, 36, 45, 55$. Let us list the factors of the first seven triangle numbers: 1: 1. 3: 1,3. 6: 1,2,3,6. 10: 1,2,5,10. 15: 1,3,5,15. 21: 1,3,7,21. 28: 1,2,4,7,14,28. We can see that 28 is the first triangle number to have over five divisors. What is the value of the first triangle number to have over five hundred divisors?

The aim of this project is to solve as many Project Euler problems as possible (registering means you can check if you have the right answer). Your write-up would explain how you solved each problem as well as the right answer. I estimate that solving about fifteen problems would be worth a B (you do have six weeks), and about twenty five would be worth an A.

Project #6 – Convex Hull Algorithms

The convex hull of a set of points on the plane is the smallest convex polygon that encloses all the points. You can visualize it as placing nails in a board at the points, then letting go a larger elastic band around the outside. The convex hull is thus a collection of some of the points in some order. The aim of this project is to implement and test some simple algorithms for finding the convex hull of a set of points.

- To start, the three points (x_1, y_1), (x_2, y_2) and (x_3, y_3) form a left/right hand turn if the point (x_3, y_3) is to the left/right of the line from (x_1, y_1) to (x_2, y_2). If it's exactly on the line, we call them collinear. While we could find the angle and hence the kind of turn, a little vector calculus tells us that the sign of the cross product of the two vectors defined by (x_1, y_1), (x_2, y_2) and (x_1, y_1), (x_3, y_3) is sufficient. Specifically, define

$$T = (x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1).$$

  If $T > 0$ then there is a left hand turn, if $T < 0$ there is a right hand turn, and if $T = 0$ then the points are collinear. Write a Matlab function `turn` that returns one for a left turn, minus one for a right turn, and zero for collinear points. The input should be the six numbers making up the coordinates of the points.

- To achieve a C grade, you need to implement the "giftwrap" algorithm. Start by identifying the point with the lowest y value. If there are more than one, choose the one with smallest x value. Since it's at the bottom, it must be part of the hull. Then at every step, the next point on the hull (going anti-clockwise) is the point such that you have a left turn from the previous point to the next point to any other point in the set. Continue until you get back to the bottom point. Your output from the function should be a pair of vectors holding the x and y values for the points on the hull in order, and it would be nice if as an option you also produced a plot of the data points and the convex hull surrounding them. An outline of the algorithm might be:

```
Choose lowest point as first point on hull, make it point one, and set done to false
While not done
    Choose point two as first point on list
    For i = two to number of points
        If point one to point two to point i is a right hand turn
            replace point two by point i
      End If
    End For
    If point two equals first hull point, done equals true,
    Else add point two to the end of the convex hull list, and make it point one
    End If
End While
```

- To achieve a B grade, you need to extensively test the giftwrap algorithm. First, the amount of time required to run the giftwrap algorithm can be measured by how many left turn tests are required in finding the convex hull. A first test would involve choosing points uniformly at random on the square $0 < x, y < 1$ (using Matlab's `rand` function). Then, for various choices of the number of points, what sort of distribution (after thousands of tests) do you get for both number of left turn tests as well as number of points in the convex hull. By just looking at averages, make plots of number of turn tests and number of points in the hull as a function of number of points. Do you see anything? Repeat with points normally distributed (check out Matlab's `randn` function), and uniformly distributed (again) with the additional points $(0, 0)$, $(1, 0)$, $(1, 1)$, $(0, 1)$ so the solution will always be a square, and anything else that inspires you.

- To achieve an A grade, you should also implement "Graham's scan," test it, and compare it to giftwrap. Graham's scan is probably the most efficient algorithm for finding the convex hull on a plane currently available. It starts by identifying the bottom point as for giftwrap, and making it the first point in the list by swapping. Then we need to sort the points in terms of angle if the bottom point is the origin. To avoid trigonometry, all the angles are between $0°$ and $180°$, and in this range $\cos\theta$ is a decreasing function. So, for every point (x_i, y_i) for $i > 1$, associate the number $-(x_i - x_1)/\sqrt{(x_i - x_1)^2 + (y_i - y_1)^2}$, and give (x_1, y_1) the number -2. Then sort on these values. To make it easy, I would suggest using Matlab's `sortrows` where you put these numbers in the first columns, and x and y values in the second and third columns.

  Now that the data is sorted, the first two points and the last point are definitely in the hull, so add the last point to the beginning. If your data is in a $3 \times n$ array `data`, then the command `data=[data(n,:); data];` will do the trick. Now we are ready to scan. The basic idea is that if three points in a row form a right turn, the middle point is not part of the hull and should be removed from the hull. An algorithm for this part might be:

  ```
  Start with M, the number of points so far in the hull, as three.
  For i = four to N+1
      While points M-1, M and i form a right hand turn, let M=M-1, End While
      Let M=M+1, and swap point M with point i
  End For
  ```

  Now, the points from 2 to M should be the hull, since the last point is repeated. This should be output from the function. Its really quite cunning how swapping points identifies the current hull without needing an additional array.

Project #7 – The Lotka-Volterra Equations

The Lotka-Volterra equations, also known as the predator-prey equations, describe the dynamics of a biological system where two species interact, one a predator, the other the prey. If x is the number of prey (like rabbits) and y is the number of predators (like foxes) then

$$\frac{dx}{dt} = \alpha x - \beta xy, \quad \text{and} \quad \frac{dy}{dt} = \delta xy - \gamma y,$$

where $\alpha, \beta, \delta, \gamma$ are positive real parameters describing the interaction between the two species, and we need some initial values $x(0) = x_0$ and $y(0) = y_0$. The rate of change of the prey increases due to population growth (α), decreases due to being eaten by the predator. The xy term describes how often predator and prey meet. The rate of change of the prey increases when there is an adequate food supply, and reduces due to death. The equations don't have an analytic solution, and so must be solved numerically.

- The simplest possible way of approximating the solution to a differential equation is Euler's method: approximate $\dfrac{dx}{dt}$ at time t by $(x(t+h)-x(t))/h$. Letting $t_i = (i-1)h$, w_{1i} approximate the solution $x(t_i)$, and w_{2i} approximate the solution $y(t_i)$, verify that the Lotka-Volterra equations can be rewritten as

$$\begin{aligned} w_{1,i+1} &= w_{1i} + h(\alpha w_{1i} - \beta w_{1i}w_{2i}), \\ w_{2,i+1} &= w_{2i} + h(\delta w_{1i}w_{2i} - \gamma w_{2i}), \end{aligned}$$

  for $i = 1, 2, \ldots$ with $w_{1i} = x_0$ and $w_{2i} = y_0$.

  To achieve a C grade, write a Matlab function that takes as input a time interval T and the number of pieces N to split the time interval $[0, T]$ into, so $h = T/N$, and outputs the approximate solution to the Lotka-Volterra equations using Euler's method as three vectors of length $N+1$: the time, $w1$ and $w2$. Use the parameters $\alpha = 2$, $\beta = 1$, $\delta = 0.25$, $\gamma = 0.1$, and let's choose $T = 20$, $x_0 = 1$, $y_0 = 1$. Produce plots of x and y as functions of t, as well as x versus y, which with no numerical error, should give a periodic solution.

- To achieve a B grade, choose some initial value of N, and keep doubling it. Compare the solutions at the time T to verify that there is convergence. Theoretically, doubling N should halve the error. Make a table of $x(T)$ and $y(T)$ as you double N, and satisfy yourself that the error is reducing. How big does N need to be to get what looks like three digits of accuracy? Does this give you a nice picture?

  Make a plot of x versus y for various initial values, which should produce a variety of closed non-intersecting curves. Use the output data to identify the period in each case.

- To achieve an A grade: Euler's method is quite poor. Taylor's method improves it by adding an additional terms to the Taylor's series. The next term would give $x(t+h) \approx x(t) + hx'(t) + (h^2/2)x''(t)$, where we take derivatives with respect to time. From the original equations we already have $x' = \alpha x - \beta xy$, and from the chain rule, you should verify that $x''(t) = \frac{d}{dt}(\alpha x - \beta xy) = \alpha x' - \beta xy' - \beta x'y = \alpha^2 x + (\beta\gamma - 2\alpha\beta)xy + \beta^2 xy^2 - \beta\gamma x^2 y$. You can come up with y'' as well, and eventually come up with a pair of new formulas for $w_{1,i+1}$, $w_{2,i+1}$ in terms of more complicated formulas. Implement this new version, and show that as you double N the error goes down by a factor of four: much more efficient.

Project #7 – Anything You Like...

As long as you can convince me that it has merit as a major project for a numerical methods course! Not straight computer science, there has to be a mathematical component.