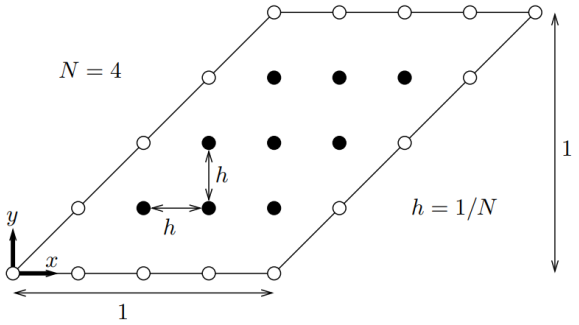
Homework 3

NERS/ENGR 570 Fall 2020
November 23, 2020
DUE BY: 11:59 12/17

Exercise 1: Laplace's Equation (100 pts)

In this problem, you will use finite-differencing to solve Laplace's equation, $\nabla^2 u = 0$, on the skewed domain illustrated below.



N is the number of intervals in each direction, for a total of $(N-1)^2$ points in the domain. The boundary conditions are Dirichlet and prescribed according to the function $u_{BC}(x, y) = xy$. Use the standard 5-point stencil for the internal points.

a) Write a program that sets up and solves a **sparse matrix system** for u at each node. Run your code for $N = 16, 32, 64$ and plot filled contours of u for each result.
b) Now write a program that uses the **red-black Gauss-Seidel** method with under-relaxation to solve for u at each node, starting with the guess $u_0(x;\ y) = 0$ on the interior nodes for a total of 100 iterations. Run your code for $N = 8, 16, 32, 64$ and plot the convergence of the discrete L_1 norm versus iteration (use a semilogy plot). Make three separate figures, one for each N, and on every figure show results using underrelaxation factors of $\omega = 1.0, 1.5, 1.7, 1.8, 1.9$.

You can use python or MATLAB scripts. Examples of the expected figures are shown as follows.

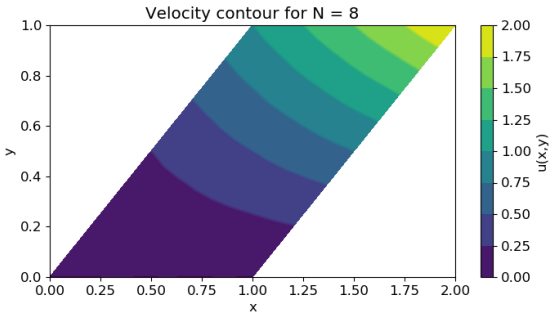


Figure 1.1 Example Solution for Exercise 1a

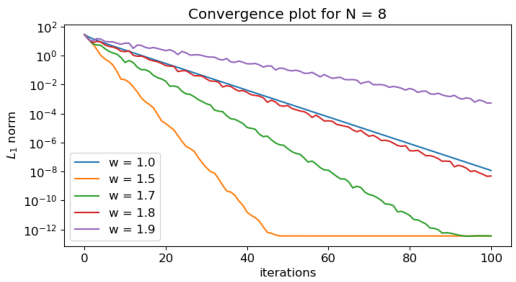


Figure 1.2 Example Solution for Exercise 1b

Exercise 2: Evolution of Plasma (200 pts)

Vlasov-Poisson equation is a differential equation used within plasma simulation to describe the evolution of the plasma (in position and velocity space). The dominating equation to solve this is given by

$$\frac{\partial f}{\partial t} + \vec{v} \cdot \frac{\partial f}{\partial \vec{x}} + \frac{q}{m}\vec{E} \cdot \frac{\partial f}{\partial \vec{v}} = 0, \tag{2.1}$$

where $f(x, v, t)$ is the distribution for the species, and $\frac{q}{m}$ is the charge to mass ratio for that species. In the 1D case,

$$\frac{\partial f}{\partial t} + v\frac{\partial f}{\partial x} + \frac{q}{m}E\frac{\partial f}{\partial v} = 0\,, \tag{2.2}$$

$$\frac{d^2\phi}{dx^2} + \rho = 0, \tag{2.3}$$

$$E = -\frac{d\phi}{dx}, \tag{2.4}$$

$$\rho = q\int f(x, v, t)dv,. \tag{2.5}$$

In this exercise, you need to solve the problem involved with two counter propagating beams with velocity 0.25 of a gaussian profiles with $\sigma = \frac{1}{4}$. The spatial perturbation $\epsilon = 0.45$ and the wavenumber $k = 2$. The phase difference is $\phi = \pi$. To perform the simulation, the problem is discretized in to I intervals in space and J intervals in velocity phase.
In mathematic form, the initial distribution is

$$f[i, j] = \exp\left(-4\left(v_j + 0.25\right)^2\right) \times \left[1 + 0.45\sin\left(4\pi \times \frac{i}{J}\right)\right]$$
$$+ \exp\left(-4\left(v_j - 0.25\right)^2\right) \times \left[1 + 0.45\sin\left(4\pi \times \frac{i}{J}h - \pi\right)\right].$$

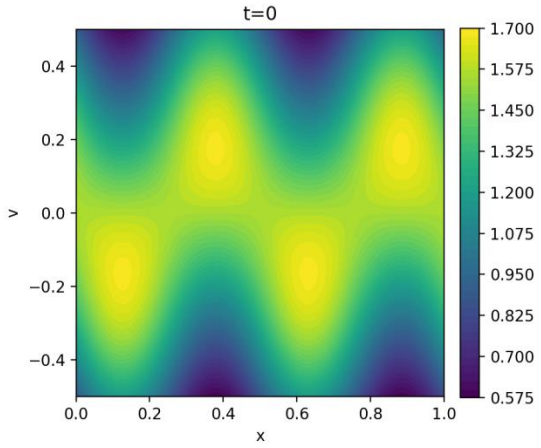


Figure 1.2 Initial distribution of the Plasma

The problem has the range $0 \leq x \leq 1$ and periodic boundary conditions. You can use $\Delta x = \Delta v = 0.01$ and $\Delta t = 0.005$.

a) Write down a discretized version of the problem, i.e. the discretized form of **Eqs 2.2--2.4**. Any discretized form is acceptable.
b) Write a program in C, C++, or Fortran to solve the problem, and plot the distribution of the plasma at $\boldsymbol{t = 1}$. (plotting may be done in any program).

Exercise 3: Determine the Coefficient of Friction (50 pts)

The Darcy-Weisbach equation is used to measure the pressure drop due to friction in an incompressible fluid through a pipe. This equation can be used to approximate the pressure drop as blood flows through the cardiovascular system, since most of the pressure drop is due to viscous effects. The Darcy-Weisbach equation is given below:

$$h_f = \frac{fLV^2}{D2g}, \tag{3.1}$$

where h_f is the pressure drop, f is the friction coefficient, L is the length of the pipe, D is the diameter of the pipe, V is the average velocity of the fluid, and g is acceleration due to gravity. The friction coefficient in the transitional regime between a smooth and rough pipe surface is given by Eq 3.2:

$$\frac{1}{\sqrt{f}} = 1.14 - 2\log\left(r + \frac{9.35}{\mathrm{Re}\sqrt{f}}\right), \tag{3.2}$$

Newto- Raphson is an iterative scheme that can be used to approximate the solution to an equation based on a reasonable first estimate. The update rule to find x given an equation $F(x)$ is as follows:

$$x^{(n+1)} = x^{(n)} - \frac{\frac{F(x^n)}{dF^{(n)}}}{dx}. \tag{3.3}$$

a) Given $F(f) = \frac{1}{\sqrt{f}} - 1.14 + 2\log\left(r + \frac{9.35}{\mathrm{Re}\sqrt{f}}\right)$, write down the expression for $\frac{dF}{df}$
b) Write the code (in C, C++, or Fortran) to calculate the friction coefficient, with initial guess f_0, r and Re as the input.
c) Provide the result when r=0.5 and $Re = 1000$.

Exercise 4: Monte Carlo Spin State (150pts)

One of the most common models used in statistical mechanics/physics is the Ising Model, which is made up of spins on a lattice. An example of a 2D-Ising model on a 5x5 lattice is shown in Figure 4.1. In the simplest formulation, these spins can have one of two values: up or down. More involved models may allow the spins to occur at angles relative to the lattice that they occur on.

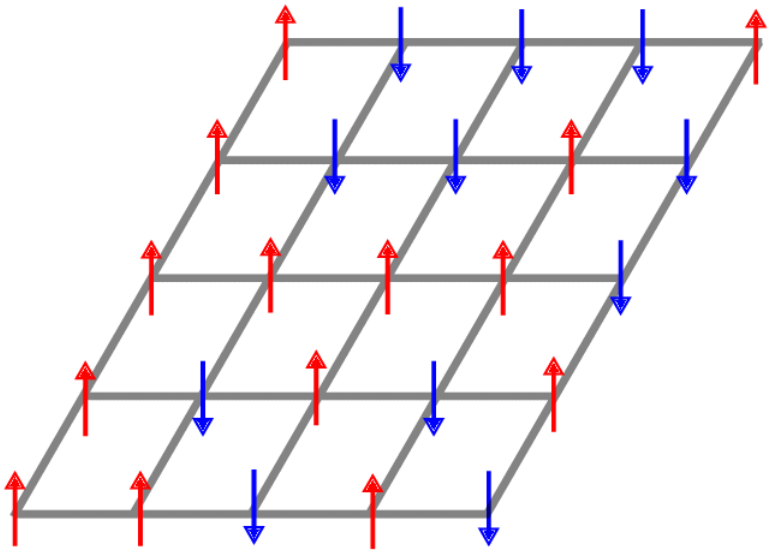


Figure 4.1 Figure 1: Schematic of a 2D-Ising Model

In the Ising system, the Hamiltonian (or energy) function is given by

$$H = -J \sum_{<i,j>}^{N} \vec{s_i} \cdot \vec{s_j} - \sum_{i=1}^{N} \vec{h} \cdot \vec{s_i} \qquad (4.1)$$

where J is the interaction energy between nearest neighbor spins and is greater than 0 for attractive interaction (i.e. up spins want to be next to other up spins), the first summation over $\langle i, j \rangle$ is the summation over each nearest neighbor j at every lattice site i (with each pair just being counted once), $\vec{s_i}$ is the spin at lattice site i (either -1 or +1), and $\vec{h}$ is an applied field that couples with the spin. Historically, the Ising model is used to explore the concept of ferromagnetism in materials, and is one of the simplest models that shows a phase transition.

For the purposes of this problem, we will assume that $\vec{h} = 0$, and thus the only energy contributions in the system come from the nearest neighbor spin interactions and entropy. While we will not derive the procedure through statistical mechanics, it should be evident that without any considerations to entropy, the system will be perfectly ferromagnetic (all spins aligned). However, such a configuration will be quite entropically unfavorable, so as the temperature increases, this entropy penalty becomes more severe, and the ferromagnetic ordering weaker and weaker, until at high enough temperatures a state with no magnetic order (all spins randomly aligned) will be favorable.

As one may imagine from the construction of the model, this problem lends itself nicely to Monte Carlo methods; indeed, while the 1D and 2D systems can be solved exactly, no exact solution exists for the 3D system, and numerical methods must be used. For this problem, you will write an implementation of a Monte Carlo method to find the ground state of the 2D Ising model given a temperature input. The rough procedure of this method is as follows:

i. Begin with a randomly initialized system
ii. Choose a random spin, flip it, and calculate the change in energy of this new system compared to the previous system
iii. Based on a criterion choose to either accept the change or discard it
iv. Repeat until some condition is met

In our case, the criteria in step 3 is where the temperature/entropy comes in. To determine if a step is accepted, the value

$$\frac{1}{1 + e^{\frac{\Delta E}{T}}}$$

is computed (note that T is in Kelvin), which gives the probability of acceptance.
We can see that a flip which significantly lowers the energy will have a high probability of acceptance, but at higher temperatures, even energetically unfavorable flips have a non-zero probability of occurring.

In this exercise, you should:

a) Provide a discussion of "end" conditions. Given that Monte Carlo is a stochastic method, what may be an appropriate stopping condition? Keep in mind that the energy of the system can fluctuate both up and down.
b) Write two helper methods: one to initialize a matrix A with a random distribution of -1's and 1's, and another to calculate the energy of the system for a given value of the interaction energy, J.
c) Write the Monte Carlo algorithm for a 2D Ising lattice in C, C++, or Fortran using OpenMP. It should make calls to the two helper functions written in part 2. Your method should use a matrix A that represents the system, a float/real *j* with the interaction energy J, a float/real *T* for the temperature of the system, an integer *n* which is the size of the $n \times n$ 2D lattice, and any other variables you may need to execute the algorithm. For the purposes of testing, you can set the size of the system to 5x5.

Exercise 5: Neutron Diffusion Equation (150pts)

The diffusion equation is an approximation to the Boltzman Neutron Transport equation that can be more easily solved computationally. The general form of the diffusion equation with vacuum boundary conditions is shown below.

$$-D\nabla^2\phi(x) + \Sigma_r(x)\phi(x) = \frac{\nu\Sigma_f}{k}\phi(x),$$
$$\phi(x) = 0, x \in \Gamma.$$

If we assume that the system is in equilibrium (k=1) and the source is known, we can rewrite the equation as,

$$-D\nabla^2\phi(x) + \Sigma_r(x)\phi(x) = Q(x).$$

Typically, we want to solve a system of equations for two energy groups. However, if we know the source term, we can then analyze only one group.

$$-D_1\nabla^2\phi_1(x) + \Sigma_{r1}(x)\phi_1(x) = Q(x).$$

Another simplification is the assumption that the flux does not vary in the transverse (y and z) directions, making the problem one dimensional in x.

$$-D_1\frac{d^2\phi_1(x)}{dx^2} + \Sigma_{r1}(x)\phi_1(x) = Q(x),$$

Using the central difference for the derivative and assuming a zero flux boundary condition, this becomes,

$$-D_1\left(\frac{\phi_1^{i-1} - 2\phi_1^i - \phi_1^{i+1}}{\Delta x^2}\right) + \Sigma_{r1}\phi_1^i = Q^i$$
$$\phi_1^0 = \phi_1^N = 0$$

This can be written in matrix form as,

$$\begin{bmatrix} \frac{2D_1}{\Delta x^2} + \Sigma_{r1} & -\frac{D_1}{\Delta x^2} & 0 & 0 & 0 \\ -\frac{D_1}{\Delta x^2} & \frac{2D_1}{\Delta x^2} + \Sigma_{r1} & -\frac{D_1}{\Delta x^2} & 0 & 0 \\ 0 & -\frac{D_1}{\Delta x^2} & \frac{2D_1}{\Delta x^2} + \Sigma_{r1} & -\frac{D_1}{\Delta x^2} & 0 \\ 0 & 0 & -\frac{D_1}{\Delta x^2} & \frac{2D_1}{\Delta x^2} + \Sigma_{r1} & -\frac{D_1}{\Delta x^2} \\ 0 & 0 & 0 & -\frac{D_1}{\Delta x^2} & \frac{2D_1}{\Delta x^2} + \Sigma_{r1} \end{bmatrix} \begin{bmatrix} \phi^1 \\ \phi^2 \\ \phi^3 \\ \phi^4 \\ \phi^5 \end{bmatrix} = \begin{bmatrix} Q^1 \\ Q^2 \\ Q^3 \\ Q^4 \\ Q^5 \end{bmatrix}$$

This example shows a 5×5 matrix but can be represented in any n by n matrix.

Please use the following to construct your linear system.

$$D_1(x) = D_1 = 0.238\text{cm}^{-1}, \Sigma_{r1}(x) = \Sigma_{r1} = 0.8\text{cm}^{-1}$$
$$Q(x) = Q = 10^{12} nt/\text{cm}^3/\text{s}$$
$$L = 10\text{cm}$$

In this exercise you should write a code in C, C++, or Fortran to solve the 1D diffusion problem using the central difference formula above and a finite element discretization with linear basis functions (use Lagrange polynomials). Plotting of output may be done in Python. Then do following investigations:

a) Derive the Finite element discretization.
b) For a cell width of 1 cm, compute the solution ϕ to the problem. Plot your results for the scalar flux as a function of distance for finite difference and finite element. Discuss the differences in the solutions.
c) Refine the spatial grid for each for finite element and finite difference down to 0.1 mm. Plot the L2 norm of the error of the solution as a function of mesh size. Use the 0.1mm solution as the reference.
d) Use the new table for the source $Q(x)$ to compute the flux. Again, plot your results and comment on the differences.

x	$Q\ (10^{12}\ nt/cm^3/s)$
[0,2]	1
[2,3]	1.5
[3,7]	2
[7,8]	1.5
[8,10]	1

Exercise 6: Maxwell's Equation (100 pts)

Gauss's Law tells us the potential field is related to the charge density by:

$$-\nabla^2\phi = \rho/\epsilon_0.$$

This is a partial differential equation that allows us to solve for the potential and therefore the E-field in space, as long as we have information on the boundary conditions/charge density. One important application of electrostatics is the coaxial cable. Coaxial cables are formed of two concentric conductors, the outer is grounded while the inner is set to some non-zero potential.

As shown below, model the cross-section of the coaxial cable by two rectangular conductors (the inner having dimensions L-by-W and the outer having dimensions S-by-S). Set the inner conductor to some voltage V, and ground the outer conductor (set potential to 0). That is, use Dirichlet boundary conditions for the outer conductor ($\phi = 0$) and inner conductor ($\phi = 1$).

This problem has two lines of symmetry, shown by the dotted lines. That is the problem remains the same under a reflection across either dotted lines. This means that we can solve the electrostatic problem in only one quadrant, since we know by symmetry that the solution to the Poisson's equation will be the same in all four quadrants. Additionally, from the symmetry of the problem, the potential has to obey Neumann boundary conditions ($\phi' = 0$) along the dotted lines.

a) Implement a finite differences discretization to solve this electrostatic problem using the Alternating direction implicit iterative method in C, C++, or Fortran. The implementation should take a few parameters necessary to describe the problem, and then setup and solve the associated linear system of equations.
b) Write a python script that calls the code developed in part a) and plots the potential distribution for $V = 1, \rho = 1$.

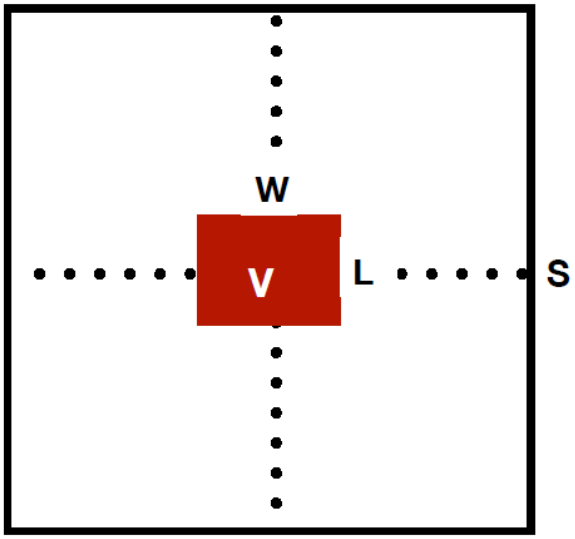


Figure 6.1 Illustration of the simplified coaxial cable model

Exercise 8: Heat Conduction (100 pts)

The heat equation is a partial differential equation that describes how the distribution of some quantity (such as heat) evolves over time in a solid medium, as it flows from places of higher to lower concentration. The heat equation is a special form of the diffusion equation. In 1D the heat equation is written as:

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} 0 \leq x \leq L$$

In this problem we will use the following initial and boundary conditions:

$$u(t = 0) = 0$$
$$u(x = 0) = 20$$
$$u(x = L) = 100$$

Where $L = 1m$, $\Delta x = 0.01\ m, \alpha = 0.1, and\ \Delta t = 0.01\ s.$

a) Provide a physical description of each term in the heat equation.
b) Use the finite difference method to rewrite the heat equation in numerical form. Hint: use a forward difference scheme in time and a central difference in space. Next, use algebraic manipulation to solve for u_i^{n+1}. Record both forms of the equation.
c) What do you expect to happen to the solution *u(t)* the following parameters change: $\alpha, \Delta t\ and\ \Delta x$?
d) Write a program in your favorite coding language (C++, C, or Fortran) that solves the 1D heat equation using the Forward Euler method. Provide plots of your solution at times t = 0, Δt = 0.5s, and $t = 1.0s$. Use n = 10.
e) *F* is a dimensionless parameter that describes the salient physical parameters of the heat equation. *F* is defined as $\alpha \frac{\Delta t}{\Delta x^2}$. Conceptually this can be though of as the ratio of diffusive transport rate to the heat storage rate. Vary α, Δt, Δx and report your findings on the solution to the heat equation. How does *u(t)* vary with *F*?
f) Lastly, can you think of any other dimensional parameters that affect the stability of other PDEs? Give an example of one and provide a brief description.

Exercise 9: Nonuniform Fast Fourier Transform (NUFFT) (150pts)

Write a C, C++, or Fortran function that computes the Fourier transform of piecewise polynomial data via the fast Gaussian gridding algorithm of [1]. Use double precision variables, and assume the following input and output:

Input: two vectors

$$v_{(M+1)\times 1} \text{ and } f_{M\times 3}$$

where $f(x) = f_{i,1} + f_{i,2}x + f_{i,2}x^2$ is the polynomial approximation of a function $f(x)$ in the sub-interval $[v_i, v_{i+1}]$. Assume $v_1 = 0 \leq v_2 \leq v_3 \ldots \leq v_{M+1} = 2\pi$.

Output is a vector $F_{M\times 1}$ where $\{F_k, k = -M/2, \ldots, M/2 - 1\}$ are the Fourier coefficients of f. Report the CPU timings for $M = 256,512,1024$.

Reference 1: Accelerating the Nonuniform Fast Fourier Transform, Greengard and Lee, SIAM Review 46, 443 (2004).

Exercise 10: Blocked Matrix-Matrix Multiply (150 pts)

Implement blocked/Tiled version of matrix-matrix multiply in C, C++, or Fortran. Compare this to the results of the naïve and BLAS implementations in Lab 04. Try to tune the block sizes to increase the performance of the tiled algorithm.

See lecture 15 slides for a description of the tiled algorithm.

Exercise 11: Non-blocking MPI (50 pts)

Modify the MPI Solution of the SimulatedAnnealing program to use non-blocking communication. Perform a strong scaling study up to 36 MPI processes. Compare the performance of 36 MPI processes on 1 node and 8 MPI processes on 4 nodes. Provide the implementation and scaling study results.