2.097J / 6.339J / 16.920J

Numerical Methods for Partial Differential Equations

Massachusetts Institute of Technology – Fall 2023

Project 1 – Finite Differences and Solution Methods

Version 2.0 *Due: Oct. 13, 2023*

This project is comprised of two problems and investigates finite difference methods for solving partial differential equations. The first problem examines finite difference approximations in non-rectangular domains. The second problem deals with iterative solution methods including Jacobi, Gauss Siedel, and multigrid techniques.

Problem 1 - Flow in a channel (55pts)

Problem Statement

We consider the problem of optimizing the cross section of a channel. The cross section, shown in Figure 1, is assumed to have a *constant* perimeter $l = 2(h - c) + 2\sqrt{b^2 + c^2}$, which is directly proportional to the amount of material required (for this problem l is assumed constant.) Therefore, the shape can be described in terms of two parameters/inputs, the height of the cross section h and the height of the angled portion of the base c.

Given c and h, the geometrical parameter b can be calculated from:

$$b = \sqrt{\left(\frac{1}{2}l - h + c\right)^2 - c^2} \tag{1}$$

We are interested in two outputs, the flow rate Q defined as the integral of the velocity u (normal

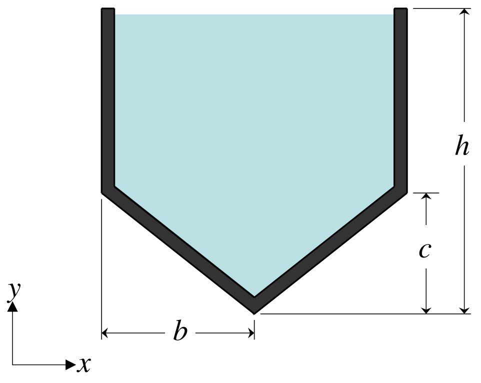


Figure 1: Channel cross section.

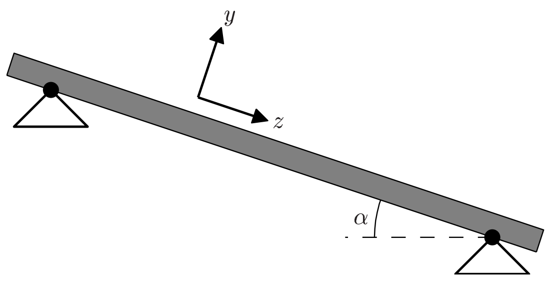


Figure 2: Flow channel.

to the cross section) over the cross section Φ

$$Q = \iint_{\Phi} u \, dx \, dy \tag{2}$$

and the moment of inertia of the cross section I. The last output, can be related to the deflection of the channel, due to the loading of the flowing fluid. The trapezoidal cross section has a thickness $t = 0.05$, and its moment of inertia I with respect to the neutral axis can be calculated from

$$I = 2\left(\frac{h+c}{2} - \bar{y}\right)^2 (h-c)t + \left(\bar{y} - c\bar{y} + \frac{1}{3}c^2\right) t\sqrt{b^2+c^2}, \tag{3}$$

where the distance to the neutral axis is from the lowest point in the section is

$$\bar{y} = \frac{h^2 - c^2 + c\sqrt{b^2+c^2}}{2(h-c) + 2\sqrt{b^2+c^2}}. \tag{4}$$

The channel is at a slope of angle α with respect to the horizontal direction, as shown in Figure 2, and the horizontal component of the gravitational force creates the pressure gradient for the downward flow. The governing equations are the steady Navier-Stokes equations,

$$\mathbf{u} \cdot \nabla \mathbf{u} + \nabla p = \mathbf{f} + \nu \nabla^2 \mathbf{u}. \tag{5}$$

With the assumption of fully developed flow, these can be reduced to Poisson's equation for the velocity component u normal to the channel cross section,

$$-\nabla^2 u = \frac{g \sin\alpha}{\nu}. \tag{6}$$

Here, g is the gravitational constant, and ν is the kinematic viscosity of the fluid. For simplicity, we assume that

$$\frac{g \sin\alpha}{\nu} = 1. \tag{7}$$

Along the walls of the channel the velocity is zero (no-slip condition), and on the free surface a zero-stress condition ($\frac{\partial u}{\partial n} = 0$) is assumed.

Numerical Procedure

We will solve Poisson's equation using a finite difference procedure. Given the symmetry of the geometry, we only consider half the channel section, as shown in Figure 3. The following boundary conditions are applied on the original domain:

$$\begin{cases} \frac{\partial u}{\partial n} = 0, & \text{in } AB \text{ and } AD \\ u = 0, & \text{in } BC \text{ and } CD. \end{cases} \tag{8}$$

To solve the problem, we map the half domain Ω to a rectangular domain $\hat{\Omega}$ and then solve the equations numerically using the finite difference method on $\hat{\Omega}$. We use a regular grid with $N \times N$ points in the computational domain, giving square cells of size $\Delta\xi = \Delta\eta = \frac{1}{N-1}$.

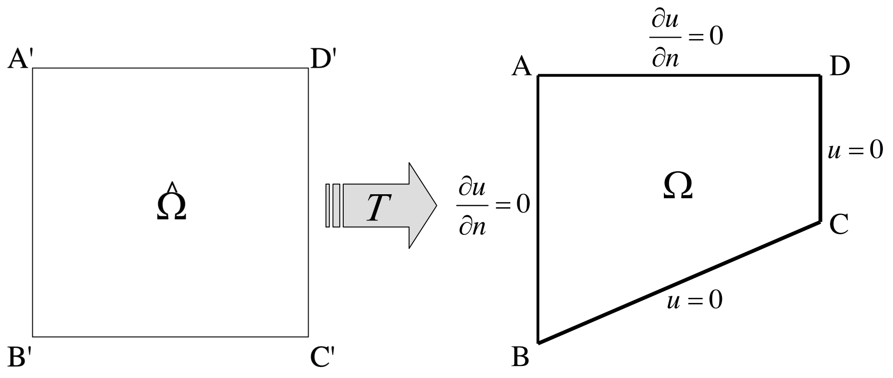

A'
D'
B'
C'
$\hat{\Omega}$
T
$\frac{\partial u}{\partial n} = 0$
A
D
$u = 0$
$\frac{\partial u}{\partial n} = 0$
Ω
C
$u = 0$
B

Figure 3: Geometrical transformation from the physical space to the computational domain.

Questions

1) (5pts) Introduce an analytical mapping T to transform the unit square computational domain $\hat{\Omega}$ (with coordinates ξ, η) to the physical domain Ω (with coordinates x, y). Write the specific equation(s) you will solve and the corresponding boundary conditions in the computational domain.

2) (12pts) Derive second-order accurate finite-difference schemes for the discretization of *all* the derivative terms in the interior of the domain, *as well as* for the boundary points. You do not have to derive special schemes for the corner points, use the top boundary scheme at corner A, and use $u = 0$ at the other three corners. Also show how you will numerically evaluate the integral for the approximate flow rate $\hat{Q}$.

3) (8pts) In a finite difference approximation, weighted combinations of the solution value at surrounding nodes are used to approximate derivatives and represent the governing differential equation at a given node. For node $N_{i,j}$, these weights are 'stamped' into appropriate positions

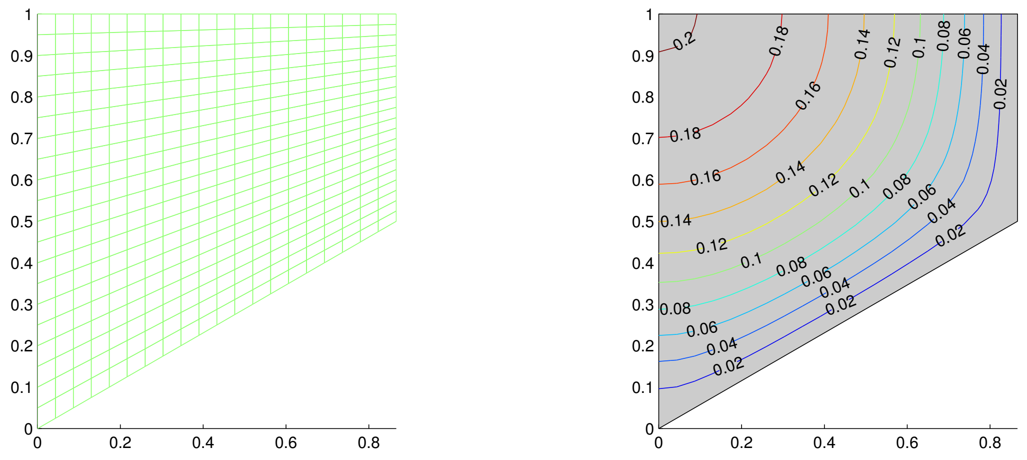


Figure 4: Sample solution, for $l = 3.0$, $c = 0.5$, $h = 1.0$, and $N = 21$. The plots show the grid (left) and contour lines of the solution (right). For these values, the flow rate for the complete cross section is $\hat{Q} = 0.129$.

of the matrix that approximates the Laplacian operator in the domain. For a node $N_{i,j}$, present the nonzero entries of its corresponding matrix row. Include cases for interior **and** boundary nodes.

4) (15pts) Write a program that solves the problem. A skeleton code for this problem has been provided in ChannelSkeleton. The use of the skeleton code is not mandatory. In addition, you may decide to use any programming language to complete this assignment. Using the program you have developed, calculate the solution and $\hat{Q}$ using a grid size $N = 21$, for $l = 3.0$, $c = 0.5$, and $h = 1.0$. Compare your results with the sample solution in Figure 4. Please show your results and the comparison.

5) (3pts) Comment briefly(several bullets) on how the program you wrote to solve the channel flow problem would be extended to form a general Poisson solver for an arbitrary 4-sided solution domain.

6) (6pts) Calculate the convergence rate for the error in the output $\hat{Q}$, and for the L_∞ and the L_2 norms of the solution. Do the calculation for $l = 3.0$, $h = 1.0$, and $c = 0.2, 0.7$ (a total of 6 convergence plots); verify that you obtain second-order convergence. Use the grid sizes $N = 11, 21, 41, 81$. Since we don't know the exact solution, use the solution for $N = 81$ as a reference. To calculate the L_∞ and the L_2 norms, restrict the solution obtained at the reference mesh, to the current coarse mesh. Please plot your solutions on a log-log plot. This is common practice, and should be applied to all future error plots even if it is not explicitly stated to do so.

7) (6pts) Keeping $l = 3$, vary the two inputs $h = [0.1, 1.0]$ and $c = [0.0, 1.0]$. Choose points on a regular grid in this two dimensional space, with constant interval 0.1 for both h and c. Each point, an (h, c) pair, represents a possible cross section. For each of these configurations, solve the problem (using $N = 21$) and calculate the flow rate $\hat{Q}$ and the moment of inertia I. Create a graph, using the two outputs and calculate the convex hull of these points. The convex hull, known as the Pareto optimal frontier, is a trade-off curve; for a specific choice of one output, we determine the optimal choice for the other output. Assuming we are interested in maximizing the flow rate, what can we say about the resulting geometry? Explain the results you obtain for the Pareto front.

Problem 2 - Iterative Methods: Jacobi, G-S, Multigrid (45pts (+10pts bonus))

Problem Statement

It is of interest to researching neuroscientists to determine which regions of the brain are active during a given neurological response. In order to non-invasively determine active parts of the brain, an electroencephalogram (EEG) is one tool which can be used. The EEG uses scalp mounted electrodes to measure the electrical pulses created by regions of the brain which are active. In addition to the electrode data, one can make assumptions pertaining to the value of the potential on the the scalp (the boundary of the domain). In order to determine which regions of the brain are active (sources of electrical activity), an inverse problem must be solved.

In addition to the EEG application, we could consider another example of an inverse problem. In this second example, a researcher may have an array of reaction chambers. Due to the constraints of the reactions being studied, it is possible that only the temperature and heat flux at the boundaries of the test chamber array can be measured. It is of interest to the research scientist to determine the positions in the array in which reactions have taken place. In order to determine the locations in the array which have undergone a reaction, the boundary measurements of the temperature and heat flux can be used, and an inverse problem may be solved to determine the heat source locations (which are equivalent to the array positions in which reactions have occurred).

In this question, we use a rudimentary representation and approach to 'solving' the inverse problem. Rather than directly solving the inverse problem (which in itself is an interesting problem), we consider the forward problem. In the forward problem, we assume the source positions and potential on the boundary are known, and calculate the corresponding boundary flux. We then compare the boundary flux for the given source arrangement with the known boundary flux for which we are trying to determine source positions. By trial and error, or by some more advanced methods and reasoning, we will eventually be able to determine the unknown source configurations by matching the known and computed boundary flux values.

To solve the equations, we will exercise the iterative solution techniques seen in class. The governing equation for the problem is Poisson's equation,

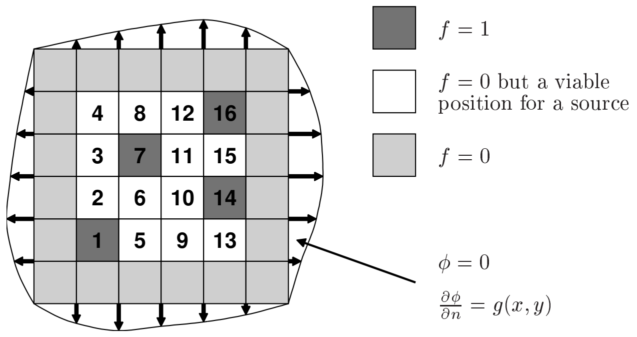
$$-\nabla^2\phi(x,y) = f, \tag{9}$$

with boundary condition $\phi(x,y) = 0$, on the entire boundary. We will examine a unit square domain, see Figure 5.

The domain is divided into 36 block regions. The 36 blocks are arranged in a 6×6 array of blocks. The 16 blocks not on the domain boundary are labeled interior blocks. All blocks are set to have an $f = 0$, with the exception of four of the interior blocks which have $f = 1$. Those four interior blocks with $f = 1$ are said to be source blocks. The challenge in this particular problem is to determine the position of 4 source blocks given the distribution of the normal flux $-\frac{\partial\phi}{\partial n} = g(x)$ (here n is the outside unit normal to the boudnary) on the boundary of the domain.

Questions

In order to solve the Poisson equation, we overlay a computational grid on the domain. For this problem, a 25×25 node computational grid is used unless otherwise specified. This would mean that a source block would correspond to a 5×5 set of nodes being set to $f = 1$.



4
8
12
16
3
7
11
15
2
6
10
14
1
5
9
13
$f = 1$
$f = 0$ but a viable position for a source
$f = 0$
$\phi = 0$
$\frac{\partial \phi}{\partial n} = g(x, y)$

Figure 5: A pictorial explanation of the problem statement. Here the source blocks are located in blocks 1, 7, 14, and 16.

1) (8pts) Write down the Jacobi and the Gauss-Siedel iteration schemes to solve the above Poisson's equation.

2) (12pts) Code the Jacobi and G-S routines. Allow variable specification of the "fineness" of the computational grid (eg, a computational grid of 7×7 nodes, 13×13 nodes, 25×25 nodes, etc. while also allowing specification of the source blocks in the RHS vector).

Note: For simplicity we shall assume that for the source blocks all nodes, including the block boundaries, have $f = 1$. Also, for two adjacent source blocks, the points on the common boundary also have $f = 1$.

a. Implement a relaxation scheme for both. Describe the equation that you use, if you haven't already in the previous question.

b. Show the convergence of your Jacobi and G-S solver for the above arrangement of blocks (1,7,14,16), with several relaxation factors. What is the optimal relaxation factor for the G-S method; this need not be a specific number, but a value within ± 0.1. In order to calculate the iteration error, you can either i) solve your problem exactly, that is $u = A^{-1}f$ and compared with the exact solution of the system, or 2) use calculate the residual $r_h = f - Au_h$ and take the norm of the residual.

Note: for this question and for the remainder of this problem, show the convergence rates by plotting the log of the error (or residual) norm vs. the number of iterations. Since we expect the iteration error (or residual) to be of the form $\sim \lambda^r$, a plot $\log ||r||$ vs. r, should be a straight line.

c. A sample solution for the example case (1,7,14,16) is provided in Table 1 (left). Verify that your solver produces a similar output. Show either your matrix results or an image of your results.

Hint: For later problems it will be convenient to construct a function that takes an input including the 4 numbers corresponding to the test blocks (B1, B2, B3, B4 $\in$ {1,...,16}), and returns the normal derivative around the perimeter as an output.

3) (15pts) Implement a multigrid routine that will perform a two-grid method. Explain the restriction and prolongation method you choose to implement. Try a relaxation factor of $\frac{1}{2}$ and $\frac{4}{5}$. Please remember how you are expected to plot your error values from the previous problem (this will be expected to extend forward into all future assignments).

a. Compare the convergence of the multigrid routine using either Jacobi and Gauss-Seidel as the relaxation scheme with Jacobi and G-S on a single grid. Which method is best? Which relaxation factor for your multigrid routine is better, $\frac{1}{2}$ or $\frac{4}{5}$?

b. Implement routines where $\nu_1 = \nu_2 = 1$ and $\nu_c = 2, 4,$ and exact if you have A available, otherwise take $\nu_c = 20$ as a proxy for the exact solution on a coarse mesh. Here ν_c refers to the number of iterations on the coarse grid. How do the routines compare? What can we conclude from this?

4) (10pts) Using the best method (based on convergence rate) and the given normal flux distribution, determine the unknown positions of the four blocks, for the $-\frac{\partial \phi}{\partial n} = g(x)$ given in Table 1 (right). (values also provided in file `flux_problem_2.txt`) You do not need to find an elaborate/efficient way to do it; you can just try all the possible combinations. Pictorially show where the sources are located.

5) **BONUS**: (Possible +10pts[1]) Write a generalized V-cycle multigrid routine which allows multiple grid refinements. Show convergence for the 2, 3, and 4 grid refinement V-cycles. How do they compare to the other iterative routines?

[1]Only considered if the assignment is submitted within the regular deadline. Additional bonus points for those exceeding 100 points in this project will be used in the final grade computation to compensate for any lost points in other homeworks and/or assignments. Note that answering the bonus question is not required to get a perfect grade in the class.

Numerical Values of Normal Flux Distribution

(1,7,14,16) Configuration

Pt	Left	Right	Bottom	Top
1	0.0000	0.0000	0.0000	0.0000
2	0.0122	0.0072	0.0122	0.0054
3	0.0244	0.0145	0.0244	0.0109
4	0.0361	0.0222	0.0360	0.0163
5	0.0466	0.0303	0.0463	0.0217
6	0.0552	0.0391	0.0548	0.0270
7	0.0612	0.0483	0.0605	0.0323
8	0.0643	0.0576	0.0633	0.0373
9	0.0649	0.0662	0.0635	0.0420
10	0.0636	0.0734	0.0618	0.0464
11	0.0611	0.0784	0.0590	0.0504
12	0.0583	0.0811	0.0561	0.0542
13	0.0553	0.0819	0.0533	0.0578
14	0.0522	0.0814	0.0507	0.0612
15	0.0489	0.0803	0.0483	0.0645
16	0.0452	0.0790	0.0459	0.0673
17	0.0411	0.0771	0.0431	0.0689
18	0.0367	0.0740	0.0399	0.0684
19	0.0319	0.0687	0.0360	0.0652
20	0.0268	0.0610	0.0315	0.0589
21	0.0216	0.0509	0.0262	0.0498
22	0.0162	0.0392	0.0202	0.0387
23	0.0108	0.0264	0.0138	0.0262
24	0.0054	0.0132	0.0070	0.0132
25	0.0000	0.0000	0.0000	0.0000

Unknown Configuration

Pt	Left	Right	Bottom	Top
1	0.0000	0.0000	0.0000	0.0000
2	0.0084	0.0062	0.0082	0.0071
3	0.0169	0.0124	0.0162	0.0141
4	0.0258	0.0186	0.0239	0.0210
5	0.0351	0.0247	0.0310	0.0278
6	0.0449	0.0307	0.0374	0.0342
7	0.0550	0.0365	0.0430	0.0402
8	0.0650	0.0419	0.0477	0.0456
9	0.0740	0.0467	0.0514	0.0503
10	0.0811	0.0507	0.0541	0.0541
11	0.0854	0.0538	0.0559	0.0569
12	0.0864	0.0557	0.0567	0.0584
13	0.0844	0.0564	0.0566	0.0587
14	0.0800	0.0559	0.0555	0.0578
15	0.0738	0.0542	0.0533	0.0556
16	0.0668	0.0513	0.0502	0.0524
17	0.0593	0.0473	0.0463	0.0481
18	0.0517	0.0426	0.0415	0.0432
19	0.0441	0.0372	0.0363	0.0376
20	0.0365	0.0314	0.0306	0.0316
21	0.0290	0.0253	0.0246	0.0254
22	0.0216	0.0191	0.0186	0.0191
23	0.0143	0.0127	0.0124	0.0128
24	0.0071	0.0064	0.0062	0.0064
25	0.0000	0.0000	0.0000	0.0000

Table 1: These are the tabulated results for the normal flux, for the (1,7,14,16) configuration (left) and for the unknown configuration (right), for which you are to solve. The points correspond to the grid points in a 25×25 node grid, for increasing y-values (left and right boundaries), and for increasing x-values (bottom and top boundaries).