2.097J / 6.7330J / 16.920J

Numerical Methods for Partial Differential Equations

Massachusetts Institute of Technology – Fall 2024

Project 1 – Finite Differences and Solution Methods

Version 1.1 *Due: Oct. 9, 2024*

Introduction to Project 1

This project is comprised of two parts and investigates finite difference methods for solving partial differential equations. The first part examines finite difference approximations on non-rectangular domains. The second part covers iterative solution methods, including Jacobi, Gauss-Seidel, and multigrid techniques.

Part 1 - Turbine blade cooling 55pts

In the first stages of a turbine, the turbine blades are subjected to a high temperature flow due to the hot gas produced in the combustor. As a result, turbine blades are actively cooled by pumping lower temperature air through the blade cooling passages. One particular passage goes through the trailing edge of the turbine blade. Cooling air flows from the inside of the blade, towards and then out of the trailing edge. Figure 1 shows the geometry.

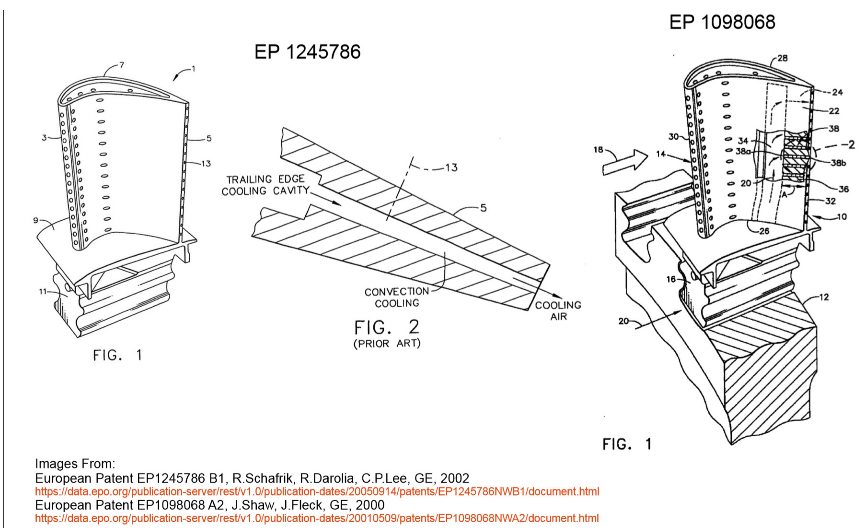

Figure 1: Trailing edge with cooling passages

In this project, we will simulate the heat transfer of an internally cooled turbine blade using a finite difference discretization of the trailing edge of the blade and the cooling flow. The geometry is shown in Figure 2. The geometry is two-dimensional, assuming the blade geometry varies slowly in the spanwise direction.

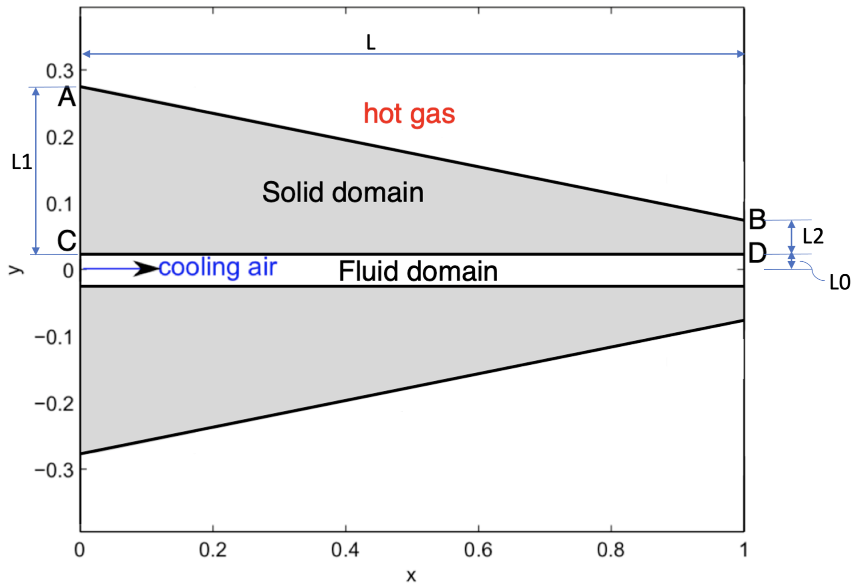

Figure 2: Geometry of blade trailing edge with cooling passage. Due to symmetry, we only need to solve the heat equation in the upper half of the trailing edge. The coordinates of the solid domain corner points are $A = (0, 0.275)$, $B = (1, 0.075)$, $C = (0, 0.025)$ and $D = (1, 0.025)$.

Recall that the conductive heat transfer rate is given by,

$$\vec{q} = -k\nabla T. \tag{1}$$

where k is the thermal conductivity of the blade metal and T is the temperature. Since we do not have any heat sources inside the blade, we will have $\nabla \cdot \vec{q} = 0$. Therefore, the steady state temperature distribution inside the turbine blade can be modeled using Laplace's equation in two-dimensions,

$$\nabla \cdot (k\nabla T) = 0, \tag{2}$$

Regarding the boundary conditions, we will assume that there is no heat transfer through the left boundary of the domain. That is, $\vec{q} \cdot \vec{n} = 0$ on AC. Here, $\vec{n}$ is a normal pointing out of the solid domain. We have convective heating due to the hot exterior flow on the external boundaries (AB and BD), and convective cooling due to the cooler interior flow along the internal boundary (CD). Therefore, the boundary conditions for boundaries AB, BD and CD can be modeled using Newton's law of cooling. For these three boundaries, the heat flux out of the blade will be given by,

$$\vec{q} \cdot \vec{n} = h\left(T - T_t\right), \tag{3}$$

where h is the convective heat transfer coefficient and T_t is the stagnation temperature of the air away from the surface. For the outer surface of the blade, the stagnation temperature will be assumed to be a constant $T_{t,\text{hot}}$.

Inside the internal cooling passage the air flows from left to right and its temperature depends on many factors, such as the rate at which the cool air is pumped, its temperature at the inlet and the heat capacity of the air. We will assume that, inside the channel, the heat transfer in the direction normal to the flow is fast enough so that the temperature is uniform in the y-direction. Thus, the temperature distribution can be modeled by the one-dimensional energy conservation equation

$$U\frac{dT_{t,\text{cool}}}{dx} = 2\vec{q}\cdot\vec{n} \tag{4}$$

where U is a normalized velocity of the cooling flow. The inlet of the cooling flow (on the left side of the domain at $x = 0$) has stagnation temperature of $T_{t,\text{cool}}$. For our analysis, we will use the following normalized values for the different constants:

$$k = 1, \qquad h = 5.0, \qquad T_{t,\text{hot}} = 1.4, \qquad T_{t,\text{cool}}(x=0) = 0.6, \qquad U = 5$$

1 Tasks

1.1 Fluid Part

1) (10 pts) For this section we shall assume that the temperature at the blade is constant and equal to 1.2. Solve equation (4) analytically. Solve equation (4) numerically using a uniform grid and two discretization methods: i) a first order backward difference method, and ii) a second order backward difference method.

Since we have the exact solution we can calculate the error using the modified L_2 ($p = 2$) norm presented in class. Start with a grid consisting of 10 equal intervals and keep doubling the number of intervals until the error is less than $1.e-4$. Show a $\log-\log$ plot of the error vs. Δx, and determine the order of converge of each scheme by examining the graphs. Since we expect the norm of the discretization error to be of the form $||e|| \sim \Delta_x^\alpha$, a plot $\log||e||$ vs $\log(\Delta x)$ should be a straight line with slope α.

Since the second order scheme requires 2 upwind points you can use the first order scheme to calculate the solution at the first point and then continue with the second order method. Note that using a backwards difference approximations, we could space march the solution and calculate one unknown at a time and avoid solving a system of equations. However, forming a matrix and solving the system of equations will help for the thermal-fluid coupling in the last questions.

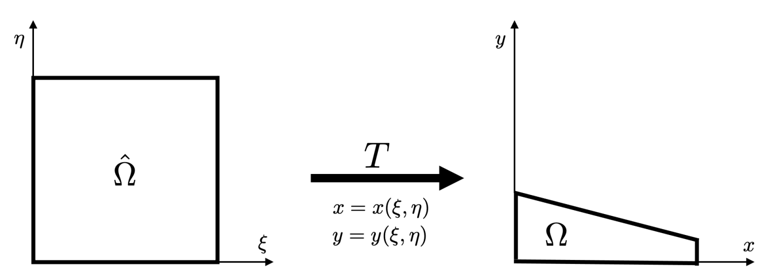

Figure 3: Mapping between physical and computational domains.

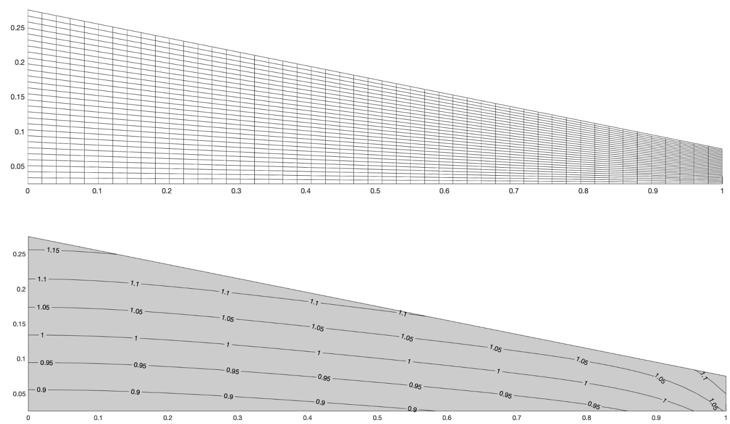

Figure 4: Sample solution on a uniform 51×31 grid. For this solution the temperature of the cooling fluid is kept to a constant and equal to 0.6.

1.2 Thermal Part

3) (5pts) Introduce an analytical mapping, T, to transform the unit square computational domain $\hat{\Omega}$ (with coordinates ξ, η) into the physical domain Ω (with coordinates x, y) as shown in figure 3. Write the specific equations and the corresponding boundary conditions in the computational domain.

Note: There are many possible choices for this mapping, but you should aim for the simplest form which is of the form $x = a + b\xi + c\eta + d\xi\eta$ for appropriate constants a, b, c and d, and with a similar expression for y.

4) (5pts) Derive second-order accurate finite-difference approximations for *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 for corners A and B and the bottom boundary scheme for corners C and D.

5) (5pts) In a finite difference approximation, weighted combinations of the solution value at surrounding nodes are used to represent the governing differential equation at a given node. Consider a node (i, j) in your grid that maps to the unknown k in your global vector. Then, these weights are 'stamped' into appropriate positions in row k of the matrix that approximates the Laplacian operator in the domain. For node (i, j), present the nonzero entries of its corresponding matrix row. Include cases for interior **and** boundary nodes.

6) (15pts) Write a program that computes the temperature distribution on the blade for a given temperature distribution of the flow in the cooling channel. A skeleton code for this problem is provided in `HeatSkeleton`. The use of the skeleton code is not mandatory. In addition, you may decide to use Matlab, Python or Julia to complete this assignment. Use a 51×31 grid and compute the solution for a constant temperature of the cooling flow equal to 0.6. Produce a contour plot of the temperature distribution computed and compare your results with the sample solution in Figure 4.

1.3 Thermal-Fluid Coupling

For this section, we will discretize the blade using 81×41 points and discretize the cooling channel using a grid of 81 points. This choice guarantees that, on the channel, the grid nodes from the fluid and solid domains coincide.

8) (10pts) Combine the codes developed for the fluid and thermal components to solve equations (2) and (4) together for the temperature distribution in the blade and in the cooling fluid simultaneously. Use the second order scheme for the flow discretization. Describe the structure of the global problem and how do the coupling terms enter into the global matrix. What is the maximum temperature on the blade? Where does it occur?

An alternative to solving the coupled problem directly is to start with a constant temperature distribution for the cooling fluid, as in section 6, and solve equation (2). Knowing the temperature distribution on the blade, solve (4). Using the latest values of the temperature distribution on the blade and the cooling channel, alternate the solution of equations (2) and (4) until the temperature distributions converge. We shall assume that convergence has been achieved when the L^2 norm of the temperature change $T_{t,cool}(x)$ from one iteration to the next one is less than $1e-7$. If you follow this approach rather than the simultaneous solution, the maximum credit for this question will only be 5 pts.

9) (5pts) How does the maximum temperature of the blade vary as the cooling flow rate is changed? In particular how does the maximum temperature change when the velocity U in equation 4 changes between 2.5 and 20 (consider intervals of U of 2.5). Plot the maximum temperature as a function of U and comment on the results.

Part 2 - Iterative Methods: Jacobi, G-S, Multigrid 45pts (+5 pts 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,

$$-\nabla^2\phi(x,y) = f, \tag{5}$$

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 $f = 1$.

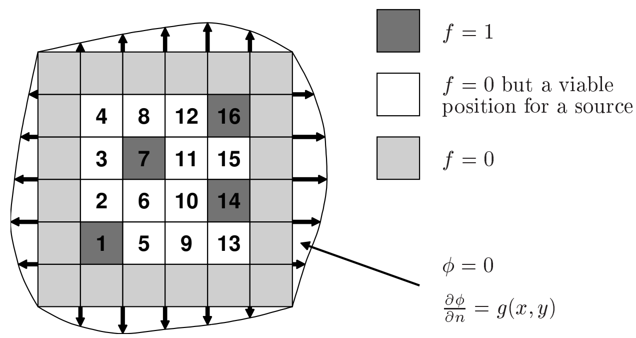

Figure 5: A pictorial explanation of the problem statement. Here the source blocks are located in blocks 1, 7, 14, and 16.

1) (5pts) Write down the Jacobi and the Gauss-Seidel iteration schemes to solve the above Poisson's equation.

2) (10pts) 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 very precise 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 $r_h = f - A u_h$ and take the norm of the residual.

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.

Note: 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 $\frac{4}{5}$.

a. Compare the convergence of the multigrid routine using 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 the coarse mesh. (here ν_c refers to the number of relaxation iterations on the coarse grid), for the multigrid. 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) (5pts) 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 does the rate of convergence change when the number of grids in increased?

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