SMA5212/16.920J/2.097J - Numerical Methods for Partial Differential Equations

Massachusetts Institute of Technology — Singapore - MIT Alliance

Problem Set 1 - Finite Differences and Iterative Methods

Handed out: February 10, 2003 — Due: March 5, 2003

Problem 1 - Flow in a channel (55p)

Problem Statement

We consider the problem of optimizing the cross section of a channel of trapezoidal shape. The cross section, shown in Figure 1, is assumed to have a *constant* perimeter $l = b + 2d$, 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 size of the base in the trapezoid b, and the height of the cross section h.

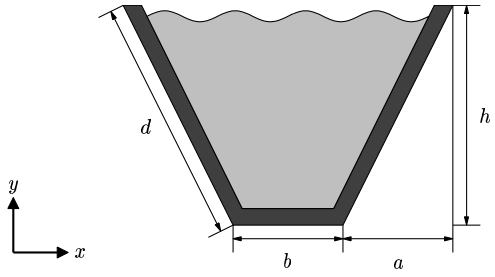

Figure 1: Trapezoidal cross section.

Given b and h, the geometrical parameters a and d can be calculated from:

$$d = \frac{1}{2}(l - b) \tag{1}$$

$$a = \sqrt{\frac{1}{4}(l - b)^2 - h^2}. \tag{2}$$

We are interested in two outputs, the flowrate Q defined as the integral of the velocity u (normal to the cross section) over the cross section Φ

$$Q = \int_{\Phi} u \, dx \, dy \tag{3}$$

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

$$I = \bar{y}^2 bt + \frac{2dt}{3}(h^2 - 3h\bar{y} + 3\bar{y}^2), \tag{4}$$

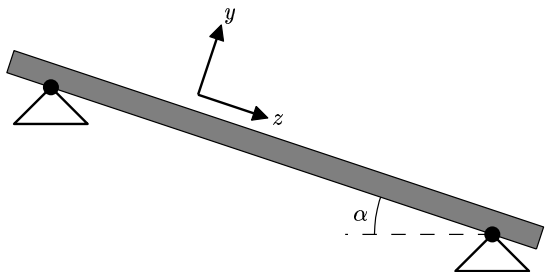

Figure 2: Flow channel.

where the distance to the neutral axis is

$$\bar{y} = h\frac{d}{2d+b}. \tag{5}$$

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

$$\mathbf{u}\cdot\nabla\mathbf{u} + \nabla p = \mathbf{f} + \nu\nabla^2\mathbf{u}. \tag{6}$$

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,

$$-\nabla^2 u = \frac{g\sin\alpha}{\nu}. \tag{7}$$

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

$$\frac{g\sin\alpha}{\nu} = 1. \tag{8}$$

Along the walls of the channel the velocity is zero, 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. Due to symmetry we only consider half the channel section, as shown in Figure 3. The following boundary conditions are applied on the original domain:

$$\begin{cases} \frac{\partial u}{\partial n} = 0, & \text{in } AB \text{ and } AD \\ u = 0, & \text{in } BC \text{ and } CD \end{cases} \tag{9}$$

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 $\Delta\xi = \Delta\eta = 1/(N-1)$.

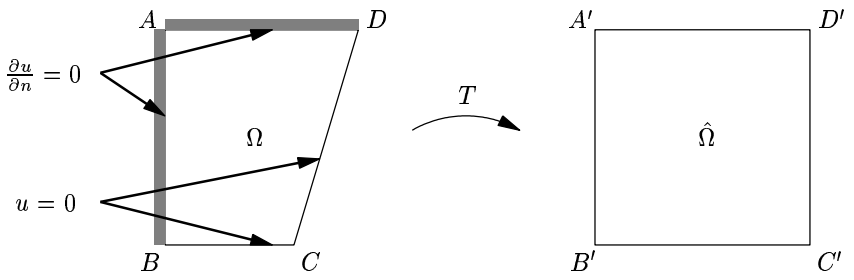

Figure 3: Geometrical transformation from the physical space to the computational domain.

Questions

1) (5p) Introduce a mapping T to transform the original domain Ω (with coordinates x, y) to the unit square $\hat{\Omega}$ (with coordinates ξ, η). Derive the equations and the boundary conditions in the new domain.

2) (10p) In the transformed domain, use Taylor series expansions to 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 left boundary scheme at corner A, and $u = 0$ at the other three corners. Also show how to evaluate the integral in the approximate flowrate $\hat{Q}$ numerically.

3) (15p) Write a program, that solves the problem. Then using your program, calculate the solution and $\hat{Q}$ using a grid size $N = 21$, for $l = 3.0$, $b = 0.5$, and $h = 1.0$. Compare your results with the sample solution in Figure 4.

4) (5p) Solve the problem for $l = 3.0$, $h = 1.0$, and $b = 0.0, 0.5, 1.0$ (three solutions), with $N = 41$. Plot the solutions and compute $\hat{Q}$.

5) (10p) 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 $b = 0.0, 0.5, 1.0$ (a total of 9 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, interpolate the solution obtained at the coarser meshes, to the reference mesh.

6) (10p) Keeping $l = 3$, vary the two inputs $h = [0.1, 1.0]$ and $b = [0.0, 1.0]$. Choose points on a regular grid in this two dimensional space, with constant interval 0.1 for both h and b (a total of 110 combinations). Each point, an (h, b) pair, represents a possible cross section. For each of these configurations, solve the problem (using $N = 21$) and calculate the flowrate $\hat{Q}$ 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. Explain the results you obtain.

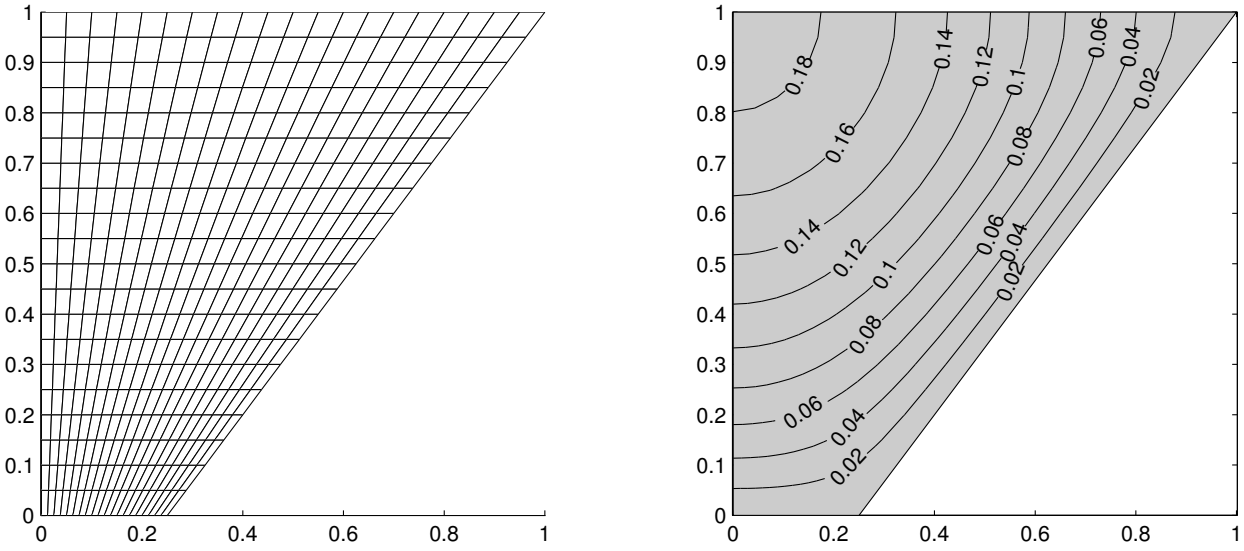

Figure 4: Sample solution, for $l = 3.0$, $b = 0.5$, $h = 1.0$, and $N = 21$. The plots show the grid (left) and contour lines of the solution (right). For these values, the flowrate $\hat{Q} = 0.0574$.

MATLAB® Hints

- It is usually necessary to *vectorize* MATLAB® code in order to get high performance. This means, for example, that for-loops should be avoided and replaced by higher-level constructs. We *do not* require you to do this, feel free to use for-loops when building up matrices and vectors. This will make the code easier to read and understand, and these (relatively) small problems are fast enough anyway.
- A uniform grid with spacings `dxi`, `deta` can be created with

  ```
  [xi,eta]=ndgrid(0:dxi:1,0:deta:1);
  ```

  This gives two $N \times N$ arrays `xi`, `eta`.
- A grid with x, y coordinates in the $N \times N$ arrays `x`, `y` can be visualized using

  ```
  mesh(x,y,0*x,0*x)
  view(2), axis equal
  ```

- Remember to make A a *sparse matrix* (or you will run out of memory): `A=sparse(N^2,N^2);`
- When filling in the A matrix, it is convenient to use a mapping function, `map=reshape(1:N^2,N,N);` The position of the grid point `i`, `j` in the linear system of equations is then `map(i,j)`.
- The solution U to the linear system of equations $AU = F$ can be computed with `U=A\F`. If `A` is a sparse matrix, this will use the direct sparse solver in MATLAB®.
- The $N^2 \times 1$ solution vector `U` can be reshaped to an $N \times N$ array with `u=reshape(U,N,N);` This format is sometimes easier to work with, for example when computing the integral for $\hat{Q}$ and when plotting the solution.
- The contour plot in Figure 4 was created with

  ```
  [cc,hh]=contour(x,y,u,0.02:0.02:max(u(:)));
  clabel(cc,hh);
  patch([0,b/2,b/2+a,0],[0,0,h,h],-ones(1,4),0,'facecolor',[.8,.8,.8])
  axis equal
  ```

  First, contour curves are made for the solution `u` on the grid `x`, `y`, all which are $N \times N$ arrays. Then labels are added, and the geometry is drawn, using the parameters in `a`, `b`, and `h`.
- Alternatively, you can create a color plot:

  ```
  surf(x,y,0*x,u)
  view(2),axis equal
  set(gcf,'renderer','zbuffer');
  shading interp, colorbar
  ```

- To plot the convex hull for the points in the vectors `Qs` and `Is`, type

  ```
  k=convhull(Qs(:),Is(:));
  plot(Qs(:),Is(:),'.',Qs(k),Is(k))
  ```

Problem 2 - Iterative Methods: Jacobi, G-S, Multigrid (45p)

Problem Statement

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

$$-\nabla^2\phi(x,y) = f, \tag{10}$$

with boundary condition $\phi(x,y) = 0$, on the entire boundary. We will examine a square domain, see Figure 5. The domain is divided into 36 main blocks, or a 6 x 6 main grid. Four of the inner 16 blocks (4 x 4 interior grid) are set to be source blocks ($f = 1$), while for the remaining blocks $f = 0$. The position of the four source blocks is unknown. You are asked to determine their position given the distribution of the normal flux $\frac{\partial\phi}{\partial n} = g(x)$ on the boundary of the domain.

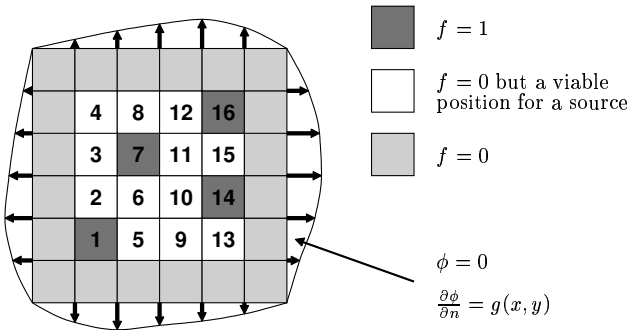

Figure 5: A pictorial explanation of the problem statement. Here the source blocks are located in blocks 1, 7, 14, and 16.

Questions

The default for these questions is a 24 x 24 grid unless otherwise specified.

1) (10p) Write down the Jacobi and the Gauss-Siedel iteration schemes to solve the above Poisson's equation and draw a block diagram or pseudo code outlining your proposed implementation.

2) (10p) Code the Jacobi and G-S routines. Allow variable specification of the "fineness" of the grid (values such as, 6 x 6, 12 x 12, 24 x 24, etc). 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.

b. Show the convergence of your Jacobi and G-S solver for the above arrangement of blocks (1,7,14,16), with various relaxation factors. What is the optimal relaxation factor for the G-S method?

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.

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) (15p) Implement a multigrid routine that will perform a two-grid method. Show and explain the restriction and prolongation method you choose to implement. Try a relaxation factor of 1/2 and 4/5.

a. Compare the convergence of the multigrid routine with the various other methods. Which method is best? Which relaxation factor for your multigrid routine is better, 1/2 or 4/5?

b. Implement a (1, 2, 1) and a (2, 4, 2) (coarse, fine, coarse), routine for the multigrid. How do the routines compare?

4) (10p) 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). Pictorially show where the sources are located.

5) **BONUS**: (Possible +20p[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 applied to gain a maximum of 100%. Additional bonus points are not carried over to future assignments.

Numerical Values of Normal Flux Distribution

(1,7,14,16) Configuration

Point	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

Point	Left	Right	Bottom	Top
1	0.0000	0.0000	0.0000	0.0000
2	0.0075	0.0051	0.0076	0.0099
3	0.0150	0.0103	0.0154	0.0198
4	0.0222	0.0154	0.0233	0.0297
5	0.0292	0.0206	0.0314	0.0396
6	0.0360	0.0259	0.0398	0.0494
7	0.0424	0.0313	0.0484	0.0588
8	0.0486	0.0369	0.0565	0.0676
9	0.0548	0.0429	0.0635	0.0748
10	0.0611	0.0492	0.0686	0.0800
11	0.0674	0.0556	0.0709	0.0823
12	0.0734	0.0619	0.0703	0.0816
13	0.0784	0.0670	0.0670	0.0784
14	0.0816	0.0703	0.0619	0.0734
15	0.0823	0.0709	0.0556	0.0674
16	0.0800	0.0686	0.0492	0.0611
17	0.0748	0.0635	0.0429	0.0548
18	0.0676	0.0565	0.0369	0.0486
19	0.0588	0.0484	0.0313	0.0424
20	0.0494	0.0398	0.0259	0.0360
21	0.0396	0.0314	0.0206	0.0292
22	0.0297	0.0233	0.0154	0.0222
23	0.0198	0.0154	0.0103	0.0150
24	0.0099	0.0076	0.0051	0.0075
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 24 x 24 grid, for increasing y-values (left and right boundaries), and for increasing x-values (bottom and top boundaries).