UNSTEADY 2D HEAT EQUATION ON CURVILINEAR MESH

Mech 588 Assignment 1

Christian Rowsell (40131393)
Mech 588 – UBC Mechanical Engineering Winter 2023

Programming Assignment:
2D Heat Equation in General Coordinates

Mech 588 — Spring, 2023

Due Date: February 10

This problem addresses issues that arise in solving the unsteady heat conduction equation, $T_t = \nabla^2 T$, on a two-dimensional non-uniform mesh.

1. Write a verification plan. Be sure to include tests for all of the key parts of your code, as well as global tests to confirm that your code correctly solves the PDE. Turn in your verification plan as part of your report, but you do not need to submit the results of your verification tests; if your code doesn't pass a reasonable set of verification cases, it won't solve the final problem, either.

The code can be broken up into the following broad categories.

1. Creation of an appropriate data structure to store mesh information, and solution information.
2. Calculation of the mesh metrics to allow for solution in the computational space instead of the physical space.
3. Computation of the flux integral for each cell.

Each of these categories can be written, and tested independently, and then once they are all verified to be working, they can be added to the main code. The intended testing for each section is as follows.

1. Creation of an appropriate data structure to store mesh information, and solution information.

To handle the data and store the solution 3 classes were used. The classes are named Vector, Operations, and Solution. The Vector class behaves as a wrapper to work on 1D arrays in a fashion similar to how 2D arrays would be utilized. The benefit of a 2D array is that you can index in both i, and j, which is like how the equations for the finite volume discretization's are described. However, 2D arrays involve more work when it comes to dynamically allocating the memory to them. Dynamic memory allocation is especially important in this case, to add for generality in reading multiple meshes. The vector class takes care of all this difficulty, to allow for focusing on the actual problem. The operations class simply allows for operator overloading for the Vector class, allowing for the possibility of adding, subtracting, and multiplying vectors. It also includes other operations such as taking the dot product and calculating the L2Norm of vectors. Both classes came from Mech 587 and have been fully tested. As such, they will not be included in the testing plan.

The solution class simply handles the storage of all the different vectors, such as the position vectors, and the temperature vector. It also handles the reading of the mesh and outputting the information. This class was tested by ensuring the mesh is read properly, and that the vectors are all properly initialized. To test that the vectors were all initialized properly, a test function was created which checks initializes a class with a set number of points in the X and Y. These vectors were then printed. If they have the correct numbers of 0's then the vector was properly initialized. To test if the mesh was read properly, the mesh function was

called for each mesh provided, and the values for X, Y, u and v at randomly chosen intervals were compared. If the values in the Solution object matched the values in the mesh file, then these tests were considered to be passed. Finally, the output was tested. The output function stores the data from the Solution in a VTK file. To test this, a solution object is created, and the T vector is assigned known values $(sin(x * \pi) * sin(y * \pi))$. This is then saved to a VTK file, which can be used to visually confirm that the output is correct.

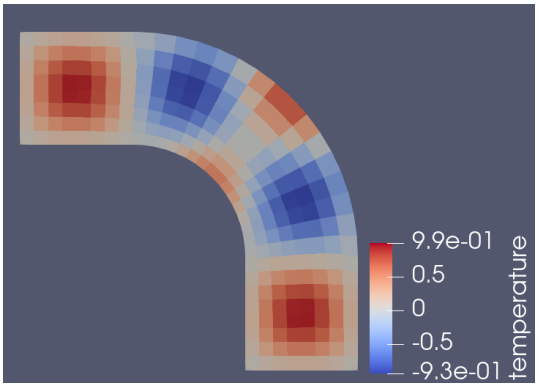


Figure 1: File Generated From Course Mesh

2. Calculation of the mesh metrics to allow for solution in the computational space instead of the physical space.

To test the mesh metrics, a MATLAB file was generated which can be used to generate new mesh files with known functions of X and Y in terms of ξ and η. This MATLAB script was tested by generating meshes, plotting them in MATLAB, and finally storing them as VTK files using the previously discussed function. They can then be visually compared to ensure the mesh file has proper formatting. Once this was confirmed, the MATLAB script allowed for the testing of the mesh metrics calculation in comparison to known values of the derivatives of x and y with respect to ξ and η. These tests were performed for 8 separate mesh metrics. These metrics include

$$On\ the\ \left(i+\frac{1}{2},j\right)\ face\text{:}\ \frac{\partial x}{\partial \xi},\frac{\partial y}{\partial \xi},\frac{\partial x}{\partial \eta},\frac{\partial y}{\partial \eta}$$

$$On\ the\ \left(i,j+\frac{1}{2}\right)\ face\text{:}\ \frac{\partial x}{\partial \xi},\frac{\partial y}{\partial \xi},\frac{\partial x}{\partial \eta},\frac{\partial y}{\partial \eta}$$

Test 1 – A rectilinear mesh.

In this test, a simple rectilinear mesh with a grid spacing of 1 is generated. This mesh is mathematically equivalent to letting y= η, and x= ξ. To pass this test, a value of 0 is expected for $\frac{\partial x}{\partial \eta}\ and\ \frac{\partial y}{\partial \xi}$, while a value of 1 is expected for $\frac{\partial y}{\partial \eta}\ and\ \frac{\partial x}{\partial \xi}$.

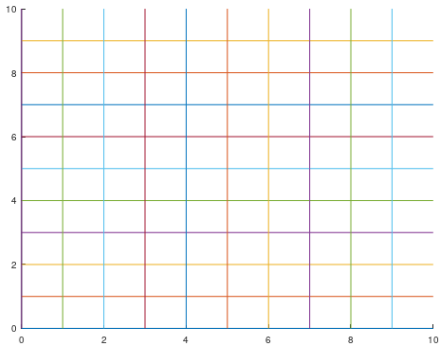


Figure 2: Rectilinear Test Mesh

Test 2 – A slanted rectilinear mesh.

This test is nearly equivalent to the previous test, except that the mesh has been rotated, so that the derivatives are no longer 0, and 1. In this case, y= 2*(η- ξ), and x=0.5*(ξ+ η). In this case, a value of 0.5 is expected for both $\frac{\partial x}{\partial \xi}$ and $\frac{\partial x}{\partial \eta}$. Meanwhile a value of 2 is expected for $\frac{\partial y}{\partial \eta}$ and a value of -2 is expected for $\frac{\partial y}{\partial \xi}$.

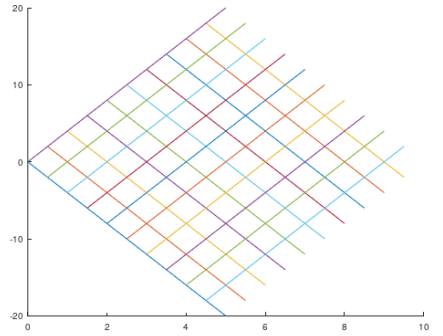


Figure 3: Slanted Rectilinear Test Mesh

Test 3 – A quadratic mesh

This test is again similar to the previous tests, with non-linear equations for η, and ξ. Here, x=0.5*(ξ^2-η^2) and y=ξη where $x \in [0.5,1.5]$ *and* $y \in [0.5,1.5]$. In this case, $\frac{\partial x}{\partial \xi}$=ξ and $\frac{\partial x}{\partial \eta} = \eta$, while $\frac{\partial y}{\partial \eta} = \xi$ and $\frac{\partial y}{\partial \xi}$.=η. In this case, constant values are not expected for the mesh metrics, and it the values of ξ and η are required to verify the mesh metrics. In order to aid in this, the mesh generation script was modified such that it assigns the value of ξ and η to the u, and v columns of the mesh. This helps to simplify the validation process.

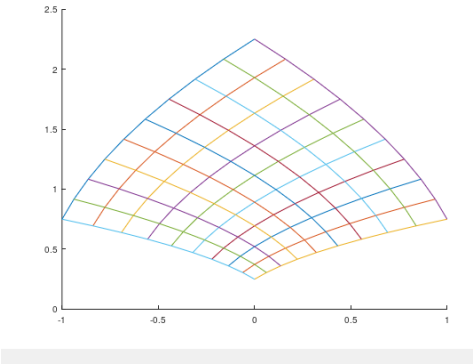


Figure 4: Quadratic Test Mesh

This mesh was then utilized to validate that the derivatives had second order accuracy. Figure 5 shows this accuracy. Both derivatives with respect to η have an accuracy of order 2, while the derivatives with respect to ξ have an order of accuracy of 1. Unfortunately, due to time constraints, this error could not be rectified.

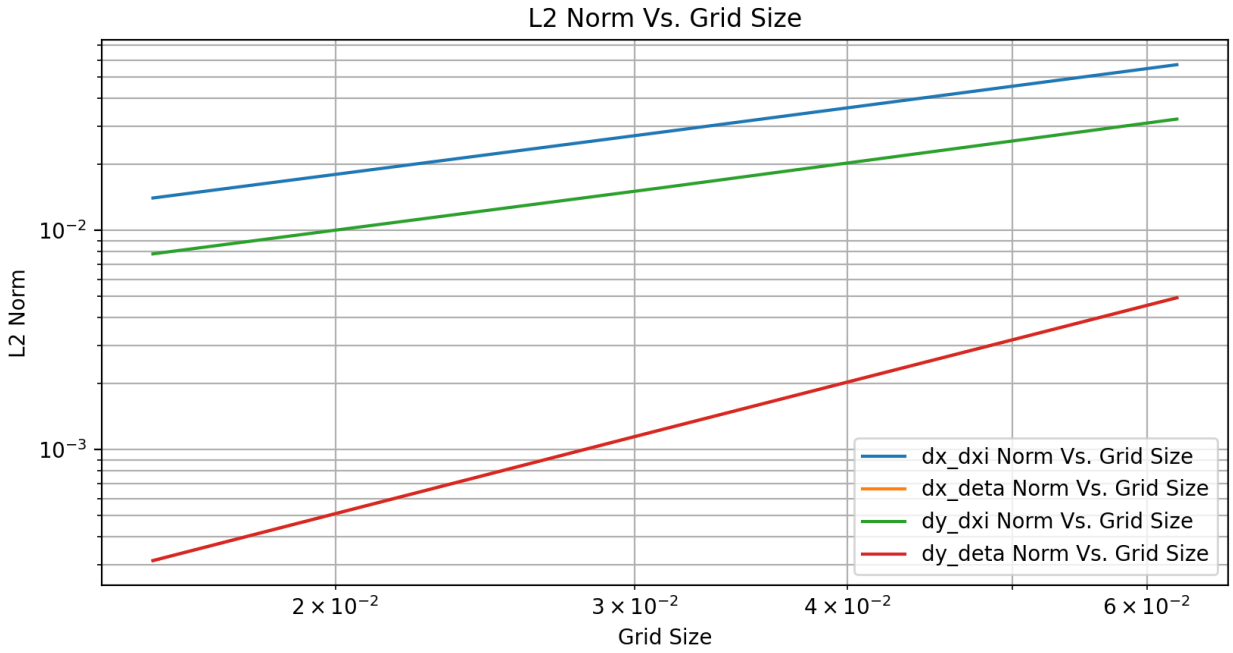


Figure 5: L2Norm Error Vs. Grid Size for Mesh Metrics

3. Computation of the flux integral

   To verify the computation of the flux integral, a problem from Mech 587 can be used. This problem involves the solution of the Laplacian equation over a unit domain, and has an exact solution of $u_e(x, y) = x^4+y^4-6x^2y^2$. The problem can be seen below in

$$\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = 0 \quad \text{in} \quad (0,1) \times (0,1),$$

$$u = \begin{cases} y^4, & x = 0, \quad 0 \le y \le 1 \\ x^4, & 0 \le x \le 1, \quad y = 0 \\ 1 - 6y^2 + y^4, & x = 1, \quad 0 \le y \le 1 \\ 1 - 6x^2 + x^4, & 0 \le x \le 1, \quad y = 1. \end{cases}$$

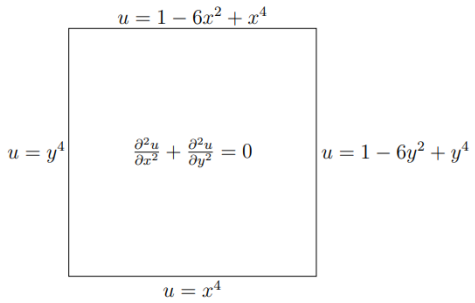


Figure 6: Computational Domain and Boundary Conditions for the Laplace Equation

This same problem can be used to validate the computation of the flux integral in this problem, utilizing a rectilinear grid generated with the MATLAB script previously discussed. If needed, further meshes with a higher complexity can be generated using the same domain.

Unfortunately, issues were found in the computation of the mesh metrics. I tried to fix them, but I could not figure out what the issue was. I spent as much time on this as I could, and ended up running out of time for the project. Overall, everything up until the mesh metrics worked, while the other functions did not work. I will spend as much time as possible to rectify these issues after, but I may have to accept my losses on this project and make up for it later. The code up until this point will be included in my submission.

In order to run the code at the point that it currently works, the following processes can be followed. The following zip files are included

- Project 1 Code.zip
- Project 1 Test Meshes.zip
- Project 1 Mesh Generation Scripts.zip

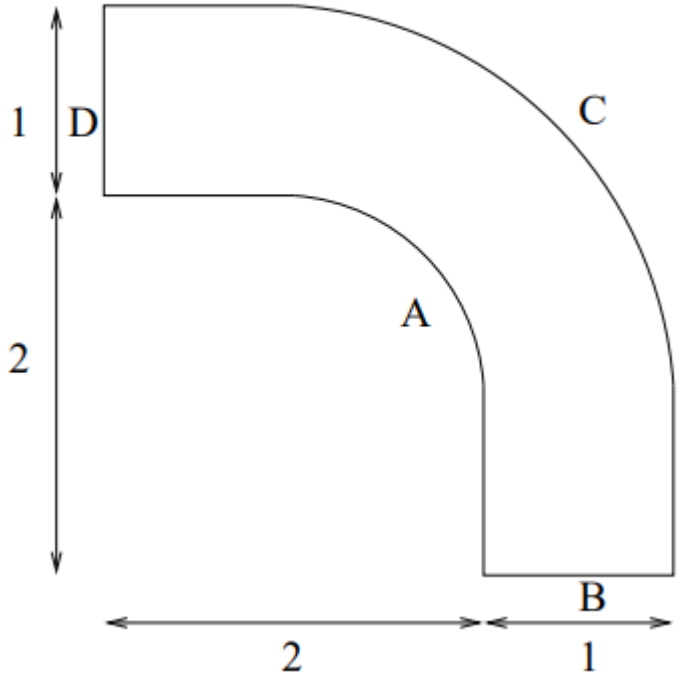
To run the main code, unzip the Project 1 Code.zip file and run the command make. The make file will compile all of the required code, and allow for running the code. In order to run the test files run the make test command, which will compile the required codes to run the tests. You will need to unzip the Test Meshes.zip file to allow the code to be ran with the appropriate mesh files. In order to run the different test cases simply uncomment them in the Test.cc file. In order to generate new scripts for the different test cases, simply unzip the Project 1 Mesh Generation Scripts.zip file. The mesh generation scripts for the rectilinear and slanted rectilinear meshes only have inputs of the number of points in η and ξ, while the mesh generation script for the quadratic case has inputs of η, ξ, max, and min where the max and min indicate the limits of the domain.

2. Write and verify a code that solves the heat equation to second-order accuracy in both time and space. You should use implicit time advance.

3. The geometry for the final non-rectilinear mesh problem is shown below. The boundary conditions are as follows:

   - Along A, B, and C, $T = 0$.
   - Along D, $T = -4(y-2)(y-3)$. This is a parabola with value of 0 at $y = 2$ and 3 and 1 at $y = 5/2$.

   The initial condition for the problem is 0.



Meshes will be provided on Canvas in 32×8, 64×16, and 128×32 sizes. In each case, i will run "tangentially" from $y = 0$ around to $x = 0$ and j will run radially from inside to outside. Data will be given for x and y for all cell corners on or inside the domain boundary.

Using these meshes, compute the solution T at time $t = 0.1$, and provide convincing evidence that your scheme is second-order accurate. In your write-up, be clear about what solver parameters you used to produce results that you present. (Roughly speaking, provide me with all the information I would need to be able to reproduce your results using your code if I were so inclined.)