Project 2: Heat Transfer Simulation of a Rocket Nozzle Wall

Part I (code verification) due Thursday, April 20 at 5 pm
Final Report due Wednesday, April 26 at 2:30 pm

Note: *Projects are meant to be open-ended and to allow some flexibility and creativity. Therefore it is important for you to show all relevant steps, numerical plots, and justifications for the choices made in your work.*

The Part I check-in should be turned in via Stellar as a PDF file. This step is important for verifying your code and making sure that you are on track to finish the project by the due date. Completing the deliverables in Part I will enable us to provide useful feedback. Turning in material for the check-in will count towards 5% of your project grade.

---

1 Background

Rocket propellant combustion temperatures often exceed the melting point of the materials used for the nozzle and thrust chamber. As a result many liquid-propellant rocket engines use regenerative cooling to make steady-state operation possible. In regenerative cooling, the fuel (or oxidizer) flows through an array of passages in the nozzle and thrust chamber walls, before being injected into the thrust chamber. The process is called regenerative because the heat lost from the hot combustion gases is not wasted: rather, it raises the temperature of the fuel before combustion.

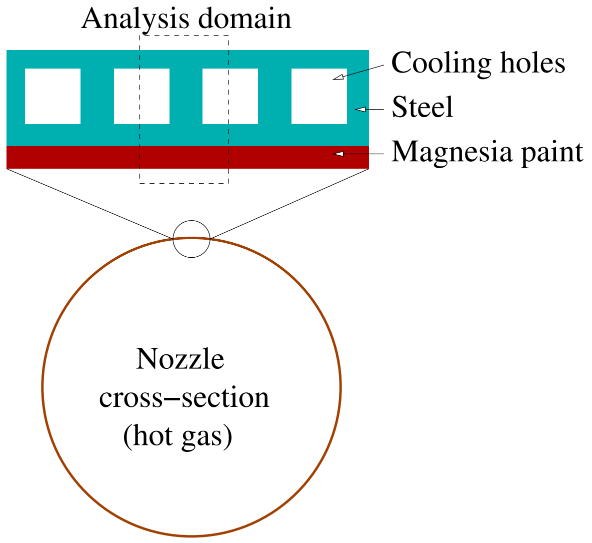

Figure 1: Cross-section of a regeneratively-cooled nozzle wall.

In this project, we will simulate the heat transfer in a regeneratively-cooled nozzle wall using a finite element discretization. A diagram of the nozzle wall cross-section is shown in Fig. 1. The nozzle wall is made of steel, with milled cooling holes. On the hot-gas side, a layer of magnesia paint is added for insulation. The nozzle diameter is much greater than the wall thickness, which allows us to neglect curvature and analyze a rectangular domain. In particular, assuming the cooling passages are evenly spaced, the analysis domain becomes the dashed rectangular region in Fig. 1.

We assume that the heat transfer between the hot gas and the inner wall (magnesia paint), and between the steel and the coolant occurs primarily by convection:

$$\mathbf{q} \cdot \mathbf{n} = h_{\text{ext}}(T - T_{\text{ext}}), \tag{1}$$

where $\mathbf{n}$ is the outward-pointing unit normal vector, so that $\mathbf{q} \cdot \mathbf{n}$ is the heat flux out of the material. In the material, $\mathbf{q} = -k\nabla T$ is the heat flux vector. T is the material temperature on the interface. h_{ext} is the convective heat transfer coefficient and T_{ext} is the bulk temperature of the external hot or cold gas. We assume constant h_{ext} and T_{ext} for this analysis, given by

Hot-gas side:	$h_{\text{ext}} = 10{,}000$ W/(m^2·K),	$T_{\text{ext}} = 3{,}000$ K,
Cold-gas side:	$h_{\text{ext}} = 20{,}000$ W/(m^2·K),	$T_{\text{ext}} = 300$ K.

Within the magnesia paint and the steel, heat is transferred by conduction according to

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

where k is the thermal conductivity. k is assumed constant for each material:

Steel:	$k = 40$ W/(m·K)
Magnesia paint:	$k = 2.5$ W/(m·K)

The minus sign in Eq. (2) is included for consistency with the conservative form derivation:

$$\nabla \cdot \mathbf{q} = 0, \qquad \mathbf{q} = -k\nabla T$$

2 Finite Element Analysis

Your primary task is to write a finite element solver to solve Eq. (2) for the steady-state temperature T, subject to the convection (Robin) boundary conditions in Eq. (1). Four triangular meshes of the nozzle analysis domain are provided for you in the MATLAB data file `Meshes.mat`. After executing `load Meshes` you will have the variables `NozzleMesh1`, `NozzleMesh2`, `NozzleMesh3`, and `NozzleMesh4`, each of which is defined as a MATLAB structure. A description of how to use these Mesh structures is given in the code shell `heat.m`. Each successive mesh is a uniform refinement of the previous mesh, and hence has four times the number of elements. Fig. 2 shows the elements that make up the first mesh, along with an enumeration of the four boundaries that your code will have to take into account. In all

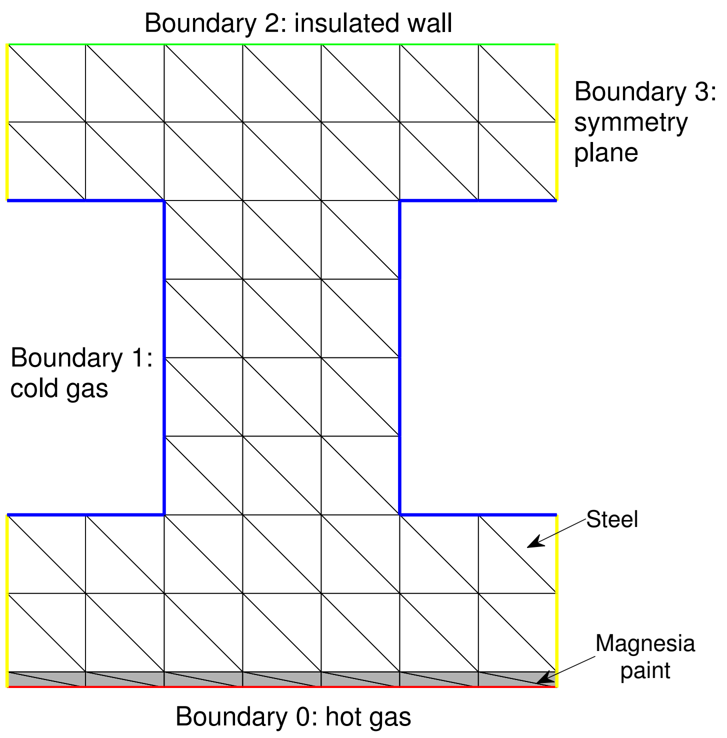

Figure 2: Coarse grid and boundary definitions for the nozzle analysis domain.

meshes, each element will be entirely within the steel material or entirely within the magnesia paint material.

You are given a shell of a finite element solver, `heat.m`. Comments in this file describe what each of the values in the Mesh structure mean, and where you need to add code. You are also given a script, `plotsolution.m`, that you can use to plot the temperature field via `plotsolution(Mesh, T)`, where `T` is the solution vector returned by `heat.m`. You can edit this script to customize your plots.

3 Part I: Code Verification

Fill in the code in `heat.m` as described in the comments in the file. You do not need to fill in the heat flux calculation until Part II. Clearly state in your writeup and code comments how you enforce the insulated wall boundary condition on Boundary 2 and the symmetry-plane boundary condition on Boundary 3.

To test whether your code works properly, run it using the simple `TestMesh` included in `Meshes.mat`. This mesh is shown in Fig. 3. Its boundaries follow the same numbering as the nozzle meshes, so you should not need to change your code to run this simplified domain.

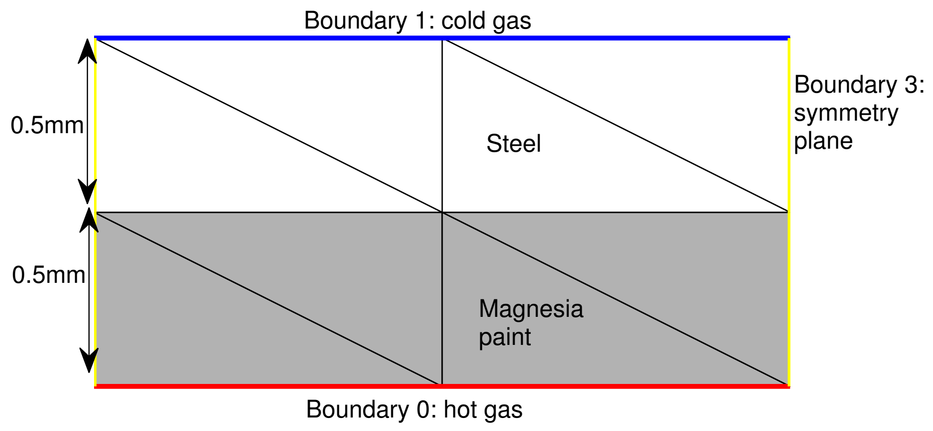

Figure 3: Test domain for code verification. The width of the domain is 2mm.

The advantage of the test domain is that it represents a one-dimensional heat transfer problem that you can solve by hand. Furthermore, since the solution is linear, if your code is working properly, the linear finite element solution will be exact to machine precision.

Deliverables

- Perform a 1D heat-transfer analysis of the test domain by hand, using the boundary conditions in Eq. (1) for the hot and cold sides. Remember to use the different conductivity values for the steel and magnesia paint. Specifically calculate three temperatures: the material temperature on the hot gas side, the temperature at the steel-magnesia paint interface, and the material temperature on the cold gas side.
- Compare your hand-calculated temperatures to ones from the code. Note, the nodes in the test mesh are ordered 1-3 on the hot gas side, 4-6 on the interface, and 7-9 on the cold gas side. The temperatures at the nodes should match your calculations.

4 Part II: Nozzle Solution

After verifying your code on the test mesh, run your code on the four nozzle meshes and perform the following tasks:

- Use the `spy` command to visualize the stiffness matrix. What controls the number of nonzero elements per row?
- Plot the temperature along the hot wall for mesh 4. Where is the temperature highest? Give a physical explanation for your observations.
- Calculate $T(1)$, the temperature of the first node, for all four meshes. Using the value from mesh 4 as the "truth" value, calculate the temperature errors, $T(1) - T_{\text{truth}}$. for

meshes 1-3. Between meshes 2 and 3, what is the convergence rate in the error with respect to mesh refinement?

Note: Calculate convergence rate as the exponent p in the expression $error = Ch^p$. Recall that the mesh size h is halved for each successive mesh.

Plot the three values of the $T(1)$ error on a log-log plot that clearly demonstrates the convergence rate.

- Add code in `heat.m` to calculate the integrated heat flux (units will be W/m) on the hot side and on the cold side of the analysis domain. Use Eq. (1) to calculate the heat flux. You should find that the heat flux integrated on the hot side exactly equals (in magnitude) the heat flux integrated on the cold side, on all the meshes. Explain why this happens.

  Note: To test your heat flux calculation, you might find it useful to run on the test mesh from Part I, on which you can calculate the integrated heat flux by hand.

- Give the value of the integrated heat flux from mesh 4, and using this value as the "truth" value plot the error convergence on meshes 1-3. What is the rate?

- **Bonus** When building the stiffness matrix K, use the `sparse(i,j,v)` command to build the sparse matrix all at once. This avoids indexed assignments that change the nonzero pattern of a sparse array, which can result in considerable overhead. Benchmark the two approaches for building K for mesh 4 using the `tic` and `toc` commands.

5 Part III: Parameter Study

In this part you will look at the sensitivity of the maximum temperature in the steel with respect to three parameters:

1. k_{magnesia}
2. $h_{\text{hot}} = h_{\text{ext}}$ on the hot gas side
3. $h_{\text{cold}} = h_{\text{ext}}$ on the cold gas side

Let each parameter vary between 0.75 and 1.25 of the original values given in the problem statement. Perform a sensitivity study by varying each one of the three parameters in turn, while holding the other two constant at the original values. When varying a parameter, choose enough points to make your results convincing.

For each parameter choice, calculate the maximum temperature *in the steel* on mesh 3. Note, on mesh 3, the steel temperatures are at nodes 117-833. You may find it useful to write a wrapper function that calls `heat.m` with parameters as inputs, and then post-processes the temperature vector.

Deliverables

- Plot T_{max} in the steel versus each parameter (three separate plots). To which parameter is the maximum steel temperature most sensitive?
- Provide temperature field plots (using the `plotsolution` script) at the extrema of the parameter ranges (six total field plots). Does the maximum steel temperature location change as the parameters are varied?
- **Bonus** Fix k_{magnesia} to its original value, and vary h_{hot} and h_{cold} *simultaneously* by evaluating T_{max} on a grid of h_{hot} and h_{cold} values. Visualize the resulting response surface using `surf` or the `contourf` commands. Comment on any the nonlinearity (or linearity) of the response surface you observed.

6 Report

Turn in a concise, but professional, report that includes the following items:

- A description of your implementation of the numerical techniques, including any relevant derivations and choices for parameter values.
- Results and discussion for each of the deliverables in Parts I – III.

Please upload your report as a PDF file to Stellar in addition to submitting a hardcopy in class. If you are printing in black and white, please make sure your plot styles remain legible.

Also, please upload your commented code to Stellar in a standalone zip file.