ASE 384P.4/EM 394F/PGE 383/CSE 393F
Coding Project 3

Due 11:59 PM November 29, 2017

The objective of this assignment is to give you experience of leveraging pre-existing code to rapidly develop new toolkits. In this case, the pre-existing code is what you developed for Project 2. The task is to adapt this code to solve the following partial differential equation:

$$\frac{\partial p}{\partial t} - \nabla \cdot (\mathbf{K}\nabla p) = q, \quad (\boldsymbol{x}, t) \in \Omega \times (0, T], \tag{1}$$

with boundary conditions given by

$$p = g_D(\boldsymbol{x}, t) \qquad \text{for} \quad \boldsymbol{x} \in \Gamma_D, t \in (0, T] \tag{2}$$

$$\boldsymbol{u} \cdot \boldsymbol{n} = g_N(\boldsymbol{x}, t) \qquad \text{for} \quad \boldsymbol{x} \in \Gamma_N, t \in (0, T] \tag{3}$$

where $\boldsymbol{u} = -\mathbf{K}\nabla p$,
and with initial condition

$$p(\boldsymbol{x}, 0) = p_0(\boldsymbol{x}) \quad \text{for} \quad \boldsymbol{x} \in \Omega \tag{4}$$

The finished solution should be capable of handling:

- 2-D grids
- quadrilateral element with bilinear shape functions
- Spatially varying full-tensor diffusion coefficient $\mathbf{K}$
- Spatially varying source term
- Point source term
- Spatially varying initial conditions
- Non-homogeneous, spatially varying Dirichlet and Neumann boundary conditions
- Backward Euler (implicit Euler) method for time discretization

For each part of the project, your program should be contained in a single .cc file and must produce point-centered VTK data readable by a visualization package like ParaView. For `matlab` codes, VTK outputs are not required but plotting capability is still needed.

- Use the VectorTools::project() command to populate the solution vector with the initial condition. For `matlab` codes, L_2 projection of the initial condition is also required.

Part(a): Benchmark

Set up your code in 2D for a unit square with vertices at $(0,0)$ and $(1,1)$ and,

$$\boldsymbol{K} = \begin{bmatrix} x_2 & 0 \\ 0 & x_1 \end{bmatrix}$$

$$\Gamma_D = \{\boldsymbol{x} | \boldsymbol{x} \in (\cdot, 0) \cup (1, \cdot)\}$$
$$\Gamma_N = \{\boldsymbol{x} | \boldsymbol{x} \in (\cdot, 1) \cup (0, \cdot)\}$$

The BVP described above admits the exact solution:

$$p = \cos(x_1 - x_2 + t)$$

You need to derive the complete governing system eq.(1) - (4) and conduct convergence study.

- Check the spatial convergence rate. With a very small time step so that it does not influence the spatial error, refine the mesh size (at least 4 times) and do the computation. Record and tabulate the L^2 error at the final time and calculate the spatial convergence rate. Compare your results with the expected convergence rate in space (conforming Galerkin with bi-linear elements).
- Check the temporal convergence rate. With a very small mesh size so that it does not influence the temporal error, refine the time step (at least 4 times) and do the computation. Record and tabulate the L^2 error at the final time and calculate the temporal convergence rate. Compare your results with the expected convergence rate in time (backward Euler method).

The workflow to check the spatial convergence rate is shown below.

- Set the time step to be Δt and keep it constant. Since we want to study the spatial convergence rate, the time step needs to be very small.
- Set the initial mesh size to be h.
- Set the final time to be T.
- Do the computation.
- Compute the L^2 error between the numerical solution and the analytical solution at the final time.
- Decrease the mesh size, do another computation and compute the L^2 error at the same final time. Repeat this process to get L^2 errors for different mesh sizes.
- Compute the spatial convergence rate.

The workflow to check the temporal convergence rate is shown below.

- Set the mesh size to be h and keep it constant. Since we want to study the temporal convergence rate, the mesh size needs to be very small.
- Set the initial time step to be Δt.
- Set the final time to be T.
- Do the computation.
- Compute the L^2 error between the numerical solution and the analytical solution at the final time.
- Decrease the time step, do another computation and compute the L^2 error at the same final time. Repeat this process to get L^2 errors for different time steps.
- Compute the temporal convergence rate.

Part(b): Application

After verifying your code from part (a), setup your code for the following problem.
The coefficient $\mathbf{K}$ is defined as:

$$\mathbf{K} = \begin{cases} \mathbf{I} & \text{if} \quad x_1 > 0.5 \\ 10\mathbf{I} & \text{if} \quad x_1 < 0.5 \end{cases}$$

The other input data is:

$$\begin{aligned} \Omega &= [0,1]^2 \\ \Gamma_D &= \partial\Omega \\ g_D &= 0 \\ q &= \begin{cases} (1+100t)\delta(x_1-0.5, x_2-0.5) & \text{if} \quad t \leq 0.01 \\ 2\delta(x_1-0.5, x_2-0.5) & \text{if} \quad t > 0.01 \end{cases} \\ p_0 &= 0 \end{aligned}$$

where q is a point source at the center of the domain whose coordinate is $(0.5, 0.5)$.

Possible extensions for bonus points (optional)

- User defined time discretization (10 points): Write a generalized time-stepping scheme that sets the value of θ at runtime. There are many scenarios which require this sort of a generalized approach, the most obvious being a motivation to study the accuracy and stability properties of different time discretizations. Besides the backward Euler method, you need to use the forward Euler (explicit Euler) and Crank-Nicolson methods to solve the above benchmark problem. To verify your code, you need to show the spatial and temporal convergence rates for the backward Euler method, and spatial convergence rates for the forward Euler and the Crank-Nicolson methods.

Submission guidelines

1.) Submit your code
Add a multi-line comment of the top of your source code that contains the following information:

- Your personal information - name, EID
- Line numbers where the following inputs are specified:
  - Diffusion coefficient
  - Forcing function
  - Dirichlet boundary values
  - Neumann boundary values
  - Reference finite element
  - Quadrature rule
  - Initial conditions
  - Projection of initial data
  - Point source
- A summary of features and capabilities which may extend beyond the mandatory part

Rename your source codes in the following format.

- For part (a): [eid]_benchmark.cc
- For part (b): [eid]_application.cc

Upload them to Canvas by 11:59 PM on 11/29/2017.

2.) Submit your report of the results
Summarize your results in a report including:

- The governing system for the benchmark problem including the governing equation, initial condition and boundary conditions.
- Convergence tables for the benchmark problem. Set the final time $T = 1$. For the backward Euler method, you need to make two seperate tables to study the spatial and temporal convergence rates respectively. In each table, you need to show the time steps, the mesh sizes, the number of cells, the number of degrees of freedom, the L^2 errors and the convergence rate. As for the optional part, the table of spatial convergence rate study is required for the forward Euler and Crank-Nicolson methods.
- Solution plots in high resolution for the benchmark problem at $T = 1$: (1) 2D plots showing the solution in color scale. (2) 3D plots showing the solution as a surface. The mesh needs to be 64 by 64 at least.

- Solution plots in high resolution for the application problem: (1) 2D plots showing the solution in color scale. (2) 3D plots showing the solution as a surface. Also plot solution profiles along $x_2 = 0.5$ at different times. Use a fine mesh and a small time step to do the computation. At minimum show the solution plots at the initial condition (t=0), at an early time (e.g. t=0.005) when the state variable starts to diffuse, at an intermediate time (e.g. t=0.01) and at a late time (e.g. t=0.02) when the state variable diffuses to the boundary.

Submit your report in class on 11/29/2017.

The templates of convergence tables are shown as below.

Table 1: Spatial convergence rate study

cycle	time step	mesh size	# cells	# dofs	L^2-error	spatial convergence rate
1	(very small)					
2	(very small)					
3	(very small)					
4	(very small)					

Table 2: Temporal convergence rate study

cycle	time step	mesh size	# cells	# dofs	L^2-error	temporal convergence rate
1		(very small)				
2		(very small)				
3		(very small)				
4		(very small)				

Option: use other programming languages

If you do not want to learn and use **deal.ii** library, you can choose to write your code from scratch using your favorite programming language such as `Matlab`. Your code needs to have the same capabilities described above.