MATH 676 Project for Wenyuan Li

Proposed project: A GMsFEM method for the heterogeneous Laplace equation Your project is concerned with developing a *generalized multiscale* finite-element method for the heterogeneous Laplace equation,

$$-\nabla \cdot \big(A(x)\nabla u\big) = f\,.$$

This is a prototypical equation that can exhibit a pronounced two (or multiscale) character depending on the coefficient matrix $A(x)$.

As a first part you will need to write a routine that given a cell patch ω_i generates a set of localized basis functions by solving an associated eigenvalue problem

$$-\nabla \cdot \big(A(x)\nabla \psi\big) = \lambda \tilde{k}(x)\psi, \quad \text{on } \omega_i,$$

with suitable boundary conditions. I will help you identifying a strategy for solving this eigenvalue problem fast and thread-parallelized. In order to store the result efficiently you will need to outline in detail how the final set of generalized basis functions enters the coarse problem.

The second part of your project consists of implementing the coarse problem incorporating your miscroscale sampling that you developed so far. You will need to demonstrate that your methods indeed converges under a suitable "refinement" strategy. For this it is best to simply overresolve the problem and create a highly refined reference solution. Then, compare your approximated coarse solution against this numerical ground truth.

Process. I suggest to have a look at `step-36` to read up on eigenvalue problems and learn how to use SLEPc for solving eigenvalue problems (even though this example step solves an unrelated PDE).

Then, it might be a good idea to start implementing your local cell problem, for example, by basing it on a modified version of `step-4`. Verify your code by making sure that you indeed solve the eigenvalue problem correctly and that the set of basis function you create make sense.

We will then explore how to parallelize the sampling procedure, i.e., how to solve the cell problem in parallel. We have various example steps in deal.II that discuss how to use multithreading for various purposes, I think `step-28` and `step-69` might have a good discussion that can help you. In any case, I will help you navigating this task.

As a final step you will have to implement the coarse problem which you can again (superficially) base on `step-4` but with a heavily modified assembly routine…

Note: This project is subject to modification by mutual agreement between you and me. The goal is to have a project that fits your research interests. If you would like to deviate from the path outlined here, talk to me!

Relevant tutorial programs: For your project, you will want to read through tutorial programs `step-4`, `step-36`, `step-28`, `step-69` . You should consider starting the program you will write for your project as a variation of one of these. (Links to all tutorial programs can be found at `https://www.dealii.org/developer/doxygen/deal.II/Tutorial.html`.)

Relevant video lectures: In addition to the resources already posted on the course website (Google Drive), you could watch lectures **13, 39, 40, 21.5, 21.55, 21.6, 34** for background material necessary for your project. Links to all videos can be found at `http://www.math.colostate.edu/~bangerth/videos.html`.

Keep notes on these resources and all background reads in your journal as on all other external resources you consult!

Project tasks: As part of your project, you will need to meet the following milestones:

- *Milestone 1 (April 4, 2023):*
  - Implement an eigenvalue solver for the cell problem. Verify that your generalized basis function is correct.
  - Start working on the thread parallelization.
  - You will need to prepare and give a 10 minute presentation in class on your progress so far.
- *Milestone 2 (May 8, 2023):*
  - Finish working on the thread parallelization.
  - Implement the coarse problem and validate that your method converges against a "numerical ground truth."
  - You will need to prepare and give a 15 minute presentation in class or during the final exam time on the results of your project.
  - You will also need to prepare a report on your project in the style of the tutorial program – i.e., including an introduction, a results section, and commented code that contains everything to run a simulation you show in your result section.

Deliverables: Your deliverables at the end of semester include the following items:

- Your final report (as discussed above in Milestone 2) as a PDF file, checked into a github repository to which you give me access.
- Your finished, documented code and all input files necessary to run it, checked into a github repository to which you give me access.
- Your finished online journal.

Grading: I will determine your final grade in this class based on the following criteria:

- Sophistication of the code beyond the program from which it was started.
- Extent of documentation in the code.
- Extent of the documentation surrounding the program, i.e., description of the equation and its properties, description of the principles used in the implementation, and documentation of worked-out examples computed with the program.
- Sophistication and realism of the testcases to which you apply your numerical scheme.
- quality and extend of entries in your journal,
- your three presentations in class.

As an example of how these reports should look like (though maybe not quite as extensive), take a look at the deal.II tutorial programs.

If you are interested, good projects may be published as part of the library and distributed with future versions (see for example the step-21, step-24, and step-25 tutorial programs that were created by students of a prior class), or as part of the code gallery (see `http://dealii.org/code-gallery.html`). Of course, you will then also be credited publicly for your work.