NUEN/PETE 689, HW-03. Due 11/22/2023 noon

Exercise 1: Affine decomposition for a 2D, parametric, diffusion-reaction problem.

You are provided with a 2D finite-element code that computes the solution of a 2D linear diffusion-reaction operator. The jupyter notebook is included in the assignment posting.

Material/source properties are piecewise constant per material/source zones. We will be

Your task is to transform this software into an affine-decomposed ROM. Specifically,

- you need to create a routine `assemble_system_per_zone` that computes the
  - stiffness and mass, matrices per material zone,
  - the matrices associated with the boundary,
  - the right-hand side vector per source zone,
  - the right-hand side per boundary side.
- you need to create a routine `combine_system_per_zone` that employs the output of the previous routine and, given material and source values, combines the various parts of the linear system into a final "Ax=b".
- You should verify that you obtain exactly the same answer whether you build the FOM system at once or you build the it using the affine decomposition of the operator.
- Then, you should generate samples for the parameters that will vary, generate snapshots for training and testing, perform an SVD, select a rank for your ROM order.
- Then, you should write a routine `compute_reduced_operator_affine` that, for a select basis U, computed the reduced affine-decomposed operators and right-hand sides, and a routine `solve_intrusive_rom_affine` that solves the ROM using the affine-decomposed operators
- Finally, you should assess how well your ROM reconstruction is as a function of the rank selected.
  - Note that the test case for which you should generate snapshots is already included in the jupyter notebook.

Hints:

`assemble_system_per_zone`: this routine should return a list of stiffness (typically called K) matrices, a list of mass matrices (typically called M), a list of boundary matrix (since we always be using 0 Dirichlet on all 4 sides, you can have a single boundary matrix). And similarly, for the right-hand side. Get inspired by the 1D code shown in class.

`combine_system_per_zone`: this routine combines the output of the `assemble_system_per_zone` routine using the values of the material properties, once through values are known.

`compute_reduced_operator_affine`: this routine is very similar to the same routine seen in the 1D code

`solve_intrusive_rom_affine`: this routine is very similar to the same routine seen in the 1D code