Dedalus

Kelvin-Helmholtz with Dedalus (a 2-D PDE)

Now that you have some experience with PDEs and Dedalus, let's take a look at the classic Kelvin-Helmholtz instability. This one can be solved on your laptops.

1. Use Mercurial and clone the Kelvin-Helmholtz repository located at `https://bitbucket.org/bpbrown/kelvin_helmholtz` If you have Mercurial on your system, this can be done with

   ```
   $ hg clone https://bitbucket.org/bpbrown/kelvin_helmholtz
   ```

   The Kelvin-Helmholtz problem will be downloaded and stored in a subdirectory named `kelvin_helmholtz`. If the `hg` command isn't recognized, please install Mercurial (`http://mercurial.selenic.com`).

   Run the basic problem by executing:

   ```
   python3 RT_incompressible.py
   ```

   and plot the results with

   ```
   python3 plot_results_parallel.py RT_incompressible slices 1 1 10
   ```

2. The equations for this Kelvin-Helmholtz problem are:

   $$\frac{\partial \boldsymbol{u}}{\partial t} + + \boldsymbol{\nabla} P - \frac{1}{Re}\boldsymbol{\nabla}^2 \boldsymbol{u} = -\boldsymbol{u} \cdot \boldsymbol{\nabla} \boldsymbol{u} \quad (1)$$

   $$\frac{\partial T}{\partial t} - \frac{1}{Pe}\boldsymbol{\nabla}^2 T = -\boldsymbol{u} \cdot \boldsymbol{\nabla} T \quad (2)$$

   $$\boldsymbol{\nabla} \cdot \boldsymbol{u} = 0 \quad (3)$$

   with vector velocity $\boldsymbol{u} = u\hat{\boldsymbol{x}} + w\hat{\boldsymbol{z}}$ and passive scalar tracer (here temperature) T. Here we have non-dimensionalized the equations based on the large-scale flow crossing time $\tau = L/U_0$, where L is the domain size and U_0 is the jump in amplitude of the horizontal shear flow across the tanh profile. The control parameters are the thermal Reynolds number Re and the Prandtl number Pr, with the Peclet number $Pe = RePr$. The initial horizontal flow and temperature field are given by a tanh. Open `equations.py` to find the equations as implemented. Note that here the boundary conditions are set up to continue enforcing the shear flow, and are set to be consistent with our initial u and T.

3. The provided case runs at $Re = 2500$ and $Pr = 1$. If you increase Re, you may also need to increase the nx and nz resolution. We suggest you take our approach, and specify the resolution in modes, then pad by a factor of $3/2$ for dealiasing.

4. The initial conditions, parameters, and simulation resolution are all set in `RT_incompressible.py`. This is also where the output and analysis is done, though the actual image generation is done in `plot_results_parallel.py`. The `RT.py` file includes a somewhat similar case, but with a different non-dimensionalization, while `RT_periodic.py` applies doubly-periodic boundary conditions. Best of luck!