# Spring 2026 EEPS1820
# Homework 2, due Feb 13, 11:59PM

COVERING: Wyngaard 2, 3, 4, 5; Oceananigans.jl; GitHub; Jupyter Notebooks

## Setting up your computer

We will be using julia, a computer language similar to python and matlab but much faster (like C++ or FORTRAN). We will also be using jupyter notebooks, to make it easy to visualize, annotate and share julia code and outputs. My github repository for the course is how I will share the code where you can begin work with you.

First, you need to install the key pieces of software. Begin by installing jupyter. Next install julia. Installation will be somewhat different depending on whether you are using Mac OSX, Linux, or Windows, but these applications are free and work on all of those platforms.

Second, start up the julia REPL (Read Evaluate Print Loop). To run julia REPL, you type "julia" in PowerShell, Terminal, or Bash. You can also find the julia icon in the start menu, applications folder, etc., and click or double-click on it to start the REPL. I've formatted this pdf in LaTeX so that you should be able to cut & paste the commands here easily into your julia REPL,

To integrate julia, you need to install the package IJulia. At the same time, we might as well install Oceananigans and GeophysicalFlows.

```
julia> using Pkg

julia> Pkg.add("IJulia")
```

Then, to install Oceananigans and GeophysicalFlows, along with a nice graphics package, do the following:

```
julia> Pkg.add("Oceananigans")
julia> Pkg.add("GeophysicalFlows")
julia> Pkg.add("CairoMakie")
```

If you are more familiar with Python, and you want to "call Python" from within julia, e.g., to plot figures with python commands or use a python package, then also install

```
 julia> Pkg.add("PyCall")
```

The same goes for "calling R" from within julia, e.g., to plot figures with R commands or use a R package, then also install

```
 julia> Pkg.add("RCall")
```

If you want to use C++ or FORTRAN functions, you can also do that as described here.

Now, let's get back to making julia compatible within jupyter notebooks. Launch jupyterlab the normal way, e.g.,

```
>> jupyter lab
```

You should now see that you should be able to start a julia notebook or julia console through jupyter! If you just want to run julia code, you can just open the julia REPL and go. You also can save a series of commands as a script, which is just a plain text file that you name to end in .jl . Jupyter notebooks are a bit easier than plain scripts, since they lead to self-documentation.

## Some julia tutorial material

There are lots of videos, tutorials, etc., about julia online. Here are some key resources.

- In the julia REPL, just type a question mark to access help mode, e.g.,

  ```
  julia> ?
  ```

  and follow this with a function, command, or package. Try "? Oceananigans", which will look like

  ```
  julia> ?
  help?>Oceananigans
  ```

- Here is the Oceananigans tutorial and some key papers (Ramadhan et al., 2020; Silvestri et al., 2023; Wagner et al., 2025).
- There are good julia practice exercises at exercism.org.
- Another student recommends this set of blog posts and this explanation of data structures.
- There are instructor-led videos at julia Academy and the julia YouTube.
- There is always the julia documentation.
- These references and more can be found at this julia learning page.
- Here are the GeophysicalFlows examples and paper (Constantinou et al., 2021).

## 1 Problem 1: Getting the jupyter notebook working

### 1.1 Finding the PS02.ipynb Notebook

The jupyter notebook is here. You can download it directly and then save it for editing. You will be editing the PS02 jupyter notebook to complete your homework assignment, but not sending it back into the class repository.

### 1.2 Opening the PS02.ipynb Notebook

You can open jupyter however you like and then open the notebook you just downloaded, or you can go to the directory where it is at the command line and type

```
>> jupyter notebook PS02.ipynb
```

Execute the steps in the notebook one at a time, by hitting Shift-Return or clicking on the play button at the top. Note the output that comes back from each step! When you reach the end of the notebook, you will be asked to upload particular figures to canvas and explain them.

## 2 Add in Uextra

This problem asks you to figure out where to add a perturbation to the initial velocity field on the second simulation (the one with the movie output). Figure this out, add Uextra, rerun the simulation, and then explain the results. Again, you'll need to upload the figures and explain them. This time, contrast them with the previous result without Uextra.

## 3 Wyngaard (2010) Problem 3.1

$$\frac{\partial U_i}{\partial t} + \frac{\partial}{\partial x_j}\left[U_i U_j + \overline{u_i u_j} - \nu\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right)\right] = -\frac{1}{\rho}\frac{\partial P}{\partial x_i}. \tag{3.7}$$

3.1 Starting from Eq. (3.7), form the equation governing the evolution of the kinetic energy of a steady, ensemble-averaged flow.

(a) Is the kinetic energy of the ensemble-averaged flow the same as the ensemble-averaged kinetic energy? Discuss.

(b) Write the term involving the viscous stress as the sum of a divergence and a dissipative term.

(c) Write the remaining terms as the sum of a divergence and a remainder, and integrate the equation over a volume. Interpret the result.

(d) Interpret the volume-integrated equation for steady flow in a pipe. What can you conclude about the role of the turbulent flux term?

You do not need to do part (e)

## 4 Wyngaard (2010) Problem 3.11

3.11 Show with an example that the local average, Eq. (3.29), does not satisfy averaging rule (2.6).

$$\overline{f}^{\mathrm{loc}}(x,t,\Delta) = \frac{1}{\Delta}\int_{-\Delta/2}^{\Delta/2} f(x+x',t)\,dx'. \tag{3.29}$$

$$\overline{\overline{a}} = \overline{a} \quad (\overline{A} = A). \tag{2.6}$$

## 5 Wyngaard (2010) Problem 3.12

3.12 Show that the local average, Eq. (3.29), commutes with both derivatives.

Hint: "Both derivatives" means $\frac{\partial}{\partial t}$ and $\frac{\partial}{\partial x}$

## References

Constantinou, N. C., Wagner, G. L., Siegelman, L., Pearson, B. C., and Palóczy, A. (2021). Geophysicalflows.jl: Solvers for geophysical fluid dynamics problems in periodic domains on cpus & gpus. *Journal of Open Source Software*, 6(60):3053.

Ramadhan, A., Wagner, G., Hill, C., Campin, J.-M., Churavy, V., Besard, T., Souza, A., Edelman, A., Ferrari, R., and Marshall, J. (2020). Oceananigans. jl: Fast and friendly geophysical fluid dynamics on gpus. *Journal of Open Source Software*, 5(53).

Silvestri, S., Wagner, G. L., Hill, C., Ardakani, M. R., Blaschke, J., Campin, J.-M., Churavy, V., Constantinou, N. C., Edelman, A., Marshall, J., et al. (2023). Oceananigans. jl: A julia library that achieves breakthrough resolution, memory and energy efficiency in global ocean simulations. *arXiv preprint arXiv:2309.06662*.

Wagner, G. L., Silvestri, S., Constantinou, N. C., Ramadhan, A., Campin, J.-M., Hill, C., Chor, T., Strong-Wright, J., Lee, X. K., Poulin, F., et al. (2025). High-level, high-resolution ocean modeling at all scales with oceananigans. *arXiv preprint arXiv:2502.14148*.

Wyngaard, J. C. (2010). *Turbulence in the Atmosphere*. Cambridge University Press.