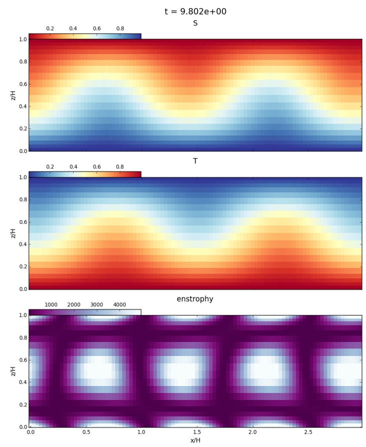
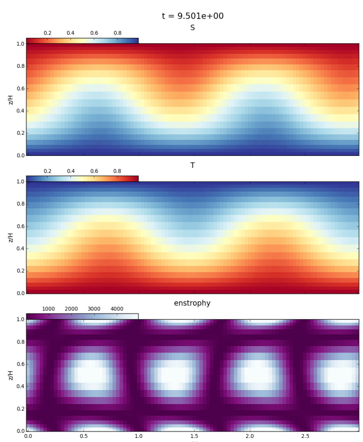
**Problem Set 9: Dedalus and doubly-diffusive convection (a 2-D PDE)**

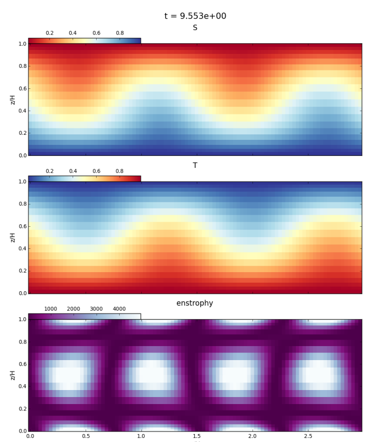
0. Through dangers untold and hardships unnumbered...installed.

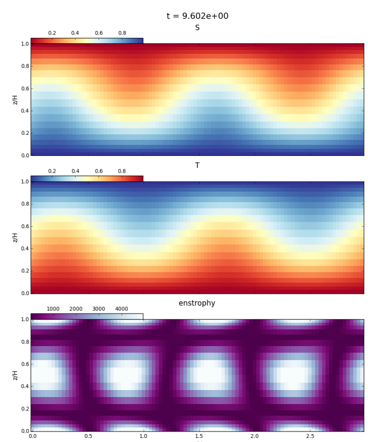
1. And it works! (as long as its own version of matplotlib is not used...)




2. These are travelling waves, as the following images show.




3.
   - The implementation of the equations of thermosolutal convection are detailed in Table 1.
   - The role of equations like $dz(w) - w_z = 0$ are to keep the problem first order. As the x differentiation is essentially just a scalar multiplication, it is not necessary to do this for the x-derivatives, but for z-derivatives this simplifies the problem.
   - The equations are correctly implemented.
   - The boundary conditions are ([bottom,top]):
     $S = [1, 0]$, $T = [1, 0]$, $u = [0, 0]$, $w = [0, 0]$, $\frac{\partial w}{\partial x}|_{\text{bottom}} \neq 0$, $P(\text{bottom}) = 0$, $\frac{\partial P}{\partial x}|_{\text{bottom}} = 0$

4. See Table 2. In `equations.py`, the linear terms are all grouped on the left-hand side of the equation, and the non-linear terms are all on the right.

5. T is destabilizing, while S is stabilizing. To change their relative roles I would set their boundary conditions to $[0, 1]$, rather than $[1, 0]$.

| Equation | Implementation | |
|---|---|---|
| (1) | `1/Pr*dt(w)  - (dx(dx(w)) + dz(w_z)) - Ra_T*T + Ra_S*S + dz(P) = -1/Pr*(u*dx(w) + w*w_z)` | |
| | `1/Pr*dt(u)  - (dx(dx(u)) + dz(u_z)) + dx(P) = -1/Pr*(u*dx(u) + w*u_z)` | |
| (2) | `dx(u) + w_z = 0` | |
| (3) | `dt(T) - (dx(dx(T)) + dz(T_z)) = -u*dx(T) - w*T_z` | |
| (4) | `dt(S) - Lewis*(dx(dx(S)) + dz(S_z)) = -u*dx(S) - w*S_z` | |
| **Implementation of** | **Translation** | **(in words)** |
| (1) | $\frac{1}{Pr}\frac{\partial w}{\partial t} - \nabla^2 w\hat{z} - (Ra_T T - Ra_S S)\hat{z} + \frac{\partial P}{\partial z} = -\frac{1}{Pr}\mathbf{u}\cdot\nabla w$ | $\hat{z}$ component of (1) |
| | velocity current (in z), diffusion of velocity, buoyancy/sinking, pressure flux, advection of velocity | |
| | $\frac{1}{Pr}\frac{\partial u}{\partial t} - \nabla^2 u\hat{x} + \frac{\partial P}{\partial x} = -\frac{1}{Pr}\mathbf{u}\cdot\nabla u$ | $\hat{x}$ component of (1) |
| | velocity current (in x), diffusion of velocity, pressure flux, advection of velocity | |
| (2) | $\frac{\partial u}{\partial x} + \frac{\partial w}{\partial z} = 0$ | the gradient of u, which = 0 |
| | outward flux of velocity is 0 | |
| (3) | $\frac{\partial T}{\partial t} - \nabla^2 T = -\mathbf{u}\cdot\nabla T$ (I condensed the Laplacians and gradients in all of these...but I think that still IDs each term ok) | |
| | temperature current, temperature diffusion, temperature advection | |
| (4) | $\frac{\partial S}{\partial t} - \tau\nabla^2 S = -\mathbf{u}\cdot\nabla S$ | |
| | solute current, solute diffusion, solute advection | |

Table 1: Translation of equations of thermosolutal convection from TS.py code.

| Equation | Linear | Nonlinear |
|---|---|---|
| (1) | $\frac{1}{Pr}\frac{\partial \mathbf{u}}{\partial t}, \nabla^2\mathbf{u}, (Ra_T T - Ra_S S)\hat{z}, \nabla P$ | $\mathbf{u}\cdot\nabla\mathbf{u}$ |
| | `1/Pr*dt(w), dx(dx(w)), dz(w_z), -Ra_T*T, Ra_S*S, dz(P)` | `-1/Pr*u*dx(w), -1/Pr*w*w_z` |
| | `1/Pr*dt(u), dx(dx(u)), dz(u_z), dx(P)` | `-1/Pr*u*dx(u), -1/Pr* w*u_z` |
| (2) | $\nabla\cdot\mathbf{u}$ | - |
| | `dx(u), w_z` | - |
| (3) | $\frac{\partial T}{\partial t}, \nabla^2 T$ | $\mathbf{u}\cdot\nabla T$ |
| | `dt(T), dx(dx(T)), dz(T_z))` | `-u*dx(T), - w*T_z` |
| (4) | $\frac{\partial S}{\partial t}, \tau\nabla^2 S$ | $\mathbf{u}\cdot\nabla S$ |
| | `dt(S), dx(dx(S)), dz(S_z))` | `-u*dx(S), -w*S_z` |

Table 2: The linear and nonlinear breakdown of the equations in both raw and implemented form.

Code used in this assignment:

```
#!bash
python3 TS.py
python3 plot_results_parallel.py TS slices 1 1 10
```