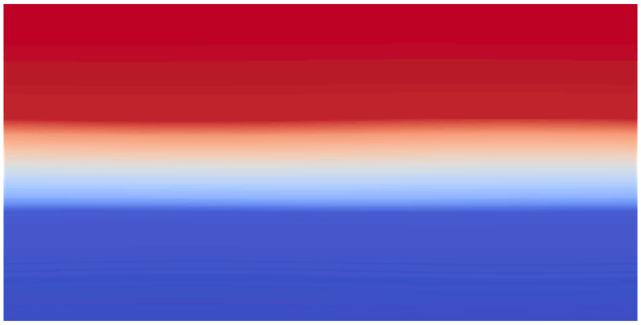
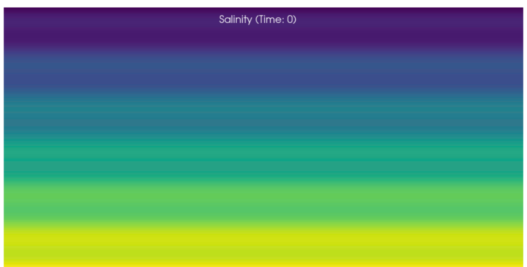
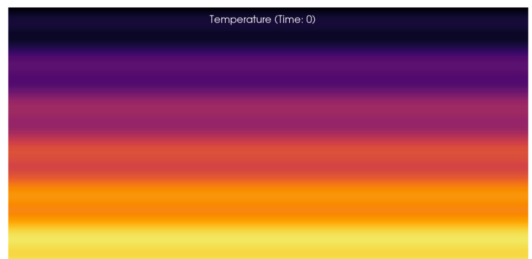
Double-Diffusive Convection: An Extensional Module of Channelflow



Salinity behavior of 2D finger regime





Salinity and temperature behaviors of 2D diffusive regime in Couette flow configuration.

ChannelFlow-DoubleDiffusiveConvection is an extensional module of Channelflow 2.0 for wall-bounded double-component problems like Double-Diffusive Convection and Binary Fluid Convection. To use this code, pls read following instruction to install Channelflow (this also contains setup on a HPC)

After knowing how to install the standard Channelflow, you can clone them to your local machine and add present `ddc` module inside by

```
git clone https://github.com/epfl-ecps/channelflow.git

cp ./CMakeLists.txt ./channelflow/CMakeLists.txt
mkdir -p ./channelflow/modules/
rm -rf ./channelflow/modules/ddc
cp -r ./ddc ./channelflow/modules/ddc
```

And build them, for example

```
mkdir -p build
cd build
cmake ../channelflow -DCMAKE_CXX_COMPILER=/usr/bin/mpicxx -DWITH_DDC=ON -DWITH_NSOLVER=ON -DCMAKE_
make -j16
```

Building DNS

First of all, you need to define governing equations of problem. In this code, we offer nondimensional governing equations, which have form:

ParseError: KaTeX parse error: {align} can be used only in display mode.

where $\boldsymbol{u}$, θ, and s are perturbations of the velocity, first scalar (temperature), and second scalar (salinity) fields. If you can not see equations, let read pdf file instead. Because this code offers a general form of governing equations, so first you need to define the problem you want to use in this code. This is perfomed by modifying controlling parameters (p_i) via a header file `ddc/macros.h`. Here, we suggest some governing equations which are nondimensionalized for specific problems:

Parameters	**Double-diffusive convection (Singh & Srinivasan 2014)**	**Double-diffusive convection (normalized by free-fall velocity) (Yang et al. JFM 2021)**	**Binary fluid convection (Mercader et al. JFM 2013)**	**Stratified plane Couette flow (Langham et al. JFM 2019)**	**Description**
p_1	Pr_T	$\sqrt{\frac{Pr_T}{Ra_T}}$	Pr	$1/Re$	$Pr_T = \frac{\nu}{\kappa_T}$ is Prandtl number
p_2	$Pr_T Ra_T$	$\frac{Ri R_\rho}{1-R_\rho}$	$Pr_T Ra_T$	Re	$Ra_T = \frac{g\alpha\Delta_T H^3}{\nu\kappa_T}$ is Thermal Rayleigh

Parameters	Double-diffusive convection (Singh & Srinivasan 2014)	Double-diffusive convection (normalized by free-fall velocity) (Yang et al. JFM 2021)	Binary fluid convection (Mercader et al. JFM 2013)	Stratified plane Couette flow (Langham et al. JFM 2019)	Description
					number
p_3	1	1	$1+R_{sep}$	$-Ri$	R_{sep} is Separation ratio
p_4	$\frac{1}{R_\rho}$	$\frac{1}{R_\rho}$	R_{sep}		$R_\rho = \frac{\alpha\Delta_T}{\beta\Delta_S}$ is Density stability ratio
p_5	1	$\frac{1}{\sqrt{Pr_T Ra_T}}$	1	$\frac{1}{RePr}$	$Re = \frac{Uh}{\nu}$ is Raynolds number
p_6	$\frac{1}{Le}$	$\frac{1}{Le\sqrt{Pr_T Ra_T}}$	$\frac{1}{Le}$		$Le = \frac{\kappa_T}{\kappa_S}$ is Lewis number
p_7			1		Ri is Richardson number

Notes that if you don't define p_5 or p_6, corresponding scalar's governing equations and buoyancy components (p_3, p_4) in momentum equations will be removed. You can create a new your own governing equations.

A correct defination looks like this

```
// Example 1: Stationary-wall bounded double-diffusive convection [Yang2016PNAS]
// Boundary velocity is not normalized into unit velocity, don't know exact U0
// Only use this form for stationary walls
// #define P1 Pr
// #define P2 Pr*Ra
// #define P3 1.0
// #define P4 1.0/Rrho
// #define P5 1.0
// #define P6 1.0/Le

// Example 2: Moving-wall bounded double-diffusive convection [Yang2021JFM]
// Velocity is normalized by free-fall velocity into unit velocity, U0=1.0
// For example, Ua=-0.5 Ub=0.5 or Ua=0 Ub=1
#define P1 sqrt(Pr/Ra)
#define P2 1.0
#define P3 1.0
#define P4 (1.0/Rrho)
#define P5 (1.0/sqrt(Pr*Ra))
#define P6 (1.0/(Le*sqrt(Pr*Ra)))

// Example 3: Binary fluid convection [Mercader2013JFM]
// #define P1 Pr
// #define P2 Pr*Ra
// #define P3 (1.0+Rsep)
// #define P4 Rsep
// #define P5 1.0
// #define P6 1.0/Le
// #define P7 1.0

// Example 4: Couette flow
// #define P1 1.0/Rey

// Example 5: Stratified plane Couette flow [Langham2019JFM]
// #define P1 1.0/Rey
// #define P2 Rey
// #define P3 (-Ri)
// #define P5 1.0/(Rey*Pr)
```

Running executable files

A exact executable command likes this:

```
mpiexec -n <ncpu> ./<exename> <option1> <option2> ...
```

with controling parameters

Option	Default	Description
`-Nx <value>`	200	Number of points along x-direction
`-Ny <value>`	101	Number of points along y-direction, notes that N_y is odd number
`-Nz <value>`	6	Number of points along z-direction, minimal number is 6 (can be used for 2D setup)
`-Lx <value>`	2	Streamwise length
`-Lz <value>`	0.004	Spanwise length, default setup is a small length representing 2D domain
`-Pr <value>`	10	Prandtl number $Pr = \frac{\nu}{\kappa_T}$
`-Ra <value>`	10^3	Thermal Rayleigh number $Ra_T = \frac{g\alpha\Delta T H^3}{\nu\kappa_T}$
`-Le <value>`	100	Lewis number $Le = \frac{\kappa_T}{\kappa_S}$
`-Rr <value>`	2	Density stability ratio $R_\rho = \frac{\alpha\Delta T}{\beta\Delta S}$
`-Ua <value>`	0	X-velocity at lower wall, U(y=a)
`-Ub <value>`	0	X-velocity at upper wall, U(y=b)
`-Wa <value>`	0	Z-velocity at lower wall, W(y=a)
`-Wb <value>`	0	Z-velocity at upper wall, W(y=b)
`-Ta <value>`	0	Temperature at lower wall, T(y=a)
`-Tb <value>`	1	Temperature at upper wall, T(y=b)

Option	Default	Description
`-Sa <value>`	0	Salinity at lower wall, S(y=a)
`-Sb <value>`	1	Salinity at upper wall, S(y=b)
`-T0 <value>`	0	Start time of DNS
`-T <value>`	20	Final time of DNS
`-dt <value>`	0.03125	Timestep
`-dT <value>`	1	Save interval
`-nl <value>`	"rot"	Method of calculating nonlinearity, one of [rot|conv|div|skew|alt|linear]

Examples:

```
mpiexec -n 16 ./ddc_simulateflow -Pr 10 -Ra 1000 -Le 100 -Rr 2 -dt 0.02 -dT 1 -T 100 -Nx 200 -Ny 8
```