Double-Diffusive Convection: An Extensional Module of Channelflow

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, the governing equations of a double-component system have form:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}_{\rm tot} \cdot \nabla \boldsymbol{u}_{\rm tot} = -\nabla p + p_1 \nabla^2 \boldsymbol{u} + p_1 p_2 (p_3 \theta - p_4 s)\mathbf{j}, \tag{1}$$

$$\frac{\partial \theta}{\partial t} + \boldsymbol{u}_{\rm tot} \cdot \nabla \theta_{\rm tot} = p_5 \nabla^2 \theta, \tag{2}$$

$$\frac{\partial s}{\partial t} + \boldsymbol{u}_{\rm tot} \cdot \nabla s_{\rm tot} = p_6 \nabla^2 s + p_7 \nabla^2 \theta, \tag{3}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{4}$$

where $\boldsymbol{u}$, θ, and s are fluctuations of the velocity, temperature, and a third field. The third field s may be the salinity in double-diffusive convection (Radko 2013) or the convective mass flux in binary fluid convection (Mercader 2013). The subscript `tot` indicates the total value of fields, which is defined as sum of base flow and fluctuation of each field. Because this code offers two options DDC and BFC, so first you need to define the problem you want to use in this code. This is perfomed by modifying controling parameters (p_i) via a header file `ddc/macros.h`.

Parameters	**Double-diffusive convection**	**Binary fluid convection**	**Description**
p_1	Pr	Pr	Pr is Prandtl number
p_2	Ra	Ra	Ra is Thermal Rayleigh number
p_3	1	$1 + R_{sep}$	R_{sep} is Separation ratio
p_4	$1/R_\rho$	R_{sep}	R_ρ is Density stability ratio
p_5	1	1	
p_6	$1/Le$	Le	Le is Lewis number
p_7	0	1	

A correct defination will likes this

```
// Example 1: for double-diffusive convection [Yang2016PNAS]
#define P1 Pr
#define P2 Ra
#define P3 1.0
#define P4 1.0/Rrho
#define P5 1.0
#define P6 1.0/Le
#define P7 0.0

// Example 2: for binary fluid convection [Mercader2013JFM]
// #define P1 Pr
// #define P2 Ra
// #define P3 (1.0+Rsep)
// #define P4 Rsep
// #define P5 1.0
// #define P6 Le
// #define P7 1.0
```

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 $N_y\%2=1$
`-Nz <value>`	10	Number of points along z-direction
`-Lx <value>`	2	Streamwise length
`-Lz <value>`	0.05	Spanwise length
`-Pr <value>`	10	Prandtl number $Pr=\frac{\nu}{\kappa_T}$

Option	Default	Description
`-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)
`-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 81 -Nz 10 -Lx 2 -Lz 0.02 -bf "laminar" -nl "conv"
```