Cubby Code

Y. Ponty[1], A. Minuissi[2]
(1) Laboratory Cassiopée, UMR 6202 CNRS,
(2) UMS Galilée , (IR) Mesocentre Crimpson
Observatoire de la Côte d'Azur
B.P. 4229, 06304 Nice Cedex 04, France

1 Equations and physical features

The cubby code is design to solve a magnetohydrodynamic system with or without a passif scalar. But it provide also a general library to solve any partial differential equation in a periodic box with a pseudo-spectral method.

$$\partial_t \mathbf{V} + \mathbf{V}.\nabla\mathbf{V} = -\nabla(\frac{P}{\rho}) + (\nabla \times \mathbf{B}) \times \mathbf{B} - \mathbf{\Omega} \times \mathbf{V} + \nu\Delta\mathbf{V} + \mathbf{F} \quad (1)$$

$$\nabla.\mathbf{V} = 0 \quad (2)$$

$$\partial_t \mathbf{B} = \nabla \times (\mathbf{V} \times \mathbf{B}) + \eta\Delta\mathbf{B} \quad (3)$$

$$\nabla.\mathbf{B} = 0 \quad (4)$$

$$\partial_t \Theta + \mathbf{V}.\nabla\Theta = \kappa\Delta\Theta + F_\Theta \quad (5)$$

There is also the possiblity to solve the induction equation or/and the passif scalar with fluctuation velocity only $(\mathbf{V} - \mathbf{V}_0)$.

1.1 Kinematic problem

Assuming a given velocity $\mathbf{V}(\mathbf{x}, t)$, We are solving only the passif vector of passif scalar equation.

$$\partial_t \mathbf{B} = \nabla \times (\mathbf{V}(\mathbf{x}, t) \times \mathbf{B}) + \eta\Delta\mathbf{B} \quad (6)$$

$$\partial_t \Theta + \mathbf{V}(\mathbf{x}, t).\nabla\Theta = \kappa\Delta\Theta + F_\Theta \quad (7)$$

1.2 numerical schemes

The Navier-Stoke equation is project on the divergence free space with the Π projector. The inertial term is computed with form $\mathbf{V} \times \omega$ with $\omega = \nabla \times \mathbf{V}$. The set of equations is solved in the spectral space.

$$\partial_t \mathbf{V} - \nu\Delta\mathbf{V} = \Pi(\mathbf{V} \times \omega + (\nabla \times \mathbf{B}) \times \mathbf{B} - \mathbf{\Omega} \times \mathbf{V} + \mathbf{F}) \quad (8)$$

$$\partial_t \mathbf{B} - \eta\Delta\mathbf{B} = \nabla \times (\mathbf{V} \times \mathbf{B}) \quad (9)$$

$$\partial_t \Theta - \kappa\Delta\Theta = -\mathbf{V}.\nabla\Theta + F_\Theta \quad (10)$$

1.3 Temporel Schemes

The equations are on the form :

$$\partial_t U - \nu U = G \tag{11}$$

The diffusion term is implemented implicitly with the exponentiel method.

$$\partial_t U_k(t) = -\nu k^2 U_k(t) + G_k(t) \tag{12}$$

Equation (12) can be written :

$$\partial_t (e^{\nu k^2 t} U_k) = e^{\nu k^2 t} G_k(t) \tag{13}$$

• The first time step is solved using the the forward Euler - second order Runge-Kutta Method. at $t + \frac{\Delta t}{2}$

$$U_k(t + \frac{\Delta t}{2}) = (U_k(t) + \frac{\Delta t}{2} G_k(t)) e^{-\nu k^2 \frac{\Delta t}{2}} \tag{14}$$

Secondly : computation of $G_k(t + \frac{\Delta t}{2})$ with $U_k(t + \frac{\Delta t}{2})$
Finaly :

$$U_k(t + \Delta t) = U_k(t)\ e^{-\nu k^2 \Delta t} + \Delta t\ G_k(t + \frac{\Delta t}{2})\ e^{-\nu k^2 \frac{\Delta t}{2}} \tag{15}$$

• For the other time step the forward Euler - Adams-Basford shemes is implemented :

$$\frac{e^{\nu k^2 (t+\Delta t)} U_k(t + \Delta t) - e^{\nu k^2 t} U_k(t)}{\Delta t} = \frac{3}{2} G_k(t) e^{\nu k^2 t} - \frac{1}{2} G_k(t - \Delta t) e^{\nu k^2 (t - \Delta t)} \tag{16}$$

$$U_k(t + \Delta t) \ = \ U_k(t) e^{-\nu k^2 \Delta t} + e^{-\nu k^2 \Delta t}\ \Delta t\ \left[\frac{3}{2} G_k(t) - \frac{1}{2} G_k(t - \Delta t) e^{-\nu k^2 \Delta t} \right] \tag{17}$$

2 Compilation

Type ./preconfigure command if you computer is inside the lists,

prompt$ ./preconfigure
prompt$ make
The name of the executable file is *cubby*.

otherwise ./configure with the right option for your system. look at the "preconfigure" file, there are plenty of example which could be helpful.
Example of configure :

prompt$ export BOOSTLIB="gcc41-mt"
prompt$ export BOOSTFLAGS=" –with-boost-mpi=$BOOSTLIB –with-boost-serialization=$BOOSTLIB –with-boost-program_options=$BOOSTLIB"
prompt$ export CXXFLAGS='-O3 -DBOOST_DISABLE_ASSERTS'
prompt$ ./configure –with-fft=fftw –with-fft-home=/opt/fftw –with-boost=/opt/boost $BOOSTFLAGS

3 Precompilation options

In the cubby.hpp header there is two options :
#define opt_vectorization force some vectorial loop for (NEC-SX) computer
#define MPI2_On Activation of the MPI2 IO (by default)

4 cubby.cfg, cubby.data, point.data

4.1 Dimension parameter

The grid resolution could be chosen in the command line or inside the **cubby.cfg** file.
prompt$ cubby −−cube-dim 64
prompt$ cubby −−x-dim=64 −−z-dim=128
prompt$ mpirun -np 4 cubby −−cube-dim 128 > resu-cubby

command	type	comment
−−cube-dim	integer	cube grid resolution
−−x-dim	integer	x grid resolution
−−y-dim	integer	y grid resolution
−−z-dim	integer	z grid resolution
−−aspect-ratio	integer	aspect ratio along the z axis (default=1)

Note the x-dim = y-dim are equal, and must be a multiple of the number of processors.
z-dim could be chosen independtly, but lower is primary number decomposition, faster the fft will be.

The dimension could be set up also inside the **cubby.cfg** file :
prompt$ more cubby.cfg

x-dim = 128
z-dim = 256
aspect-ratio = 2

With the **aspect-ratio** > 1, the simulation domain will $(2\pi)^2$ in the xy plane
and [aspect-ratio *(2π)] along the z-axis.
In this case the wave number $kz = \frac{k}{aspect-ratio}$ with $k = 0, ..., \text{z-dim}/2$.
(In the following document **aspect-ratio** will be also noted as a_r.)

4.2 cubby.data, point.data

The cubby.data is necessary to run the code, and provide all the parameters and options available inside the two sub section bellow. The point.data gives the position of the probe re-scale inside a $[0 .. 1.0]^3$ cube.

See an example of the **point.data** file which provide 10 probes along a diagonale of the cube :
promp$ more point.data
0.0 0.0 0.0
0.1 0.1 0.1
0.2 0.2 0.2
0.3 0.3 0.3
0.4 0.4 0.4
0.5 0.5 0.5
0.6 0.6 0.6
0.7 0.7 0.7
0.8 0.8 0.8
0.9 0.9 0.9

4.3 Run time options

By default the code is running only for the hydrodynamic equations (1-2). The other equations or options could be activated in *cubby.data* just by naming this option :

4.3.1 Equations

magnetic Activation of induction equation (3-4)
scalar Activation of the passif scalar equation (5)
fluctuation the induction equation and the scalar equation are driven by only the fluctuation part of the velocity ($\mathbf{V}-\mathbf{V}_0$). The mean flow velocity come from external files (vmx_r, vmy_r, vmz_r).

kinematic Activation of kinematic run (V is constant or depend in time mathematicaly or by adding noise)
kinematic_noise Activation of adding noise into a steady velocity for kinematic computation.

4.3.2 Average in time fields

During the run time, some fields could be averaged. first add the option
time_average Necessary for the activation of all the time average options

Secondly add :
fluid_average :: vmx.dat,vmy.dat,vmz.dat,vms.dat
$vms =< \frac{1}{2}(\vec{v})^2 >$
potential_fluid_average :: psims.dat, wms.dat
psims = time average the potential vector energy ($< \frac{1}{2}(\vec{\psi})^2 >$ with $\vec{u} = \nabla \times \psi$)
wms = time average of the Enstrophy ($< \frac{1}{2}(\nabla \times \vec{v})^2 >$)
helicity_average :: hm.dat, hms.dat
$hms =< \frac{1}{2}(\vec{v}.\nabla \times \vec{v})^2 >$
magnetic_average :: bmx.dat,bmy.dat,bmz.dat,bms.dat
$bms =< \frac{1}{2}(\vec{b})^2 >$
electromotive_force_average :: emx.dat,emy.dat,emz.dat, ems.dat
$\vec{em} =< \vec{u} \times \vec{b} >$ and $ems =< \frac{1}{2}(\vec{u} \times \vec{b})^2 >$
This option record also the lorentz force time average :: jxbmx.dat,jxbmy.dat,jxbmz.dat, jxbms.dat
$\vec{jxbm} =< (\nabla \times b) \times b >$, jxbms $=< \frac{1}{2}((\nabla \times b) \times b)^2 >$
potential_magnetic_average :: ams.dat, jms.dat
$ams =< \frac{1}{2}(\vec{A})^2 >$ with $\vec{b} = \nabla \times \vec{A}$
$jms =< \frac{1}{2}(\nabla \times b)^2 >$
scalar_average sm.dat

Note when the option **fluctuation** is activated, the average in time of the velocity fluctuation will be computed $fluctm =< \vec{u} - \vec{uo} >$. :: fluctmx.dat, fluctmy, fluctmz.dat, fluctms.dat.

4.4 Parameters

The control parameter of the run are in cubby.data and point.data : The order of the parameter does matter and there is always a default value.

remark: *real* = *double* in general but can by setup to *float*

type	Variable	default value	comment
Physical Parameters:			
real	**dt or time_step**	$1.e-3$	time step
int	**Nts or nb_time_steps**	100	Number of time step
int	**dealias_v**	1	**dealias_v**=1 dealiasing of the v vector field (at 2N/3)
int	**dealias_b**	1	**dealias_b**=1 dealiasing of the b vector field (at 2N/3)
int	**dealias_s**	1	**dealias_s**=1 dealiasing of the s scalar field (at 2N/3)
int	**lorentz**	0	**lorentz**=1 Lorentz force actif
int	**coriolis_force**	0	**coriolis_force** = 1, 2, 3 rotation along X,Y, Z axis.
real	**rotation**	0.0	Coriolis Force strengh : $-\vec{r}otation \times v(x)$
int	**turb_visc**	0	turbulent viscosity model : hyperviscosity, LES (see chapter turbulent model)
int	**step_turb**	10	take in count the turbulent model **turb_visc** each **step_turb** time step.
int	**K**	1	LES cut off wave number (default $K = K_{max}$)
int	**bullard_on**	0	bullard dynamo (not yet available)
int	**bullard_step**	1	bullard dynamo step (not yet available)
Penalization Parameters:			
int	**penalization_v**	0	penalization for the velocity [1: obstacle sphere center in the box R="radius"] [2: cylinder boundary Radius=radius π, vertical boundaries = $2\pi a_r DH < z < 2\pi a_r(1-DH)$] [3: open cylinder along z axis, Radius="radius*π"] [4: bounded box ($2\pi DH < (x,y,z) < 2\pi(1-DH)$)] [5: sphere boundary with a Radius=radius π]
int	**penalization_f**	0	penalization for the forcing
int	**penalization_b**	0	penalization for the magnetic field
real	DH	$\frac{1}{16}$	radius in XY plane from the center line $X = \pi, Y = \pi$
real	radius	$1-\frac{1}{8}$	heigh parameter
real	DH_f	$\frac{1}{16}$	radius in XY plane from the center line $X = \pi, Y = \pi$ for the forcing
real	radius_f	$1-\frac{1}{8}$	heigh parameter for the forcing

Starting and parameters field :			
real	**nu**	1.0	fluid viscosity ν
int	**iv_type**	5	Velocity initialisation
int	**iv_option**	3	Velocity initialisation
real	**iv_coeff**	1.0	Velocity initialisation
real	**iv_ampl**	1.0	Velocity initialisation
int	**V0_on**	0	Mean velocity initial condition $V_0x = 1$,$V_0y = 2$,$V_0z = 3$
real	**V0_ampl**	1.0	Mean velocity field initial condition amplitude
real	**eta**	1.0	magnetic diffusivity η
int	**ib_type**	4	Magnetic initialisation
int	**ib_option**	0	Magnetic initialisation
real	**ib_coeff**	1.0	Magnetic initialisation
real	**ib_ampl**	1.0	Magnetic initialisation
int	**B0_on**	0	Mean Magnetic field initial condition $B_0x = 1$,$B_0y = 2$,$B_0z = 3$
real	**B0_ampl**	1.0	Mean Magnetic field initial condition amplitude
real	**kappa**	1.0	scalar diffusivity κ
int	**is_type**	1	Scalar initialisation
int	**is_option**	1	Scalar initialisation
real	**is_coeff**	0.1	Scalar initialisation
real	**is_ampl**	1.0	Scalar initialisation
int	**S0_on**	0	Mean scalar initial condition $S_0 = 1$
real	**S0_ampl**	1.0	Mean scalar field initial condition amplitude

Forcing :			
int	**forcing**	0	Forcing of the velocity
int	**option_force**	0	Forcing of the velocity
real	**coeff_force**	1.0	Forcing of the velocity
real	**force_amplitude**	1.0	Forcing of the velocity
int	**time_forcing**	0	Number of time step when the forcing is changing (only when the force depend of the time)
int	**control_forcing**	0	Control the amplitude of the forcing : $F_{rms} = 1.0$
int	**lap_forcing**	0	Take the laplacian of the forcing $F \rightarrow -\Delta F$
int	**is_type**	0	Forcing of the scalar
int	**is_option**	0	Forcing of the scalar
real	**is_coeff**	1.0	Forcing of the scalar
real	**is_ampl**	1.0	Forcing of the scalar

3D Field outputs :			
int	**step_out**	0	Number of field output in the spectral space (i.e :fvx.dat) all different time are in same file (see **Ext**) .
int	**step_begin_store**	-1	begin **step_out** after the **step_begin_store** time step
int	**step_out_b**	0	starting index for **step_out**
int	**Ext**	0	if **Ext**=1 the field file fvx.dat will split for different time fvx_00001.dat
int	**step_out_float**	0	Number of field output in *float* type in the real space (i.e vx.dat) all different time are in same file (see **Ext_float**).
int	**step_out_float_b**	0	starting index for **step_out_float**
int	**Ext_float**	0	if **Ext_float**=1 the field file vx.dat will split for different time 00001_vx.dat
int	**step_out_double**	0	Number of field output in *double* type in the real space (i.e vx.dat) all different time are in same file (see **Ext_double**).
int	**step_out_double_b**	0	starting index for **step_out_double**
int	**Ext_double**	0	if **Ext_double**=1 the field file vx.dat will split for different time 00001_vx.dat
int	**V_mean**	0	Number of time step to record the time average of the vector fields
int	**TA_option**	0	**TA_option**=1 rescaling for the time average of $B = B\sqrt{2 * \text{Energy}_B}$
int	**V_mean_out**	0	Number of time step for writing temporally the average in time fields (*not yet implemented).*
int	**v_storage**	0	Storing v in real space in type *real* at $t = 0$ & T_{max} only
int	**b_storage**	0	Storing b in real space in type *real* at $t = 0$ & T_{max} only
int	**s_storage**	0	Storing s in real space in type *real* at $t = 0$ & T_{max} only
int	**jump**	1	Reduction of the size by **jump** for **storage** (*not yet implemented)*

type	Variable	default value	comment
various outputs :			
int	**v_check**	0	check various quantities of the vector field(s)
int	**dv_stats**	0	create histogram for the last time of the run
int	**dv_max**	0	max for the histogram of the velocity
int	**db_max**	0	max for the histogram of the magnetic field
int	**e_spectrum**	0	Energy, Forcing transfert and Enstrophy spectra : **e_spectrum** times in the run (Create Hydro : ev.dat & MHD : eb.dat hq.dat (see chapter Unidimensional outputs for details))
int	**out_energy**	0	Energy, tranfert : **out_energy** times during the run (Create in Hydro : energ_v.dat, transf.dat & MHD : energ_b.dat)
int	**step_VPoint**	0	Number of probe outputs for vector fields (v,b) during the run (Probe positions are setup in the point.data file) (Creating in Hydro : pt_v.dat & MHD : pt_v.dat)
int	**step_SPoint**	0	Number of probe outputs for scalar field (s) during the run (Probe positions are setup in the point.data file) (Creating pt_s.dat)
int	**step_max**	0	Number of record of the maximum value of all vectors and scalars
int	**alpha_out**	0	record "alpha_out" times the electromotrice force and the magnetic field in the $(0, 0, Kz = 1/a_r)$ mode
int	**zwpm_max**	0	Number of record of the maximum value of the vector ZWpm Vorticity W+J and W-J
int	**zpm_on**	0	Number of record of Z+ Z-, shear pseudo and 2 dimensional spectral

type	Variable	default value	comment
adding noise for the velocity, magnetic, scalar initial condition (I.C.) and forcing velocity :			
int	**noise_on_v**	0	adding noise for velocity I.C.
int	**noise_on_b**	0	adding noise for magnetic I.C.
int	**noise_on_s**	0	adding noise for scalar I.C.
int	**noise_on_f**	0	adding noise for the forcing .
int	**noise_type**	2	type of noise
int	**noise_option**	2	option for the noise type
real	**noise_coeff**	2.0	coefficient for noise type
real	**real noise_ampl**	0.1	amplitude of the noise
Extra parameter :			
real	**A,B,C**	$(1.0, 1.0, 1.0)$	Parameter for the ABC forcing or field
real	**Xi**	1.0	Extra parameter for the G.O Robert flow
real	**Omega**	1.0	Extra parameter for the Taylor Green depending of the time
real	**Ampl_Omega**	1.0	Extra parameter for the Taylor Green depending of the time

5 Initialisation of scalar and vector fields or Forcing

Each scalar and vector field/forcing must be set up in **cubby.data**
Four parameters could be chosen :

int **ic_type** :	the type of initialisation	[**ic_type**]
int **option** :	integer option of initialisation	[**ic_option**]
real **coeff_option** :	real option coefficient of initialisation	[**ic_coeff**]
real **ampl**:	amplitude product of	[**ic_ampl**]

like :
ic_type : iv_type , ib_type, is_type , forcing , forcing_s
option : iv_option , ib_option , is_option, option_force, option_force_s
coeff_option : iv_option, ib_option, coeff_force , coeff_force_s
ampl : iv_ampl, ib_ampl, is_ampl, force_amplitude, force_amplitude_s

5.0.1 ic_type = 0 : restart

Read the files f[char_field][xyz]_r from the current directory in **spectral** .

ic_option = 0 : leave unchange the time and dissipation coefficient
ic_option = 1 : set up the current computing time from the readed file $time = t_{old}$
ic_option = 2 : the dissipation coefficient is set up from the readed file $\nu = \nu_{old}$
ic_option = 3 : option 1 and 2 together ($\nu = \nu_{old}$ and $time = t_{old}$)

Read the files [char_field][xyz]_r from the current directory in **real space** .

ic_option = -1 : set up the current computing time from the readed file $time = t_{old}$
ic_option = -2 : the dissipation coefficient is set up from the readed file $\nu = \nu_{old}$
ic_option = -3 : option 1 and 2 together ($\nu = \nu_{old}$ and $time = t_{old}$)

if (ic_coeff < -1) the readed field is amplified by a factor ic_ampl.

5.0.2 ic_type = 1 (default) : noise

option = 1 : white noise
option = 2 : gaussian noise
default: white noise

At large scale $kx, ky, kz = (0., 1., -1.)$

5.0.3 ic_type = 2 : Gaussian noise with fixed spectra

The field is set up as gaussian noise with a given 1D spectra :
ic_option = 1 : $E(k) = k^{-5/3}\ e^{-4\pi/N\ k}$.
ic_option = 2 : $E(k) = k^{-11/3}\ e^{-4\pi/N\ k}$.
ic_option = 3 : $E(k) = k^4\ e^{-2\ (k/\mathbf{ic\_coeff})^2}$ (ic_coeff is needed).
ic_option = 4 : $E(k) = 1$ ic_option = 5 : $E(k) = k^2\ e^{-2\ (k/\mathbf{ic\_coeff})^2}$ (ic_coeff is needed).
ic_option = 6 : $E(k) = k^{0.2}\ e^{-2\ (k/\mathbf{ic\_coeff})^2}$ (ic_coeff is needed).

default : ic_option = 1

5.0.4 ic_type = 3 : Robert flow

The field is set up to be the G.O Robert Flow :

$$v = ic\_ampl \begin{bmatrix} \sin(k_0\ xo)\cos(ko * yo) \\ -\cos(k_0\ x)\sin(k_0\ y) \\ \sqrt{(2)}\xi\ \sin(k_0\ x)\sin(k_0\ y) \end{bmatrix}$$

The Coefficients k_0=ic_coeff, **Xi** is a external parameter taken equal to $Xi = 1$ by default and can be change in cubby.data .

5.0.5 case 4 : ABC flow

The field is set up to be the Arnold Beltrami Childress (ABC) Flow :

$$v = ic\_ampl \begin{bmatrix} A\sin(k_0 z) + C\cos(k_0 y) \\ B\sin(k_0 x) + A\cos(k_0 z) \\ C\sin(k_0 y) + B\cos(k_0 x) \end{bmatrix}$$

The Coefficients A, B, C can be set up in the input parameter file :*cubby.data*, they are been taken equal to one by default ($A = B = C = 1$) and is given k_0=ic_coeff.

5.0.6 case 5 : Taylor Green flow

option 1 : along the x axis : $v = ic\_ampl \begin{bmatrix} 0 \\ (\sin(k_0 y)\cos(k_0 z)\cos(k_0 x) \\ -\cos(k_0 y)\sin(k_0 z)\cos(k_0 x) \end{bmatrix}$

option 2 : along the y axis : $v = ic\_ampl \begin{bmatrix} -\cos(k_0 z)\sin(k_0 x)\cos(k_0 y) \\ 0 \\ \sin(k_0 z)\cos(k_0 x)\cos(k_0 y) \end{bmatrix}$

option 3 : along the z axis : $v = ic\_ampl \begin{bmatrix} \sin(k_0 x)\cos(k_0 y)\cos(k_0 z) \\ -\cos(k_0 x)\sin(k_0 y)\cos(k_0 z) \\ 0 \end{bmatrix}$

with k_0=ic_coeff and option 3 is the default (Nore et al. , Ponty et al PRL 2004) .

5.0.7 case 51 : Taylor Green flow time dependence

option 1 : along the x axis : $v = ic\_ampl \begin{bmatrix} 0 \\ (\sin(k_0 y)\cos(k_0 z)\cos(k_0 x) + Ampl\cos(\Omega * t) \\ -\cos(k_0 y)\sin(k_0 z)\cos(k_0 x) + Ampl\cos(\Omega * t) \end{bmatrix}$

option 2 : along the y axis : $v = ic\_ampl \begin{bmatrix} -\cos(k_0 z)\sin(k_0 x)\cos(k_0 y) + Ampl\cos(\Omega * t) \\ 0 \\ \sin(k_0 z)\cos(k_0 x)\cos(k_0 y) + Ampl\cos(\Omega * t) \end{bmatrix}$

option 3 : along the z axis : $v = ic\_ampl \begin{bmatrix} \sin(k_0 x)\cos(k_0 y)\cos(k_0 z) + Ampl\cos(\Omega * t) \\ -\cos(k_0 x)\sin(k_0 y)\cos(k_0 z) + Ampl\cos(\Omega * t) \\ 0 \end{bmatrix}$

with k_0=ic_coeff, $Ampl$ = Ampl_Omega and Ω= Omega are extra parameters.

5.0.8 case 6 : Orszag Tang velocity

The flow is defined : $v = ic\_ampl\ [-2\sin(k_0 y), 2\sin(k_0 x), 0]$ with k_0=ic_coeff. with k_0=ic_coeff.

5.0.9 case 7 : Orszag Tang magnetic

$v = ic\_ampl\sqrt{2/3}\begin{bmatrix} -2\sin(2k_0y) + \sin(k_0z) \\ 2\sin(k_0x) + \sin(k_0z) \\ sin(k_0x) + sin(k_0y) \end{bmatrix}$ with k_0=ic_coeff.

5.0.10 case 8 : X point velocity

$v = ic\_ampl[-cos(k_0y), cos(k_0x), ic\_coeff]$ and k_0=ic_option .

5.0.11 case 9 : 1 flux tube

$v = tube(x_1, \phi, R, q) = ic\_ampl\ \Pi(1 - r/R)\ \frac{1+\cos(\pi r/R)}{1+q^2r^2}\begin{bmatrix} q\tilde{y} \\ -q(x - x1) \\ -1 \end{bmatrix}$
with $\tilde{y} = \cos(\phi)(y - \pi) - sin(\phi)(z - \pi)$ and $r^2 = (x - x1)^2 + \tilde{y}^2$

5.0.12 case 10 : Linton velocity

$v = ic\_ampl\begin{bmatrix} -\sin(x)(\cos(y) + \cos(z)) \\ \cos(x)\sin(y) \\ \cos(x)\sin(z) \end{bmatrix}$

5.0.13 case 10 : Tubes Linton

$v = tubes\_Linton(\sigma, \beta, Pt1 = (ampl_1, x_1, y_1, z_1), Pt2 = (ampl2, x_2, y_2, z_2))$
$= ampl_1\frac{e^{-((x-x_1)^2+(y-y_1)^2)/2\sigma^2}}{\sigma\sqrt{2\pi}}\frac{1}{1+\beta^2((x-x_1)^2+(y-y_1)^2)}\begin{bmatrix} -\beta(y - y_1) \\ \beta(x - x_1) \\ 1 \end{bmatrix} +$
$ampl_2\frac{e^{-((x-x_2)^2+(z-z_2)^2)/2\sigma^2}}{\sigma\sqrt{2\pi}}\frac{1}{1+\beta^2((x-x_2)^2+(z-z_2)^2)}\begin{bmatrix} \beta(z - z_2) \\ 1 \\ -\beta(x - x_2) \end{bmatrix}$
with $\sigma = 11\pi/48, \beta = ic\_coeff$
$x_1 = 2\pi/4, y_1 = \pi, z_1 = 0, x_2 = 7\pi/4, y_2 = 0, z_2 = \pi$

5.0.14 case 101 : Tubes Linton

Linton tubes with
$\sigma = 22\pi/48, \beta = ic\_coeff$
$x_1 = 2\pi/4, y_1 = \pi, z_1 = 0, x_2 = 7\pi/4, y_2 = 0, z_2 = \pi$

5.0.15 case 12 : Ponomarenko velocity

5.0.16 case 13 : Ponomarenko velocity +filter

5.0.17 case 14 : Taylor Green modified flow

$v = ic\_ampl\begin{bmatrix} \sin(k_1x + \phi)\cos(k_1y + \phi)(A_1 + A_2\cos(k_2z + \phi)) \\ -\cos(k_1x + \phi)\sin(k_1y + \phi)(A_1 + A_2\cos(k_2z + \phi)) \\ 0 \end{bmatrix}$
with k_1 = ic_option,$k_2 = C$, $A_1 = A$, $A_2 = B$ & ϕ =Omega.

5.0.18 case 15 : Sine flow

$$v = ic\_ampl \begin{bmatrix} \sin(ko\ z) \\ \sin(ko\ x) \\ \sin(ko\ y) \end{bmatrix} \text{ with } k_0\text{=ic\_coeff.}$$

5.0.19 case 16 : Sine flow depending of the time

$$v = ic\_ampl \begin{bmatrix} \sin(ko\ z + \epsilon\ cos(\Omega\ time)) \\ \sin(ko\ x + \epsilon\ cos(\Omega\ time + 2\pi/3)) \\ \sin(ko\ y + \epsilon\ cos(\Omega\ time + 4\pi/3)) \end{bmatrix} \text{ with } k_0\text{=ic\_coeff, } \epsilon\text{=Ampl\_Omega and } \Omega\text{=Omega.}$$

5.0.20 case 20 : Kida velocity

$$v = ic\_ampl \begin{bmatrix} \sin(x)(cos(3y)\cos(z) - \cos(y)\cos(3z)) \\ \sin(y)(cos(3z)\cos(x) - \cos(z)\cos(3x)) \\ \sin(z)(cos(3x)\cos(y) - \cos(x)\cos(3y)) \end{bmatrix}$$

5.0.21 case 21 : Kida magnetic

$$v = ic\_ampl \begin{bmatrix} \sin(4x)(\cos(2y) + \cos(2z)) - 2\sin(2x)(\cos(4y) + \cos(4z)) \\ \sin(4y)(\cos(2z) + \cos(2x)) - 2\sin(2y)(\cos(4z) + \cos(4x)) \\ \sin(4z)(\cos(2x) + \cos(2y)) - 2\sin(2z)(\cos(4x) + \cos(4y)) \end{bmatrix}$$

5.0.22 case 45 : ABC+TG

5.0.23 case 50 : VKE velocity

5.0.24 adding noise

For adding noise to the I.C. or the velocity forcing just switch on the value :
noise_on_v for the velocity
noise_on_b for the magnetic field
noise_on_s for the scalar
noise_on_f for the velocity
and choose the type of noise as 5.0.3 subsection with
noise_type (2)
noise_option (2)
noise_option (2.0)
noise_ampl (0.1)
(values as default).

5.0.25 Table

ic_type	Type	ic_option	Comment	ic_coeff
0	restart	1	$time = t_{old}$	$-2 \rightarrow v = v * ampl$
0	restart	2	$\nu = \nu_{old}$	$-2 \rightarrow v = v * ampl$
0	restart	3	$time = t_{old}$ and $\nu = \nu_{old}$	$-2 \rightarrow v = v * ampl$
1	noise	1	white noise at large scale $kx, ky, kz = (0., 1., -1.)$	
1	noise	2	gaussian noise at large scale $kx, ky, kz = (0., 1., -1.)$	
2	Gaussian noise with spectra	1	$E(k) = k^{-5/3}\ e^{-4\pi/N\ k}$	
2	Gaussian noise with spectra	2	$E(k) = k^{-11/3}\ e^{-4\pi/N\ k}$	
2	Gaussian noise with spectra	3	$E(k) = k^4\ e^{-2\ (k/ic\_coeff)^2}$	needed
4	$ABC(k_0)$		$(A = B = C = 1)$ set by default	k_0
5	Taylor Green (k_0)	1	along x axis	k_0
5	Taylor Green (k_0)	2	along y axis	k_0
5	Taylor Green (k_0)	3	along z axis	k_0
51	Taylor Green (k_0)	1	along x axis time dependence	k_0
51	Taylor Green (k_0)	2	along y axis time dependence	k_0
51	Taylor Green (k_0)	3	along z axis time dependence	k_0
6	Orszag Tang velocity			k_0
7	Orszag Tang magnetic			k_0
8	Xpoint velocity	k_0	Yan Flow	vz
9	1 tube		$tube(x1, \phi, R, q)$ $x1 = PI, q = 10, R = PI/8$	ϕ
10	tubes Linton		$sigma = 11\pi/48$	beta
101	tubes Linton		$sigma = 22\pi/48$	beta
11	Linton velocity			
14	Taylor Green modified		$k_2 = C$, $A_1 = A$, $A_2 = B$, ϕ =Omega	k_1
15	Archontis flow		$(sin(k_0\, z), sin(k_0\, x), sin(k_0\, y))$	k_0
20	Kida velocity			
20	Kida velocity + noise	2	noise=$5.010^{-2} k^4\ e^{-2\ (k/coeff\_option)^2}$	needed
21	Kida magnetic			
21	Kida magnetic + noise	2	noise=$5.010^{-2} k^4\ e^{-2\ (k/coeff\_option)^2}$	needed
45	TG+ABC			
50	VKE			

Commented example of **cubby.data** file :

magnetic		mhd equation
time_average		allow the time average facilities
fluid_average		fluid average on
magnetic_average		fluid average on
helicity_average		helicity average on
dt	1.e-3	time_step
Nts	10000	number of time step
step_begin_store	2	start the storage of spectral 3D files after the step 2
step_out	1	only 1 storage of spectral space 3D files (at the end for restart)
step_out_float	10	11 storages of real space of the vector field (v and b)
iv_type	0	restart v with fvx_r fvy_r fvz_r spectral files
iv_option	3	take the viscosity and the time of the restart files
iv_coeff	1.0	
iv_ampl	1.0	
ib_type	4	ABC initialisation
ib_option	1	
ib_coeff	1.0	
ib_ampl	1.0e-10	amplitude of the magnetic field
nu	0.01	viscosity
eta	0.01	magnetic diffusivity
forcing	5	Taylor-green forcing
option_force	3	along the z direction
coeff_force	1.0	
force_amplitude	3.0	amplitude of the forcing
e_spectrum	1000	number of 1D isotrope spectrum of v and b (ev.dat, eb.dat)
out_energy	1000	number of total energy for v and b
v_check	1	check a t=begin and end, sereval statistics on v and b, print in the standat output
v_storage	1	store v in real space at t==begin and end
b_storage	1	store b in real space at t==begin and end
dv_stats	1	compute histogram at t=end
step_VPoint	1000	record the v danb for the probe provided by point.data
step_max	1000	number of the computation of the max of v, w and b,j
V_mean	10	allow the average in time and give the accumulation frequency of the average in time field, here each “10”time step
zpm_on	1000	allow the Z+ Z-, shear pseudo spectras recording “1000”times inside the run

6 Post-process utilities