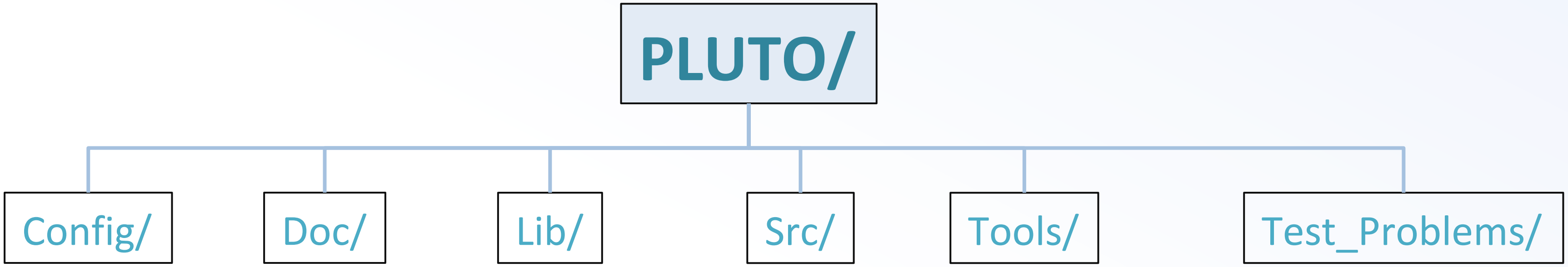
A Practical Introduction to the PLUTO Code

B. Vaidya

A. Mignone

Directory Structure



- Config/: contains machine architecture dependent files, such as information about C compiler, flags, library paths and so on. Useful for creating the *makefile*;
- Doc/: documentation directory;
- Lib/: repository for additional libraries;
- Src/: main repository for all **.c* source files with the exception of the *init.c* file, which is left to the user;
- Tools/: Collection of useful tools, such as Python scripts, IDL visualization routines, pyPLUTO, etc...;
- Test_Problems/: a directory containing several test-problems used for code verification.

EXAMPLE #1:
THE SHOCK-TUBE PROBLEM

Preparing to Run PLUTO

- PLUTO should be compiled and executed in a separate working directory which may be anywhere on your local hard drive
- To this end, we first need to set the environment variable PLUTO_DIR to point to this directory. In a bash shell,

```
> export PLUTO_DIR=/fullpath/PLUTO  # set it also in your .bashrc or similar
```

- Change directory to Test_Problems/Shock_Tube:

```
> cd Test_Problems/Shock_Tube
```

- In order to configure PLUTO, 4 basic steps are needed:

1. Creating the problem header file (*definitions.h*);
2. Choosing the *makefile*;
3. Tuning the runtime initialization file *pluto.ini*;
4. Coding initial & boundary conditions (*init.c*);

Through the Python script

Manually

The Python Menu (Step #1 & #2)

- Run the *python* script:

```
> python $PLUTO_DIR/setup.py
```

- The script will now enter into the main menu:

```
>> Python setup (May 2018) <<

Working dir: /Users/mignone/tmp/PLUTO/Test_Problems/Shock_Tube
PLUTO dir  : /Users/mignone/tmp/PLUTO/

 Setup problem
 Change makefile
 Auto-update
 Save Setup
 Quit
```

- Press enter under "*Setup problem*" →

```
>> Setup problem <<


 PHYSICS                          HD
 DIMENSIONS                       1
 COMPONENTS                       1
 GEOMETRY                         CARTESIAN
 BODY_FORCE                       NO
 FORCED_TURB                      NO
 COOLING                          NO
 RECONSTRUCTION                   LINEAR
 TIME_STEPPING                    HANCOCK
 DIMENSIONAL_SPLITTING            YES
 NTRACER                          0
 USER_DEF_PARAMETERS              0
```

The “Setup problem” menu

- In a self-explanatory way, the setup menu allows the user to configure the basic features for your problem:

```
>> Setup problem <<


  PHYSICS                       HD
  DIMENSIONS                    1
  COMPONENTS                    1
  GEOMETRY                      CARTESIAN
  BODY_FORCE                    NO
  FORCED_TURB                   NO
  COOLING                       NO
  RECONSTRUCTION                LINEAR
  TIME_STEPPING                 HANCOCK
  DIMENSIONAL_SPLITTING         YES
  NTRACER                       0
  USER_DEF_PARAMETERS           0
```

Physics Module, tells PLUTO the equations we want to solve.

Sets the dimensions and vector components

Select the geometry

Select reconstruction and time stepping methods

Number of passive scalars (colors)

Number of user-defined, runtime parameters

- By pressing enter, you’ll be directed to a second menu →

The “Physics” Sub-menu

- In the following menu, only directive relative to the HD module will be shown

```
>> HD Menu <<


 EOS                                  IDEAL
 ENTROPY_SWITCH                       NO
 THERMAL_CONDUCTION                   NO
 VISCOSITY                            NO
 ROTATING_FRAME                       NO
```

Selects the Equation of State (EOS) ant to solve.

Allows to solve the entropy equation rather than the total energy equation

Enable/disable thermal conduction in the energy equation.

Enable/disable viscous stress tensor (Navier-Stokes)

Enable/disable rotating frame of reference

- For the problem at hand, we neglect dissipative effects.

Makefile Menu

- Once we’re back to the main menu, we can select a different makefile (→ “*Change makefile*”)

```
>> Change makefile <<


 Aurora.gcc.defs
 Aurora.mpicc.defs
 CYGWIN_NT-6.1.x86_64.gcc.HDF5.defs
 Darwin.gcc.defs
 Darwin.mpicc.defs
 FERMI.mpixlc.defs
 Linux.gcc.defs
 Linux.mpicc.defs
 MARCONI.mpiicc.defs
 Template.defs
 debug.defs
 phw184mc.gcc.defs
 phw184mc.mpicc.defs
 preprocessed.defs
 profile.defs
```

- Configuration files are taken from the Config/ directory.
- If you have no idea, simply go with *Linux.gcc.defs* (for a serial run) or *Linux.mpicc.defs* (for a parallel run).

Specifying Initial Conditions (step #3):

- Initial condition are coded inside *init.c* file using the `Init()` function;

```
/* ******************************************************************** */
void Init (double *v, double x1, double x2, double x3)
/*
 *
 ********************************************************************** */
{
  #if EOS == IDEAL
   g_gamma = 1.4;
  #endif

  if (fabs(x1) < 0.5) {
    v[RHO] = 1.0;
    v[PRS] = 1.0;
  }else{
    v[RHO] = 0.125;
    v[PRS] = 0.1;
  }
  v[VX1] = 0.0;

}
```

- This file is always part of your local working directory.

Runtime Parameters (step #4): pluto.ini

- At runtime, PLUTO reads the pluto.ini file which is used to control several options use by the code at runtime, such as grid generation, CFL number, boundary conditions, output type and so forth.

- The file contains several blocks, of the form

```
[Block]

label       ...  fields  ...
label       ...  fields  ...
label       ...  fields  ...
```

- This file can be edited manually.

```
[Grid]

X1-grid    1    0.0   1600    u    1.0
X2-grid    1    0.0      4    u    1.0
X3-grid    1    0.0      1    u    1.0

[Time]

CFL                    0.9
CFL_max_var    1.1
tstop          0.2
first_dt       1.e-6

[Solver]

Solver       hllc

[Boundary]

X1-beg     outflow
X1-end     outflow
X2-beg     periodic
X2-end     periodic
X3-beg     outflow
X3-end     outflow

[Uniform Grid Output]

uservar    0
dbl        1.0   -1   single_file
flt        0.02   -1   single_file
vtk       -1.0   -1   single_file
tab        1.0   -1
ppm       -1.0   -1
png       -1.0   -1
log        10
analysis  -1.0   -1

[Parameters]
```

Compiling and Running

- PLUTO can now be compiled by typing “make” at the command prompt:

```
> make
```

- You can now run the code by typing:
  - Local machine, serial run:

```
> ./pluto
```

  - Local machine, parallel run:

```
> mpirun -np <n> ./pluto
```

Output log

```
...
> Assigning initial conditions (Startup) ...
> Normalization Units:

  [Density]:      1.673e-24 (gr/cm^3), 1.000e+00 (1/cm^3)
  [Pressure]:     1.673e-14 (dyne/cm^2)
  [Velocity]:     1.000e+05 (cm/s)
  [Length]:       1.496e+13 (cm)
  [Temperature]:  1.203e+02 X (p/rho*mu) (K)
  [Time]:         1.496e+08 (sec), 4.744e+00 (yrs)

> Number of processors: 1
> Proc size:            1600
> Writing file #0 (dbl) to disk...
> Writing file #0 (flt) to disk...
> Starting computation...

step:0; t = 0.0000e+00; dt = 1.0000e-06; 0.0 %
        [Mach = 0.000000]
step:10; t = 1.5937e-05; dt = 2.5937e-06; 0.0 %
         [Mach = 0.043457]
step:20; t = 5.7275e-05; dt = 6.7275e-06; 0.0 %
         [Mach = 0.137542]

...

step:840; t = 1.9894e-01; dt = 2.5050e-04; 99.5 %
          [Mach = 0.930084]
> Writing file #1 (dbl) to disk...
> Writing file #10 (flt) to disk...

> Total allocated memory    2.84 Mb
> Elapsed time              0d:0h:0m:0s
> Average time/step         0.00e+00  (sec)
> Local time                Tue Dec  4 12:52:18 2018
> Done
```

Physical units

Current time

Current time step

Maximum Mach number at each step

Everything's fine !

Visualizing Data

- Different visualization packages may be used to display PLUTO data-files. Some popular packages are reported below:

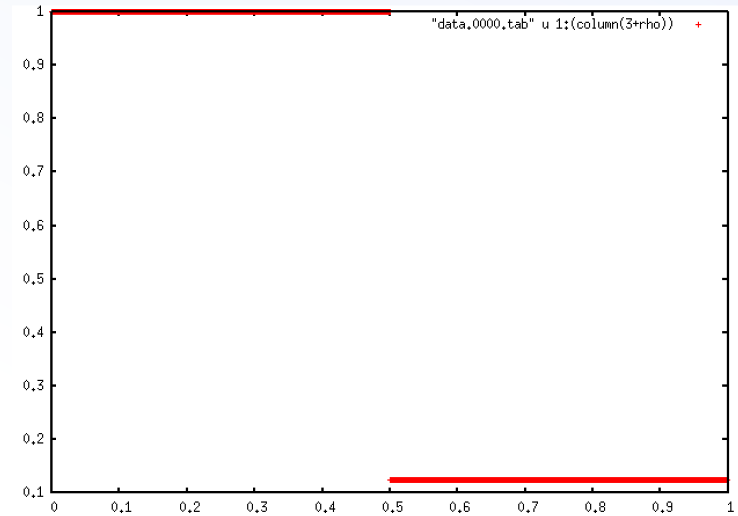
	Gnuplot	**IDL**	**Matlab**	**Python**	**VisIt**	**Paraview**
Free/Open Source	Yes	No	No	Yes	Yes	Yes
File support	ASCII (.tab)	All	All	All	VTK, HDF5	VTK, HDF5
Data Analysis	No	Yes	Yes	Yes	No	No

- If you're not familiar with any of these packages, you can use gnuplot since it's practically available by default on any Linux platform.

Visualization with Gnuplot

- ASCII datafile (`.tab`) can be easily visualized with GNUPLOT, using the plot command, e.g.

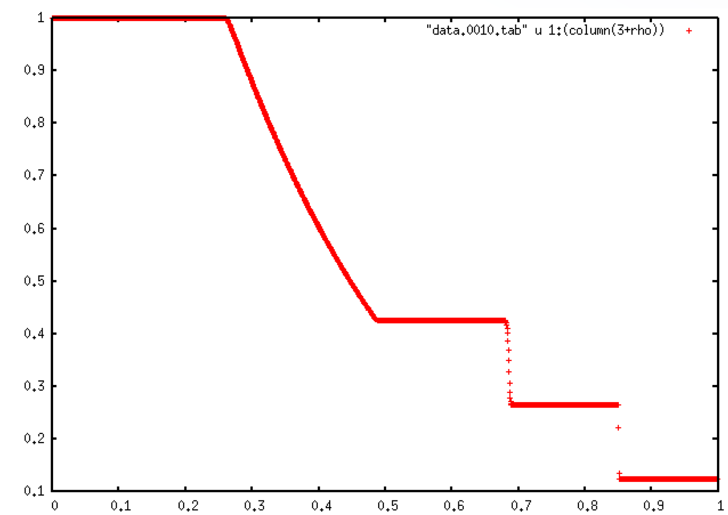
```
gnuplot> plot “data.0000.tab” using 1:3
```



- This will produce a 1D plot (x,y) of the initial condition (‘`0000`’).
- The keyword ‘`using`’ specify that the x-coordinate is taken from the 1st column and the y-coordinate from the 3rd column (= density);

- To plot the solution at the 10th output, type

```
gnuplot> plot “data.0010.tab” using 1:3
```



- Other variables may be plotted as well (4 = velocity, 5 = gas pressure);

Visualization with Gnuplot

- Variable names (together with grid information) are written by PLUTO to a special file (`pluto.gp`) at runtime*;

- You can load them as follows:

```
gnuplot> load “pluto.gp”
gnuplot> plot "data.0005.tab" u 1:(column(3+vx1))
```

- For the shock tube, variable names are: `rho` (density), `vx1` (x-velocity) and `prs` (gas pressure).

- A simple animation script can also be used to produce an animation,

```
gnuplot> load “shock_tube.gp”
```

*You must enable the GNUPLOT macro inside your definitions.h, e.g.

```
#define GNUPLOT      TRUE
```

EXAMPLE #2:
2D MHD BLAST WAVE PROBLEM

Setting up the Problem

- Change directory to Test_Problems/MHD_Blast and run the Python script

```
> python $PLUTO_DIR/setup.py
```

- The main setup menu should look like

```
>> Setup problem <<

 PHYSICS                          MHD
 DIMENSIONS                       2
 COMPONENTS                       3
 GEOMETRY                         CARTESIAN
 BODY_FORCE                       NO
 FORCED_TURB                      NO
 COOLING                          NO
 RECONSTRUCTION                   LINEAR
 TIME_STEPPING                    HANCOCK
 DIMENSIONAL_SPLITTING            NO
 NTRACER                          0
 USER_DEF_PARAMETERS              2
```

We are now using the *MHD module*, in *2 dimensions* (3 components) and have *2 user-defined parameters*.

Setting the Parameters

- We can use the constrained transport method to control the divergence of B, and define the name of 2 parameters that will be used later in our problem definition.

- Just scroll down with your arrow keys and check that the your userdef parameter names are already set to

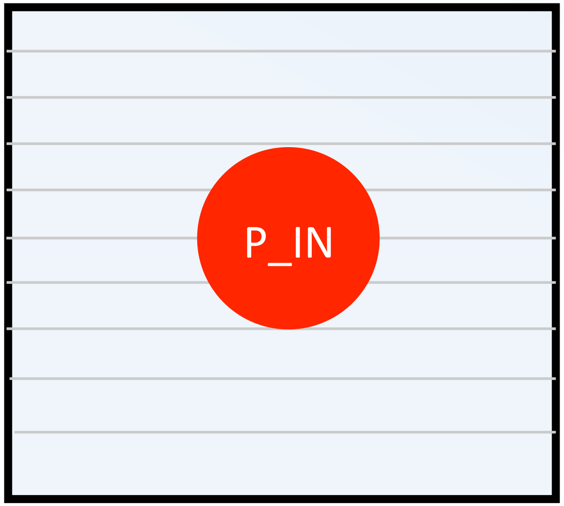
```
>> User-defined Parameters <<


 0                              P_IN
 1                              BETA
```

The value of the over-pressurized region

The plasma beta parameter

Setting Initial & Boundary Conditions

- We consider a 2D Cartesian box, filled with uniform density fluid and horizontal magnetic field. An over-pressurized region is set inside a circle of radius r_0:



- For simplicity we choose a zero-gradient boundary conditions in all directions.
- The over-pressurized region drives a blast wave delimited by an outer fast forward shock propagating (nearly) radially while magnetic field lines pile up behind the shock thus building a region of higher magnetic pressure.

Specifying Initial Conditions:

- Initial conditions are coded inside `init.c` file using the `Init()` function;
- This file is always in your local working directory.
- Parameters are included using the global array `g_inputParam[<name>]`.
- The value of these parameters is read at runtime from pluto.ini

```
void Init (double *v, double x1, double x2, double x3)
{
  double r, theta, phi, B0;

  r = D_EXPAND(x1*x1, + x2*x2, + x3*x3);
  r = sqrt(r);

  v[RHO] = 1.0;
  v[VX1] = 0.0;
  v[VX2] = 0.0;
  v[VX3] = 0.0;
  v[PRS] = 1.0/g_gamma;
  if (r <= 0.1) v[PRS] = g_inputParam[P_IN];

  B0     = sqrt(2.0/g_gamma/g_inputParam[BETA]);

  v[BX1] = B0;
  v[BX2] = 0.0;
  v[BX3] = 0.0;

  v[AX1] = 0.0;
  v[AX2] =  v[BX3]*x1;
  v[AX3] = -v[BX2]*x1 + v[BX1]*x2;

}
```

Runtime Parameters: pluto.ini

- At runtime, PLUTO reads the pluto.ini file which controls several options use by the code at runtime, such as grid generation, CFL number, boundary conditions, output type and so forth.

- Make sure you're writing using your preferred data format.

- Problem-parameters are set at the end.

```
[Grid]

X1-grid    1    -0.5    192    u    0.5
X2-grid    1    -0.5    192    u    0.5
X3-grid    1    -0.5      1    u    0.5

[Time]

CFL              0.4
CFL_max_var      1.1
tstop            5.0e-2
first_dt         1.e-6

[Solver]

Solver         hll

[Boundary]

X1-beg         outflow
X1-end         outflow
X2-beg         outflow
X2-end         outflow
...

[Static Grid Output]

...
dbl        -2.5e-3  -1    single_file
flt         2.5e-3  -1    single_file
...

[Parameters]

P_IN                        1.e2
BETA                        0.1
```

Compiling and Running

- Compile by typing “make” at the terminal

- Run by typing either

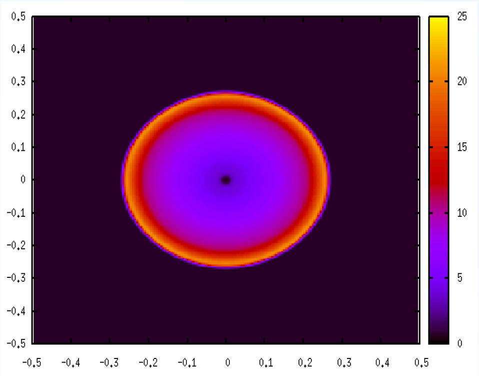
```
> ./pluto    # Serial run using (using e.g., Linux.gcc.defs)
```

or

```
> mpirun -np <n> ./pluto    # Parallel run (using e.g., Linux.mpicc.defs)
```

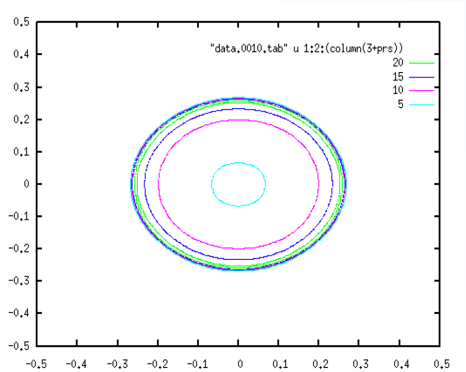
- Depending on your processor speed, the code may take few minutes to run….

Visualization with Gnuplot



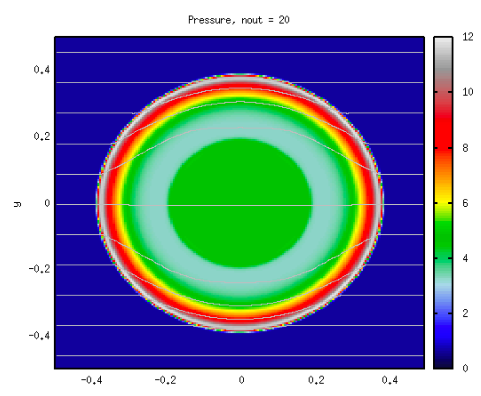
- 2D visualization can also be done with gnuplot using the `splot` command.
- To produce a coloured map:

```
gnuplot> load “pluto.gp”
gnuplot> set pm3d map
gnuplot> splot "data.0010.tab" u 1:2:(column(3+prs))
```



- To draw contour levels,

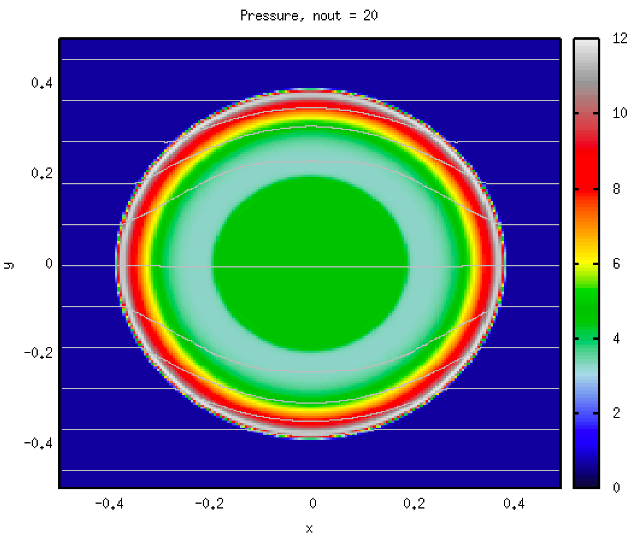
```
gnuplot> load “pluto.gp”
gnuplot> set contour base
gnuplot> set view map
gnuplot> unset surface             # Disable surface plot
gnuplot> set style data lines      # Use lines instead of points
gnuplot> splot "data.0010.tab" u 1:2:(column(3+prs))
```

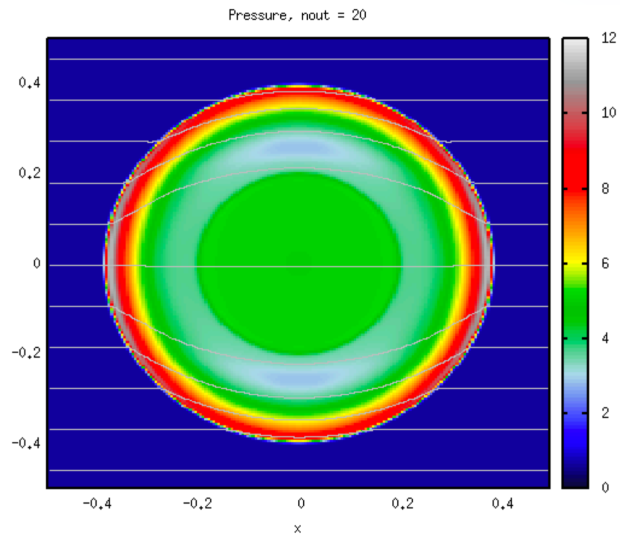


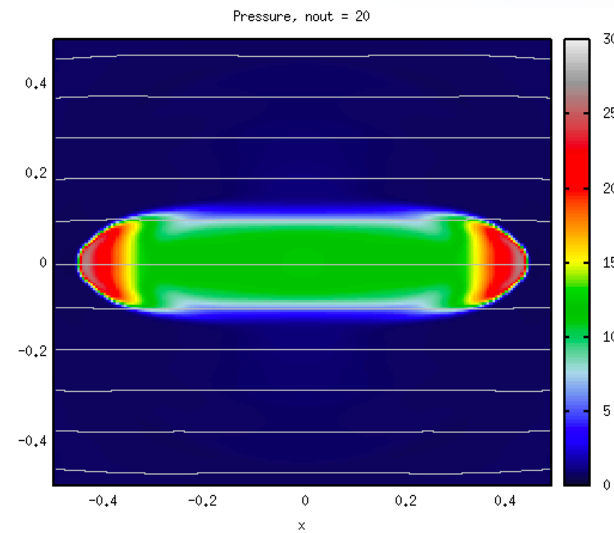
- The script “`mhd_blast.gp`” can be used to produce an animation with field lines.

Parameter Study

- Try different computations by decreasing the plasma β parameter (100, 1, 0.01). What happens ? Explain.





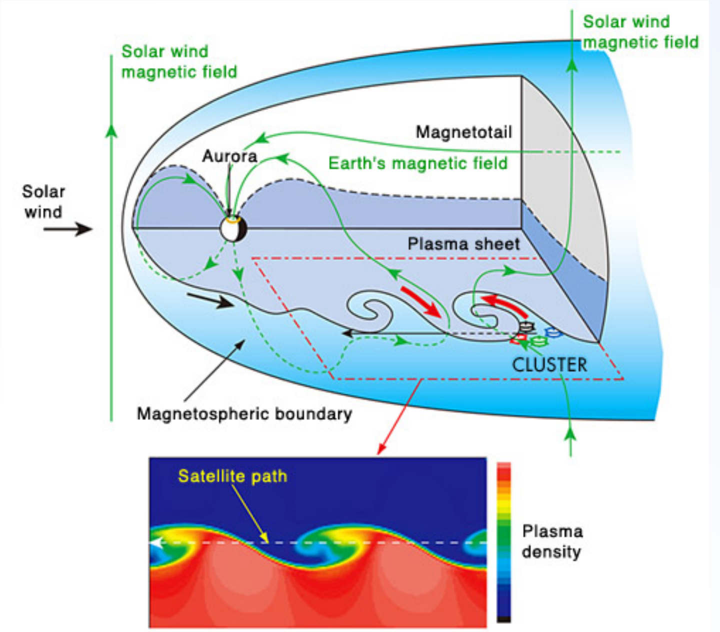


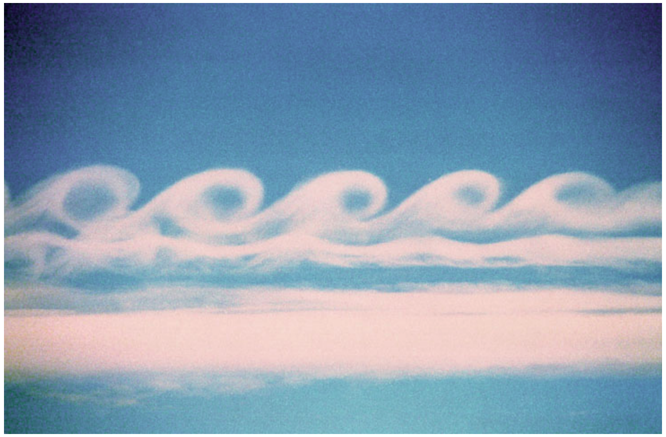
EXAMPLE #3:
2D THE KELVIN-HELMHOLTZ INSTABILITY

Kelvin-Helmholtz Instability

- The Kelvin–Helmholtz Instability (KHI) develops at the interface between two fluids in relative motion;

- Important in atmospheric flows, interaction between solar wind and magnetosphere (space weather), supersonic jet propagation (astrophysics), etc…





Equations

- The instability may be analyzed using the compressible Euler equations,

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) & = & 0 \\ \rho \left(\dfrac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} \right) + \nabla p & = & 0 \\ \dfrac{\partial p}{\partial t} + \boldsymbol{u} \cdot \nabla p + \gamma p \nabla \cdot \boldsymbol{u} & = & 0 \end{cases}$$

- In the vortex-sheet approximation, the velocity at equilibrium has a jump in the transverse direction

$$\mathbf{v} = \begin{cases} +M/2\hat{\mathbf{e}}_x & \text{for} \quad y > 0 \\ -M/2\hat{\mathbf{e}}_x & \text{for} \quad y < 0 \end{cases}$$

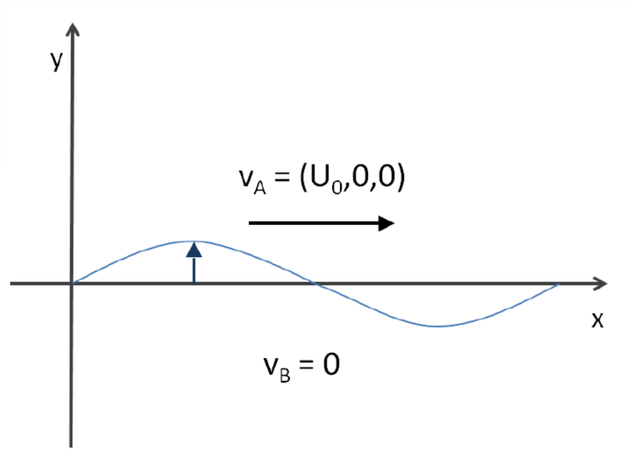
- Equilibrium density and pressure are constant,

Linear Theory

- A linearization of the equations can be carried out for small perturbations,

$$q(\mathbf{x},t) = q_0 + q'(\mathbf{x},t)$$

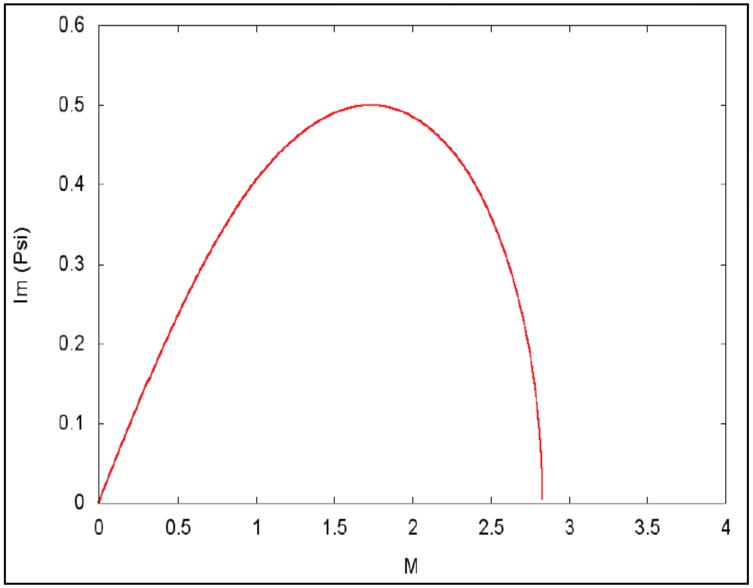
- where $q' = f(y)e^{i(kx - \omega t)}$ is a complex quantity.



- The linearization process leads to the following dispersion relation

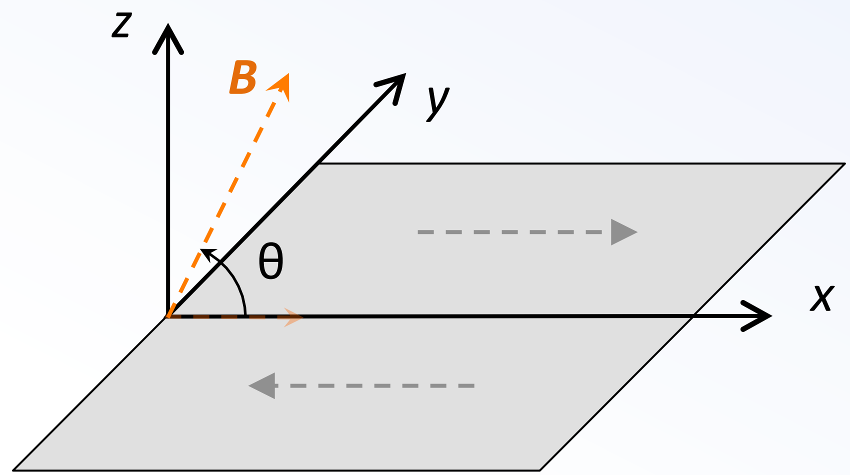
$$\frac{\omega}{kc_s} = \frac{M}{2} \pm i\left[\sqrt{M^2+1} - \left(\frac{M^2}{4}+1\right)\right]^{1/2}$$

- A complex value of ω(k) indicates an instability.

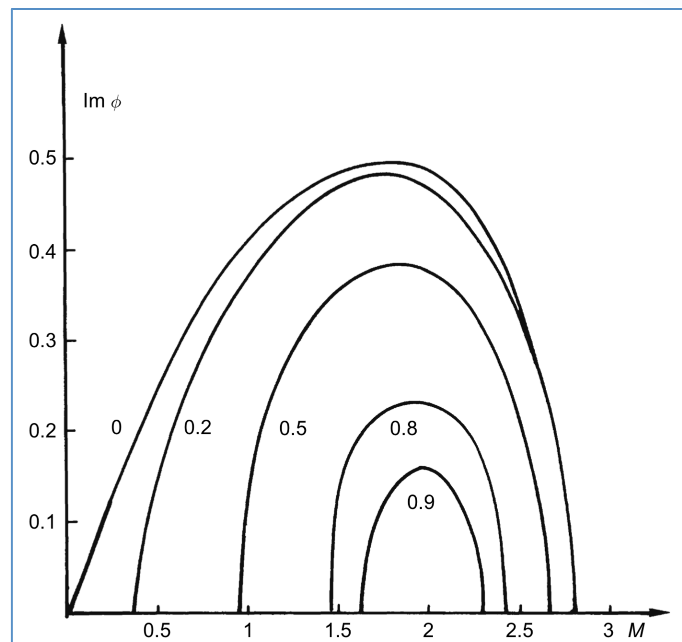


Linear Theory: MHD

- When a magnetic field is included, the linearization leads to a 10^{th} degree polynomial;

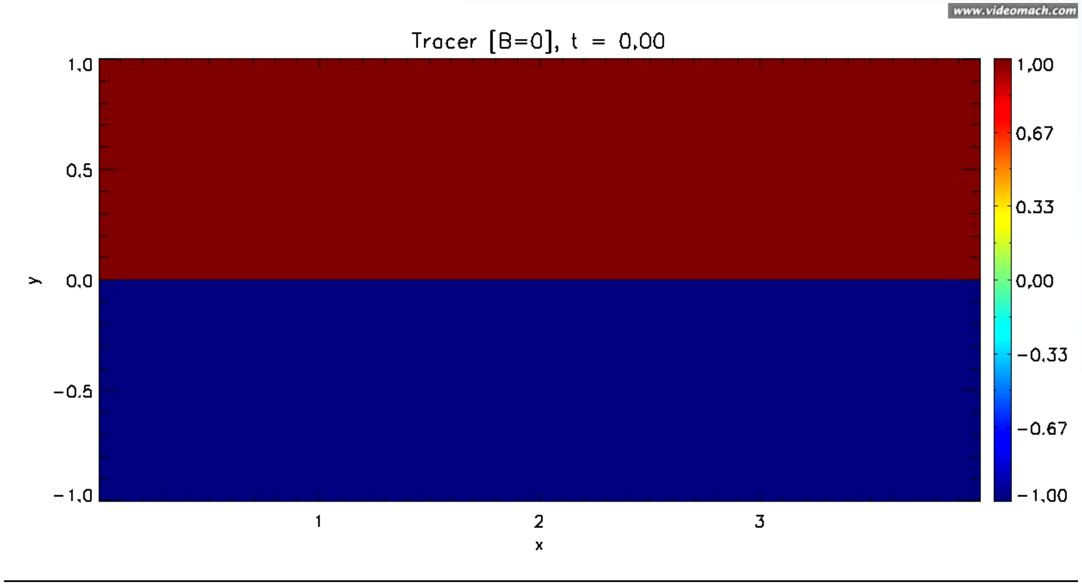


- The effect of B on the development of the KHI strictly depends on its geometry:
  - θ = π/2: when **v** and **B** are perpendicular the features of the instability are basically unmodified if we redefine the Mach number as $M = v/\mathrm{sqrt}(vA^2 + cs^2)$;
  - θ = 0: when **v** and **B** are perpendicular, the magnetic field has a stabilizing effect: for M < 2vA/cs the flow is stable, while the supersonic cut off decreases from M = sqrt(8) → 2 and for vA/cs > 1 the two limits coincide: highly magnetized flows are always stable against the KHI.



KH: Nonlinear evolution

- The growth of instability leads to a deformation of the interface creating a typical roll-up structure with vortex formation:



- Kinetic (ordered) energy is dissipated into disordered energy and the growth of perturbation at the interface leads to the generation of interacting vortices.

- Vortex merging leads to larger velocity shear and mixing of fluids from the two different regions.

Setting up the Problem

- Change directory to Test_Problems/Kelvin_Helmholtz and run the Python script

```
> python $PLUTO_DIR/setup.py
```

- The main setup menu should look like

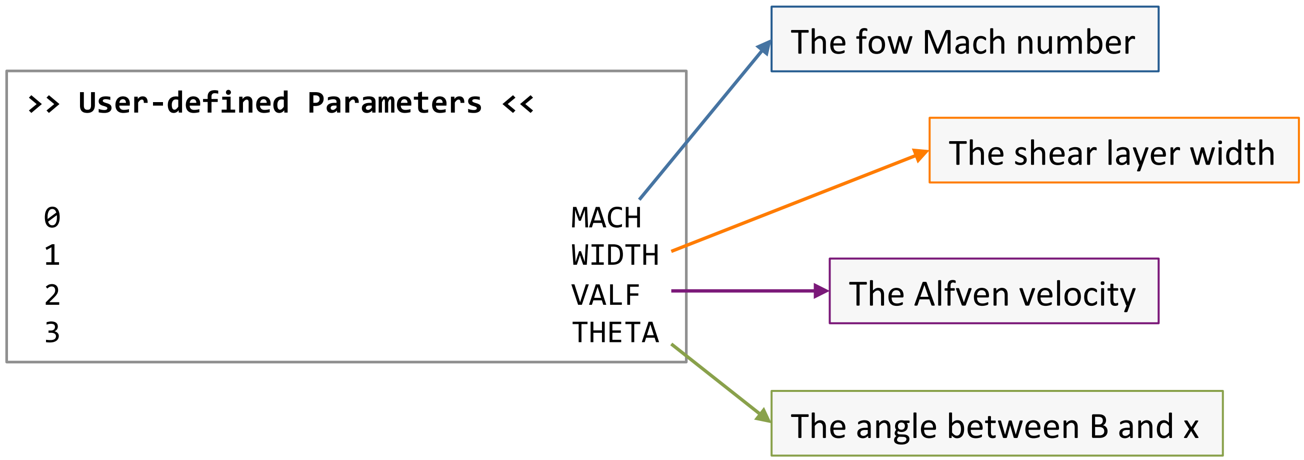
```
>> Setup problem <<

 PHYSICS                         MHD
 DIMENSIONS                      2
 COMPONENTS                      3
 GEOMETRY                        CARTESIAN
 BODY_FORCE                      NO
 FORCED_TURB                     NO
 COOLING                         NO
 RECONSTRUCTION                  PARABOLIC
 TIME_STEPPING                   CHARACTERISTIC_TRACING
 DIMENSIONAL_SPLITTING           NO
 NTRACER                         1
 USER_DEF_PARAMETERS             4
```

- We are now using the *MHD module*, in *2 dimensions* (3 components) and have *4 user-defined parameters*.

Setting the Parameters

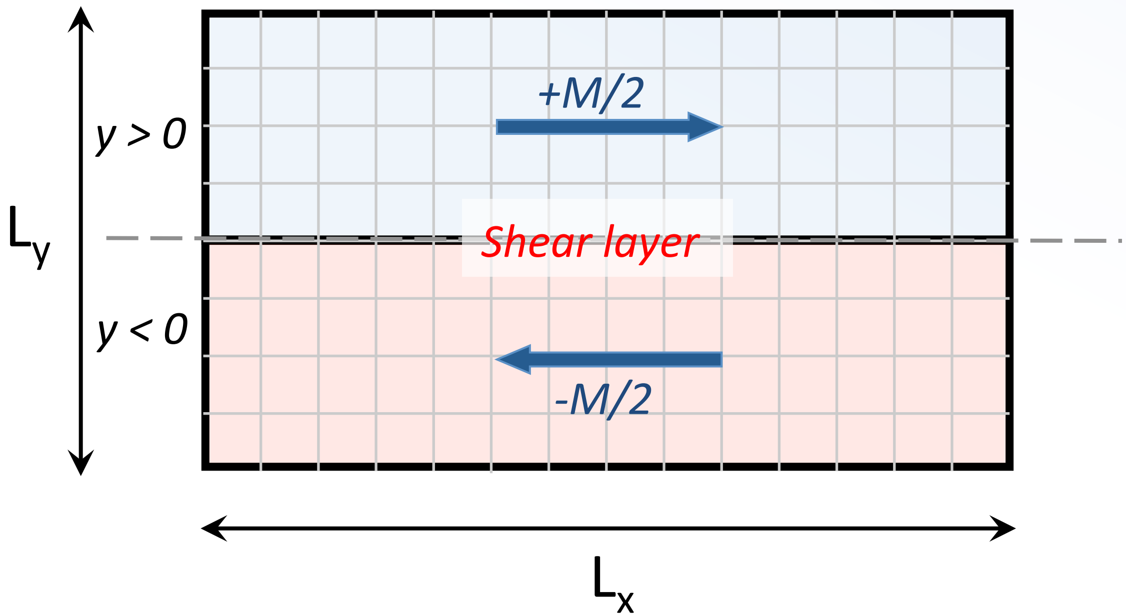
- We can use the constrained transport method to control the divergence of B, and define the name of 4 parameters that will be used later in our problem definition.

- Just scroll down with your arrow keys and check that the your userdef parameter names are already set to



Setting Initial & Boundary Conditions

- We consider a 2D Cartesian box, filled with two uniform fluids ($\rho = 1$, $p = 1/\Gamma$) in pressure equilibrium and in relative motion:



- For simplicity we choose a periodic boundary in the x-direction and zero-gradient boundary conditions in the y-direction.

Specifying Initial Conditions:

- Initial condition must be coded inside `init.c` file using the `Init()` function;
- This file is always in your local working directory.

```c
/* ********************************************************************* */
void Init (double *v, double x, double y, double z)
/*
 *
 ********************************************************************* */
{
  static int first_call = 1;
  double rnd, eps;
  double M     = g_inputParam[MACH];
  double w     = g_inputParam[WIDTH];
  double vA    = g_inputParam[VALF];
  double theta = g_inputParam[THETA]/180.0*CONST_PI;

  if (first_call == 1){          /* Seed random number sequence */
    srand(time(NULL) + prank);
    first_call = 0;
  }
  rnd = (double)(rand())/((double)RAND_MAX + 1.0); /* Generate random number */
  eps = 0.01*M;                                    /* Perturbation amplitude */

  v[RHO] = 1.0;                                 /* Set constant density  */
  v[VX1] = 0.5*M*tanh(y/w);                     /* Main flow             */
  v[VX2] = eps*2.0*(rnd-0.5)*exp(-y*y*10.0);    /* Vertical perturbation */
  v[VX3] = 0.0;

  v[PRS] = 1.0/g_gamma;                 /* Set constant pressure (cs = 1) */
  v[TRC]  = (y < 0.0 ? 1.0:-1.0);       /* Passive tracer (color)         */

  #if PHYSICS == MHD
  v[BX1] = vA*cos(theta);          /* Magnetic field lies in the x-z plane.      */
  v[BX2] = 0.0;                    /* Theta is the angle between B and x-direction */
  v[BX3] = vA*sin(theta);

  v[AX1] = 0.0;
  v[AX2] = 0.0;
  v[AX3] = y*v[BX1];
  #endif
}
```

Runtime Parameters: pluto.ini

- At runtime, PLUTO reads the pluto.ini file which is used to control several options use by the code at runtime, such as grid generation, CFL number, boundary conditions, output type and so forth.

- This file can be edited manually.

```
[Grid]

X1-grid  1   0.0  160   u   1.0
X2-grid  1  -1.0  320   u   1.0
X3-grid  1   0.0    1   u   1.0

[Time]

CFL              0.8
CFL_max_var      1.1
tstop            10.0
first_dt         1.e-4

[Solver]

Solver         roe

[Boundary]

X1-beg        periodic
X1-end        periodic
X2-beg        outflow
X2-end        outflow
X3-beg        outflow
X3-end        outflow

[Static Grid Output]

uservar    0
dbl        10.0  -1   single_file
flt       -1.0  -1   single_file
vtk        0.1  -1   single_file
tab       -0.1  -1
ppm       -1.0  -1
png       -1.0  -1
log        10
analysis  -1.0  -1

[Parameters]

MACH                        5.0
WIDTH                       0.00001
VALF                        0.1
THETA                       90.0
```

Compiling and Running

- Compile by typing “make” at the terminal

- Run by typing either

```
> ./pluto   # Serial run (Linux.gcc.defs must have been selected)
```

or

```
> mpirun -np <n> ./pluto   # Parallel run (Linux.mpicc.defs must have been selected)
```

- Depending on your processor speed, the code may take few minutes to run….

Visualization with IDL

- Set the IDL path to include PLUTO IDL script directory:

```
> export IDL_PATH='<IDL_DEFAULT>:'$HOME/tmp/PLUTO/Tools/IDL
```

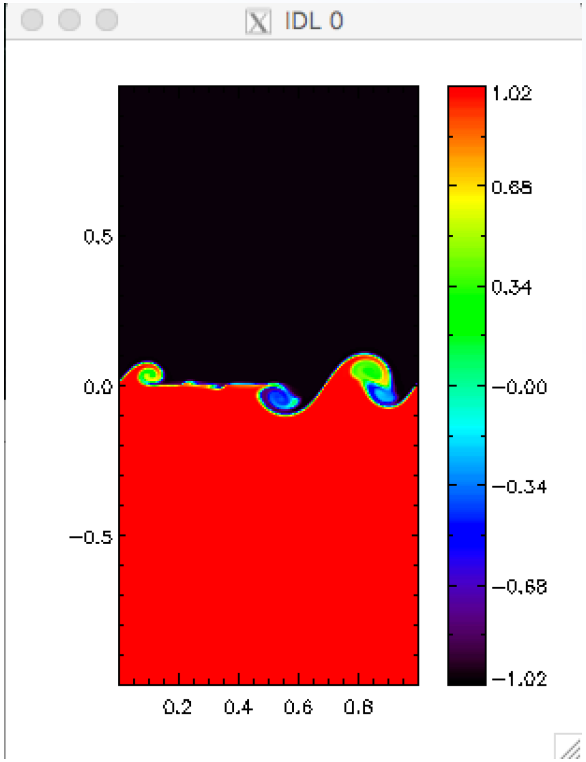
- Launch IDL,

```
IDL> pload,/vtk,25
```

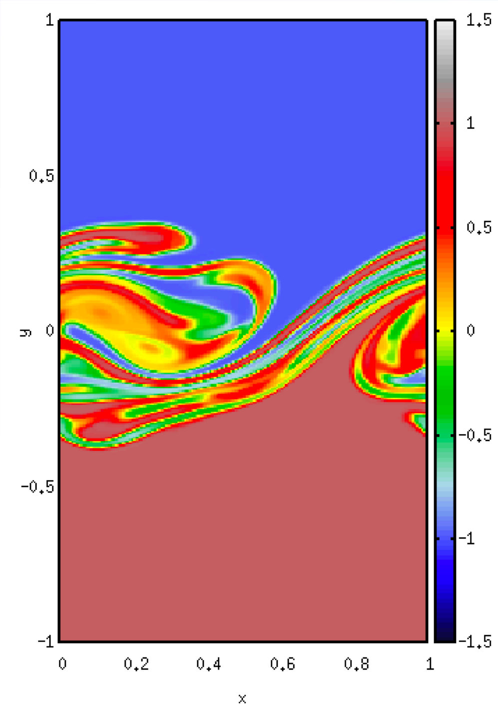
Load output data #25 in vtk format

```
IDL> display,x1=x1,x2=x2,/vbar,tr1
```

Display, the tracer field adding Coordinates and vertical colorbar.



Analysis of the KHI



- The script “kh.gp” can be used to produce an animation with gnuplot.
- Study the evolution of the instability without magnetic field (vA = 0).
  - Can you identify the linear phase and the transition to the subsequent nonlinear evolution ?
  - Can you measure the growth rate of the instability ?
- Increase the magnetic field strength (0 < vA < 1) and confirm that the instability becomes suppressed as vA → cs in the parallel case (θ = 0).
- Change the magnetic field geometry by considering the perpend. case (θ = π/2). What do you see ?

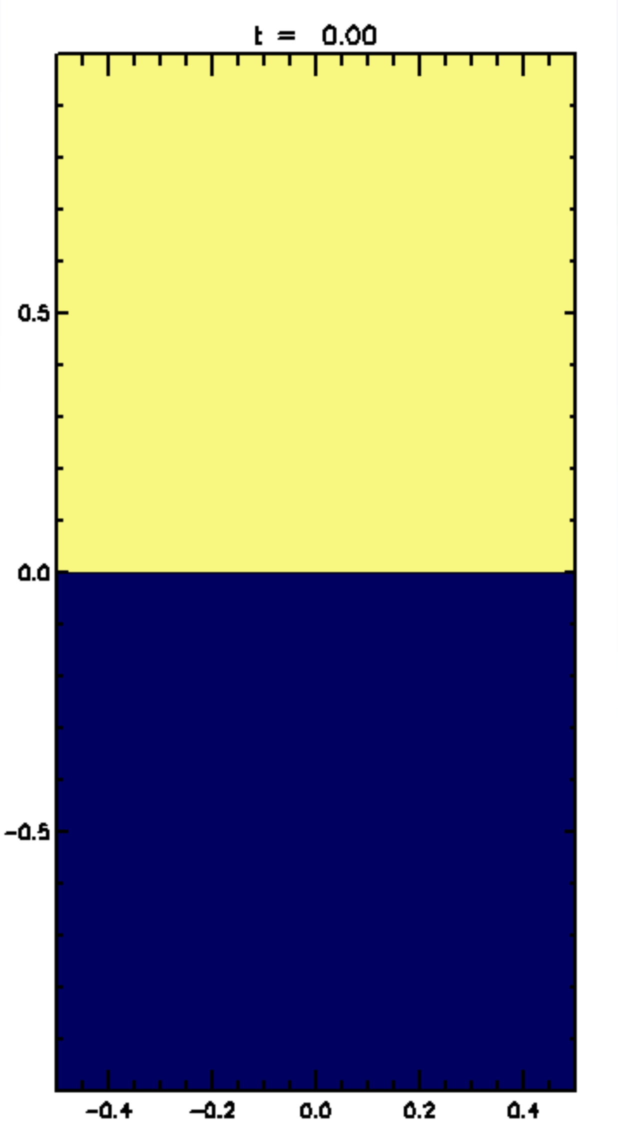
EXAMPLE #4:
RAYLEIGH-TAYLOR INSTABILITY

The Rayleigh-Taylor Instability

- The Rayleigh-Taylor (RT) instability occurs at the interface between two fluids of different densities, when the light one supports the heavier ones against gravity.

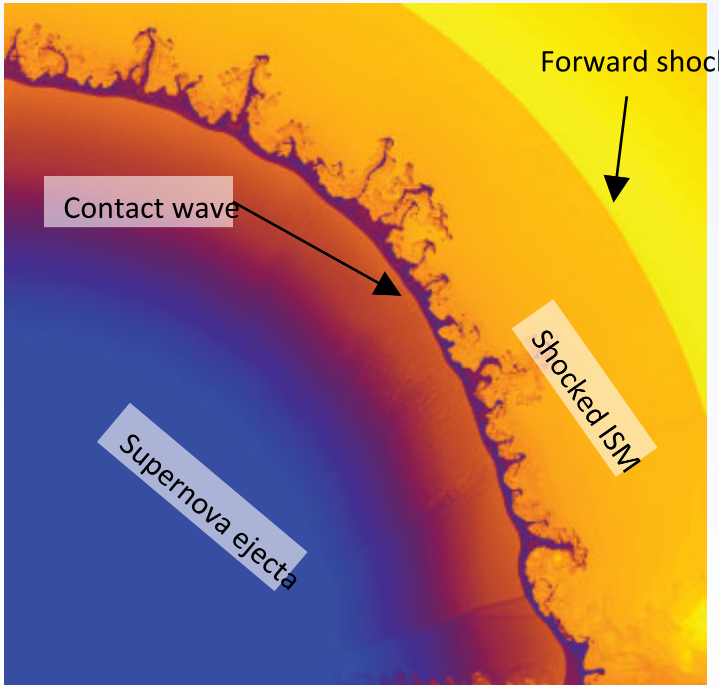
- The amplitude growths and the further upward motion of the lighter fluid assumes the form of rising bubble while the sinking fluid become finger-shaped.

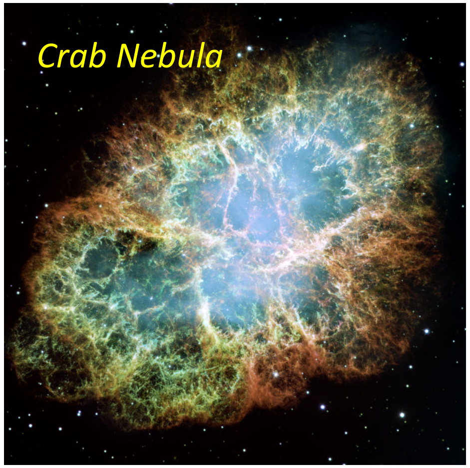
- As the instability proceeds, fingers evolve into mushroom-like vortex motion accompainied by secondary shear flow instabilities.



Astrophysical Application: SN Remnants

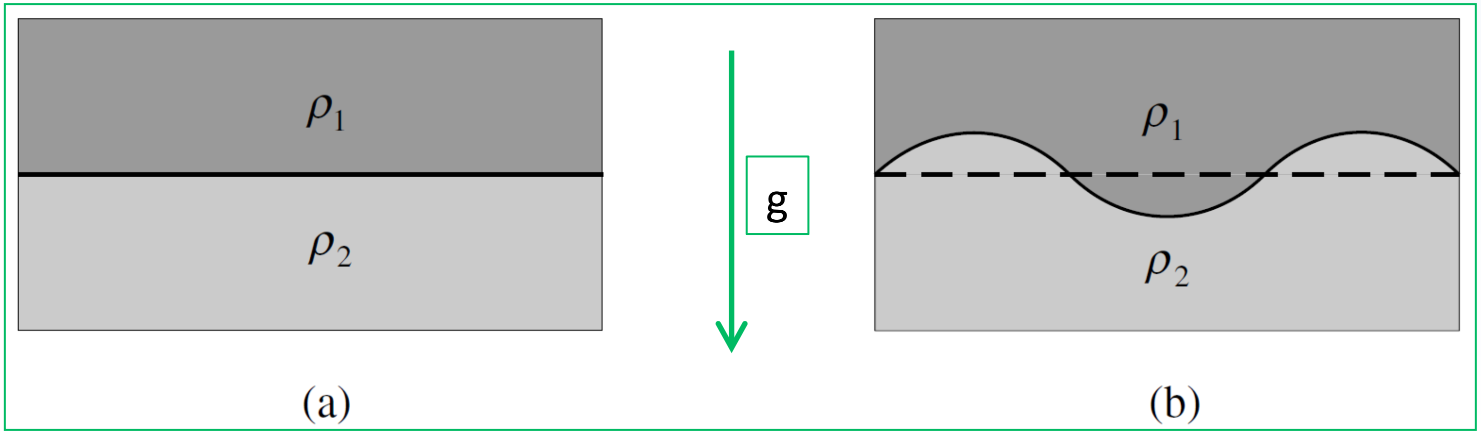
- In a supernova (SN) explosion a large amount of energy is released resulting in a formation of a large scale SN remnant (SNR)
- The dense shell of ejected material *decelerates* in a rarefied interstellar medium (ISM) and is unstable to RT-type instabilities.
- Responsible for finger-like structures of material protruding from the contact discontinuity between the two media.
- These instabilities modify the morphology of the SNR causing a departure of the ejecta from spherical symmetry.





RTI: Analysis

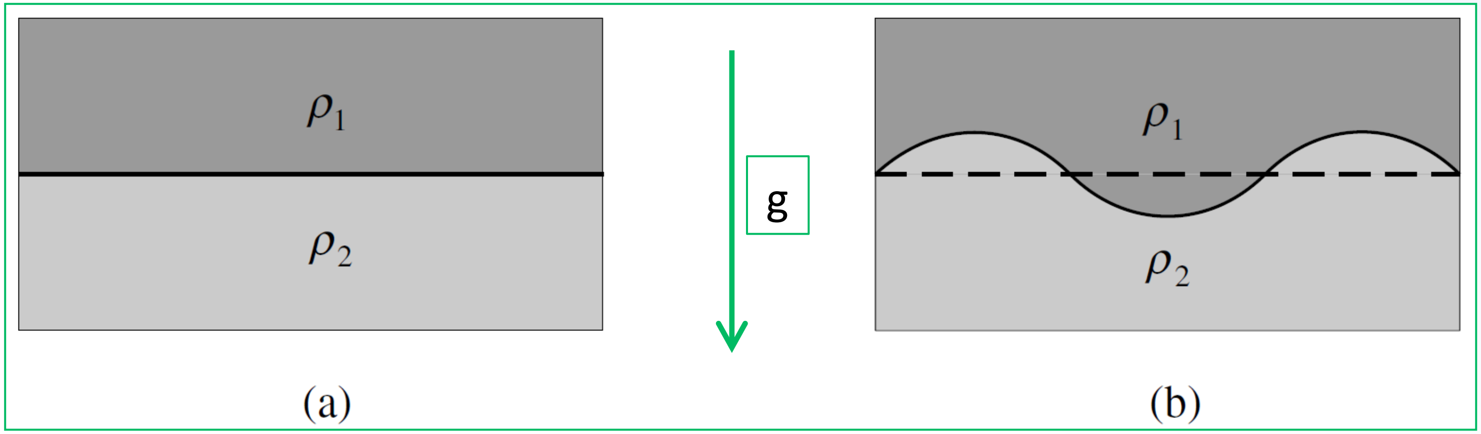
- Consider the case where two fluids with densities ρ_1 and ρ_2 are on top of each other:



- This is an equilibrium situation if the stratification is supported by a pressure gradient: grad(p) = ρg.

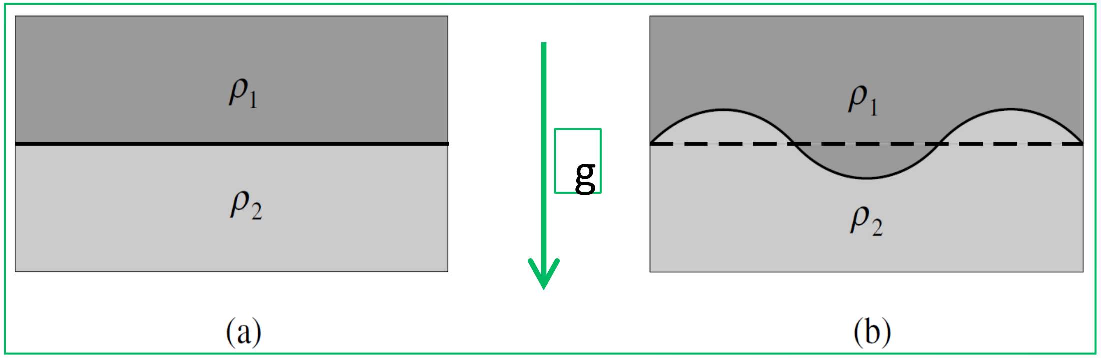
- Is this a stable equilibrium ?

Rayleigh-Taylor Instability (RTI)



- Let's now perturb the equilibrium by rippling the boundary layer.
- The fluid element of density ρ_1 has moved downwards with consequent loss of gravitational potential energy while the opposite is true of the fluid element of density ρ_2.
- It is intuitively obvious that the only stable equilibrium is to have the denser fluid supporting the less dense. The instability that arises when $\rho_2 < \rho_1$ is called the Rayleigh–Taylor instability.

RTI: Linear Analysis



- From normal mode analysis it is found that, in absence of magnetic field,

$$\boxed{\omega^2 = -\frac{kg(\rho_1 - \rho_2)}{\rho_1 + \rho_2}}$$

- If $\rho_1 < \rho_2$ the equilibrium is *stable* → surface gravity waves

- If $\rho_1 > \rho_2$ the equilibrium is *unstable* → Rayleigh-Taylor instability

RTI: Linear Evolution

- When $\rho_1 > \rho_2$ the growth rate is purely imaginary and this corresponds to an instability:

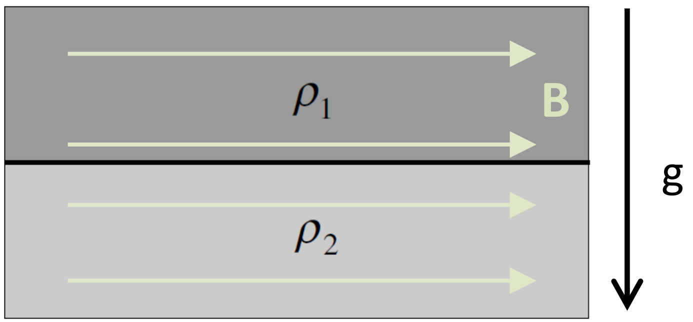
$$\omega = \pm i\sqrt{kg\frac{\rho_1 - \rho_2}{\rho_1 + \rho_2}}$$

- The heavier fluid will try to sink below the lighter fluid.

- Clearly, the fastest growth rate $(kg)^{1/2}$ occurs when $\rho_1 >> \rho_2$.

RTI: Effects of Magnetic Fields

- In a plasma, density and pressure are tied, magnetic field can change pressure but not density.

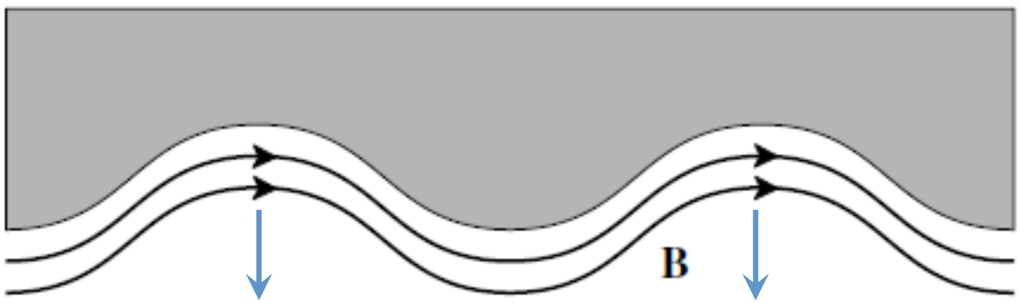


(a)

- The presence of a uniform magnetic field has the effects of reducing the growth rate of modes parallel to it, although the interface still remains Rayleigh–Taylor unstable in the perpendicular direction.

Magnetized RTI: Linear Analysis

- The growth rate now becomes $\omega^2 = -|\boldsymbol{k}|\, g \dfrac{\rho_1 - \rho_2}{\rho_1 + \rho_2} + 2\dfrac{(\boldsymbol{k} \cdot \boldsymbol{B}_0)^2}{4\pi(\rho_1 + \rho_2)}$
- Shorter wave modes with $k > \dfrac{g(\rho_1 - \rho_2)}{2(B_0^2/4\pi)\cos^2\theta}$ are stabilized.
- Bending of field lines resists to the growth of perturbation:

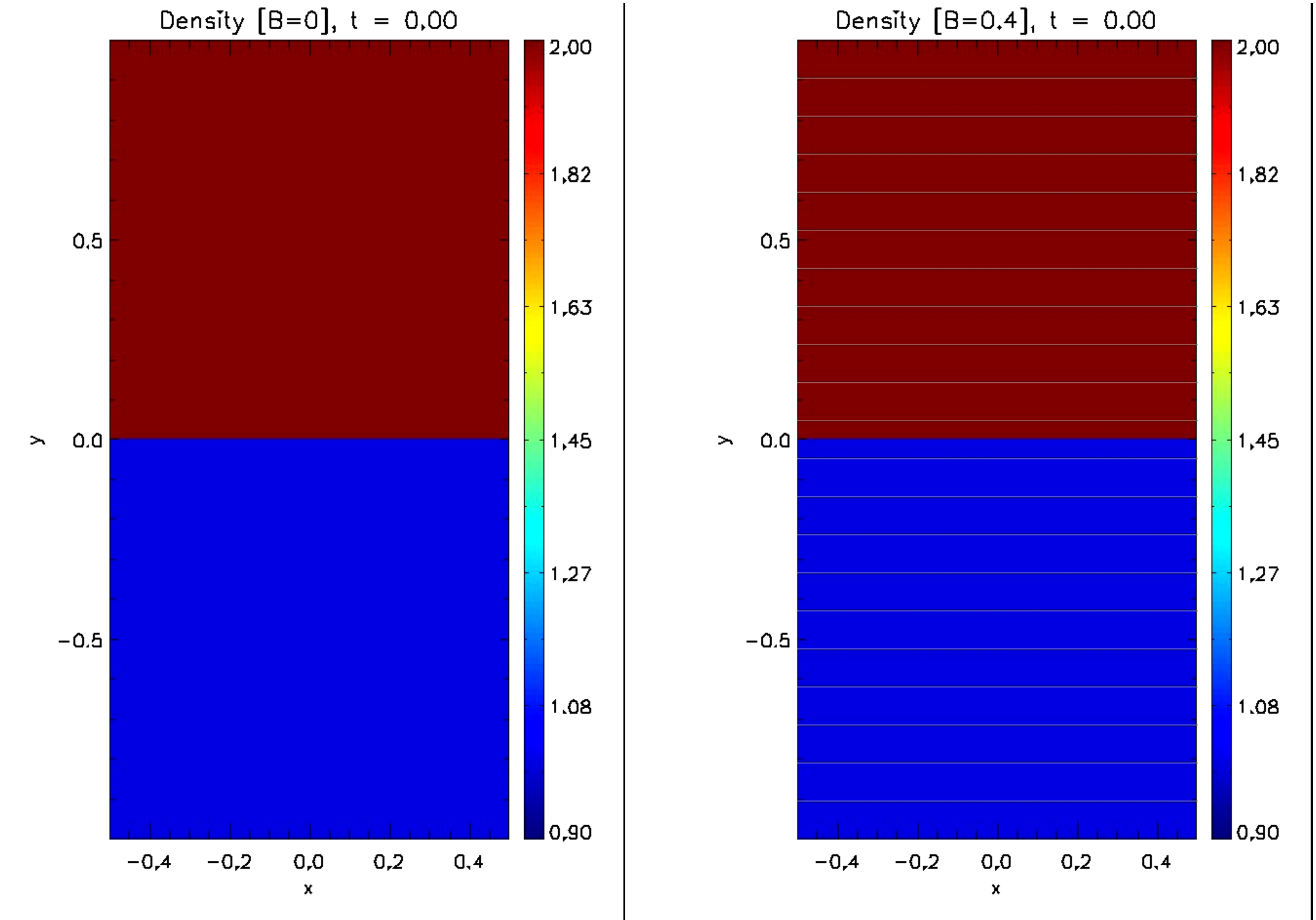


(b)

- We conclude that the k-th wavenumber becomes stabilized when

$$\frac{B^2}{4\pi} \geq \frac{g(\rho_1 - \rho_2)}{2k\cos^2\theta}$$

Nonlinear Evolution: HD vs MHD



Initial condition

```c
/* ********************************************************************* */
void Init (double *v, double x, double y, double z)
/*
 *
 *********************************************************************** */
{
  static int first_call = 1;
  double rnd, Bc;

  if (first_call == 1){          /* Seed random number sequence */
    RandomSeed(time(NULL),0);
    first_call = 0;
  }
  rnd = RandomNumber(-1,1);         /* Generate random number in (-1,1) */

  if (y < 0.0) v[RHO] = 1.0;
  else         v[RHO] = g_inputParam[ETA];

  v[PRS] = 1.0/g_gamma + v[RHO]*g_grav*y;  /* Hydrostatic balance */
  v[VX1] = v[VX2] = v[VX3] = 0.0;

/* -----------------------------------
    Add perturbation
   ----------------------------------- */

  v[VX2] = 1.e-2*rnd*exp(-y*y*200.0);

/* -----------------------------------
    Set magnetic field in units of Bc
   ----------------------------------- */

  #if PHYSICS == MHD
  Bc = sqrt((g_inputParam[ETA] - 1.0)*fabs(g_grav)); /* Critical field */
  v[BX1] = g_inputParam[CHI]*Bc/sqrt(4.0*CONST_PI);
  v[BX2] = 0.0;
  v[BX3] = 0.0;
  v[AX1] = v[AX2] = 0.0;
  v[AX3] = y*v[BX1];
  #endif

}
```

We define the density ratio $\eta = \rho_2/\rho_1$ and normalize density to the fluid on top (ρ_1=1).

Hydrostatic equilibrium is defined by

$$\rho(y) = \begin{cases} 1 & \text{for} \quad y \le 0 \\ \eta & \text{for} \quad y > 0 \end{cases}, \qquad p = p_0 + \rho y g$$

The parameter χ controls the magnetic field strength:

$$\frac{B}{\sqrt{4\pi}} = \chi\sqrt{\frac{g\rho_1}{2k_{\min}\cos^2\theta}}$$

We expect instability when

$$0 \le \chi \le \sqrt{1-\eta}$$

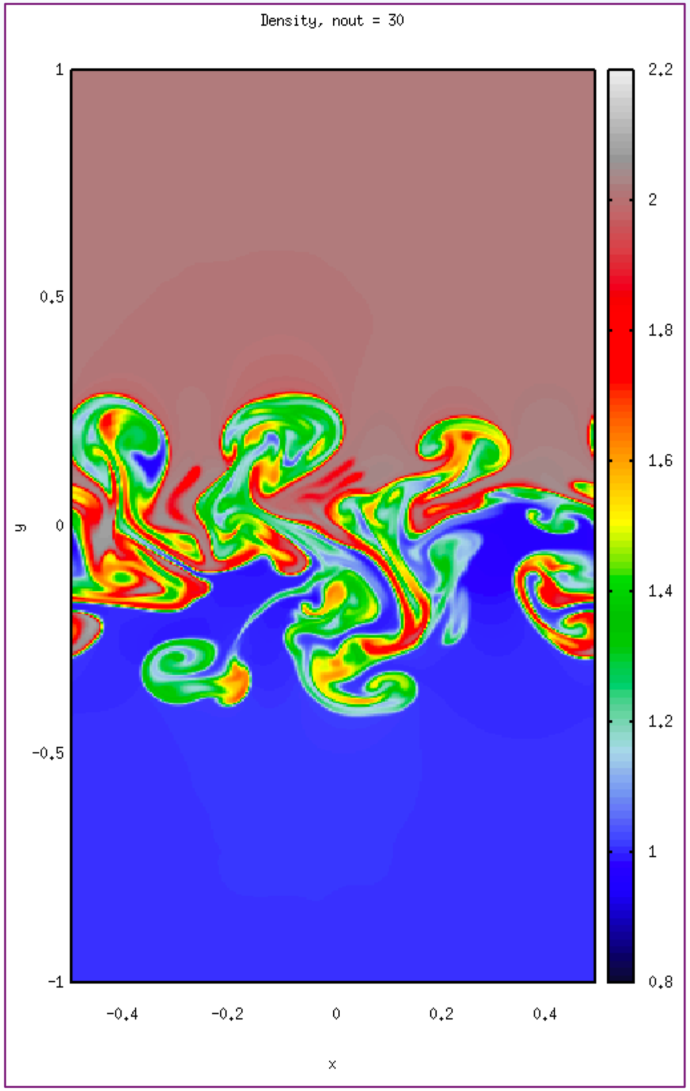
Analysis of the RTI

- Start from a non-magnetized configurations ($\chi = 0$) and check the evolution of the instability;

- Re-run by slightly increasing the magnetic field strength ($\chi = 0.1, 0.2\ldots$). What do you see ? Explain.

- How do small-wavelength modes grow compared to large scale perturbations ?

- Set $\rho_1=\rho_2$. Waves are formed. What are they ?



THE END