Star-Planet Interaction with PLUTO code

Miljenko Čemeljić

Nicolaus Copernicus Astronomical Center, PAN Warsaw

&

Institute of Physics, Silesian University in Opava, Czech Republic

&

ASIAA Visiting Scholar, Taipei, Taiwan

in collaboration with Jacobo Varela, Universidad Carlos III de Madrid and

Ruchi Mishra in CAMK, Warsaw

General outline

- **INTRO: Motivation, auroras on exoplanets and planets around pulsars**
- **PART I:** General introduction to PLUTO code and its physics modules, installation of the code, testing of the installation with Sod shock tube test in 1D, visualization with gnuplot.
- **PART II:** Setup of 2D simulation from Test Problem template: Orszag-Tang test in 2D and 3D. Detailing of the setup and used files. Visualization of the results with Paraview. Animation of the results, saving of animation.
- **PART III:** Setup of SPI simulations. Step-by-step explanation of the setup. Definition of parameters in the setup. Running of the code in a linux cluster. Visualization of the results with Paraviev, streamlines of the magnetic field and velocity.

- Concluding remarks. Definition of individual tasks.

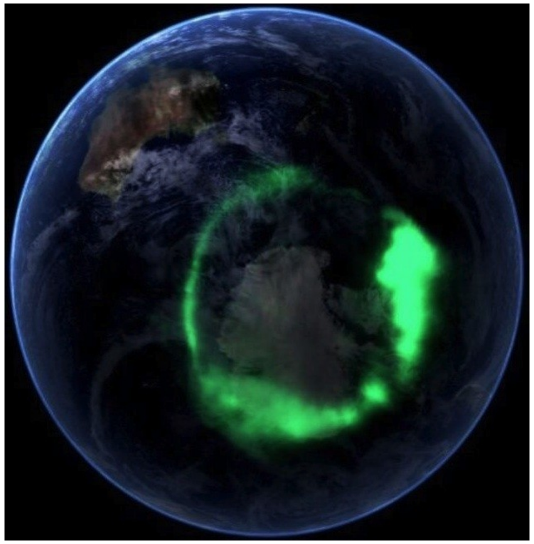
-Check webpage https://web.tiara.sinica.edu.tw/~miki/mikiplutoSPI.html

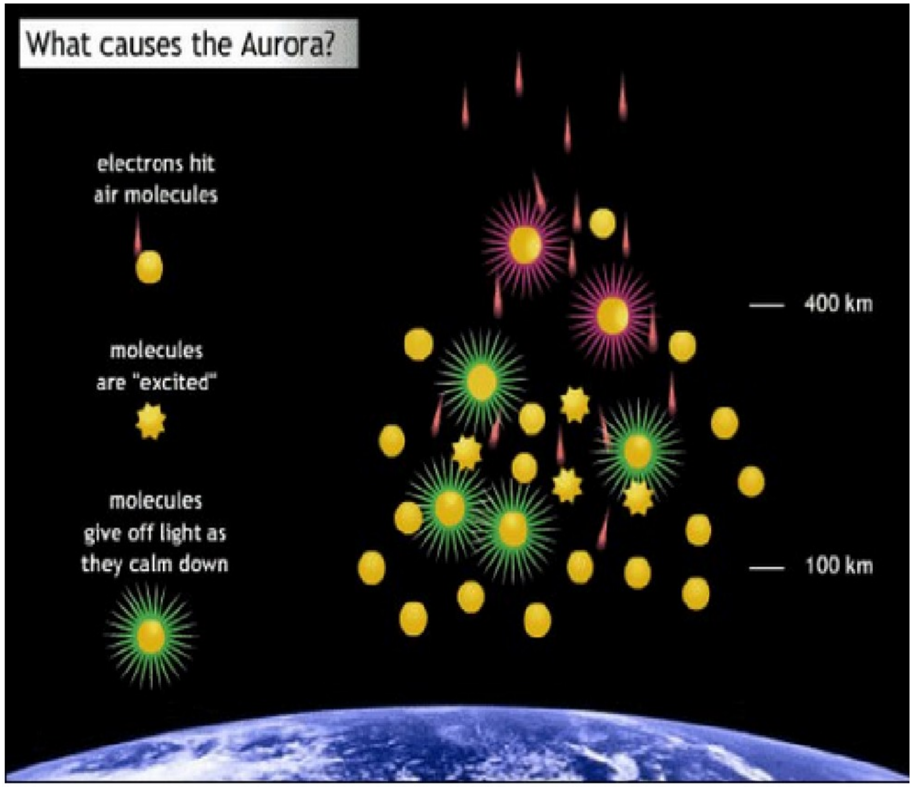
Required packages: PLUTO source code, C-compiler (+MPI), Gnuplot, Paraview.

Auroras on exoplanets and planets around pulsars

- Auroras in Solar System
- Star-planet magnetospheric interaction in simulations
- Results for planets around Sun and exoplanets
- Planets around pulsars, list of (possible) objects
- Preliminary results in our simulations with pulsar parameters
- Summary

Introduction-Earth aurora




What causes the Aurora?
electrons hit air molecules
molecules are "excited"
molecules give off light as they calm down
400 km
100 km

High-speed particles from the Sun, mostly electrons, strike oxygen and nitrogen atoms in Earth's upper atmosphere. Credit: NASA

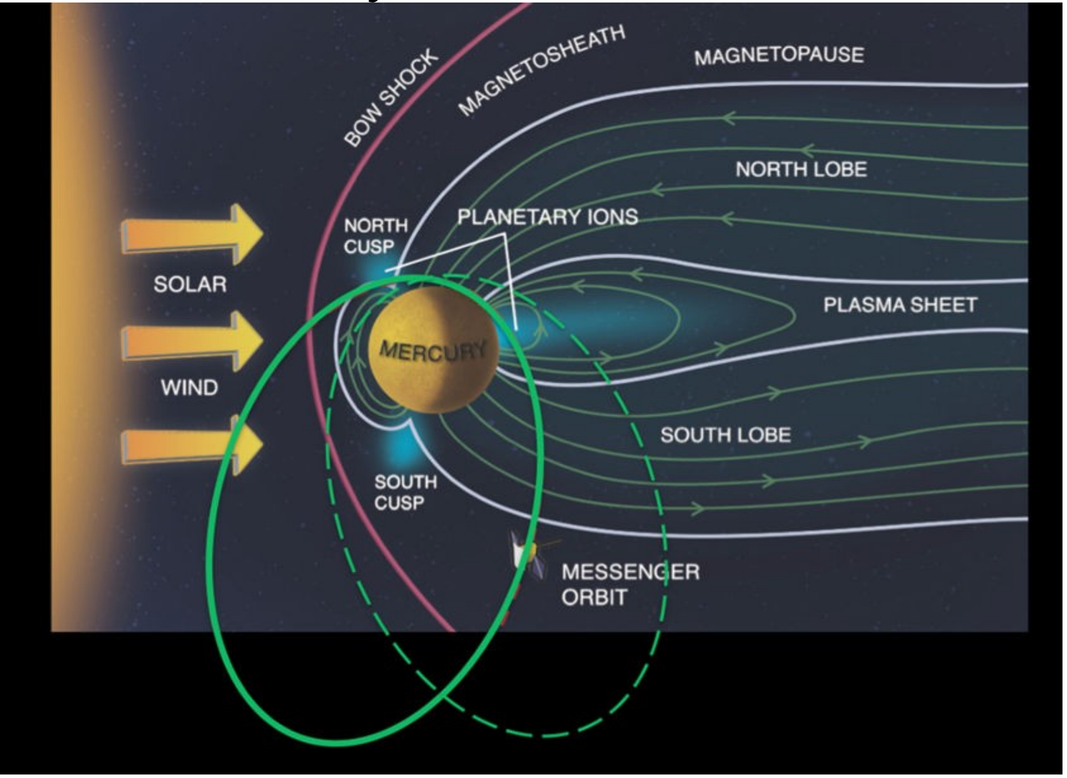
Aurora, named **aurora borealis** (by the Greek goddess of dawn, Aurora, and Greek name for northern wind, Boreas) by Pierre Gassendi in 1621, forms as an outcome of the interaction of a parent star magnetic field with the planetary field. On Earth, aurora is visible close to the geographic poles, since they are also currently close to the magnetic poles of Earth.

Different gases in the upper layers of the atmosphere are emitting light of different colors in collision with particles from the solar wind (mostly electrons in this case). Oxygen emits greenish or brown-red, and nitrogen blue or red light.

Aurora on Mercury, Venus

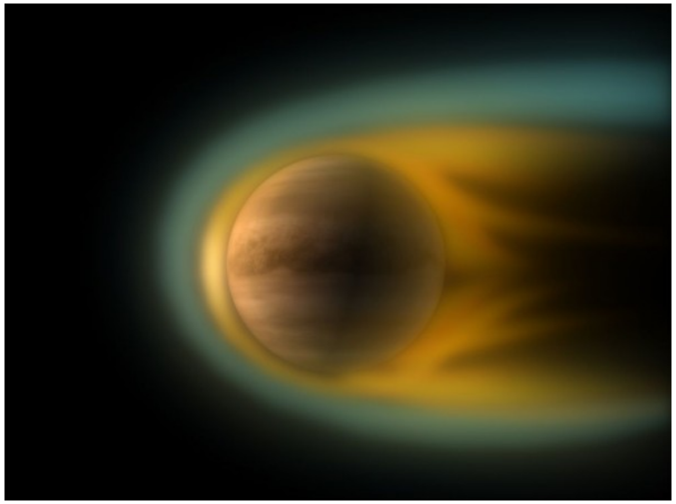
Except on Earth, auroras are found on most of the planets in the Solar system.

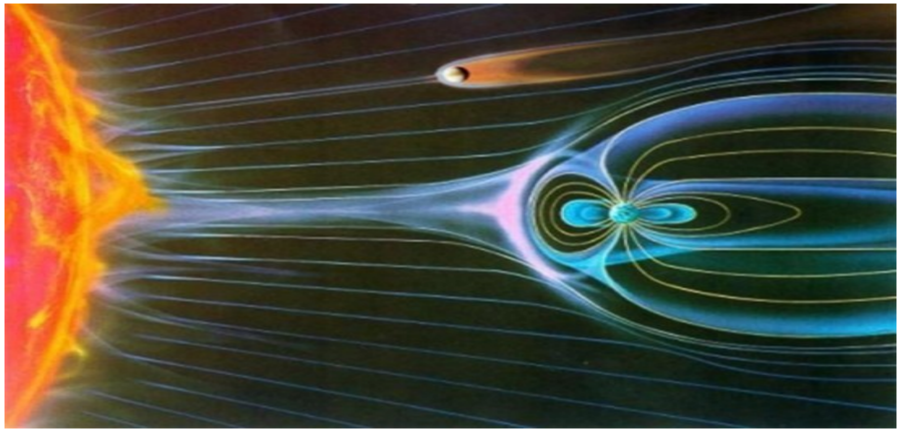
Mercury



Mercury magnetic field is well measured thanks to Messenger probe. Its aurora is similar to Earth's.

Venus

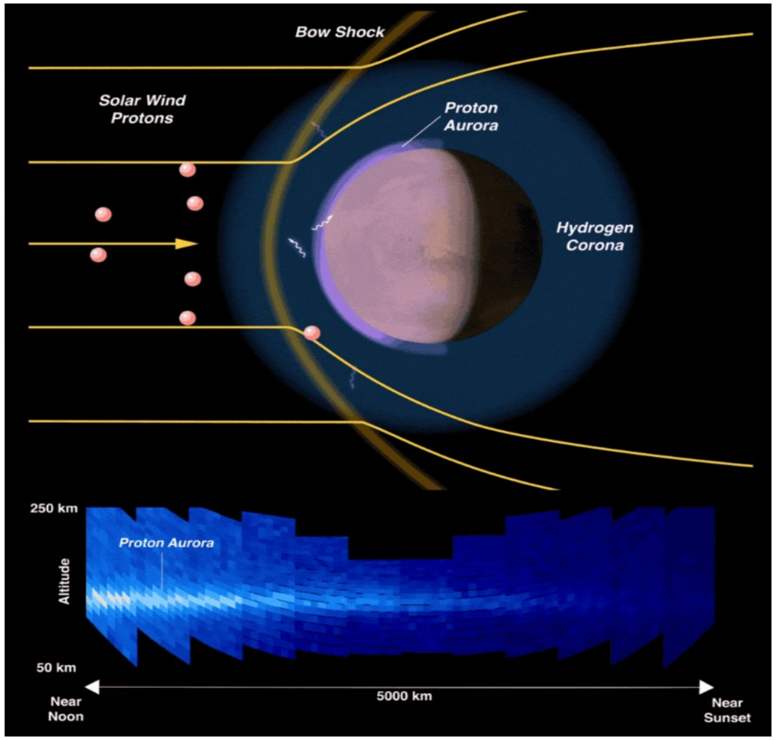




Venus has smaller aurora towards Sun than Earth, here I show a comparison.

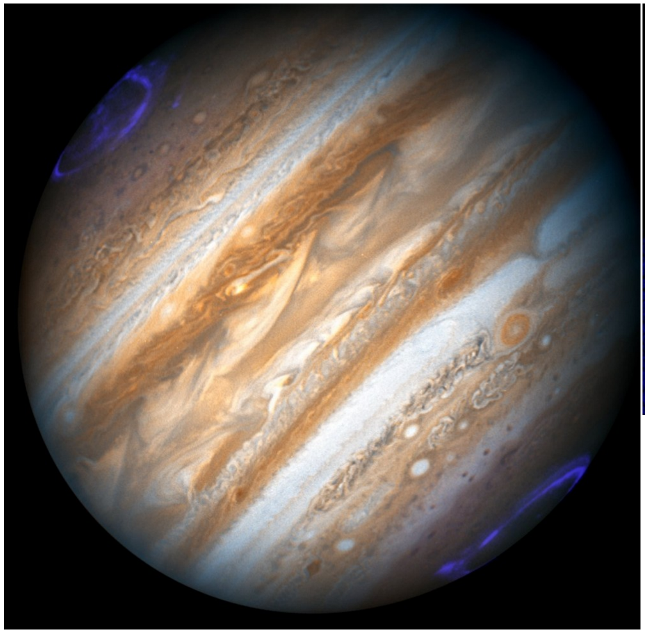
Mars aurora

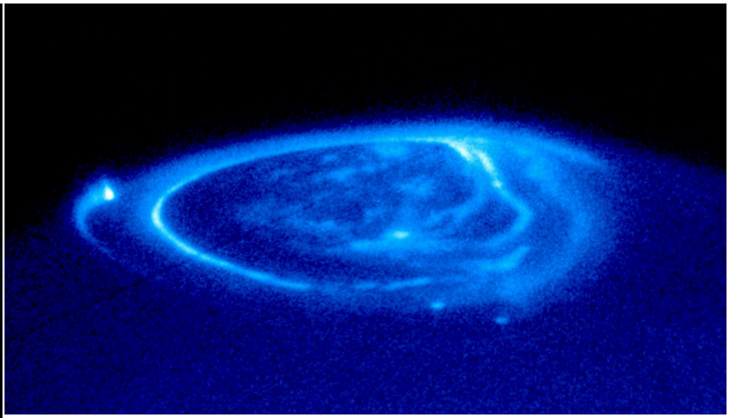
Even in the planets like Mars, which do not have significant magnetic field, we observe aurora, formed as a result of interaction of particles-here mostly protons- from the solar wind shock where the planet moves through the wind. It is most visible at the sunny side of the planet.



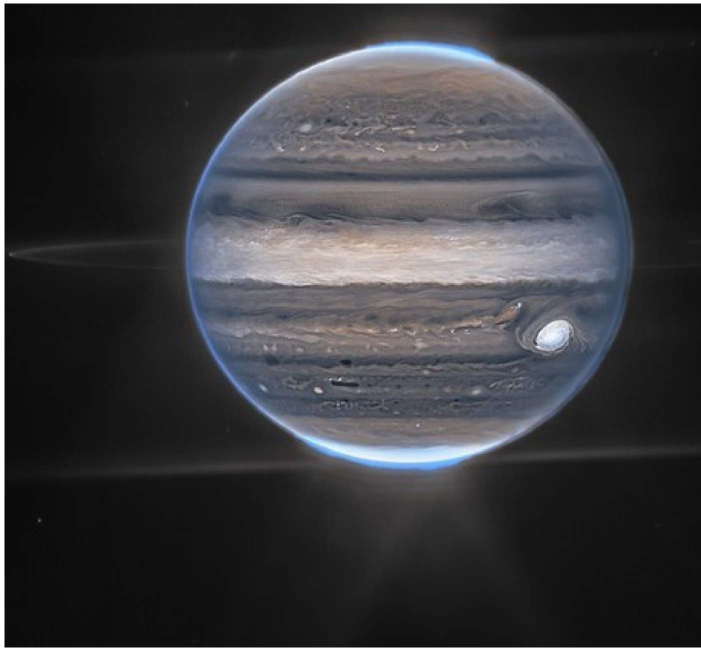
Auroras on large gaseous planets

Aurora is observed also on Jupiter and Saturn. On the gas planets aurora is visible mostly in ultra-violet, so we can observe it from outside our atmosphere.

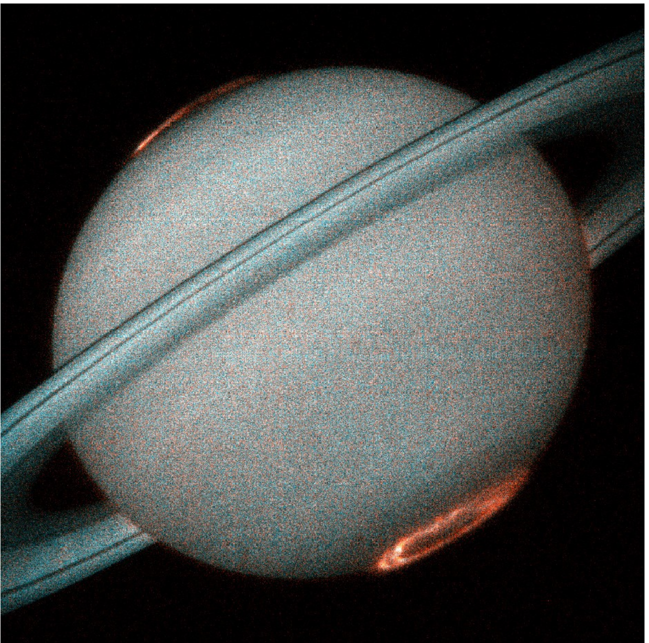




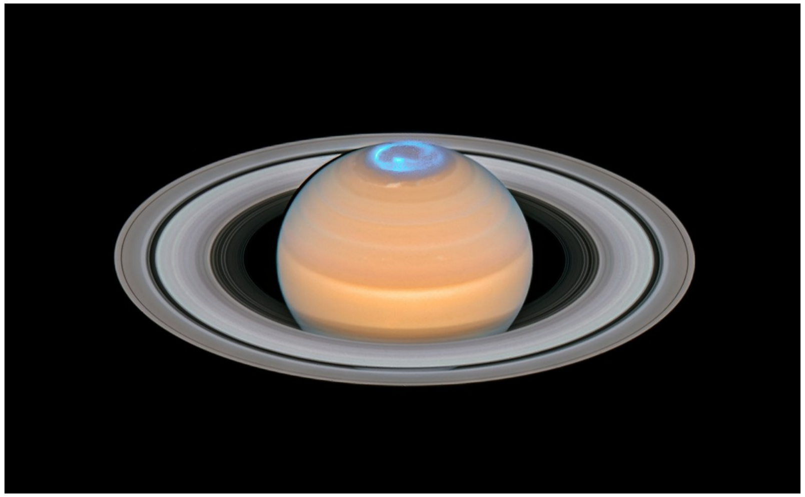
Spots in aurora on Jupiter are magnetically connected with its satellites: the spot on the left side is connected with Io, bottom two with Ganymede and Europe.

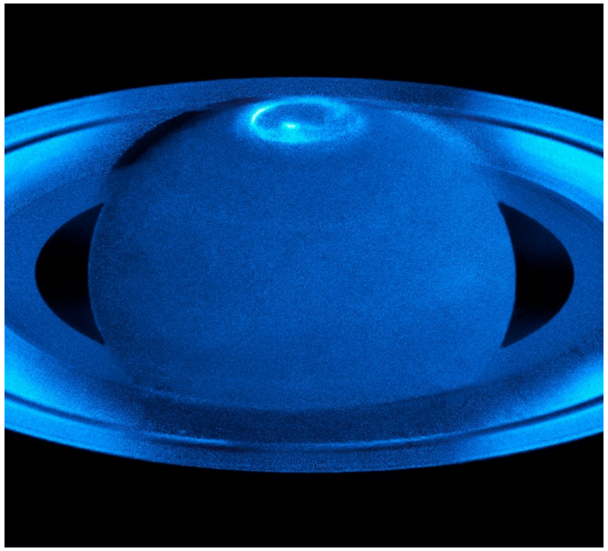


JWST's capture of aurora on Jupiter



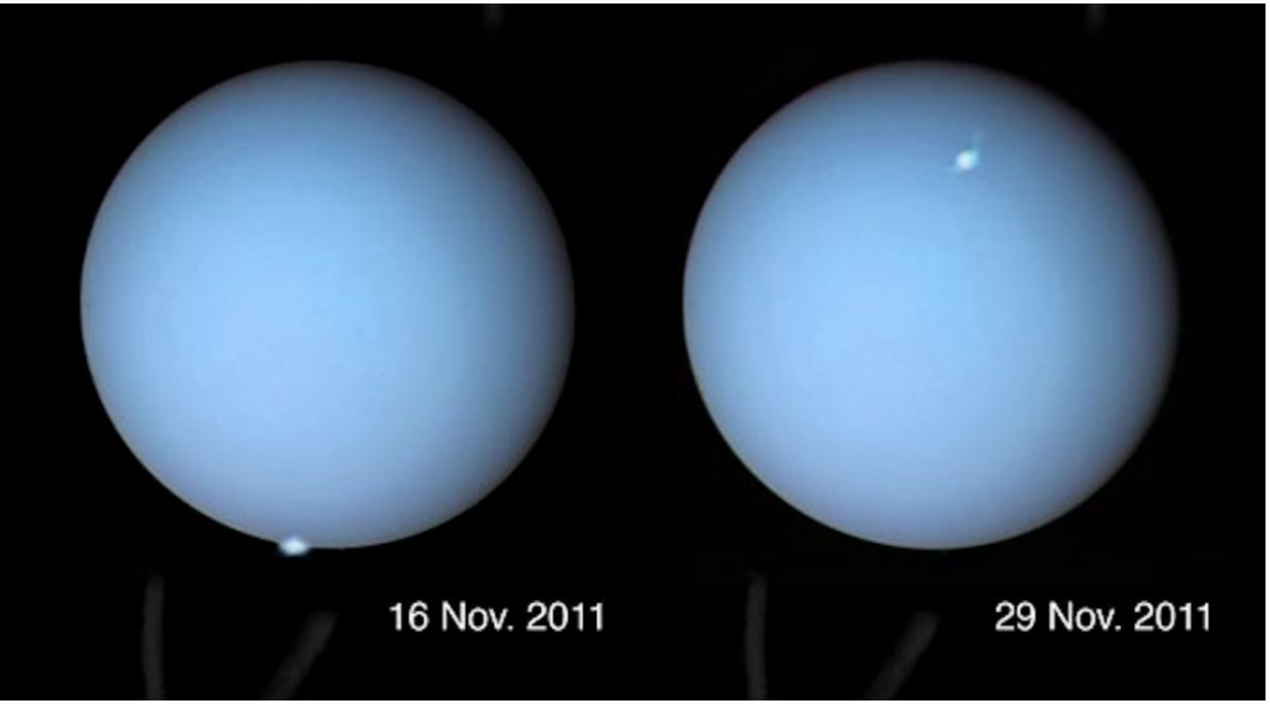
Saturn also features polar aurora:

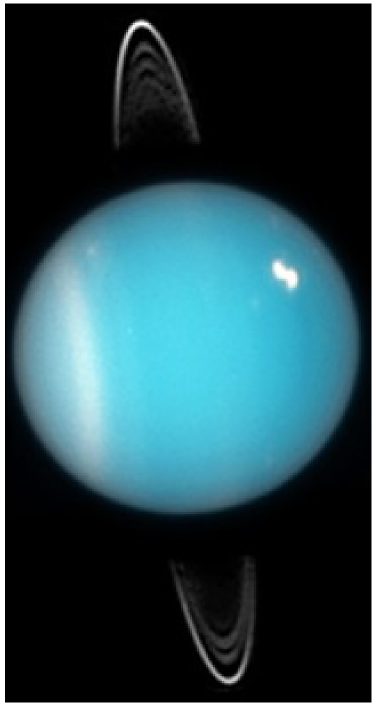




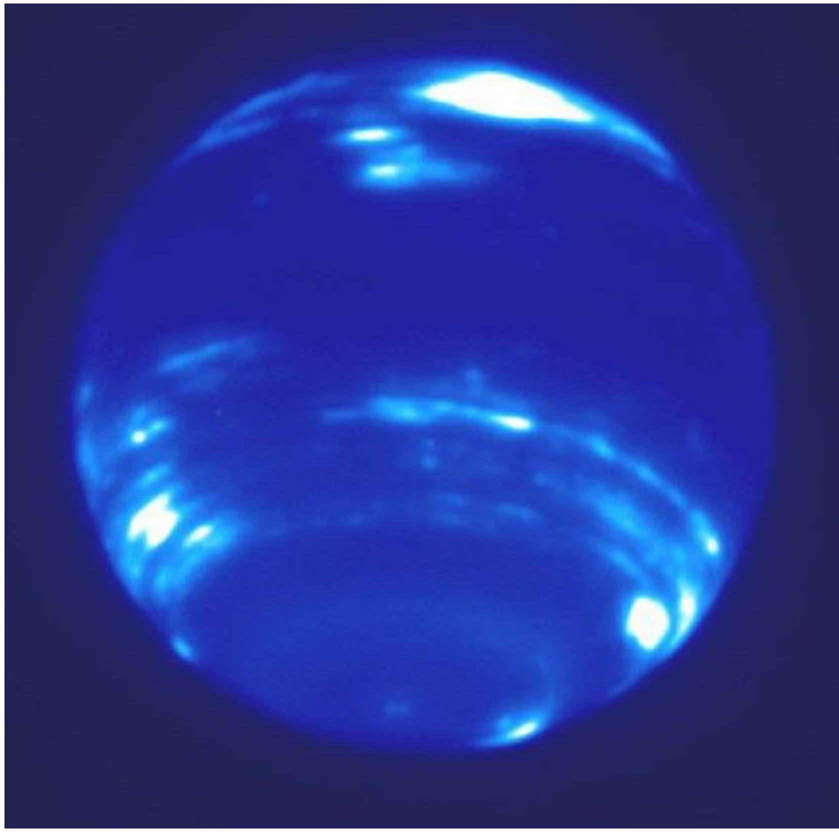
Aurora on Uranus

HST observed auroras on Uranus:





And Keck on Neptune:

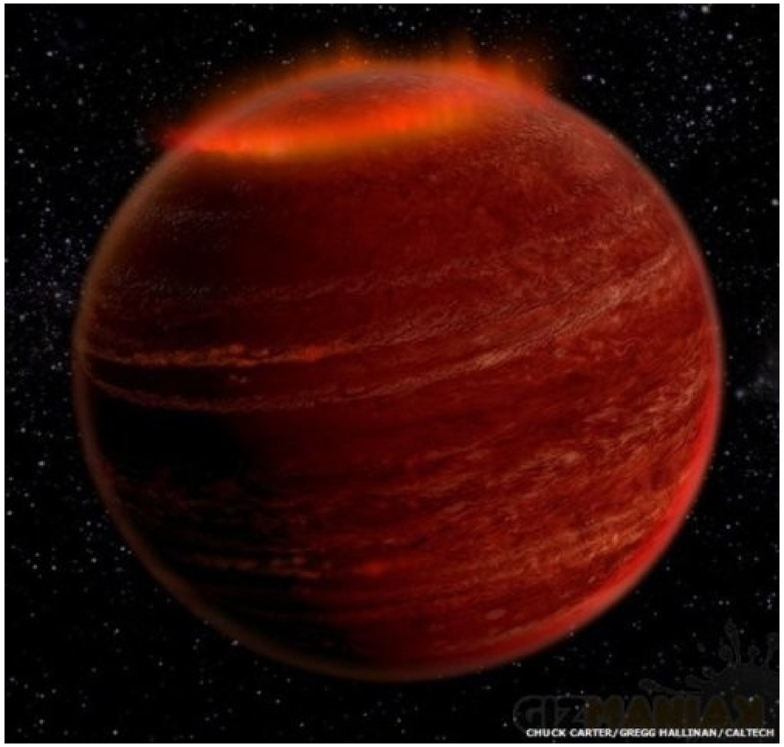


Extrasolar and exoplanet auroras

•As for now, we have an observation of extrasolar aurora on a brown dwarf LSR J1835+3259, 18 lyrs from us, in Lyra. There are more of similar objects which show characteristic spectral features which point to aurora. Shown is an artist impression, not the real observation. It is reddish aurora, from more hydrogen in the atmosphere, and about million times more intense, because of larger magnetic field.

•Such an aurora should also be of different nature, because there is no other star for producing the stellar wind.

•A model for aurora requires a continuously replenished body of plasma within the magnetosphere. This mass-loading can be achieved in multiple ways, including interaction with the interstellar medium, a volcanically active orbiting planet or magnetic reconnection at the photosphere. Alternatively, an orbiting planetary body embedded within the magnetosphere could provide magnetospheric interaction.



In the cases of **exoplanets**, we also expect auroras, and we can use the same simulations and make the predictions for different kinds of planets.

In the cases of **planets around pulsars**, which were actually the first observed exoplanets, we can expect similar effects. Because of much larger field involved, they could behave different from usual planet aurora.

Here we try to make the first such model, by introducing necessary modifications in our star-planet interaction setup.

Numerical simulations of star-planet interaction

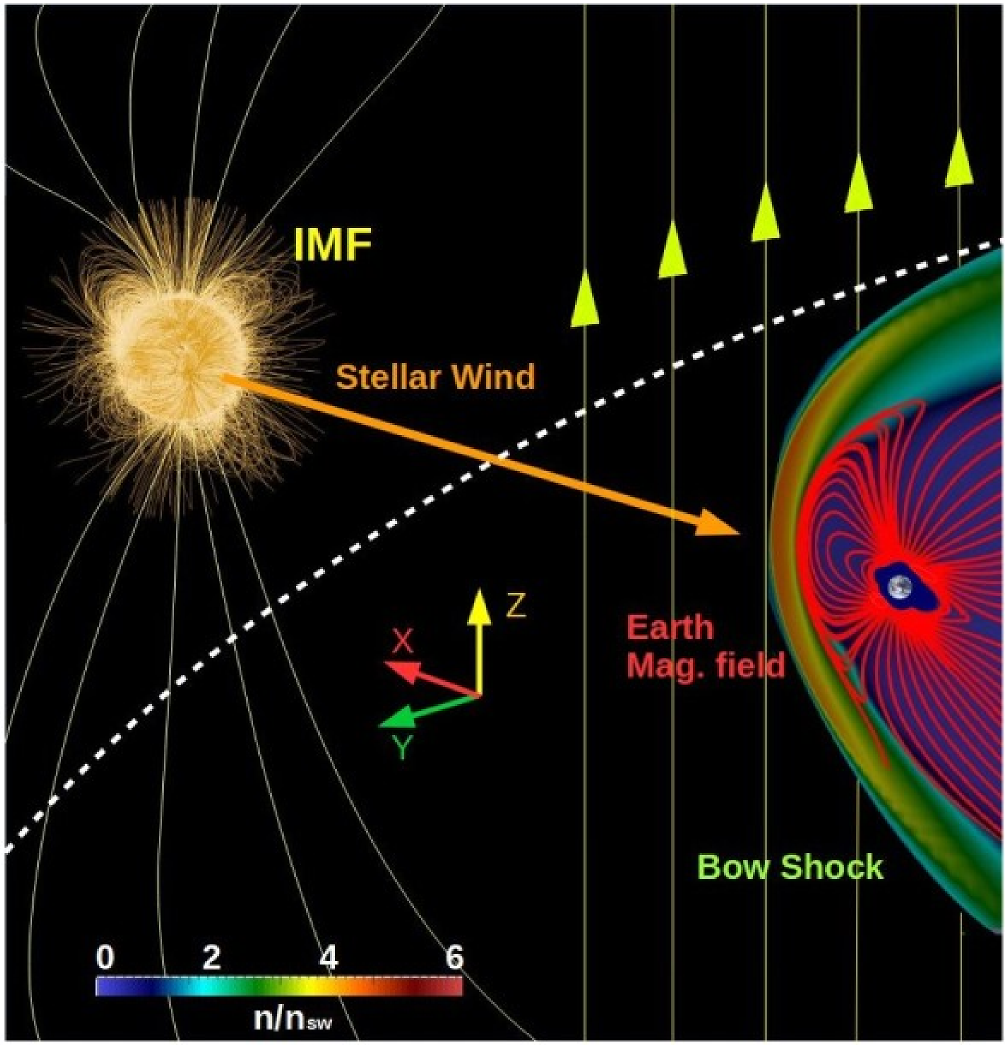


Fig. 1. 3D view of a typical simulation setup. We show the density distribution (color scale), Earth magnetic field lines (red lines), and IMF (yellow lines). The yellow arrows indicate the orientation of the IMF (northward orientation). The dashed white line shows the beginning of the simulation domain (the star is not included in the model).

- In a series of works by Varela et al. (e.g. A&A, 616, A182, 2018; A&A 659, A10, 2022) are given numerical simulations of planetary magnetospheric response in extreme solar wind conditions, using the PLUTO code.
- Such simulations are valid for Earth and exoplanets.
- We use this setup as a template for the much larger magnetic field of pulsar.

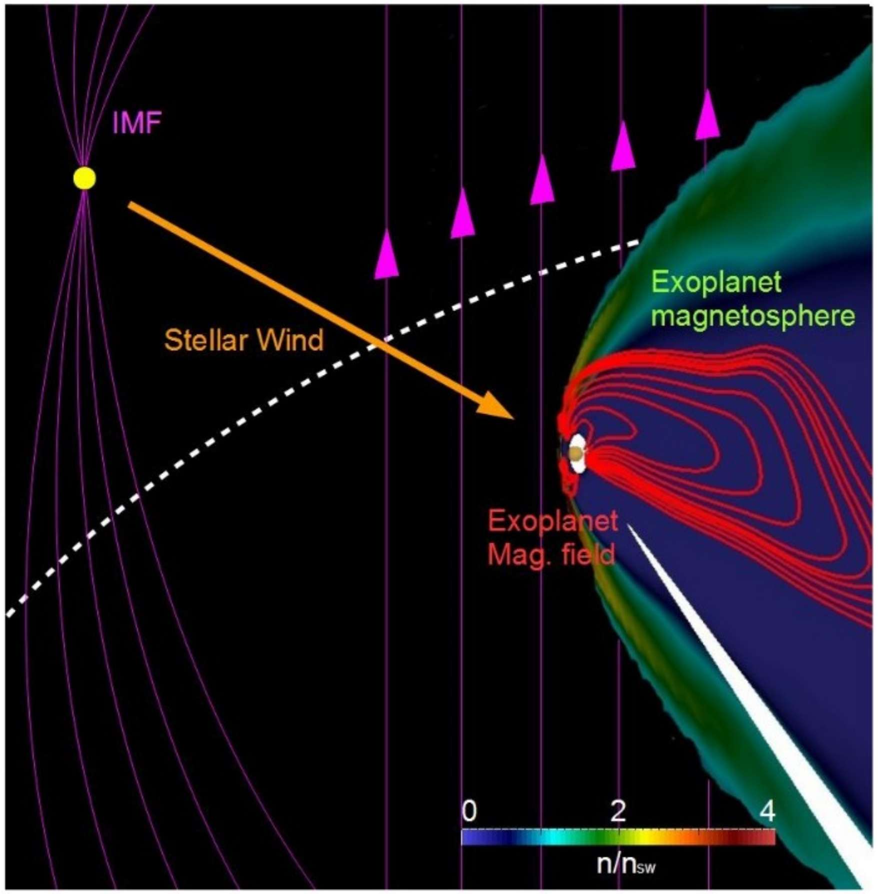
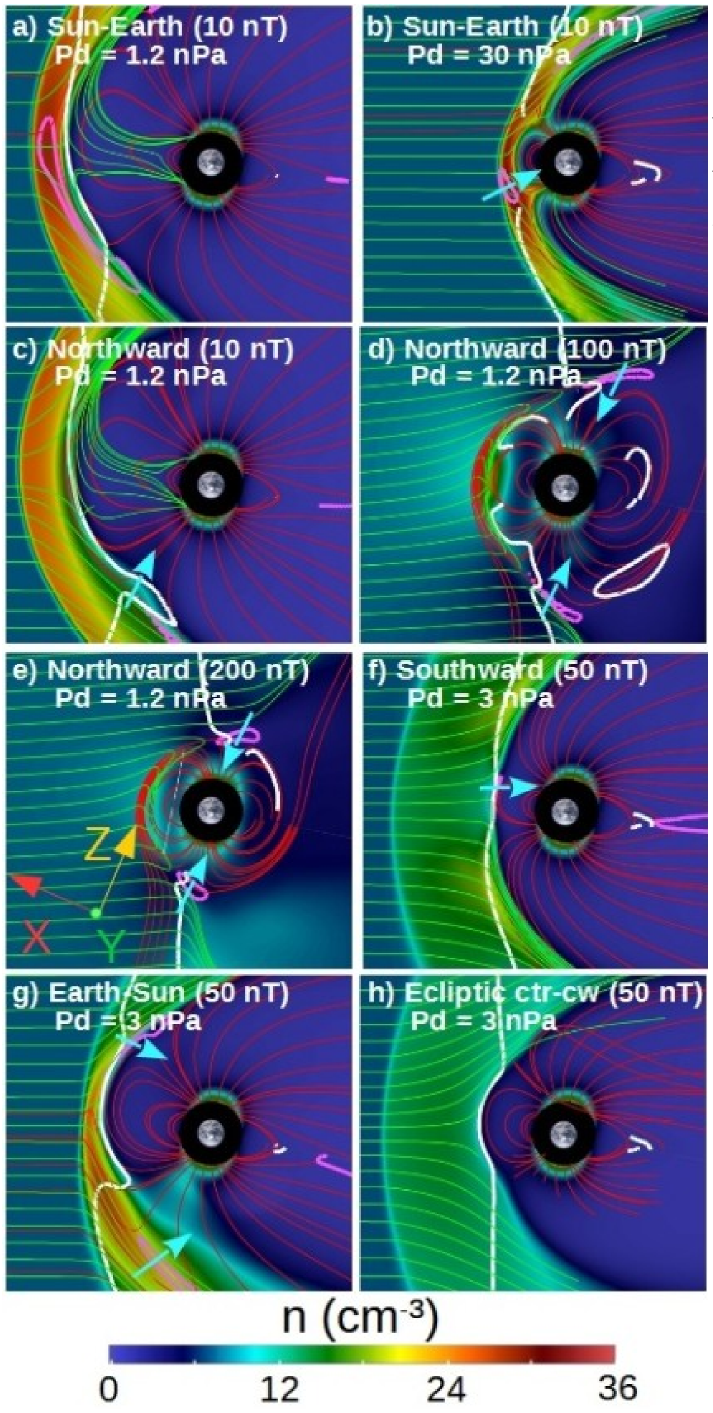


Fig. 1. 3D view of the system. Density distribution (color scale), field lines of the exoplanet magnetic field (red lines) and IMF (pink lines). The arrows indicate the orientation of the IMF (Northward orientation). Dashed white line shows the beginning of the simulation domain.

Numerical simulations of Sun-Earth magnetospheric interaction



Some results in the Sun-Earth simulations, where we can directly compare with measurements of the fields from orbiters.

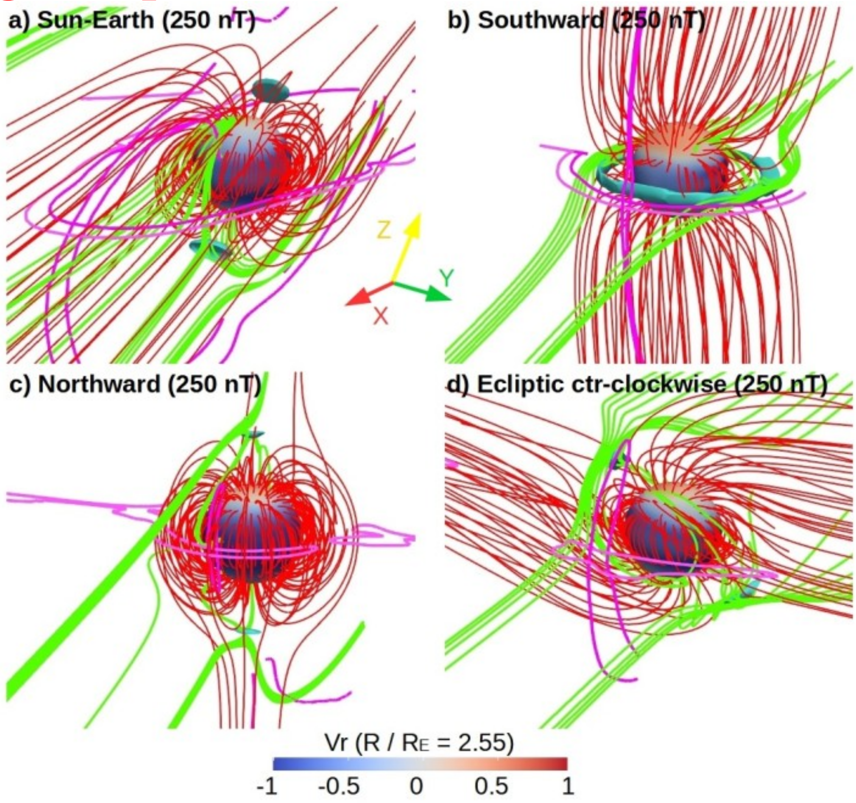


Fig. 3. 3D view of the Earth magnetosphere topology if $|B|_{\mathrm{IMF}} = 250\,\mathrm{nT}$ for (*a*) a Sun–Earth, (*b*) southward, (*c*) northward, and (*d*) ecliptic ctr-clockwise IMF orientations. We show the Earth magnetic field (red lines), SW stream functions (green lines), and isocontours of the plasma density for 6–9 cm^{-3}, indicating the location of the BS (pink lines). The cyan isocontours indicate the reconnection regions ($|B| = 60\,\mathrm{nT}$).

Fig. 2. Polar cut (*XY* plane) of the plasma density in simulations with (*a*) Sun–Earth IMF orientation $|B_{\mathrm{IMF}}| = 10\,\mathrm{nT}$ $P_{\mathrm{d}} = 1.2\,\mathrm{nPa}$, (*b*) Sun–Earth IMF orientation $|B_{\mathrm{IMF}}| = 10\,\mathrm{nT}$ $P_{\mathrm{d}} = 30\,\mathrm{nPa}$, (*c*) northward IMF orientation $|B_{\mathrm{IMF}}| = 10\,\mathrm{nT}$ $P_{\mathrm{d}} = 1.2\,\mathrm{nPa}$, (*d*) northward IMF orientation $|B_{\mathrm{IMF}}| = 100\,\mathrm{nT}$ $P_{\mathrm{d}} = 1.2\,\mathrm{nPa}$, (*e*) northward IMF orientation $|B_{\mathrm{IMF}}| = 200\,\mathrm{nT}$ $P_{\mathrm{d}} = 1.2\,\mathrm{nPa}$, (*f*) southward IMF orientation $|B_{\mathrm{IMF}}| = 50\,\mathrm{nT}$ $P_{\mathrm{d}} = 3\,\mathrm{nPa}$, (*g*) Earth–Sun IMF orientation $|B_{\mathrm{IMF}}| = 50\,\mathrm{nT}$ $P_{\mathrm{d}} = 3\,\mathrm{nPa}$, and (*h*) ecliptic ctr-cw IMF orientation $|B_{\mathrm{IMF}}| = 50\,\mathrm{nT}$ $P_{\mathrm{d}} = 3\,\mathrm{nPa}$. Earth magnetic field (red lines), SW stream functions (green lines), $|B| = 10\,\mathrm{nT}$ isocontour of the magnetic field (pink lines), and $v_{\mathrm{r}} = 0$ isocontours (white lines). The bold cyan arrows show the regions in which the plasma is injected into the inner magnetosphere.

Numerical simulations of Sun-Mercury interaction

Similar study was also done for Mercury, where we have a wealth of data from Mariner 10 mission, which measured the dipole moment, and later Messenger mission, which provided more precise measurements for the multipolar representation.

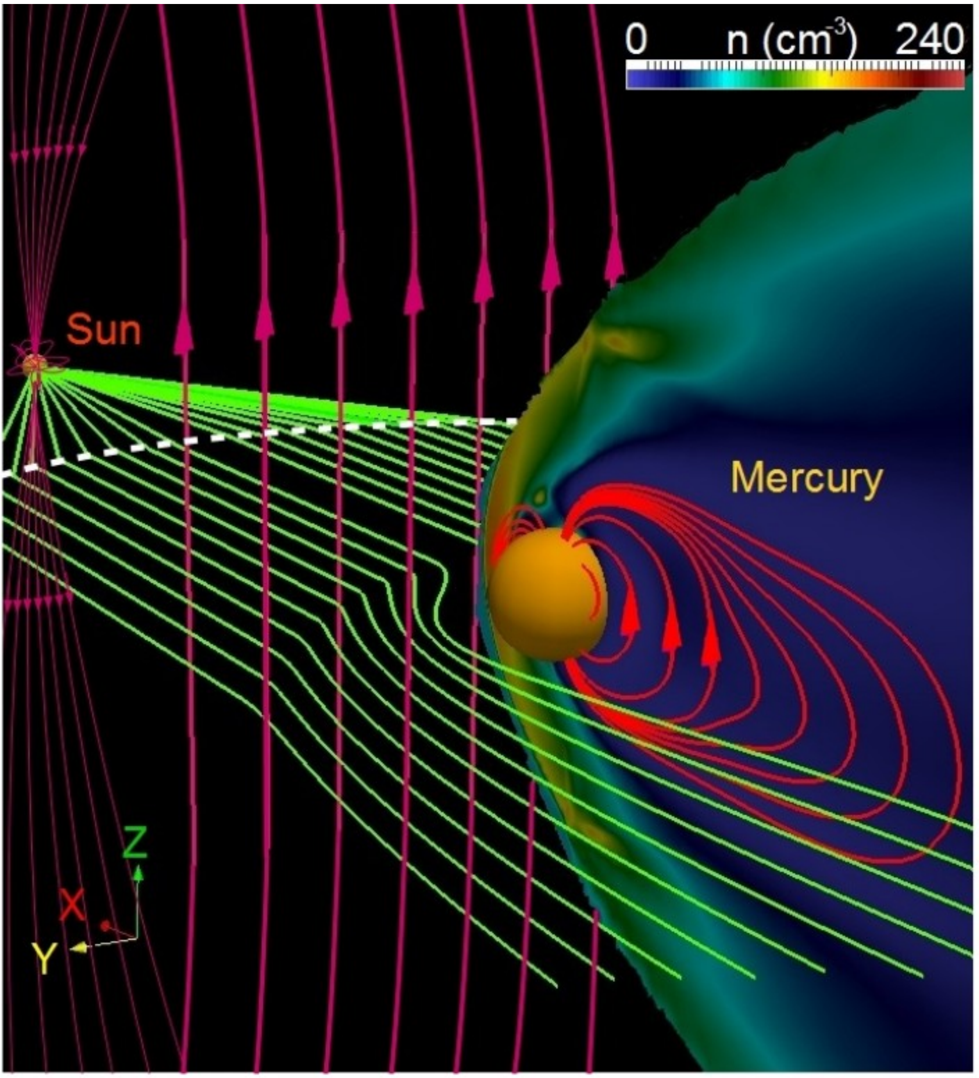


Fig. 1. 3D view of the system. Density distribution (color scale), field lines of the Hermean magnetic field (red lines), IMF (pink lines) and solar wind stream lines (green lines). The arrows indicate the orientation of the Hermean and interplanetary magnetic fields (case Bz). Dashed white line shows the beginning of the simulation domain.

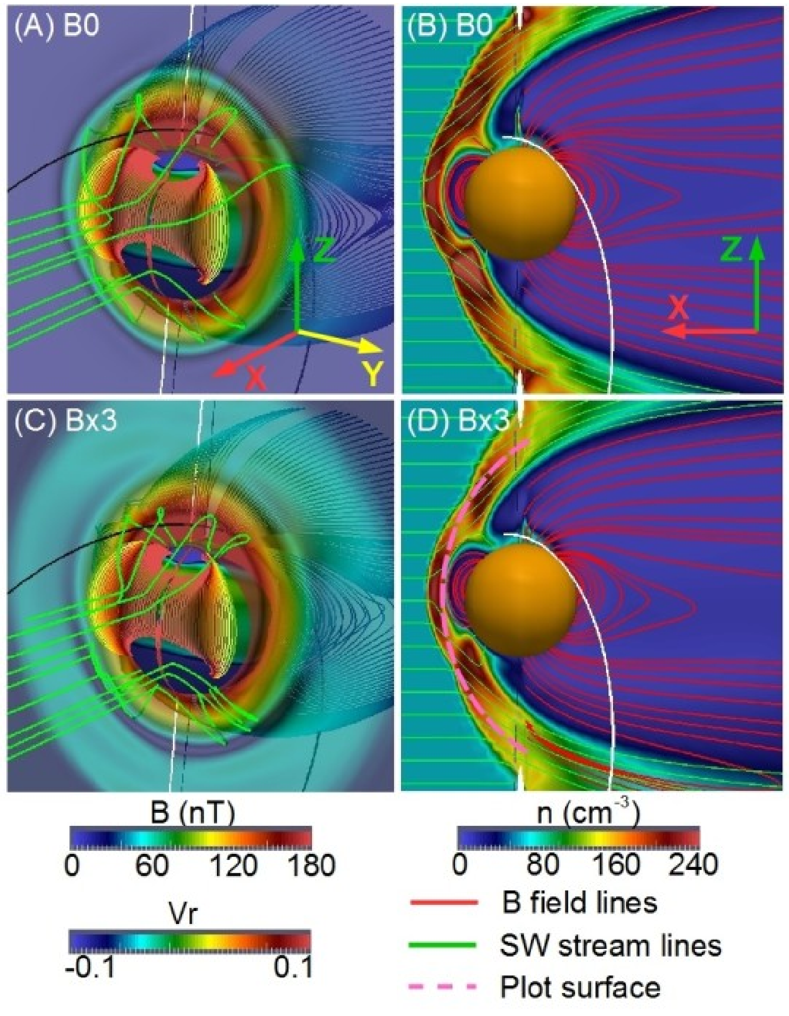


Fig. 2. Hermean magnetic field lines with the intensity imprinted on the field lines by a color scale for the reference case (A) and simulation Bx3 (C). Magnetic field intensity at the frontal plane $X = 0.3R_M$. SW stream lines (green). Inflow/outflow regions on the planet surface (blue/red). Polar plot of the density distribution (displaced $0.1R_M$ in Y direction) for the reference case (B) and simulation Bx3 (D). Dashed pink curve indicates the surface plotted in figures 3 and 4.

Planets around pulsars

•First exoplanets were found in orbit around a Galactic disk 6.2-ms pulsar PSR1257+12 (Wolszczan & Frail, 1992). PSR stands for "Pulsating Source of Radio" followed by the pulsar's right ascension and degrees of declination, B is added nowadays in the official name, to mark that coordinates are for the 1950.0 epoch, so official name is PSR B1257+12, or PSR J1300+1240 in epoch 2000, assigned with "J". Usually pulsars older than 1993 retain this epoch names, but since the newer epoch includes position precisely to minutes in the name, all pulsars have their J epoch names.

•The precise timing of millisecond pulsars was instrumental for the discovery.

•This pulsar rotates about 161 times per second, 9650 rpm, period $P = 6.219 \times 10^{-3}$ s and a period derivative of $\dot{P} = 1.2 \times 10^{-19}$. In a standard magnetic dipole spindown model this gives a dipole magnetic field of $B = 3 \times 10^{19} (P\dot{P})^{1/2}$ G $\approx 8.8 \times 10^{8}$ G and a characteristic age $\tau = P/(2\dot{P}) = 8 \times 10^{8}$ yrs. It is the fastest moving pulsar, with transverse velocity 326km/s and its surface is hot, 29 000 K. It is 2300 ly (710 pc) from us, in the constellation Virgo.

•The detected planets (B and C today) were reported to have masses of at least 4.3 and 3.9 Earth masses. Their respective distances from the pulsar are 0.36 AU and 0.46 AU, and they move in almost circular orbits with periods of 66.6 and 98 days. The third planet of 0.02 Earth masses (=double Moon mass), which is a planet A today, with period of 25 days, positioned closer, at 0.19 AU, was identified after additional analysis of the data (Wolszczan 1994) [Scherer et al. (1997) pointed out that its 25.3 day orbital period is close to the solar rotation period at the 17deg solar latitude of PSR 1257+12, and suggested that the modulation might actually be due to modulation in the electron density of the solar wind in that direction. But, such an effect was not observed in other millisecond pulsars, and also the oscillation amplitude does not depend on the radio frequency (Wolszczan et al. 2000b), which would follow for a plasma effect. So, it is rather still a planet]. Orbits for A, B and C are similarly inclined 50, 53, 47 degrees, respectively.

•The fourth possible planet in this system (Wolszczan 1996) was later dismissed (Wolszczan et al. 2000a).

•This were the first exoplanets, which was long anticipated, but nobody expected it around a pulsar! The first planet around a "normal" star was found only in 1995. It was the first "hot Jupiter", a large gaseous planet with a surprising period of 4.2 days, orbiting very closely the star 51 Pegasi.

•In 2015 naming of exoplanets was given to the public in NameExoWorlds campaign, and pulsar, which is an undead star, got the name of a Lich, undead character from fantasy fiction, known for controlling other undead creatures with magic. Planets are named Draugr, Poltergeist and Phobetor for planets A, B and C, respectively (by increasing distance), by Norse mythology undead, noisy ghosts of supernatural world, and a character from Ovid's Metamorphoses (one of the thousand sons of Somnus=Sleep who appears in dreams in the form of beasts).

Formation of planets around pulsars

The formation mechanisms of planets around pulsars can be divided into presupernova and postsupernova scenarios.

-**Presupernova** scenario includes formation of planets around an ordinary star and either surviving the evolution (and a series of catastrophic events along it) or being captured by a NS.

• In **postsupernova** cases, planets are either formed from the material around newly formed NS, or they are the last stage in the formation of some binary millisecond pulsars.

•For the rocky planets at circular orbits, a good possibility are mergers like WD+WD or WD+NS, or the remnant disk of the material from a Be star forming a binary with a NS. The first planets of Wolszczan best match the WD-WD merger scenario, so that planets are formed out of the debris of a merged companion star that used to orbit the pulsar when it was a white dwarf.

•Planets around pulsars seem to be rare, there are only few cases in about 3000 pulsars, all found by pulsar timing variations. Of more than 5000 currently known exoplanets, less than 10 around pulsars are confirmed

•A special feature because it *revived the field*: Interesting inconclusive one: (1982, 1994?) PSR B1937+21 close (few degrees) by the 1st discovered pulsar PSR B1919+21 (by Jocelyn Bell), this is the 1st discovered ms pulsar, 1.5ms (624 rotations in a second!), a companion of 0.001 M_Earth, like Ceres, at 2.7 AU, asteroid belt? More observations needed. Also points to a large precision of the method, when long observations available.

List of planets around pulsars

-(1993) PSR B1620−26 A + WD with one exoplanet (2.5+/- 1 M_Jupiter, orbiting them at 23 AU, period 36500 days~10 yrs, found from Doppler shifts it induced on the orbits of stars). It is in Scorpius, at a distance 12.4ly away, just outside the core of the globular cluster M4 which is 12.2 bln years old. Stars are hot: <30 000 K and <25 200 K. It is most probably a captured planet.

-(2006) 4U 0142+61, a magnetar (supernova about 100 000 yrs ago, 0.63 solar luminosities, rotates with 8.7s period) in Cassiopeia, at 13 000 ly from us, debris disk detected, at 1.6 mln km from the star, contains about 10 Earth masses of material, mostly heavier metals.

-(2011) The “diamond-planet” system PSR J1719–1438 is a millisecond pulsar surrounded by a Jupiter-mass companion ay least 23 times denser than water, thought to have formed via ablation (evaporation) of its donor star. It is a 27 000km radius 10^31 carats diamond crystal core remaining from the evaporated white dwarf, at 600 000 km from the star, has 2hr10’ rotation period. – of the similar kind is a Black Widow pulsar PSR B1957+20 (1988) in Sagitta constellation, with a period of 1.6ms and large mass, 1.6-2.4 M_Sun. It has a ~M_Jupiter companion, probably a brown dwarf, orbiting it with a period of 9.2hrs, making a 20min eclipses, through which the object was found.

-(1968 pulsar, 2013 asteroid?) PSR J0738−4042, encounter with an asteroid or in-falling debris from a disk. It is a bright, radio-emitting neutron star at a distance 37 000 ly in constellation Puppis, with rotational properties similar to the main population of middle aged, isolated, radio pulsars, P and Pdot 0.267 1/s (375ms) and -1/15e^-14 1/s^2, collected 24 yrs of data, so one can check the timing in detail.

-(1979 planets rejected, 2017 disputed) PSR B0329+54, 3 460 ly away in Camelopardalis, period 0.71452 s, 5 million yrs old. Remains the possibility of a long period planet.

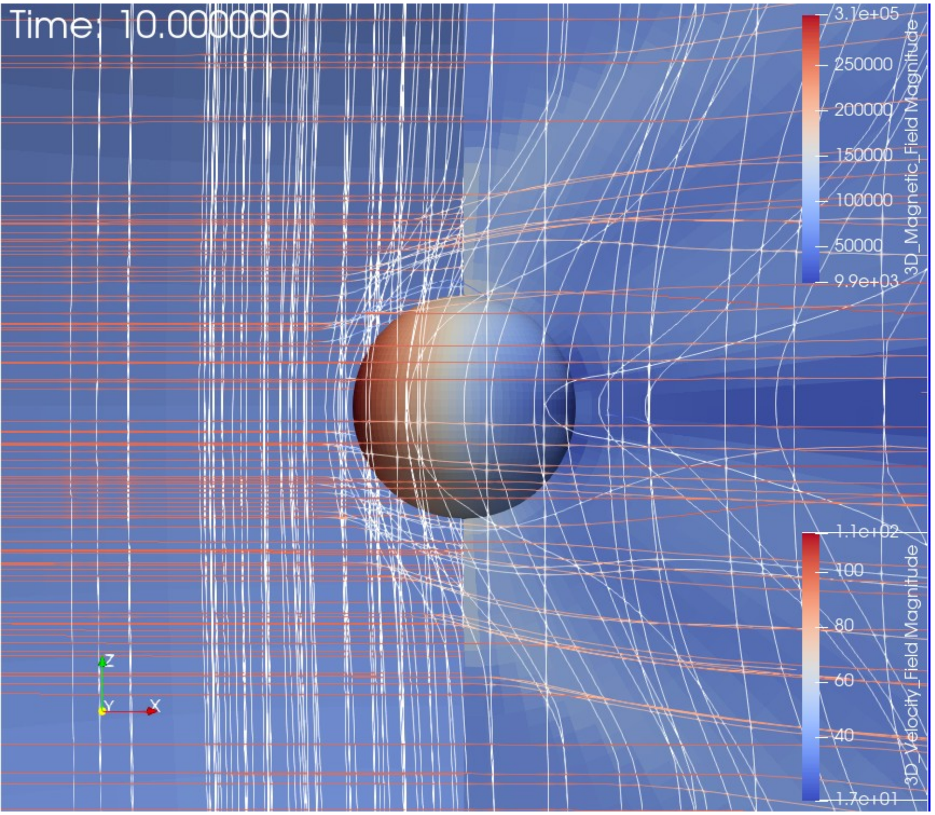
-(1968 Puschino pulsar, 2014 planets) PSR B0943+10 is an 5mln years old pulsar in Leo, 2 000 ly away, with period 1.1s. Two gas giant planets,masses 2.8 and 2.6M_Jup with 730 and 1460 days orbital period, 1/8 and 2.9 AU radius orbits, respectively. There are more tentative objects with planets of Jupiter mass, like low luminosity (2017) PSR J2322−2650, with planet of 0.8M_Jup in the orbit with 0.32d at 0.01 AU; (2022) PSR J2007-3120 with a 0.008 M_J planet with 723d period; (2020, FAST) confirmed in globular cluster M13, PSR J1641+3627F, 3ms pulsar with a 0.16M_Sun mass companion, probably a WD, not a planet; the binary millisecond pulsar (2021, FAST) PSR J1641+3627E (also M13E) is a black widow with a companion mass around 19.42 M_J, 0.11d period; (2013) PSR J1544+493 eclipsing black widow 2.16 ms pulsar with a close companion of 18M_Jup at 2.19h orbit; (2016) PSR J0636+5129 with a 8M_J companion in 96min orbit; (1996, 2001?) PSR J2051−0827, 28.3M_Jup at 0.1d period orbit, (2000) PSR J1807-2459, 9.4M_J, 0.07days period.

Numerical simulations with the NS-planet interaction

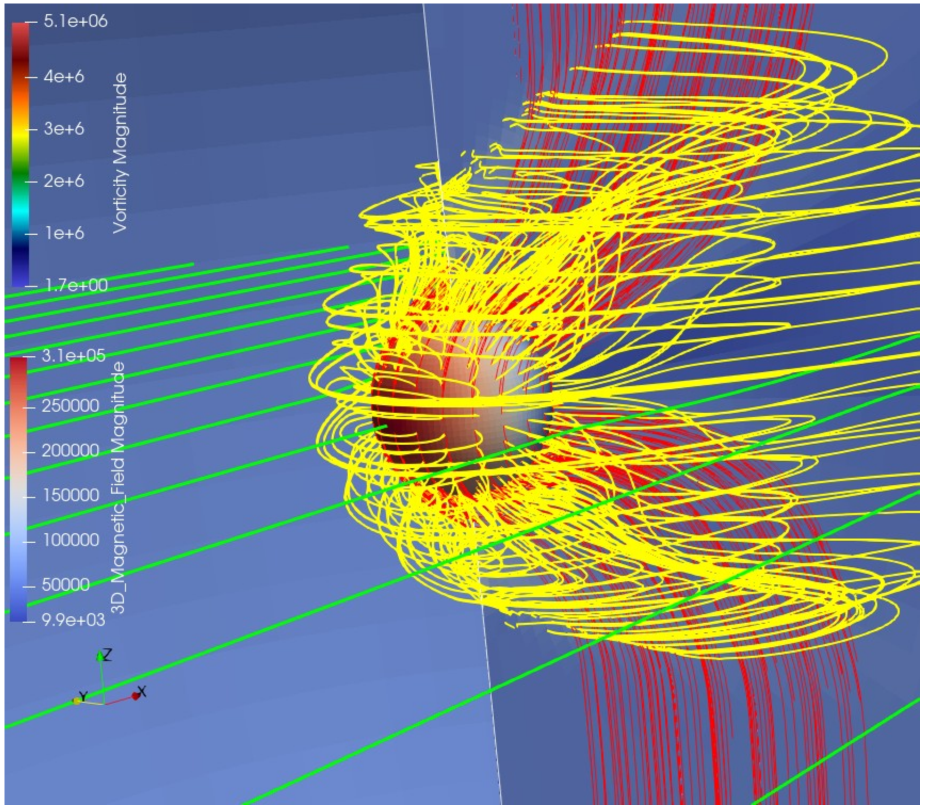
I show preliminary results in our simulations with NS parameters. We are increasing the stellar magnetic field in the simulations-to accommodate for the large field we increase the density of the interplanetary medium, local magnetic field strength near the planet and stellar wind velocity. We probe for the different planetary surface boundary conditions (conducting, ferromagnetic) - this is potentially interesting for the planetary study: planets around NS could have some extreme physical properties.

Conducting planet (B_planet=0):

Bsw=3.0, Vsw=1.e9



Currents (yellow), Vsw(green), mag.field (red)

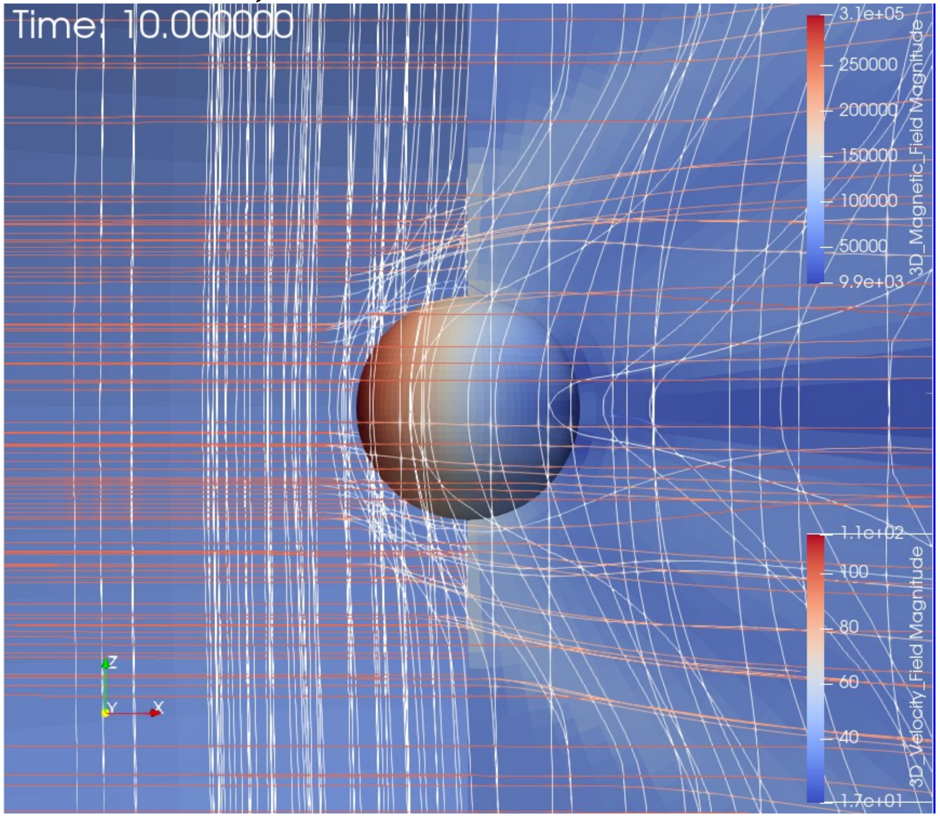


For the conducting planet atmosphere case, electric current loops remain close to the planet surface.

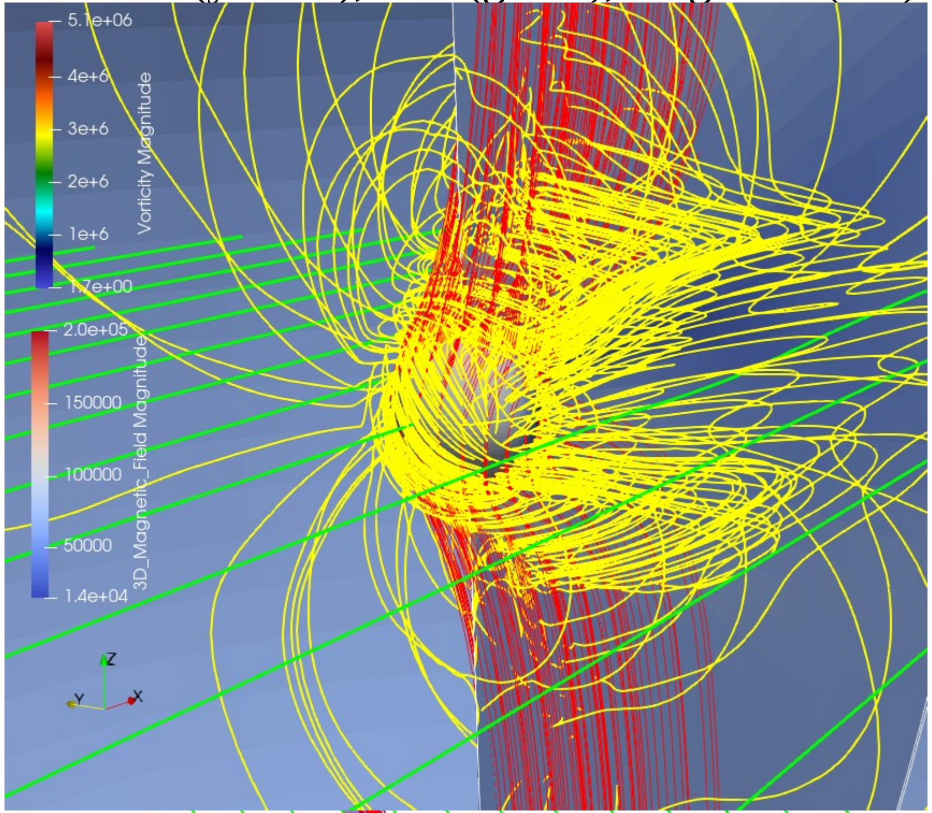
Numerical simulations with the NS-planet interaction

In the case of feromagnetic planet surface, results are different, currents point to an extended dipolar *electric field* structure. Work is in progress to understand the possible auroral effects.

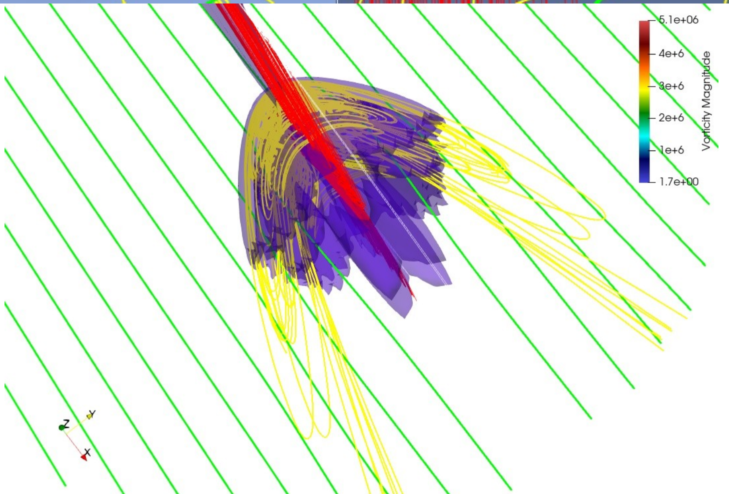
Bsw=3.0, Vsw=1.e9



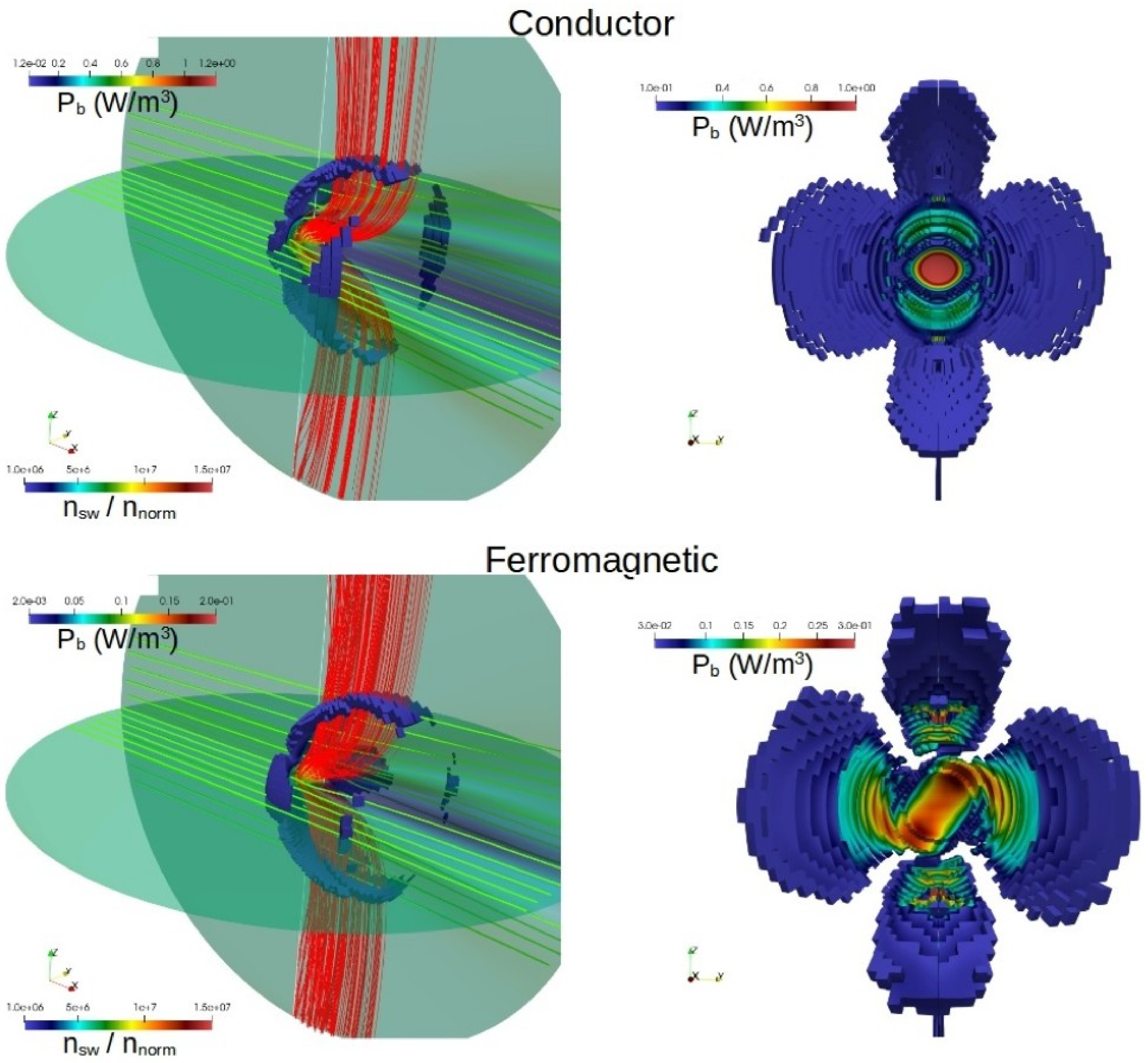
Currents (yellow), Vsw(green), mag.field (red)



Equatorial view-Alfven "wings":



Radio emission from non-magnetic planets


Conductor
1.2e-02 0.2 0.4 0.6 0.8 1 1.2e+00
P_b (W/m³)
1.0e+06 5e+6 1e+7 1.5e+07
n_{sw} / n_{norm}
1.0e-01 0.4 0.6 0.8 1.0e+00
P_b (W/m³)
Ferromagnetic
2.0e-03 0.05 0.1 0.15 2.0e-01
P_b (W/m³)
1.0e+06 5e+6 1e+7 1.5e+07
n_{sw} / n_{norm}
3.0e-02 0.1 0.15 0.2 0.25 3.0e-01
P_b (W/m³)

Left panels: Iso-volume of Poynting flux divergence in cases with non-magnetic planet. Red lines are the magnetic field lines and green lines are the velocity streamlines of stellar wind. *Right panels:* Mag. power in the same cases. A surface with the maximum radiated power is located in the nose of the bow shock, because of bending and compression of inter-planetary magnetic field.

- El.mag. emission is 100 million times more intense than in the Sun-Earth case.
- We suggest that it could be observable even with the current instruments.

Summary on auroras

- Auroras are present in almost all planets in the Solar system.
- We have a tool to model star-planet magnetospheric interaction.
- Planets around pulsars are not very often, <0.5%, with a variety of possible kinds of evolution. I give an overview for a better idea of current status.
- “Usual” rocky planets were the 1st to be observed. Their evolution could be quite normal.
- We try our tool for aurora on the pulsar planets.
- Measuring the radio emission from pulsar planets would give us an additional window to study of pulsar wind.
- Radio emission from pulsar planets could be visible even with current instruments.

PLUTO lectures outline, Part I

- “Simulations” ver. “computations”.
- PLUTO physics modules.
- Documentation, test problems, templates.
- Practical part: installing the code
- Linux primer, Linux shell variables
- Testing with the use of 1D Sod shock tube test provided with the code.
- Visualization with gnuplot.

Simulation ver. computation

- Computation is actually an evaluation: you plug the numbers in a known algorithm and obtain the result. For this, you need to know the analytical expression. You still can use calculator or a computer to actually obtain numbers and plot e.g. trajectory of a bullet, but you **do** know the equation for the solution.

- Simulation is when you do not know the equation for the solution. You set the governing equations, e.g. differential equations, and use a numerical method to find the solutions. This is usually done in time steps, and we obtain the solution to some precision. We do not know the analytical expression for the solution, we just have numbers=numerical simulation.

- Why PLUTO? I used other codes, but PLUTO is constantly evolving (and is used in variety of problems) **and** the development is followed in the manual-which is not so often the case in the coding world. It gives chance to students to go through the-still steep-learning curve in the shortest possible time. Check also Doxygen html files in PLUTO/Doc .

PLUTO physics modules, equations

6.1 The HD Module

The HD module may be used to solve the Euler or the Navier-Stokes equations of classical fluid dynamics. The relevant source files and definitions for this module can be found in the Src/HD directory.

With the HD module, **PLUTO** evolves in time following system of conservation laws:

$$\frac{\partial}{\partial t}\begin{pmatrix} \rho \\ \boldsymbol{m} \\ E_t + \rho\Phi \end{pmatrix} + \nabla\cdot\begin{pmatrix} \rho\boldsymbol{v} \\ \boldsymbol{m}\boldsymbol{v} + p\mathsf{I} \\ (E_t + p + \rho\Phi)\boldsymbol{v} \end{pmatrix}^T = \begin{pmatrix} 0 \\ -\rho\nabla\Phi + \rho\boldsymbol{g} \\ \boldsymbol{m}\cdot\boldsymbol{g} \end{pmatrix} \tag{6.1}$$

where ρ is the mass density, $\boldsymbol{m} = \rho\boldsymbol{v}$ is the momentum density, $\boldsymbol{v}$ is the velocity, p is the thermal pressure and E_t is the total energy density:

$$E_t = \rho e + \frac{\boldsymbol{m}^2}{2\rho}\,. \tag{6.2}$$

An equation of state provides the closure $\rho e = \rho e(p, \rho)$.

The source term on the right includes contributions from body forces and is written in terms of the (time-independent) gravitational potential Φ and and the acceleration vector $\boldsymbol{g}$ (§5.4).

PLUTO physics modules, equations

6.2 The MHD Module

The MHD module is suitable for the solution of ideal or resistive (non-relativistic) magnetohydrodynamical equations. Source and definition files are located inside the Src/MHD directory.

With the MHD module, **PLUTO** solves the following system of conservation laws:

$$\begin{array}{lcll}
\dfrac{\partial\rho}{\partial t} & + & \nabla\cdot(\rho\boldsymbol{v}) & = & 0 \\
\dfrac{\partial\boldsymbol{m}}{\partial t} & + & \nabla\cdot\left[\boldsymbol{m}\boldsymbol{v}-\boldsymbol{B}\boldsymbol{B}+\mathsf{I}\left(p+\dfrac{\boldsymbol{B}^2}{2}\right)\right]^T & = & -\rho\nabla\Phi+\rho\boldsymbol{g} \\
\dfrac{\partial\boldsymbol{B}}{\partial t} & + & \nabla\times(c\boldsymbol{E}) & = & 0 \\
\dfrac{\partial(E_t+\rho\Phi)}{\partial t} & + & \nabla\cdot\left[\left(\dfrac{\rho\boldsymbol{v}^2}{2}+\rho e+p+\rho\Phi\right)\boldsymbol{v}+c\boldsymbol{E}\times\boldsymbol{B}\right] & = & \boldsymbol{m}\cdot\boldsymbol{g}
\end{array} \quad (6.4)$$

where ρ is the mass density, $\boldsymbol{m}=\rho\boldsymbol{v}$ is the momentum density, $\boldsymbol{v}$ is the velocity, p is the gas (thermal) pressure, $\boldsymbol{B}$ is the magnetic field[2] and E_t is the total energy density:

$$E_t=\rho e+\frac{\boldsymbol{m}^2}{2\rho}+\frac{\boldsymbol{B}^2}{2}\,. \quad (6.5)$$

where an additional equation of state provides the closure $\rho e=\rho e(p,\rho)$ (see Chapter 7). The source term on the right includes contributions from body forces and is written in terms of the (time-independent) gravitational potential Φ and and the acceleration vector $\boldsymbol{g}$ (see §5.4).

In the third of Eq. (6.4), $\boldsymbol{E}$ is the electric field defined by the expression

$$c\boldsymbol{E}=-\boldsymbol{v}\times\boldsymbol{B}+\frac{\eta}{c}\cdot\boldsymbol{J}+\frac{\boldsymbol{J}}{ne}\times\boldsymbol{B} \qquad \left(\boldsymbol{J}=c\nabla\times\boldsymbol{B}\right) \quad (6.6)$$

where the first term is the convective term, the second term is the resistive term (η denotes the resistivity tensor, see §8.2) while the third term is the Hall term (§8.1). Note that the speed of light c never enters

PLUTO physics modules, equations

6.3 The RHD Module

The RHD module implements the equations of special relativistic fluid dynamics in 1, 2 or 3 dimensions. Velocities are always assumed to be expressed in units of the speed of light. The special relativistic module comes with 2 different equations of state, and it also works in curvilinear coordinates. Gravity in Newtonian approximation can also be incorporated. The relevant source files and definitions for this module can be found in the Src/RHD directory.

The special relativistic module evolves the conservative set $\boldsymbol{U}$ of state variables

$$\boldsymbol{U} = \Big(D,\ m_1,\ m_2,\ m_3,\ E_t\Big)^T$$

where D is the laboratory density, $m_{x1,x2,x3}$ are the momentum components, E_t is the total energy (including contribution from the rest mass). The evolutionary conservative equations are

$$\frac{\partial}{\partial t}\left(\begin{array}{c} D \\ \boldsymbol{m} \\ E_t \end{array}\right) + \nabla\cdot\left(\begin{array}{c} D\boldsymbol{v} \\ \boldsymbol{m}\boldsymbol{v} + p\mathsf{I} \\ \boldsymbol{m} \end{array}\right)^T = \left(\begin{array}{c} 0 \\ \boldsymbol{f}_g \\ \boldsymbol{v}\cdot\boldsymbol{f}_g \end{array}\right)$$

where $\boldsymbol{v}$ is the velocity, p is the thermal pressure. Primitive variables $\boldsymbol{V}$ always include the rest-mass density ρ, three-velocity $\boldsymbol{v} = (v_{x1}, v_{x2}, v_{x3})$ and pressure p. With **PLUTO** 4.4, the acceleration term $\boldsymbol{f}_g$ is treated consistently[3] with the formalism of [Tau48]. If $\boldsymbol{a}$ is the acceleration vector,

$$\boldsymbol{f}_g = \rho\gamma^2\left[\gamma^2\boldsymbol{v}\left(\boldsymbol{v}\cdot\boldsymbol{a}\right) + \boldsymbol{a}\right] . \qquad (6.12)$$

PLUTO physics modules, equations

6.4 The RMHD Module

The RMHD module implements the equations of (ideal) special relativistic magnetohydrodynamics in 1, 2 or 3 dimensions. Velocities are always assumed to be expressed in units of the speed of light. Source and definition files are located inside the Src/RMHD directory.

The RMHD module solves the following system of conservation laws:

$$\frac{\partial}{\partial t}\begin{pmatrix} D \\ \boldsymbol{m} \\ E_t \\ \boldsymbol{B} \end{pmatrix} + \nabla\cdot\begin{pmatrix} D\boldsymbol{v} \\ w_t\gamma^2\boldsymbol{v}\boldsymbol{v} - \boldsymbol{b}\boldsymbol{b} + \mathsf{I}p_t \\ \boldsymbol{m} \\ \boldsymbol{v}\boldsymbol{B} - \boldsymbol{B}\boldsymbol{v} \end{pmatrix}^T = \begin{pmatrix} 0 \\ \boldsymbol{f}_g \\ \boldsymbol{v}\cdot\boldsymbol{f}_g \\ \boldsymbol{0} \end{pmatrix} \tag{6.13}$$

where D is the laboratory density, $\boldsymbol{m}$ is the momentum density, E is the total energy (including contribution from the rest mass) while $\boldsymbol{f}_g$ is an acceleration term (see 6.3).

Primitive variables are similar to the RHD module but they also contain the magnetic field, $\boldsymbol{V} = (\rho, \boldsymbol{v}, p, \boldsymbol{B})$. The relation between $\boldsymbol{V}$ and $\boldsymbol{U}$ is

$$\begin{aligned} D &= \gamma\rho \\ \boldsymbol{m} &= w_t\gamma^2\boldsymbol{v} - b^0\boldsymbol{b} \\ E_t &= w_t\gamma^2 - b^0b^0 - p_t \end{aligned} \quad , \quad \left\{\begin{aligned} b^0 &= \gamma\boldsymbol{v}\cdot\boldsymbol{B} \\ \boldsymbol{b} &= \boldsymbol{B}/\gamma + \gamma(\boldsymbol{v}\cdot\boldsymbol{B})\boldsymbol{v} \\ w_t &= \rho h + \boldsymbol{B}^2/\gamma^2 + (\boldsymbol{v}\cdot\boldsymbol{B})^2 \\ p_t &= p + \frac{\boldsymbol{B}^2/\gamma^2 + (\boldsymbol{v}\cdot\boldsymbol{B})^2}{2} \end{aligned}\right.$$

A note on public vs. non-public modules.

Besides the official code release, a few modules have not yet been made available with the standard public version, as they are still under active development or testing stage. In other circumstances, a private module may have been implemented under specific collaboration policies, which do not grant its public distribution. These non-offical modules include:

- Lagrangian Particle Module
  (Developers: B. Vaidya [*bvaidya@iiti.ac.in*], D. Mukherjee [*dipanjan@iucaa.in*]);
- Dust Module
  (Developers: A. Mignone [*mignone@to.infn.it*], M. Flock [*flock@mpia.de*]);
- Relativistic Resistive MHD
  (Developers: A. Mignone [*mignone@to.infn.it*], G. Mattia [*mattia@mpia.de*]);

Distribution, private sharing and usage of these modules is permitted only in the form of a collaboration between our partner institutions network, requiring co-authorship from at least one of the module developers.

PLUTO physics modules, equations

6.5 The Resistive RMHD (*ResRMHD*) Module

Note: This module is not part of the public code release, see "Terms & Conditions of Use" at the beginning of this guide

The ResRMHD module deals with the non-ideal relativistic MHD equations using the approaches discussed in [MMBD19]. Source and definition files are located inside the Src/ResRMHD directory.

The set of resistive relativistic equations arising from the time and space split of the covariant are, in vectorial form,

$$\begin{aligned}
&\frac{\partial D}{\partial t} + \boldsymbol{\nabla}\cdot(D\boldsymbol{v}) = 0, \\
&\frac{\partial \boldsymbol{m}}{\partial t} + \boldsymbol{\nabla}\cdot(w\boldsymbol{u}\boldsymbol{u} + p\mathsf{I} + \mathsf{T}) = 0, \\
&\frac{\partial \mathcal{E}}{\partial t} + \boldsymbol{\nabla}\cdot\boldsymbol{m} = 0, \\
&\frac{\partial \boldsymbol{B}}{\partial t} + \boldsymbol{\nabla}\times\boldsymbol{E} = 0, \\
&\frac{\partial \boldsymbol{E}}{\partial t} - \boldsymbol{\nabla}\times\boldsymbol{B} = -\boldsymbol{J},
\end{aligned} \tag{6.14}$$

where I is the identity matrix and the fluid conserved variables are the density $D = \rho\gamma$ as measured in the laboratory frame, the total momentum density $\boldsymbol{m} = w\gamma\boldsymbol{u} + \boldsymbol{E}\times\boldsymbol{B}$, and the total energy density

$$\mathcal{E} = w\gamma^2 - p + \mathcal{P}_{\rm EM}\,. \tag{6.15}$$

In the expressions above, $w = \varepsilon + p$ is the specific enthalpy and $\mathcal{P}_{\rm EM} = (E^2 + B^2)/2$ denotes the EM energy density. Finally,

$$\mathsf{T} = -\boldsymbol{E}\boldsymbol{E} - \boldsymbol{B}\boldsymbol{B} + \tfrac{1}{2}(E^2 + B^2)\mathsf{I} \tag{6.16}$$

PLUTO physics modules, equations

7. Equation of State

In the current implementation, **PLUTO** describes a thermally ideal gas obeying the *thermal* Equation of State (EOS)

$$p = nk_BT = \frac{\rho}{m_u\mu}k_BT \tag{7.1}$$

where p is the pressure, n is the total particle number density, k_B is the Boltzmann constant, T is the temperature, ρ is the density, m_u is the atomic mass unit and μ is the mean molecular weight. The thermal EOS describes the thermodynamic state of a plasma in terms of its pressure p, density ρ, temperature T and chemical composition μ. Eq. (7.1) is written in CGS physical units. Using code units for p and ρ while leaving temperature in Kelvin, the thermal EOS is conveniently re-expressed as

$$p = \frac{\rho T}{\mathcal{K}\mu} \qquad \Longleftrightarrow \qquad T = \frac{p}{\rho}\mathcal{K}\mu \qquad \left(\text{where} \quad \mathcal{K} = \frac{m_u v_0^2}{k_B}\right) \tag{7.2}$$

where $\mathcal{K}$ is the **KELVIN** macro which depends explictly on the value of `UNIT_VELOCITY`.

PLUTO physics modules, equations

8. Nonideal Effects

In this chapter we give an overview of the code capabilities for treating dissipative (or diffusion) terms which, at present, include

- Hall MHD (MHD), described in §8.1;
- Resistivity (MHD), described in §8.2;
- Thermal conduction (HD, MHD), described in §8.3.
- Viscosity (HD, MHD), described in §8.4;

Each modules can be individually turned on from the physics sub-menus accessible via the Python script.

PLUTO documentation, templates and test examples

- PLUTO is freely available. Source files of the code are downloadable from http://plutocode.ph.unito.it/
- After unpacking, the source code is in PLUTO directory. In PLUTO/Doc is the manual, userguide.pdf. Follow the “Quick start” at the beginning of the document to **install and test the code. Produce the gnuplot output specified in the manual, to verify if the setup works.**
- The code comes with templates of subroutines which are usually changed. They are given in the PLUTO/Src/Templates
- Test examples, under /PLUTO/Test_Problems, are basic versions of setups used in previous versions of the code, during the developments of the setups for simulations presented In publications by various groups. It is a good library of examples for faster start of one’s own project.
- I will provide the setup for HD and MHD accretion disk for this lectures.

Linux shell variable setup, aliases

- We follow the “Quick start” from the beginning of the PLUTO/Doc/userguide.pdf

- Instead of every time after login repeating
  export PLUTO_DIR=/home/user/PLUTO # in bash shell
  Create in .bash_aliases (or .bashrc) a shell variable with path to PLUTO:
  export PLUTO_DIR="/home/miki/PLUTO"

- Also, it is useful to create an alias (shortcut) for running the setup.py, e.g.:
  alias pls='python $PLUTO_DIR/setup.py'

- For runs on multiple processors on laptop, useful is alias like:
  alias plutorun6='mpirun -np 6 ./pluto'

Test problem: Sod shock tube in 1D-use #04 if problems with #01

0.2 Running a simple shock-tube problem

PLUTO can be quickly configured to run one of the several test problems provided with the distribution. Assuming that your system satisfies all the requirements described in the next chapter (i.e. C compiler, Python, etc..) you can quickly setup **PLUTO** in the following way:

1. Change directory to any of the test problems under PLUTO/Test_Problems, e.g.

   ```
   ~> cd $PLUTO_DIR/Test_Problems/HD/Sod
   ```

2. Copy the header and initialization files from a configuration of our choice (e.g. #01):

   ```
   ~/PLUTO/Test_Problems/HD/Sod> cp definitions_01.h definitions.h
   ~/PLUTO/Test_Problems/HD/Sod> cp pluto_01.ini pluto.ini
   ```

3. Run the Python script using

   ```
   ~/PLUTO/Test_Problems/HD/Sod> python $PLUTO_DIR/setup.py
   ```

   and select "Setup problem" from the main menu, see Fig. 1.2. You can choose (by pressing Enter) or modify the default setting using the arrow keys.

4. Once you return to the main menu, select "Change makefile", choose a suitable makefile (e.g. Linux.gcc.defs) and press enter.

   All the information relevant to the specific problem should now be stored in the four files init.c (assigns initial condition and user-supplied boundary conditions), pluto.ini (sets the number of grid zones, Riemann solver, output frequency, etc.), definitions.h (specifies the geometry, number of dimensions, reconstruction, time stepping scheme, and so forth) and the makefile.

5. Exit from the main menu ("Quit" or press 'q') and type

   ```
   ~/PLUTO/Test_Problems/HD/Sod> make
   ```

   to compile the code.

6. You can now run the code by typing

   ```
   ~/PLUTO/Test_Problems/HD/Sod> ./pluto
   ```

   At this point, **PLUTO** reads the initialization file pluto.ini and starts integrating. The run should take a few seconds (or less) and the integration log should be dumped to screen.

Files for setup and their modifications

- The PLUTO code structure ensures that original source will not be changed, for our simulations we just append to it our version of some files in a pre-compiling step.
- The changed version of a file in the work directory (for which I suggest the name /home/Pluto) has, by default in PLUTO, priority to the original version in PLUTO/Src directory.

- The files to be copied into the work directory from the working version into a new setup are **init.c, pluto.ini, definitions.h**, (+userdef_output.c, res_eta.c, visc_nu.c for the accretion disk setup).

- The file definitions.h is the only *.h file changed in the work directory. All the other *.h files are to be changed directly in PLUTO/Src directory.

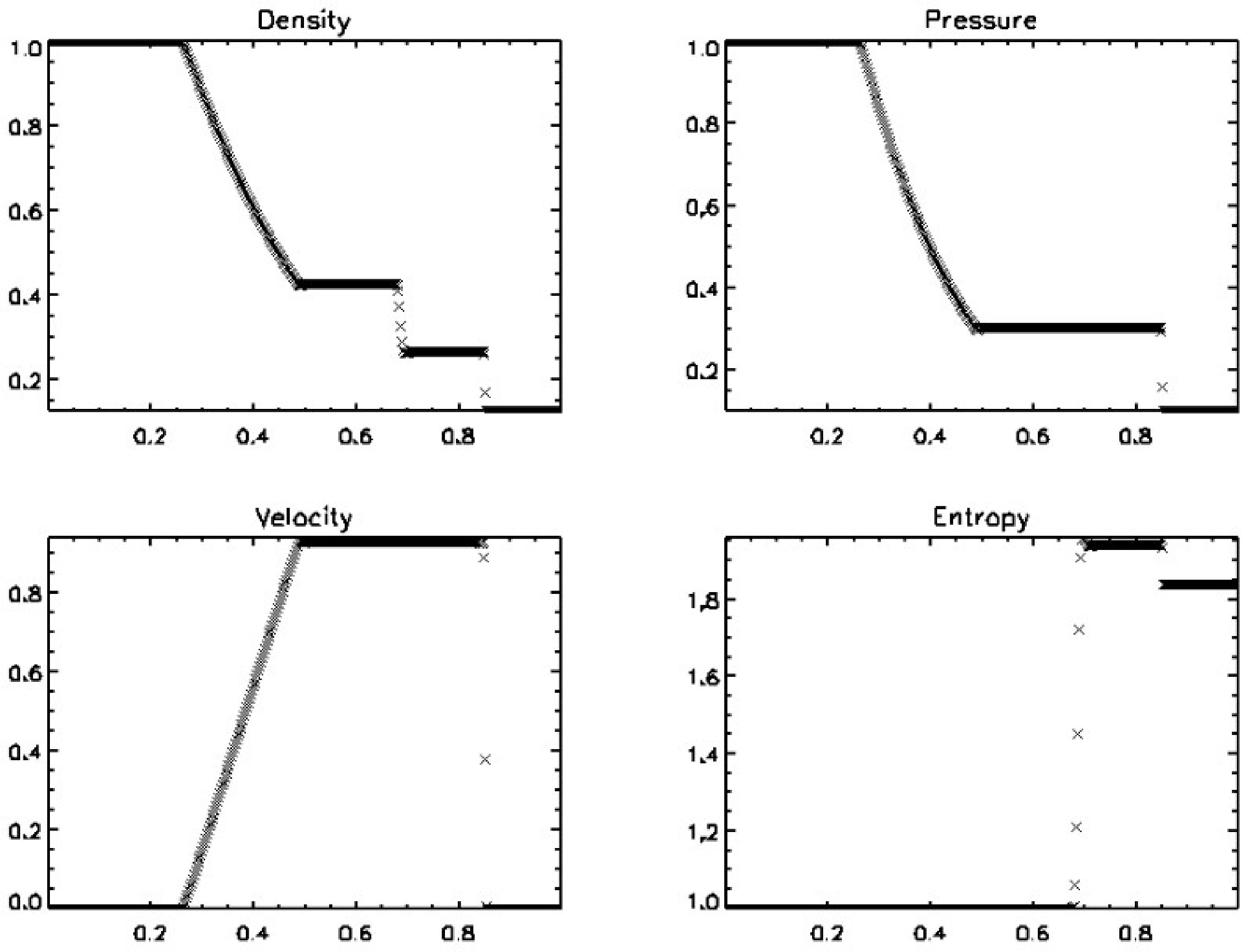
- In the file **pluto.ini** are defined the grid, solvers and run parameters.
- Most of the entries in **definitions.h** are done through the python environment during the setup of the run, but some entries are to be done by hand editing the definitions.h file, prior to compilation.
- In the file **init.c** is defined the physics setup.

Test problem: Sod shock tube in 1D

Detailed Description **from Doxygen file:** The Sod shock tube problem is one of the most used benchmark for shock-capturing schemes. It is a one-dimensional problem with initial condition given by a discontinuity separating two constant states:

$$\begin{aligned}(\rho,\, v_x,\, p)_L &= (1, 0, 1) && \text{for} \quad x < 0.5\\ (\rho,\, v_x,\, p)_R &= \left(\tfrac{1}{8}, 0, \tfrac{1}{10}\right) && \text{for} \quad x > 0.5\end{aligned}$$

The evolved structured at t=0.2 is shown in the panels below and consists of a left-going rarefaction wave, a right-going contact discontinutity and a right-going shock wave. The results shown here were carried with PARABOLIC interpolation, CHARACTERISIC_TRACING time stepping and the two_shock Riemann solver on 400 zones (**configuration #04**).

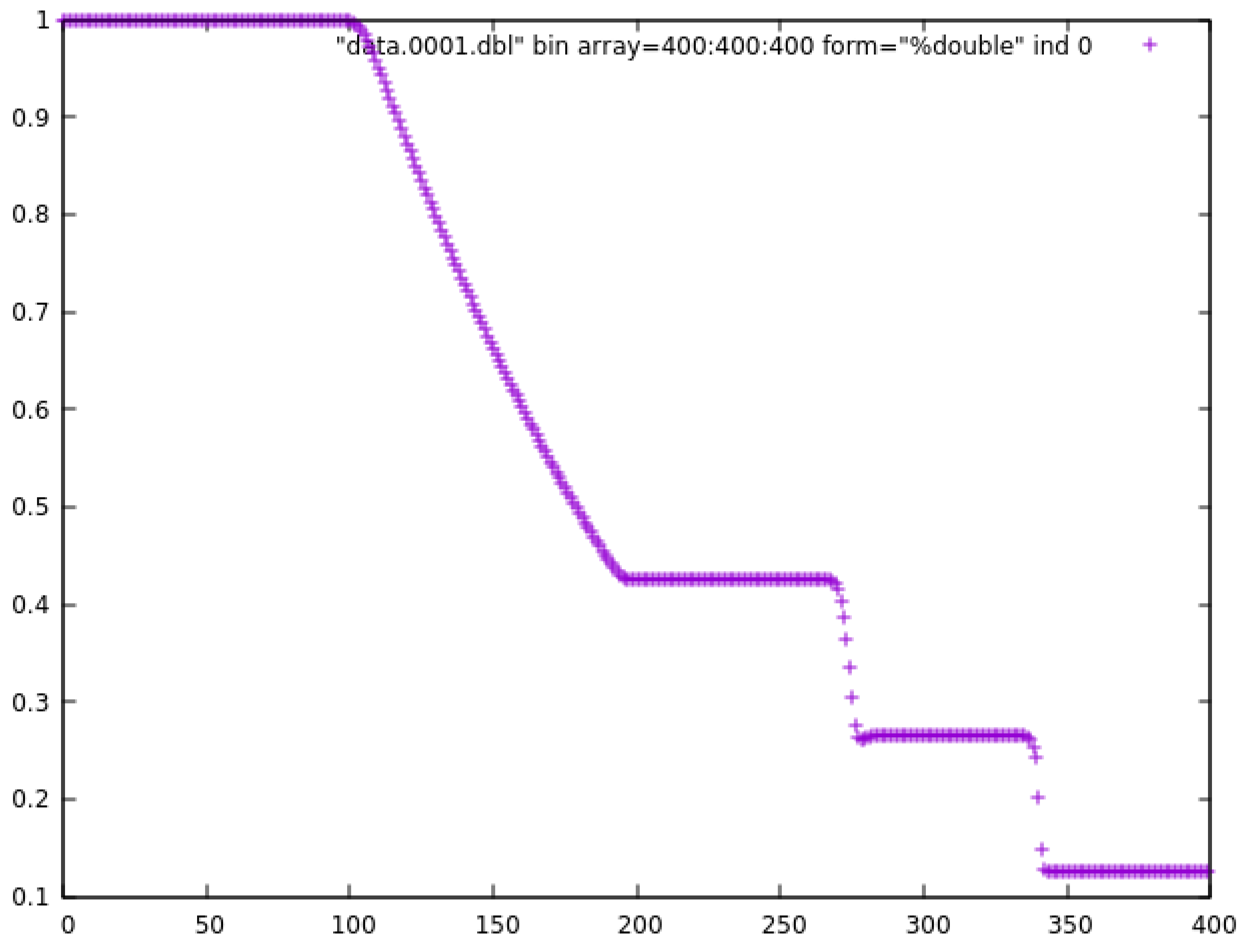


Test problem: Sod shock tube in 1D

Typing in terminal:

```
gnuplot> plot "data.0001.dbl" bin array=400:400:400 form="%double" ind 0
```

You should obtain:



Summary of the PLUTO lectures Part I

- We got acquainted with PLUTO and its physics modules.
- I show where to find documentation, test problems and templates.
- Practical part: we went through the code installation.
- We went through the needed essentials of Linux
- We tested the setup using the 1D Sod shock tube test provided with the code.
- We used visualization of the results with gnuplot.

Outline, Part II

- Initial and boundary conditions setup for 2D Orszag-Tang test.
- Hands-on introduction to setup files of PLUTO.
- Output formats & files which we should save for later analysis.
- Visualization of the results with Paraview.
- Movie time: animation and saving of the movie.

2D test problem: Orszag-Tang

0.3 Running the Orszag-Tang MHD vortex test

1. Change directory to PLUTO/Test_Problems/MHD/Orszag_Tang.
2. Choose a configuration (e.g. #02) and copy the corresponding configuration files, i.e.,

   ```
   ~/PLUTO/Test_Problems/MHD/Orszag_Tang> cp definitions_02.h definitions.h
   ~/PLUTO/Test_Problems/MHD/Orszag_Tang> cp pluto_02.ini pluto.ini
   ```

3. Run the Python script:

   ```
   ~/PLUTO/Test_Problems/MHD/Orszag_Tang> python $PLUTO_DIR/setup.py
   ```

   select "Setup problem" and choose the default setting by pressing enter;
4. Once you return to the main menu, select "Change makefile" and choose a suitable makefile (e.g. Linux.gcc.defs) and press enter.
5. Exit from the main menu ("Quit" or press 'q'). Edit pluto.ini and, under the *[Grid]* block, lower the resolution from 512 to 200 in both directions (`X1-grid` and `X2-grid`). Change `single_file`, in the "dbl" output under the *[Static Grid Output]* block, to `multiple_files`. Finally, edit definitions.h and change `PRINT_TO_FILE` from *YES* to *NO*.
6. Compile the code:

   ```
   ~/PLUTO/Test_Problems/MHD/Orszag_Tang> make
   ```

7. If compilation was successful, you can now run the code by typing

   ```
   ~/PLUTO/Test_Problems/MHD/Orszag_Tang> ./pluto
   ```

   At this point, **PLUTO** reads the initialization file pluto.ini and starts integrating. The run should take a few minutes (depending on the machine you're running on) and the integration log should be dumped to screen.

Orszag-Tang I.C. & B.C.

For each simulation we need to define **initial** and **boundary** conditions.

I introduce some numerical simulations terminology here in ***bold***:

We set velocity and magnetic field in a 2D Cartesian coordinates, in a ***computational box*** with (x,y,0)=(256x256x0) grid cells and a ***physical domain*** x,y=[0,2*pi]

Velocity components: v = (-sin y, sin x, 0)

Magnetic field: B = (-sin y, sin 2x, 0)

Density: rho = 25/9

Pressure: p   = 5/3

Orszag-Tang I.C. & B.C.

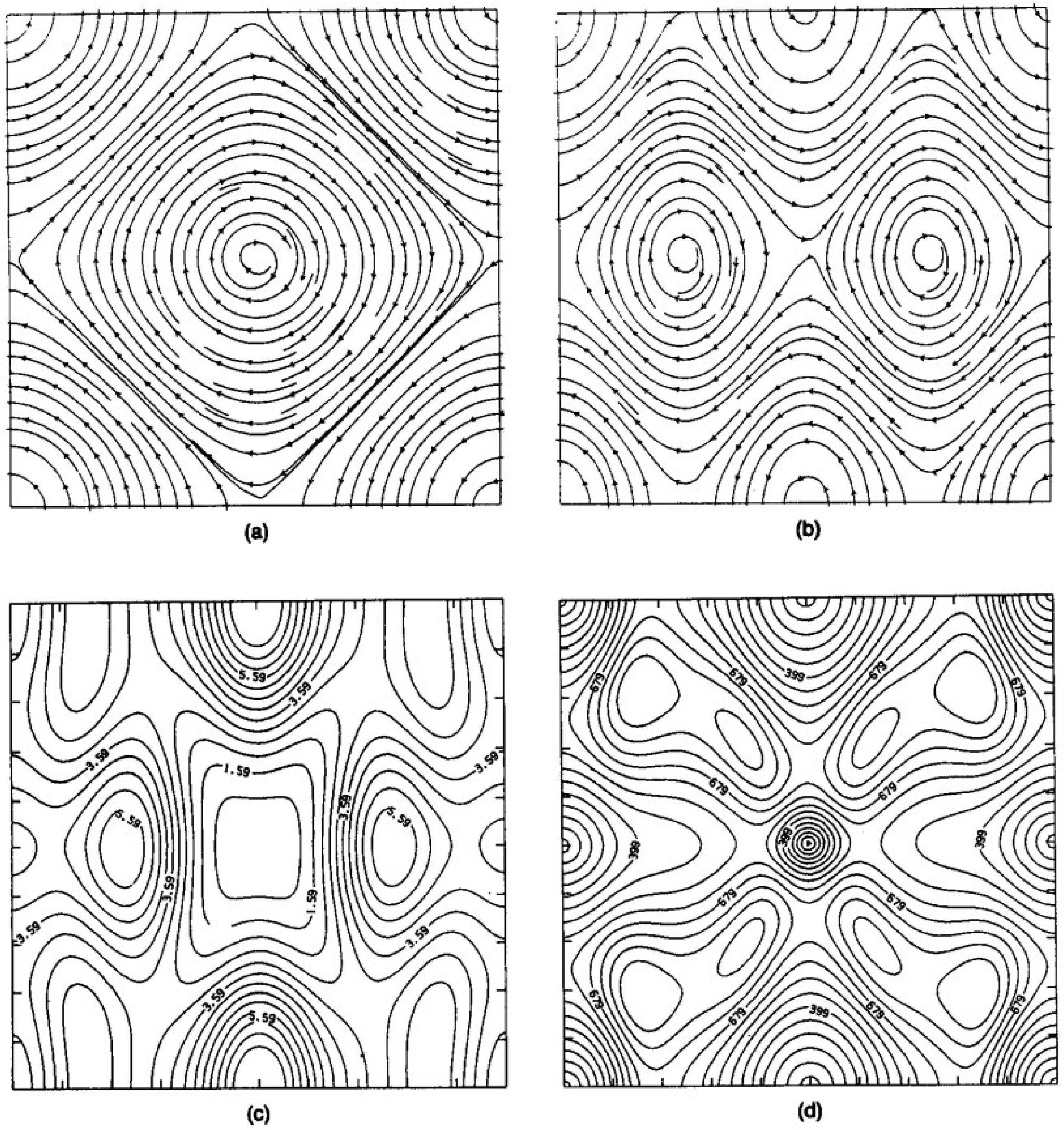
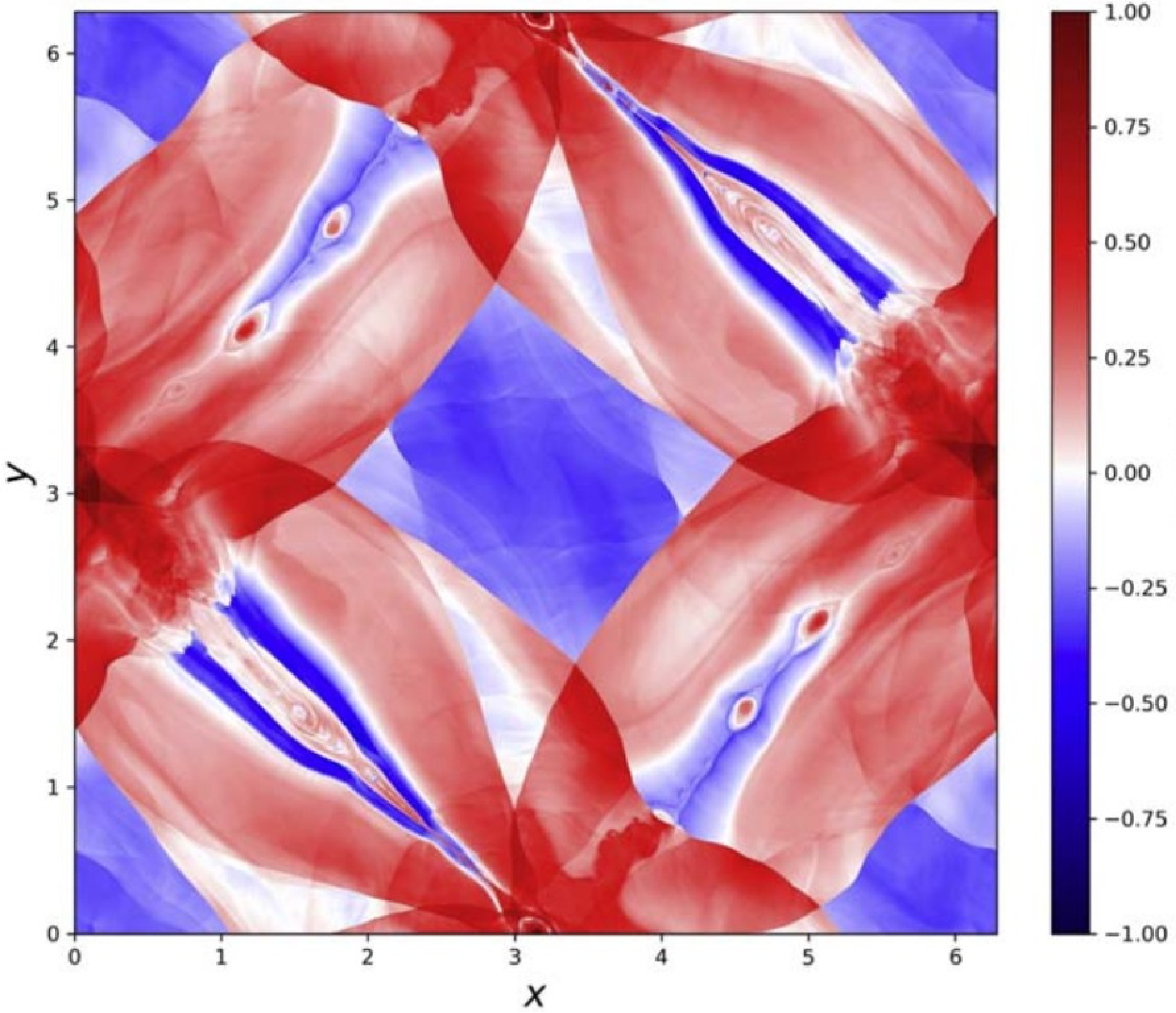


FIG. 1. Initial conditions: (a) velocity field; (b) magnetic field; initial conditions specific to the $M = 0.6$ case: (c) thermal pressure field; (d) local Mach number. Minimum and maximum values are 0.73 and 7.9 for thermal pressure and 0.0 and 0.97 for local Mach number.

OT, standard test but not so standard results



Hands-on introduction to setup files of PLUTO

- Guided tour through the source files:

  -pluto.ini contains definition of:
  grid, solver, choice of boundary conditions, output steps, parameters.

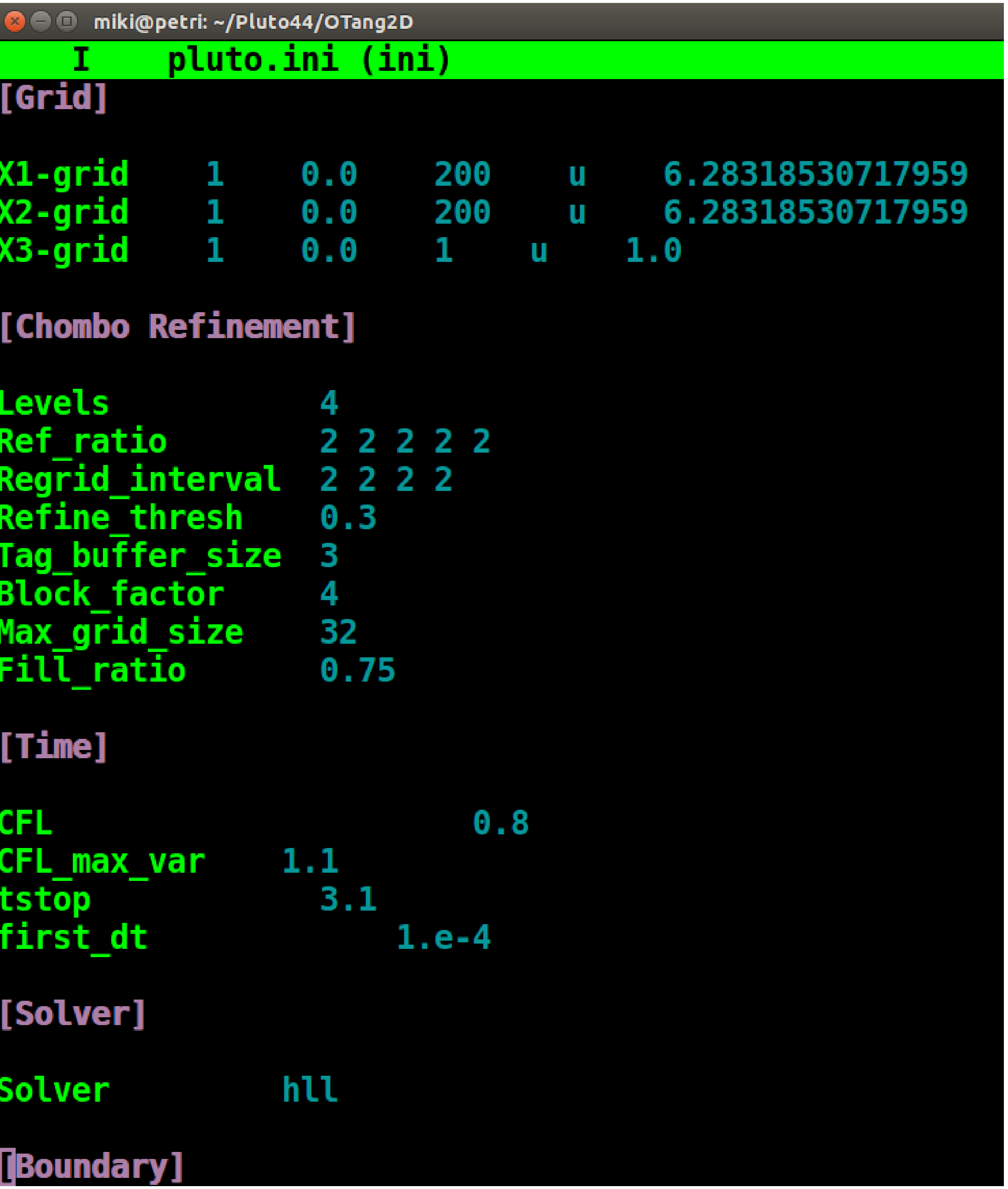
  -Definition of output formats.

  -Output files, files to save for later analysis and eventual restart.

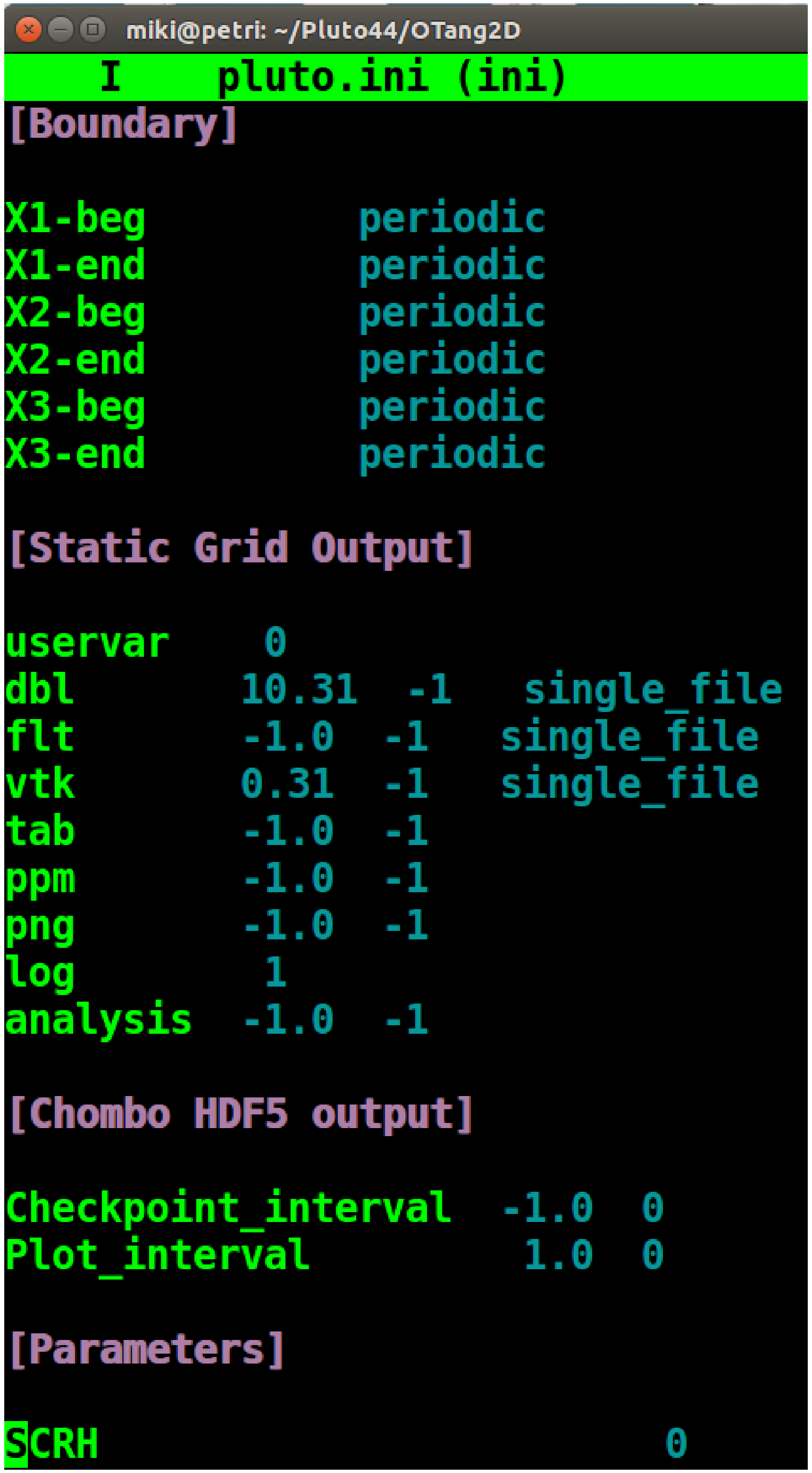
  -definitions.h: macros setting.

  -init.c: physical setup, equations for **initial** and **boundary** conditions.

Hands-on introduction: pluto.ini



```
miki@petri: ~/Pluto44/OTang2D
    I     pluto.ini  (ini)
[Grid]

X1-grid    1    0.0     200     u     6.28318530717959
X2-grid    1    0.0     200     u     6.28318530717959
X3-grid    1    0.0     1     u     1.0

[Chombo Refinement]

Levels                4
Ref_ratio             2 2 2 2 2
Regrid_interval       2 2 2 2
Refine_thresh         0.3
Tag_buffer_size       3
Block_factor          4
Max_grid_size         32
Fill_ratio            0.75

[Time]

CFL                           0.8
CFL_max_var       1.1
tstop                 3.1
first_dt                  1.e-4

[Solver]

Solver            hll

[Boundary]
```

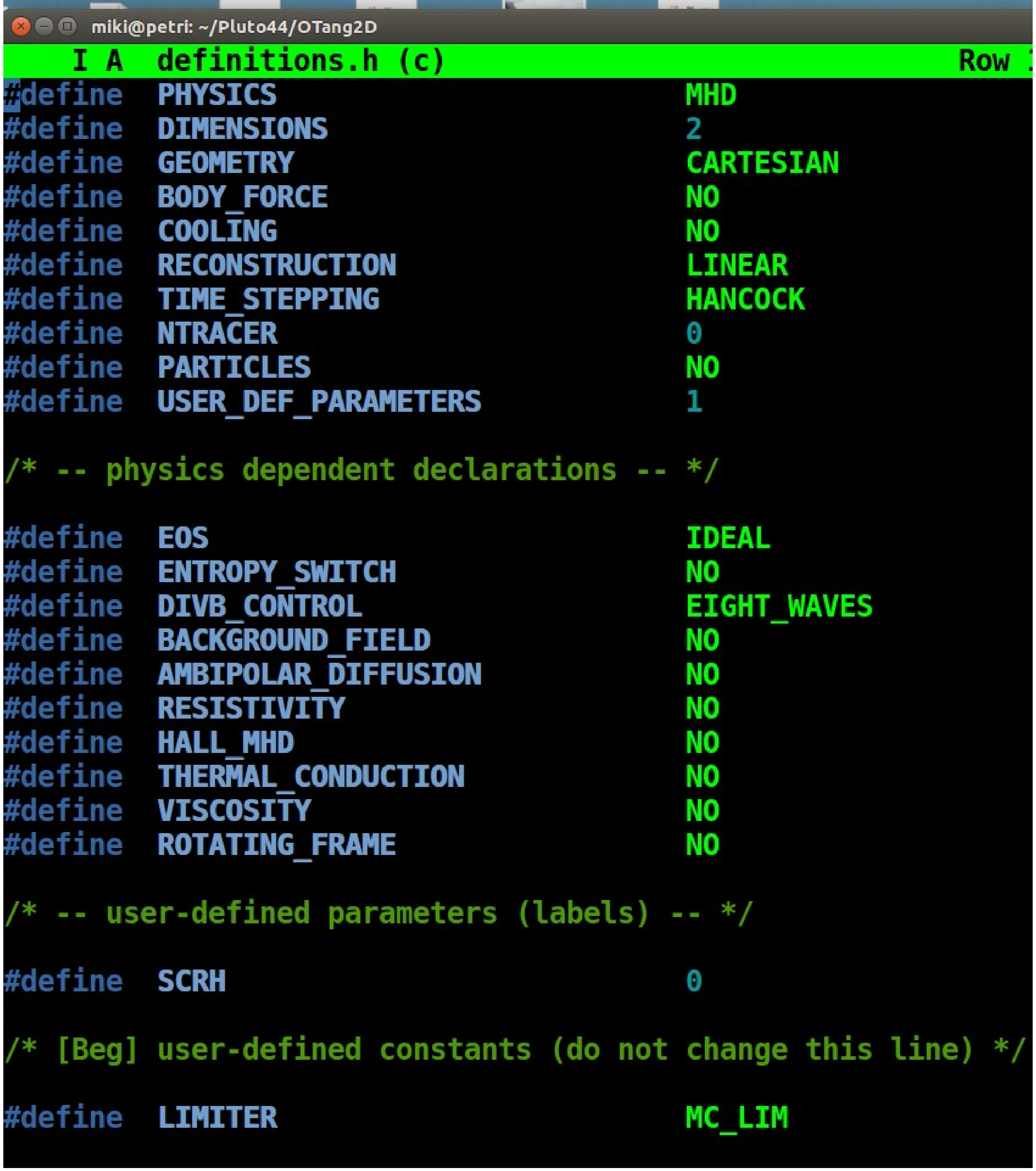


```
miki@petri: ~/Pluto44/OTang2D
    I     pluto.ini  (ini)
[Boundary]

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

[Static Grid Output]

uservar     0
dbl          10.31   -1    single_file
flt          -1.0   -1   single_file
vtk          0.31   -1   single_file
tab          -1.0   -1
ppm          -1.0   -1
png          -1.0   -1
log           1
analysis   -1.0   -1

[Chombo HDF5 output]

Checkpoint_interval  -1.0   0
Plot_interval          1.0   0

[Parameters]

SCRH                           0
```

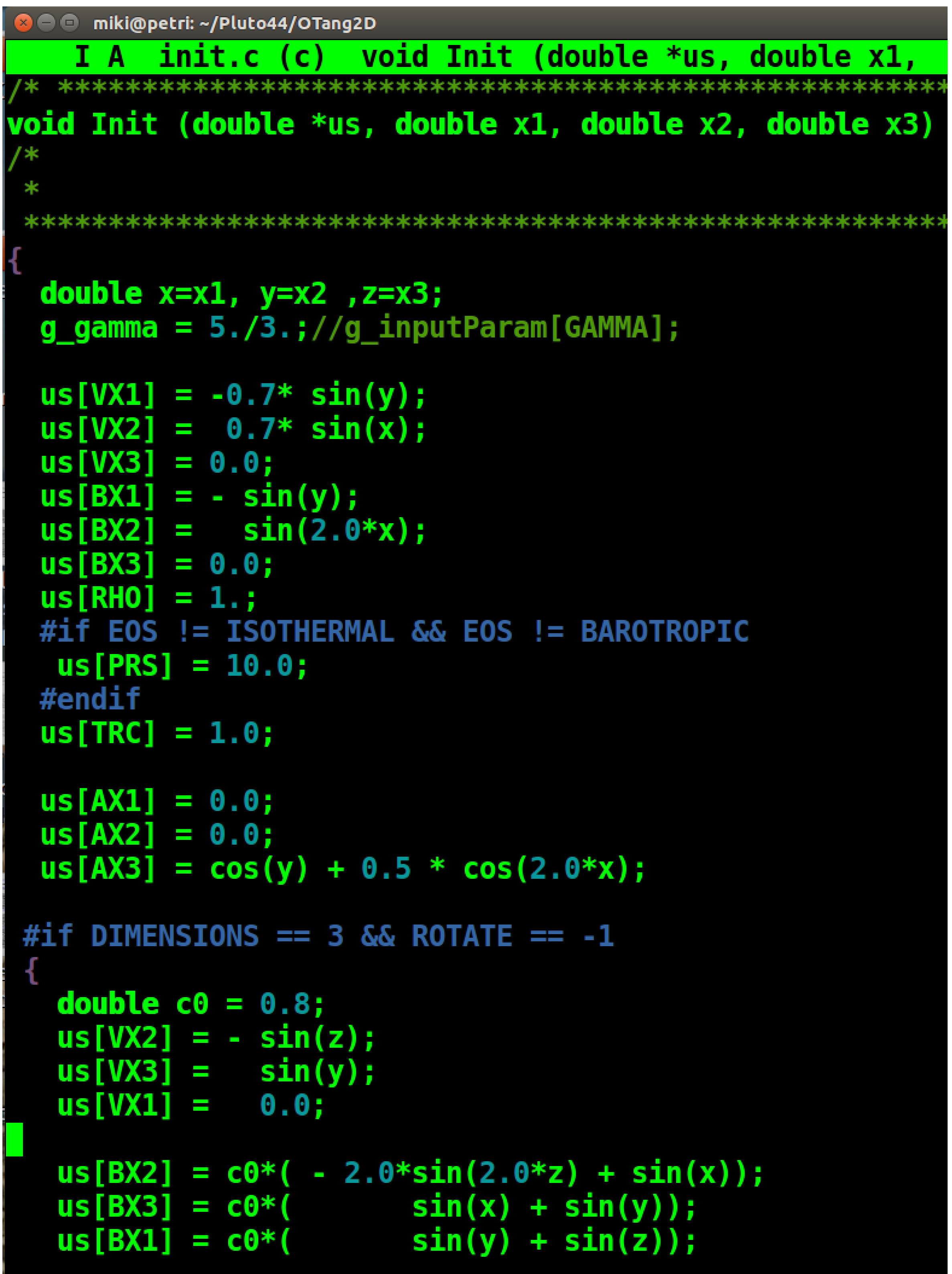
Output formats & files to save

- In pluto.ini we defined which output:
- data.xxxx.dbl files – viewing, processing with Python, idl etc. packages.
  -also, for restart, together with restart.out.
- data.xxxx.vtk files – for viewing with Paraview, VisIt.
- Together with data.xxxx.dbl and .vtk files, always save also, in the same directory with the results, pluto.ini, definitions.h. init.c, grid.out, dbl.out, restart.out, to know which setup produced the files, and also for some analysis and plotting packages, which might need them.
- grid.out files defines the geometry, dbl.out lists the output variables

Hands-on introduction: definitions.h



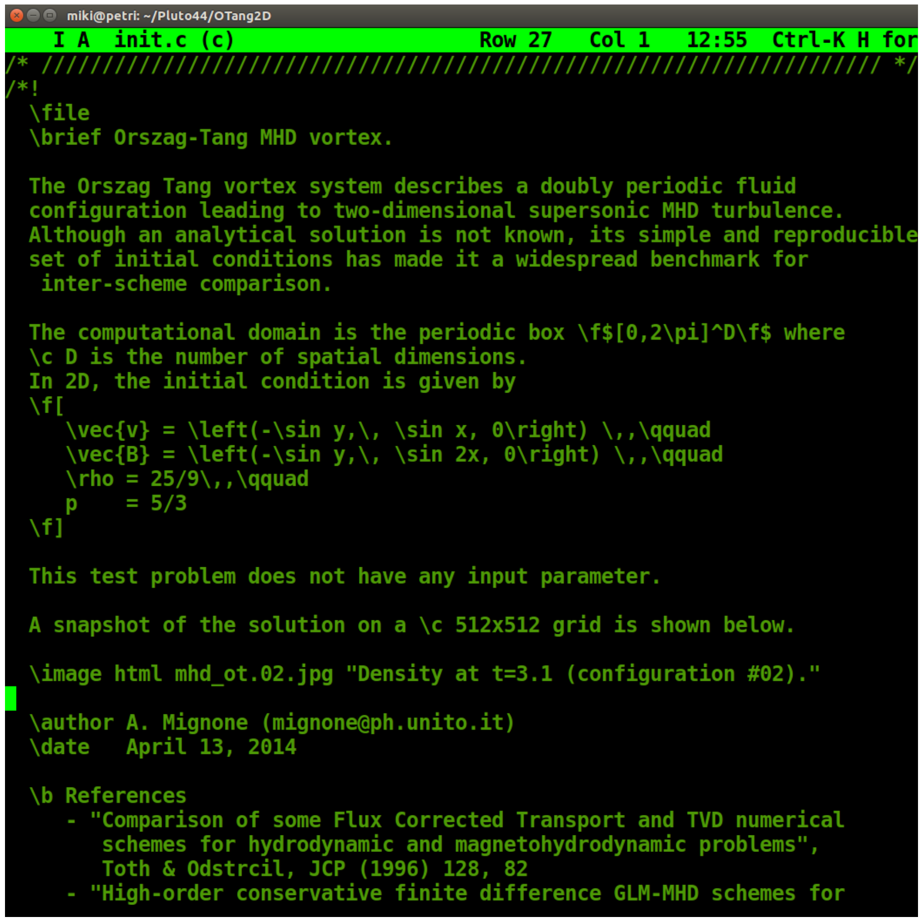
```
miki@petri: ~/Pluto44/OTang2D
    I A  definitions.h (c)                                   Row 
#define  PHYSICS                        MHD
#define  DIMENSIONS                     2
#define  GEOMETRY                       CARTESIAN
#define  BODY_FORCE                     NO
#define  COOLING                        NO
#define  RECONSTRUCTION                 LINEAR
#define  TIME_STEPPING                  HANCOCK
#define  NTRACER                        0
#define  PARTICLES                      NO
#define  USER_DEF_PARAMETERS            1

/* -- physics dependent declarations -- */

#define  EOS                            IDEAL
#define  ENTROPY_SWITCH                 NO
#define  DIVB_CONTROL                   EIGHT_WAVES
#define  BACKGROUND_FIELD               NO
#define  AMBIPOLAR_DIFFUSION            NO
#define  RESISTIVITY                    NO
#define  HALL_MHD                       NO
#define  THERMAL_CONDUCTION             NO
#define  VISCOSITY                      NO
#define  ROTATING_FRAME                 NO

/* -- user-defined parameters (labels) -- */

#define  SCRH                           0

/* [Beg] user-defined constants (do not change this line) */

#define  LIMITER                        MC_LIM
```



```
miki@petri: ~/Pluto44/OTang2D
   I A  init.c (c)  void Init (double *us, double x1,
/* *****************************************************
void Init (double *us, double x1, double x2, double x3)
/*
 *
 *****************************************************
{
  double x=x1, y=x2 ,z=x3;
  g_gamma = 5./3.;//g_inputParam[GAMMA];

  us[VX1] = -0.7* sin(y);
  us[VX2] =  0.7* sin(x);
  us[VX3] = 0.0;
  us[BX1] = - sin(y);
  us[BX2] =   sin(2.0*x);
  us[BX3] = 0.0;
  us[RHO] = 1.;
  #if EOS != ISOTHERMAL && EOS != BAROTROPIC
   us[PRS] = 10.0;
  #endif
  us[TRC] = 1.0;

  us[AX1] = 0.0;
  us[AX2] = 0.0;
  us[AX3] = cos(y) + 0.5 * cos(2.0*x);

 #if DIMENSIONS == 3 && ROTATE == -1
 {
   double c0 = 0.8;
   us[VX2] = - sin(z);
   us[VX3] =   sin(y);
   us[VX1] =   0.0;

   us[BX2] = c0*( - 2.0*sin(2.0*z) + sin(x));
   us[BX3] = c0*(       sin(x) + sin(y));
   us[BX1] = c0*(       sin(y) + sin(z));
```

Hands-on introduction: init.c

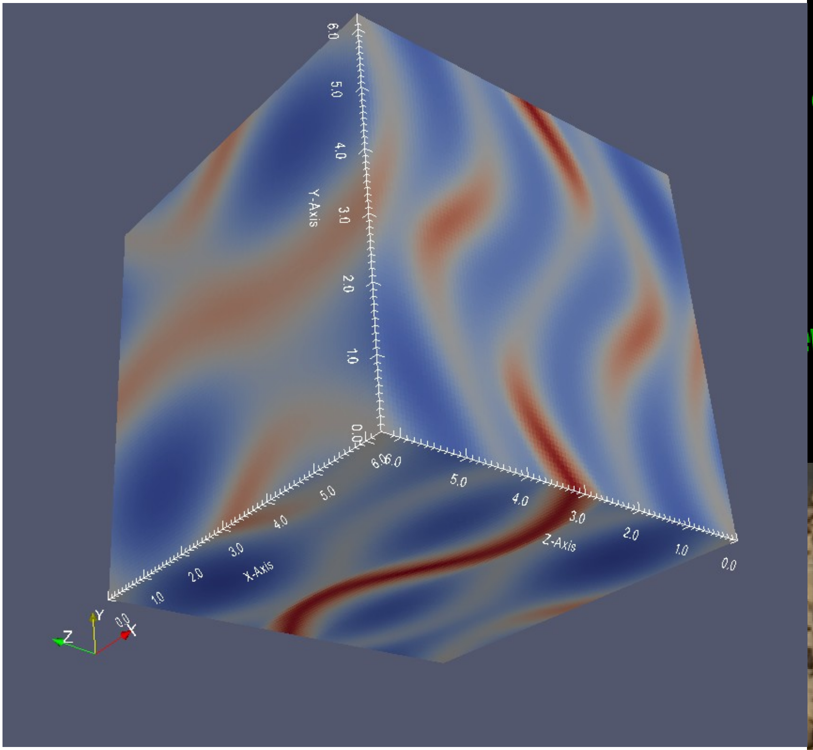
- Note that at the beginning of each file in PLUTO is a brief description about what it does. It is a very good practice, follow it.

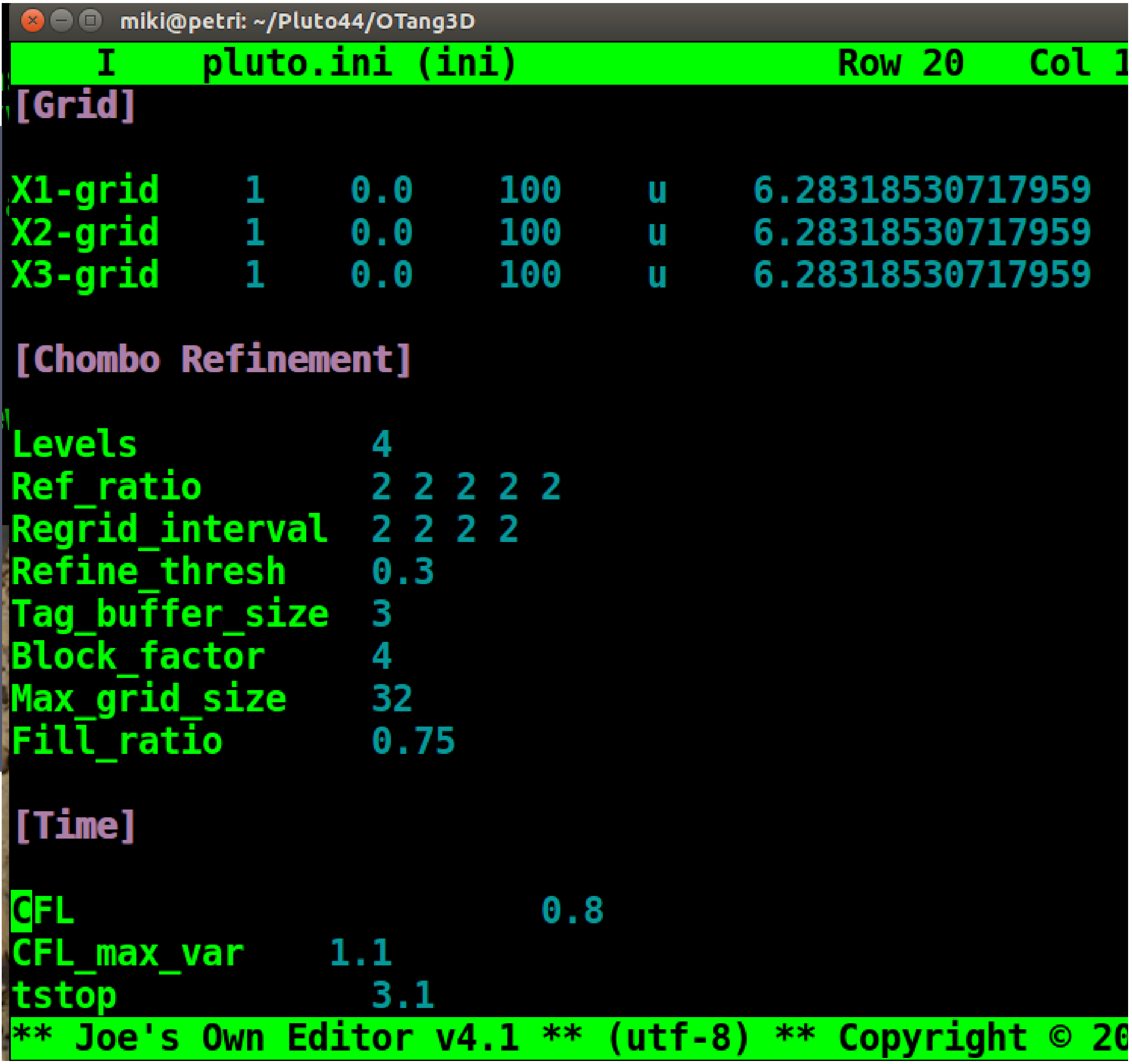


```
miki@petri: ~/Pluto44/OTang2D
   I A  init.c (c)                    Row 27    Col 1    12:55  Ctrl-K H for
/* ///////////////////////////////////////////////////////////////////// */
/*!
  \file
  \brief Orszag-Tang MHD vortex.

  The Orszag Tang vortex system describes a doubly periodic fluid
  configuration leading to two-dimensional supersonic MHD turbulence.
  Although an analytical solution is not known, its simple and reproducible
  set of initial conditions has made it a widespread benchmark for
   inter-scheme comparison.

  The computational domain is the periodic box \f$[0,2\pi]^D\f$ where
  \c D is the number of spatial dimensions.
  In 2D, the initial condition is given by
  \f[
     \vec{v} = \left(-\sin y,\, \sin x, 0\right) \,,\qquad
     \vec{B} = \left(-\sin y,\, \sin 2x, 0\right) \,,\qquad
     \rho = 25/9\,,\qquad
     p    = 5/3
  \f]

  This test problem does not have any input parameter.

  A snapshot of the solution on a \c 512x512 grid is shown below.

  \image html mhd_ot.02.jpg "Density at t=3.1 (configuration #02)."

  \author A. Mignone (mignone@ph.unito.it)
  \date   April 13, 2014

  \b References
     - "Comparison of some Flux Corrected Transport and TVD numerical
       schemes for hydrodynamic and magnetohydrodynamic problems",
       Toth & Odstrcil, JCP (1996) 128, 82
     - "High-order conservative finite difference GLM-MHD schemes for
```

Setup of 3D simulation, Orszag-Tang test in 3D

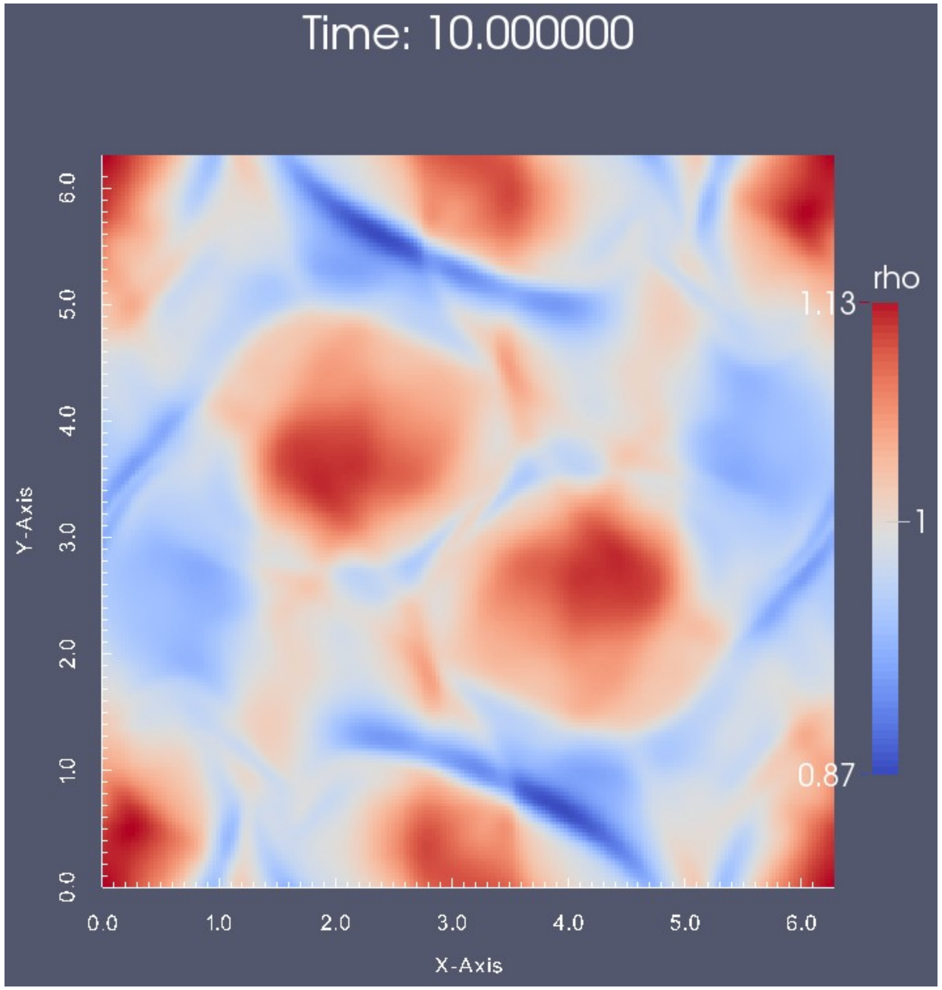
- Simple, just copy one of 3D setups from PLUTO/Test_Problems/MHD/Orszag_Tang, perform a setup.py (made a pls alias?) to pre-set a 3D run, and add copy the corresponding pluto.ini (e.g. _07).
- Visualisation of the result in 3D with Paraview:

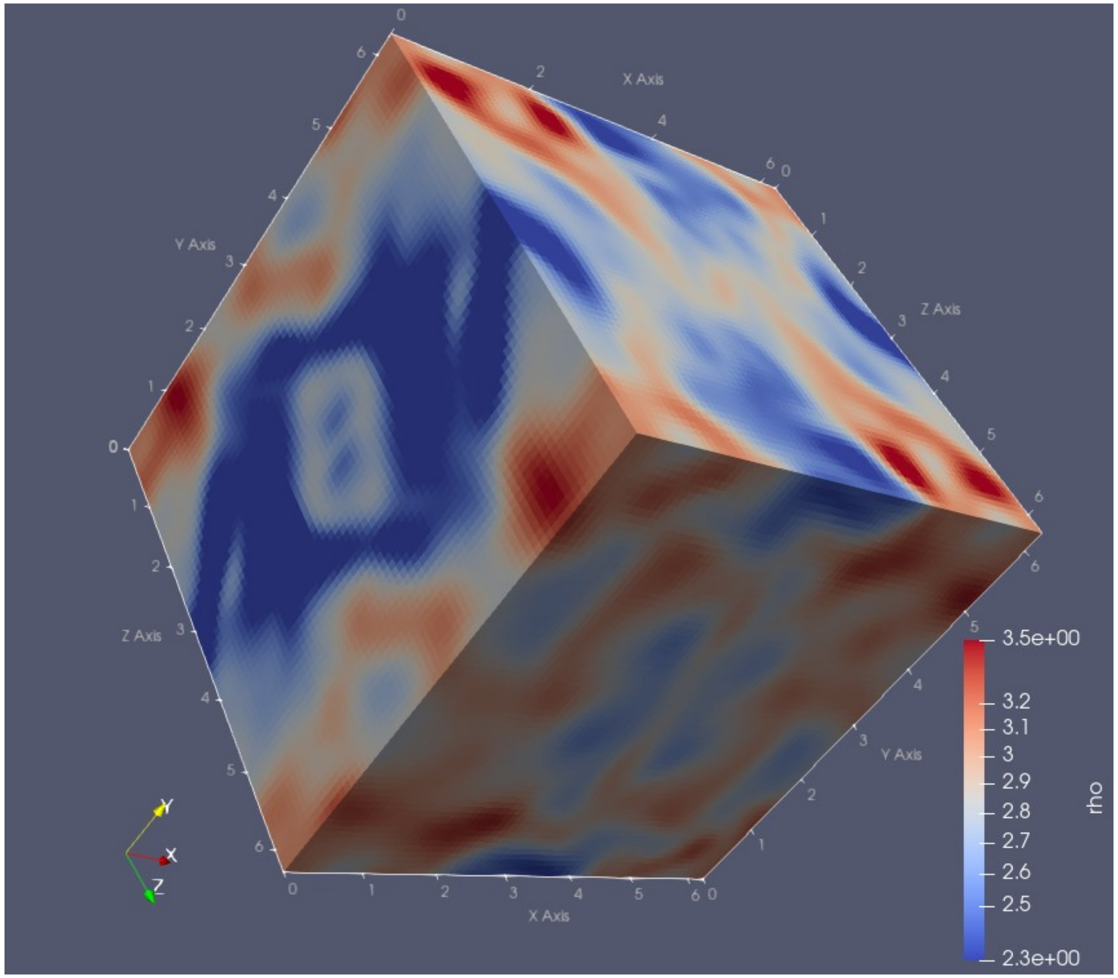




```
   I     pluto.ini (ini)                     Row 20    Col 1
[Grid]

X1-grid     1     0.0     100     u     6.28318530717959
X2-grid     1     0.0     100     u     6.28318530717959
X3-grid     1     0.0     100     u     6.28318530717959

[Chombo Refinement]

Levels              4
Ref_ratio           2 2 2 2 2
Regrid_interval     2 2 2 2
Refine_thresh       0.3
Tag_buffer_size     3
Block_factor        4
Max_grid_size       32
Fill_ratio          0.75

[Time]

CFL                         0.8
CFL_max_var       1.1
tstop               3.1
** Joe's Own Editor v4.1 ** (utf-8) ** Copyright © 20
```

Visualization and animation with Paraview in 2D, 3D





Summary of the Part II

- We described, set and run the 2D Orszag-Tang test.
- I detailed the setup files of PLUTO.
- We learned which are the output formats and plotting in 2D with Paraview.
- We set and run the 3D Orszag-Tang test.
- We learned to use Paraview for 3D results.
- We learned how to animate results with Paraview.

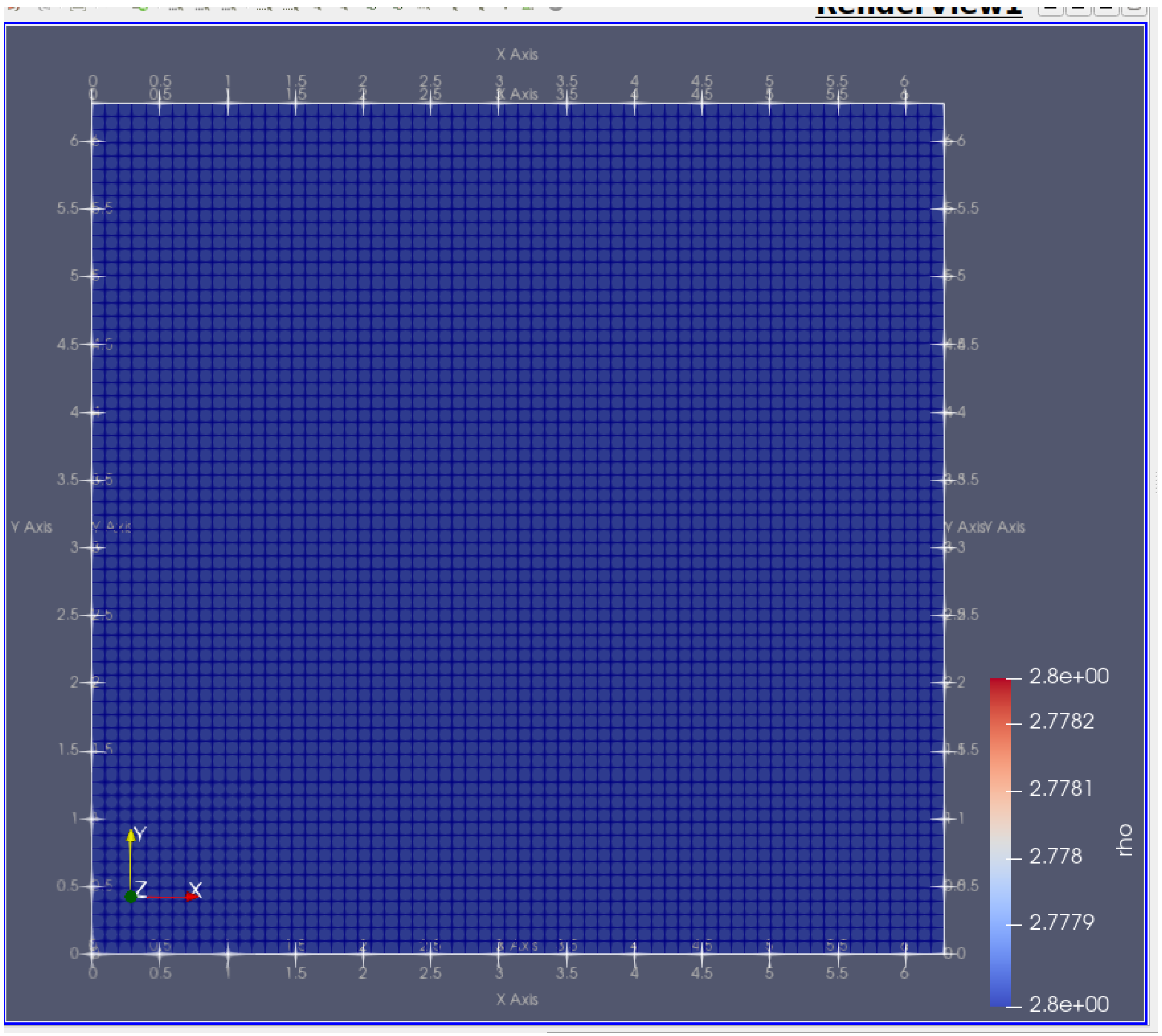
Outline, Part III, Setup of the SPI simulations

- Initial and boundary conditions setup for SPI simulations
- Hands-on introduction to setup files.
- Visualization of the results with Paraview.
- Movie time: animation and saving of the movie.

Types of grid and needed resolution

- Computation is performed on a grid. More grid cells, better precision.
- Cartesian, polar, cylindrical, spherical grid possible in PLUTO
- A static grid can be equidistant, logarithmic,…
- There can be more nested static grids.
- With CHOMBO package, one can use Adaptive Mesh Refinement (AMR), where the mesh is refined in the places where some condition, given by the user, is satisfied.

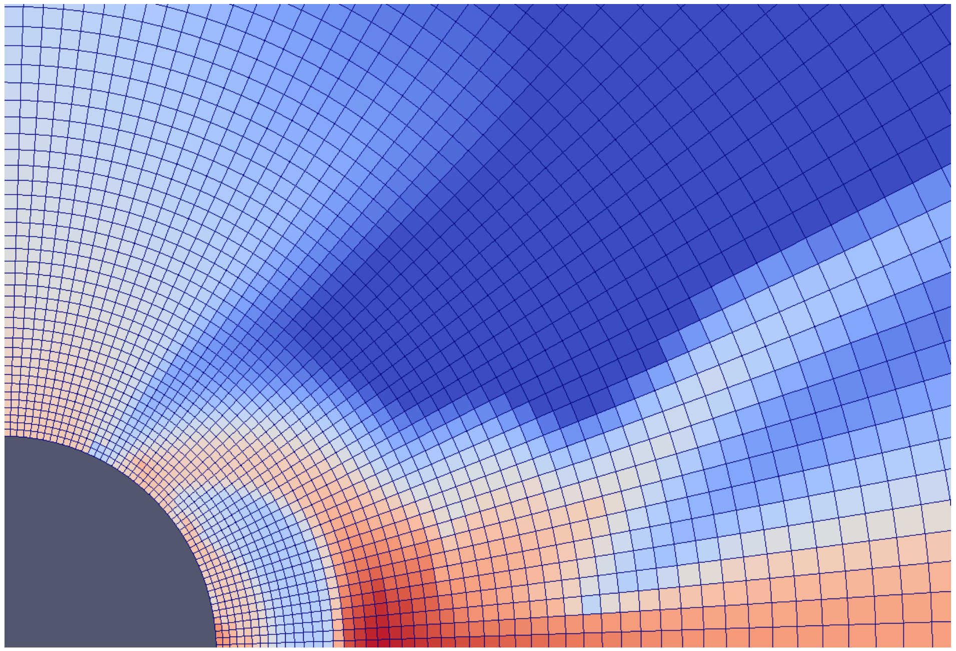
Resolution needed in a setup

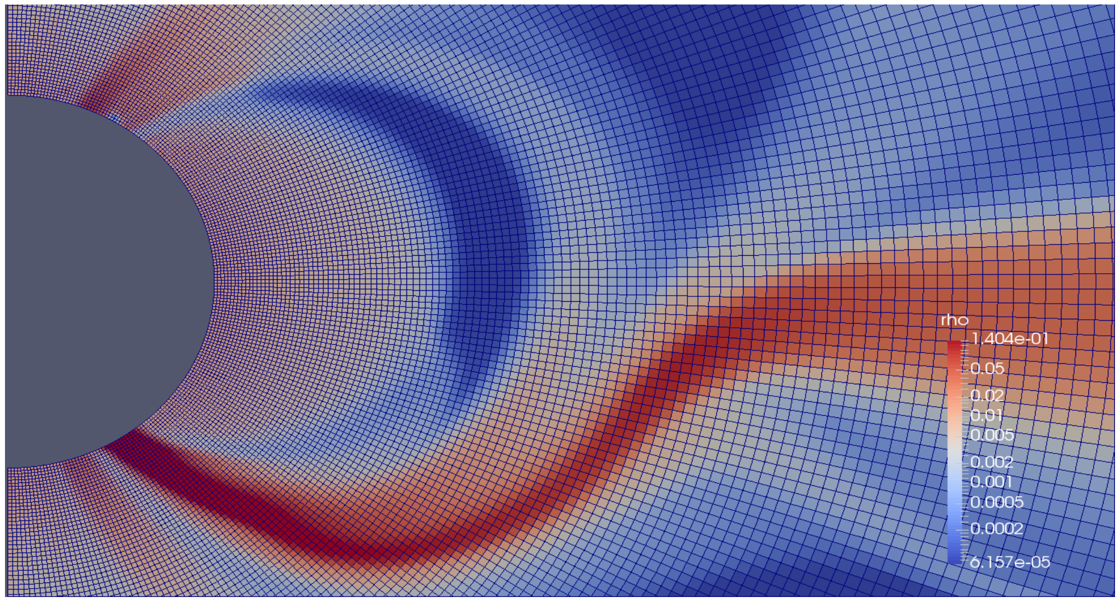


- Resolution defines precision of a solution.
- Larger resolution means larger files and longer, sometimes impossibly long computation.

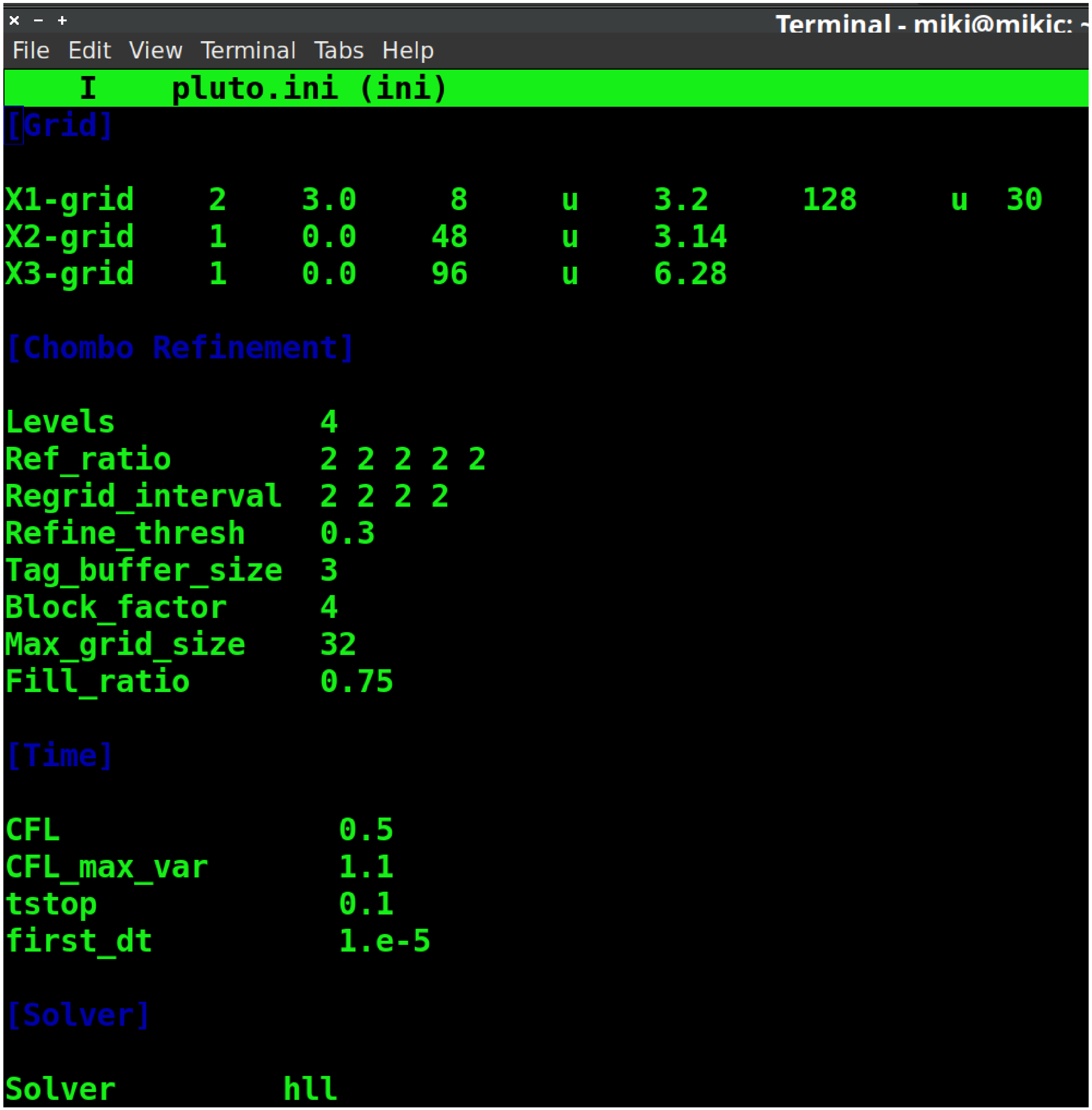
Resolution needed in a setup-an example from star-disk simulations

- Resolution in my production runs is Rxϑ=[217x100] grid cells in ϑ=[0,π/2], with a logarithmic grid spacing in the radial direction. For testing I also use Rxϑ=[109x50] grid cells, which gives qualitatively correct results.
- The accretion column is well resolved if *a rule of thumb* is satisfied that there is at least that many grid cells how many there is independent variables (5 in HD case, 8 in MHD case in our setups: density, pressure, 3 components of **v** and **B**).
- I did also ϑ=[0,π] cases in Rxϑ=[217x200] grid cells, as well as Rxϑ=[109x100] grid cells. Now there is no need to define the equatorial boundary condition, the simulation self-consistently computes across the domain. This case usually leads to an asymmetry with respect to the equatorial plane.



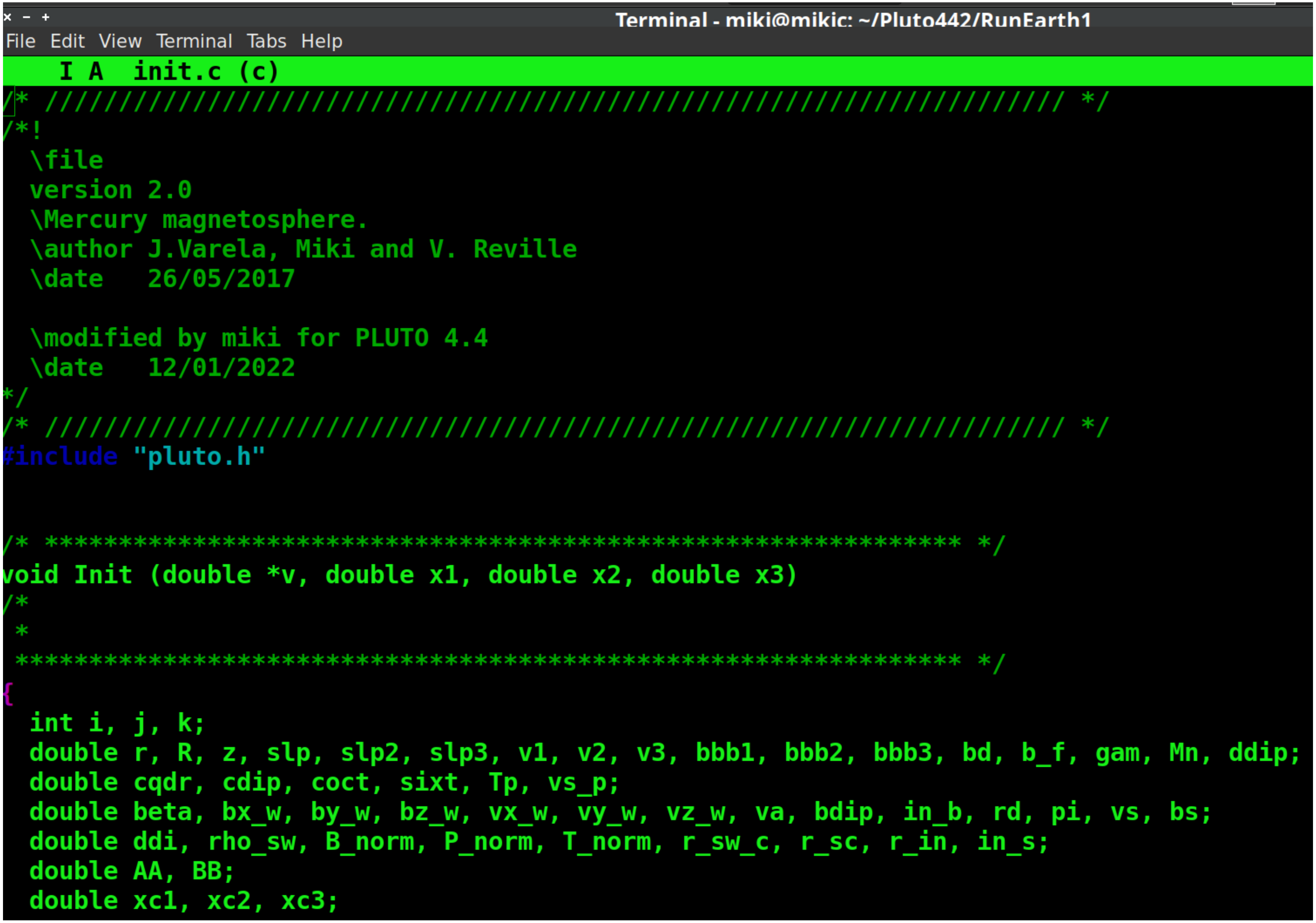


Star-planet: pluto.ini



```
     I     pluto.ini (ini)
[Grid]

X1-grid    2    3.0     8     u    3.2     128     u  30
X2-grid    1    0.0    48     u    3.14
X3-grid    1    0.0    96     u    6.28

[Chombo Refinement]

Levels           4
Ref_ratio        2 2 2 2 2
Regrid_interval  2 2 2 2
Refine_thresh    0.3
Tag_buffer_size  3
Block_factor     4
Max_grid_size    32
Fill_ratio       0.75

[Time]

CFL               0.5
CFL_max_var       1.1
tstop             0.1
first_dt          1.e-5

[Solver]

Solver         hll
```

Star-planet: init.c



Terminal - miki@mikic: ~/Pluto442/RunEarth1

File Edit View Terminal Tabs Help

```
    I A  init.c (c)
/* ///////////////////////////////////////////////////////////////////// */
/*!
  \file
  version 2.0
  \Mercury magnetosphere.
  \author J.Varela, Miki and V. Reville
  \date   26/05/2017

  \modified by miki for PLUTO 4.4
  \date   12/01/2022
*/
/* ///////////////////////////////////////////////////////////////////// */
#include "pluto.h"


/* ************************************************************** */
void Init (double *v, double x1, double x2, double x3)
/*
 *
 ************************************************************** */
{
  int i, j, k;
  double r, R, z, slp, slp2, slp3, v1, v2, v3, bbb1, bbb2, bbb3, bd, b_f, gam, Mn, ddip;
  double cqdr, cdip, coct, sixt, Tp, vs_p;
  double beta, bx_w, by_w, bz_w, vx_w, vy_w, vz_w, va, bdip, in_b, rd, pi, vs, bs;
  double ddi, rho_sw, B_norm, P_norm, T_norm, r_sw_c, r_sc, r_in, in_s;
  double AA, BB;
  double xc1, xc2, xc3;
```

Numerical methods in PLUTO for star-planet interaction

•**Some examples of what you can usually read in a setup description:**

-Simulations were performed using the second-order piecewise linear reconstruction.

-Van Leer limiter, which is more diffusive and enhances stability, is used in density and magnetic field and a minmod (monotonized central differences) limiter in pressure and velocity.

-An approximate Roe solver (hlld in the pluto options) was used

-The second-order time-stepping (RK2) was employed.

-$\nabla \cdot \mathbf{B} = 0$ was maintained by the constrained transport method.

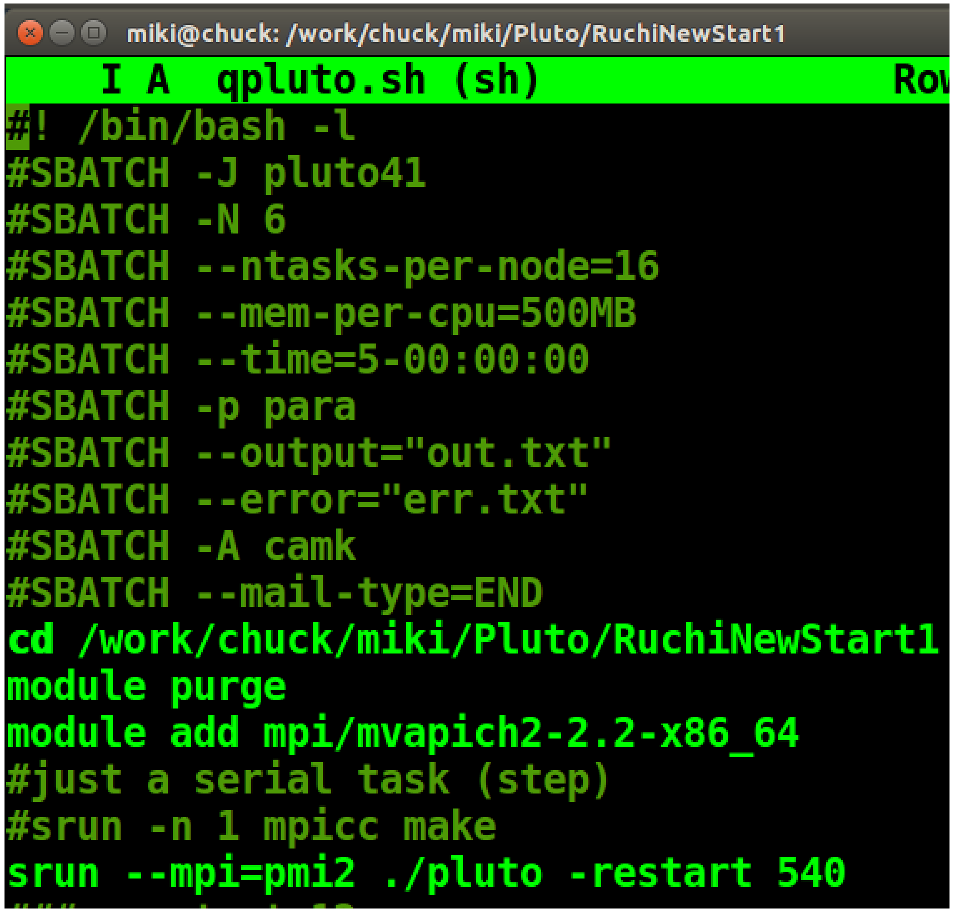
Running on a Linux cluster

- Usually Linux clusters and supercomputers use management and queuing system. I will describe two of them, which work in a similar way. Think of them just as an expanded command for running your job.

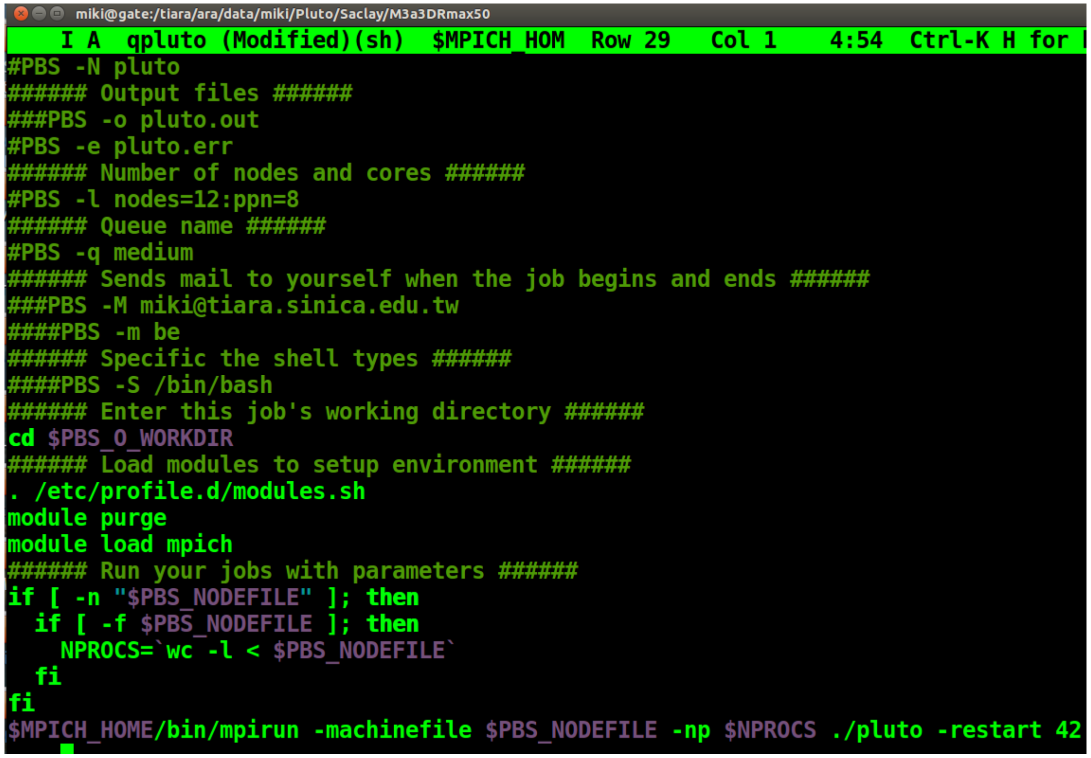
  - **SLURM** - a free one, became quite reliable so one does not need to pay for management, which could come with a significant cost. After creating a slurm_job_file, execute sbatch slurm_job_file . Most often used commands: sbatch, squeue, scancel.

  - **PBS** – (Portable Batch System), there are Open (free) and Pro (not free) versions, also very similar is its fork, **TORQUE.** After creating a PBS or TORQUE job script pbs_job_file, execute in terminal: qsub pbs_job_file.

  Most often used commands are: qsub, qstat, qdel, qmgr and xpbs, pbsjobs (located in /home/Tools/bin) for additional detail about queued and running jobs.

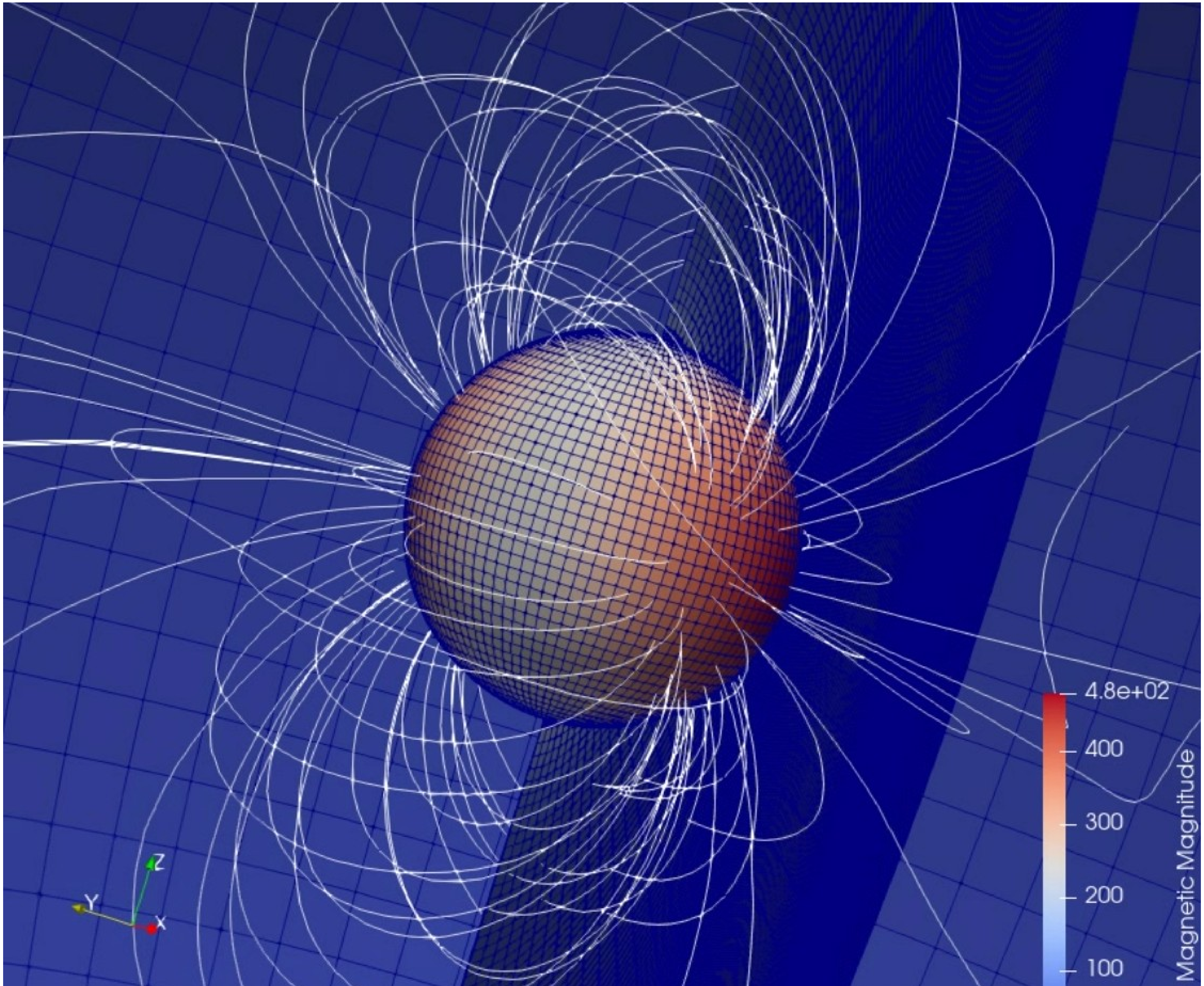

miki@chuck: /work/chuck/miki/Pluto/RuchiNewStart1

```
    I A  qpluto.sh (sh)                   Rov
#! /bin/bash -l
#SBATCH -J pluto41
#SBATCH -N 6
#SBATCH --ntasks-per-node=16
#SBATCH --mem-per-cpu=500MB
#SBATCH --time=5-00:00:00
#SBATCH -p para
#SBATCH --output="out.txt"
#SBATCH --error="err.txt"
#SBATCH -A camk
#SBATCH --mail-type=END
cd /work/chuck/miki/Pluto/RuchiNewStart1
module purge
module add mpi/mvapich2-2.2-x86_64
#just a serial task (step)
#srun -n 1 mpicc make
srun --mpi=pmi2 ./pluto -restart 540
```


miki@gate:/tiara/ara/data/miki/Pluto/Saclay/M3a3DRmax50

```
    I A  qpluto (Modified)(sh)  $MPICH_HOM  Row 29   Col 1    4:54  Ctrl-K H for
#PBS -N pluto
###### Output files ######
###PBS -o pluto.out
#PBS -e pluto.err
###### Number of nodes and cores ######
#PBS -l nodes=12:ppn=8
###### Queue name ######
#PBS -q medium
###### Sends mail to yourself when the job begins and ends ######
###PBS -M miki@tiara.sinica.edu.tw
####PBS -m be
###### Specific the shell types ######
####PBS -S /bin/bash
###### Enter this job's working directory ######
cd $PBS_O_WORKDIR
###### Load modules to setup environment ######
. /etc/profile.d/modules.sh
module purge
module load mpich
###### Run your jobs with parameters ######
if [ -n "$PBS_NODEFILE" ]; then
  if [ -f $PBS_NODEFILE ]; then
    NPROCS=`wc -l < $PBS_NODEFILE`
  fi
fi
$MPICH_HOME/bin/mpirun -machinefile $PBS_NODEFILE -np $NPROCS ./pluto -restart 42
```

Star-planet: plots in 3D, streamlines with Paraview



Summary, Part III

- We learned how to set the SPI simulations
- Plotting in Paraview, also streamlines of velocity and magnetic field
- We learned to run a job on Linux cluster with queuing system.
- Animation and saving of the results.

--Tasks, final goals.

Lectures summary & Concluding remarks

-In 3 lectures in 5 days, I presented the PLUTO code, in hands-on approach.

-PLUTO is well used (=tested and updated) in the astrophysics community, is user-friendly (good documentation) and very well written (good tool for hands-on learning of C programming language).

- We started with general description and followed with a standard Sod shock (in 1D, HD) and Orszag-Tang (MHD) test in 2D and 3D.

- Along the way, we learned visualization in gnuplot and Paraview

- I detailed the 3D setup and showed the script for running on the linux cluster under slurm queuing system.

- We defined the tasks for each participant.