CSAM 2015, FZ Julich (CHARM school 14-18 september2015)

Preliminaries

```
After login to JUDGE, download MPI-AMRVAC in your home directory
git clone https://gitlab.com/mpi-amrvac/amrvac.git

# modify (using vi) ~/.bashrc to setup environment variables
# in ~/.bashrc add

export AMRVAC_DIR=$HOME/amrvac
PATH="$AMRVAC_DIR:$AMRVAC_DIR/tools:./:$PATH"

# copy hands-on tests to your work folder
cp -r ~/amrvac/tests/CSAM2015/* $WORK/
cd $WORK

# select a project, e.g., jetCloudInteractions

cd jetCloudInteractions

# look at the first line(s) of a par-file, e.g., amrvac2D.par and find a command
# "$AMRVAC_DIR/setup.pl ..." to setup basic parameters

$AMRVAC_DIR/setup.pl -d=22 -phi=0 -z=0 -g=16,16 -p=hd -eos=default -nf=1 -ndust=0 -u=nul -arch=default

# compile the code and get the executable file amrvac
make

# make a job script, e.g., use 1 node (12 CPUs) to run for at most 10 hours
vi job.mb

  #!/bin/bash -x

  #MSUB -l nodes=1:ppn=12

  #MSUB -l walltime=10:00:00

  cd /work/hpclab/youruserid/jetCloudInteractions

  mpiexec -np 12 ./amrvac -i amrvac2D.par

# submit the job to JUDGE to run

msub job.mb

# more useful commands:
```

```
# get detailed information about a job

checkjob -v <jobid>

# show status of your (running) jobs

showq -u <youruserid>

# show status of all (running) jobs

showq [-r]

# cancel a job

canceljob <jobid>
```

The subdirectory
`$AMRVAC_DIR/tests/CSAM2015`
has a large number of tests for you to explore: Note that we would appreciate to be informed when you would start from any of these setups to do your own work, maybe we can collaborate. We certainly like to be acknowledged for our efforts.

jetCloudInteractions

In this Euler (gas dynamical) test, we simulate the impact of a Mach 12 jet flow as it passes through lighter external medium (density contrast 3), and finally impacts on a spherical (density stratified) cloud which is denser than the jet. It is inspired from a 3D SPH study done by De Gouveia del Pino, E., ApJ 526, 862-873 (1999).

The user file can be used for either a 2D (purely planar, Cartesian) or a 3D setup. It is meant to demonstrate the use of having an additional tracer quantity added to the Euler equations.

The amrvacusr.t file shows how to set up the initial condition, and how to control the left ($x = 0$) inlet boundary by fixing all (primitive) quantities.

The parfiles demonstrate the use of combining boundary prescriptions that are already available (free-outflow with 'cont' or 'noinflow' conditions) with a special (here, fixed) treatment of the inlet.

Use

```
diff amrvac2D.par amrvac3D.par
```

to spot simple differences between a 2D and a 3D variant. Check for `mxnest` to see how many grid levels you use, together with the (AMR level one) grid settings in `nxlone1, nxlone2, nxlone3`.

You can try to experiment with different resolutions, different numerical schemes (it now uses a strong stability preserving Runge-Kutta scheme, and a MP5 limiter in an HLLC discretization).

dustyRM

This is a 2D hydro setup, meant to demonstrate the use of gas-dust evolutions. The setup can be done without any dust species (`parfile_nodust`), or with an arbitrary number of dust bins respresenting a dust grain size distribution. An example for 4 dust species (`parfile_4dust`) is given.

The setup studies a Mach 2 shock, impinging on an inclined density discontinuity (density contrast of 3). Dust is only present beyond the density discontinuity. The pure hydro setup will show you double mach reflection (twice, both bottom and top are handled reflective), along with Richtmyer-Meshkov (i.e. Kelvin-Helmholtz on the shocked contact) instability development. If dust is included, the dust grain size will determine in how far the dust is coupled to the gas, and possibily evacuated from the vortices.

The `amrvacusr.t` file shows you how to add additional output variables to the vtu files which are generated during runtime. In this example, the vtu files contain the primitive variables, together with a Schlieren plot of the density (i.e. the stretched density gradient).

Check out how we run at Courant parameter of 1.5 (thanks to the `ssprk54` time stepping).

Note: the `normvar(*)` array is used only in the conversion to vtu files, and here also in the drag force computation when dust is present. Note that Paraview may think that the ranges for (dust) densities are too small to render, but this is obviously not true (use the calculator to multiply to a range that paraview does recognize...).

Related research is in 'Effect of dust on Kelvin-Helmholtz instabilities', T. Hendrix & R. Keppens, 2014, Astron. & Astrophys. 526, A114, `doi:10.1051/0004-6361/201322322`

windbubble

This is also a 2D hydro evolution, but in an axisymmetric setup, where there is a user-defined source term, as well as (local) radiative loss effects incorporated. The setup models a stellar wind bubble, as it moves through the ISM (the simulation is performed in the frame fixed on the star, the ISM blows past the star from above).

The `amrvacusr.t` and `amrvacusrpar.t` shows how to incorporate the radiative loss module
(see e.g. `INCLUDE:amrvacmodules/cooling.t` and other statements accessing its functionality),
where we have several means to handle the energy loss term, and many of the frequently used cooling tables pre-coded.

Check out how we enforce AMR in the wind bubble region, using the `specialrefine_grid` subroutine.

orszagtang

This directory contains a setup for the Orszag-Tang problem, a typical MHD testcase. The code is set up to handle both 2D and 3D variants, under either an isothermal closure (no energy equation) or in fully adiabatic ideal MHD (with energy evolution). This example serves to show the use of the `-eos=` flag at compile time. The test lets a Mach 1 vortical flow distort a series of magnetic islands.

Examples are given for 2D and 3D, for isothermal and full MHD (`amrvac_otmhd2D.par`, `amrvac_otmhd3D.par`, `amrvac_otmhdiso2D.par`, `amrvac_otmhdiso3D.par`). Use `diff` to spot obvious differences in these parfiles.

Boundaries are (double to triple) periodic. Note the settings for resolution (up to 5 grid levels in the 2D runs, domain decomposition at 100^3 in the 3D runs). When simulated far enough, and at high resolution, these ideal MHD 2D runs will show nice tearing instabilities on the developing current sheets. Note that there is no explicit resistivity here. The monopole condition is handled through a diffusive approach (`typedivbfix='linde'`).

doubleGEM

This test is meant to illustrate the use of resistive to Hall-MHD evolutions. It simulates the double periodic, double GEM test (this test generalizes the standard Geospace Environment Modeling Challenge, a standard reconnection setup, to a setup where energy conservation can be verified easily, see 'Resistive MHD reconnection: resolving long-term, chaotic dynamics', R. Keppens, O. Porth, K. Galsgaard, J.T. Frederiksen, A.L. Restante, G. Lapenta, & C. Parnell, 2013, Phys. of Plasmas 20, 092109 (17pp). `doi: 10.1063/1.4820946`).

The setup here illustrates the use of Hall MHD, where one additionally needs to activate Hall-physics in the `definitions.h` file. Also, it shows how to use the GLM approach to monopole control, which also requires defining in `definitions.h`. Look for

```
#define GLM
#define HALL
```

and spot the corresponding code parts in the user file (amrvacusr.t) or the source code (once compiled, these are in `F90sources/`, where you can inspect the `amrvacphys.f` file, compared to the `amrvacphys.t` file from the `$AMRVAC_DIR/src/mhd` directory).

Note that our (explicit) treatment of the Hall term makes the Hall run (`doublegemmhdrunA`) much more challenging than the resistive MHD run (`doublegemmhdrunB`).

See how we use the `iprob` problem switch in the amrvacusr.t and the parfiles, to use the same compiled code for running different problems.

We set up the problems to use fourth order finite differences here (using MP5 limiting) and strong stability preserving Runge-Kutta schemes. Note the use of extra ghost cells (`dixB`).

The test is done in 2.5D (invariance in the ignored z-direction), note that the resistive run keeps B_z and v_z zero at all times, while Hall physics causes the generation of out-of-plane vector components.

A visco-resistive variant of this test was discussed in 'MPI-AMRVAC for solar and astrophysics', O. Porth, C. Xia, T. Hendrix, S.P. Moschou, & R. Keppens, 2014, ApJS 214, 4 (26pp) `doi:10.1088/0067-0049/214/1/4` .

promRT

This directory allows you to explore (recent) research into Rayleigh-Taylor dynamics in macroscopic prominence segments, using the ideal MHD assumption. In a stratified atmosphere, connecting chromosphere to corona, we have a prominence which is unstable to Rayleigh-Taylor instabilities. The `amrvacusr.t` contains many aspects for you to explore, namely: how we setup a stratified atmosphere for a given temperature distribution, and how we use the initial stratification also in the boundary conditions (top and bottom boundaries are treated special here, where the y-direction has gravity). The code is ok for both 2.5D or 3D runs, but obviously the 3D run, for simulations over long timescale, will need to be done overnight.

Check out how we activate a tracer to identify different parts (chromosphere, corona, prominence), and how in post-processing, we can use the user convert option to collect integrals over domain parts, to quantify information like the average temperature in some region, the volume and mass, etc.

The initial condition uses velocity pertubations with random phases, check out how this is to be done when running the parallel, block-adaptive code.

The boundary conditions ate top and bottom are instructive to see how extrapolations can be easily done on primitive variables. We have external gravity as a main physical ingedient here. Also, the log file is coded up very generally to store integrals over the domain.

The 3D setup also shows you how to generate on-the-fly collapsed views, along the three coordinate axes.

Related work is described in: 'Solar prominences: "double, double ... boil and bubble"', R. Keppens, X. Cia, & O. Porth, 2015, ApJ Letters 806, L13 (7pp). `doi:10.1088/2041-8205/806/1/L13`

lfff

This is a test to demonstrate how to generate a potential or linear force-free field extrapolation from a given (HMI) magnetogram. One can download such magnetograms from the SDO/HMI website, you then get these in FITS format. An IDL script `converthmi.pro` is provided, that assumes you have an IDL license and access to the library solar software (SSW) for SDO data handling. This routine just converts the FITS file into a (binary, uniform Cartesian grid) dat file, containing the vertical magnetic field component only. For convenience, an already converted magnetogram (in *dat) format is provided as well.

The user file shows how we use the LFFF module (see the `INCLUDE::amrvacmodules/lfff.t` statement) to then generate a potential or linear force-free (look for the `alpha` parameter) magnetic field extrapolation. You should set the code to a 3D isothermal MHD run (use the `-eos=iso` switch), and the parfile is set up to stop when the initial condition for a possible dynamical evolution is generated (i.e., the only thing that is computed here is the LFFF extrapolation, you should use Paraview to visualize the magnetogram and some selected fieldlines).

Note: to see fieldlines properly in Paraview, we need corner-valued data. This can be done directly on converting to vtu, by selecting e.g. `convert_type='vtuBmpi'`. You can also let the code dump the actual cell-centered values `convert_type='vtuBCCmpi'`, but then need to use a CelldatatoPointdata filter, before you can start drawing fieldlines. Another quirk of paraview is that it needs the Interpolator type to be set to `Interpolator with Cell locator`, before the Streamline tracer will be able to handle field lines across resolution changes (which we have in this 3-level AMR extrapolated field).

A description of this functionality and how the LFFF extrapolation works technically is found in (the references of) 'MPI-AMRVAC for solar and astrophysics', O. Porth, C. Xia, T. Hendrix, S.P. Moschou, & R. Keppens, 2014, ApJS 214, 4 (26pp) `doi:10.1088/0067-0049/214/1/4` .

pfss

This is a very similar exercise as the one above, but this time generating a Potential Field Source Surface model from a global magnetogram. It also has an IDL routine `convertsynoptic.pro` provided for converting downloadable synoptic maps to more easily handled maps of the radial field component, on a spherical grid, uniform in both spherical angles. This file is read in (and an example *dat for a specific date is given) by the user file (again with isothermal MHD).

Check the user file to see how we use the PFSS module (see the `INCLUDE::amrvacmodules/pfss.t` statement). The par file is set to stop after the field is generated. You can visualize again with Paraview. Note how we can use AMR to zoom in on specific active regions.

A description of this functionality and how the PFSS extrapolations compares to local cartesian extrapolations, can be found in 'MPI-AMRVAC for solar and astrophysics', O. Porth, C. Xia, T. Hendrix, S.P. Moschou, & R. Keppens, 2014, ApJS 214, 4 (26pp) `doi:10.1088/0067-0049/214/1/4` .

stressedIsland

This setup explores the explosive formation of a current sheet from an X-point configuration. We solve the equations of force-free magnetodynamics, valid for highly magnetised plasma. The overall evolution takes around one lightcrossing time. You can explore the impact of resistivity `(eqpar(kpar_))` on the reconnection rate and the width of the current sheet. There is quite a bit of run-time analysis coded up in `analysis.t`, e.g. it measures the reconnection rate and gives maximal values for the current and E^2/B^2.

Along with Maxwells equations, we push test-particles, each advanced due to the Lorentz force given by the electric and magnetic fields of the simulation. This particle-treatment is activated by defining in definitions.h:

```
#define PARTICLES
#define PARTICLES_LORENTZ
```

and by including the corresponding subroutines:

```
INCLUDE:amrvacmodules/handle_particles.t
INCLUDE:amrvacmodules/integrate_particles.t
```

In Paraview, you can load particle *.csv data and visualize them on top of the fluid data. Use the filter `TableToPoints` to convert the data first. Try to identify particles that show interesting evolution and follow them in the simulation. This is done by setting `follow(index) = .true.` in subroutine `init_particles` and re-running the simulation.

coalescenceInstability

Similarly to stressedIsland, this setup explores the formation of a current-sheet. Two flux tubes are slightly pushed together, starting the reconnection process at their interface. Since the currents in the tubes are parallel, the tubes begin to attract each other which speeds up the formation of the current-sheet, and the tubes merge into a single one.

This setup again uses force-free magnetodynamics, simulating infinitely magnetised plasma. Tearing instability sets in at Lundquist numbers of ~ 2000 and you can see the formation of plasmoids and saturation of the reconnection rate.

You can explore how the evolution depends on the perturbation `(eqpar(vcoll_))` and on the resistivity `(eqpar(kpar_))`. As in stressedIsland, 100 000 test-particles are advanced with the simulation and meaningful quantifications are done during runtime in `analysis.t`.

particleSnapshot

This example illustrates how to push particles through a static snapshot from an MHD simulation. The snapshot is taken from the coalescenceInstability with parameter `eqpar(kpar_)=4000` and 10000 particle-orbits are integrated with the Lorentz-force. Paraview lets you follow the particle orbits and do histograms of the particle energy. Note how particles accumulate in the central plasmoid where they reach high Lorentz factors. Following this example, you can adjust the setup to take output from any other MHD simulation.