Progresses on HYMAGYC IMASification

G. Vlad, S. Briguglio, V. Fusco, G. Fogaccia (ENEA Frascati)
D. Yadykin (Chalmers Univ., PSNC/IPPLM ACH (ACH-04))

TSVV #10 General meeting 2022.10.04

Progress in using HYMAGYC with IMAS-IDSs

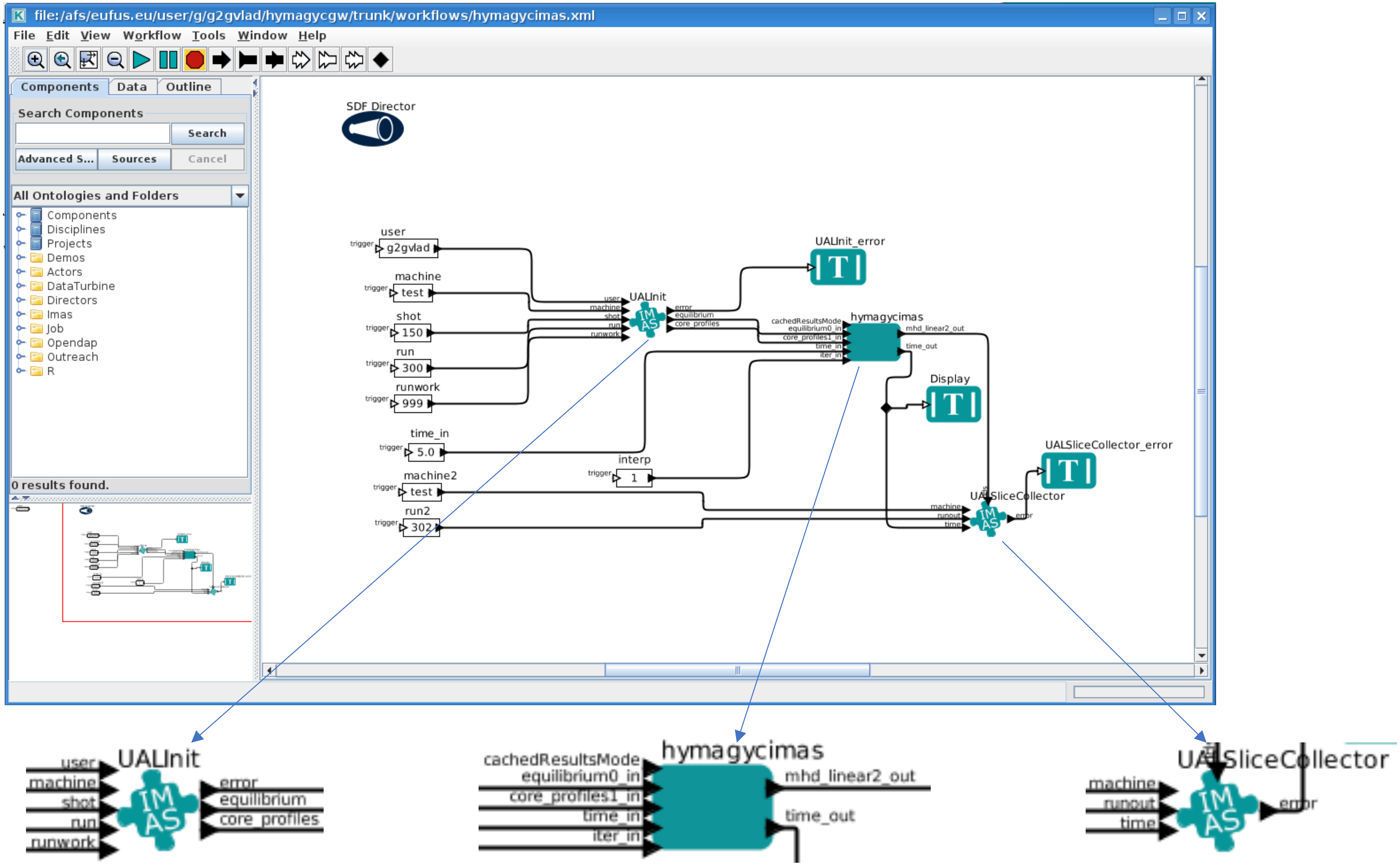
Short recaps on HYMAGYC structure:

- HYMAGYC is a hybrid MHD-Gyrokinetic code to study Energetic Particle
- Thermal (core) plasma described by linear full resistive MHD equations => arbitrary 2D equilibria (e.g., from CHEASE);
- Energetic particles population described by the nonlinear gyrokinetic Vlasov equation, $k_\perp \rho_E = O(1)$ limit, solved by particle-in-cell (PIC) techniques; flux coordinates system (s, χ, φ)
- The core of the field solver (linear full resistive MHD equations) originates from the code MARS [Bondeson, Vlad, Lütjens (IAEA Technical Committee Meeting on Advances in Simulations and Modelling of Thermonuclear Plasmas, Montreal, 1992, page 306, Vienna, Austria, 1993)], transformed from an eigensolver to an initial value code.
- Gyrokinetic contribution (fully nonlinear) enters via the divergence of the pressure tensor of the energetic particle species in the momentum equation.

- Original IMASification of HYMAGYC followed the IMASification of MARS, which is part of the EQSTABIL Workflow (EQSTABIL workflow for MHD equilibrium and stability chain analysis)

Progress in using HYMAGYC with IMAS-IDSs

Status at the end of 2021: minimal KEPLER Workflow



Progress in using HYMAGYC with IMAS-IDSs

UALInit actor - Universal Access Layer Initializator:

- open the Database, extract the required IDSs (equilibrium, core_profiles,…) and passes to the next actor

hymagycimas actor:

- it contains the actual code HYMAGYC prepared to receive the input data from the IMAS database;
- Code parameters entered via xml files;
- It transforms the input data to the actual data required by HYMAGYC to run (e.g., equilibrium geometry, metric tensor coefficients, initial conditions, change COCOS conventions, normalizations, etc.);
- Execute the Job;
- Forward the final output to the UALSliceCollector.

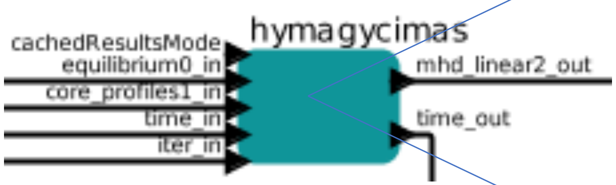
UALSliceCollector actor - Universal Access Layer actor

- Writes the output data to the Database;
- Close the Database.

Progress in using HYMAGYC with IMAS-IDSs

Although the minimal HYMAGYCIMAS Workflow works correctly, being able to execute, on the Gateway machine, a parallel job executed in batch, its structure for outputting data was not satisfying, being able to write them only at the end of the simulation (i.e., it was outputting only a snapshot of the last timestep of the simulation).

The obvious solution was to insert, within the time loop of the code, the suitable instructions to output snapshots of the simulation at (assigned intermediate) times during the run:




```
do it=1,nsteps
      call solve_fields(…)
      call gyrokinetics(…)
      if ((it/nprint)*nprint.eq.it) then
            call ids_put_slice(…)
      endif
enddo
```

Trivial solution but it requires the “propagation” of the IDSs DataStructure within the kernel of the code…

Developments/updatings of the Workflow using a fortran driver which calls the hymgycimas library (more agile environment than Kepler…)

Progress in using HYMAGYC with IMAS-IDSs

- The IDS used for storing MHD like data (field solver part) is the mhd_linear one (Magnetohydrodynamic linear stability).
- Multiple calls inside the time loop:

  ```
  call ids_put_slice(idx_o,"mhd_linear",mhd_out_ids,retstatus)
  ```

  results in “appending” each time slice to the previous ones:
- Here is an example excerpt of an IDS output containing 10 time slice (every 100 time-steps, nprint=100) (using idsdump command):

class mhd_linear
⋮
Attribute time_slice
    class time_slicedynamic
    time_slicedynamic[0] =
    … datastructure fields… Physical quantities at t=0.
    time_slicedynamic[1] =
    … datastructure fields… Physical quantities at t=2.
    ⋮
    time_slicedynamic[10] = *Physical quantities at t=20.*
    … datastructure fields…
Attribute time
[0. 2. 4. 6. 8. 10. 12. 14. 16. 18. 20.]

Progress in using HYMAGYC with IMAS-IDSs

- To avoid to output, at each time slice, quantities which are constant in time, we have reduced at minimum the definitions of these quantities in the mhd_linear ids;
- Here is a (non exhaustive) list of MHD quantities filled at each output time step:

```
type(ids_mhd_linear)      :: mhd_out_ids
mhd_out_ids%time_slice(1)%toroidal_mode(1)%n_tor=int(rn(2))
mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%grid%dim2(j)=rm(j,2)
mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%grid%dim1(ii1)=cs(ii0)
mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%coordinate_system%r(ii1,j)=r0exp*my_r(ii0,j)
mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%coordinate_system%z(ii1,j)=r0exp*my_z(ii0,j))
mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%tau_alfven(ii1)
mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%velocity_perturbed%coordinate1%real
mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%velocity_perturbed%coordinate1%imaginary
mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%velocity_perturbed%coordinate2%real
mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%velocity_perturbed%coordinate2%imaginary
mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%velocity_perturbed%coordinate3%real
mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%velocity_perturbed%coordinate3%imaginary
mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%b_field_perturbed%coordinate1%real
mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%b_field_perturbed%coordinate1%imaginary
mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%b_field_perturbed%coordinate2%real
mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%b_field_perturbed%coordinate2%imaginary
mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%b_field_perturbed%coordinate3%real
mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%b_field_perturbed%coordinate3%imaginaryreal
mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%pressure_perturbed%real
mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%pressure_perturbed%imaginary
```

Annotations (in order of the lines above, from n_tor onward):

- *toroidal mode number n*
- *poloidal mode numbers m*
- *$s \propto$ sqrt(psi) (radial mesh)*
- $R(s,\chi)$
- $Z(s,\chi)$
- Alfvén time $\tau_A(s)$
- *$v^s_{m,n}$ (Real contravariant perturbed velocity)*
- *$v^s_{m,n}$ (Imm. …)*
- *$v^\chi_{m,n}$ (Real contr. pert. velocity)*
- *$v^\chi_{m,n}$ (Imm. …)*
- *$v^\varphi_{m,n}$ (Real contr. pert velocity*
- *$v^\varphi_{m,n}$ (Imm. …)*
- $b^s_{m,n}$
- $b^s_{m,n}$
- $b^\chi_{m,n}$
- $b^\chi_{m,n}$
- $b^\varphi_{m,n}$
- $b^\varphi_{m,n}$
- $p_{m,n}$
- $p_{m,n}$

Progress in using HYMAGYC with IMAS-IDSs

- Here is a (non exhaustive) list of MHD quantities filled at each output time step (cont'd):

mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%a_field_perturbed%coordinate1%real $A_{s;m,n}$ *(Real covariant perturbed vector potential)*

mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%a_field_perturbed%coordinate1%imaginaryreal $A_{s;m,n}$ *(Imm….)*

mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%a_field_perturbed%coordinate2%real $A_{\chi;m,n}$ *(Real covariant perturbed vector potential)*

mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%a_field_perturbed%coordinate2%imaginary $A_{\chi;m,n}$ *(Imm….)*

mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%a_field_perturbed%coordinate3%real $A_{\varphi;m,n}$ *(Real covariant perturbed vector potential)*

mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%a_field_perturbed%coordinate3%imaginary $A_{\varphi;m,n}$ *(Imm….)*

mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%phi_potential_perturbed%real $\phi_{m,n}$ *(Real perturbed scalar potential)*

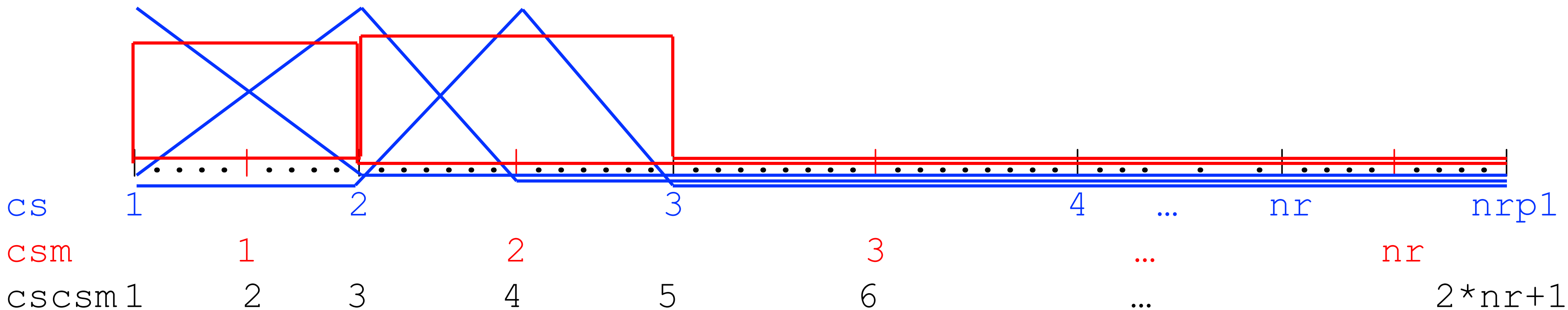
mhd_out_ids%time_slice(1)%toroidal_mode(1)%plasma%phi_potential_perturbed%imaginary $\phi_{m,n}$ *(Imm….)*

Progress in using HYMAGYC with IMAS-IDSs

- Radial independent variable used in HYMAGYC (similarly to MARS): s ∝ sqrt(psi) [0.,…,1]

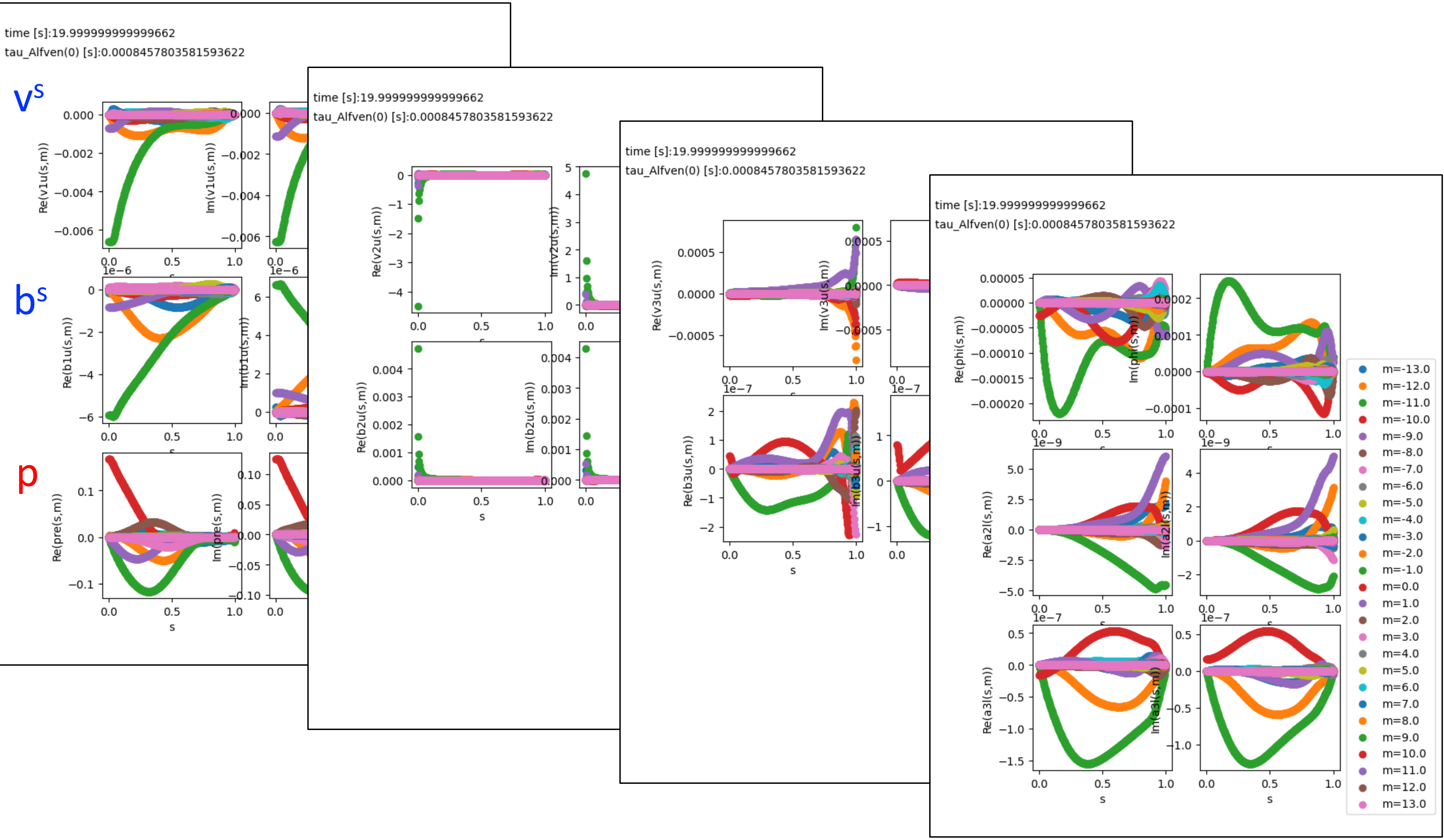
- The algorithm used in the field solver section of HYMAGYC (similarly to MARS) requires two different radial meshes, the so-called integer mesh (cs) and the half-integer mesh (csm), defined as:

csm(i)=[cs(i)+cs(i+1)]/2.

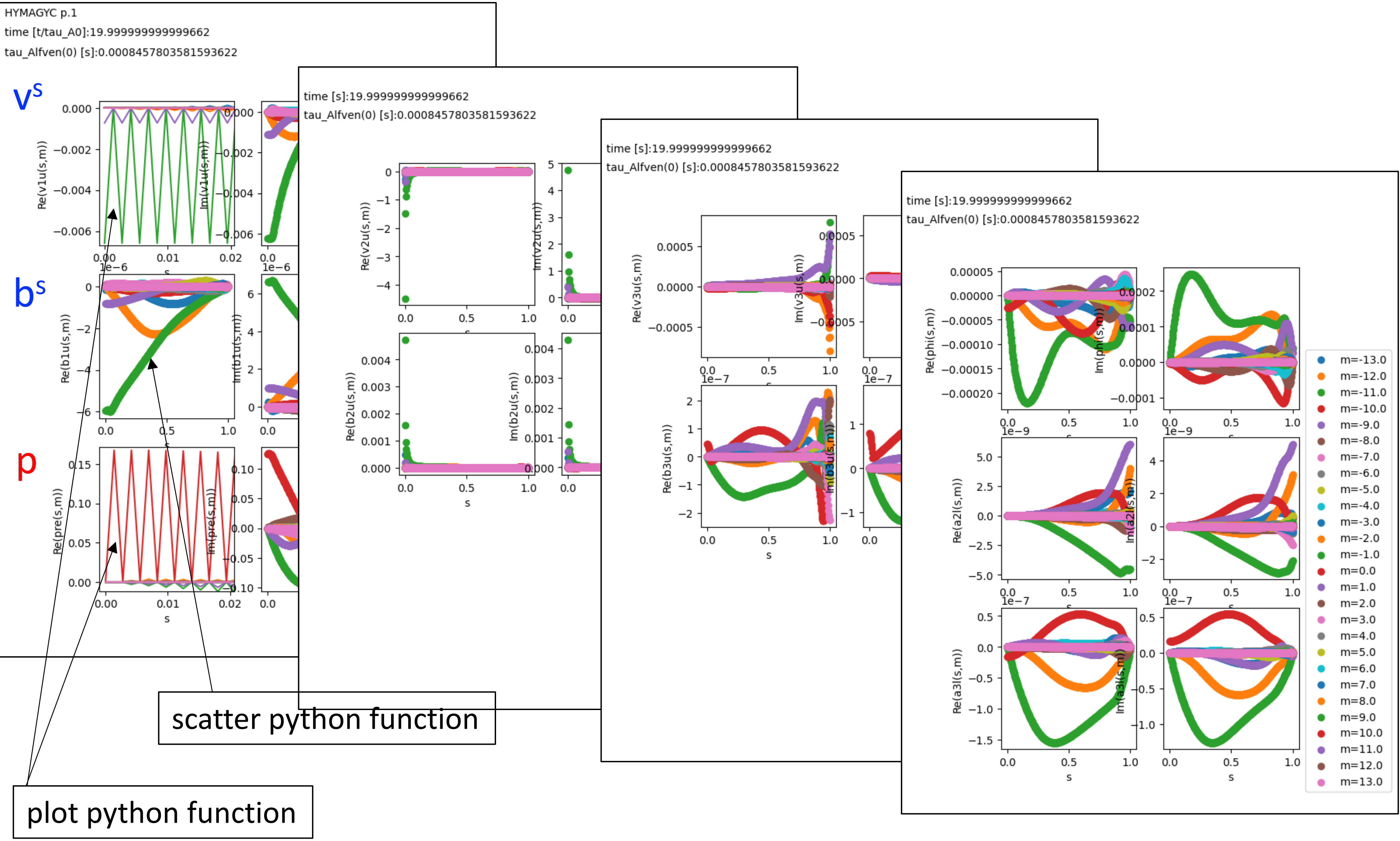


- Radial discretization uses two different meshes (integer and half-integer), corresponding to two different choices of Finite Elements (piecewise linear and piecewise constant)
- different physical quantities are defined on different meshes (or using different FE functions)
- E.g., v^s, v^χ, v^φ, b^s, b^χ, b^φ, j^s, j^χ, j^φ, p.
- To preserve this information in the mhd_linear IDS, we have defined all the physical quantities on a generalized radial mesh, obtained by the union of the cs and csm meshes: cscsm=cs ∪ csm, setting conventionally to “zero” the value of the quantities on the grid points to which they do not belong to.

Progress in using HYMAGYC with IMAS-IDSs



Progress in using HYMAGYC with IMAS-IDSs



Progress in using HYMAGYC with IMAS-IDSs

IMASification of HYMAGYC is proceeding (thanks to Dmitro Yadykin…!):

- basics for the fluid part almost ready (few quantities are still missing in the mhd_linear IDS)
- missing dump&restart option under IMAS, I would suggest to postpone such issue…;
- gyrokinetic output still to be considered, we plan with Dmitro to have a survey on the basilar quantities to be stored shortly;
- an issue would be what to do for the post processing software and graphics (which is almost essential to interpret the results of a run);
- Transition from SVN to GITLAB to be done urgently…

Recent upgrading and applications of HYMAGYC

- benchmark between HYMAGYC, MEGA, ORB5: nonlinear simulations for AUG-NLED case, off-axis energetic particle profile (activity carried on also within the ENR-ATEP project)
- Thermal ions treated gyrokinetically (TSVV#2)