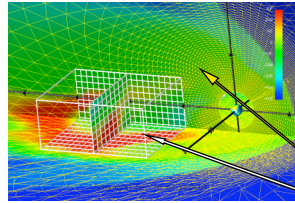


## Introduction

**Overture** is a set of C++ classes providing two main functionalities: 1) A high level programming environment for writing PDE solvers in complex geometries that hide intrinsic details of the numerical algorithms from the user yet leaving the possibility of low level control, and 2) Interpolation between overset grids (moving and static). **InterComm** is a runtime library (with Fortran, C, and C++ interfaces) that provides means for controlled data transfers between separate running executables and easy (automatic) access to distributed data on both ends of the communication channel. We show examples of Overture PDE solving capabilities applied to the plasma sheet and ionosphere, as well as interpolation and data transmission (InterComm) between these codes and the Lyon-Fedder-Mobarry global MHD model.

## Embedded Plasma Sheet Code



The code uses the PDE solving capabilities of Overture to solve the 3 dimensional ideal MHD equations in a cartesian grid embedded within a volume of the LFM

Stretched spherical LFM grid

Regular cartesian Plasma Sheet grid

$$\frac{\partial}{\partial t} \begin{bmatrix} \rho \\ \rho \mathbf{u} \\ \mathbf{B} \\ E \end{bmatrix} + \nabla \cdot \begin{bmatrix} \rho \mathbf{u} \\ \rho \mathbf{u} \mathbf{u} + I \left( p + \frac{1}{2} B^2 \right) - \mathbf{B} \mathbf{B} \\ \mathbf{u} \mathbf{B} - \mathbf{B} \mathbf{u} \\ \left( E + p + \frac{1}{2} B^2 \right) \mathbf{u} - \mathbf{B} (\mathbf{u} \cdot \mathbf{B}) \end{bmatrix} = 0$$

$\mathbf{u}$   
(state vector)
 $\mathbf{F}$   
(flux vector)

The conservative formulation of the ideal MHD equations (shown at left) are solved to 4<sup>th</sup> order spatial accuracy using the Overture tools. Spurious spatial oscillations are controlled by a 4th order dissipation term explicitly added to all equations.

4<sup>th</sup> order Adams-Bashforth time marching scheme in 1D:

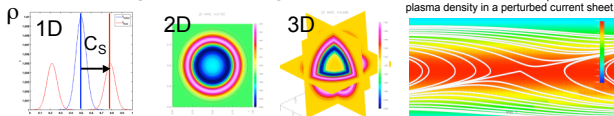
$$u^{n+1} = u^n - \left( 55 \frac{\partial}{\partial x} F_x^n - 59 \frac{\partial}{\partial x} F_x^{n-1} + 37 \frac{\partial}{\partial x} F_x^{n-2} - 9 \frac{\partial}{\partial x} F_x^{n-3} \right) \frac{\Delta t}{24}$$

Overture code implementing above scheme:

$$u[n+1] = u[n] - ( 55 * Fx[n].x() - 59 * Fx[n-1].x() + 37 * Fx[n-2].x() - 9 * Fx[n-3].x() ) * (dt/24)$$

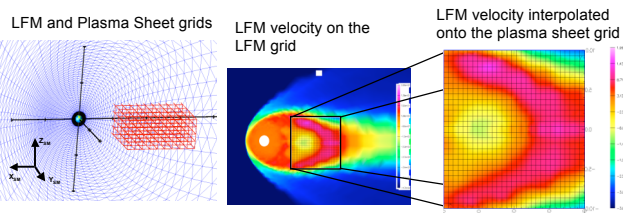
The code uses a 4<sup>th</sup> order Adams-Bashforth time marching scheme to advance the MHD state vector  $\mathbf{u}$ . Note the ease of taking derivatives in the code using the Overture classes.

## Code Testing/Benchmarking results



Classic computational tests: Wave Propagation Speed, and the GEM reconnection challenge.

## Overlapping Plasma Sheet and LFM grids

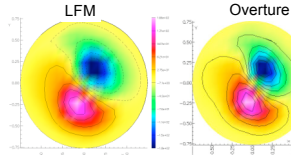


## Ionospheric Solver

The solver solves the usual Poisson's equation in the high latitude ionosphere:

$$\nabla_{\perp} \cdot (\hat{\Sigma} \cdot \nabla_{\perp} \Phi) = j_{\parallel}$$

### Solution for $\Phi$



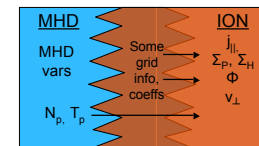
• LFM grid is in the HDF database  
 •  $\Sigma_p$  and  $\Sigma_H$  are calculated locally (according to Fedder et al., 1995) or by an ITM model (TIEGCM) and transmitted via InterComm.  
 •  $J_{\parallel}$  transmitted via InterComm from LFM

```

Excerpts from the code:
/* Get the previously created LFM grid */
CompositeGrid lfmIonGrid;
GetFromDataBase(lfmIonGrid, "lfm_ion_grid.hdf");
// Specify equation */
scalar1 = sigPsin(thetaGrid)/sinDelta/sinDelta;
scalar3 = sigH/sinDelta;
coeff1 = op.derivativeScalarDerivativeCoefficients(scalar1,0,0);
coeff2 = op.derivativeScalarDerivativeCoefficients(sigP,1,1);
coeff3 = op.derivativeScalarDerivativeCoefficients(sigH,0,1);
coeff4 = op.derivativeScalarDerivativeCoefficients(sigH,1,0);
coeff = multiply(sin(thetaGrid),coeff1) + coeff2
// Pedersen
f = jpar*op(sin(thetaGrid),coeff3) - multiply(sin(thetaGrid),coeff4); //Hall
f = jpar*op(sin(thetaGrid),coeff3) + sin(thetaGrid,2)*sinDelta;
// right-hand side
/* Apply boundary conditions */
coeff.applyBoundaryConditionCoefficients(0,0,BCTypes::dirichlet,BCTypes::allBoundaries);
coeff.applyBoundaryConditionCoefficients(0,0,BCTypes::extrapolate,BCTypes::allBoundaries);
// Solve */
Oges solver(lfmIonGrid);
solver.setCoefficientArray(coeff);
solver.set(OgesParameters::THEsolverType,OgesParameters::SLAP);
solver.set(OgesParameters::gmes);
solver.set(OgesParameters::THEtolerance,1.e-6);
solver.solve(psi,f);
    
```

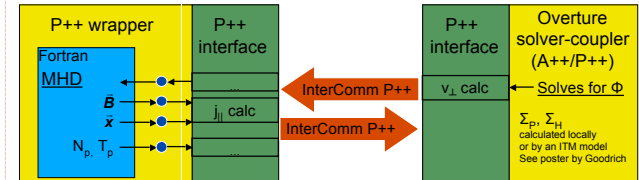
## Building interfaces for Coupling

### Conventional LFM model



• Some grid and other geometric quantities are calculated in the "gray" area: the codes cannot be decoupled  
 • Ionospheric module calculates parallel current, conductivities, solves for the potential, and calculates the electric field at the inner MHD boundary: usurps the role of interfaces

### Modular MHD-Ionosphere solver/coupler interfaces based on Overture and InterComm



### High-level InterComm interface:

```

LFM:
/* Create a communication EndPoint */
int localTasks = 1, remoteTasks = 1;
int ic_err;
IC_EndPoint lfm_ion("lfm:ion", localTasks,
remoteTasks, IC_EndPoint::IC_COLUMN_MAJOR,
IC_EndPoint::IC_COLUMN_MAJOR, ic_err);

/* Export array */
doubleArray jPara(nj,nk);
lfm_ion.exportArray(jPara,ic_err);

Ionosphere:
/* Create a communication EndPoint */
int localTasks = 1, remoteTasks = 1;
int ic_err;
IC_EndPoint ion_lfm("ion:lfm", localTasks,
remoteTasks, IC_EndPoint::IC_COLUMN_MAJOR,
IC_EndPoint::IC_COLUMN_MAJOR, ic_err);

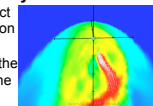
/* Import array */
doubleArray jPara;
ion_lfm.importArray(jPara,ic_err);
    
```

\*note that the parallel array descriptors are automatically extracted by InterComm from P++ arrays. The user does not need to have any a priori information about data distribution.

## Future Work

### Plasma Sheet Code

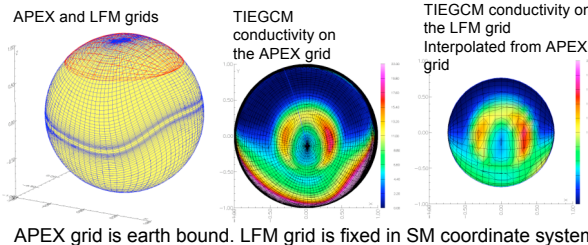
- **Add Hall Term**
  - Test with whistler wave and GEM & Newton Reconnection Challenge Initial Conditions.
- **Implement coupling to the LFM**
  - Pass data between two codes using InterComm library.
  - Run coupled code in 1-way and 2-way tests.
- **Flow Channel Analysis**
  - Measure the effect the Hall term has on the size of flow channels seen in the plasma sheet of the LFM.



### Ionosphere

- **Ionospheric Module Interface**
  - The interface on the MHD side is already implemented. The interface in the ionospheric module is in development.
- **Test**
  - Test the resulting modular coupled code against the conventional LFM.
- **Couple to TIE-GCM**
  - The LFM ionospheric conductance model has been moved to the new ionospheric module. The next step is to implement the coupling to an ITM model (TIE-GCM).

## Overlapping APEX and LFM grids



APEX grid is earth bound. LFM grid is fixed in SM coordinate system