TECH-X SIMULATIONS EMPOWERING YOUR INNOVATIONS

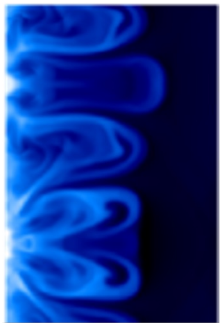
Introduction to USim

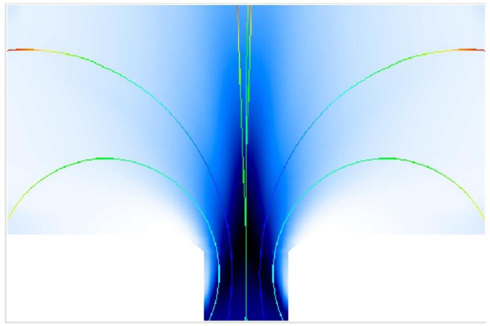
Tech-X Corporation
5621 Arapahoe Avenue, Suite A
Boulder, CO 80303

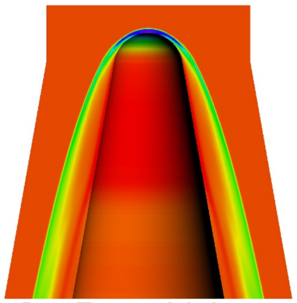
USim: Fluid, Plasma and Electromagnetic Modeling on Unstructured Meshes

- Supports hydrodynamics, magnetohydrodynamics, Hall magnetohydrodynamics, two-fluid plasmas, Navier-Stokes and Maxwell's equations
- Includes multi-species, multi-temperature versions of fluid models

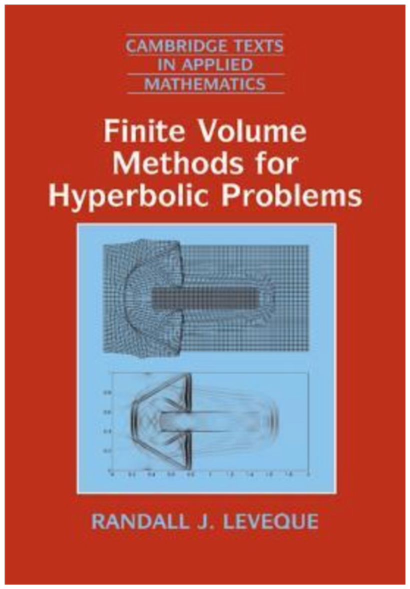
Dense Plasma Focus


Z-Pinch


Magnetic Nozzles

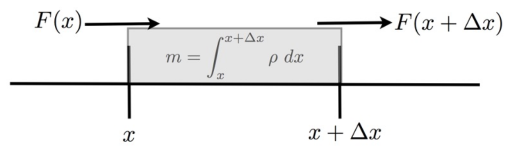

Re-Entry Vehicles

USim solves flux-conservative equation sets using Finite Volume algorithms


CAMBRIDGE TEXTS IN APPLIED MATHEMATICS
Finite Volume Methods for Hyperbolic Problems
RANDALL J. LEVEQUE

Many of the USim algorithms are described in detail in this book

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot [\mathcal{F}(\mathbf{w})] = 0$$



As an example, Ideal MHD:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot [\rho \, \mathbf{u}] = 0$$

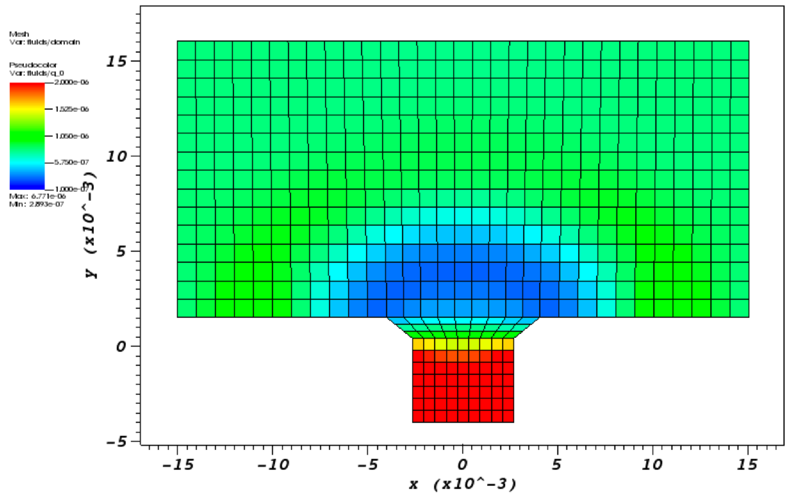
$$\frac{\partial \rho \, \mathbf{u}}{\partial t} + \nabla \cdot \left[\rho \, \mathbf{u} \, \mathbf{u}^T - \mathbf{b} \, \mathbf{b}^T + \mathbb{I}\left(P + \tfrac{1}{2} \, |\mathbf{b}|^2\right)\right] = 0$$

$$\frac{\partial E}{\partial t} + \nabla \cdot [(E + P)\mathbf{u} + \mathbf{e} \times \mathbf{b}] = 0$$

$$\frac{\partial \mathbf{b}^{\text{plasma}}}{\partial t} + \nabla \times \mathbf{e} + \nabla \psi = 0$$

$$\frac{\partial \psi}{\partial t} + \nabla \cdot [c_{\text{fast}}^2 \, \mathbf{b}] = 0$$

USim models are composed of the following building blocks



- Defining the Simulation Grid
- Allocating Simulation Memory
- Initializing the Fluid
- Evolving the Fluid
- Applying Boundary Conditions
- Advancing By A Time Step

Generic .pre File

```
$ USER_VARIABLES

tStart = 0.0
tEnd = $TEND$
numFrames = NUMDUMPS
initDt = 0.1

<Component fluids>

  <Grid domain>
  </Grid>

  <DataStruct q>
  </DataStruct>

  <Updater init>
  </Updater>

  <UpdateStep initStep>
     updaters = [init]
  </UpdateStep>

  <UpdateSequence sequence>
    startOnly = [initStep]
    loop = [ ]
  </UpdateSequence>

</Component>
```

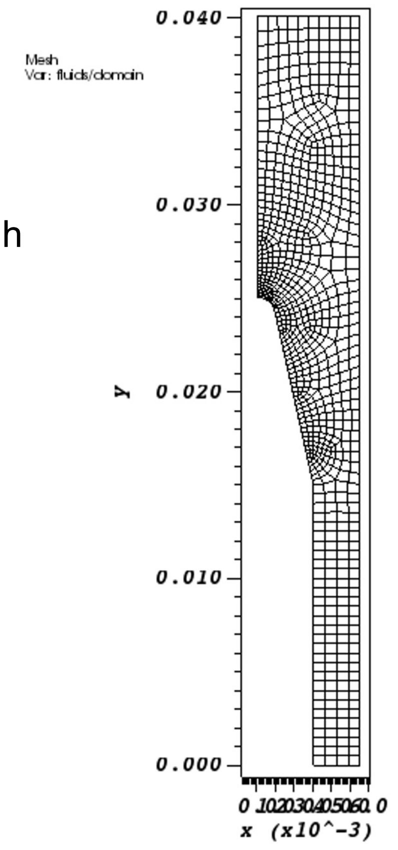
- Defining the Simulation Grid
- Allocating Simulation Memory
- Initializing the Fluid
- *Evolving the Fluid*
- *Applying Boundary Conditions*
- *Advancing By A Time Step*

$\}$ *Coming Later*

Generic .pre File

```
$ USER_VARIABLES

tStart = 0.0
tEnd = $TEND$
numFrames = NUMDUMPS
initDt = 0.1

<Component fluids>

  <Grid domain>
  </Grid>

  <DataStruct q>
  </DataStruct>

  <Updater init>
  </Updater>

  <UpdateStep initStep>
     updaters = [init]
  </UpdateStep>

  <UpdateSequence sequence>
    startOnly = [initStep]
    loop = [ ]
  </UpdateSequence>

</Component>
```

Mesh



Generic .pre File

```
$ USER_VARIABLES

tStart = 0.0
tEnd = $TEND$
numFrames = NUMDUMPS
initDt = 0.1

<Component fluids>

  <Grid domain>
  </Grid>

  <DataStruct q>
  </DataStruct>  ←

  <Updater init>
  </Updater>

  <UpdateStep initStep>
    updaters = [init]
  </UpdateStep>

  <UpdateSequence sequence>
    startOnly = [initStep]
    loop = [ ]
  </UpdateSequence>

</Component>
```

Equation variables

$$q = \begin{bmatrix} \rho \\ \rho u_x \\ \rho u_y \\ \rho u_z \\ E \\ b_x \\ b_y \\ b_z \\ \psi \\ S_e \end{bmatrix}$$

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot [\mathcal{F}(\mathbf{w})] = 0$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot [\rho\, \mathbf{u}] = 0$$

$$\frac{\partial \rho\, \mathbf{u}}{\partial t} + \nabla \cdot \left[\rho\, \mathbf{u}\, \mathbf{u}^T - \mathbf{b}\, \mathbf{b}^T + \mathbb{I}\left(P + \frac{1}{2}\, |\mathbf{b}|^2\right)\right] = 0$$

$$\frac{\partial E}{\partial t} + \nabla \cdot [(E + P)\mathbf{u} + \mathbf{e} \times \mathbf{b}] = 0$$

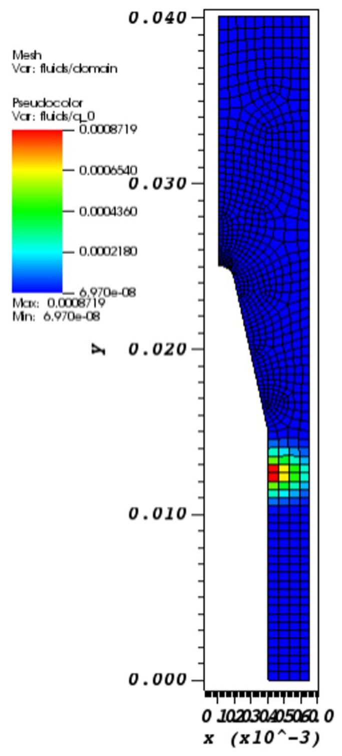
$$\frac{\partial \mathbf{b}^{\text{plasma}}}{\partial t} + \nabla \times \mathbf{e} + \nabla \psi = 0$$

$$\frac{\partial \psi}{\partial t} + \nabla \cdot \left[c_{\text{fast}}^2\, \mathbf{b}\right] = 0$$

Generic .pre File

```
$ USER_VARIABLES

tStart = 0.0
tEnd = $TEND$
numFrames = NUMDUMPS
initDt = 0.1

<Component fluids>

  <Grid domain>
  </Grid>

  <DataStruct q>
  </DataStruct>

  <Updater init>
  </Updater>

  <UpdateStep initStep>
     updaters = [init]
  </UpdateStep>

  <UpdateSequence sequence>
    startOnly = [initStep]
    loop = [ ]
  </UpdateSequence>

</Component>
```

$\leftarrow$ Initial conditions



Condensed DPF .pre File

```
$ USER_VARIABLES

tStart = 0.0
tEnd = $TEND$
numFrames = NUMDUMPS
initDt = 0.1

<Component fluids>

 <Grid domain>
 </Grid>

 <DataStruct q>
 </DataStruct>

 <Updater init>
 </Updater>

 <Updater hyper>
  kind = unstructMuscl2d
  <Equation mhd>
   kind = twoTemperatureMhdDednerEqn
  </Equation>
  <Source resistivity>
  </Source>
 </Updater>

 <Updater timeUpdater>
  <TimeIntegrator rkUpdater>
   kind = rungeKutta2d
  </TimeIntegrator>
 </Updater>

<Updater timeRestriction>
  kind = timeStepRestrictionUpdater2d
 </Updater>

 <Updater boundaryCondition>
  kind = mhdBc2d
 </Updater>

 <UpdateStep initStep>
   updaters = [init,initResistivity,initChargeState,getHypDT]
 </UpdateStep>

 <UpdateStep hyperStep>
  updaters = [timeRestriction,timeUpdater]
 </UpdateStep>

 <UpdateStep copyStep>
   updaters = [boundaryConditions]
 </UpdateStep>

 <UpdateSequence sequence>
  startOnly = [initStep]
  loop = [hyperStep, copyStep]
 </UpdateSequence>

</Component>
```

TECH-X SIMULATIONS EMPOWERING YOUR INNOVATIONS

User Defined Variables

Start by defining any variables you would like to use later in the file

```
$ OHMIC_RESISTIVITY = 1.0e-6
$ ION_CHARGE_STATE = 10.0
$ ATOMIC_WEIGHT = 6.94
$ ALPHA = 0.0001
$ BETA = 0.0001
$ ION_TEMP = 11.6e3

$ CURRENT = 15.0e3

$ CFL = 0.4
$ NUMDUMPS = 1
$ TEND = 1.0e-16
```

```
$ USER_VARIABLES

tStart = 0.0
tEnd = $TEND$
numFrames = NUMDUMPS
initDt = 0.1

<Component fluids>

  <Grid domain>
  </Grid>

  <DataStruct q>
  </DataStruct>

  <Updater init>
  </Updater>

  <Updater hyper>
    kind = unstructMuscl2d
    <Equation mhd>
      kind = twoTemperatureMhdDednerEqn
    </Equation>
    <Source resistivity>
    </Source>
  </Updater>

  <Updater timeUpdater>
    <TimeIntegrator rkUpdater>
      kind = rungeKutta2d
    </TimeIntegrator>
  </Updater>
```

Required Top-Level Parameters

These define how long the simulation will run and how much output to create

```
tStart = 0.0
tEnd = $TEND$
numFrames = NUMDUMPS
initDt = 0.1
```

```
$ USER_VARIABLES

tStart = 0.0
tEnd = $TEND$
numFrames = NUMDUMPS
initDt = 0.1

<Component fluids>

  <Grid domain>
  </Grid>

  <DataStruct q>
  </DataStruct>

  <Updater init>
  </Updater>

  <Updater hyper>
    kind = unstructMuscl2d
    <Equation mhd>
      kind = twoTemperatureMhdDednerEqn
    </Equation>
    <Source resistivity>
    </Source>
  </Updater>

  <Updater timeUpdater>
    <TimeIntegrator rkUpdater>
      kind = rungeKutta2d
    </TimeIntegrator>
  </Updater>
```

Initial Setup Blocks

Everything is inside a single Component block

```
<Component fluids>
  kind = updaterComponent
```

Your unstructured grid will define the domain size

```
  <Grid domain>
    file = myMeshFile.msh
  </Grid>
```

Create as many data structures as desired. Some are determined based on the type of equations you are solving, such as q. Others you can create purely for visualization or post-processing purposes, such as Temperature

```
  <DataStruct q>
    numComponents = 10
  </DataStruct>
```

Set initial conditions on any of the DataStruct

```
  <Updater init>
    exprs = ["density", "0.0", "0.0", "0.0", "pressure/(gamma-1.0)", "0.0","0.0","0.0","0.0","electronEntropy"]
  </Updater>
```

```
$ USER_VARIABLES

tStart = 0.0
tEnd = $TEND$
numFrames = NUMDUMPS
initDt = 0.1

<Component fluids>

  <Grid domain>
  </Grid>

  <DataStruct q>
  </DataStruct>

  <Updater init>
  </Updater>

  <Updater hyper>
    kind = unstructMuscl2d
    <Equation mhd>
      kind = twoTemperatureMhdDednerEqn
    </Equation>
    <Source resistivity>
    </Source>
  </Updater>

  <Updater timeUpdater>
    <TimeIntegrator rkUpdater>
      kind = rungeKutta2d
    </TimeIntegrator>
  </Updater>
```

Equation Blocks

What type of equation is being solved (Equation kind), and what scheme should we use to solve that equation (Updater kind)? Also, if there are any "source" terms on the equation, such as resistivity, put those here.

```
 <Updater hyper>
   kind = unstructMuscl2d

   <Equation mhd>
     kind = twoTemperatureMhdDednerEqn
   </Equation>

   <Source resistivity>
   </Source>

 </Updater>
```

```
$ USER_VARIABLES

tStart = 0.0
tEnd = $TEND$
numFrames = NUMDUMPS
initDt = 0.1

<Component fluids>

 <Grid domain>
 </Grid>

 <DataStruct q>
 </DataStruct>

 <Updater init>
 </Updater>

 <Updater hyper>
   kind = unstructMuscl2d
   <Equation mhd>
     kind = twoTemperatureMhdDednerEqn
   </Equation>
   <Source resistivity>
   </Source>
 </Updater>

 <Updater timeUpdater>
   <TimeIntegrator rkUpdater>
     kind = rungeKutta2d
   </TimeIntegrator>
 </Updater>
```

Time Updater Blocks

Update in time. Here the rungeKutta scheme is used.
The UpdateSequence tells what order to do the updates in.

```
 <Updater rkUpdater>

   <TimeIntegrator rkUpdater>
    kind = rungeKutta2d
   </TimeIntegrator>

   <UpdateSequence sequence>
    loop = [boundaries,current,hyper]
   </UpdateSequence>

   <UpdateStep boundaries>
     updaters = [bcOpen, bcWall, bcOpen1,bcSource, bcAxis]
   </UpdateStep>

   <UpdateStep current>
     updaters = [setMagneticField,computeCurrent]
   </UpdateStep>

   <UpdateStep hyper>
     updaters = [computeEField,hyper]
   </UpdateStep>

 </Updater>
```

```
$ USER_VARIABLES

tStart = 0.0
tEnd = $TEND$
numFrames = NUMDUMPS
initDt = 0.1

<Component fluids>

 <Grid domain>
 </Grid>

 <DataStruct q>
 </DataStruct>

 <Updater init>
 </Updater>

 <Updater hyper>
  kind = unstructMuscl2d
  <Equation mhd>
   kind = twoTemperatureMhdDednerEqn
  </Equation>
  <Source resistivity>
  </Source>
 </Updater>

 <Updater timeUpdater>
  <TimeIntegrator rkUpdater>
   kind = rungeKutta2d
  </TimeIntegrator>
 </Updater>
```

Time Step Restriction Blocks

USim will calculate a time step, however, if there are any further physics restrictions on the time step, find those here and pass the information back to the "hyper" updater.

```
<Updater getHypDT>
  kind = timeStepRestrictionUpdater2d

  <TimeStepRestriction idealMhd>
    kind = hyperbolic2d
  </TimeStepRestriction>

  <TimeStepRestriction quadratic>
    kind = quadratic2d
  </TimeStepRestriction>

</Updater>
```

```
<Updater timeRestriction>
  kind = timeStepRestrictionUpdater2d
</Updater>

<Updater boundaryCondition>
  kind = mhdBc2d
</Updater>

<UpdateStep initStep>
   updaters = [init,initResistivity,initChargeState,getHypDT]
</UpdateStep>

<UpdateStep hyperStep>
  updaters = [timeRestriction,timeUpdater]
</UpdateStep>

<UpdateStep copyStep>
   updaters = [boundaryConditions]
</UpdateStep>

<UpdateSequence sequence>
  startOnly = [initStep]
  loop = [hyperStep, copyStep]
</UpdateSequence>

</Component>
```

TECH-X SIMULATIONS EMPOWERING YOUR INNOVATIONS

Boundary Condition Blocks

What boundary conditions are applied. Create as many as necessary.

```
<Updater bcOpen>
  kind = mhdBc2d
</Updater>
```

```
<Updater timeRestriction>
  kind = timeStepRestrictionUpdater2d
</Updater>

<Updater boundaryCondition>
  kind = mhdBc2d
</Updater>

<UpdateStep initStep>
   updaters = [init,initResistivity,initChargeState,getHypDT]
</UpdateStep>

<UpdateStep hyperStep>
  updaters = [timeRestriction,timeUpdater]
</UpdateStep>

<UpdateStep copyStep>
   updaters = [boundaryConditions]
</UpdateStep>

<UpdateSequence sequence>
  startOnly = [initStep]
  loop = [hyperStep, copyStep]
</UpdateSequence>

</Component>
```

Update Sequence

Create as many updateSteps as necessary. These are then passed to an UpdateSequence which will do them in order.

```
  <UpdateStep initStep>
     updaters = [init,initResistivity,initChargeState,getHypDT]
  </UpdateStep>

  <UpdateStep hyperStep>
    updaters = [getHypDT,rkUpdater]
  </UpdateStep>

  <UpdateStep copyStep>
     updaters = [copier,bcOpen, bcWall, bcOpen1,bcSource, bcAxis,computeMagEnergy1,
computeTotalMagEnergyChange, computeTotalMagEnergy,  computeVoltage]
     syncVars = [q, qnew]
  </UpdateStep>

  <UpdateSequence sequence>
    startOnly = [initStep]
    loop = [hyperStep, copyStep]
    writeOnly = [pressureStep]
  </UpdateSequence>
```

```
<Updater timeRestriction>
  kind = timeStepRestrictionUpdater2d
</Updater>

<Updater boundaryCondition>
  kind = mhdBc2d
</Updater>

<UpdateStep initStep>
   updaters = [init,initResistivity,initChargeState,getHypDT]
</UpdateStep>

<UpdateStep hyperStep>
  updaters = [timeRestriction,timeUpdater]
</UpdateStep>

<UpdateStep copyStep>
   updaters = [boundaryConditions]
</UpdateStep>

<UpdateSequence sequence>
  startOnly = [initStep]
  loop = [hyperStep, copyStep]
</UpdateSequence>

</Component>
```