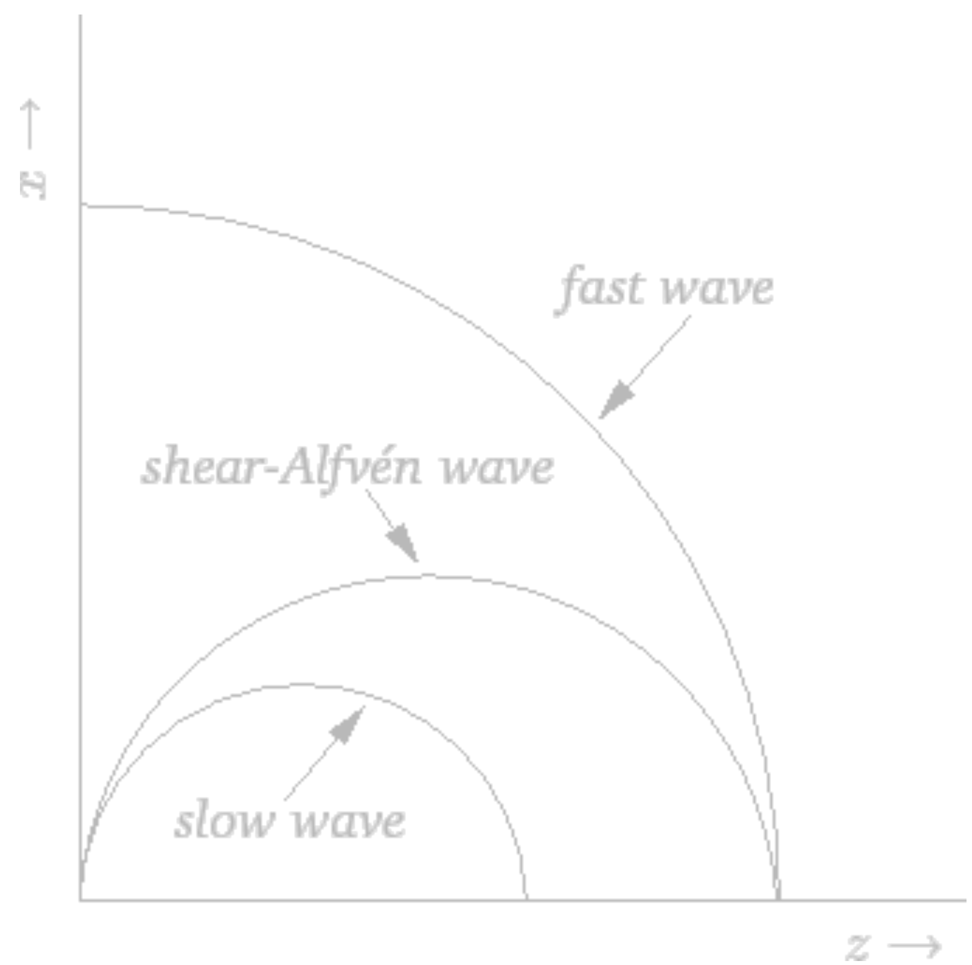
Topic 1

How to Write a 1-D MHD Code using Finite Difference?



Outline

- Review of 1-D MHD equations and normalization
- Initial and Boundary Conditions
- Put everything together
- Once through the mhd.m code

1-D MHD equations

Plasma variables

$$\rho, u_x, u_y, p$$

Field variables

$$B_y, E_z, J_z$$

Put the equations together:

$$\frac{\partial}{\partial t}\rho = -\nabla\cdot\rho\mathbf{u}$$

$$\rho\frac{D\mathbf{u}}{Dt} = -\nabla p + \mathbf{J}\times\mathbf{B}$$

$$p = \frac{\beta_0}{2}\rho^\gamma$$

$$\frac{\partial\mathbf{B}}{\partial t} = -\nabla\times\mathbf{E}$$

$$\mathbf{E} = -\mathbf{u}\times\mathbf{B}$$

$$\mathbf{J} = \nabla\times\mathbf{B}$$

$$\frac{\partial}{\partial t}\rho = -\frac{\partial}{\partial x}(\rho u_x)$$

$$\frac{\partial u_x}{\partial t} = -u_x\frac{\partial u_x}{\partial x} + \frac{1}{\rho}\left(-\frac{\partial p}{\partial x} - J_z B_y\right)$$

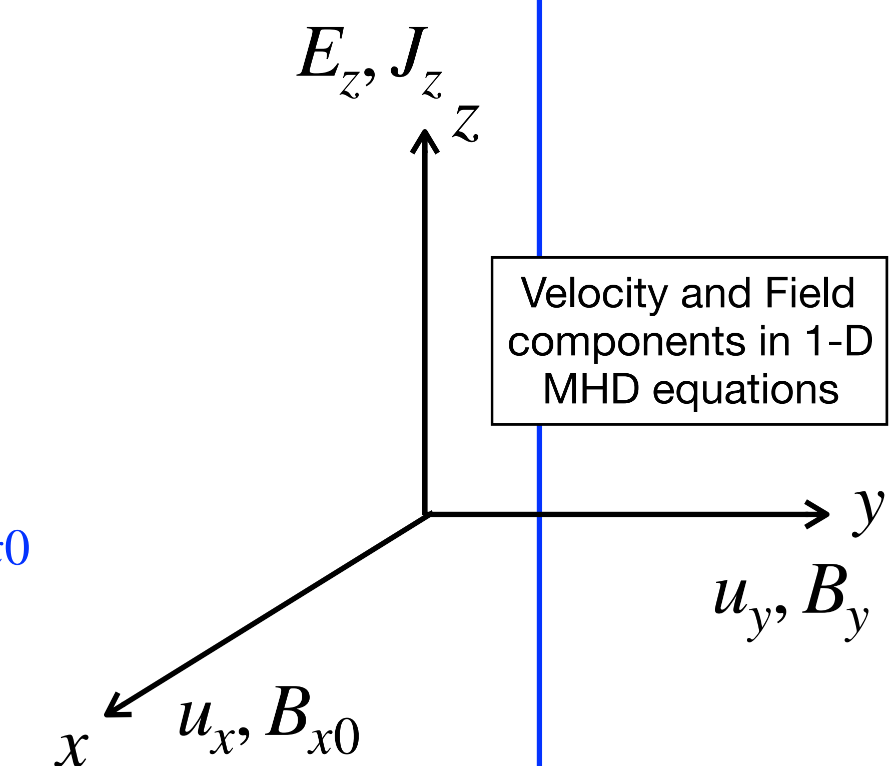
$$\frac{\partial u_y}{\partial t} = -u_x\frac{\partial u_y}{\partial x} + \frac{1}{\rho}J_z B_{x0}$$

$$p = \frac{\beta_0}{2}\rho^\gamma$$

$$\frac{\partial B_y}{\partial t} = \frac{\partial E_z}{\partial x}$$

$$E_z = -u_x B_y + u_y B_{x0}$$

$$J_z = \frac{\partial B_y}{\partial x}$$



Velocity and Field components in 1-D MHD equations

Spatial Derivatives - Finite Difference

1-D MHD equations

$$\frac{\partial}{\partial t}\rho = -\frac{\partial}{\partial x}(\rho u_x)$$

$$\frac{\partial u_x}{\partial t} = -u_x\frac{\partial u_x}{\partial x} + \frac{1}{\rho}\left(-\frac{\partial p}{\partial x} - J_z B_y\right)$$

$$\frac{\partial u_y}{\partial t} = -u_x\frac{\partial u_y}{\partial x} + \frac{1}{\rho}J_z B_{x0}$$

$$\frac{\partial B_y}{\partial t} = \frac{\partial E_z}{\partial x}$$

$$E_z = -u_x B_y + u_y B_{x0}$$

$$J_z = \frac{\partial B_y}{\partial x}$$

$$p = \frac{\beta_0}{2}\rho^\gamma$$

Finite-Difference Approximations

$$\frac{\partial \rho}{\partial t}\Big|_i = \frac{u^n_{x,i+1}\rho^n_{i+1} - u^n_{x,i-1}\rho^n_{i-1}}{2\Delta x}$$

$$\frac{\partial u_x}{\partial t}\Big|_i = -u_{x,i}\frac{u^n_{x,i+1} - u^n_{x,i-1}}{2\Delta x} + \frac{1}{\rho^n_i}\left(\frac{p^n_{i+1} - p_{i-1}}{2\Delta x} - J^n_{z,i}B^n_{y,i}\right)$$

$$\frac{\partial u_y}{\partial t}\Big|_i = -u_{x,i}\frac{u^n_{y,i+1} - u^n_{y,i-1}}{2\Delta x} + \frac{1}{\rho^n_i}J^n_{z,i}B_{x0}$$

$$\frac{\partial B_y}{\partial t}\Big|_i = -\frac{E^n_{z,i+1} - E^n_{z,i-1}}{2\Delta x}$$

$$E^n_{z,i} = -u^n_{x,i}B^n_{y,i} + u^n_{y,i}B_{x0}$$

$$J^n_{z,i} = \frac{B^n_{y,i+1} - B^n_{y,i-1}}{2\Delta x}$$

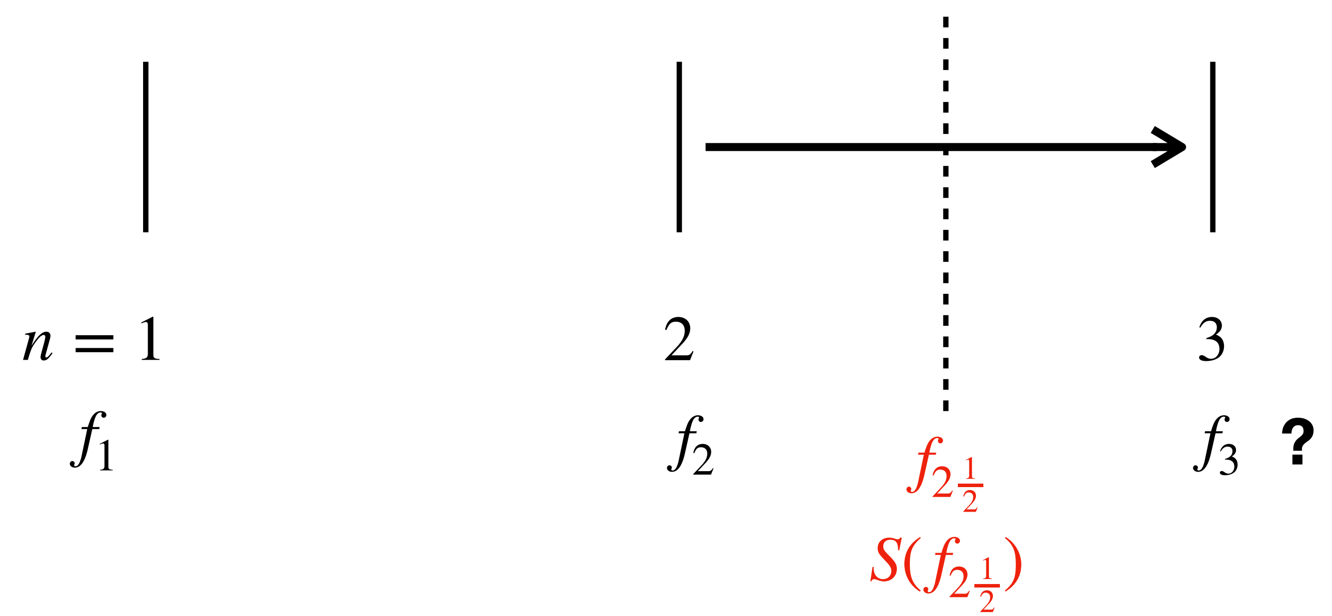
$$p^n_{z,i} = \frac{\beta_0}{2}(\rho^n_i)^\gamma$$

Time Evolution - Leapfrog Trapezoidal

There are many choices of getting a viable second order time stepping method, we will use a kind of “predictor - corrector” scheme named “*Leapfrog Trapezoidal*” (LT) method in the following discussions, which uses two sup-steps for the time evolution

Assume that f is known at t1 and t2, corresponding to n=1 and n=2, we want to calculate f at t3 using the differential equation

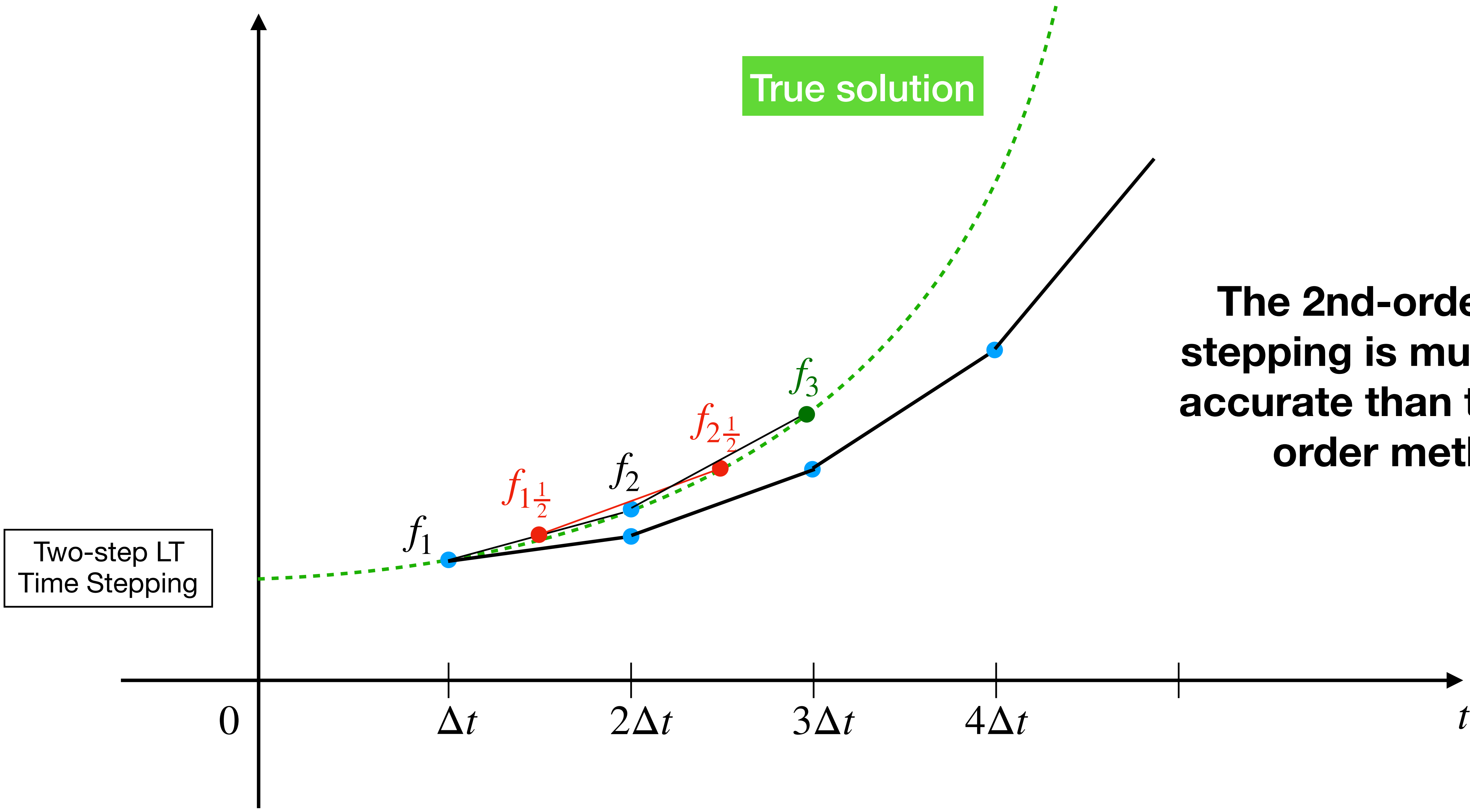
$$\frac{\partial f}{\partial t} = S(f)$$



- Going from f2 to f3 using f2 and S(f2) gives the first order time stepping
- If we know f2.5 and S(f2.5), we get something like a central difference scheme for f3:

$$\frac{f_i^{n+1} - f_i^n}{\Delta t} = S(f_i^{n+\frac{1}{2}}) \quad \Longrightarrow \quad f_i^{n+1} = f_i^n + \Delta t S(f_i^{n+\frac{1}{2}})$$

Interpretation of the LT method



The 2nd-order time stepping is much more accurate than the first-order method

$$f^{n+\frac{1}{2}} = \frac{1}{2}(f^n + f^{n-1}) + \Delta t \cdot S(f^n)$$

$$f^{n+1} = f^n + \Delta t \cdot S(f^{n+\frac{1}{2}})$$

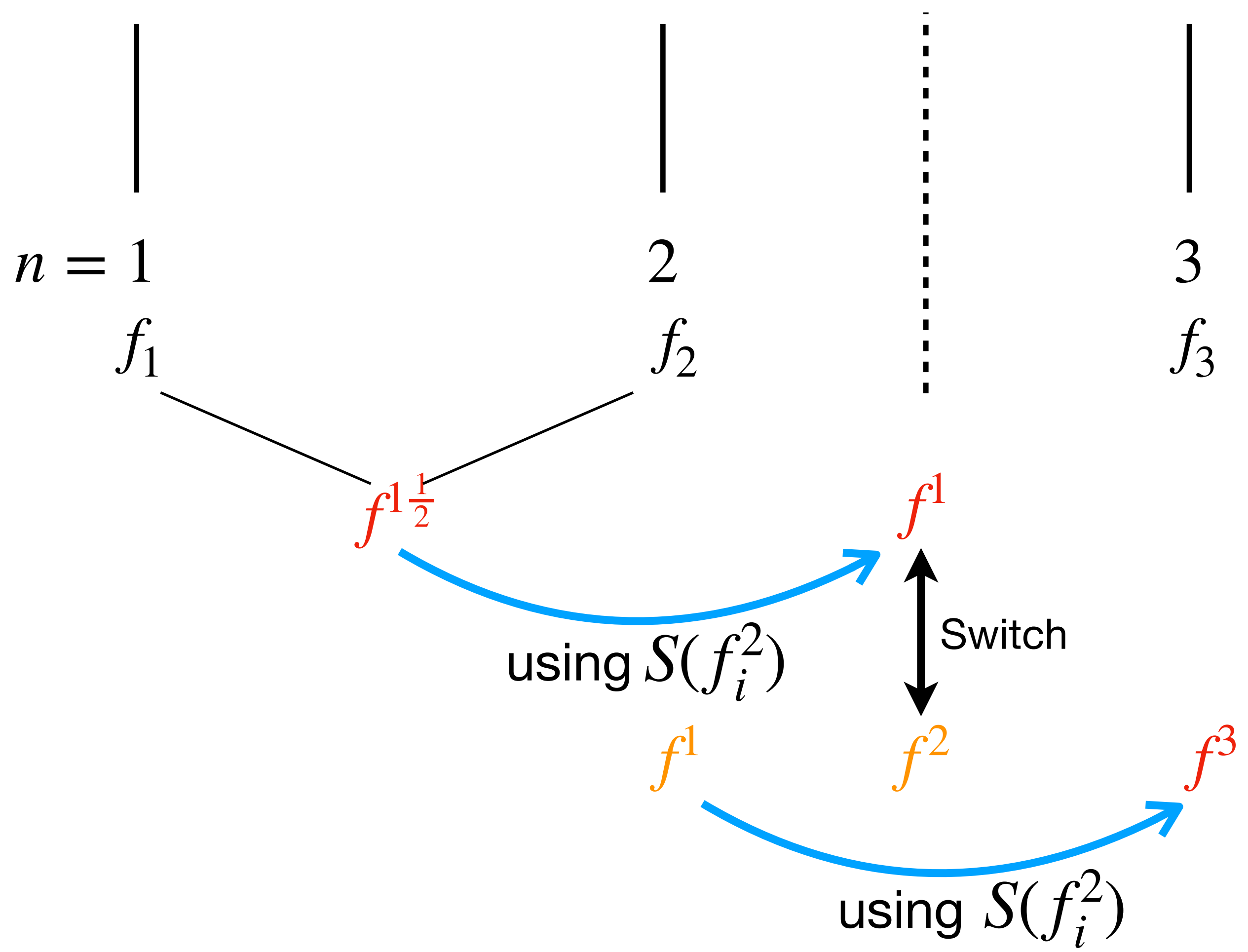
Leapfrog Trapezoidal time stepping

The LT algorithm for time stepping goes like

Two-step LT Time Stepping

$$\boxed{\begin{aligned} f^{n+\frac{1}{2}} &= \frac{1}{2}(f^n + f^{n-1}) + \Delta t \cdot S(f^n) \\ f^{n+1} &= f^n + \Delta t \cdot S(f^{n+\frac{1}{2}}) \end{aligned}} \qquad n = 2,3,4,\ldots$$

In general we need to store the variables (arrays) at each new time step (and each half time step like f2.5), it could be a waste of resource while the scale of the simulation is enormous. So we do something like this to get around:



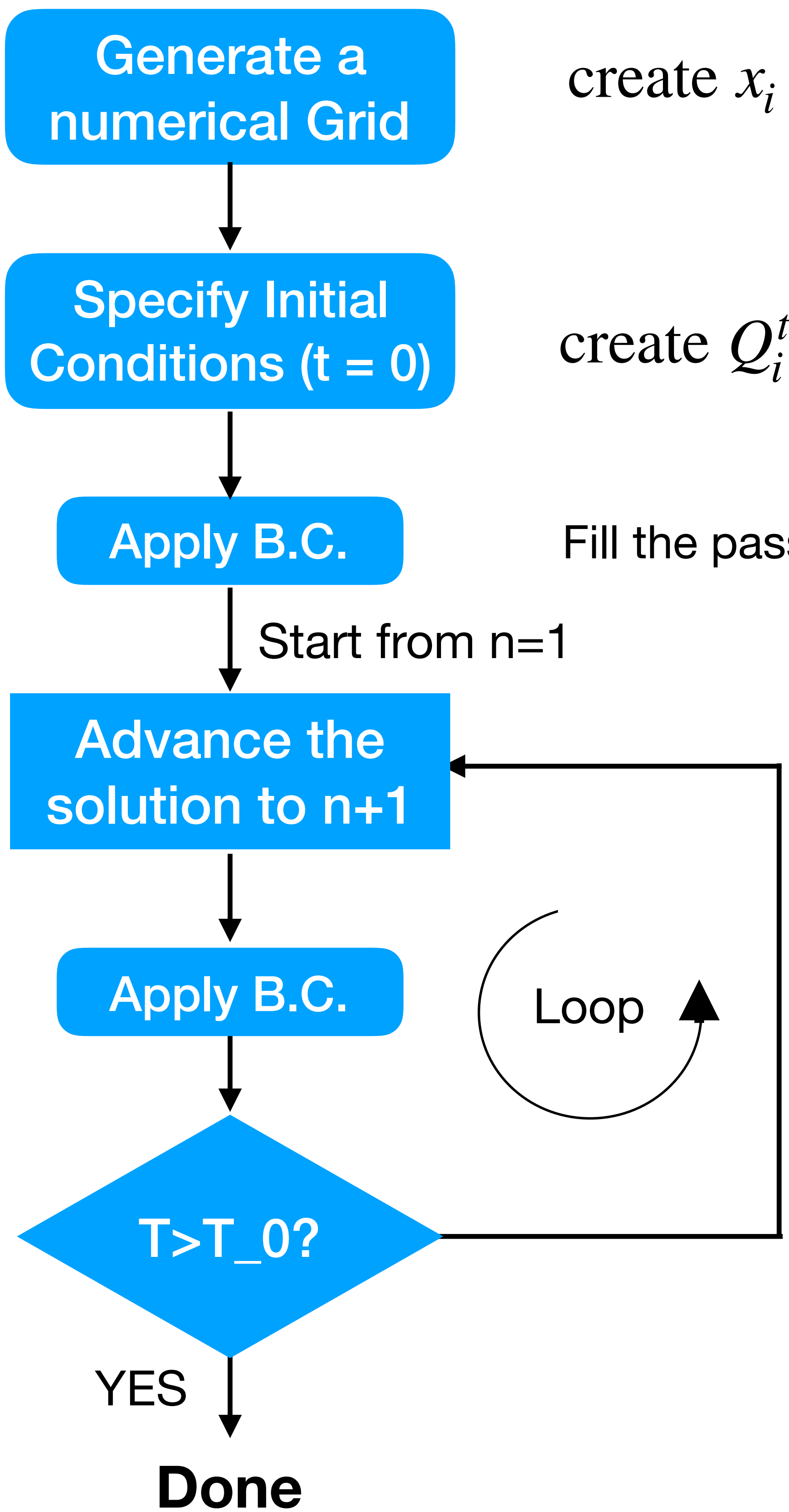
Algorithm used in code

STEP 1: $f^1 = \frac{1}{2}(f^1 + f^2) + \Delta t \cdot S(f^2)$

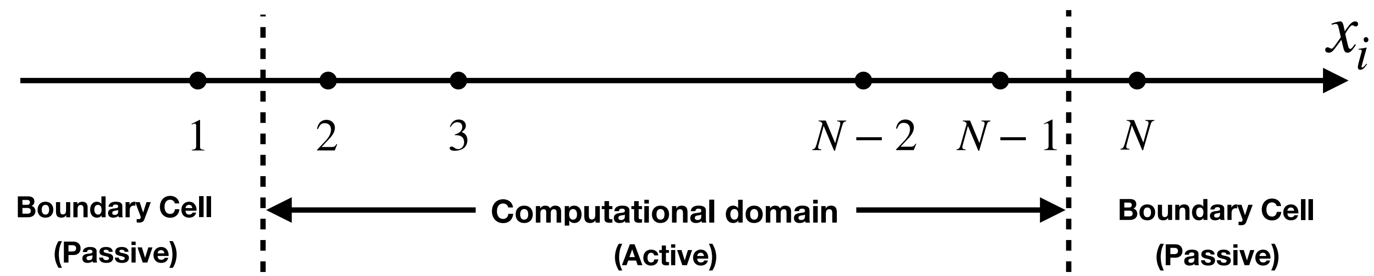
STEP 2: switch f1 and f2

STEP 3: $f^2 = f^1 + \Delta t \cdot S(f^2)$

Build the code w/ predictor-corrector steps



create x_i



create $Q_i^{t=0}$ $\quad$ e.g., $\rho_i^{t=0}$ = something

Fill the passive cells for derivatives at the end points

$$f^{n+\frac{1}{2}} = \frac{1}{2}(f^n + f^{n-1}) + \Delta t \cdot S(f^n)$$

$$f^{n+1} = f^n + \Delta t \cdot S(f^{n+\frac{1}{2}})$$

Two-step LT Time Stepping

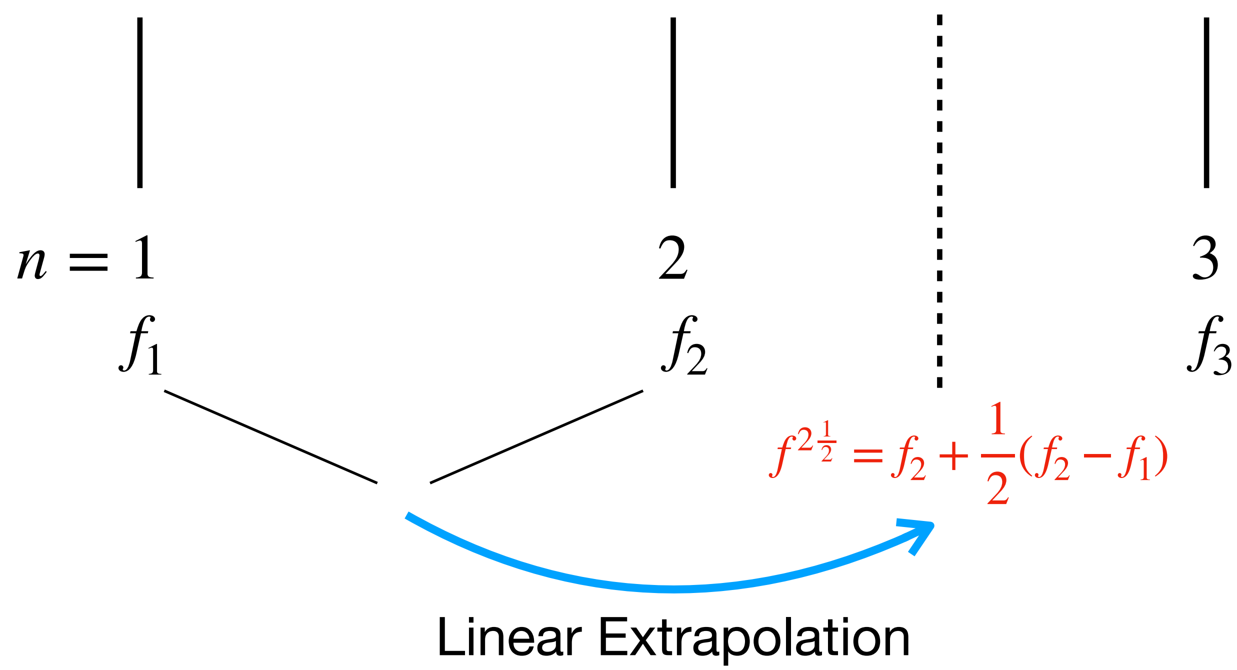
Fill the passive cells for derivatives at the end points

Determine whether reaching the end time (or other criteria)

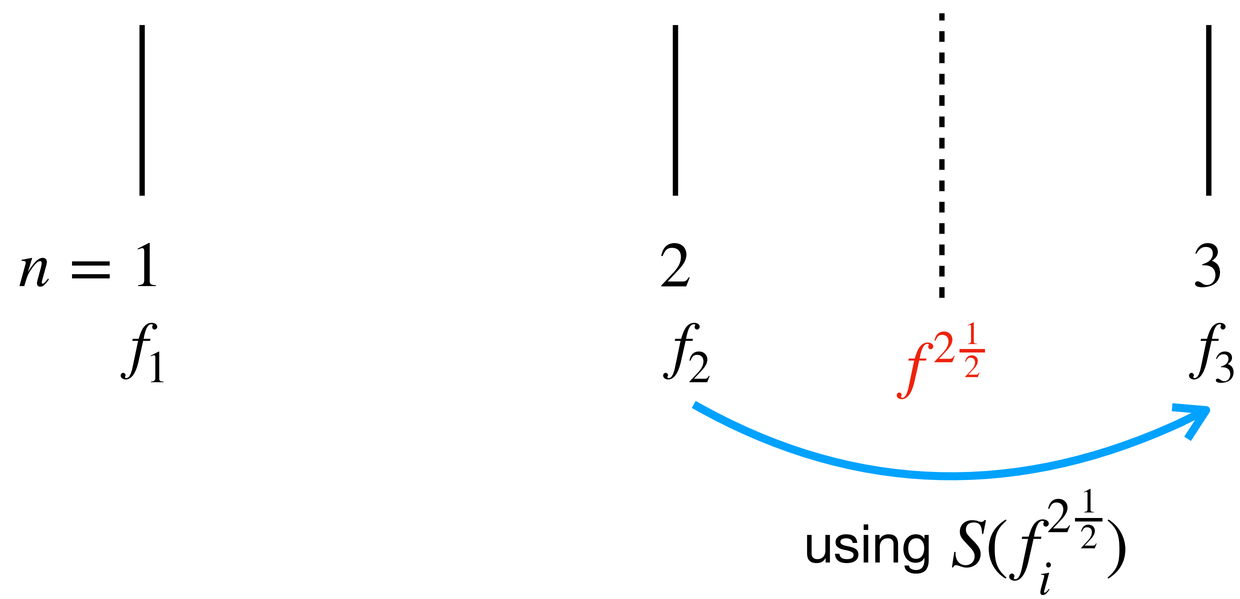
One-Step predictor-corrector method

Adams-Bashforth Method

The calculation of f2.5 is the ***predictor*** step - which is a linear extrapolation



Going from f2 to f3 is the same ***corrector*** step as leapfrog trapezoid method



Physics

Energy in an MHD code (magnetic)

The magnetic energy equation is basically the Poynting theorem applied to the plasma:

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E} \xrightarrow{\cdot\,\mathbf{B}} \mathbf{B}\cdot\frac{\partial \mathbf{B}}{\partial t} = -\mathbf{B}\cdot\nabla\times\mathbf{E} \xrightarrow{\nabla\cdot(\mathbf{E}\times\mathbf{B}) = \mathbf{B}\cdot\nabla\times\mathbf{E} - \mathbf{E}\cdot\nabla\times\mathbf{B}}$$

$$\longrightarrow \frac{\partial}{\partial t}\left(\frac{1}{2}B^2\right) = -\mathbf{E}\cdot\nabla\times\mathbf{B} - \nabla\cdot(\mathbf{E}\times\mathbf{B}) \xrightarrow[\mathbf{J} = \nabla\times\mathbf{B}]{\text{Ampere's law}}$$

$$= -\mathbf{E}\cdot\mathbf{J} - \nabla\cdot(\mathbf{E}\times\mathbf{B})$$

So the volume-integrated magnetic energy equation is written as

$\boxed{\text{Magnetic Energy}}$

$$\int_V \frac{\partial}{\partial t}\left(\frac{1}{2}B^2\right) dV = \int_V \left[-\nabla\cdot(\mathbf{E}\times\mathbf{B}) - \mathbf{E}\cdot\mathbf{J}\right] dV$$

Converts magnetic energy to/from kinetic energy

And

$\boxed{\text{Kinetic Energy}}$

$$\int_V \frac{d}{dt}\left(\frac{1}{2}\rho u^2\right) dV = \int_V \left[-\nabla\cdot\left(\frac{1}{2}\rho u^2\mathbf{u}\right) - \mathbf{u}\cdot\nabla p + \mathbf{J}\cdot\mathbf{E}\right] dV$$

Converts internal energy to/from kinetic energy

$\boxed{\text{Internal Energy}}$

$$\int_V \frac{\partial}{\partial t}\left(\frac{p}{\gamma - 1}\right) dV = \int_V \left[-\nabla\cdot\left(\frac{\gamma p}{\gamma - 1}\mathbf{u}\right) + \mathbf{u}\cdot\nabla p\right] dV$$

Physics

Energy in an MHD code (total)

Adding the three energy equations together

$$\int_V \frac{d}{dt}\left(\frac{1}{2}\rho u^2 + \frac{p}{\gamma-1} + \frac{1}{2}B^2\right) dV = \int_V \left[-\nabla\cdot\left(\frac{1}{2}\rho u^2\mathbf{u}\right) - \nabla\cdot\left(\frac{\gamma p}{\gamma-1}\mathbf{u}\right) - \nabla\cdot(\mathbf{E}\times\mathbf{B})\right] dV$$

Total Energy

$$= -\int_V \nabla\cdot\left[\left(\frac{1}{2}\rho u^2\mathbf{u}\right) + \left(\frac{\gamma p}{\gamma-1}\mathbf{u}\right) + (\mathbf{E}\times\mathbf{B})\right] dV$$

$$\xrightarrow[\int_V \nabla\cdot A\, dV = \oint_S \mathbf{A}\cdot d\mathbf{S}]{\text{Divergence Theorem}} \quad = -\oint_S d\mathbf{S}\cdot\left[\left(\frac{1}{2}\rho u^2\mathbf{u}\right) + \left(\frac{\gamma p}{\gamma-1}\mathbf{u}\right) + (\mathbf{E}\times\mathbf{B})\right]$$

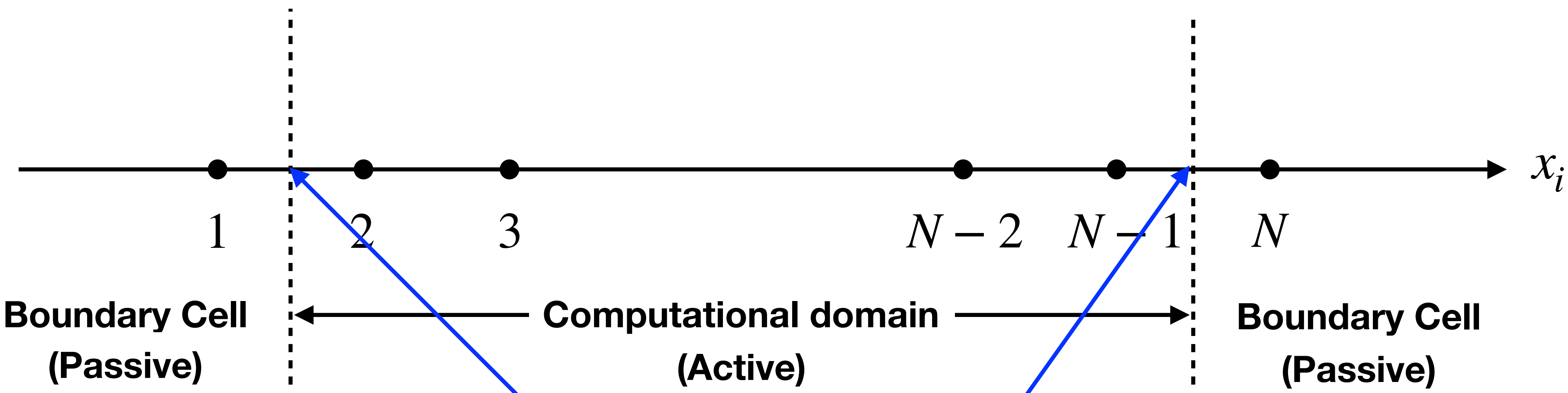
Given certain boundary conditions the surface integral $\oint_S$ may go to zero, leading to total **energy conservation**

$$\mathscr{E}_T = \frac{1}{2}\rho u^2 + \frac{p}{\gamma-1} + \frac{1}{2}B^2 \quad \boxed{\text{Total Energy}}$$

And the following term is called the total energy flux

$$\mathbf{F}_{\mathscr{E}} = \frac{1}{2}\rho u^2\mathbf{u} + \frac{\gamma p}{\gamma-1}\mathbf{u} + \mathbf{E}\times\mathbf{B} \quad \boxed{\text{Total Energy Flux}}$$

Energy Conservation in 1-D MHD code



$$\frac{d}{dt}\int_V \mathscr{E}_T dV = -\oint_S d\mathbf{S}\cdot\boxed{\left(\frac{1}{2}\rho u^2\mathbf{u} + \frac{\gamma p}{\gamma-1}\mathbf{u} + \mathbf{E}\times\mathbf{B}\right)}$$

Total Energy Flux

If the total energy flux is zero on the boundary grid cells, then the total energy is conserved

$$\frac{d}{dt}\int_V \mathscr{E}_T dV = 0$$

This is called energy-conserving boundary conditions for MHD

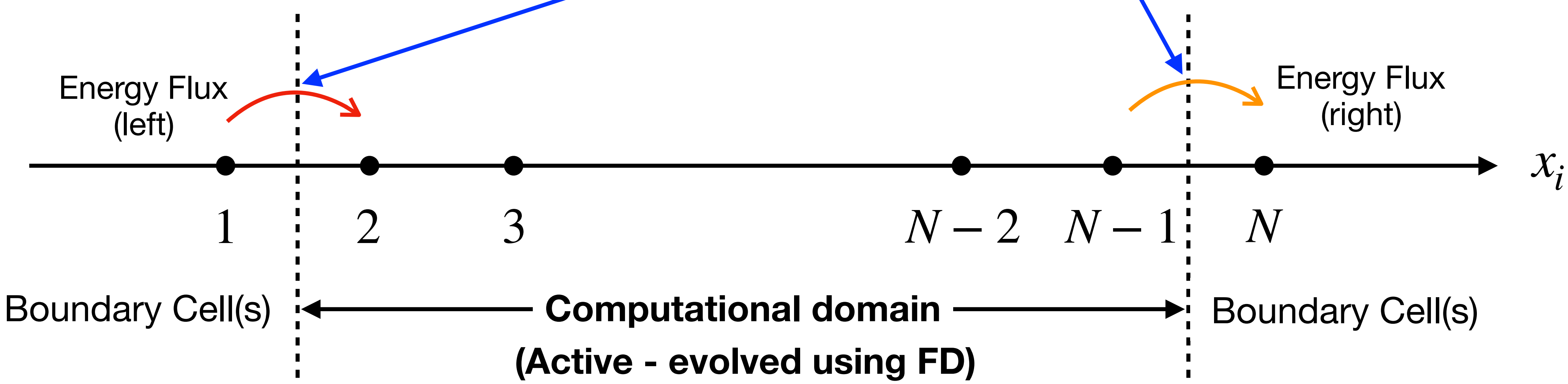
Boundary Conditions in 1-D MHD code

Another type of boundary conditions that conserves total energy is called "***Hard Wall***"

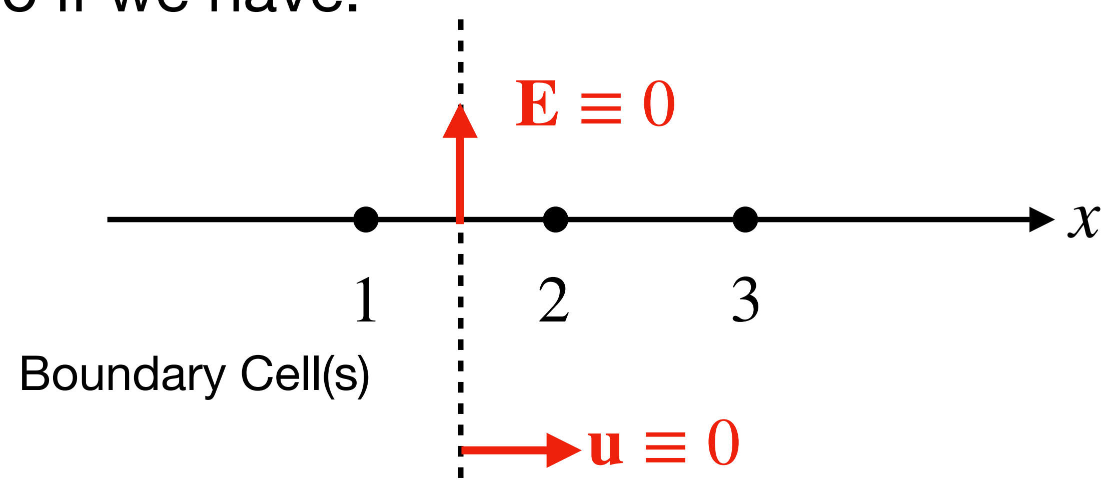
Recall the energy flux on the boundary is calculated as

$$\frac{d}{dt}\int_V \mathscr{E}_T dV = -\oint_S d\mathbf{S}\cdot \boxed{\left[\cancelto{0}{\left(\frac{1}{2}\rho u^2 + \frac{\gamma p}{\gamma - 1}\right)\mathbf{u}} + \mathbf{E}\times\mathbf{B}\right]}$$

Total Energy Flux Through boundaries



So if we have:



$$\left[\left(\frac{1}{2}\rho u^2 + \frac{\gamma p}{\gamma - 1}\right)\mathbf{u} + \mathbf{E}\times\mathbf{B}\right]_{1\frac{1}{2}} \equiv 0$$

Total energy flux is zero on boundary

What does “Hard-wall" boundary mean?

The “hard-wall” boundary puts a constraint on the normal component of the velocity, I.e., there’s no flow velocity across the boundary interface (wall). Thus the mass flux through the interface is also expected to be zero - the total mass within the computation domain is conserved (to the scheme accuracy)

Let’s take a look at the mass equation

$$\frac{\partial}{\partial t}\rho = -\nabla \cdot \rho \mathbf{u} \xrightarrow{\int_V dV} \int_V \left(\frac{\partial}{\partial t}\rho = -\nabla \cdot \rho \mathbf{u} \right) dV$$

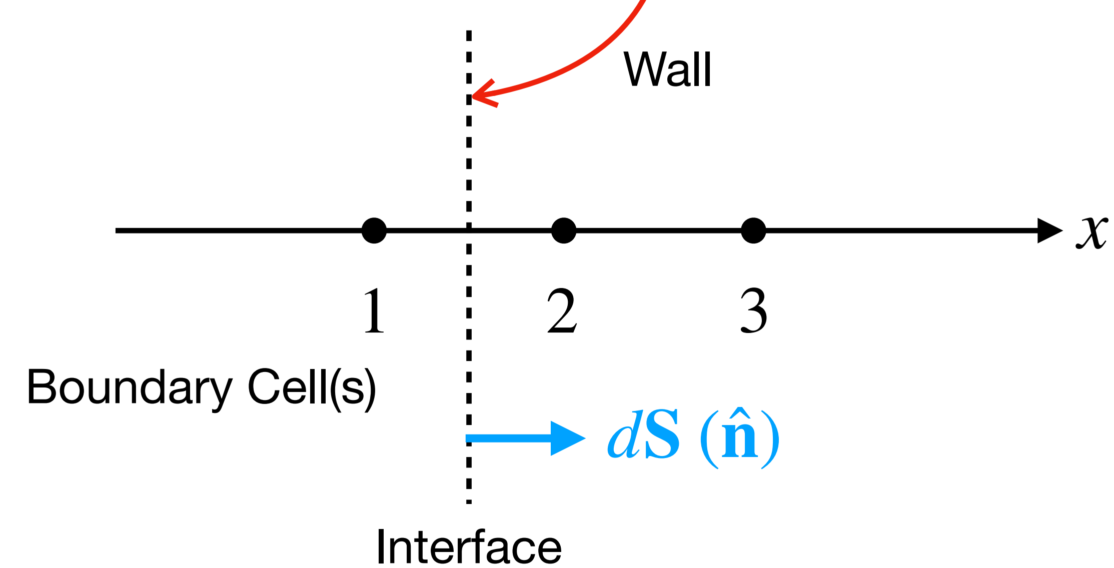
$$\longrightarrow \frac{\partial}{\partial t}\int_V \rho dV = -\int_V \nabla \cdot \rho \mathbf{u} dV \longrightarrow \frac{\partial}{\partial t}\int_V \rho dV = -\oint_{\boxed{S}} \rho \mathbf{u} \cdot d\mathbf{S}$$

So if $\mathbf{u} \cdot d\mathbf{S} \equiv 0$

We have

$$\oint_S \rho \mathbf{u} \cdot d\mathbf{S} \equiv 0 \longrightarrow \frac{\partial}{\partial t}\int_V \rho dV = 0$$

total mass unchanged



How to implement hard-wall boundary?

Hard-wall condition requires the normal component of the velocity across the wall is zero

Hard wall: $\underset{\text{normal}}{u_n} = \mathbf{u} \cdot \hat{\mathbf{n}} \equiv 0$

In the code, velocities are specified at grid points 1,2,3,… N. How's interface velocity calculated?

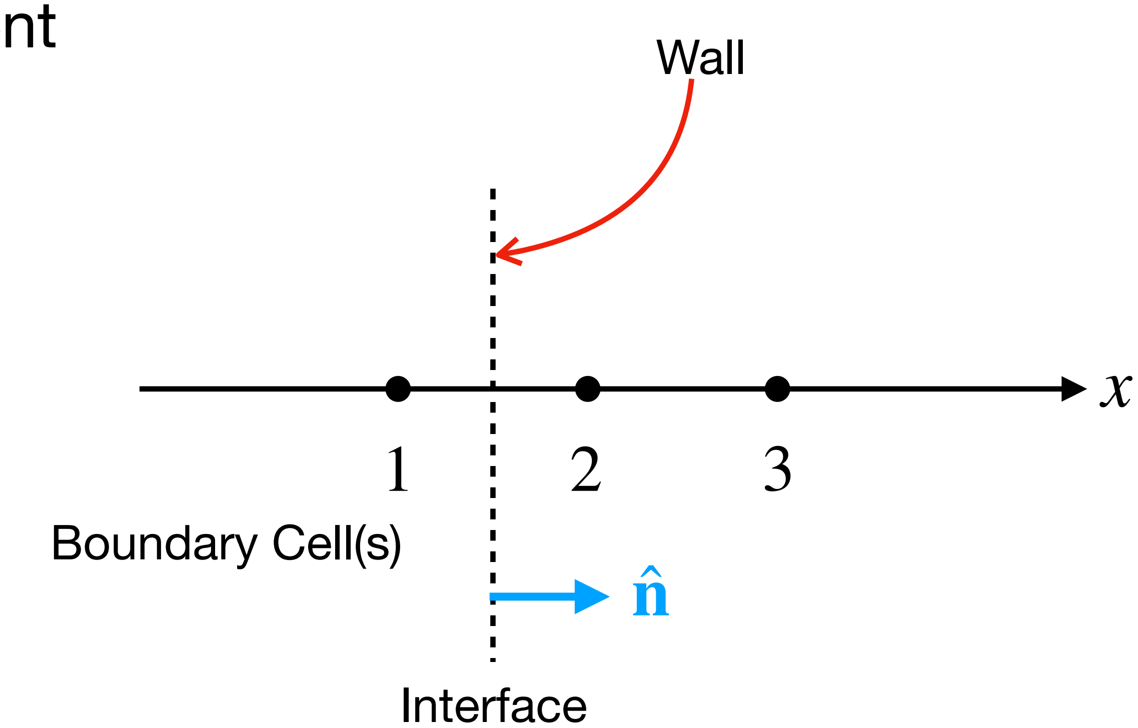
It's done based on the boundary cell(s):

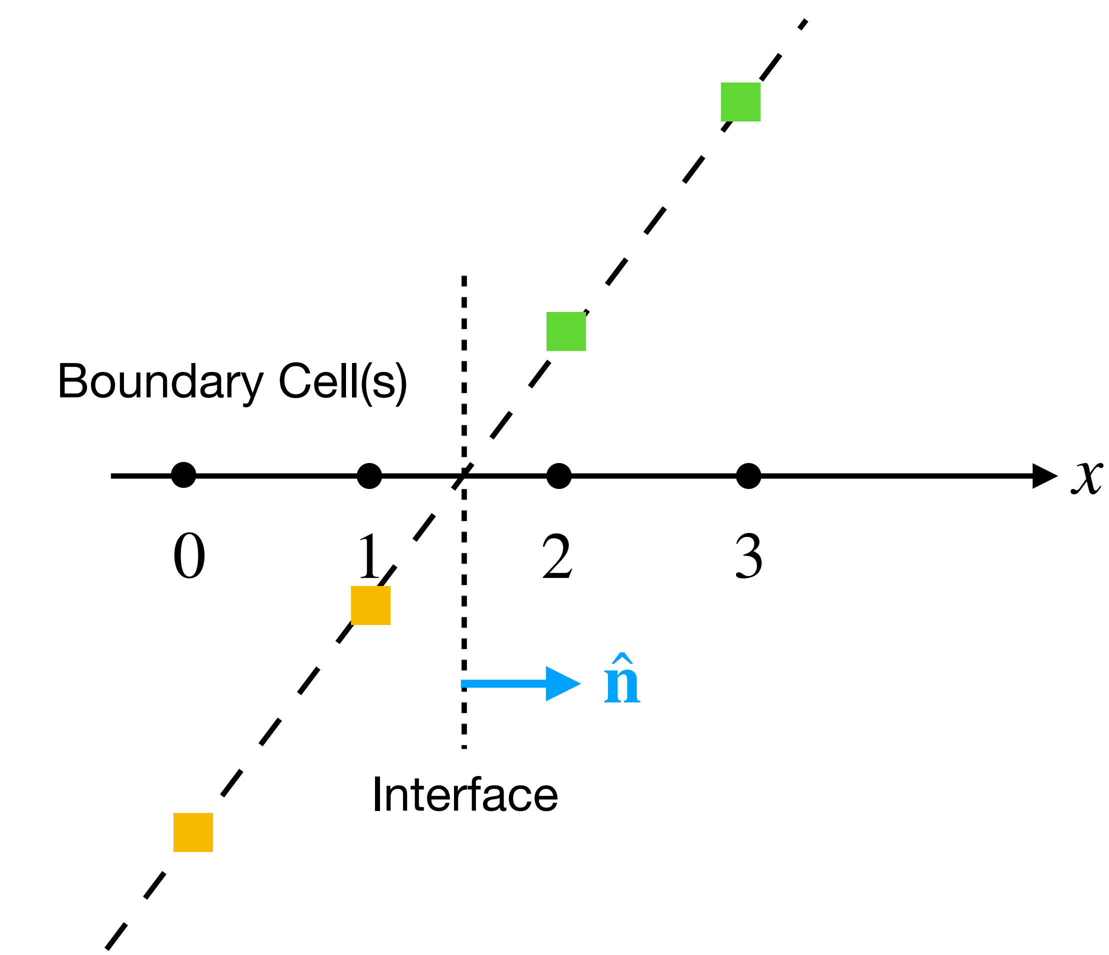
If we set the boundary cells with values that are anti-symmetric with the active computational cells right next to the boundary, the "interpolated" value on the interface is going to be exactly zero (why?)

This is also called **"anti-symmetric" boundary conditions. In the code, simply do:**

$$u_x(1) = -u_x(2)$$

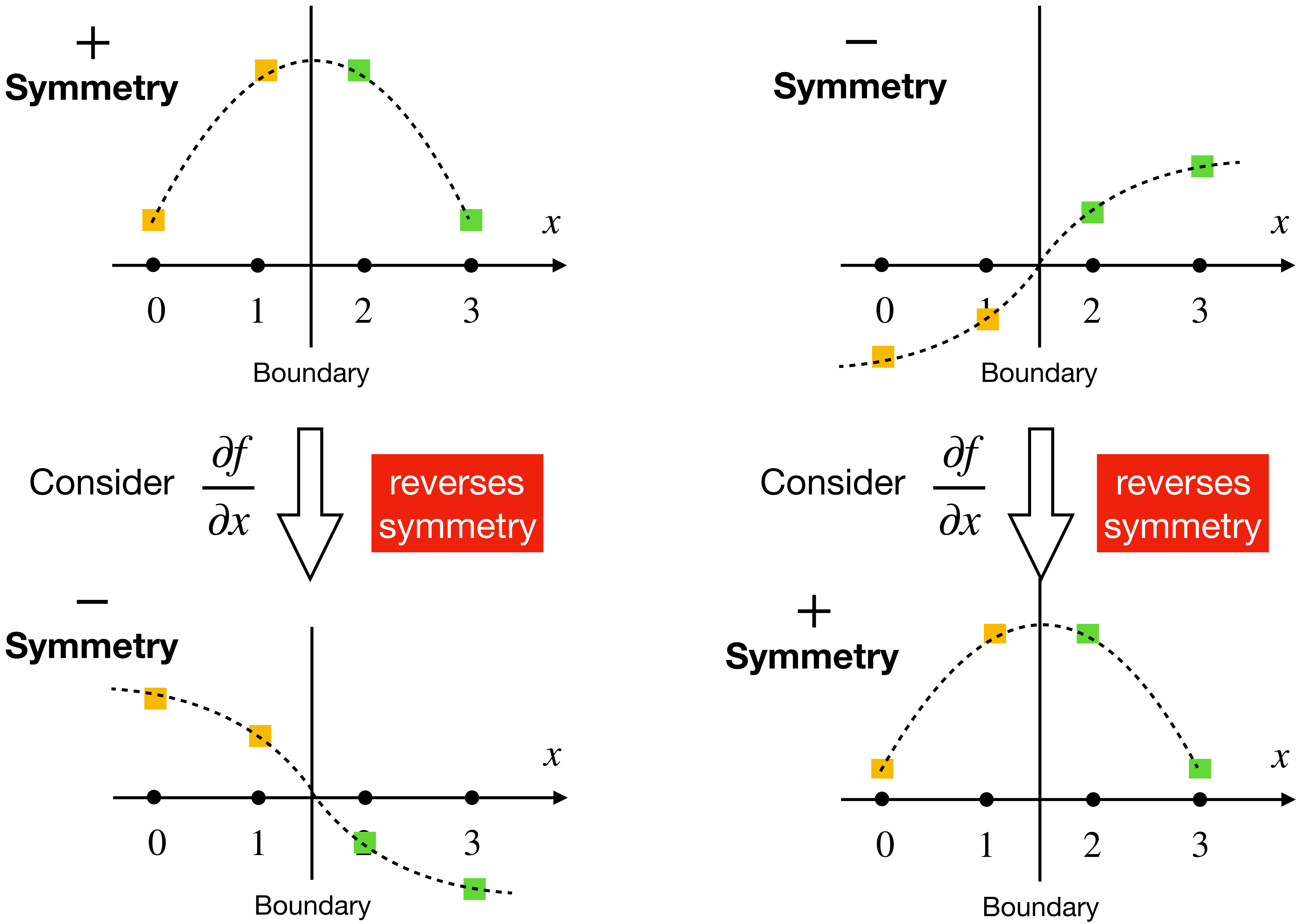
$$u_x(N) = -u_x(N-1)$$





Properties of symmetric boundaries

Two types of symmetries:



Properties of symmetric boundaries

Now let's get the **full set** of boundary conditions for 1-D MHD equations, which conserves mass and energy:

Normal velocity: **Hard wall** $\longrightarrow$ $u_x \ (-)$

Tangential electric field: **Perfect conductor** $\longrightarrow$ $E_z \ (-)$

Plasma density pressure: **Equilibrium** $\longrightarrow$ $\rho, p \ (+)$

What about other vector components - B_y, u_y

$$B_y : \quad \frac{\partial B_y}{\partial t} = \frac{\partial E_z}{\partial x} \xrightarrow[\partial_x\,(-)]{E_z\,(-)} B_y \sim (-)(-) \longrightarrow B_y \ (+)$$

This is because $\dfrac{\partial}{\partial t}$ does not change symmetry

$$J_z : \quad J_z = \frac{\partial B_y}{\partial x} \xrightarrow[\partial_x\,(-)]{B_y\,(+)} J_z \sim (+)(-) \longrightarrow J_z \ (-)$$

Check the density equation:

$$\rho : \quad \frac{\partial}{\partial t}\rho = -\frac{\partial}{\partial x}(\rho u_x) \xrightarrow[\partial_x\,(-)]{\rho\,(+)\,u_x\,(-)} \rho \sim (+)(-)(-) \sim (+)$$

Symmetry consistant!

Properties of symmetric boundaries

Also let's check the equation of u_x:

$$u_x : \quad \frac{\partial u_x}{\partial t} = -u_x \frac{\partial u_x}{\partial x} + \frac{1}{\rho}\left(-\frac{\partial p}{\partial x} - J_z B_y \right)$$

$$(-)$$

$$\frac{(-)(-)(-)}{(-)} \qquad (+) \qquad \frac{\big(\,(-)(+) \quad (-)(+)\,\big)}{(-)}$$

$$(-)$$

Symmetry consistant!

$$u_y : \quad \frac{\partial u_y}{\partial t} = -u_x \frac{\partial u_y}{\partial x} + \frac{1}{\rho} J_z B_{x0}$$

$$(-)(-) \qquad \frac{(+)(-)}{(-)}$$

$$\Rightarrow \quad u_y \ (-)$$

It is straightforward to show that u_y must be anti-symmetric to make the symmetry in the equation of u_y consistent

Also can see from Ohm's law;

$$E_z = -u_x B_y + u_y B_{x0}$$

Summary of Energy-Conserving boundaries

Plasma variables: $\rho, p \; (+) \quad u_x, u_y \; (-)$

Magnetic fields: $B_y \; (+)$

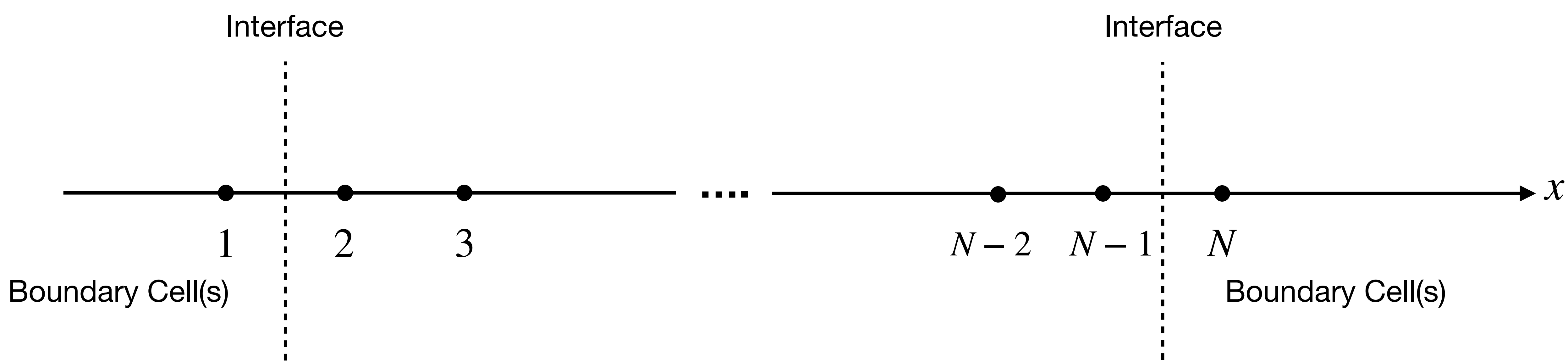
Primary Variables

Electric fields: $E_z \; (-)$

Currents: $J_z \; (-)$

Derived Variables

How to implement in the code?



Left bound

$$u_x(1) = -u_x(2)$$
$$\rho(1) = \rho(2)$$

Right bound

$$u_x(N) = -u_x(N-1)$$
$$\rho(N) = \rho(N-1)$$

Put Everything Together : 1-D MHD solver

Finite-Difference Approximations

$$\frac{\partial \rho}{\partial t}\Big|_i = \frac{u^n_{x,i+1}\rho^n_{i+1} - u^n_{x,i-1}\rho^n_{i-1}}{2\Delta x}$$

$$\frac{\partial u_x}{\partial t}\Big|_i = -u_{x,i}\frac{u^n_{x,i+1} - u^n_{x,i-1}}{2\Delta x} + \frac{1}{\rho^n_i}\left(\frac{p^n_{i+1} - p_{i-1}}{2\Delta x} - J^n_{z,i}B^n_{y,i}\right)$$

$$\frac{\partial u_y}{\partial t}\Big|_i = -u_{x,i}\frac{u^n_{y,i+1} - u^n_{y,i-1}}{2\Delta x} + \frac{1}{\rho^n_i}J^n_{z,i}B_{x0}$$

$$\frac{\partial B_y}{\partial t}\Big|_i = -\frac{E^n_{z,i+1} - E^n_{z,i-1}}{2\Delta x}$$

$$E^n_{z,i} = -u^n_{x,i}B^n_{y,i} + u^n_{y,i}B_{x0}$$

$$J^n_{z,i} = \frac{B^n_{y,i+1} - B^n_{y,i-1}}{2\Delta x}$$

$$p^n_{z,i} = \frac{\beta_0}{2}(\rho^n_i)^\gamma$$

Two-step LT Time Stepping

$$\frac{\partial f}{\partial t} = S(f)$$

$$f^{n+\frac{1}{2}} = \frac{1}{2}(f^n + f^{n-1}) + \Delta t \cdot S(f^n)$$

$$f^{n+1} = f^n + \Delta t \cdot S(f^{n+\frac{1}{2}})$$

$$n = 2,3,4,...$$

Boundary conditions

Plasma variables: **Magnetic fields:**

$\rho, p \; (+)$ $\quad u_x, u_y \; (-)$

$B_y \; (+)$

Electric fields: **Currents:**

$E_z \; (-)$

$J_z \; (-)$

Once though the mhd.m