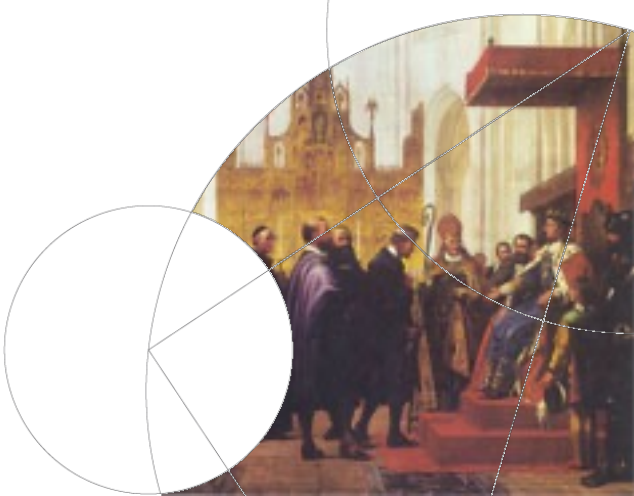
Faculty of Science

Computational Astrophysics

2b. Hydrodynamics
Riemann Solvers, shocks, and AMR

Troels Haugbølle
Niels Bohr Institute
University of Copenhagen



Topics today

- Overview of hydrodynamics
- Jump conditions and shock properties
- Shocks and Riemann solver
- Consistency and Approximate solvers
- **Assignment 2b:**
  - Blast wave
- Adaptive Mesh Refinement

Hydrodynamics

- Hydrodynamics describes the evolution of a fluid.
- Many formulations exist depending on the subject, but in astrophysics we normally consider the compressible Euler equations
- These are governed by three conservations principles for mass, momentum, and energy

$$\partial_t \rho + \nabla \cdot [\rho \, \mathbf{v}] = 0$$
$$\partial_t \rho \, \mathbf{v} + \nabla \cdot [\rho \, \mathbf{v} \otimes \mathbf{v} + P \mathbf{I}] = 0$$
$$\partial_t E + \nabla \cdot [(E + P) \, \mathbf{v}] = 0$$

- E is the total energy: $E = \rho e_{int} + \frac{1}{2}\rho v^2$, with equation of state $P(\rho, T)$
- The difficulty with solving these equations comes predominantly from the Navier-Stokes equation that is non-linear in the velocity. This is what generates the rich structure seen in e.g. turbulent flows and enables shocks.

Why do shocks occur in nature? What is a shock?

Shocks are in some sense very non-linear waves. They occur due to two reasons:

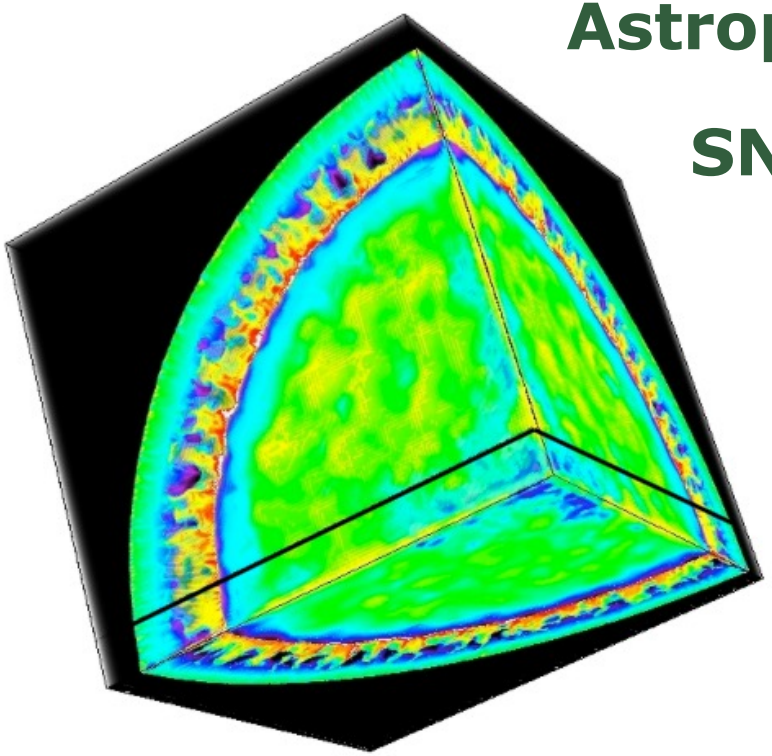
- Wave steepening related to the density dependence of the speed of sound: higher density → higher temperature → higher sound speed. Upper crest gradually overtakes (weak shock)

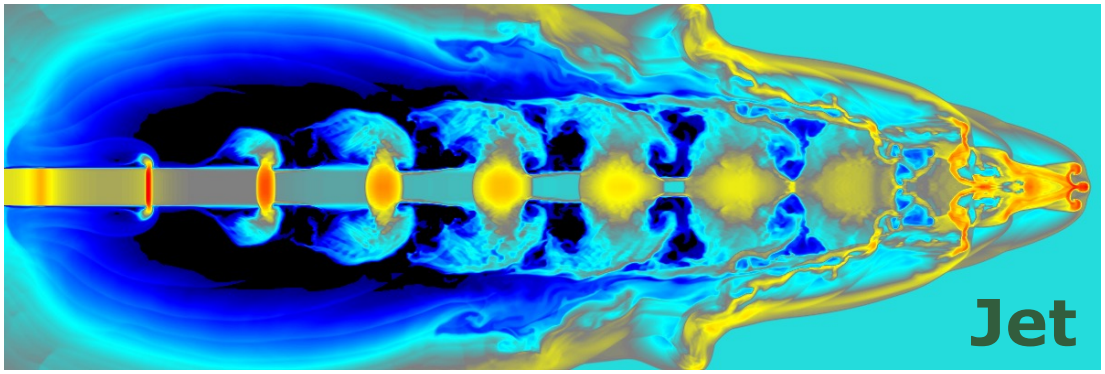
- Failure of the fluid to communicate about a discontinuity – supersonic shocks (strong shocks)

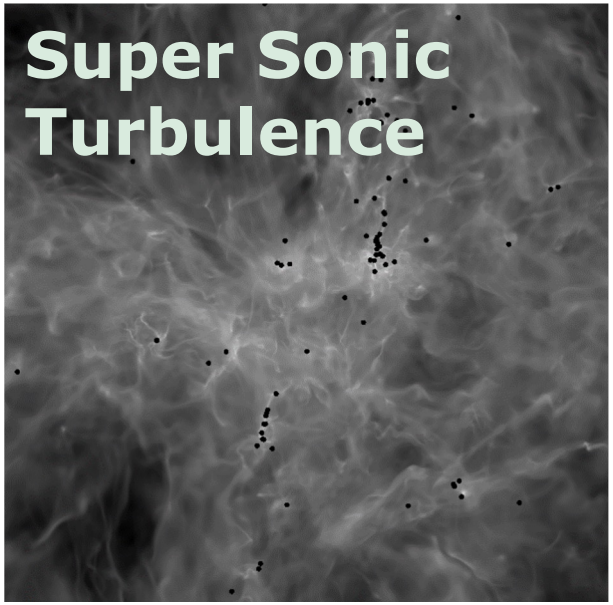
- End result: ***kinetic energy is converted to heat***

Astrophysics is full of shocks

SN remnant







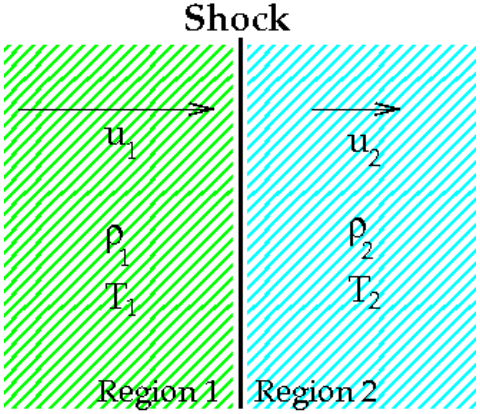
How to calculate shock properties

- Start with the conservation of mass, momentum, and energy:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}) = 0$$

$$\frac{\partial \rho \vec{u}}{\partial t} + \nabla \cdot (\rho \vec{u}\vec{u} + P\vec{I}) = 0$$

$$\frac{\partial E_{tot}}{\partial t} + \nabla \cdot ((E_{tot} + P)\vec{u}) = 0, \quad E_{tot} = \rho e + \frac{1}{2}\rho \vec{u}^2$$



- Assume we are in the rest-frame of the shock, then the flux is zero, and we get the ***shock-jump or Rankine-Hugoniot conditions***

$$\rho_1 u_1 = \rho_2 u_2$$

$$\rho_1 u_1^2 + P_1 = \rho_2 u_2^2 + P_2$$

$$u_1\left(\rho_1 e_1 + \frac{1}{2}\rho_1 u_1^2 + P_1\right) = u_2(\rho_2 e_2 + \frac{1}{2}\rho_2 u_2^2 + P_2)$$

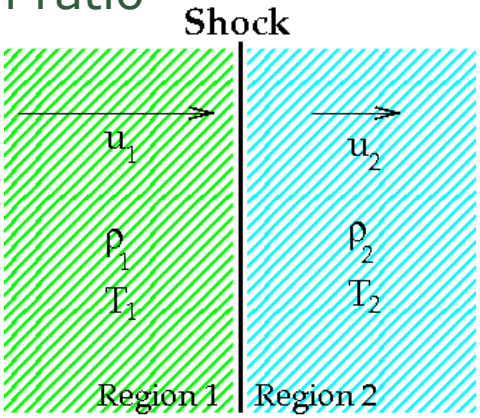
How to calculate (adiabatic) shock properties

- This can be used to calculate e.g. the compression ratio

$$\frac{\rho_2}{\rho_1} = \frac{P_2 + m^2 P_1}{P_1 + m^2 P_2}, \qquad m = \frac{\gamma - 1}{\gamma + 1}$$

$$\gamma = \frac{5}{3} : \text{max compression is } 4$$

$$\gamma = 1 : \text{max compression is infinite}$$



- In a moving shock, the shock speed can also be calculated

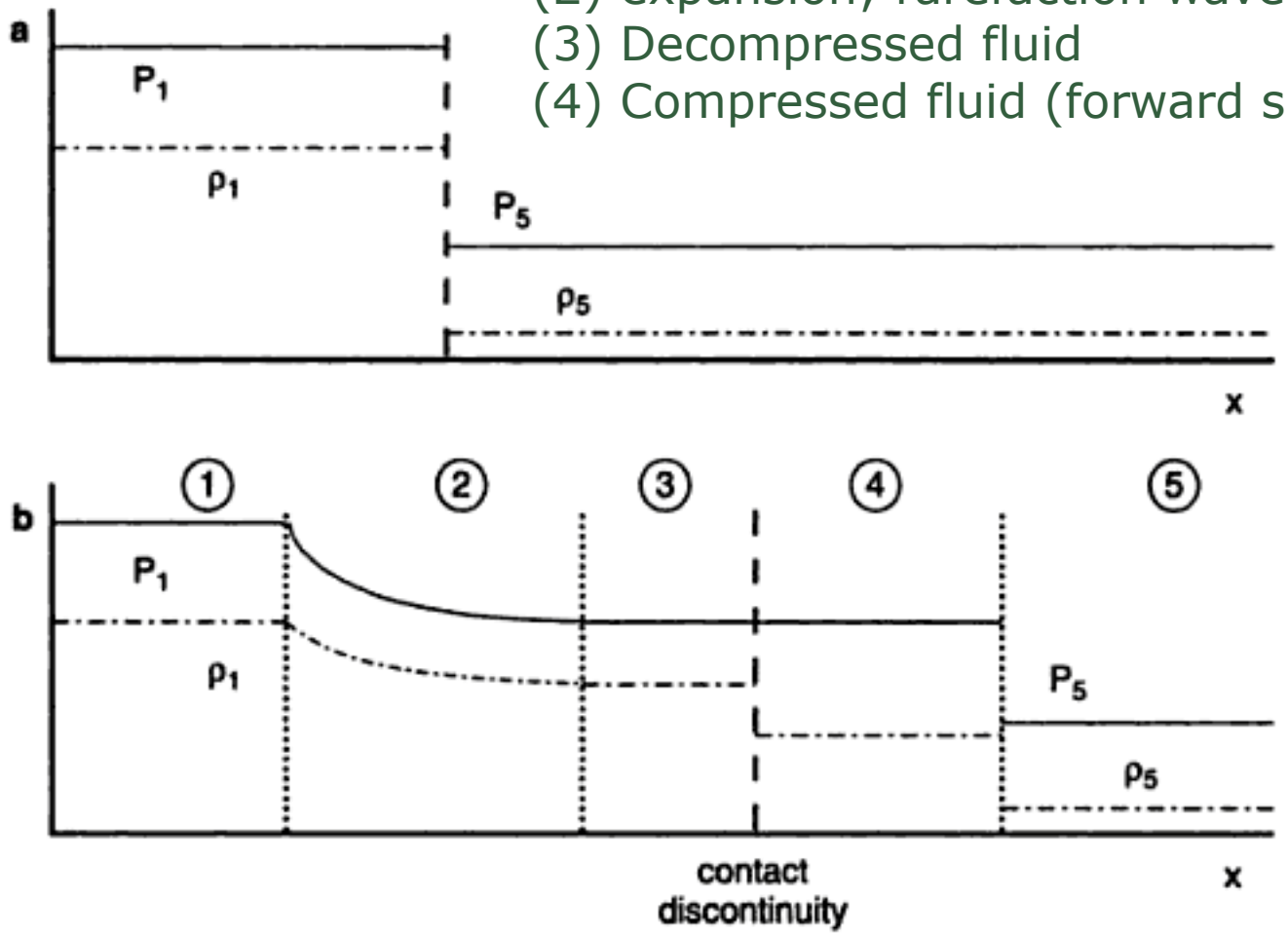
- Shocks are often characterized by the Mach number, the ratio between the shock speed and the upstream (or incoming) sound speed

$$M = \frac{u_1}{c_{s,1}}, \qquad c_s^2 = \gamma \frac{P}{\rho}$$

$$\frac{\rho_2}{\rho_1} = \frac{(\gamma + 1) M_1^2}{2 + (\gamma - 1) M_1^2}$$

HD shock structure in general (between two cells)

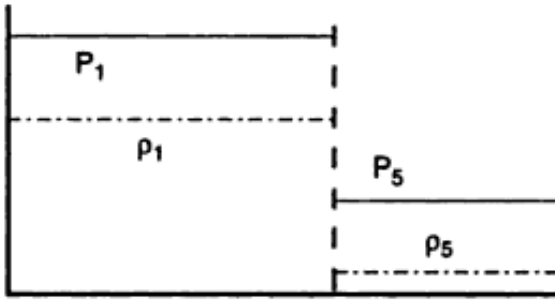
(1)+(5) undisturbed fluid
(2) expansion; rarefaction wave (reverse shock)
(3) Decompressed fluid
(4) Compressed fluid (forward shock)a



(see also fig. 6.5 in the book)

Shocks as a probe of numerical methods

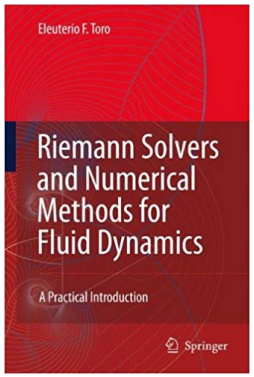
Test	ρ_L	u_L	p_L	ρ_R	u_R	p_R
1	1.0	0.75	1.0	0.125	0.0	0.1
2	1.0	-2.0	0.4	1.0	2.0	0.4
3	1.0	0.0	1000.0	1.0	0.0	0.01
4	5.99924	19.5975	460.894	5.99242	-6.19633	46.0950
5	1.0	-19.59745	1000.0	1.0	-19.59745	0.01
6	1.4	0.0	1.0	1.0	0.0	1.0
7	1.4	0.1	1.0	1.0	0.1	1.0



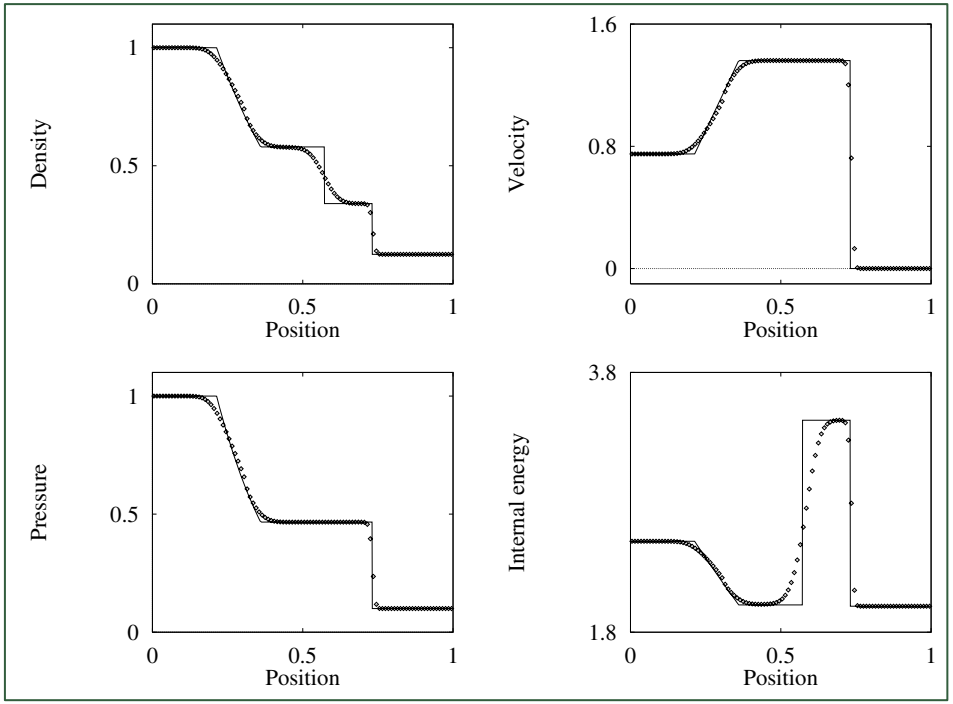
- *Shock Tubes* has long been a way to test numerical methods
- It is possible with iterative methods to find the exact solution to the shock problem
- Alternative is to run a code at high resolution
- Good selection of problem's given in Toro's book

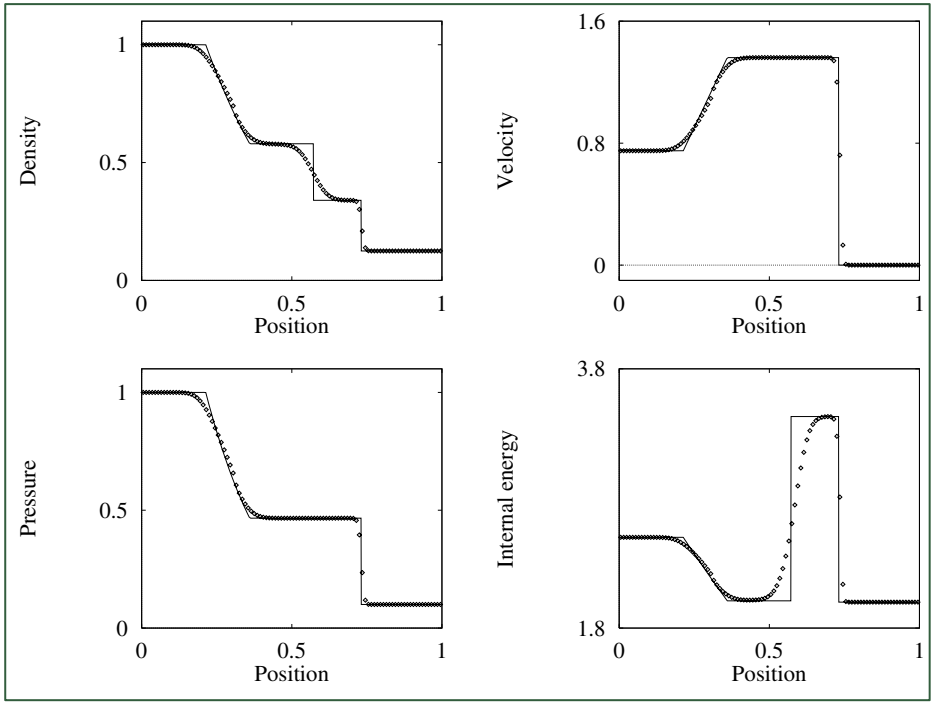
https://link.springer.com/book/10.1007/b79761

Free PDF(!) when downloaded on KU premise.



Shocks as a probe of numerical methods



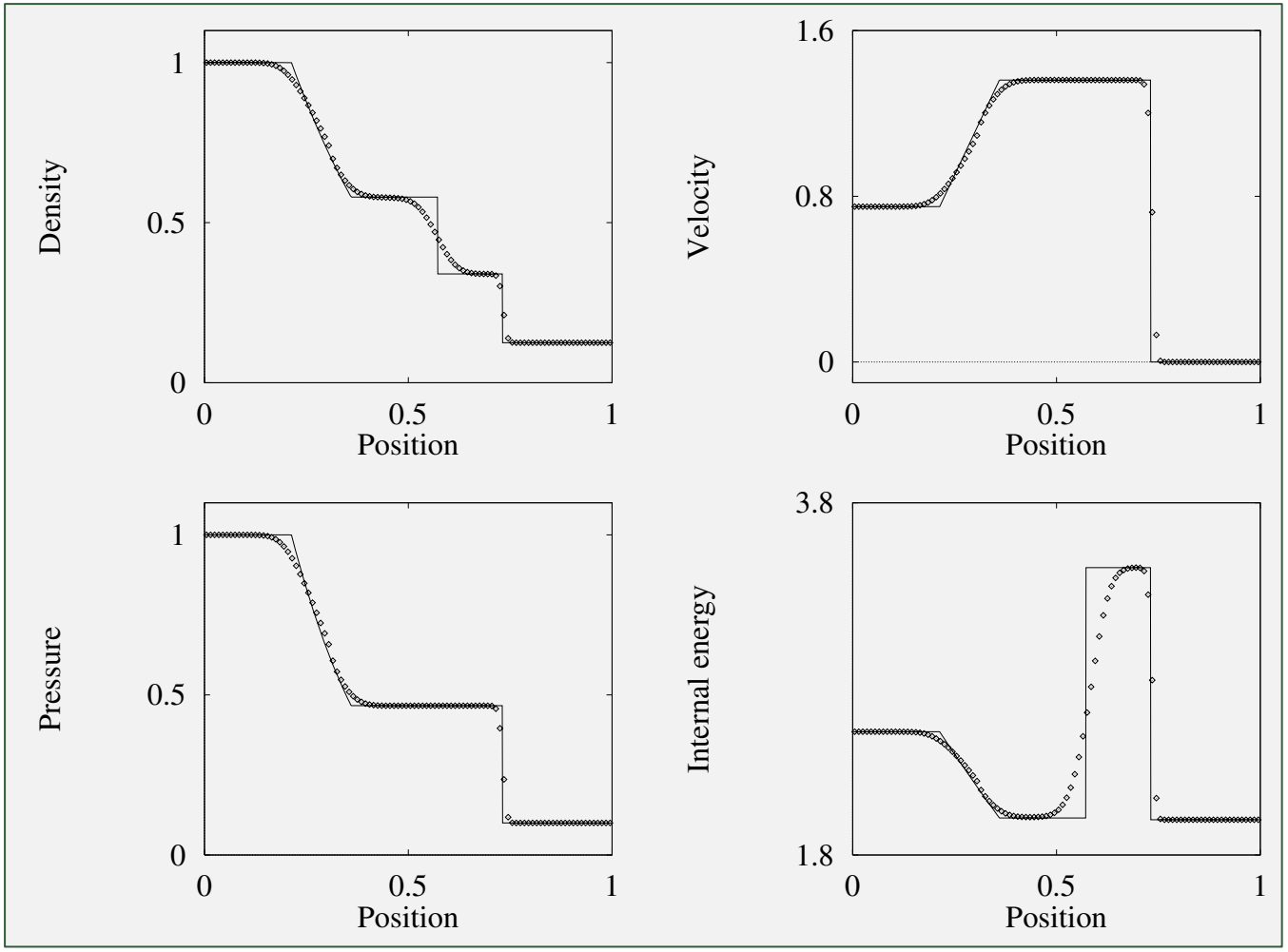


HLLC solver

HLL solver

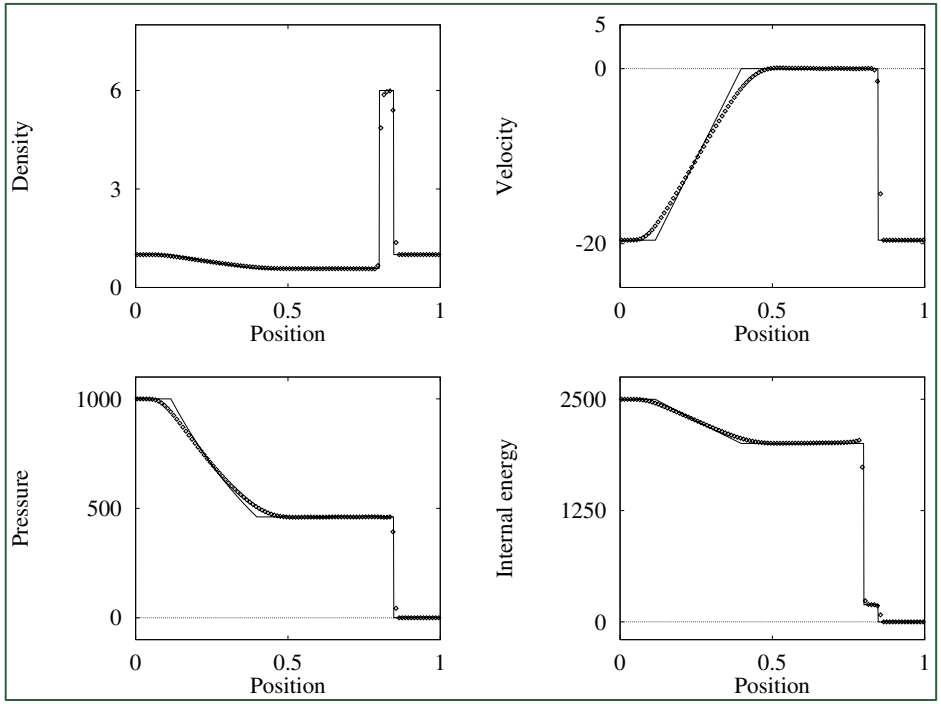
Test	ρ_{L}	u_{L}	p_{L}	ρ_{R}	u_{R}	p_{R}
1	1.0	0.75	1.0	0.125	0.0	0.1

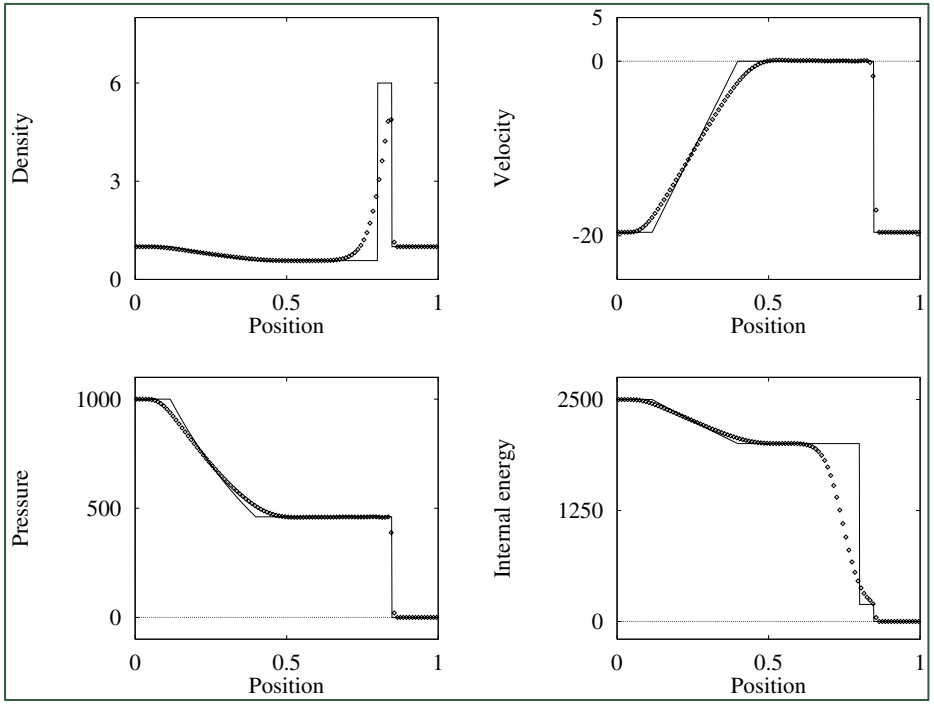
Shocks as a probe of numerical methods



Test	ρ_{L}	u_{L}	p_{L}	ρ_{R}	u_{R}	p_{R}
1	1.0	0.75	1.0	0.125	0.0	0.1

Shocks as a probe of numerical methods





HLLC solver

HLL solver

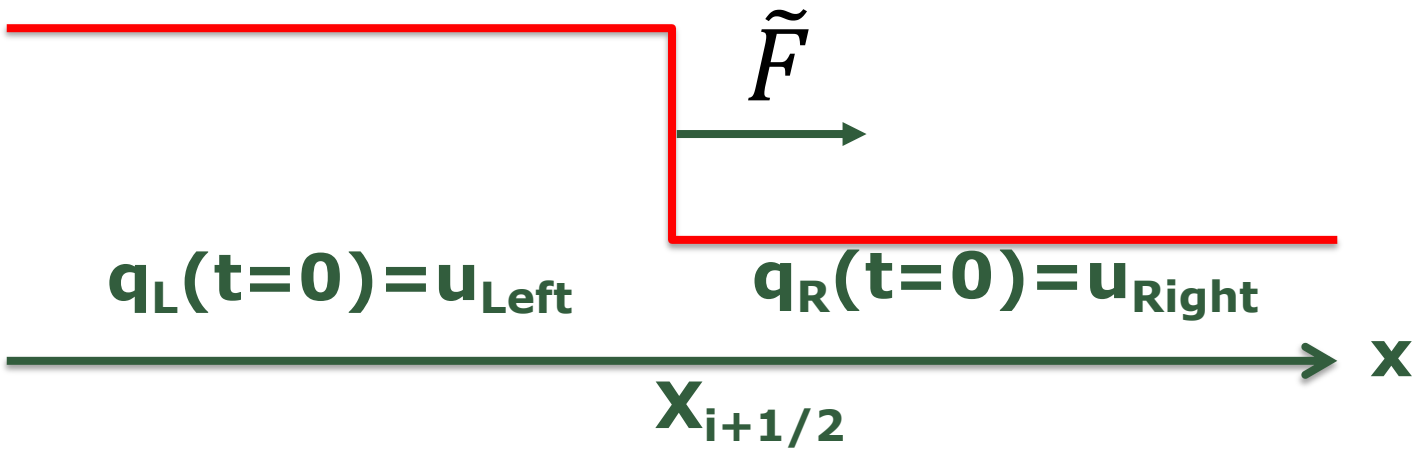
Test	ρ_{L}	u_{L}	p_{L}	ρ_{R}	u_{R}	p_{R}
5	1.0	-19.59745	1000.0	1.0	-19.59745	0.01

Godunov method

- Exploit the wave nature and full conservation properties of the equations and find the flux through a cell interface

- Use the flux to update the values in the cells

$$\tilde{F}_{i+1/2}^{n+1/2} = \frac{1}{\Delta t}\int_{t}^{t+\Delta t} dt\, F_{i+1/2}\left[q\left(x_{i+1/2}\right)\right]$$



Godunov method

- Exploit the wave nature and full conservation properties of the equations and find the flux through a cell interface

- Use the flux to update the values in the cells

$$\tilde{F}_{i+1/2}^{n+1/2} = \frac{1}{\Delta t}\int_{t}^{t+\Delta t} dt\, F_{i+1/2}\left[q\left(x_{i+1/2}\right)\right]$$

- Flux depends on wave speeds

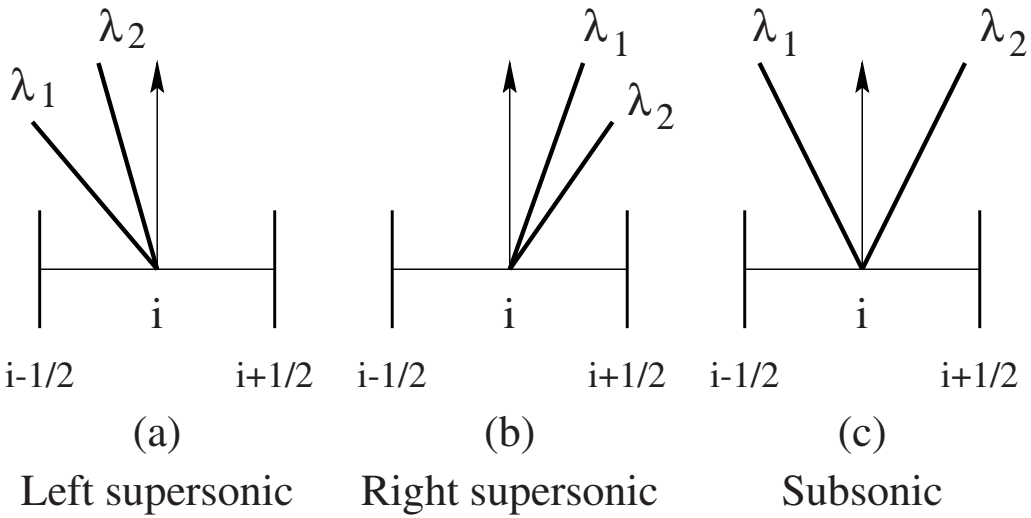
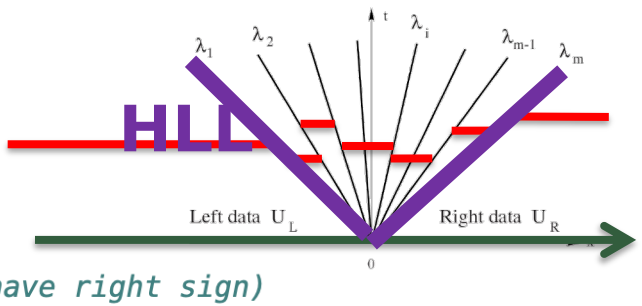


Fig. 8.2. Possible flow patterns in cell I_i at time n: (a) supersonic flow to the left (b) supersonic flow to the right (c) subsonic flow

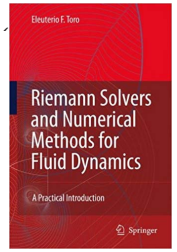
Approximate Riemann Solvers – HLL

```
# HLL (Harten, Lax, van Leer) is a bit less diffuse.
# We compute individual wave speeds for each state
# ql, qr are state vectors for the _primitive_ variables
def HLL(ql,qr):
    # sound speed for each side of interface (l==left, r==right)
    c_left  = (ql.gamma*ql.P / ql.D)**0.5
    c_right = (qr.gamma*qr.P / qr.D)**0.5
    c_max = np.maximum(c_left,c_right)

    # maximum wave speeds to the left and right (guaranteed to have right sign)
    SL = np.minimum(np.minimum(ql.U,qr.U)-c_max,0) # <= 0.
    SR = np.maximum(np.maximum(ql.U,qr.U)+c_max,0) # >= 0.

    # Hydro conservative variable
    Ul = primitive_to_conservative(ql)
    Ur = primitive_to_conservative(qr)

    # Hydro fluxes
    Fl = Hydro_Flux(ql,Ul)
    Fr = Hydro_Flux(qr,Ur)

    # HLL flux based on wavespeeds. If SL < 0 and SR > 0 then mix state appropriately
    # The general form is
    #    (SR * F_left - SL * F_right + SL * SR *(U_right - U_left)) / (SR - SL)
    # where U is the state vector of the conserved variables
    Flux = void()
    Flux.D  = (SR*Fl.D  - SL*Fr.D  + SL*SR*(Ur.D  - Ul.D )) / (SR - SL)
    Flux.mU = (SR*Fl.mU - SL*Fr.mU + SL*SR*(Ur.mU - Ul.mU)) / (SR - SL)
    # Flux.Etot = ...

    return Flux
```



Approximate Riemann Solvers – HLL

$$\int_{x_L}^{x_R} \mathbf{U}(x,T)dx = \int_{x_L}^{x_R} \mathbf{U}(x,0)dx + \int_0^T \mathbf{F}(\mathbf{U}(x_L,t))dt - \int_0^T \mathbf{F}(\mathbf{U}(x_R,t))dt$$

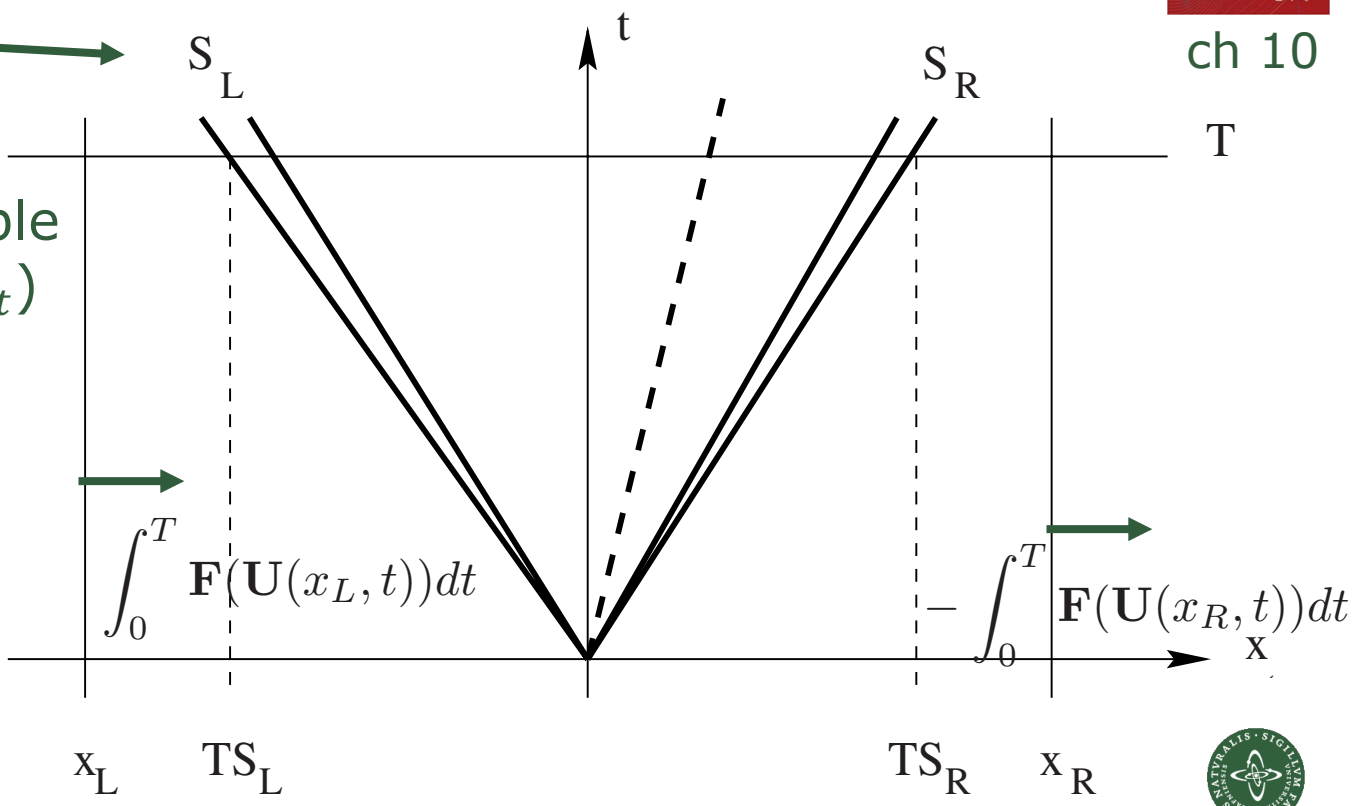


ch 10

Fastest speed

U: conserved variable
U = $(\rho, \rho u, \rho v, \rho w, E_{tot})$
F: flux for U

Strategy:
Exploit that U is conserved and only changes due to flux at boundaries

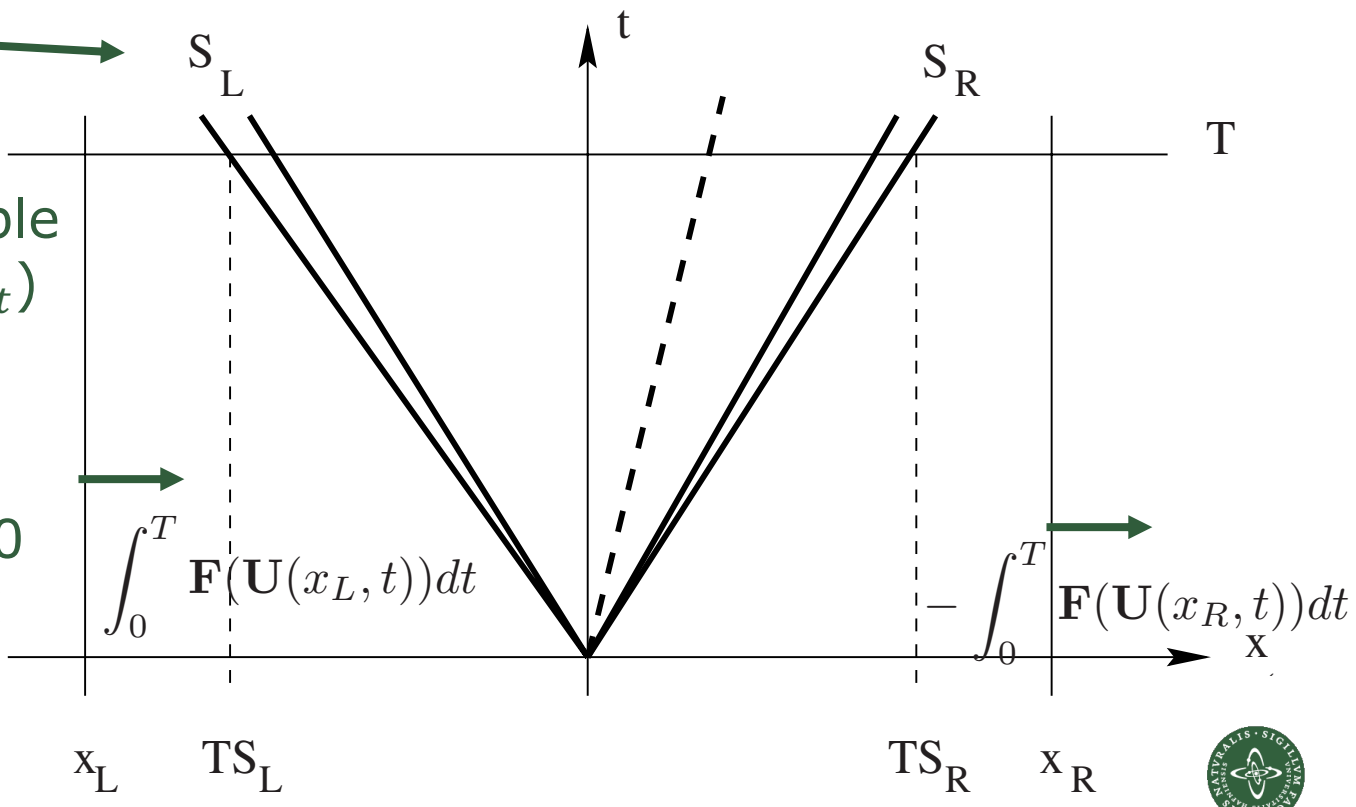


Approximate Riemann Solvers – HLL

$$\int_{x_L}^{x_R} \mathbf{U}(x,T)dx = \int_{x_L}^{x_R} \mathbf{U}(x,0)dx + \int_0^T \mathbf{F}(\mathbf{U}(x_L,t))dt - \int_0^T \mathbf{F}(\mathbf{U}(x_R,t))dt$$

$$= x_R\mathbf{U}_R - x_L\mathbf{U}_L + T(\mathbf{F}_L - \mathbf{F}_R)$$

Fastest speed



U: conserved variable
U = $(\rho, \rho u, \rho v, \rho w, E_{tot})$
F: flux for U

Integrals:
U is constant at T=0
changes due to flux at boundaries, but U constant at boundaries

Approximate Riemann Solvers – HLL

$$\int_{x_L}^{x_R} \mathbf{U}(x,T)dx = \int_{x_L}^{TS_L} \mathbf{U}(x,T)dx + \int_{TS_L}^{TS_R} \mathbf{U}(x,T)dx + \int_{TS_R}^{x_R} \mathbf{U}(x,T)dx$$

U: conserved variable

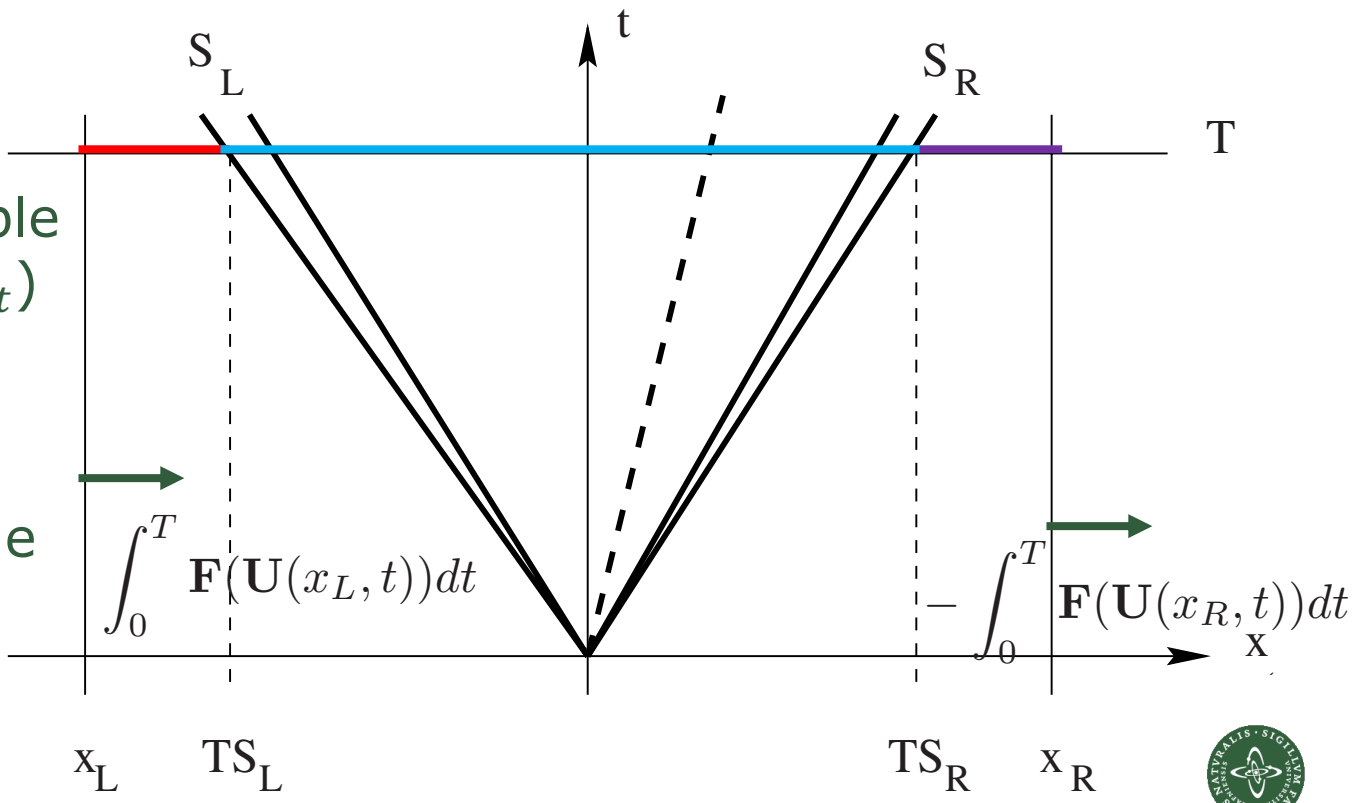
U = $(\rho, \rho u, \rho v, \rho w, E_{tot})$

F: flux for U

Spatial integrals:

U is constant outside

[S_L: S_R]-cone



Approximate Riemann Solvers – HLL

$$\int_{x_L}^{x_R} \mathbf{U}(x,T)dx = \int_{x_L}^{TS_L} \mathbf{U}(x,T)dx + \int_{TS_L}^{TS_R} \mathbf{U}(x,T)dx + \int_{TS_R}^{x_R} \mathbf{U}(x,T)dx$$

$$= \int_{TS_L}^{TS_R} \mathbf{U}(x,T)dx + (TS_L - x_L)\mathbf{U}_L + (x_R - TS_R)\mathbf{U}_R$$

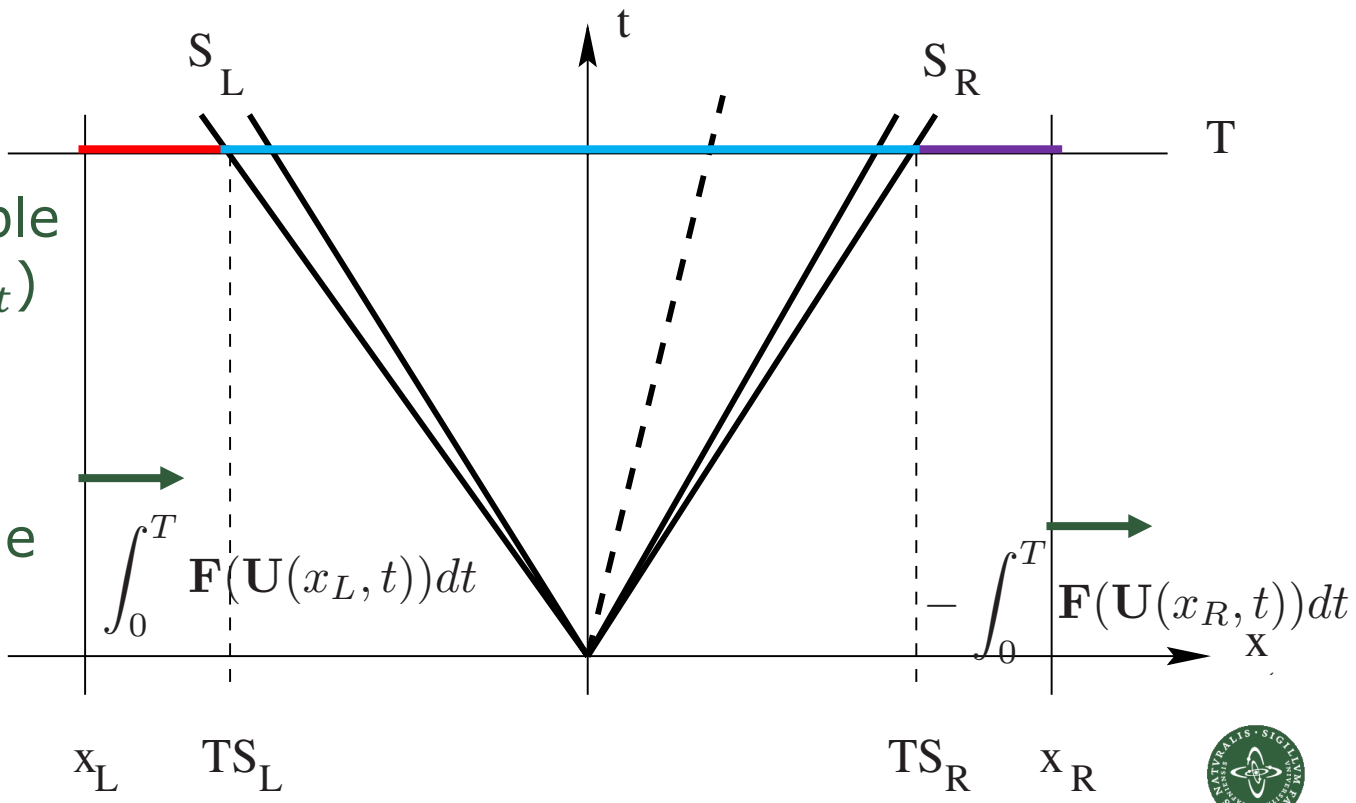
U: conserved variable

U = $(\rho, \rho u, \rho v, \rho w, E_{tot})$

F: flux for U

Spatial integrals:

U is constant outside [S_L: S_R]-cone



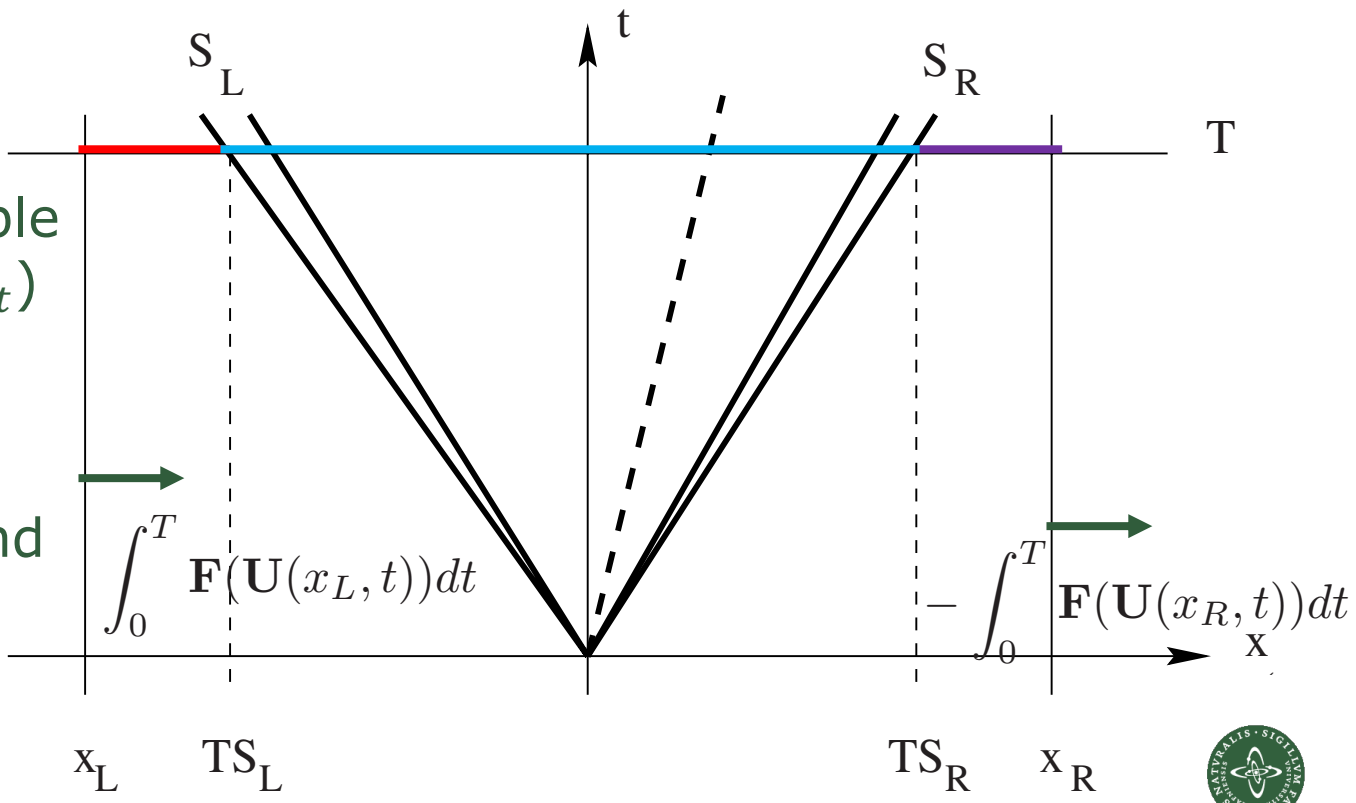
Approximate Riemann Solvers – HLL

$$\int_{x_L}^{x_R} \mathbf{U}(x,T)dx = \int_{x_L}^{TS_L} \mathbf{U}(x,T)dx + \int_{TS_L}^{TS_R} \mathbf{U}(x,T)dx + \int_{TS_R}^{x_R} \mathbf{U}(x,T)dx$$

$$x_R\mathbf{U}_R - x_L\mathbf{U}_L + T(\mathbf{F}_L - \mathbf{F}_R) = \int_{TS_L}^{TS_R} \mathbf{U}(x,T)dx + (TS_L - x_L)\mathbf{U}_L + (x_R - TS_R)\mathbf{U}_R$$

U: conserved variable
U = $(\rho, \rho u, \rho v, \rho w, E_{tot})$
F: flux for U

Combine:
Combine results, and then arbitrary boundaries (x_L, x_R) drop out



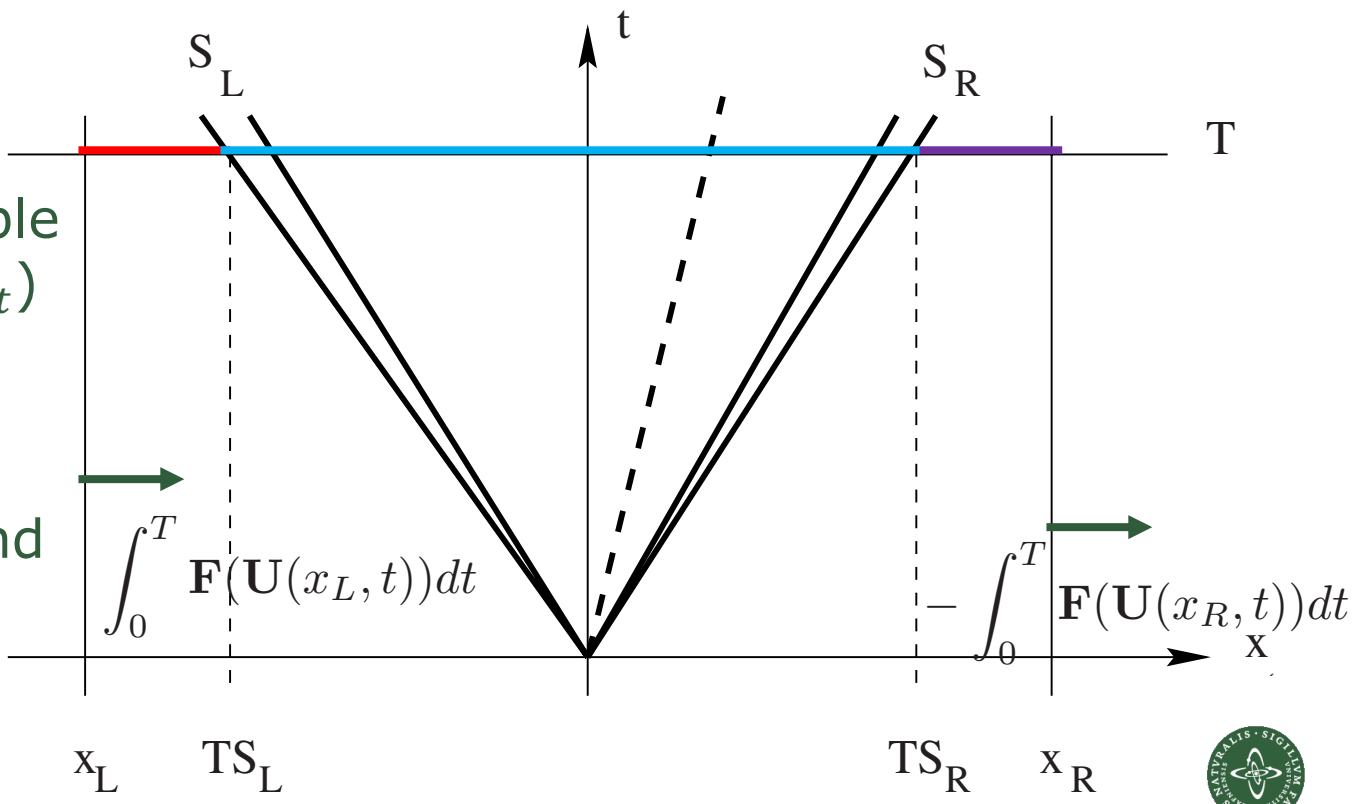
Approximate Riemann Solvers – HLL

$$\frac{1}{T(S_R - S_L)} \int_{TS_L}^{TS_R} \mathbf{U}(x,T)dx = \frac{S_R\mathbf{U}_R - S_L\mathbf{U}_L + F_L - F_R}{S_R - S_L}$$

$$x_R\mathbf{U}_R - x_L\mathbf{U}_L + T(\mathbf{F}_L - \mathbf{F}_R) = \int_{TS_L}^{TS_R} \mathbf{U}(x,T)dx + (TS_L - x_L)\mathbf{U}_L + (x_R - TS_R)\mathbf{U}_R$$

U: conserved variable
U = $(\rho, \rho u, \rho v, \rho w, E_{tot})$
F: flux for U

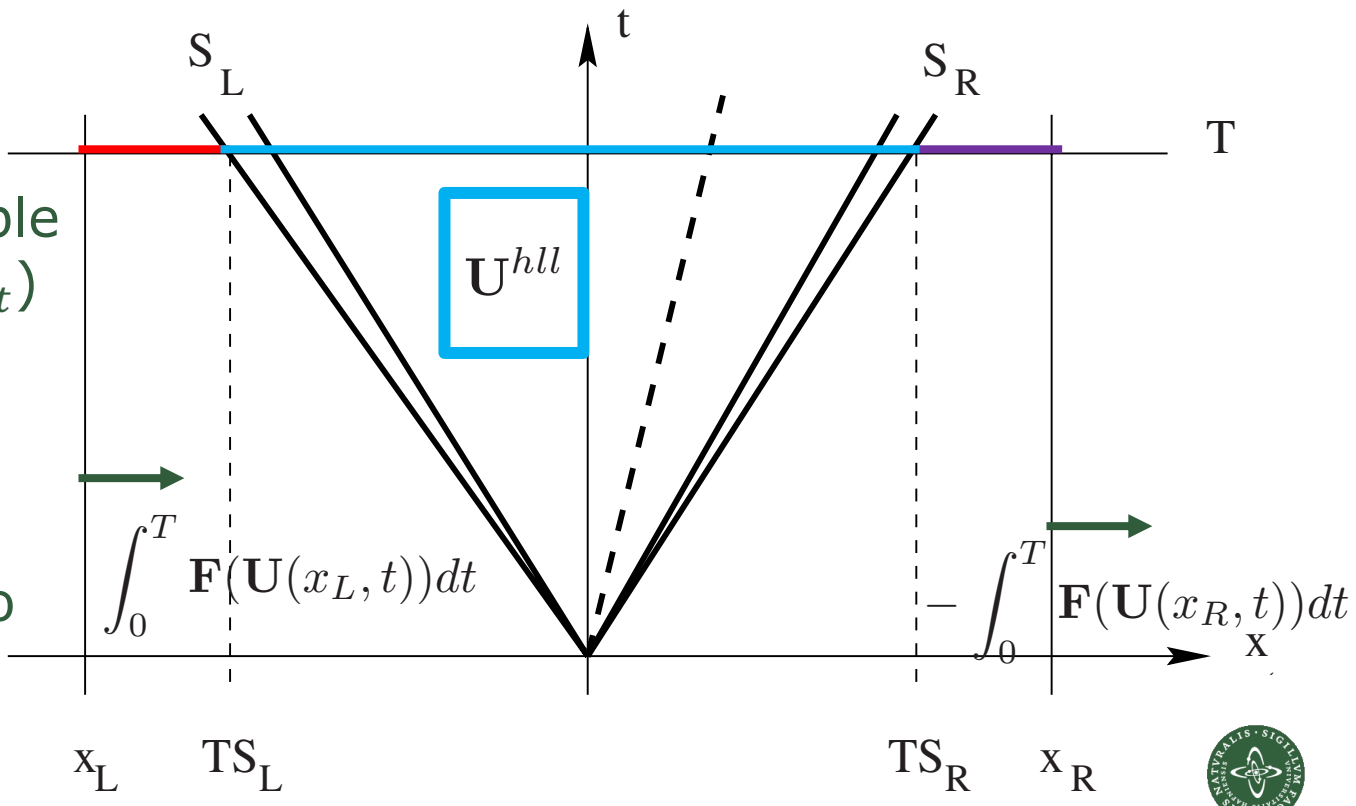
Combine:
Combine results, and then arbitrary boundaries (x_L, x_R) drop out



Approximate Riemann Solvers – HLL

$$\mathbf{U}^{hll} = \frac{1}{T(S_R - S_L)} \int_{TS_L}^{TS_R} \mathbf{U}(x,T)dx = \frac{S_R\mathbf{U}_R - S_L\mathbf{U}_L + F_L - F_R}{S_R - S_L}$$

$$x_R\mathbf{U}_R - x_L\mathbf{U}_L + T(\mathbf{F}_L - \mathbf{F}_R) = \int_{TS_L}^{TS_R} \mathbf{U}(x,T)dx + (TS_L - x_L)\mathbf{U}_L + (x_R - TS_R)\mathbf{U}_R$$



U: conserved variable
U = $(\rho, \rho u, \rho v, \rho w, E_{tot})$
F: flux for U

Average state:
U is conserved and only changes due to flux at boundaries.
New average state: $\mathbf{U}^{hll}$

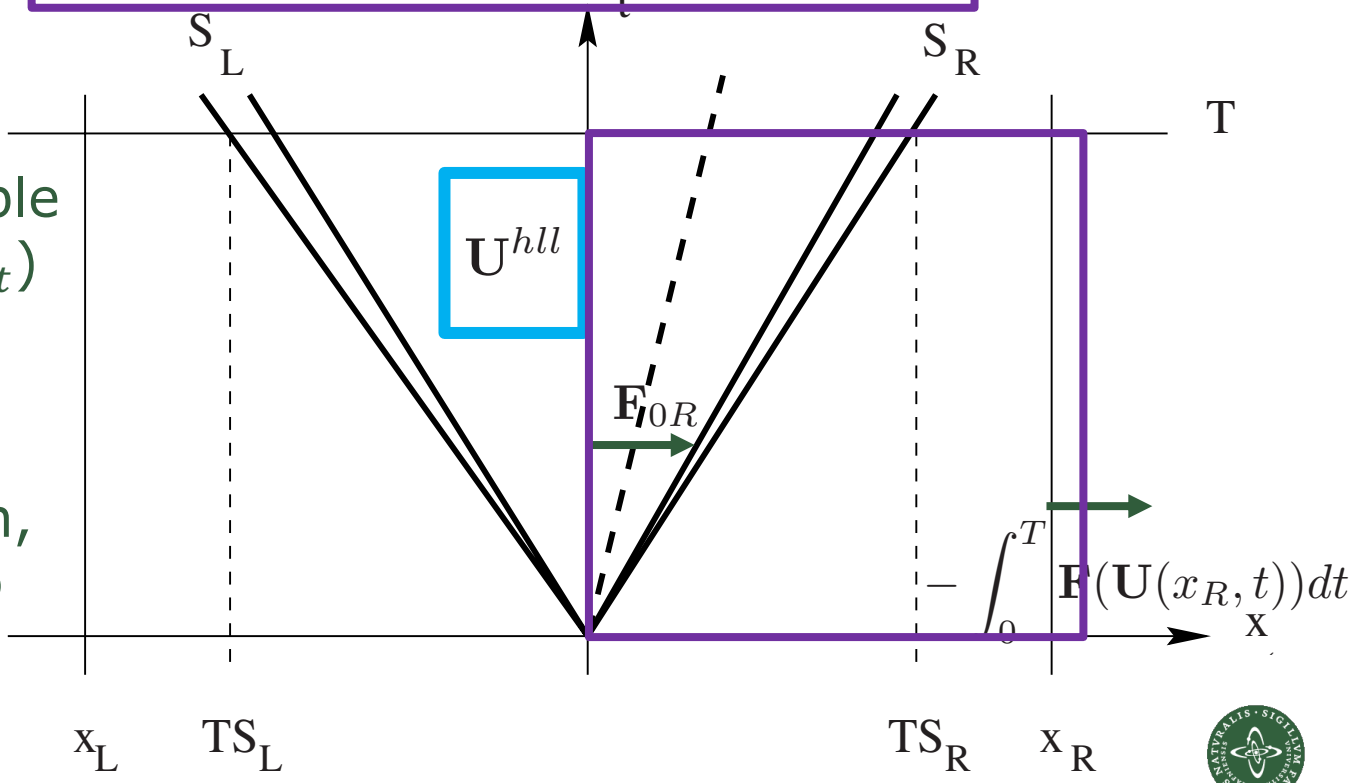
Approximate Riemann Solvers – HLL

$$\mathbf{U}^{hll} = \frac{1}{T(S_R - S_L)} \int_{TS_L}^{TS_R} \mathbf{U}(x,T)dx = \frac{S_R\mathbf{U}_R - S_L\mathbf{U}_L + F_L - F_R}{S_R - S_L}$$

$$\mathbf{F}_{0R} = \mathbf{F}_R - S_R\mathbf{U}_R + \frac{1}{T}\int_0^{TS_R} \mathbf{U}(x,T)dx$$

U: conserved variable
U = $(\rho, \rho u, \rho v, \rho w, E_{tot})$
F: flux for U

Interface flux:
Repeat computation, but only at x > 0 to find interface flux



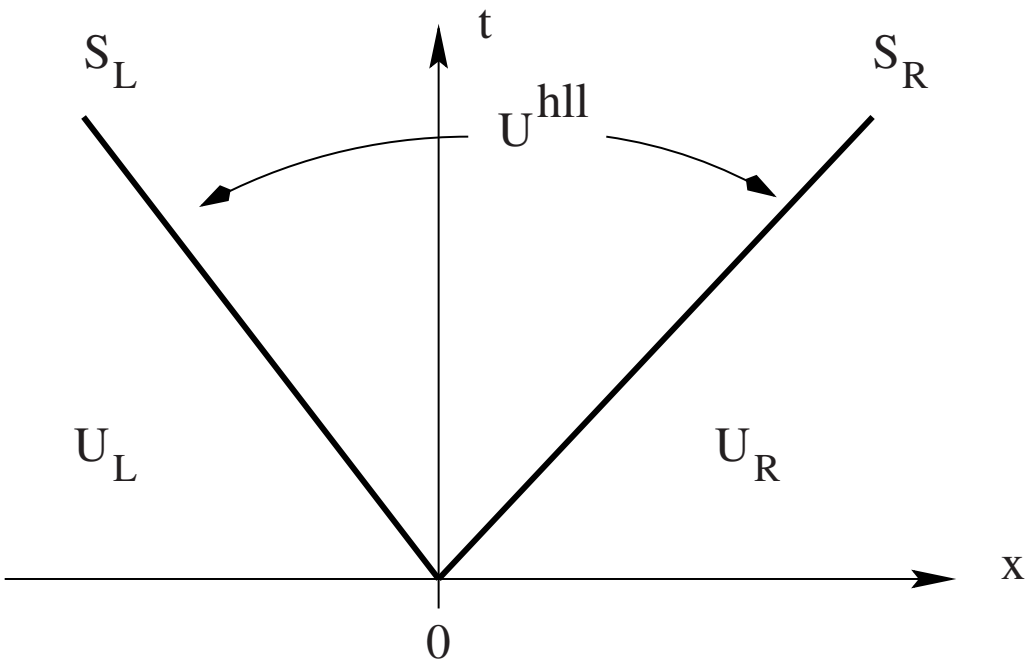
Approximate Riemann Solvers – HLL

$$\mathbf{U}^{hll} = \frac{1}{T(S_R - S_L)} \int_{TS_L}^{TS_R} \mathbf{U}(x,T)dx = \frac{S_R\mathbf{U}_R - S_L\mathbf{U}_L + F_L - F_R}{S_R - S_L}$$

$$\mathbf{F}_{0R} = \mathbf{F}^{hll} = \frac{S_R\mathbf{F}_L - S_L\mathbf{F}_R + S_L S_R(\mathbf{U}_R - \mathbf{U}_L)}{S_R - S_L}$$

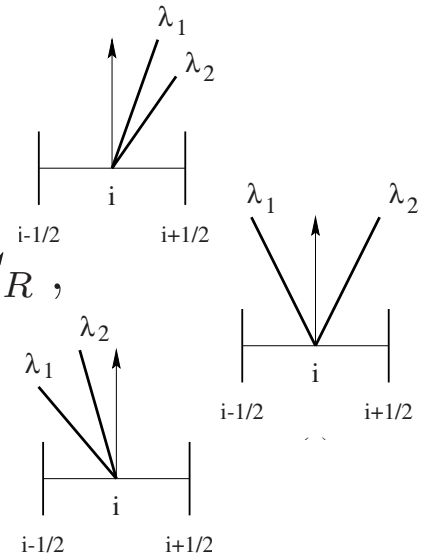
U: conserved variable
U = $(\rho, \rho u, \rho v, \rho w, E_{tot})$
F: flux for U

Interface flux:
Evaluate integral with ***U^hll state*** and rearrange to find ***HLL flux***



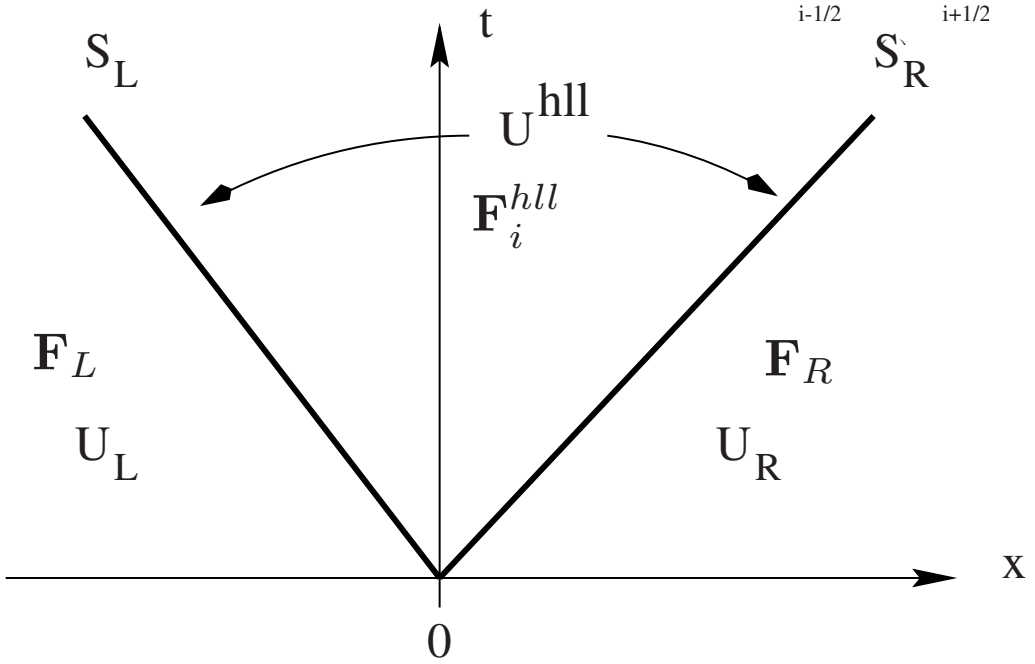
Approximate Riemann Solvers – HLL

$$\mathbf{F}_i^{hll} = \begin{cases} \mathbf{F}_L & \text{if} \quad 0 \le S_L \,, \\ \dfrac{S_R\mathbf{F}_L - S_L\mathbf{F}_R + S_L S_R(\mathbf{U}_R - \mathbf{U}_L)}{S_R - S_L} \,, & \text{if} \quad S_L \le 0 \le S_R \,, \\ \mathbf{F}_R & \text{if} \quad 0 \ge S_R \,. \end{cases}$$



U: conserved variable
U = $(\rho, \rho u, \rho v, \rho w, E_{tot})$
F: flux for U

S_L: $\min(u_L,u_R) - c_{sound}$
S_R: $\max(u_L,u_R) + c_{sound}$



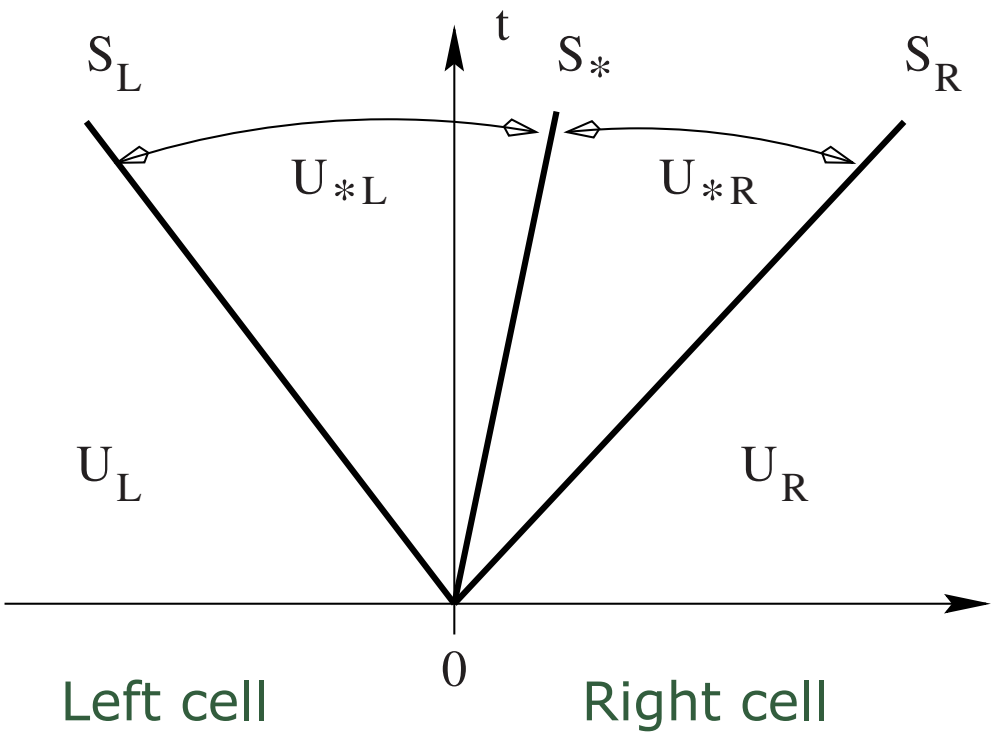
Approximate Riemann Solvers – HLLC

- HLLC solver adds a new characteristic speed
- Less diffusive than HLL, more physical
- Very stable solver for hydrodynamics
- Widely used method of choice in codes!
- States found by assuming pressure and velocity continous at S_*
- Review by Toro 2019 - doi: 10.1007/s00193-019-00912-4

U: conserved variable
$U = (\rho, \rho u, \rho v, \rho w, E_{tot})$
F: flux for U

S_L: $\min(u_L, u_R) - c_{sound}$
S_R: $\max(u_L, u_R) + c_{sound}$

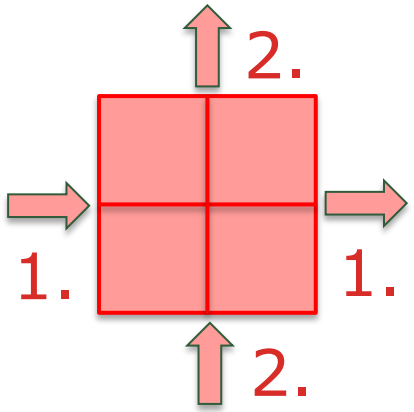
S_*: determined from jump conditions!!



Higher dimensionalities

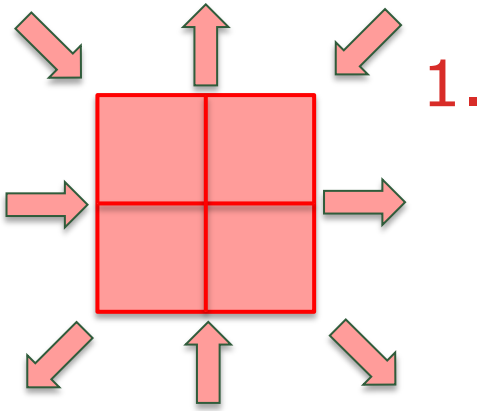
(non)-split fluxes:

- Compute fluxes sequen-tially thorugh each edge (face) in 2D (3D).
- Problem is overadvection where density or pressure becomes negative
- Solution also not manifestly isotropic
- Can induce *carbuncle* instability

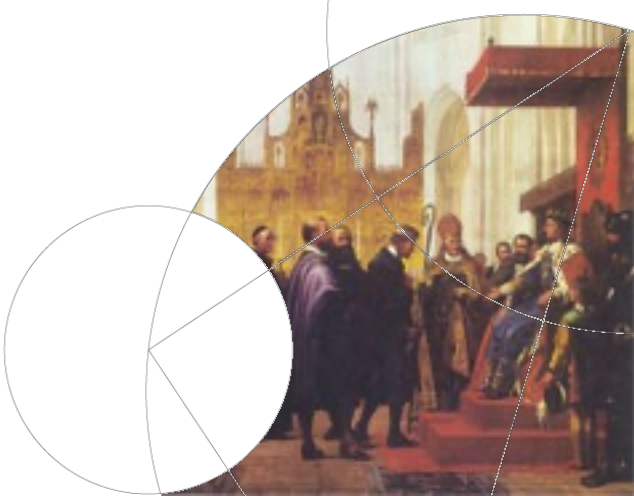


Higher dimensionality fluxes:

- Compute fluxes thorugh each edge and corner (face, edge) in 2D (3D).
- This is much more isotropic
- Super complicated to solve Riemann problem.
- Done only by few groups. Implementation papers are small books.



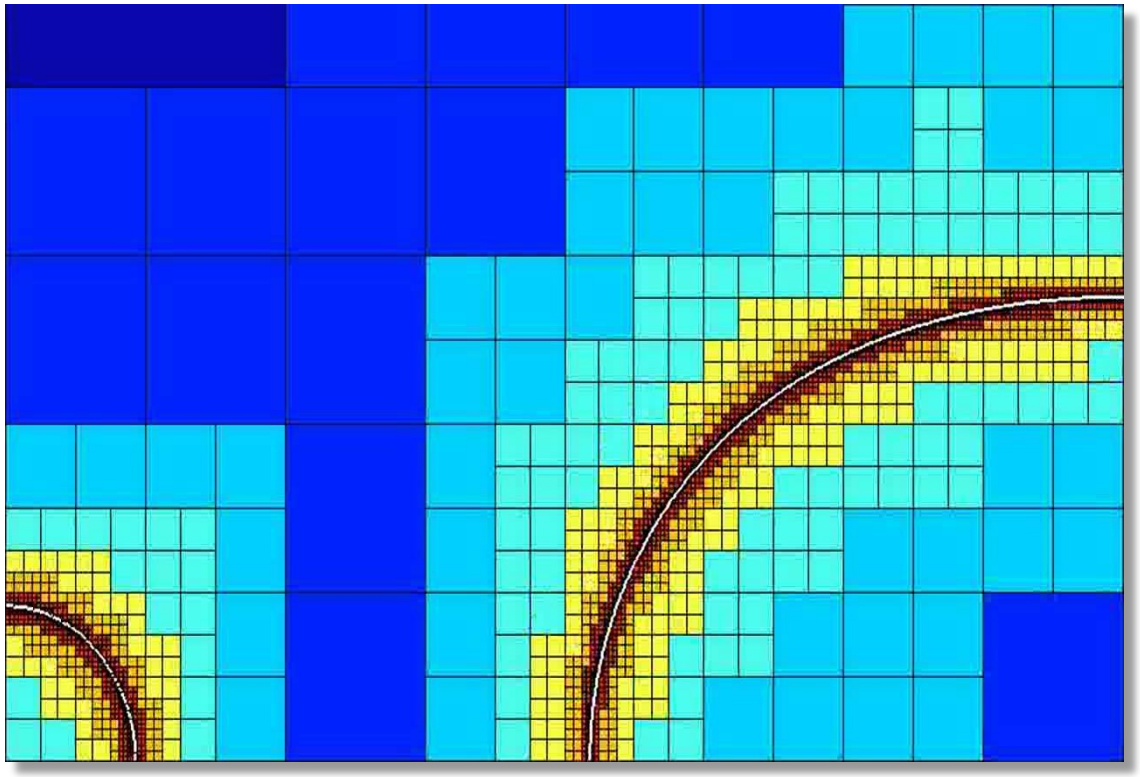
Adaptive Mesh Refinement (AMR)



What is Adaptive Mesh Refinement (AMR) ?

Some of you may know about Adaptive Mesh Refinement or have heard about the concept. But here is a quick summary:

- AMR-codes can (recursively) resolve small details, by using patches (small or large) with increasingly large resolutions
- It is *adaptive*, because the cell placement can change with time



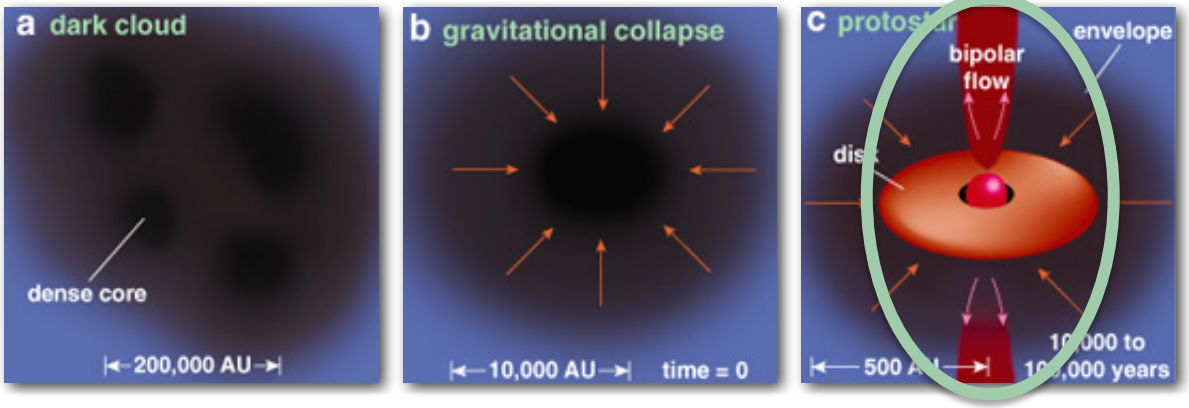
Motivations for Adaptive Mesh Refinement

- Fluid dynamics in three dimensions is costly:
  - Cost of a uniform grid scales as the resolution to the fourth power
  - Even today only ~1024^3 is routine, and the largest unigrid run to date is ~16384^3
- Many problems in astrophysics contain relevant, coupled processes at very different scales
  - Use a sub-grid model description
  - Use different resolution at different places → AMR in space
- If velocities are approximately (order of magnitude!) constant the dynamical time-scale scales with the physical scale
  - Large scales evolve slower than small scales → AMR in time
- Using adaptive meshes we can "easily" supply realistic boundaries to a local problem; the ladder of astrophysical AMR is:
  - Cosmology → Galaxy formation → Star formation → Planet Formation

Multiscale Astrophysics

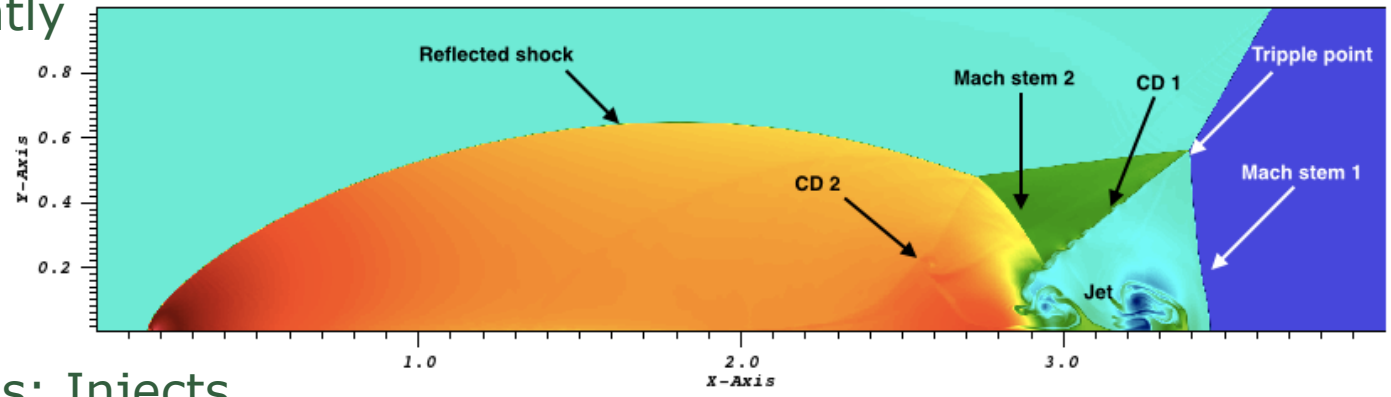
[Spitzer Science Center]

- Selfgravity: induces collapse, with a rapid decrease in scales

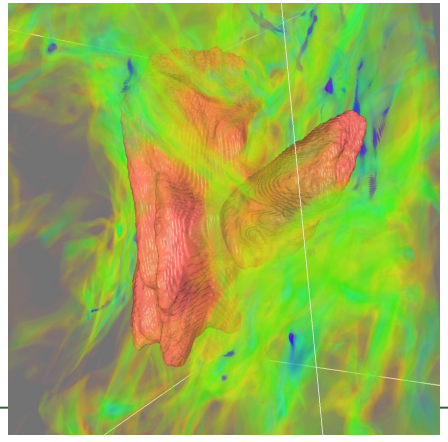


[Double Mach shock with Flash]

- Shocks: inherently localized; often part of complex flows



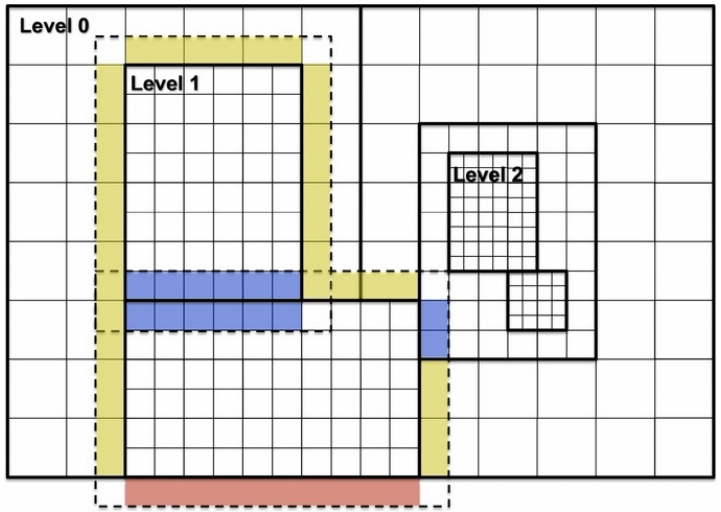
- Compact sources: Injects energy from the smallest scales to the largest



[Ionization front in a molecular cloud launched by central source]

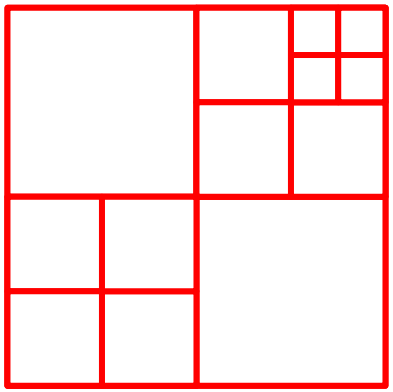
Flavors of Adaptive Mesh Refinement

- Block based AMR (original Collela, Berger, Oliger ***'84 & '89***) [FLASH, **Enzo, Nirvana, Pluto, AZeus**]
  - Use patches of higher resolution completely contained inside lower resolution patches

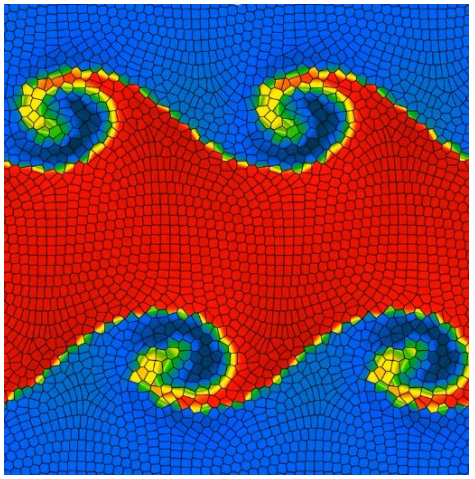


[PLUTO AMR paper]

- Oct based Fully-Threaded-Tree (Khokhlov '98) [**RAMSES**, AMRVAC, ART]
  - Refine on a cell-by-cell basis, with one cell being split in to 8 (in 3D)

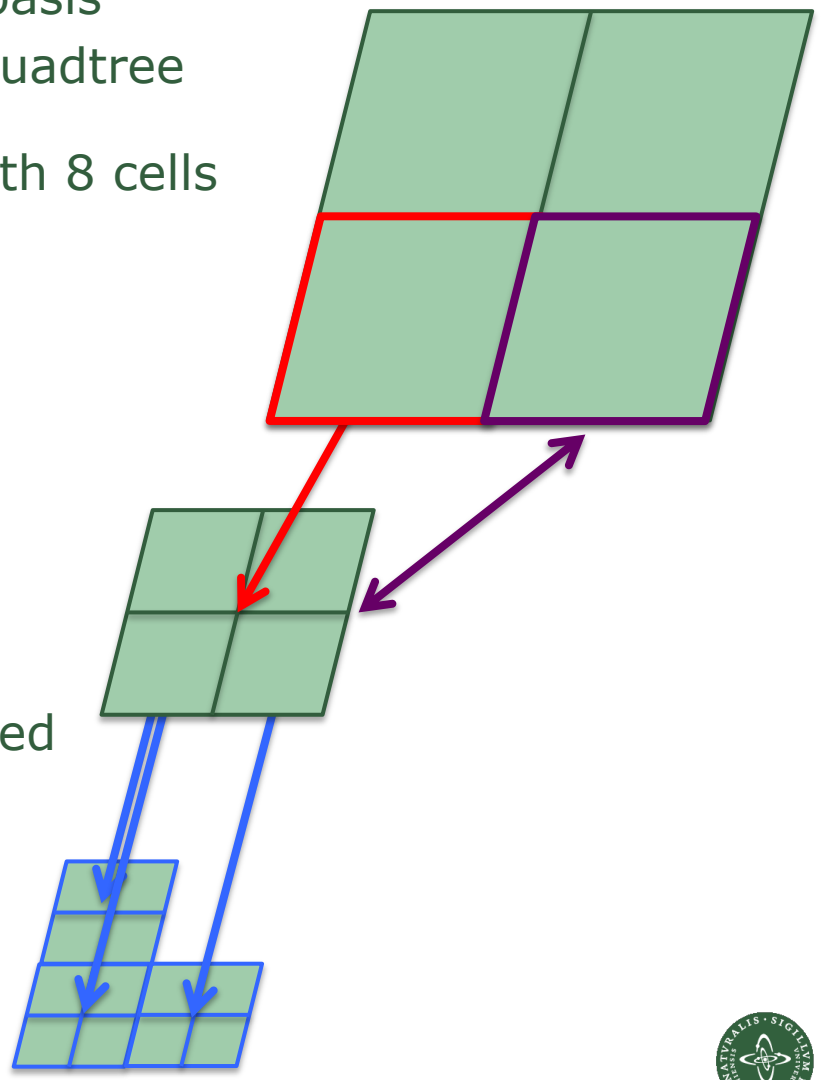


- Unstructured Meshes [AREPO, GIZMO, finite elements]
  - Partition space using one volume per tracer particle; f.ex. using a Voronoi tessellation.



Fully-Threaded-Tree Adaptive Mesh Refinment

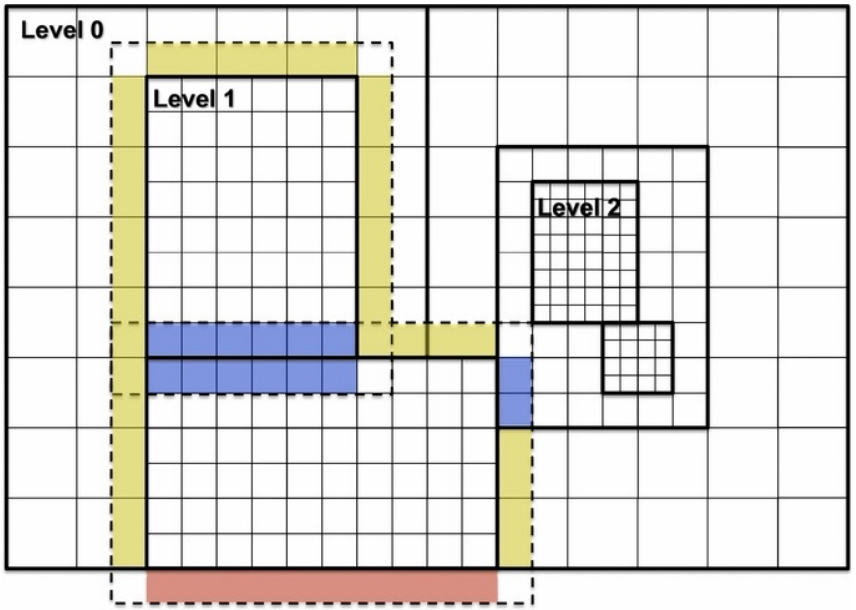
- Refinement is done on a cell-by-cell basis
  - 3D: octree (8 cells per oct), 2D: quadtree
- Each cell can be refined to one oct with 8 cells
  - Very adaptive grid
- Very simple relationship structure
  - 1 parent cell
  - 6 neighbor parent cells
  - Potentially 8 children octs
- Everything is constructed recursively
  - Position and relationships are picked up from parent cell at creation
- All cells in the tree are kept
  - Leaf cells have no children octs
  - Refined cells are inactive



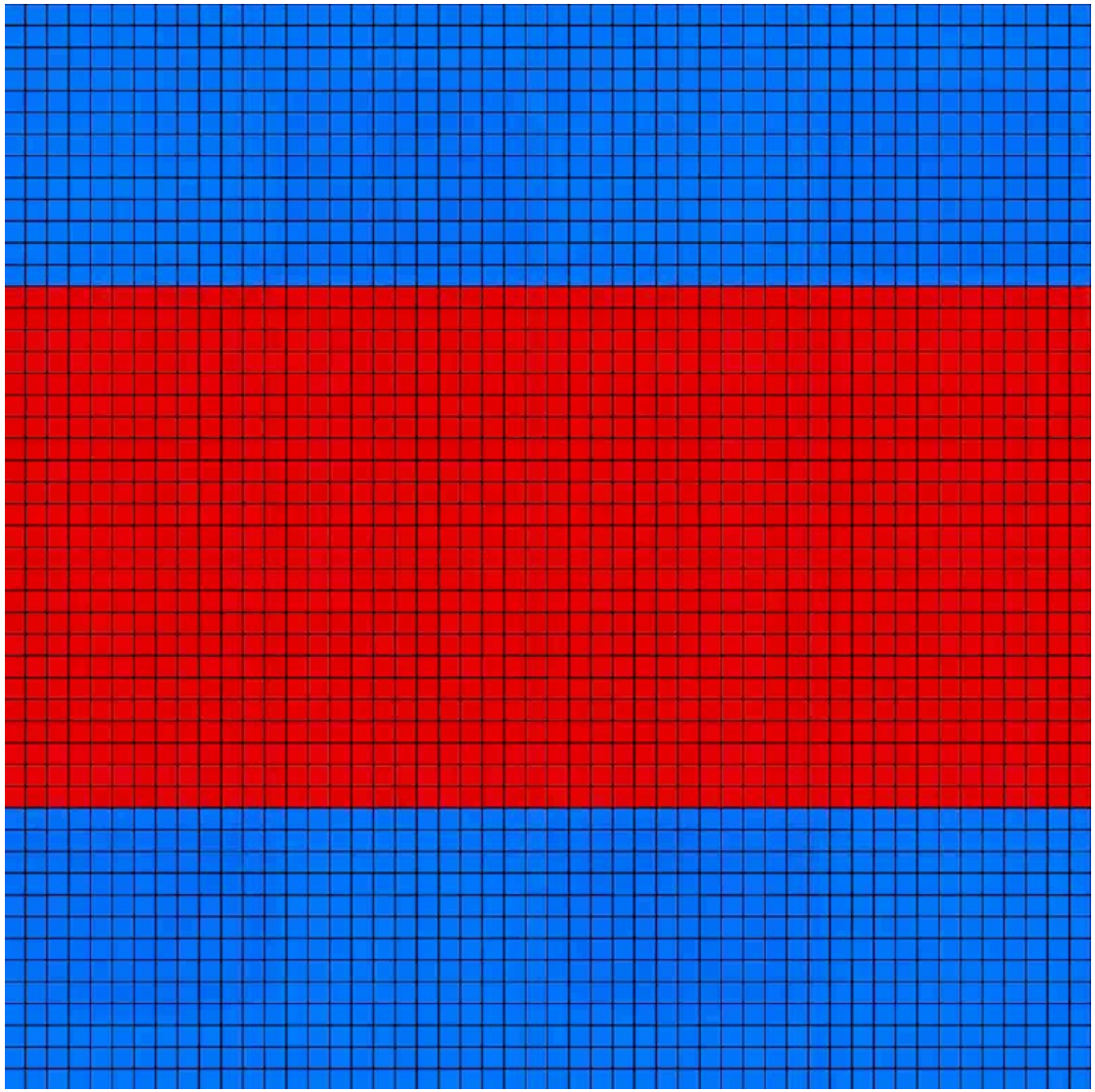
Block based Adaptive Mesh Refinement

- Refinement is done in a number of cells inside a patch to create a new patch
- Typical size of a patch is $(12\text{-}16)^{Ndim}$
  - Reasonable grid size. Efficient to manage
- Complex but flexible relationship structure
  - 1 parent patch (in simplest version)
  - N_{bor} neighbor patches (at different levels)
  - N_{child} children patches
- Everything is constructed recursively
  - Position and relationships are picked up from parent patches at creation
- Normally all cells in a patch are kept
  - Leaf cells have no patches on top
  - Refined cells are inactive

[PLUTO AMR paper]

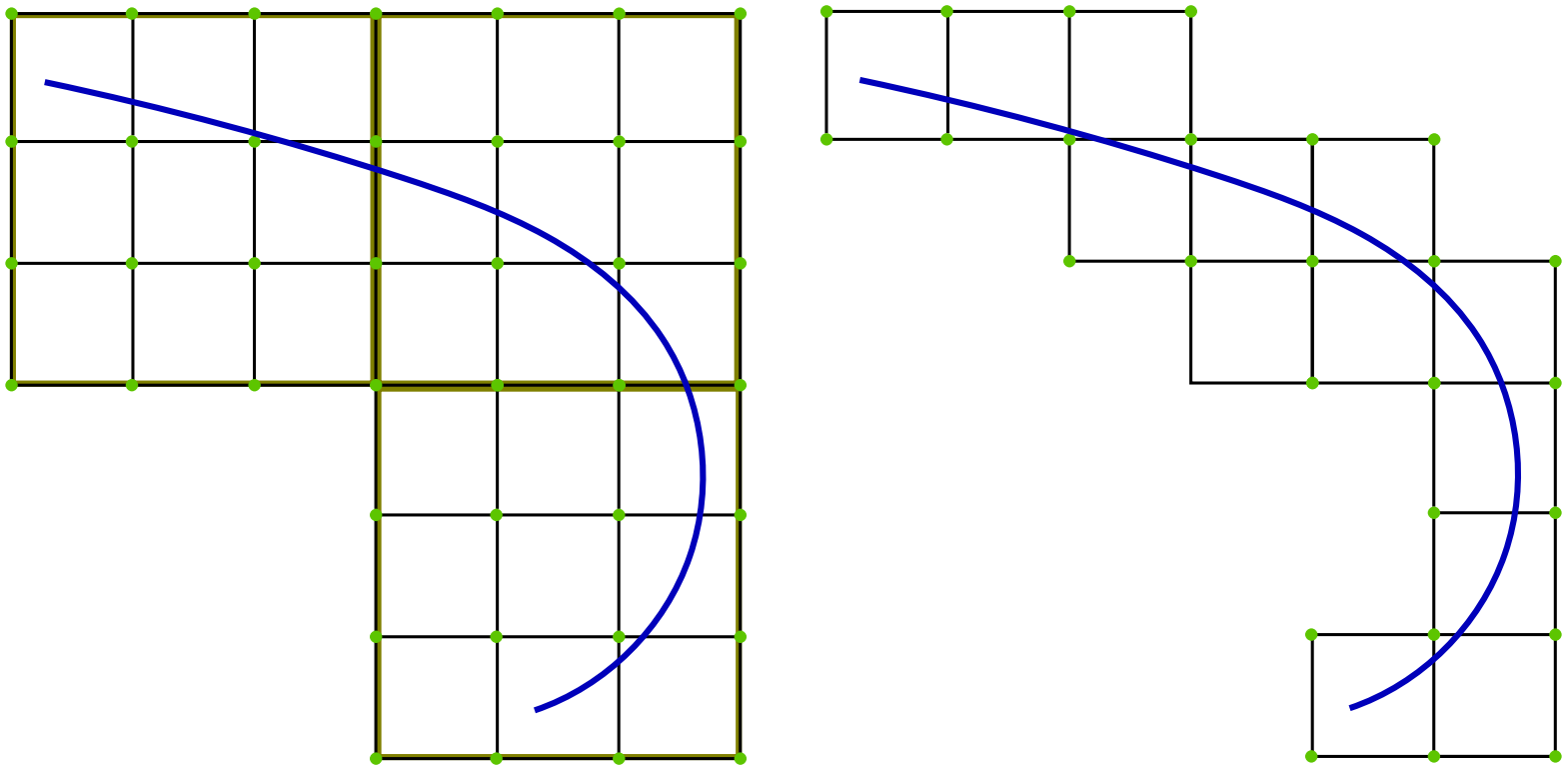


Unstructured (moving) meshes



- Unstructured codes are using tracer points for their geometry.
- F.x. the AREPO and GIZMO codes use unstructured and meshless representations, respectively.
- Their representation has the important advantage (over fixed-grid codes), to respect Galilean invariance; i.e., their results are the same, independent of any bulk motion of the system under study.
- The Courant condition is only due to *relative motion*.
- But cost is high!

Patch versus Tree based AMR



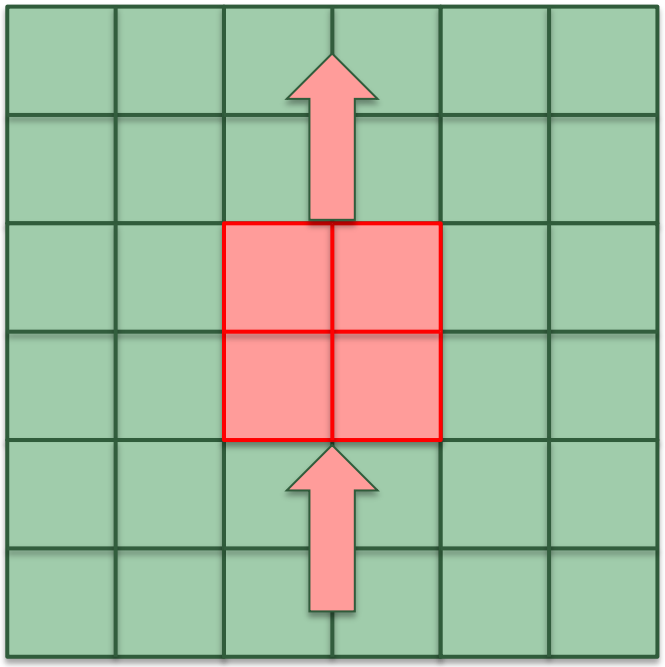
- A patch based AMR structure is much better at adapting to complex flows, like turbulence
- CPU overhead is larger; hard to vectorize efficiently due to oct-by-oct structure. Difficult to move to GPUs

Ingredients of an AMR method

Fluid Dynamics on an AMR patch / in an oct

- The fluid dynamics in an adaptive mesh is solved exactly as on a normal mesh

1. Extract patch
2. Pick up "guard cells":
   1. Check if neighbors exist
   2. Else create new cells on-the-fly

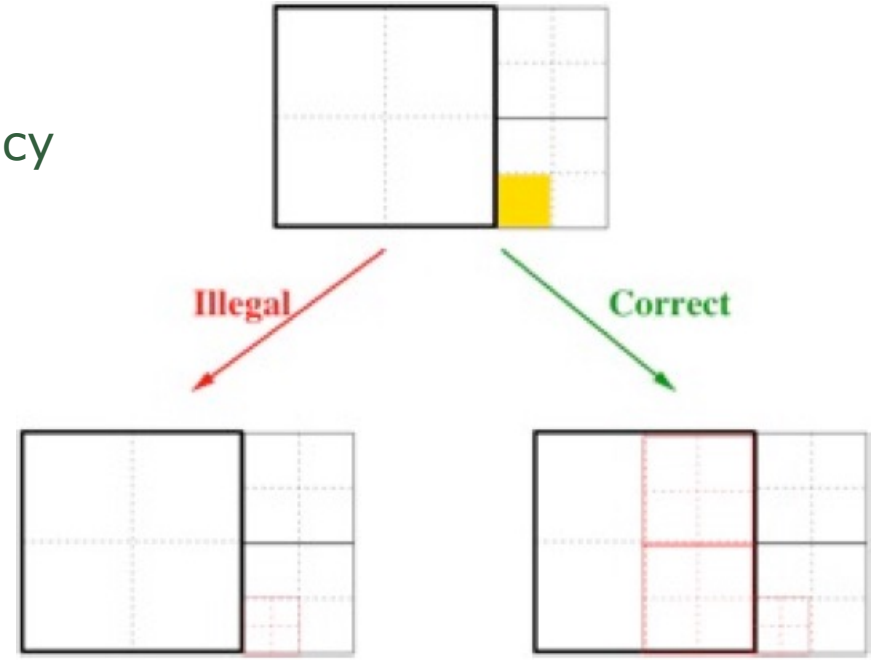


- Calculate fluxes with a finite volume method
- Hard to use higher order methods; we typically only use 2 guard cells to maintain a reasonable surface-to-volume ratio
  - Motivates the use of Godunov methods or very compact finite differences
- *Parent neighbors always need to exist*, or we cannot get boundaries on-the-fly

AMR Refinement Rules

- Most codes enforce mesh consistency and grading of patches
  - All neighbors have to be at the same level or one level below or above

- Specific Criteria
  - Jeans Criteria
  - Gradients
  - Quasi-Lagranian
  - Geometrical (zoom)



Refinement Criteria

- Jeans / Truelove Criteria
  - If the Jeans length is not resolved, collapse becomes unphysical
- Gradients
  - Beware of shocks; if encounter a real jump you get AMR catastrophe -> individual max level for gradient-refinement
- Quasi-Lagranian
  - Related to Jeans. F.x. refine every time density goes up with 4 to have same Jeans resolution on all levels
- Geometrical (zoom)
  - Only allow code to refine in specific region. Adapt center to flow; Gallilean transform; follow star

Refinement Criteria

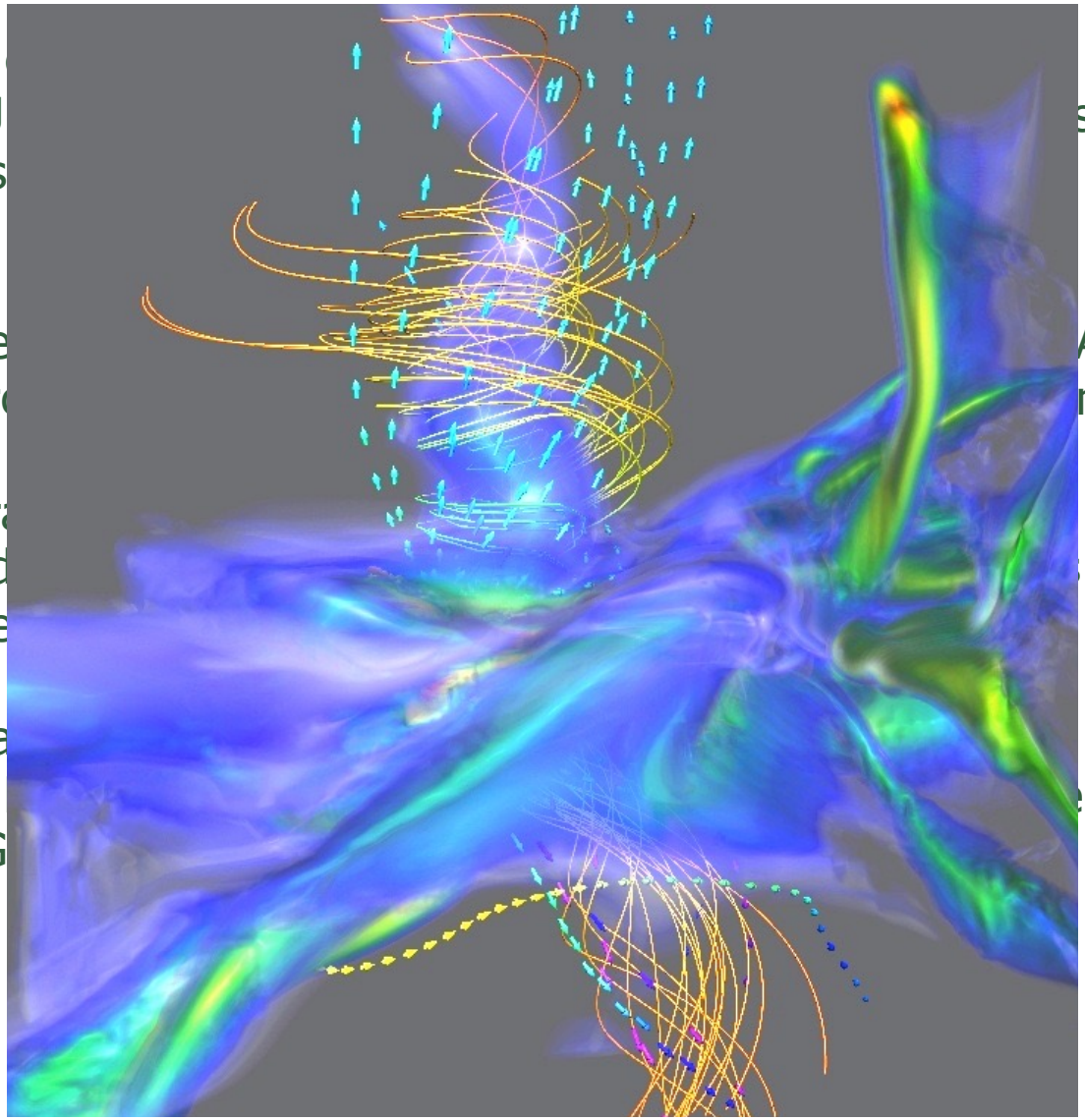
- Jeans / Tru
  - If the J
    unphys
- Gradients
  - Beware
    catastr
- Quasi-Lagra
  - Related
    to have
- Geometrica
  - Only al
    flow; G

s

AMR
nement

up with 4

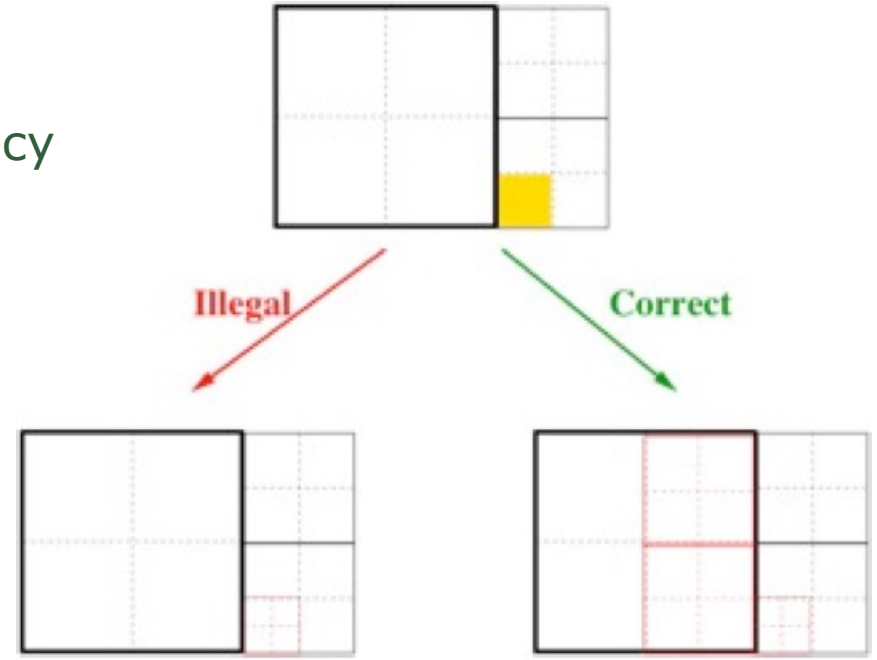
nter to



[Refinement to pick up Jet; VAPOR viz by M. Küffmeier]

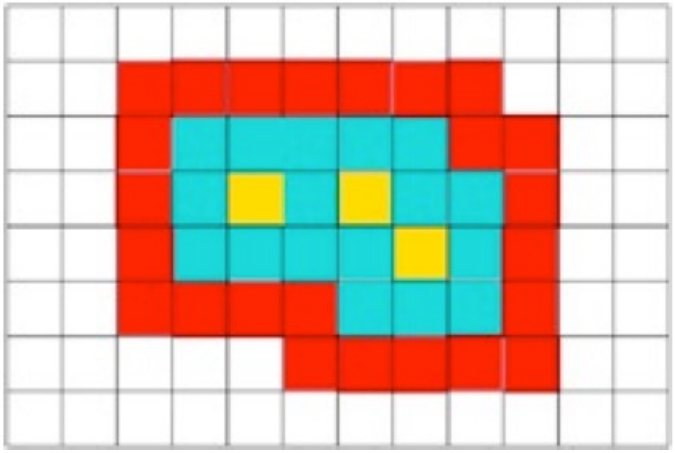
AMR Refinement Rules

- Most codes enforce mesh consistency and grading of patches
  - All neighbors have to be at the same level or one level below or above



- Specific Criteria
  - Jeans Criteria
  - Gradients
  - Quasi-Lagranian
  - Geometrical (zoom)

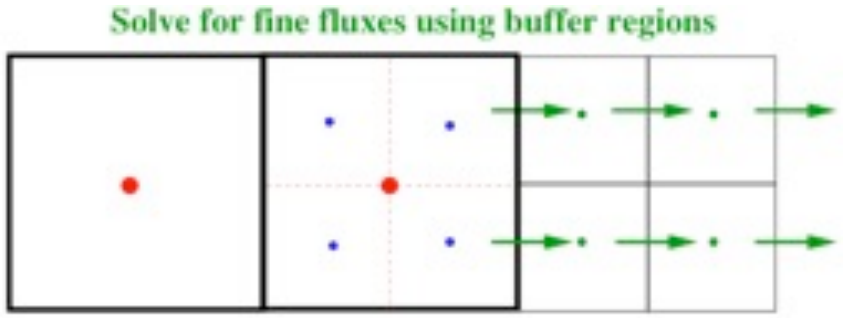
- Smoothing is often done
  - Patch has to be convex
  - Expand patch sizes with *nexpand* layers (ex. 2 layers)



[from lectures by Romain Teyssier]

Interpolation and Flux consistency

- *Prolongation:* Interpolation to finer meshes
  - Creation of new cells
  - On-the-fly boundary cells
- *Restriction:* Averaging to coarser levels
  - Destruction / derefinement
  - Filling of the threaded tree
- Flux correction at boundaries
- Different interpolators
  - Conservative / internal energy
  - Apply different slope limiters
  - ***When changing the resolution the method looses one order***



[from lectures by Romain Teyssier]

Time-adaptivity: Fine→coarse level time Evolution

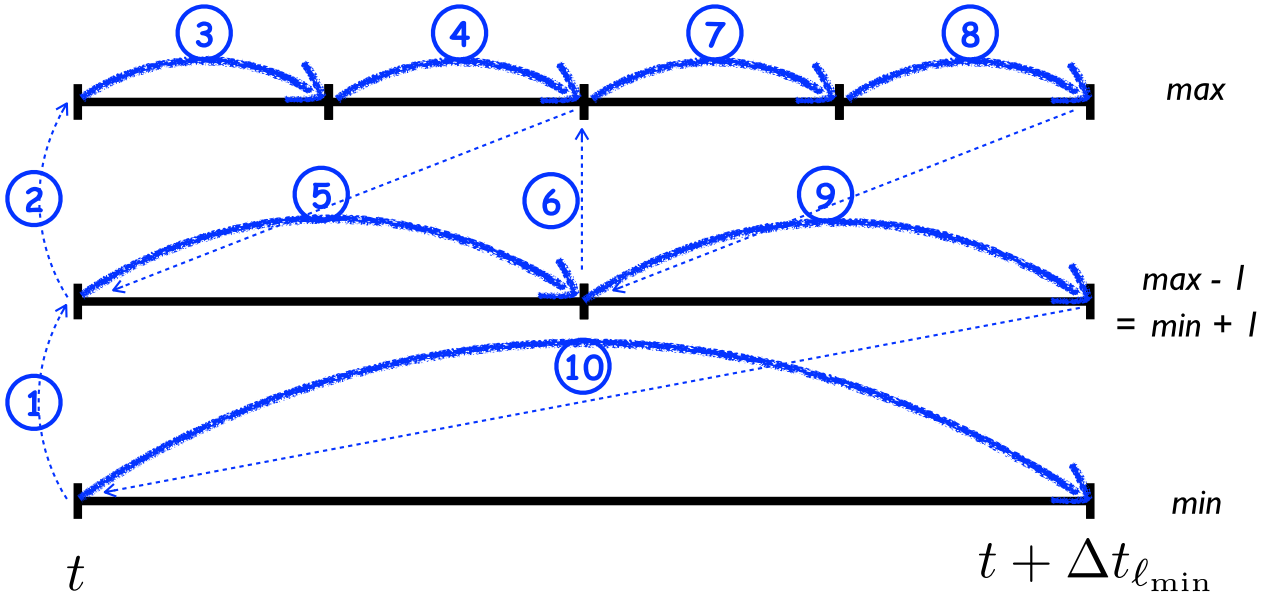
- Evolves in a W-cycle recursively from coarser to finer mesh and back

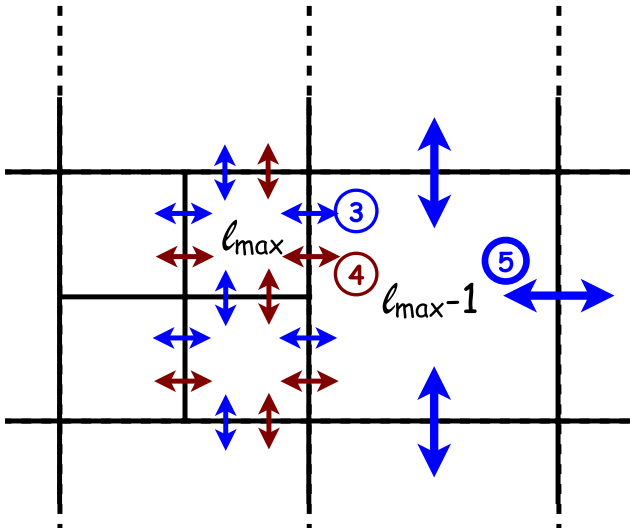
1. Prepare: Check on coarse level if we need to create new cells, prepare boundary conditions for finer levels. Then go to finer level.

2+6: Repeat step 1 and recursively progress to finer levels.

3-5,7-10. Evolve: Solve dynamics on finest level; update Courant, update flux for coarser cells via neighbor pointer; flag any cells on this level that have to be refined or destroyed.

Then recursively go to coarser level (diagonal lines) or repeat timestep





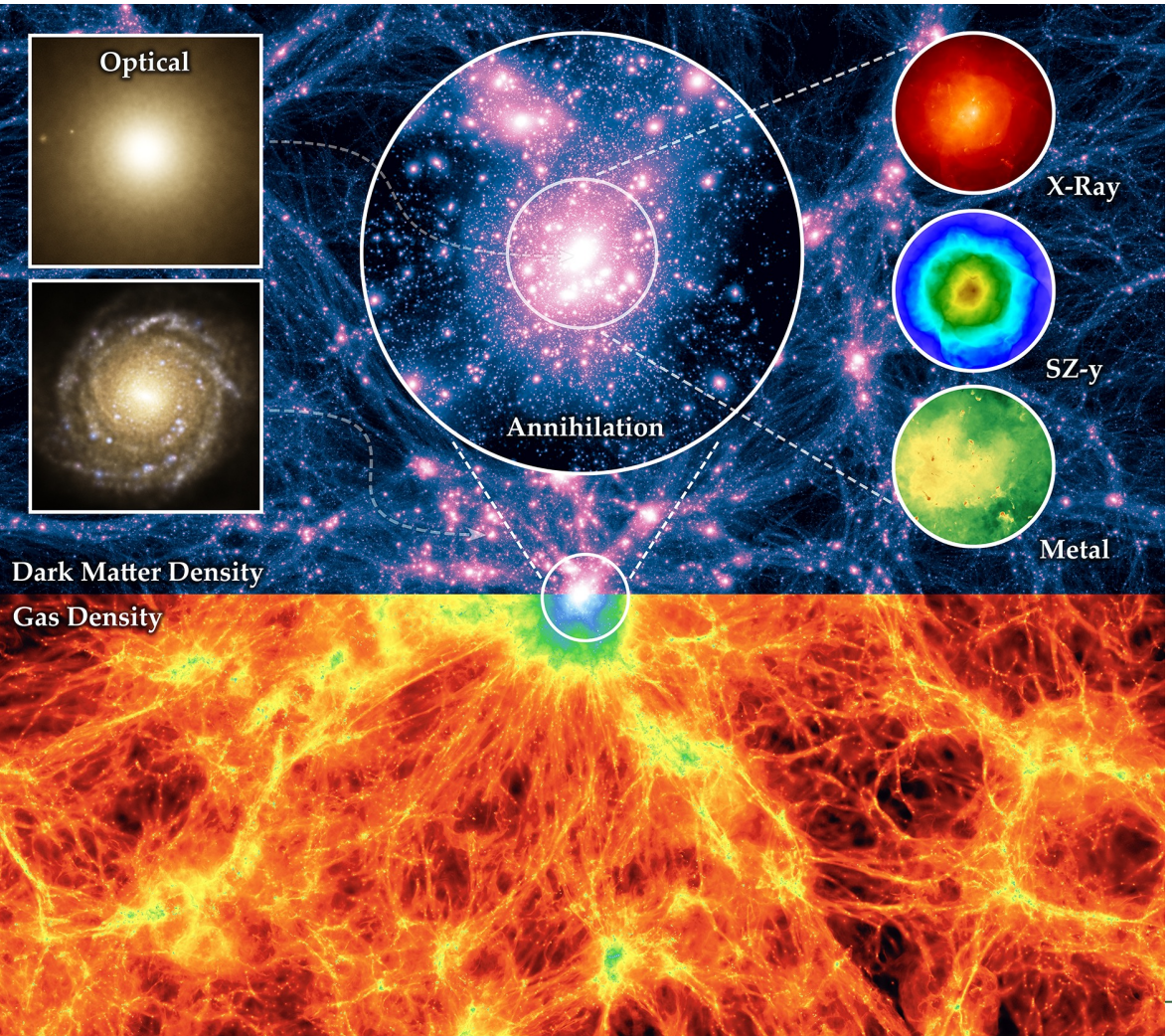
[Figs from Rosdahl 2013]

Astrophysical Applications

Cosmology and Star Formation

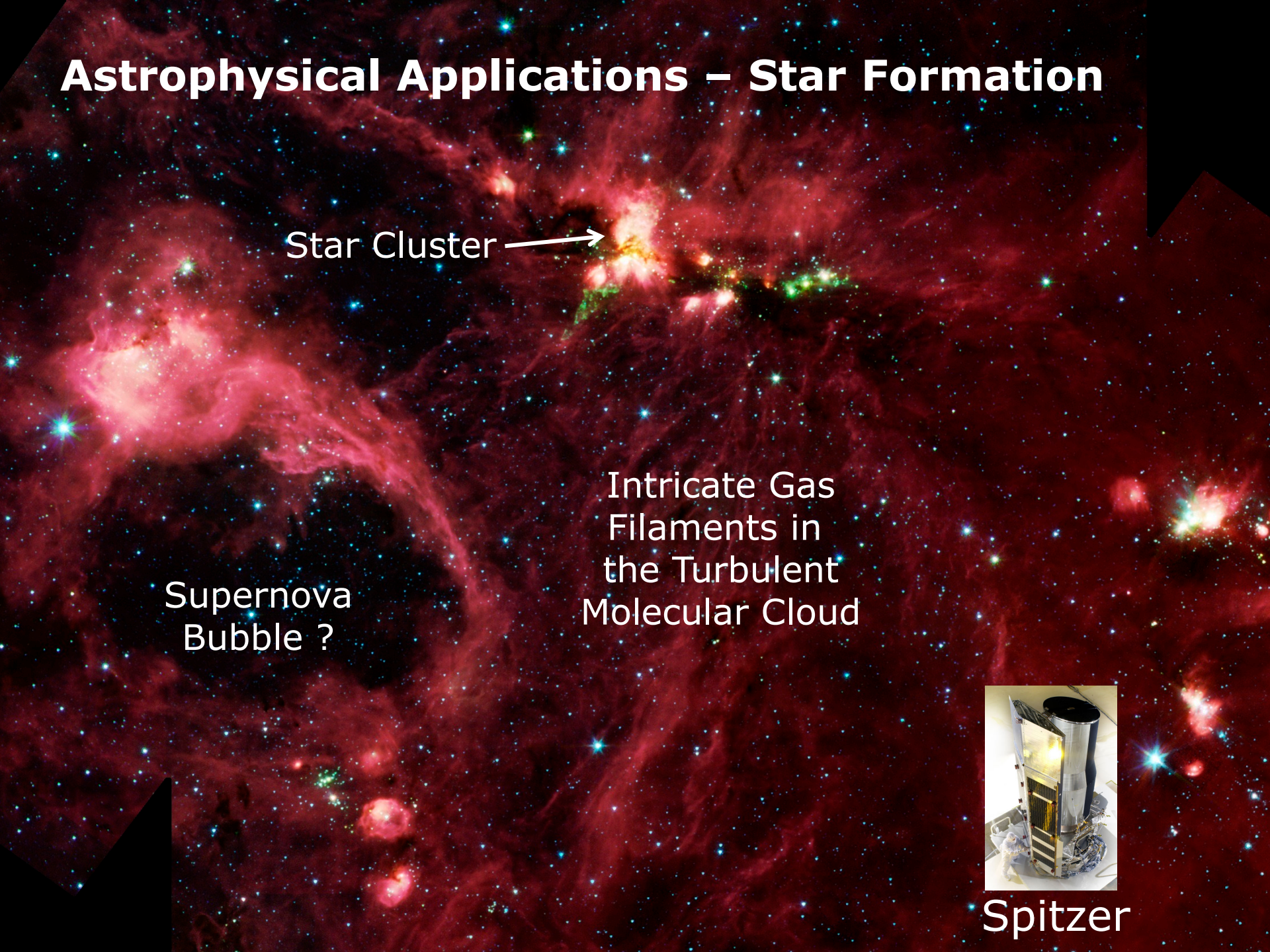
Astrophysical Applications - Cosmology

- ART, Enzo, FLASH, and RAMSES are in widespread use
- "Cosmology" is inherently a Multi-Scale problem



- Exemplified here by the illustris simulation, objects at the largest scales are coupled over at least four orders of magnitude
- Universe is statistically homogenous at >100 Mpc h^{-1}
- Galaxies have sizes of <100 kpc
- The large-scale filamentary web is needed to supply the boundary conditions for the formation and evolution of clusters and galaxies
- This was realized early, and "zoom-AMR" was used since the late '90

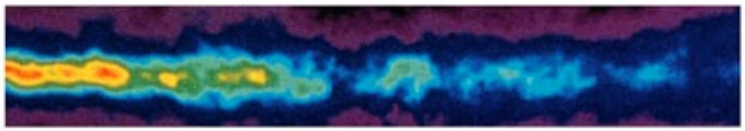
Astrophysical Applications – Star Formation

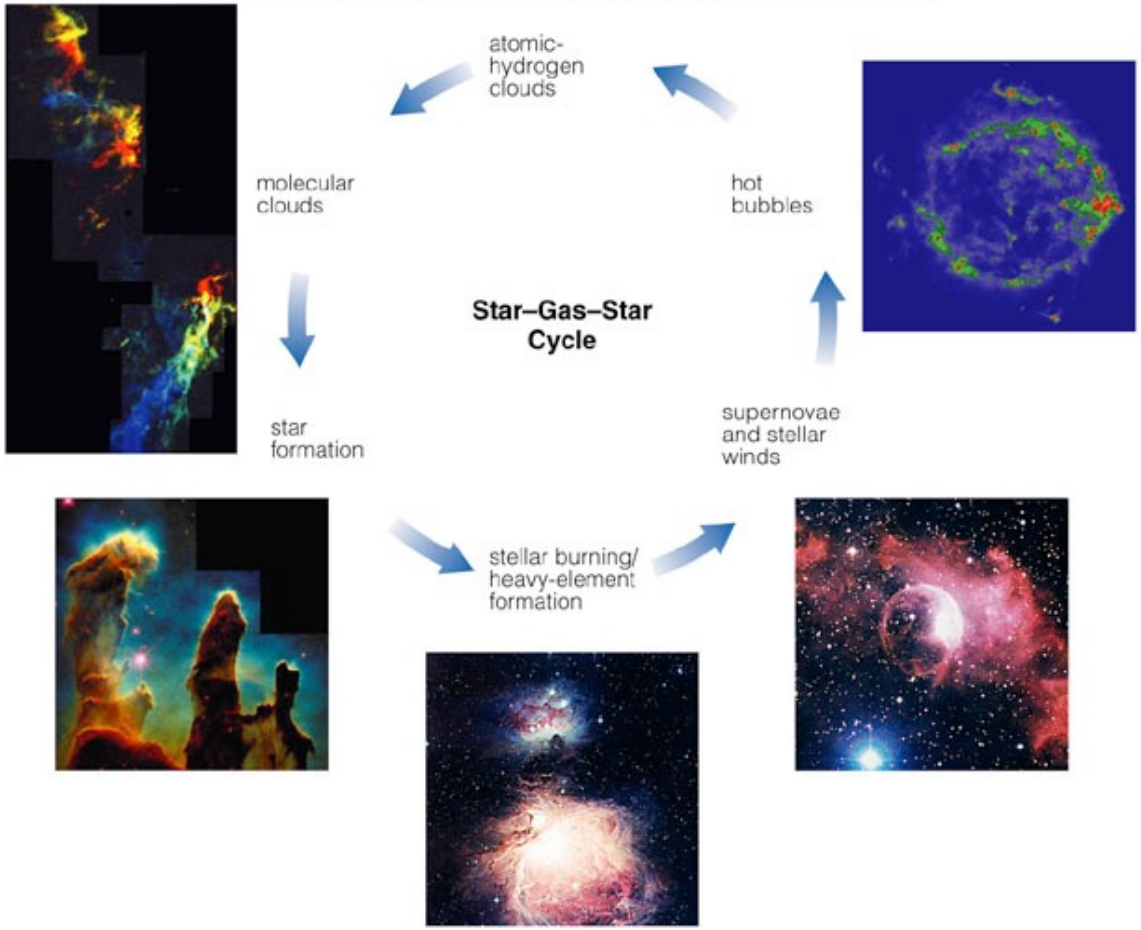


Spitzer

Astrophysical Applications – Star Formation

- Enzo, FLASH, Nirvana, Pluto, and RAMSES are in widespread use
- "Star Formation" is an even more extreme Multi-Scale problem
- 100 pc → 0.01 AU
- 10^9 dynamic range

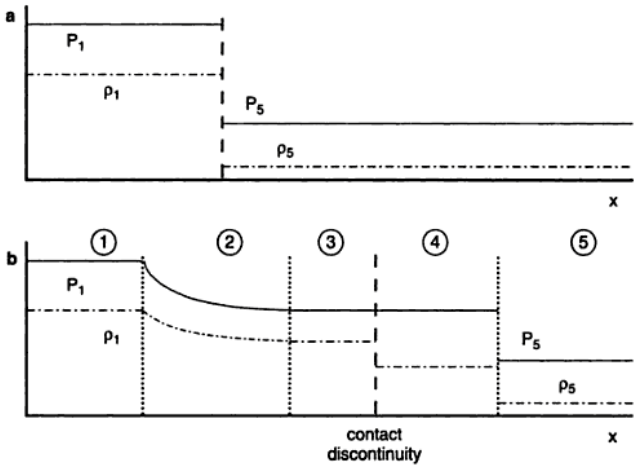


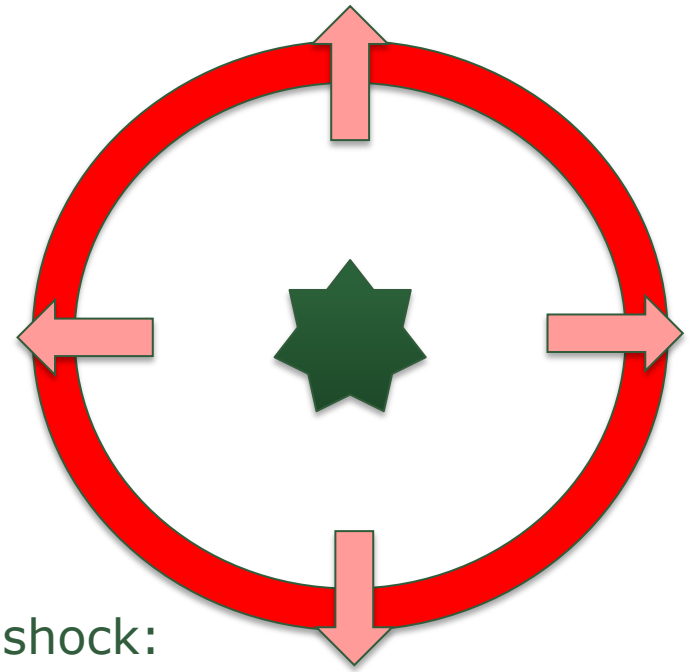


[Bennett et al: Stars, Galaxies, and Cosmology]

Exercise: Sedov Blast Wave in 2D

- A simplified Supernova explosion: Concentrate thermal energy and add mass loading in a small volume. An expansion wave is launched:





See also ch 3.8 in notes from Theoretical Astrophysics

- Example of a pressure driven shock:
  - If there is mass it starts by coasting (free-stream phase)
  - The pressure is large and a strong shock is launched (Sedov phase)
  - Afterwards we reach a snow-plow phase where the shock is pushing ISM
  - When shock has displaced as much inertia as inside, it starts to decay
  - In the Sedov phase it evolves according to a simple scaling relation