Numerical simulation techniques

Pekka Janhunen, FMI/Space
Graduate Summer School Lecture,
Kiljavanranta, 16 June 2005

Scope of this lecture

- Plasma simulation techniques:
  - MHD simulation (fluid simulation)
  - particle simulation
  - hybrid simulation
  - Vlasov simulation
- Application areas:
  - space plasma
  - fusion and laboratory plasma (not covered here)
  - fluid dynamics, aerodynamics (not covered here)
  - methods also applicable to simulation of gravity (galaxy, stellar clusters; not covered here)

Fluid simulation: Euler equations

Primitive form:

$$\begin{aligned} \frac{\partial \rho}{\partial t} &= -\nabla \cdot (\rho \mathbf{v}) \\ \rho \frac{d\mathbf{v}}{dt} &= -\nabla P + \nu \rho \nabla^2 \mathbf{v} \\ \frac{d}{dt}\left(P \rho^{-\gamma}\right) &= 0 \end{aligned}$$

Conservative form:

$$\begin{aligned} \frac{\partial \rho}{\partial t} &= -\nabla \cdot \mathbf{p} \\ \frac{\partial \mathbf{p}}{\partial t} &= -\nabla \cdot \left(\frac{\mathbf{pp}}{\rho} + P\mathrm{I} \right) \\ \frac{\partial U}{\partial t} &= -\nabla \cdot \left[(U + P) \frac{\mathbf{p}}{\rho} \right] \end{aligned}$$

Conservative form guarantees correct shock speed and preserves conserved quantities to roundoff error when discretised using finite-volume method (FVM)

$$P = (\gamma - 1)\left(U - \frac{\mathbf{p}^2}{2\rho} \right)$$

MHD equations: primitive form

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho \mathbf{v})$$
$$\rho \frac{d\mathbf{v}}{dt} = \mathbf{j} \times \mathbf{B} - \nabla P$$
$$\frac{dP}{dt} = -\gamma P \nabla \cdot \mathbf{v}$$
$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{v} \times \mathbf{B})$$

where $\mathbf{j} \equiv (\nabla \times \mathbf{B})/\mu_0$ and $d/dt \equiv \partial/\partial t + \mathbf{v} \cdot \nabla$.

Primitive variable 8-tuple: (ρ,**v**,P,**B**)

γ=adiabatic index, usually γ=5/3; generally γ=(f+2)/f where f=number of degrees of freedom

The primitive form is the simplest to write down and remember

MHD equations: conservative form

$$
\begin{aligned}
\frac{\partial \rho}{\partial t} &= -\nabla \cdot \mathbf{p} \\
\frac{\partial \mathbf{p}}{\partial t} &= -\nabla \cdot \left[\frac{\mathbf{p}\mathbf{p}}{\rho} + \left(P + \frac{\mathbf{B}^2}{2\mu_0} \right) \mathrm{I} - \frac{1}{\mu_0} \mathbf{B}\mathbf{B} \right] \\
\frac{\partial U}{\partial t} &= -\nabla \cdot \left[\left(U + P - \frac{\mathbf{B}^2}{2\mu_0} \right) \frac{\mathbf{p}}{\rho} + \frac{1}{\mu_0 \rho} \mathbf{B} \times (\mathbf{p} \times \mathbf{B}) \right] \\
\frac{\partial \mathbf{B}}{\partial t} &= \nabla \times \left(\frac{\mathbf{p}}{\rho} \times \mathbf{B} \right) \qquad (1)
\end{aligned}
$$

- Variable 8-tuple is now $(\rho, \mathbf{p}=\rho\mathbf{v}, U, \mathbf{B})$ where $U = P/(\gamma-1)+p^2/(2\rho)+B^2/(2\mu_0)$

MHD: Semiconservative form

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

$$\frac{\partial \mathbf{p}}{\partial t} = -\nabla \cdot \left(\frac{\mathbf{pp}}{\rho} + P\mathrm{I} \right) + \mathbf{j} \times \mathbf{B}$$

$$\frac{\partial U}{\partial t} = -\nabla \cdot \left[(U + P) \frac{\mathbf{p}}{\rho} \right] + \mathbf{j} \cdot \mathbf{E}$$

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times \left(\frac{\mathbf{p}}{\rho} \times \mathbf{B} \right)$$

$$\mathbf{p} \equiv \rho \mathbf{v}$$

$$\mathbf{j} \equiv (1/\mu_0) \nabla \times \mathbf{B}$$

$$\mathbf{E} \equiv \mathbf{B} \times \mathbf{v}$$

$$U \equiv \frac{P}{\gamma - 1} + \frac{1}{2} \rho \mathbf{v}^2$$

Non-conservative electromagnetic force and energy flux added to conservative fluid (Euler) equations

Does *not* guarantee correct shock speed or exact energy and momentum conservation

Non-idealities in MHD

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho \mathbf{v}) + D_\rho \nabla^2 \rho$$

$$\rho \frac{d\mathbf{v}}{dt} = \mathbf{j} \times \mathbf{B} - \nabla P + \nu \nabla^2 (\rho \mathbf{v})$$

$$\frac{dP}{dt} = -\gamma P \nabla \cdot \mathbf{v} + D_P (\gamma - 1) \nabla^2 U$$

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{v} \times \mathbf{B}) - \nabla \times \left(\frac{\mathbf{j}}{en} \times \mathbf{B} \right) + \frac{\eta}{\mu_0} \nabla^2 \mathbf{B}$$

Hall-term

- note that $\mathbf{j} = en(\mathbf{v}_i - \mathbf{v}_e)$; in MHD, $v = \mathbf{v}_i$, thus **B** is frozen into electron flow when Hall-term included
- if diffusion parameters are not constant, they must appear inside gradient

FVM Godunov-type methods

- FVM = Finite-volume method:
  Store ***volume averages*** of quantities over each computational cell (instead of pointwise values)
- Equations of the form ***du/dt = -div(something)*** are naturally discretised with FVM using cell interface fluxes. This yields automatically to a ***conservative discretisation***.
- Godunov-type method:
  - Propagate staircase FVM representation exactly
  - Compute new cell averages from propagated state
  - Godunov-type method can be written down with interfaces fluxes only (no cell averaging needed)

Riemann solvers

- “Riemann solver” = Solution of initial-value problem with stepwise initial data
- In 1-D Euler equations, even exact Riemann solver is known
- In 1-D MHD, practically usable exact solver does not exist
- In 2-D and 3-D, no exact solvers, but alternating direction discretisation works rather well in practice
- An exception: div(B)=0 constraint is problematic in MHD in 2-D and 3-D since it is essentially multidimensional

1-D approximate Riemann solvers

- Harten-Lax-vanLeer (HLL) solver:
  Uses one intermediate state which propagates away from the discontinuity and which is defined as the spatial average of the *exact* Riemann problem solution
- Easy to construct for any equation system, only interface fluxes needed
- If continuum problem preserves positivity, so does the HLL-discretisation (!)
- Drawback: Has high diffusion
- Recent developments: Generalise HLL to have more intermediate states

1-D approx. Riemann solvers, cont'd

- ***Roe's approximate Riemann solver:***
  Linearise equations around the interface and solve the linearised Riemann problem exactly
  - Need to specify the averaging scheme used (one possibility: "√ρ-averaging = Roe-averaging")
  - Needs to be able to solve the linearised eigensystem (eigenvalues, left and right eigenvectors analytically); this is possible but rather cumbersome for MHD
  - Can yield negative pressures if the interface jump is large

MHD monopole removal methods

- So-called ***elliptic cleaning:***
  write $\mathbf{B}' = \mathbf{B} + \mathrm{grad}(\Delta\Phi)$ and require $\mathrm{div}(\mathbf{B}')=0$, which gives Poisson equation for scalar field $\Delta\Phi$, with $\mathrm{div}(\mathbf{B})$ as source term.
  May produce negative pressures since P depends on **B**. One way to fix is to break energy conservation locally in these (hopefully rare) cases.
- Yee-mesh method: store **B** as interface surface averages (only normal component stored on each surface). Breaks conservation, method no longer fully FVM.

Origin of the monopole problem

- If problem is solved alternating in X,Y,Z, in each 1-D subproblem the 1-D div(**B**) is just dB_x/dx etc. which can be zero only if **B**-field is trivial. Thus, the 1-D subproblems always "see" nonzero div(**B**), even if the true 3-D div(**B**) vanishes in some discrete sense.
- Continuum MHD equations have *no solution* if the initial state has nonzero div(**B**)
- Adding monopoles to MHD theory at the fundamental level is one possibility, and has given rise to some new numerical methods

Domain of validity of MHD

- Euler equations hold for scales much larger than collision mean free path (counterexample: re-entry vehicle physics)
- Likewise for MHD, but in addition, in the perpendicular direction it suffices scale to be much larger than ion Larmor radius
- Non-MHD scale processes, if nonlinear, may have global consequences that make MHD invalid *at all scales* in principle. (This holds also in fluid dynamics.)
- Mass, momentum and energy conservation described by MHD are exactly valid. To the extent these fix the solution, MHD is good.

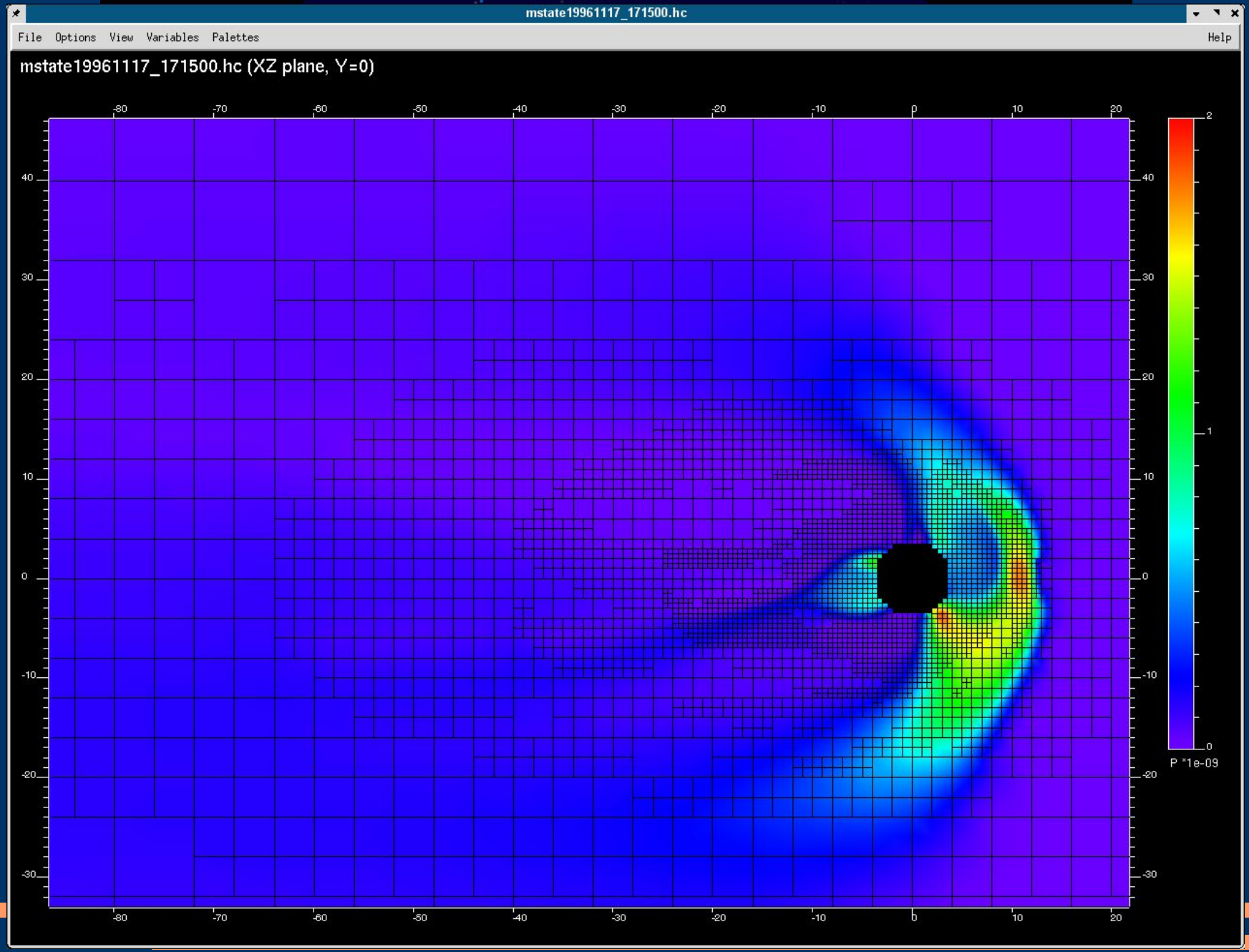
MHD implementation issues

- Serious MHD code typically must use some form of *grid adaptation*, which inevitably makes the implementation rather complex.
- To handle the complexity, careful planning and good language (***C***++!) are needed. Heavy hand-optimisation of time-critical code, although desirable, is not always realistic because of the complexity.
- MHD is never very simple to program, not even on uniform grid. Even in fluid dynamics there has been room for commercial software industry (Fluent, Fidap, Elmer...), and FD is just a subset of MHD.

Adaptive mesh techniques

- Grid types
  - uniform grid
  - stretched or deformed uniform grid
  - cell-by-cell hierarchically refined cubic grid
  - block-by-block refined, locally uniform grid
  - fully general grid containing arbitrary-shaped cells
- Grid can be fixed in time (*adapted* grid) or change dynamically during the run (*adaptive* grid). In the latter case, the grid refinement and coarsening may be based on the solution alone (*fully automatic*) or include a user-specified component (*semiautomatic*).

Hierarchically refined cubic grid



Block-refined grids

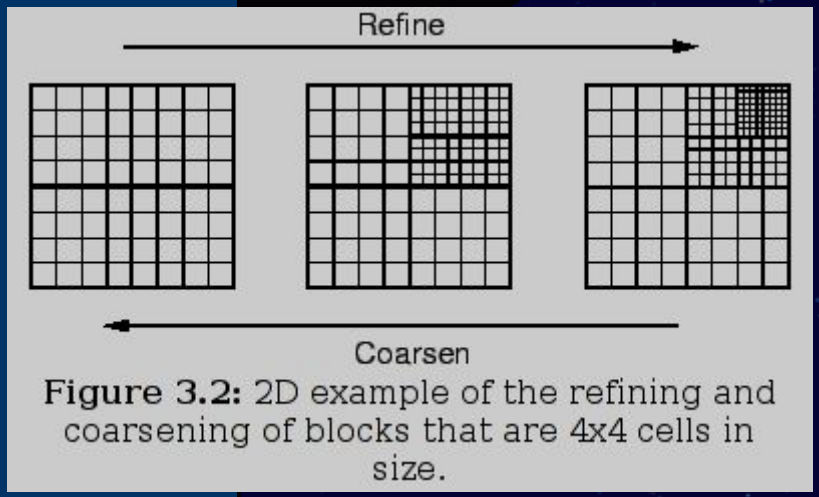


Figure 3.2: 2D example of the refining and coarsening of blocks that are 4x4 cells in size.

(csem.engin.umich.edu/docs/)

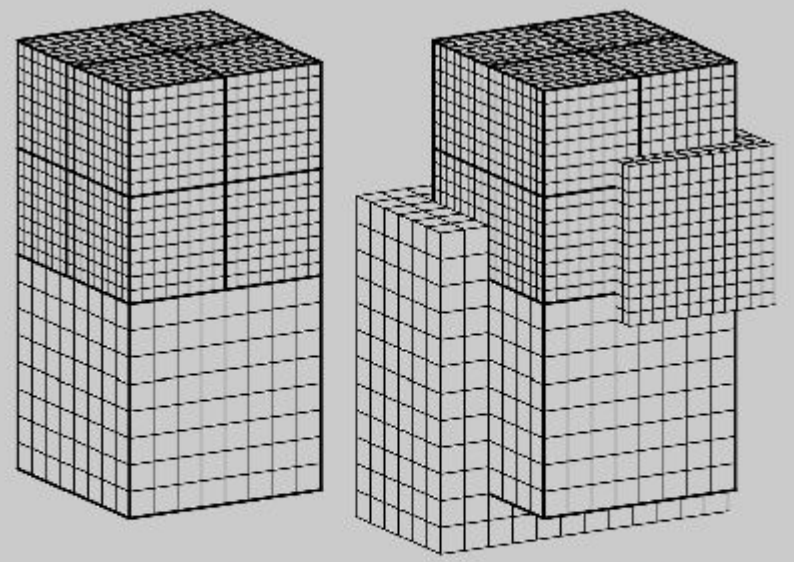
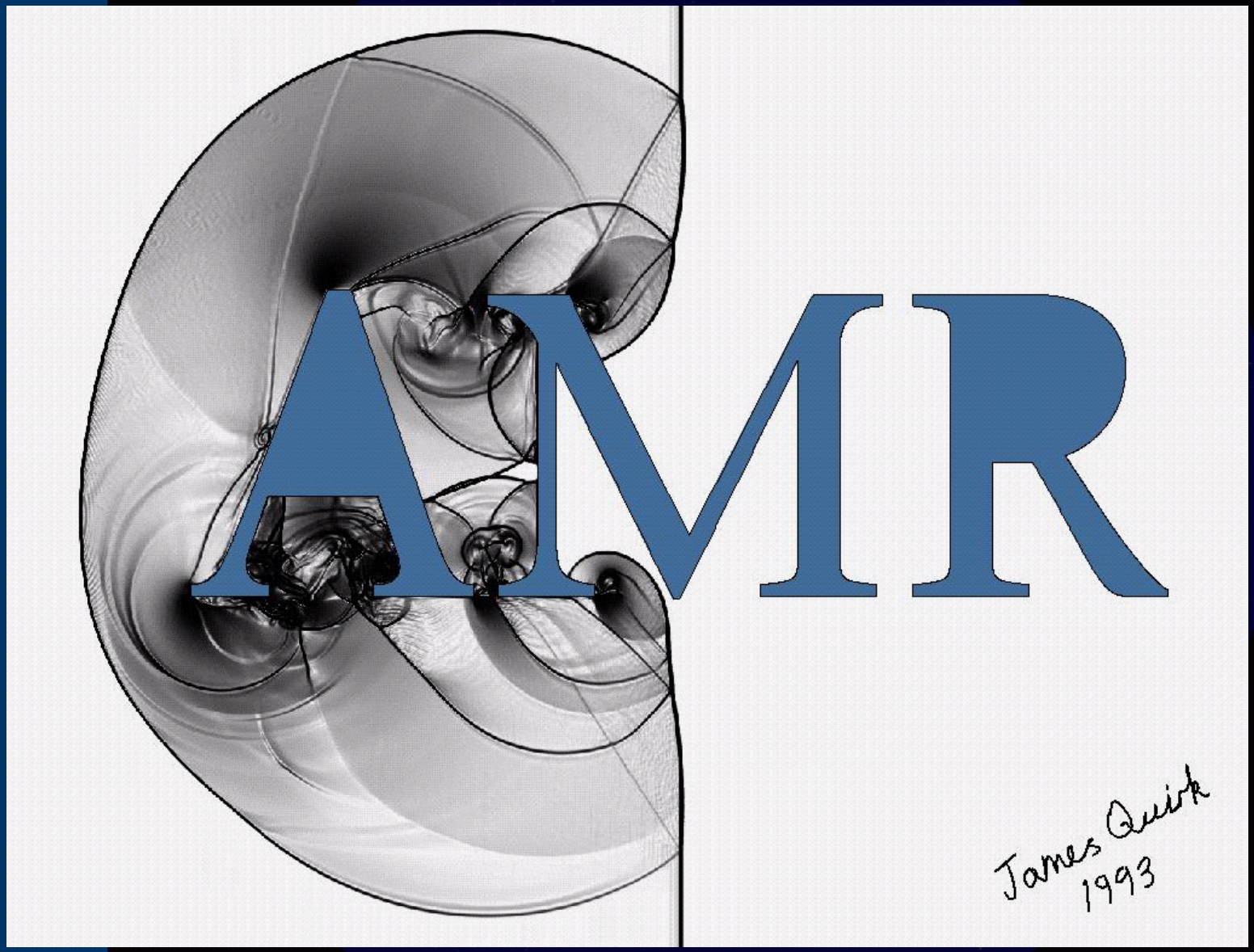


Figure 3.1: (left) Self-similar blocks used in parallel block-based AMR scheme. (right) Self-similar blocks illustrating the double layer of ghost cells for both coarse and fine blocks.

- Block-refined grids are locally uniform and thus fast to traverse; on the other hand there is some overhead because some cells are refined unnecessarily

AMR = Adaptive Mesh Refinement



(www.icase.edu/docs/hilites/jjq/images/amr.gif)

Time discretisation

- Time discretisation can be *explicit* or *implicit*.
- In *explicit* scheme, timestep at a cell must be shorter than fastest wave travel time across cell (the Courant condition, or CFL condition).
- The maximum usable timestep may vary widely across the grid, because both Alfven speed and cell size differ. To save computing time, one may use *temporal subcycling* to take short steps only where needed.
- *Implicit MHD* has been studied and to some extent used by the Michigan group.

Parallelisation

- In large codes, parallelisation is usually needed nowadays.
- Most widely used parallelisation strategy is domain decomposition, i.e. each processor owns a specific domain and handles all computation in that domain. Domain boundaries may move or remain fixed, depending on the application.
- Domain-decomposed MHD on uniform or stretched grids has been in use for some years.
- How to combine domain decomposition with adaptive gridding is an active research topic.
- Usually one uses the MPI library (Message Passing Interface).

Particle simulation

- Idea: Plasma particles move in their own self-consistent fields
- Components: Particle mover, Field equation solver, Charge & current density accumulation
- Types:
  - electrostatic model
  - Darwin (magnetoionic) model (zero divergence-free part of displacement current in Ampere's law)
  - fully electromagnetic model
- Main applicability: Small scales
- Main issue: Particle noise ~ $1/\sqrt{N}$

Explicit and implicit time integration

- Explicit time integration: Timestep must be (clearly) smaller than:
  - inverse electron plasma frequency (electrostatic)
  - electron travel time across grid cell (electrostatic)
  - light travel time across grid cell (electromagnetic)
- Implicit time integration does not have this restriction, but it must solve more complicated elliptic equation than Poisson
- Despite much research (~30 years) and promising results, implicit schemes have not replaced explicit schemes in practice (Why? Good question...)

Components of particle simulation

- ***Particle mover and field interpolation:***
  - interpolate E and B fields at particle position from the gridded fields
  - use time-symmetric Boris-Buneman Lorentz force integrator (while alternatives exist, this is virtually the standard)
- ***Field solver:***
  - Poisson solver in electrostatic case
  - in electromagnetic case, also need time integration of Ampere and Faraday laws (often in staggered (Yee-) grid)
- ***Charge and current density accumulation:***
  - ***must*** use the same interpolation scheme as in particle mover, otherwise self-force instability!!

Implementation issues

- Particle simulation algorithms are so simple that heavy hand-optimisation *is* typically done (in contrast to MHD simulations):
  - integer-valued coordinates (~2* speedup)
  - cache-friendliness, single pass through particle tables which combines both mover and accumulation phases
  - assembly language, multimedia instructions
  - often memory size is bottleneck, store particle data with 32 bits or even less
  - if parallelisation needed, must throw particles back and forth between processors (not too difficult, provided that the grid part parallelises smoothly)

Sources of information

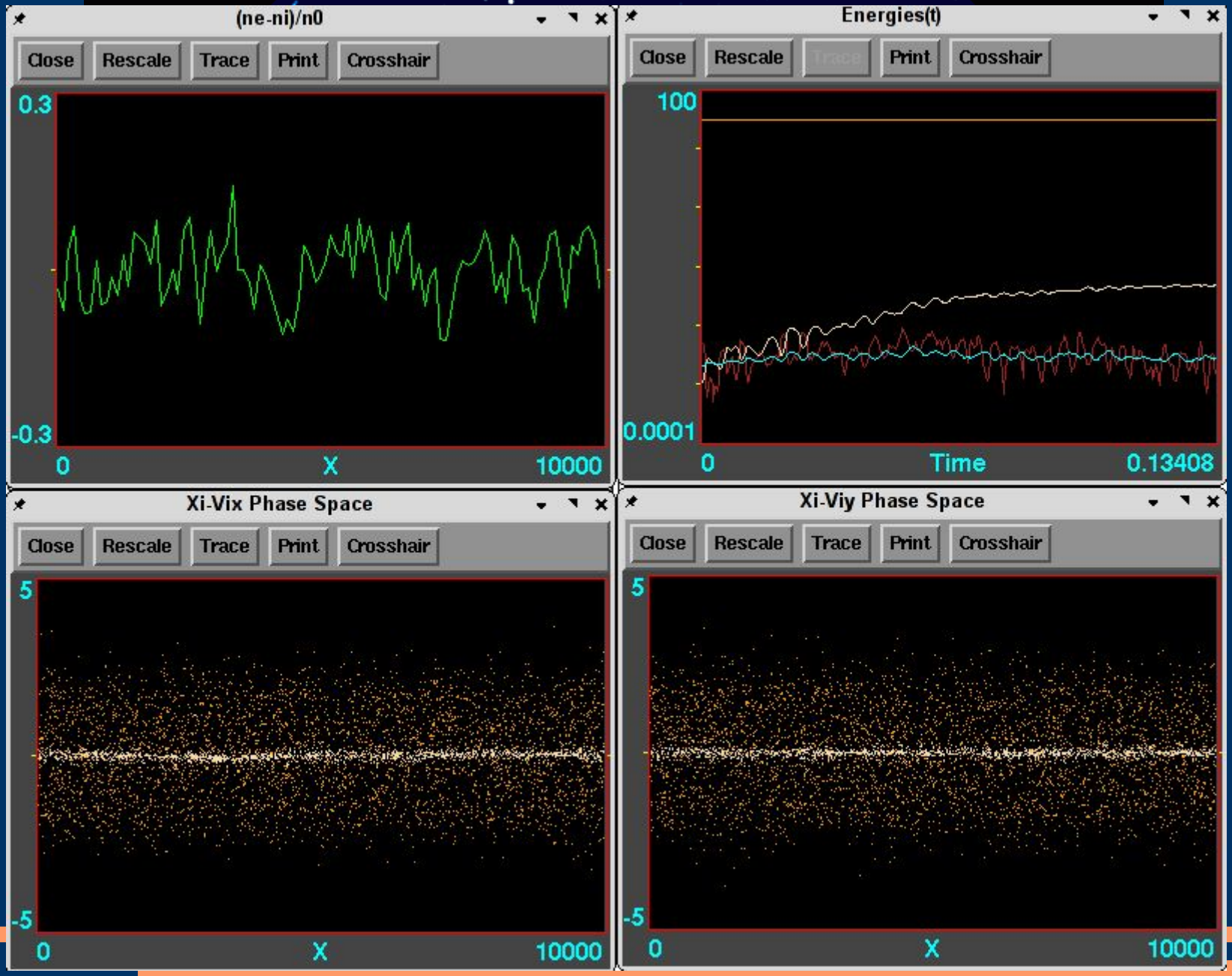
- Two books above all:
  - ***Birdsall and Langdon,* Plasma physics via computer simulation** (comprehensive, some errors, some topics difficult but not necessarily needed in applications)
  - ***Hockney and Eastwood,* Computer simulation using particles** (different notation for interpolation functions, includes also applications to gravity and semiconductors)
- In contrast to MHD/fluid simulations, the recipe here is basically: ***Read these and go to work!***
- Basic theory became ready in the 1970's and early 1980's, no big new developments after that
- But it's essential to study these books carefully!

Domain of validity

- (Explicit) electrostatic model:
  - Spatial grid size must be of the order of electron Debye length or somewhat smaller
  - Timestep must be smaller than ~0.2*inverse electron plasma angular frequency and electron traveltime across grid cell, whichever is smaller
- (Explicit) electromagnetic model:
  - in addition, timestep must be smaller than light traveltime across grid cell (!)
- Thus, one is limited to rather small scales
  - For example, Debye length is ~100 m in auroral magnetosphere, region contains ~10^5 Debye lengths in field-aligned direction (and each cell must contain ~100 particles to achieve ~10% noise level)

Public domain software

Code ES1, XES1 and Xgrafix library (ptsg.eecs.berkeley.edu)



Hybrid simulation

- Idea: Fluid electrons, particle ions
- Benefits:
  - Longer timesteps can be taken because electron plasma frequency and gyromotion need not be followed
  - Grid spacing can be much larger than Debye length (its natural scale is so-called ion inertial length c/ω_{pi})
- Drawbacks:
  - Electron kinetic effects not included
  - Conservative formulation as in MHD not known
- All in all:
  - Hybrid simulation has the potential of carrying over many of the benefits of particle simulation to larger scale size problems

Hybrid simulation applications

- Small, weakly magnetised magnetospheres
  - All terrestrial planets except Earth:Mercury, Venus, Mars, Titan
  - Medium-sized rocky bodies: Moon, asteroids
  - Comets (although object very small, its plasma environment is much larger)
- Currently not well applicable to more strongly magnetised objects such as the Earth (reasons are partly unclear)
- Local simulations in magnetotail
- Local simulations at bow shock/magnetosheath

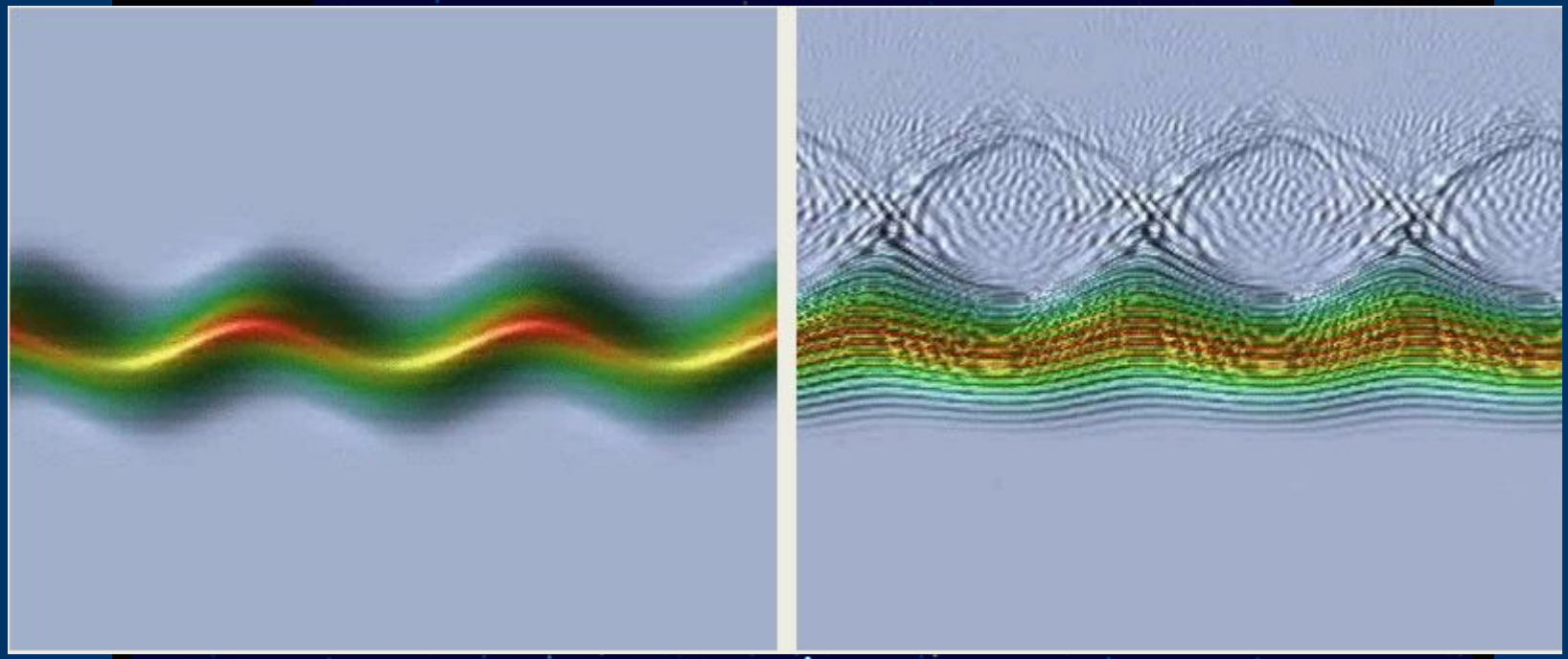
Vlasov simulation

- Idea: Like kinetic simulation, but instead of using particles, simulate distribution function by a phase space grid
- Grid size and timestep issues rather similar to corresponding particle simulation
- Benefits compared to particle simulation:
  - No particle noise
- Drawbacks:
  - Phase space is 6-D (!)
- Thus far not used too much, but importance likely to grow in the future (!)

Vlasov simulation techniques

- ***Semi-Lagrangian method***
  - Use Liouville theorem: f constant along phase space flowlines. Integrate backward along flowlines and interpolate from old function.
  - Non-conservative
- The Rasio method (always go back to initial condition)
- Velocity space Fourier-transform (or other transform) method (Bengt Eliasson; only for uniform grids)
- Conservative semi-Lagrangian method of Iske and Käser (only in 1+1 D)
- ***FVM conservative methods***
- Arakawa-type finite difference method (only 1+1 D)
- "Phase space cloud" method of Alard and Colombi

Velocity-Fourier Vlasov example



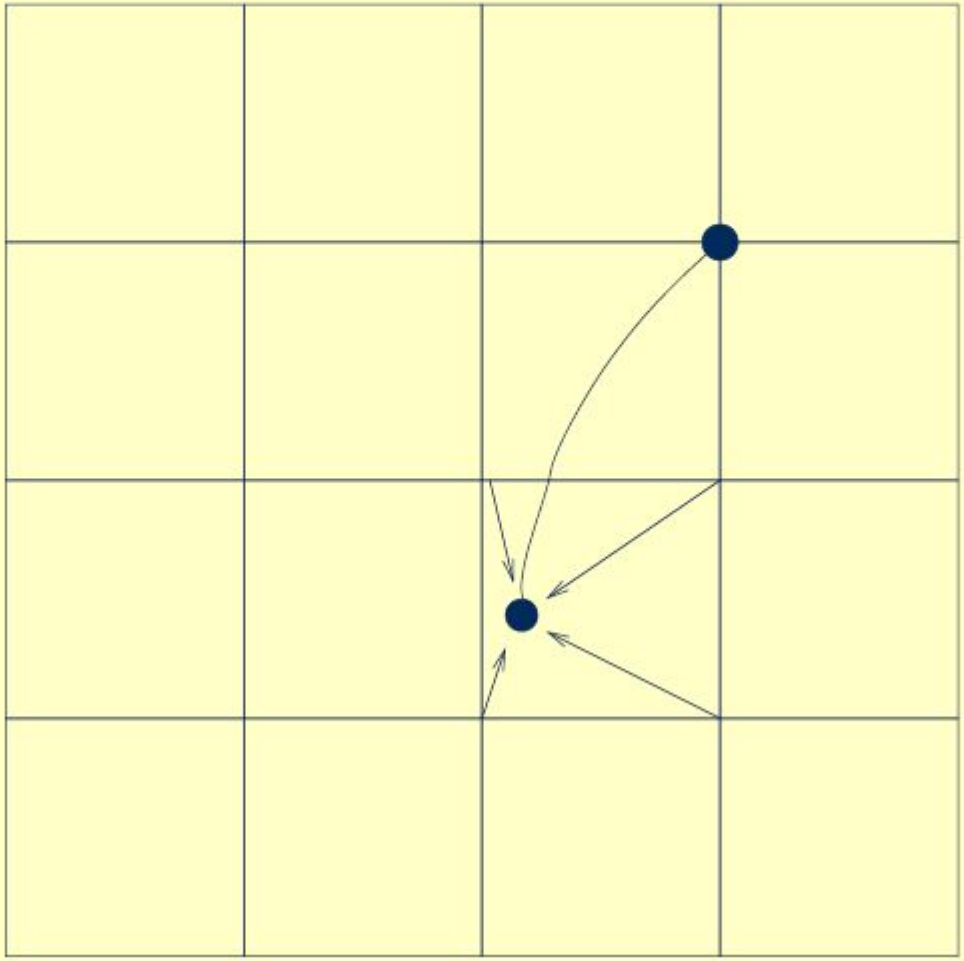
(Bengt Eliasson)

Semi-Lagrangian principle

www.isis.rl.ac.uk/AcceleratorTheory/workshop/talks/Sonnendrucker.pdf

The backward semi-Lagrangian Method

- f conserved along characteristics
- Find the origin of the characteristics ending at the grid points
- Interpolate old value at origin of characteristics from known grid values
  $\rightarrow$ High order interpolation needed




L U N C H

Grid types & approximation schemes

- In all simulation types (even in particle simulation), approximating a continuous function (*field*) on a finite grid is needed.
- Grid types:
  - uniform (Cartesian) grid
  - stretched (or mapped) uniform grid
  - hierarchically refined Cartesian (HC) grid
  - fully general grid
  - all have also *triangle/tetrad/simplex* variants
- Approximation schemes:
  - finite difference (FDM; tabulation of points and interpolation)
  - finite volume (FVM; store cell volume averages)
  - finite element (FEM; most general; represent function in each cell by algebraic formula)

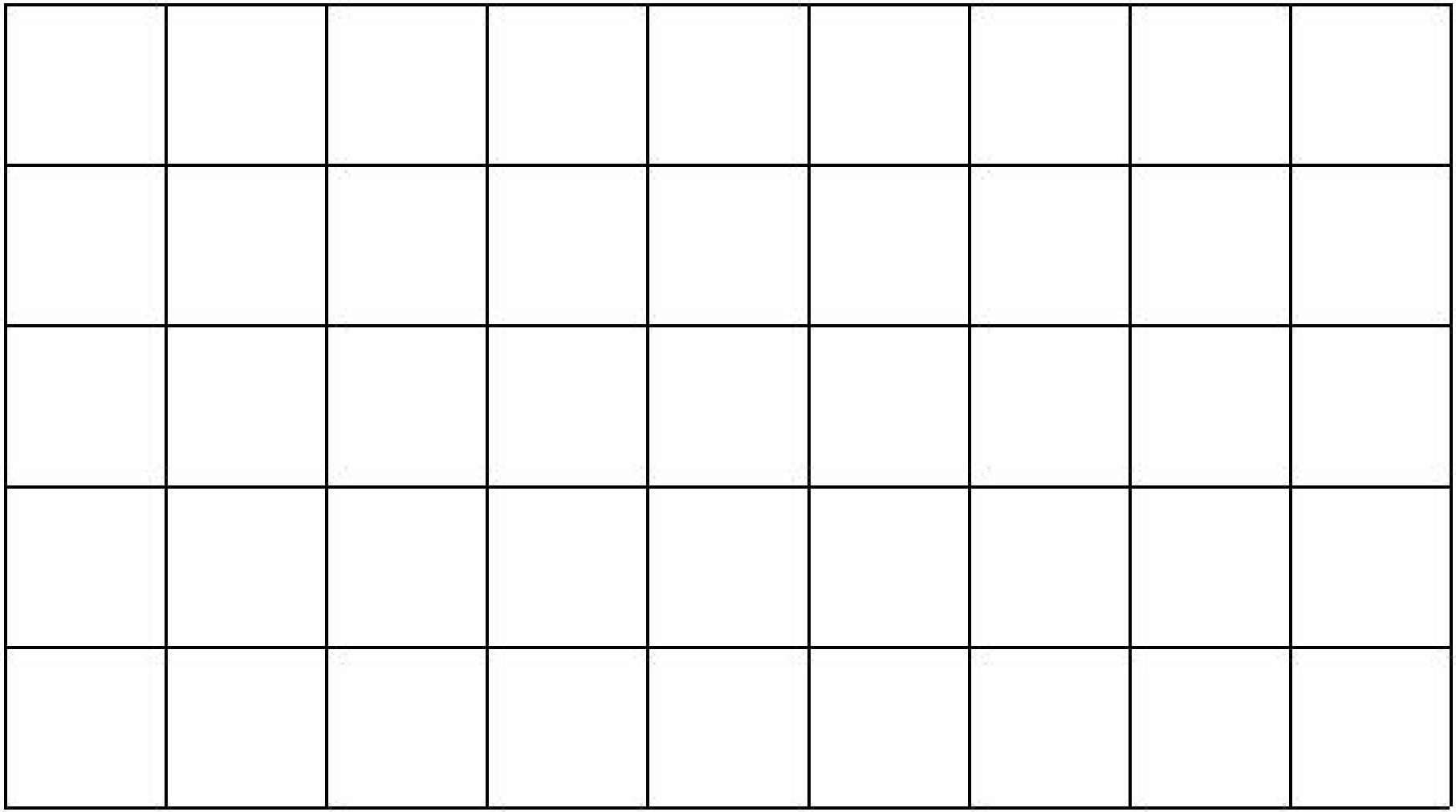
Approximation schemes

- FDM/FVM/FEM sometimes yield the *same* numerical method
- FVM is natural when conservation laws are involved
- FEM is like FDM but can handle more flexibly complex geometries and boundary shapes
- On the other hand FEM is also like generalised FVM
- Classification FDM/FVM/FEM is not fundamental or strict
- FEM is not the “best” and FDM not the “worst”
- Fourier (or other transform) representation can be seen as extreme case of FEM. (Fourier methods are potentially very good, but typically tricky to get working robustly in fluid dynamics and MHD)

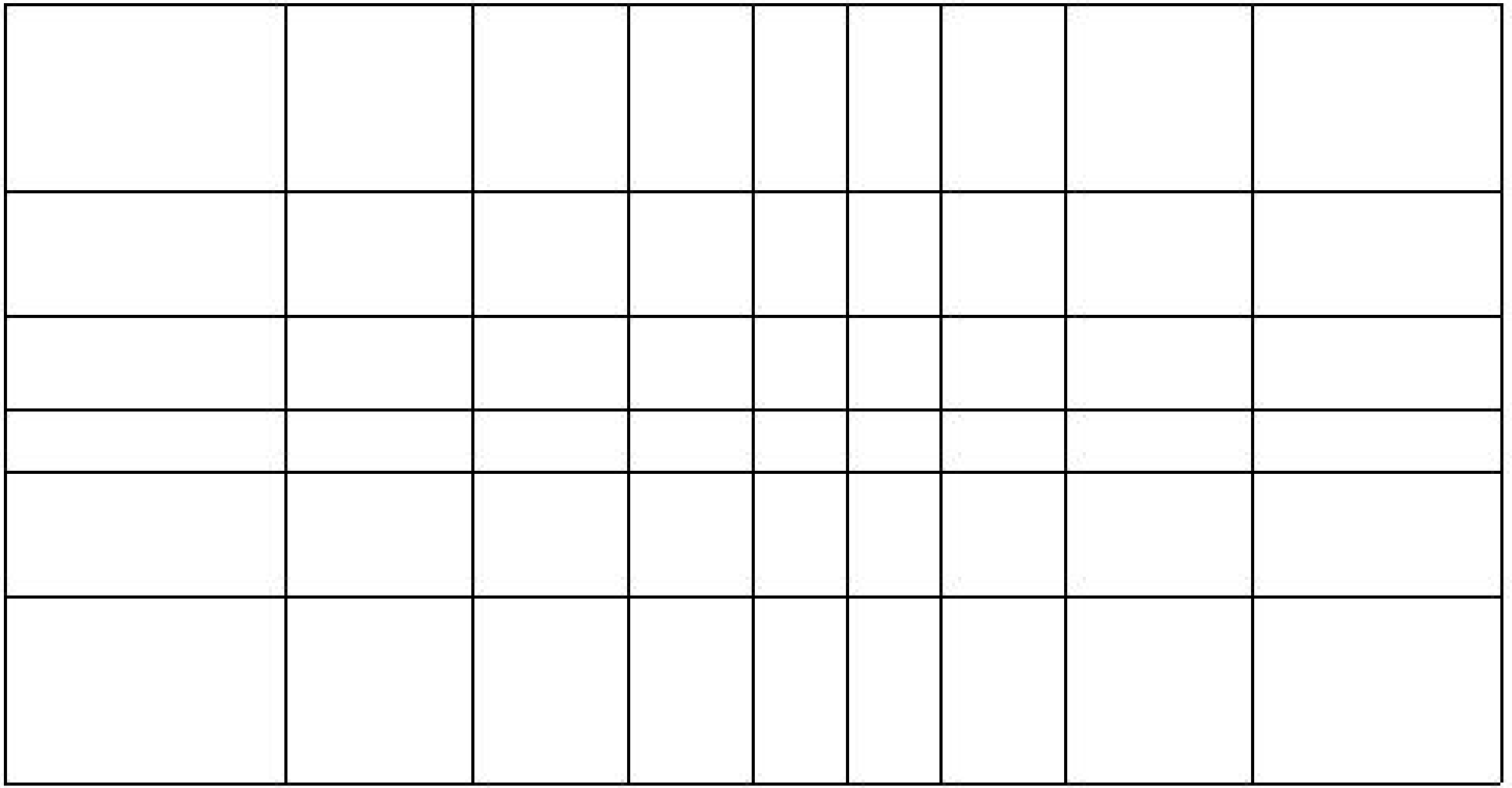
Grid types

- Rectangle/cubic/hexahedral grids typically work better numerically than triangle/tetrad/simplex grids. The reasons are not too well known.
- Computer time and memory saving due to use of adaptive grid types can be huge, and increases with problem size.
- Having said that, it is also true that the approximation quality never improves by removing grid points.
- *In given CPU-time and memory*, however, use of adaptive grid is likely to be the *optimal solution* (*sometimes by quite a large margin*).
- If particles are involved and if grid is adaptive, *particle splitting and joining* are typically needed as well. Join as 3->2 (energy+momentum conservation!)

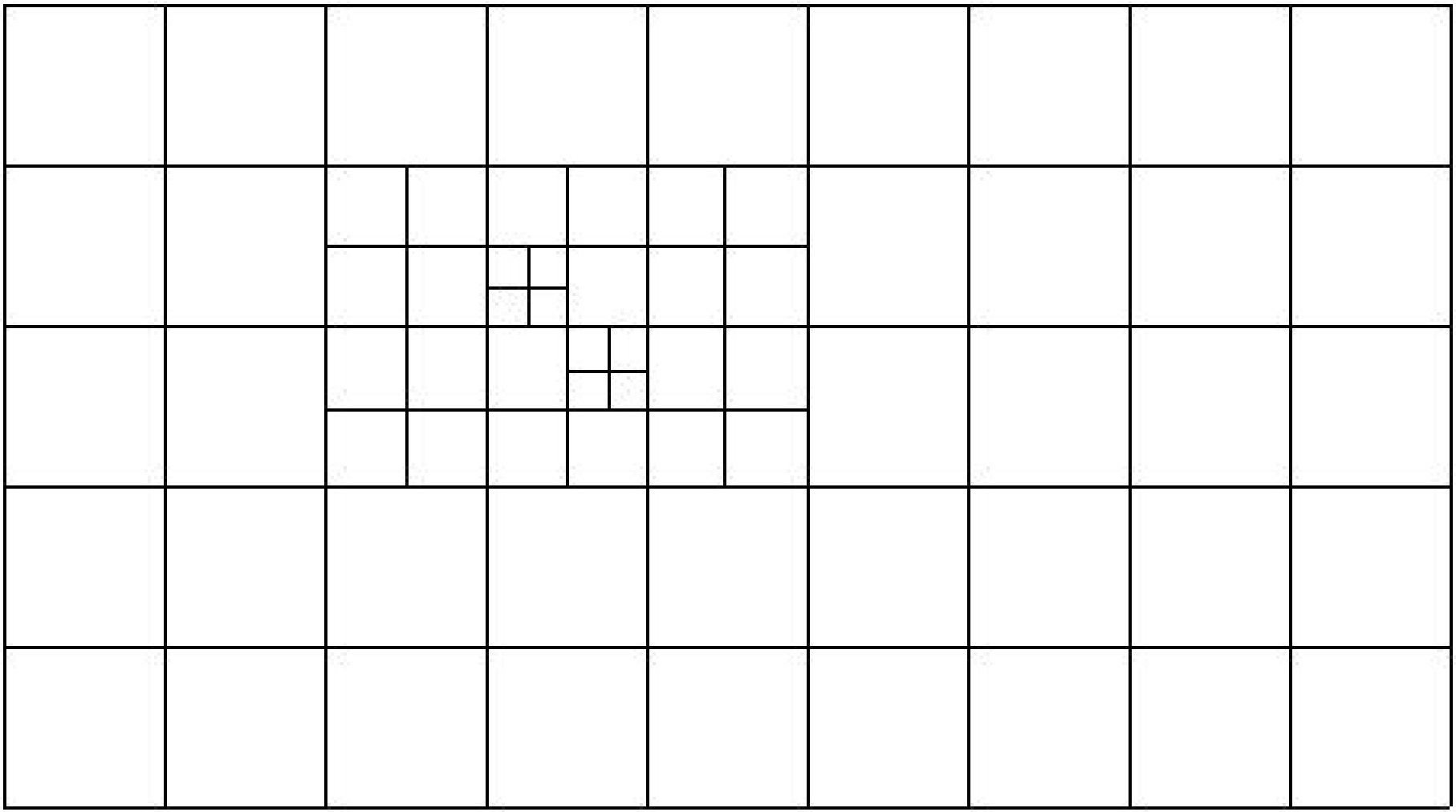
Uniform grid



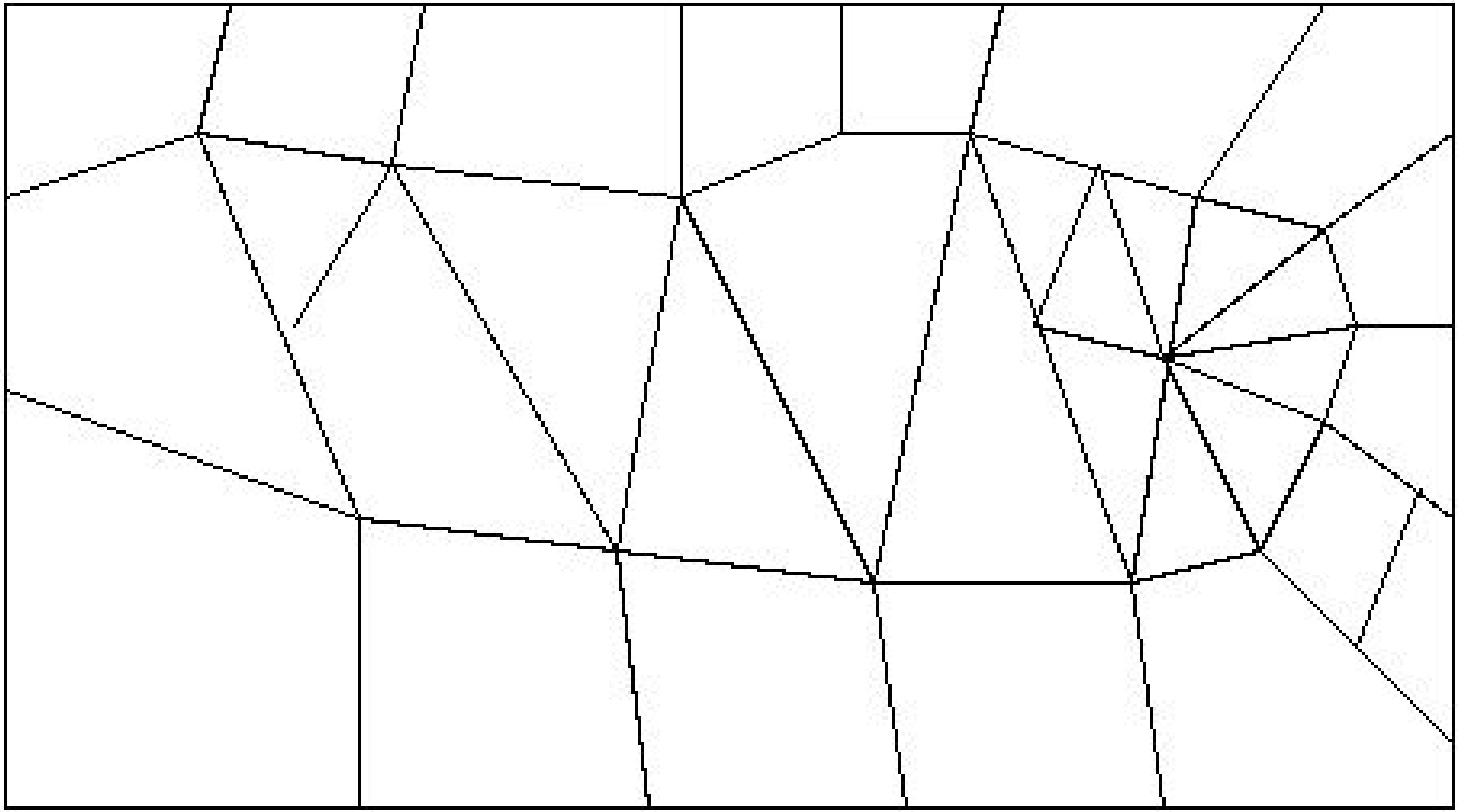
Stretched grid



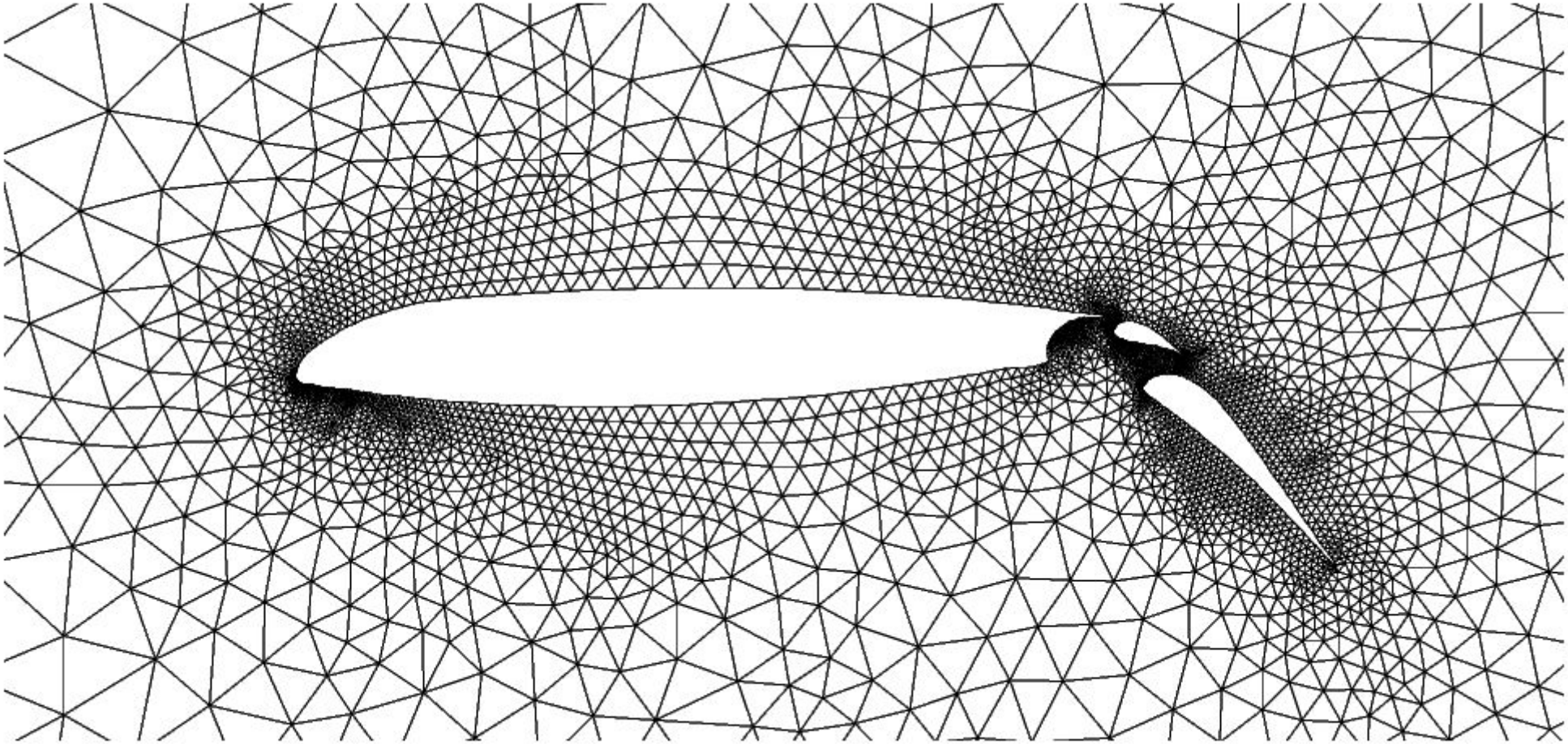
Hierarchical Cartesian (HC) grid



General grid



General grid



From Bern, Eppstein and Gilbert, “Provably good mesh generation”

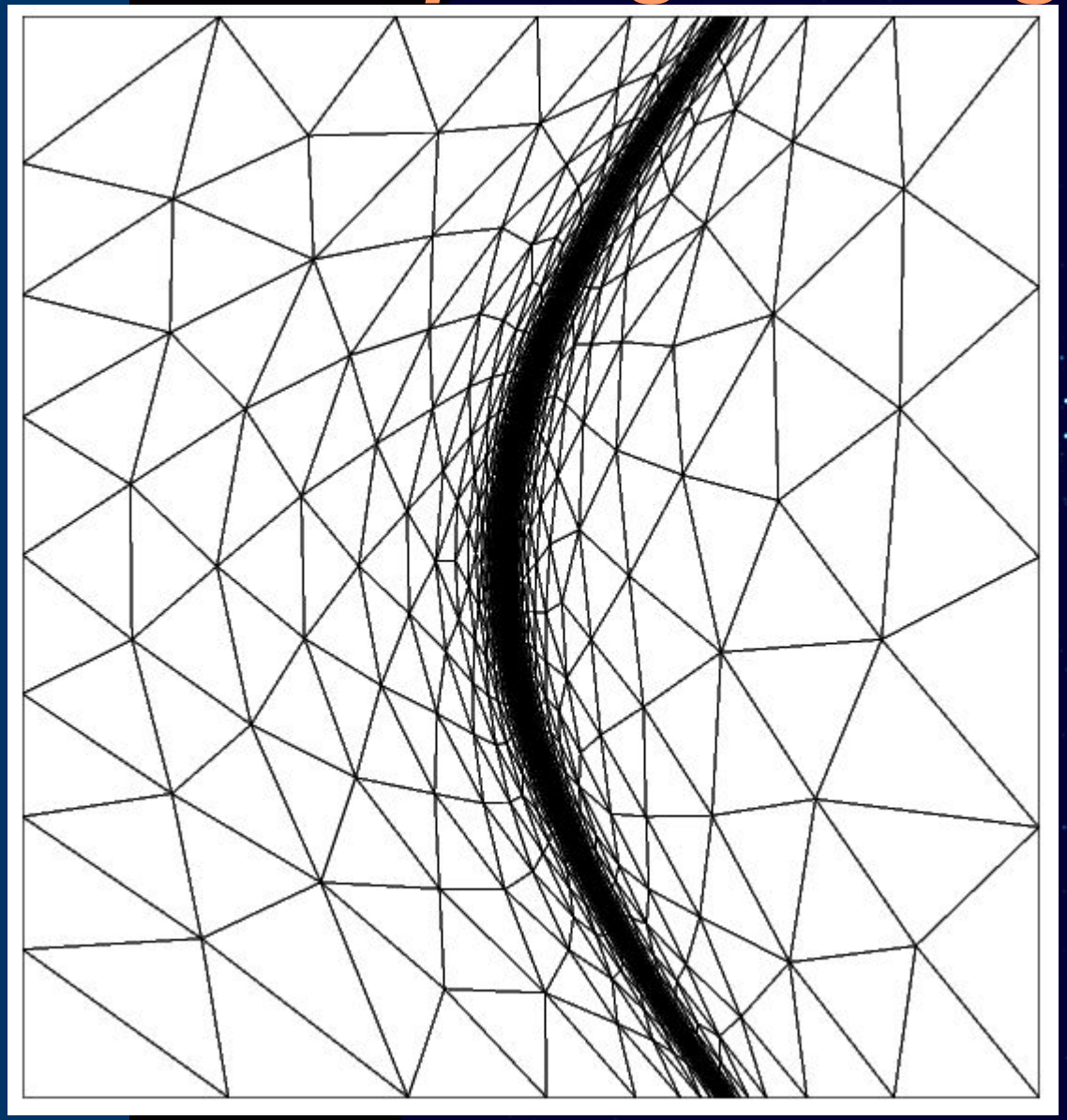
Interpolation from a grid

- To get any numbers out of the simulation, one needs to interpolate:
  - interpolation from uniform grid is handled in textbooks
  - stretched grid is a simple variant of this
  - in HC-grid, cell lookup takes *O(*level*)* operations, i.e. rather fast but not as fast as on uniform grid
- In general grid, cell lookup is nontrivial:
  - linear search from a cell list is usually too slow
  - caching strategies are possibly needed
  - or one has to use auxiliary HC-grid
- Conservative interpolation (convolution of cloud and cells) should in principle be used with FVM, although in practice one often uses pointwise interpolation even with volume grid

How much does adaptivity help?

- Quite a lot.
- More accurate answer: Depends on whether you use HC-grid or even more general grid. Although much better than stretched uniform grid, the HC-grid still gets you only part of the benefits.
- In HC-grid:
  - Representing a surface such as a *magnetopause* or *bow shock* effectively reduces dimension by one: from $O(N^3)$ we get down to $O(N^2)$
  - *R*od or line (~ magnetosphere): from $O(N^3)$ to $O(N)$
  - A point (~Earth): from $O(N^3)$ down to $O(1)$
- With anisotropic "brick grid":
  - We get down to $O(N)$ or smaller in all the above cases
  - If grid cells allowed to bend, we get down to $O(1)$

Anisotropic, general grid

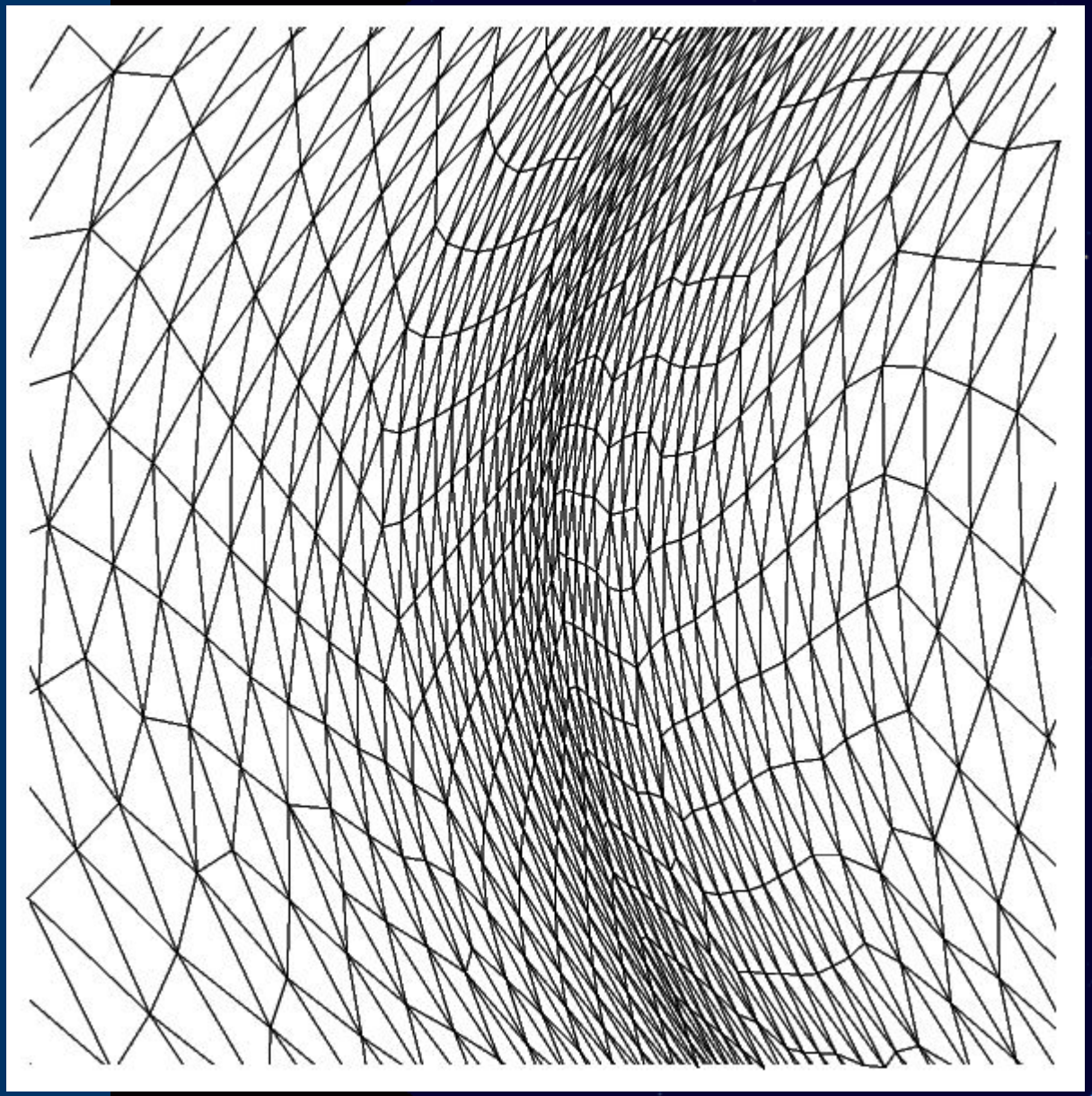


This looks nice, but has one drawback still: it is triangular.

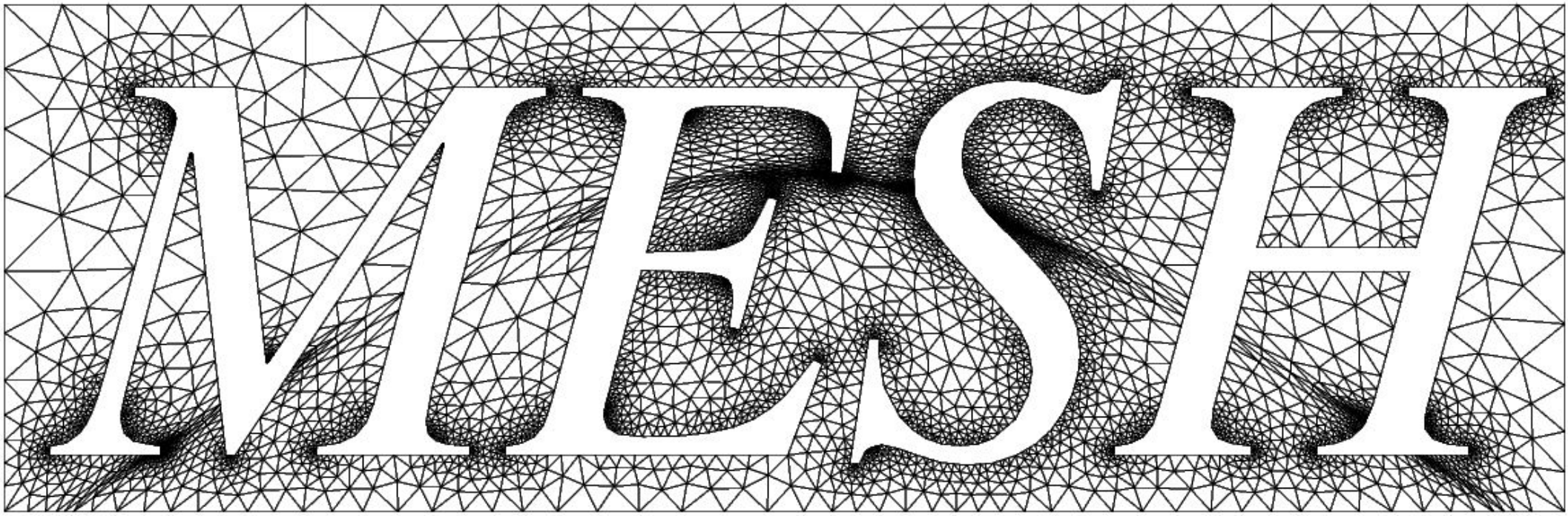
Recall that hexahedral grids typically work better numerically.

From Bossen and Heckbert, “A pliant method for aniso. mesh gen.”

Detail of the previous

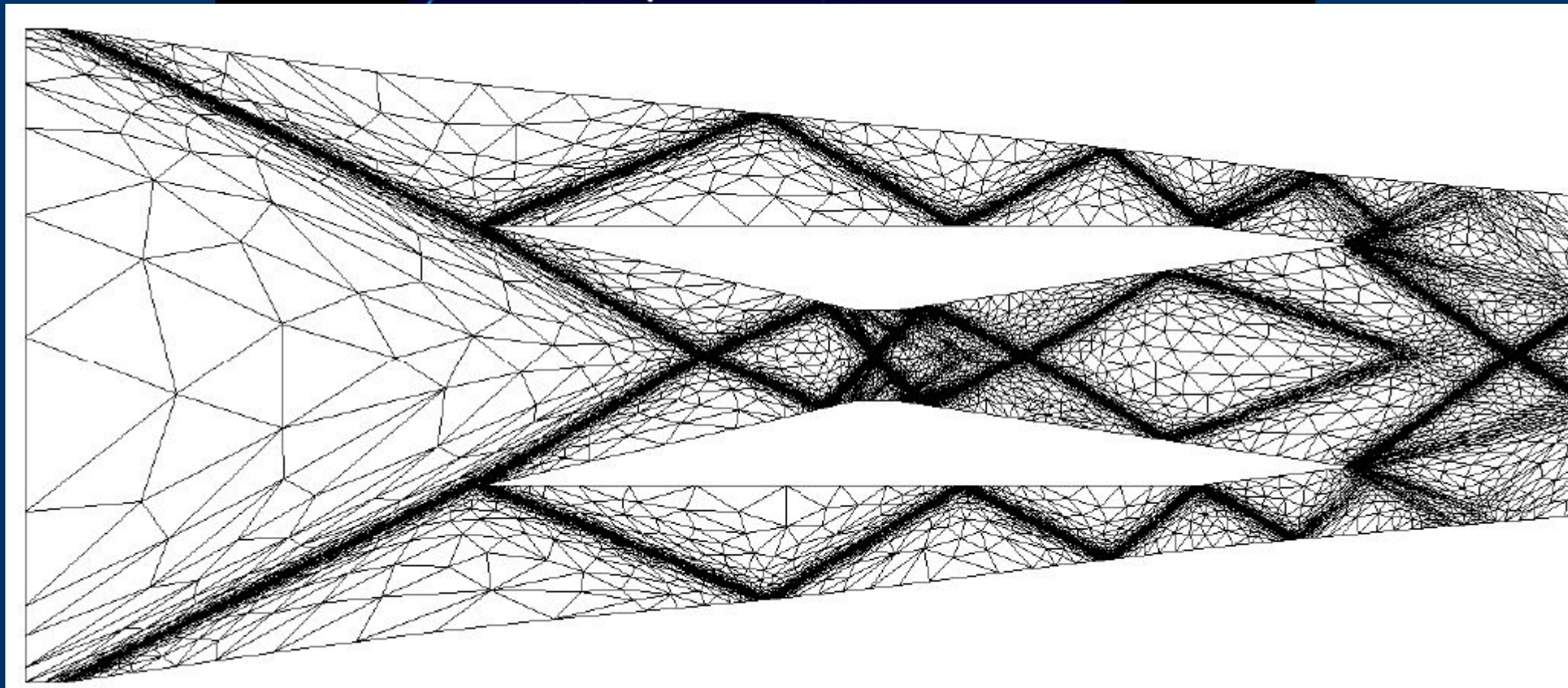


From Bossen and Heckbert, “A pliant method for aniso. mesh gen.”



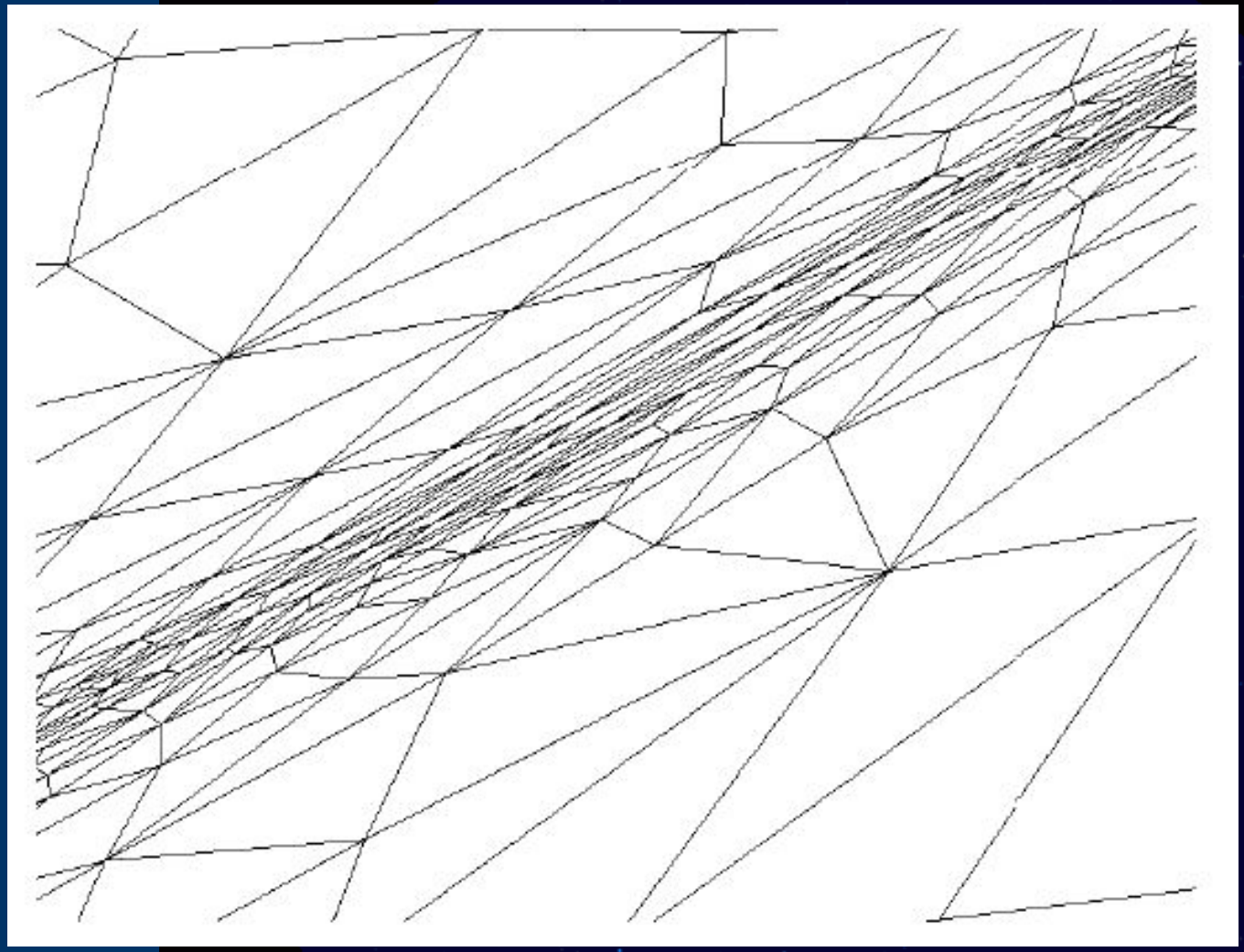
From Bossen and Heckbert, “A pliant method for aniso. mesh gen.”

2-D supersonic flow channel



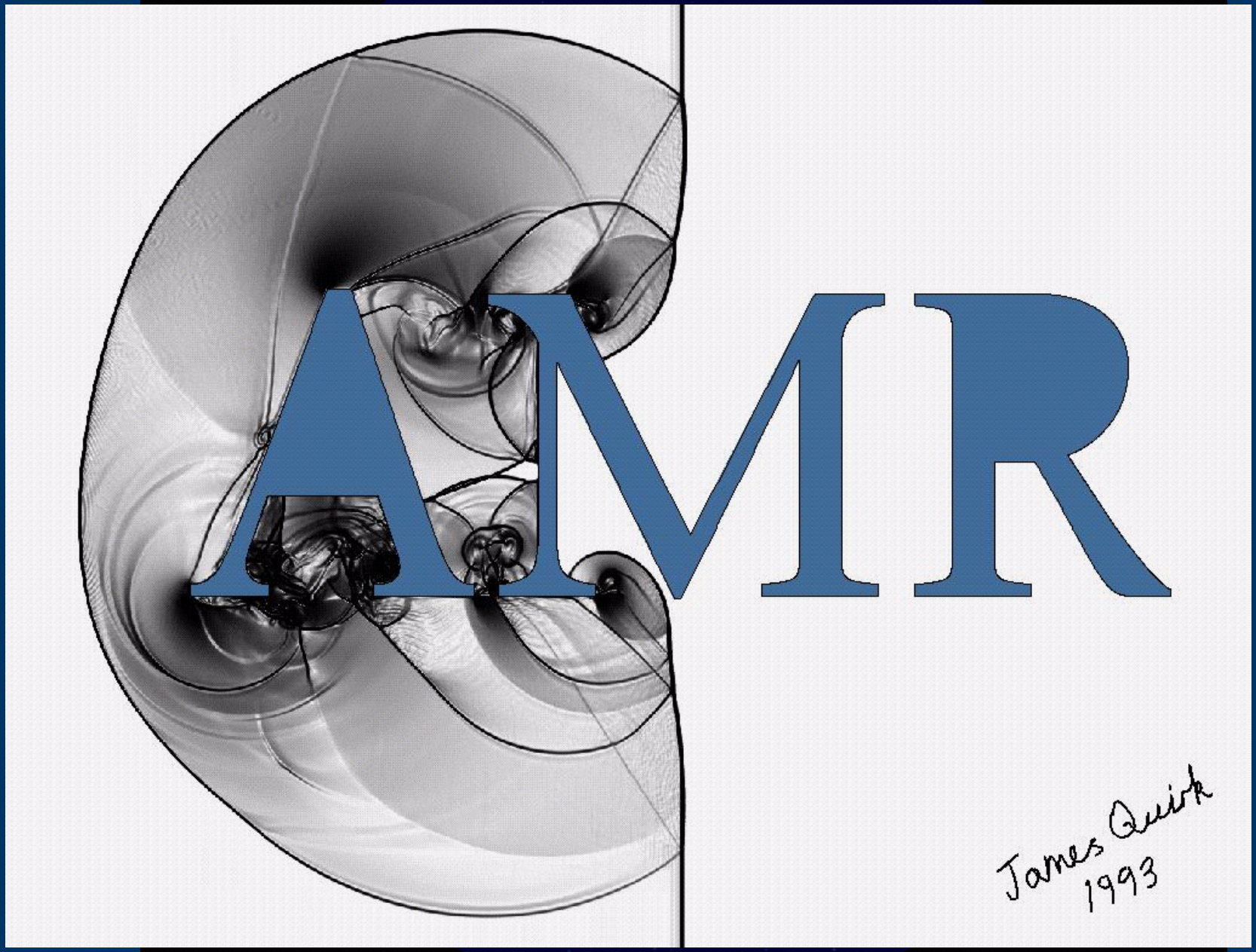
From Frey and Alauzet, “Anisotropic mesh generation for transient flows simulations”

Local zoom of the previous



From Frey and Alauzet, “Anisotropic mesh generation for transient flows simulations”

I show this once more...



Particle propagators

- Everyone must know the *Buneman-Boris time-symmetric Lorentz force integrator*:
  - It's energy conservative
  - It's simple to program and executes fast
  - Even if run with too large timestep, it remains stable and reproduces Larmor motion with exaggerated Larmor radius
- Its relativistic generalisation is also easy
- Switching from *Buneman* to guiding centre approximation and back is a possibility to increase performance
  - Each switch randomises phase and breaks some invariants
- Using the *monopole solver* may be an alternative
  - (Janhunen and Olsson, 2002)

Buneman-Boris Lorentz force integrator

$$\frac{\mathbf{x}_i^{n+1/2} - \mathbf{x}_i^{n-1/2}}{\Delta t} = \mathbf{v}_i^n$$

$$\frac{\mathbf{v}_i^{n+1} - \mathbf{v}_i^n}{\Delta t} = \frac{q_i}{m_i}\left[\mathbf{E} + \frac{1}{2}\left(\mathbf{v}_i^n + \mathbf{v}_i^{n+1}\right) \times \mathbf{B}\right]$$

- Solve algebraically for $\mathbf{x}_i^{n+1/2}$ and $\mathbf{v}_i^{n+1}$
- E and B are interpolated at $x_i^{n+1/2}$
- For implementation with least FLOPs, see Birdsall and Langdon book

The “monopole solver”

- Charged particle orbit in dipole field cannot be solved analytically, but in magnetic monopole field it can (spiral on a cone)
- Idea: Approximate **B**-field around particle by a monopole field (the monopole is not at centre of Earth but at some other location which varies as the particle moves)
- Since the solution is exact, timestep can be *much longer* than with Buneman-Boris integrator
- Monopole field is *convergent* and thus often better local approximation than constant field

Elliptic equations

- Types of elliptic equations:
  - Poisson
  - Separable coefficients (Poisson and Helmholz in spherical coordinates, e.g.)
  - General elliptic equations (coefficients depend on all coordinates in non-factorable way)
- Where they arise from:
  - Electrostatic particle or Vlasov simulation (Poisson)
  - Implicit particle simulation (General elliptic)
  - Current continuity equation in ionosphere (General elliptic)
  - div(B) removal in MHD (Poisson)
  - Electromagnetic particle sim. (Poisson + Helmholz)

"Rapid" elliptic solvers

- Applicable to constant-coefficient equations or equations where coefficients factor like f(x) g(y)
- If parallelisation strategy allows it, FFT method is straightforward and nearly optimal speed (although not quite; for better, see Hockney and Eastwood book)
- One of the dimensions can also be done with tridiagonal solver, which can be parallelised better than FFT e.g. by pipelining
- Equally important than speed is the fact that the FFT/tridiag methods are usually quite stable and produce reliable answer

Iterative elliptic solvers

- If rapid solvers are not applicable, iterative solvers must be used
- There are many (Gauss, Gauss-Seidel, SOR, ADI, SIP, multigrid,...), but only one really works well: the Conjugate Gradient (or Bi-CG) algorithm
- Look up the pseudocode from Numerical Recipes and implement in your language
  - It's beautiful, relatively simple and very general!
- In many cases you don't even need a preconditioner
  - But restarting the algorithm every now and then is simple to program and may be beneficial also
- CG has applications also in advanced data analysis (e.g. auroral tomography)

About roundoff error in general

- When grids become larger, roundoff error becomes more and more of a problem
- The problem is especially prominent with elliptic solvers
- For example, the SIPSOL solver works very well up to grid sizes about 50x50; for larger than that, it often does not converge at all
- It is easy to generate example problems (e.g., just by using random number coefficients) where any known elliptic solver fails to converge
- Robustness is much more severe problem than what you get by e.g. reading “Numerical Recipes” (which is otherwise very recommendable book!)

Future prospects of simulations

- Simulations will probably separate into two disciplines:
  - (1) Quick solution of problems using self-written simple programs on your PC or laptop
  - (2) Large, professional-quality parallelised software systems emerge for attacking challenging problems
- Since CPU speed increases:
  - Importance of ***numerical stability and robustness*** will increase
  - Importance of ***grid adaptivity*** will increase
  - Importance of ***Vlasov*** simulation will increase
  - Importance of ***multiphysical problems*** will increase

Some advice

- Computing speed grows exponentially with time. Thus every year, some problems turn from non-solvable to solvable. This has been the situation ever since the first computers were built.
- However, not all relevant problems are solved during the year they first become solvable!
- Thus, there are physically relevant problems that are easy to simulate nowadays, which would have been at “grand challenge” class 10-20 years ago, but which nobody yet simulated.
- Thus, while hard problems are challenging, do not ignore the easy ones!

Prospects in the application domain

- In plasma physics, simulation models have not yet reached the maturity of other “normal” physics:
  - In my opinion this is mainly due to the fact the plasma physics should really be done in 6-D phase space.
  - For an arbitrary planet, shape and size of magnetosphere is simulated correctly, but not such processes as rate of atmospheric erosion or coherent radio output power (for the latter two, predicting even the correct order of magnitude is hard)
  - Compare this to climate simulation (neutral fluid dynamics with radiation), where e.g. global temperature comes out correctly to ~1 % or better
  - Plasma ***is*** inherently difficult, both in theory, and to simulate.

Quantifying plasma difficulties

- In neutral fluid, we have the *sound wave* and the *entropy wave* (*contact discontinuity*). Sound is typically approximated away in geophysical flows.
- In MHD, the sound wave splits into *slow*, *Alfven* and *fast magnetosonic* wave. Propagation properties depend on **B**-field direction.
- In real (i.e. kinetic-based) plasma physics, we have almost too many wave modes to list and remember:
  - whistler waves (modified fast magnetosonic)
  - ion acoustic waves
  - ion Bernstein waves
  - lower and upper hybrid waves
  - etc., etc., (e.g., all electron modes missing from this list...)

Importance of plasma in space

- Gravity governs the universe, but plasma physics has its role to play everywhere. Many of the nontrivial questions in space are plasma-physical:
  - ***reconnection*** and its resulting particle acceleration
  - ***shock acceleration***
  - ***dynamo action*** (magnetic fields of stars, planets, magnetars..)
  - details of gamma ray bursts, gamma flares, supernovas
  - details of solar system formation
  - details of stellar evolution (loss of angular momentum?)
  - primordial magnetic fields (are there any?)
- Many of these have *astrobiological dimension,* e.g:
  - stability of planetary atmospheres on low-mass planets&stars
  - details of life-threatening supernovas and gamma ray bursts
  - strength of flares on lighter-than-Sun stars

Importance of plasma in space 2

- To attack some of these questions, we need *multiphysical advanced simulations*, *together with* good physical understanding of their results.
- For example, for stars: "*relativistic* or *non-relativistic MHD with gravity, radiation and particle physics & good equations of state, parallelised on automatically adapted grid*" would be high on the wish list
- Or, for planets: May I have one of your "*parallelised Vlasov solvers with anisotropically adaptive grid, with neutral collisions and charge-exchange processes and modelled small-scale wave-electron interactions*" (with home delivery, please)

You were seeing

- MHD
- Particle simulation
- Hybrid simulation
- Vlasov simulation

- Grid types
- Particle propagators
- Elliptic solvers
- And some future prospects