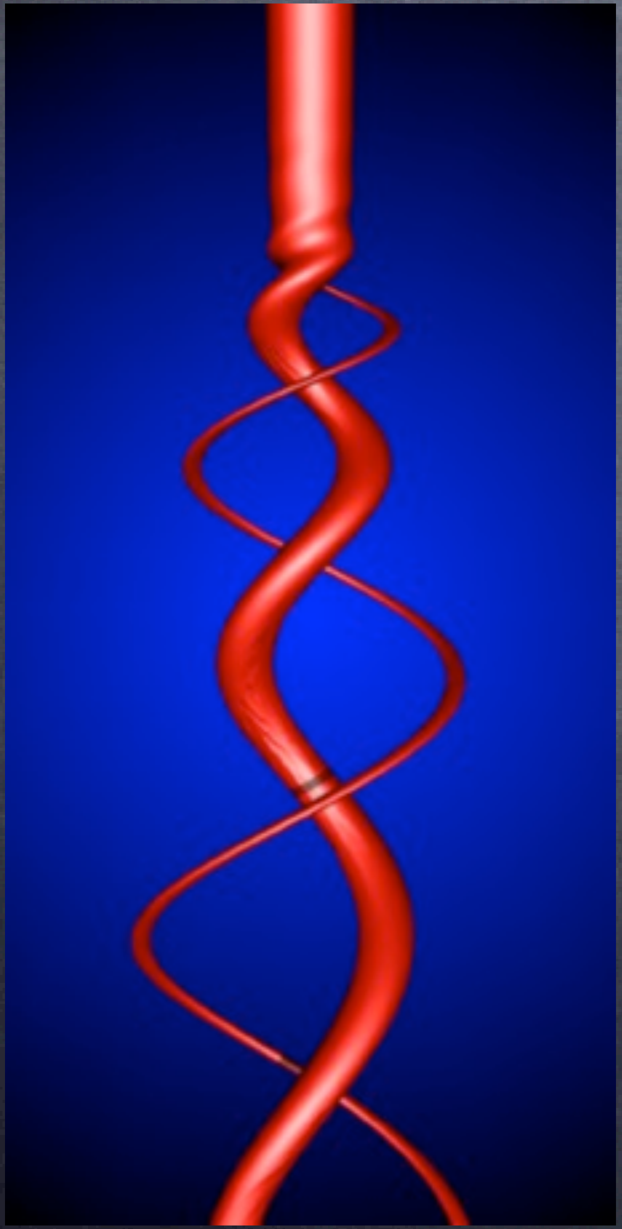
Introduction to Theory and Numerics of Partial Differential Equations V: And now for something completely different...

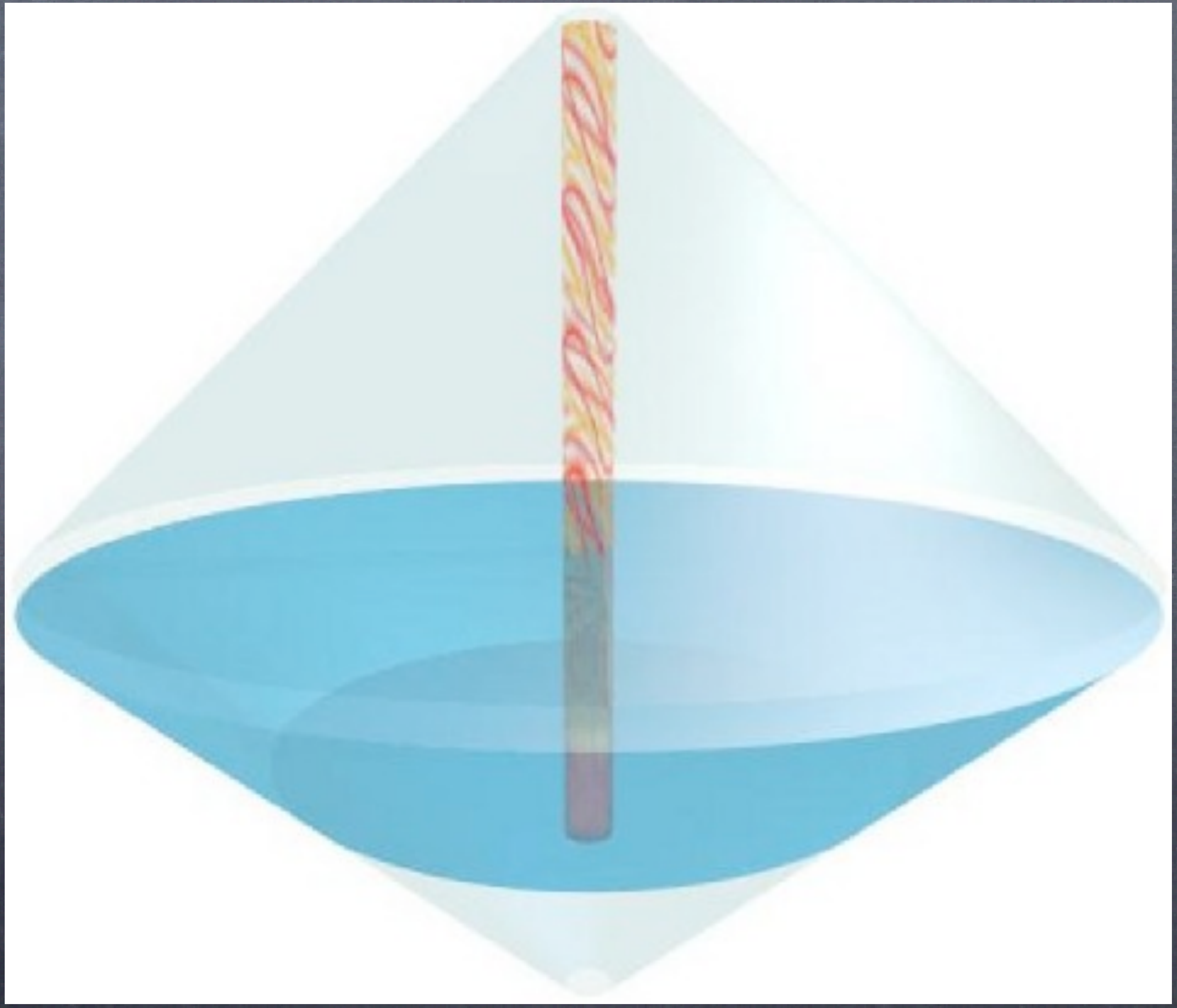


Sascha Husa

ICTS Summer School on Numerical Relativity

Bangalore, June 2013

Capturing radiation and infinite domains



Isolated systems as models of sources of GW

Key concepts to describe "astrophysical" processes in GR: essential independence of the large-scale structure of the universe, "radiation leaves system".

→ physical idealization: isolated system – geometry "flattens at large distances" or approaches some cosmological background geometry.

GR: mass, momentum, emitted gravitational radiation can not be defined unambiguously in local/quasilocal way – only make sense in asymptotic limits.

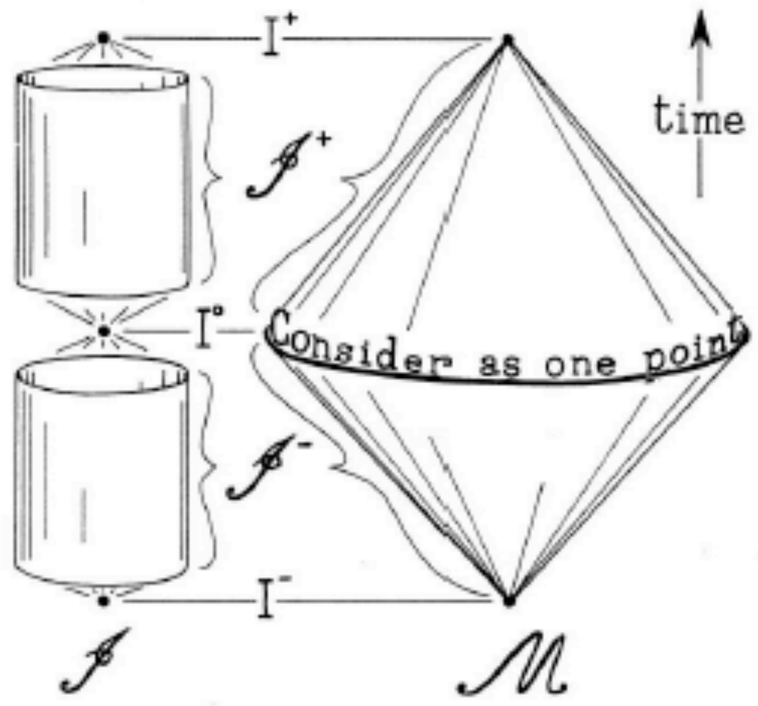
Formalize as asymptotically flat or asymptotically de Sitter/AdS spacetimes

AF spacetimes usually used to model sources of GWs.

Beware: there are 3 directions toward infinity: timelike / spacelike / null!

Compactification in these 3 directions behaves very differently.

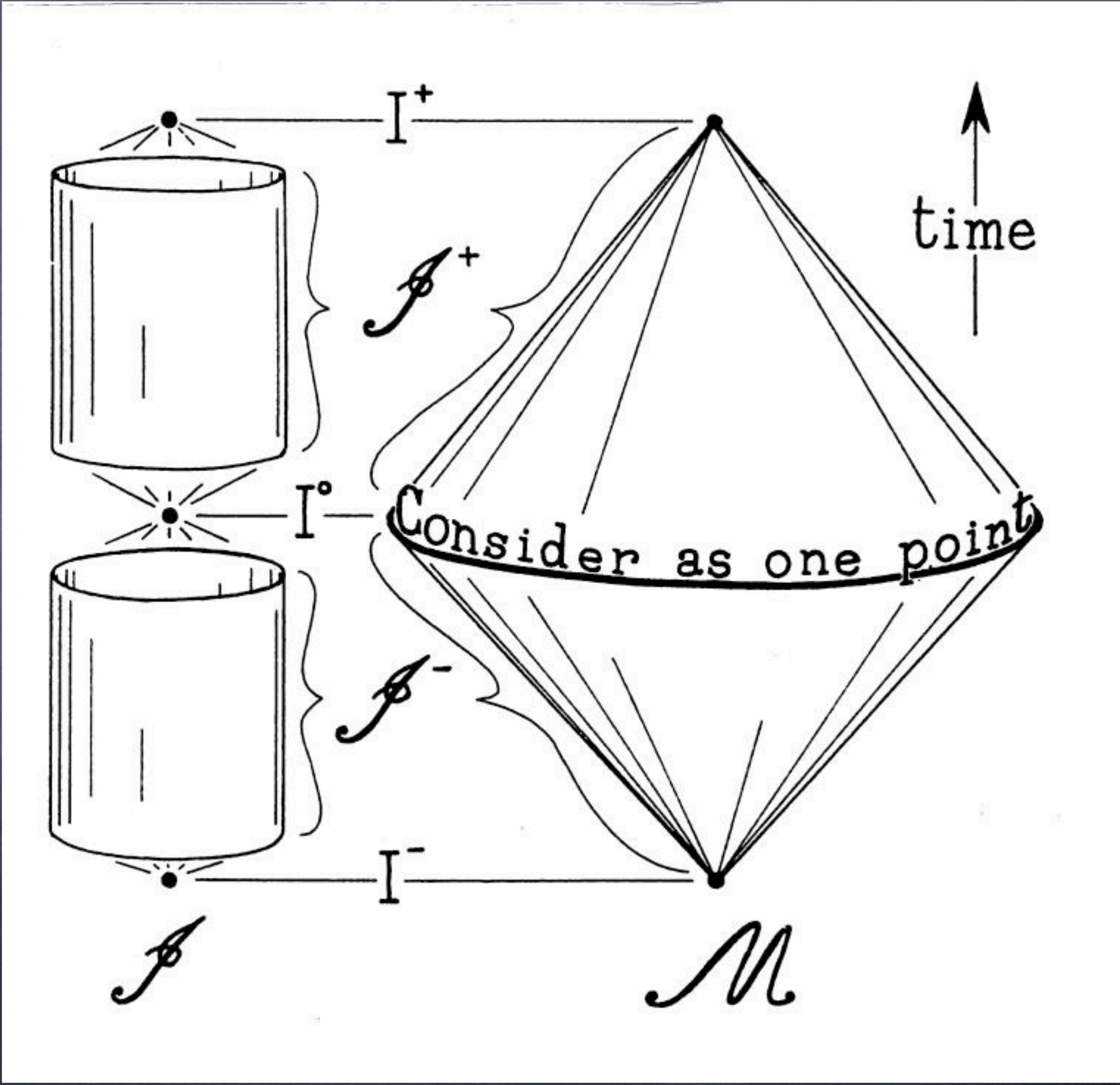
Key idea: use conformal rescalings (Penrose).



Penrose diagram

Dispaly global causal structure of spacetime, uses conformal rescaling

$g_{\text{compact}} = \Omega^2 \, g_{\text{phys}}$



Conformal Compactification

Using conformal rescaling to an unphysical spacetime, we can discuss asymptotics in terms of local differential geometry.

$\tilde{g}_{ab} = \Omega^{-2} g_{ab}, \quad \tilde{\mathcal{M}} = \{p \in \mathcal{M} \mid \Omega(p) > 0\}, \quad \text{“}\infty\text{''} = \partial\tilde{\mathcal{M}} = \{p \in \mathcal{M} \mid \Omega(p) = 0\}.$

Einstein's vacuum equations in terms of Ω & g_{ab}:

$$\tilde{G}_{ab}[\Omega^{-2} g] = G_{ab}[g] - \frac{2}{\Omega}\left(\nabla_a \nabla_b \Omega - g_{ab} \nabla_c \nabla^c \Omega\right) - \frac{3}{\Omega^2} g_{ab} \left(\nabla_c \Omega\right) \nabla^c \Omega.$$

singular for $\Omega = 0$, multiplication by $\Omega^2 \rightarrow$ degenerate principal part @ $\Omega = 0$.

- asymptotically flat: null infinity is null.
- asymptotically de Sitter: null infinity is space-like.
- asymptotically anti-de Sitter: null infinity is time-like – boundary data required.

Different approaches to compactified equations:

- Evolution along null surfaces – works well, but not general due to caustics.
- Compactification along spacelike directions: standard method to construct initial data, underresolved blue-shifted waves in evolution.
- $K = const.$ – spacelike “hyperboloidal” surfaces reaching null infinity ($t^2 - \vec{x}^2 = const.$ in Minkowski).

Wave extraction via Ψ_4

Wave-zone: adopt transverse-traceless (TT) gauge, all the information about the radiative degrees of freedom contained in h_{ij}:

$$h_{ij} = h_+(\mathbf{e}_+)_{ij} + h_\times(\mathbf{e}_\times)_{ij}, \tag{5}$$

$$(\mathbf{e}_+)_{ij} = \hat{\iota}_i\hat{\iota}_j - \hat{\phi}_i\hat{\phi}_j\,, \qquad \text{and} \qquad (\mathbf{e}_\times)_{ij} = \hat{\iota}_i\hat{\phi}_j + \hat{\iota}_j\hat{\phi}_i\,. \tag{6}$$

Newman-Penrose scalar method, in wave zone: $\mathrm{h} = h_+ - \mathrm{i}h_\times$ as

$$\mathrm{h} = \lim_{r\to\infty}\int_0^t \mathrm{d}t' \int_0^{t'} \mathrm{d}t''\Psi_4, \qquad \Psi_4 = -R_{\alpha\beta\gamma\delta}n^\alpha\bar{m}^\beta n^\gamma\bar{m}^\delta, \tag{7}$$

Null-tetrad ℓ (in), n (out), m, $\bar{m}$,

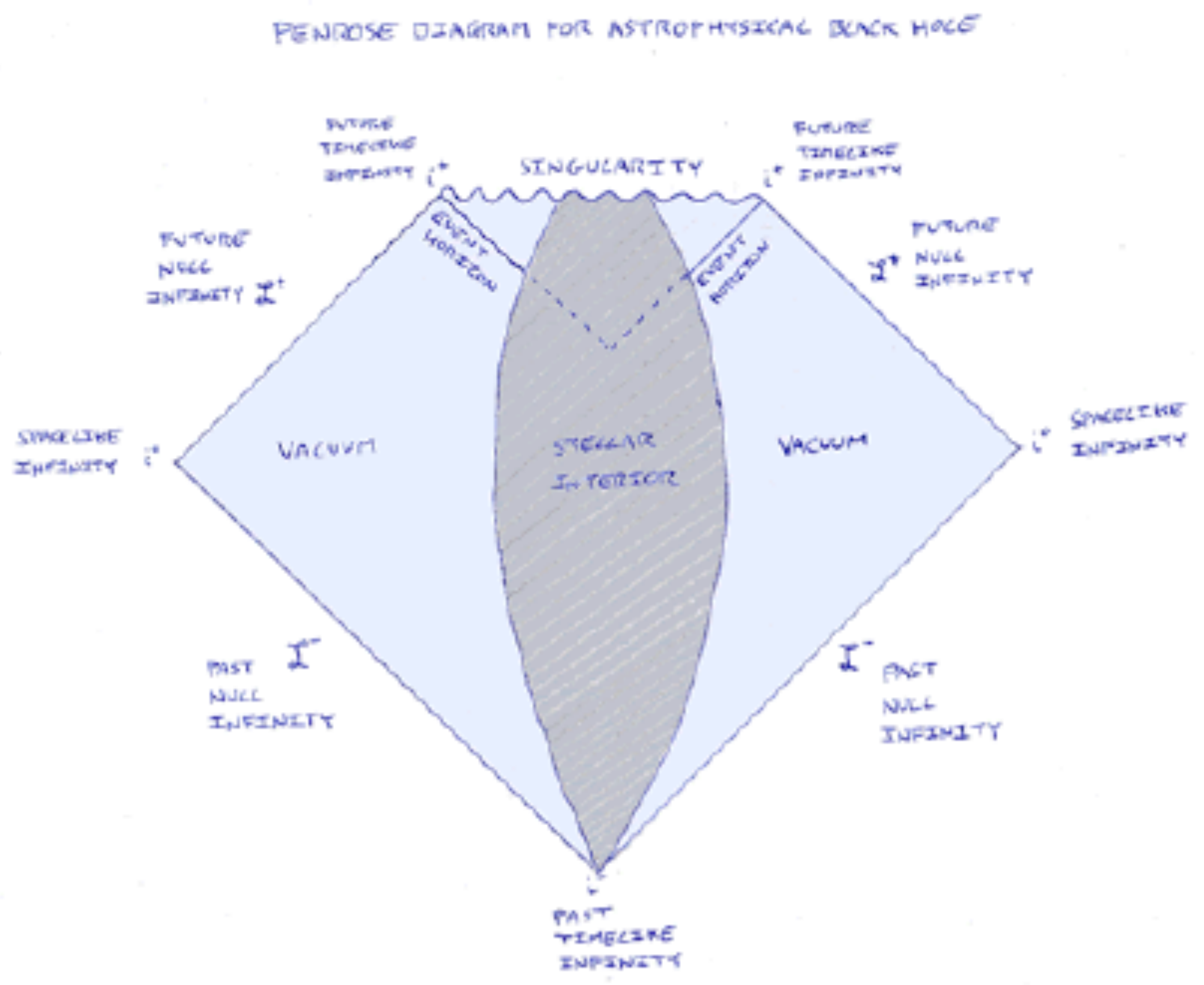
$$-\ell\cdot n = 1 = m\cdot\bar{m}, \tag{8}$$

Spin-weight -2 fields represent symmetric trace-free tensor fields on a sphere (in our case $R_{\alpha\beta\gamma\delta}n^\alpha n^\gamma$) in terms of a complex scalar field. Freedom in the choice of tetrad used in defining Ψ_4!

Radiation "lives" at null infinity

Taking appropriate limit in $\mathcal{M}$, worldlines of increasingly distant geodesic observers converge to null geodesic generators of $\mathscr{I}^+$ (proper time $\to$ Bondi time)!

Compactification at i^0 leads to "piling up" of waves, at $\mathscr{I}^+$ this effect does not appear – waves leave the physical spacetime through the boundary $\mathscr{I}^+$.


PENROSE DIAGRAM FOR ASTROPHYSICAL BLACK HOLE
FUTURE TIMELIKE INFINITY i^+
SINGULARITY
i^+ FUTURE TIMELIKE INFINITY
FUTURE NULL INFINITY $\mathscr{I}^+$
EVENT HORIZON
EVENT HORIZON
$\mathscr{I}^+$ FUTURE NULL INFINITY
SPACELIKE INFINITY i^0
VACUUM
STELLAR INTERIOR
VACUUM
i^0 SPACELIKE INFINITY
PAST NULL INFINITY $\mathscr{I}^-$
$\mathscr{I}^-$ PAST NULL INFINITY
i^-
PAST TIMELIKE INFINITY

Observers situated at "astronomical" distances are modeled @ $\mathscr{I}^+$.

E.g. computing the signal at a GW detector, $\mathscr{I}$ more realistically corresponds to an observer sufficiently far way from the source to treat the radiation linearly, but not so far away that cosmological effects have to be taken into account.

Spherical harmonics decomposition

Project onto spin-weighted $s = -2$ spherical harmonics $Y_{\ell m}^{-2}$, e.g.

$$Y_{2-2}^{-2} \equiv \sqrt{\frac{5}{64\pi}} (1 - \cos\iota)^2 \, e^{-2i\phi}, \quad Y_{22}^{-2} \equiv \sqrt{\frac{5}{64\pi}} (1 + \cos\iota)^2 \, e^{2i\phi}. \tag{9}$$

$$A_{\ell m} = \langle Y_{\ell m}^{-2}, \Psi_4 \rangle = \int_0^{2\pi} \int_0^{\pi} \Psi_4 \overline{Y_{\ell m}^{-2}} \sin\theta d\theta d\phi \tag{10}$$

Orthonormality:

$$\frac{dE}{dt} = \lim_{r\to\infty} \left[\frac{r^2}{16\pi} \sum_{l,m} \left| \int_{-\infty}^{t} A_{\ell m} d\tilde{t} \right|^2 \right]. \tag{11}$$

Amplitude-phase split:

$$A_{\ell m}(t) = A(t) e^{i\varphi(t)}, \quad \omega(t) = \frac{\partial\varphi(t)}{\partial_t}, \tag{12}$$

Radiated energy, linear and angular momentum

Radiated energy and linear & angular momentum:

$$\frac{dE}{dt} = \lim_{r\to\infty}\left[\frac{r^2}{16\pi}\int_\Omega \left|\int_{-\infty}^t \Psi_4 d\tilde{t}\right|^2 d\Omega\right], \tag{13}$$

$$\frac{dP_i}{dt} = -\lim_{r\to\infty}\left[\frac{r^2}{16\pi}\int_\Omega \ell_i \left|\int_{-\infty}^t \Psi_4 d\tilde{t}\right|^2 d\Omega\right], \tag{14}$$

$$\begin{aligned}\frac{dJ_z}{dt} = -\lim_{r\to\infty}\Bigg\{\frac{r^2}{16\pi}\mathrm{Re}\Bigg[\int_\Omega\left(\partial_\phi\int_{-\infty}^t \Psi_4 d\tilde{t}\right) \\ \left(\int_{-\infty}^t\int_{-\infty}^{\hat{t}} \overline{\Psi_4} d\tilde{t} d\hat{t}\right) d\Omega\Bigg]\Bigg\},\end{aligned} \tag{15}$$

At finite extraction radius we need to perform "double Richardson extrapolation" – in extraction radius and grid spacing.

Total energy, linear and angular momentum

Define surface integrals (ADM integrals)

$$E(r) = \frac{1}{16\pi}\int_{S_r} \sqrt{g}\, g^{ij} g^{kl} \left(g_{ik,j} - g_{ij,k}\right) dS_l, \tag{16}$$

$$P_j(r) = \frac{1}{8\pi}\int_{S_r} \sqrt{g}\left(K^i_j - \delta^i_j K\right) dS_i, \tag{17}$$

$$J_j(r) = \frac{1}{8\pi}\epsilon_{jl}{}^{m}\int_{S_r} \sqrt{g}\, x^l \left(K^i_m - K\delta^i_m\right) dS_i \tag{18}$$

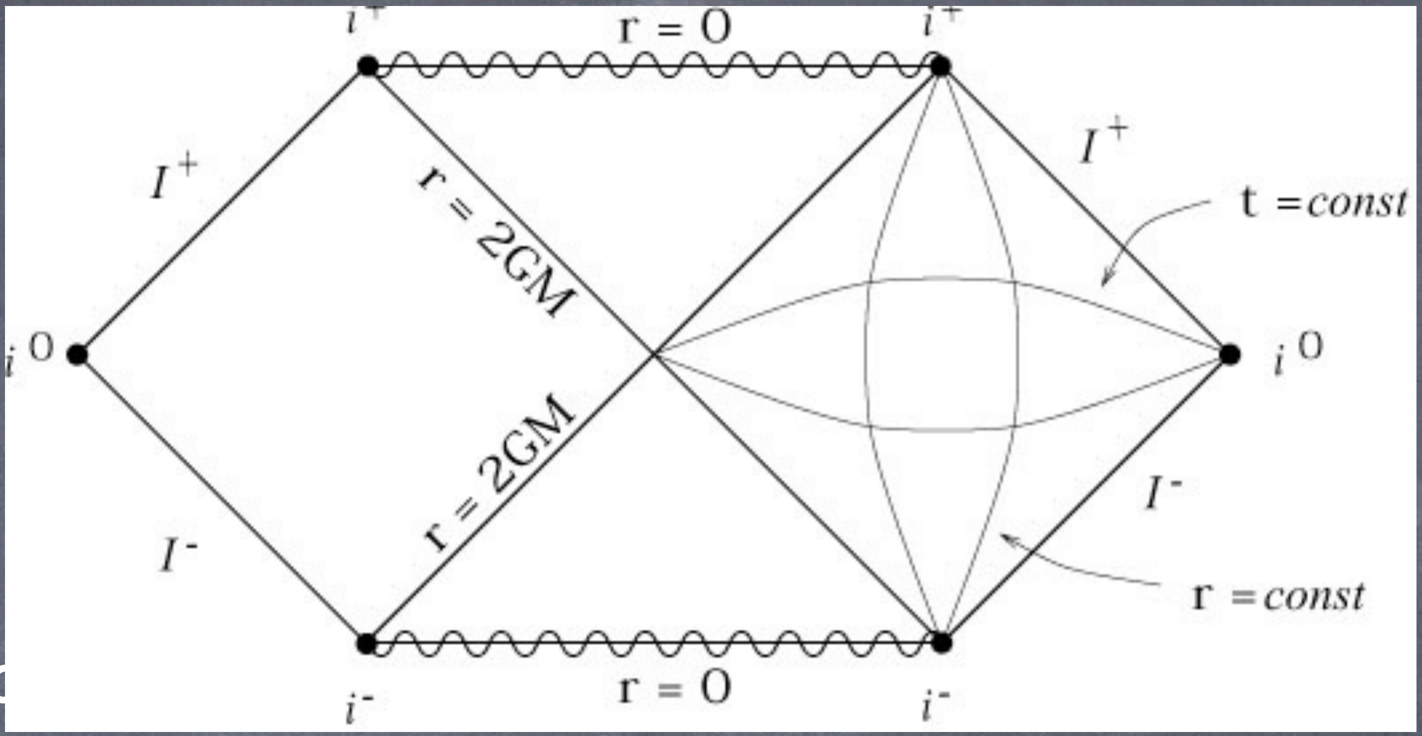
which have to be evaluated in an asymptotically Cartesian coordinate system.

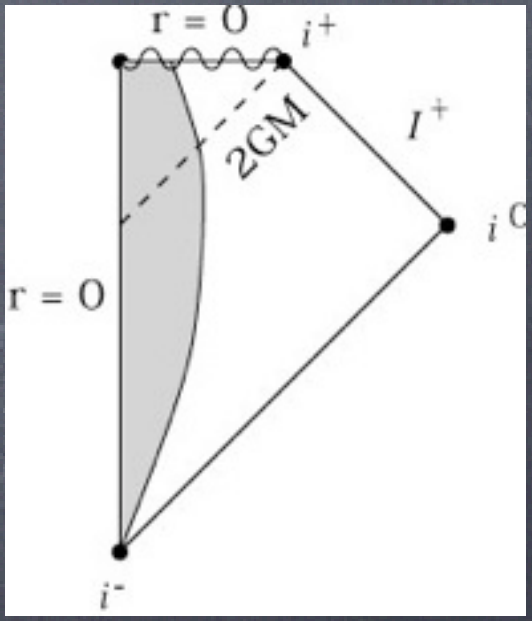
$$M_{ADM} = \lim_{r\to\infty} E(r), \quad P_j = \lim_{r\to\infty} P_j(r), \quad J_j = \lim_{r\to\infty} J_j(r)$$

and the rest mass M_R can be defined as $M_R^2 = M_{ADM}^2 - \sum_{j=1,3} P_j P_j$.

Bondi quantities: take values at fixed retarded time.

black holes and event horizons

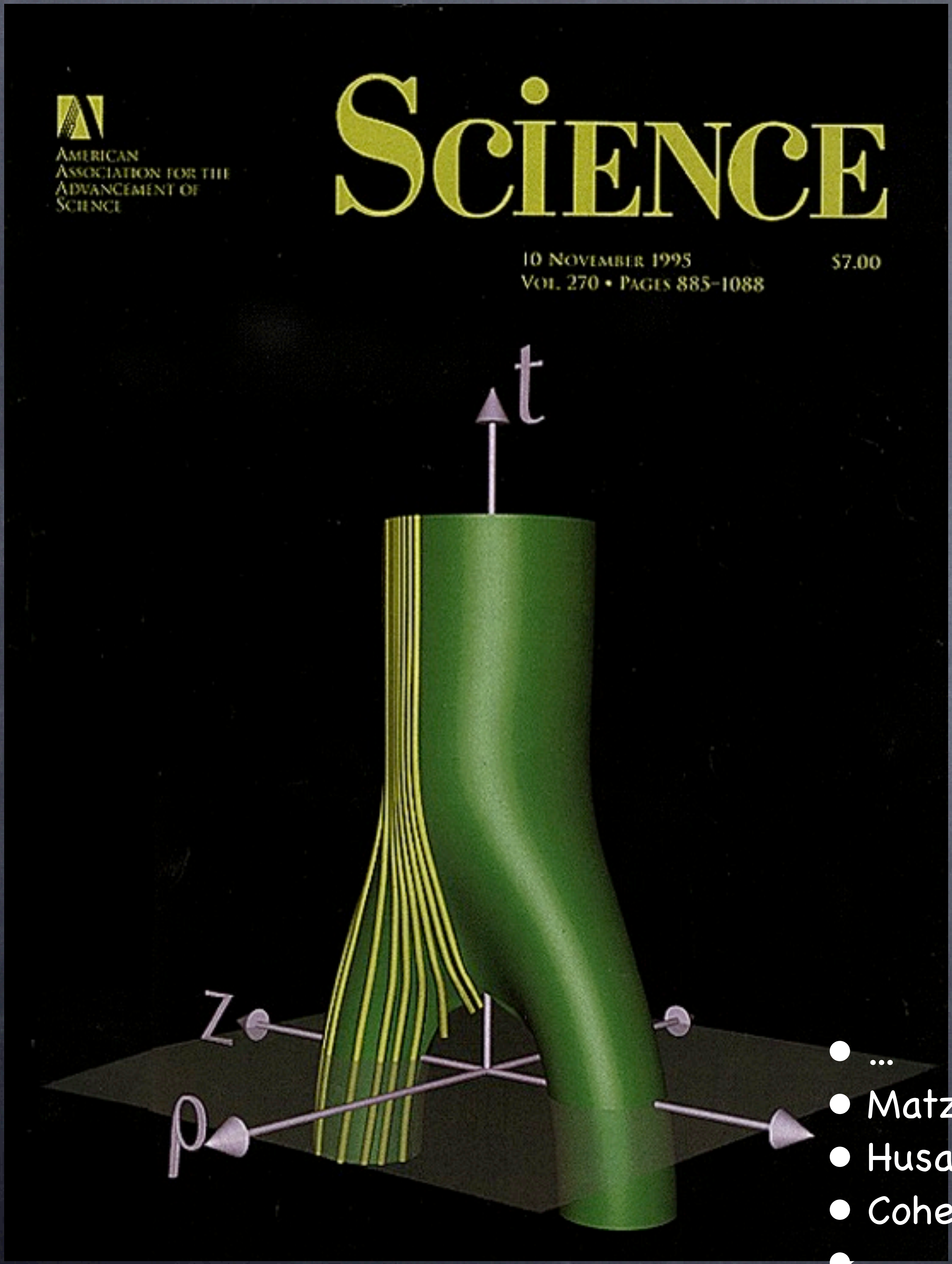


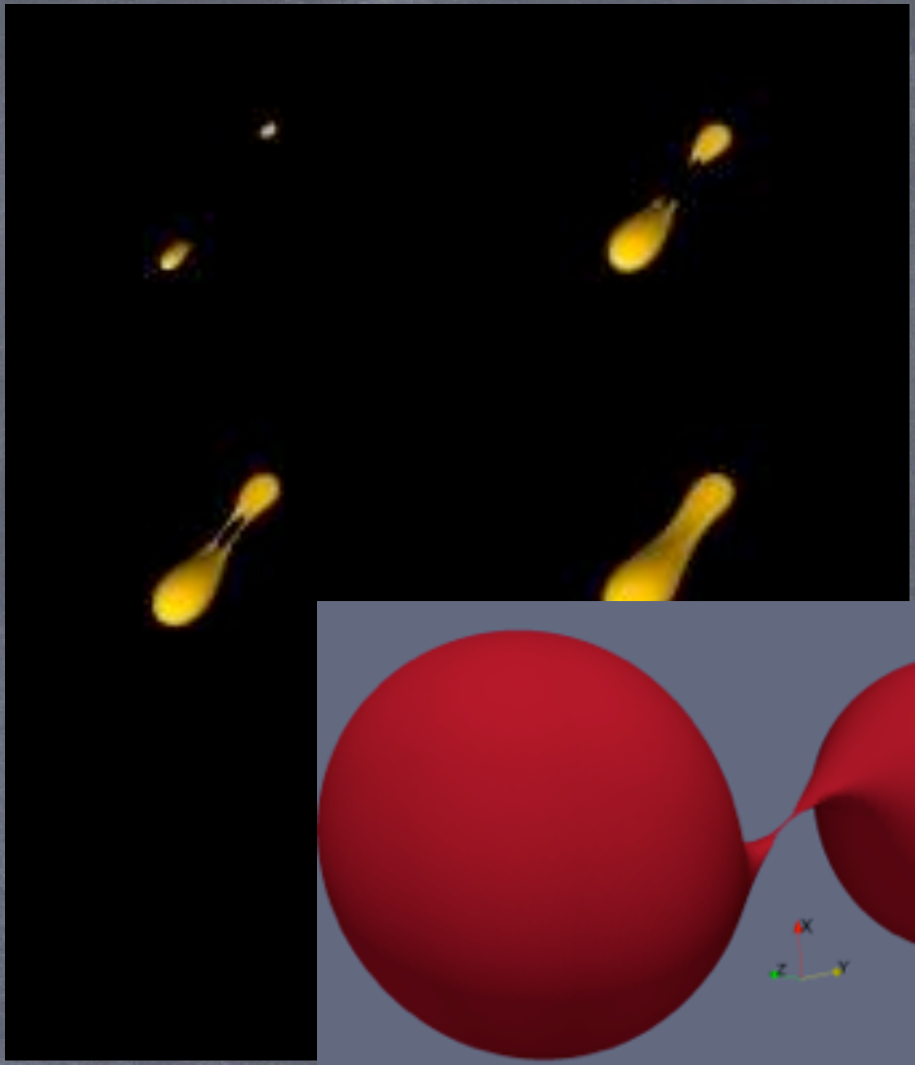


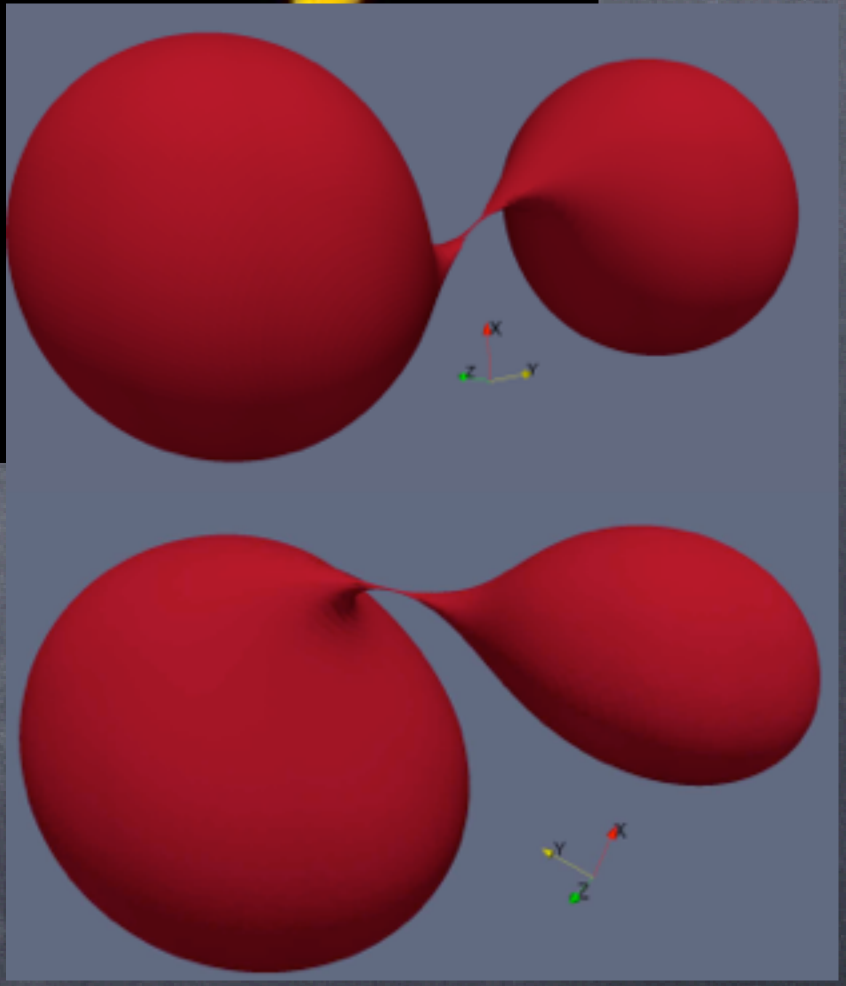
- Event horizon: null surface trac from i^+, BH is the spacetime region inside the EH.
- Applying the rigorous definition in a numerical spacetime won't work.
- Reasonable approximation: trace back spherical null surface starting well after merger.
- level set method: EH @ $f(x^i,t) = 0$ -> $g^{\alpha\beta}\partial_\alpha\partial_\beta f = 0$

$$\partial_t f = \frac{-g^{ti\partial_i f + \sqrt{(g^{ti}\partial_i f)^2 - g^{tt}g^{ij}\partial_i f \partial_j f}}}{g^t t}$$

- re-initialize to deal with steepening gradients
- Diener CQG20 (2003), Cohen+ CQG26 (2009) [geodesics]


American Association for the Advancement of Science
Science
10 November 1995
Vol. 270 • Pages 885-1088
$7.00
t
z
ρ

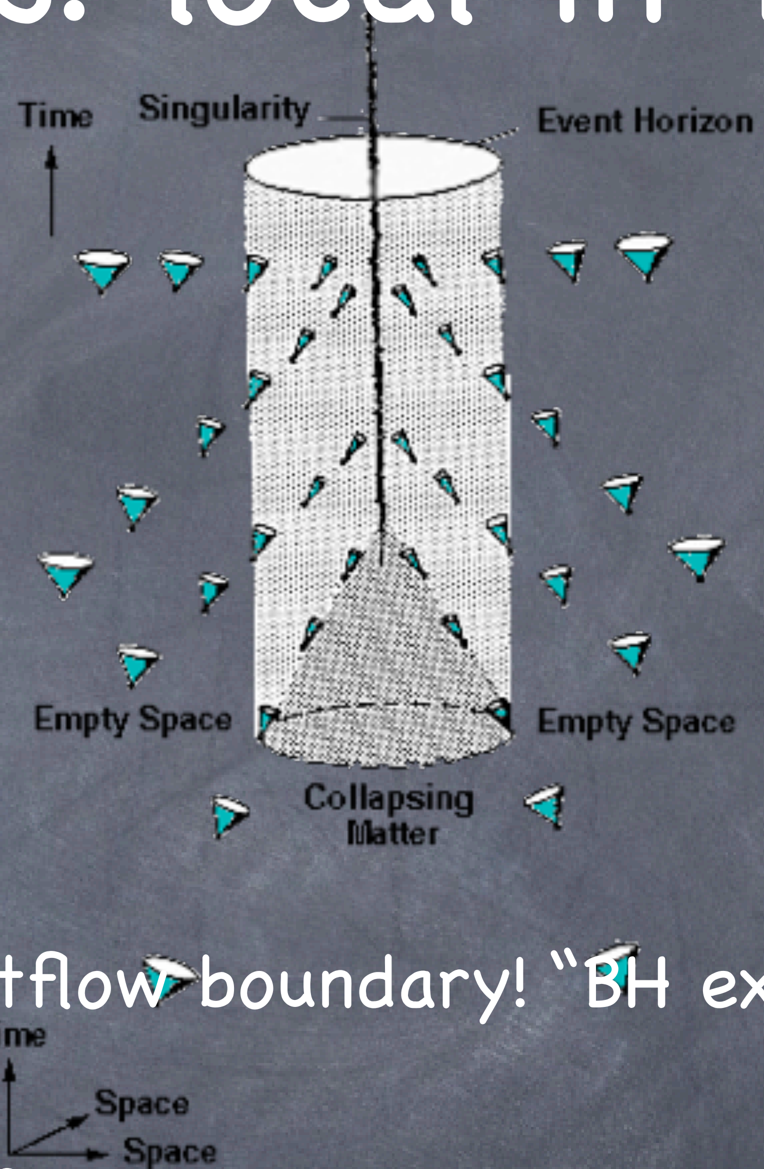




- ...
- Matzner+ Science 270 (1995)
- Husa, Winicour, Phys.Rev. D60 (1999) 084019
- Cohen+, Phys. Rev. D 85, 024031 (2012)
- ...

Apparent horizons: local in time

- Trapped surface: spacelike 2-surface with the property that the ingoing and outoing wavefronts decrease in area. Light cones "tip over"!
- AH: outermost "marginally" trapped surface.



- Characteristic speeds ≤ c: AH is a outflow boundary! "BH excision".
- Singularity theorems:
  very general conditions: trapped surface => spacetime singularity
- Elliptic equation for constant expansion surfaces:

$$\theta_\pm = K - r^a r^b K_{ab} \pm D_a r^a$$

- Solve directly with elliptic solver or more robust parabolic flow.
- Thornburg, Liv. Rev. Rel. 10 (2007), BUT: flow **is** fast.

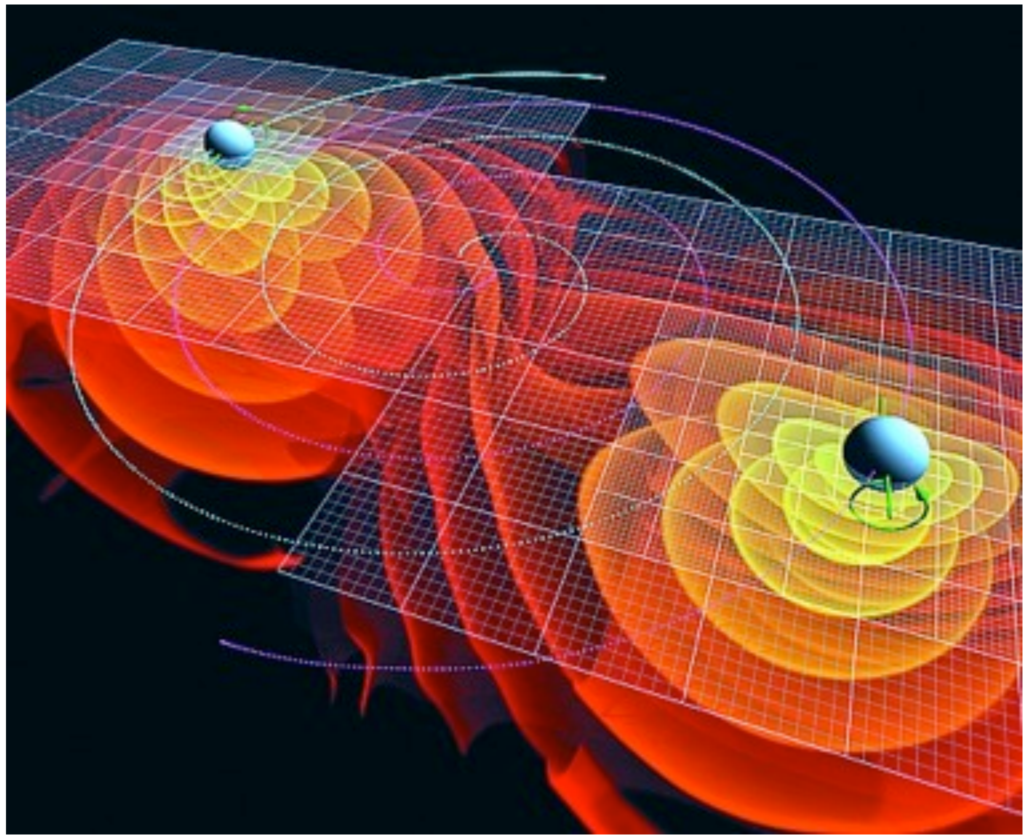
Scales and mesh refinement

- binary black holes (compact objects) have many different length scales:
  - the individual black holes
    - (radiation efficiency ~ compactness of source)
    - m1/m2 > 2 creates trouble!
- orbital scale: typically start at separations > 10 M
- wave scale for spherical harmonic (l,m):

$$\lambda = T,\ T_{\text{wave}} = \frac{T_{\text{wave}}}{m} \qquad 10M \leq T^0_{QNM} \leq 16M$$

$$M\omega_{orb} = \left(\frac{R}{M}\right)^{-3/2} \quad T_{orb}(10M) \approx 200M$$

- $1/r^n$ background falloff
- ambiguity in boundary conditions:
  - causally isolate boundaries -> 1000s of M



AMR computational infrastructure

- Adaptive mesh refinement significantly increases the complexity of a numerical code!
- Good news: For compact binaries, we do not need a general mesh refinement algorithm, just let refinement boxes follow the objects (possible exception with very dynamical matter fields).
- Coarser grids allow bigger time steps!
  - efficient algorithm needs to allow for different time steps for different grids!
  - more difficult to get a stable discretization -> Berger-Oliger
  - increases complexity
  - load balancing becomes more difficult!
  - outer grids typically dominate memory requirements, innermost grids determine speed.

Berger-Oliger mesh refinement

- Adaptive mesh refinement significantly increases the complexity of a numerical code!

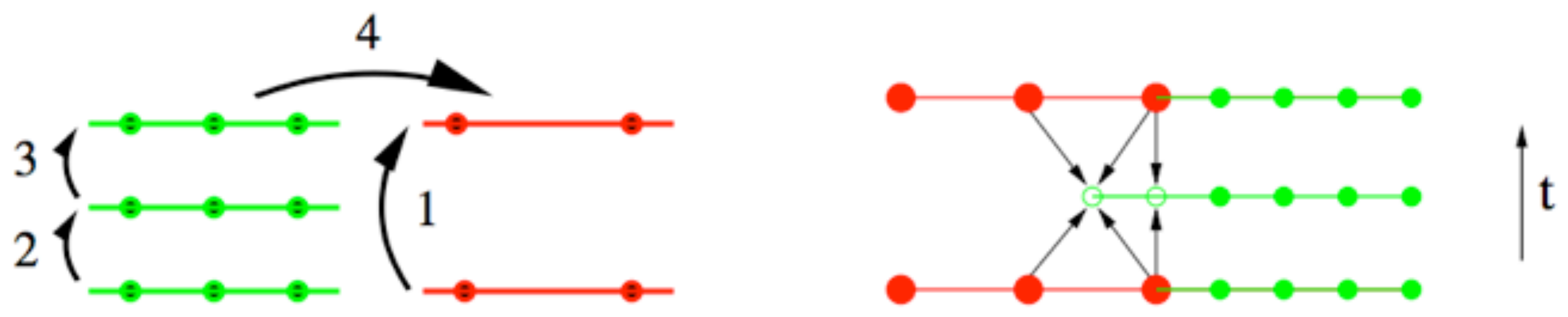


Figure 1. Berger-Oliger time stepping details, showing a coarse and a fine grid, with time advancing upwards. **Left:** Time stepping algorithm. First the coarse grid takes a large time step, then the refined grid takes two smaller steps. The fine grid solution is then injected into the coarse grid where the grids overlap. **Right:** Fine grid boundary conditions. The boundary points of the refined grids are filled via interpolation. This may require interpolation in space and in time.

- buffer points (ghost zones) are required for stability and accuracy
- with enough buffer points, one can avoid time interpolation (2nd order)
- restriction: from fine to coarse grid
- prolongation: from coarse to fine grid

Alternatives to finite differencing: Spectral methods

- Higher order finite difference can give us better accuracy, what is the highest order we can use in a grid?
- We can always use all the points in the grid, i.e. use higher and higher order as we refine the grid.
- Corresponds to representing our solution not piecewise by local polynomials (Taylor series), but global basis functions defined over the whole grid.
- Systematic approach: Search for solutions u_N in a finite-dimensional sub-space P_N of some Hilbert space W, to the PDE L u = F, L some diff. op.
- Residual (error): **R = L u_N - F**
- Expansion functions = trial functions : basis of P_N : (ϕ_1, ϕ_2,N.., ϕ_N)
- u is expanded in terms of the trial functions: $$u_N = \sum_{n=1} \tilde{u}_n \phi_n(x)$$
- Test functions : family of functions (ψ_1, ψ_2, ..., ψ_N) to define the smallness of the residual R, by means of the Hilbert space scalar product:

$$\forall n : (\psi_n, R) = 0$$

Classification of methods

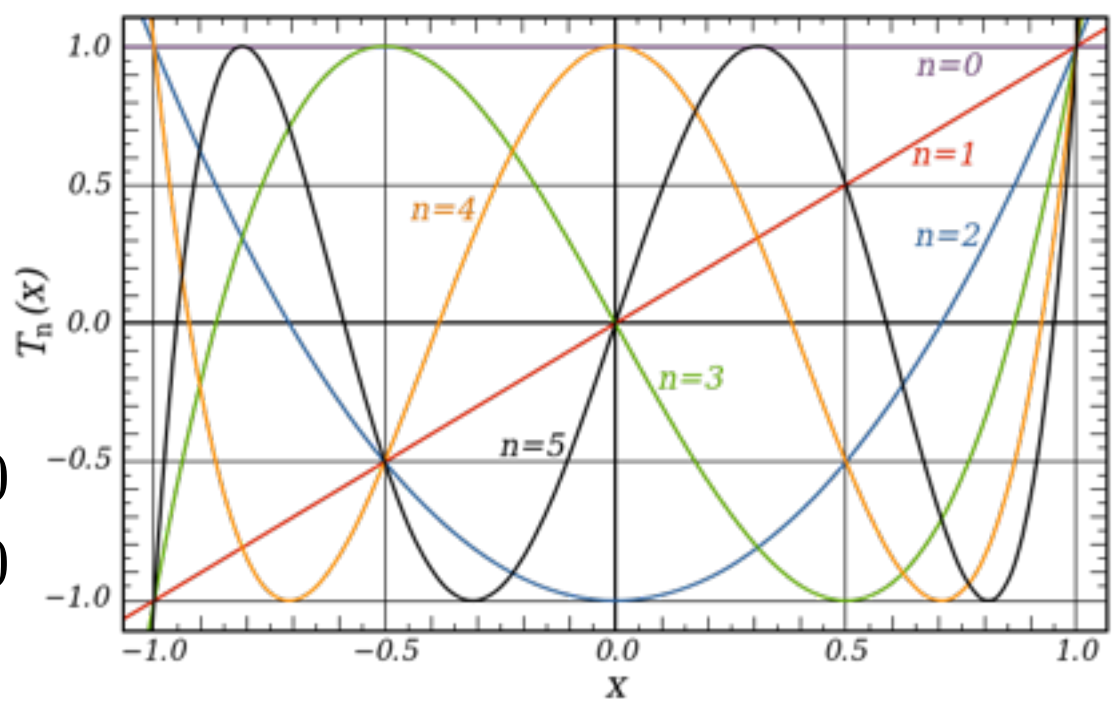
- <u>**According to the trial functions ϕ_n**</u>:
- **Finite difference**: trial functions = overlapping local polynomials (low order)
- **Finite elemen**t: trial functions = local smooth functions (polynomial of fixed degree which are non-zero only on subdomains of U)
- **Spectral**: trial functions = complete basis of global smooth functions (example: Fourier series)

$$u_N = \sum_{n=1}^{N} \tilde{u}_n \phi_n(x) = \sum_{n=1}^{N} \tilde{u}_n e^{i\omega_n x}$$

- For non-periodic functions, it has become standard to use Chebyshev polynomials, orthogonal polynomials $T_n(x)$:

$$T_n(x) = \cos(n \arccos x)$$
$$T_n(\cos\vartheta) = \cos(n\vartheta)$$

$$\int_{-1}^{1} T_n(x)\, T_m(x) \frac{dx}{\sqrt{1-x^2}} = \begin{cases} 0 & : \quad n \neq m \\ \pi & : \quad n = m = 0 \\ \frac{\pi}{2} & : \quad n = m \neq 0 \end{cases}$$

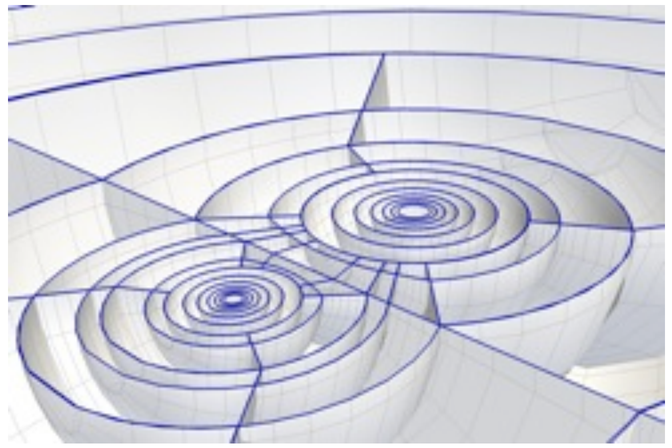


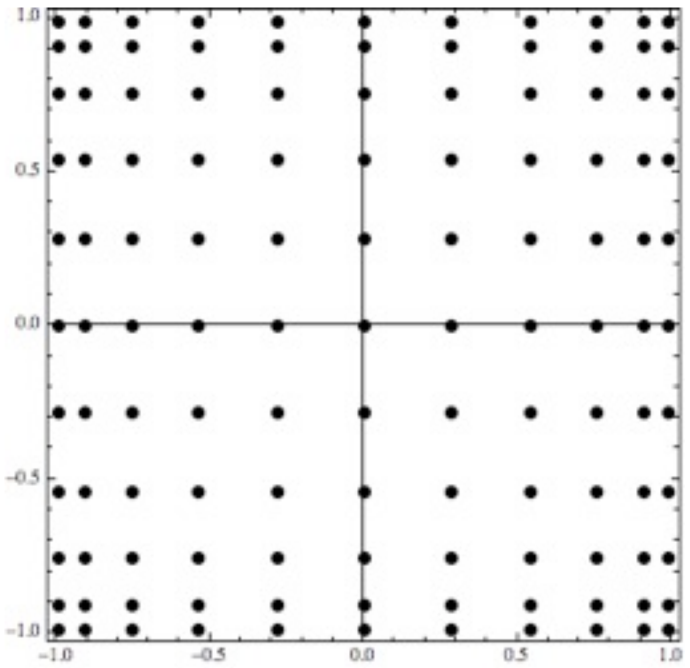
Classification of methods

- According to the test functions ψ_n:
- **collocation or pseudospectral method**: test functions = delta functions at special points, called collocation points: $\psi_n = \delta(x - x_n)$.
  - Idea: use both the spectral and the "physical space" grid, compute derivatives in the spectral representation, and source terms by evaluating the u_N on the collocation points.
  - e.g. multiplication of $u_N \times u_N$ is expensive in Fourier space, but cheap in physical space.
- **Galerkin method**: test functions = trial functions: $\psi_n = \phi_n$ and each ϕ_n satisfies the boundary conditions.
- **tau method**: (Lanczos 1938) test functions = (most of) trial functions: $\psi_n = \phi_n$, but the ϕ_n do not satisfy the boundary conditions; the latter are enforced by an additional set of equations.

Features of pseudo-spectral methods

- Exponential convergence for smooth functions.
    - + “global” method: BH excision preferred over punctures - keep clear of singularities
    - ongoing work to combine spectral+FD ideas for punctures
- Need to cover computational domain by orthogonal basis functions -> want simple geometrical domain.
- Cover more complicated geometries by multiple domains (e.g. when 2 BHs are cut out)
- Chebyshev grid is more dense at the edges -> CFL requires small Δt

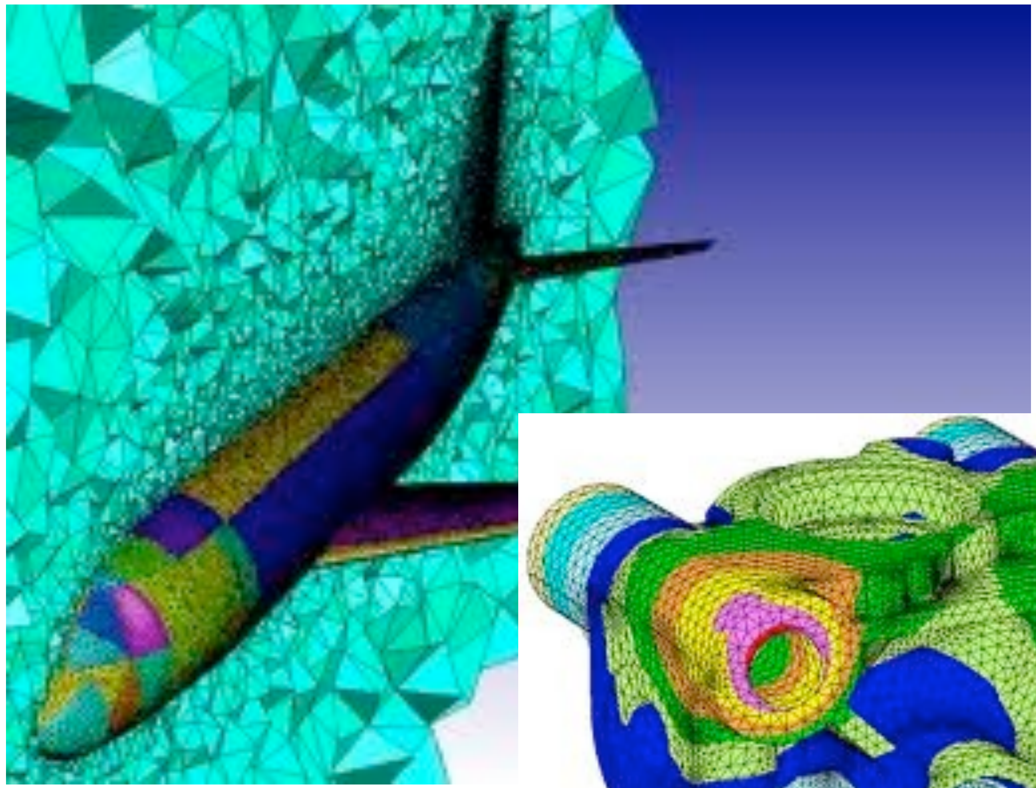
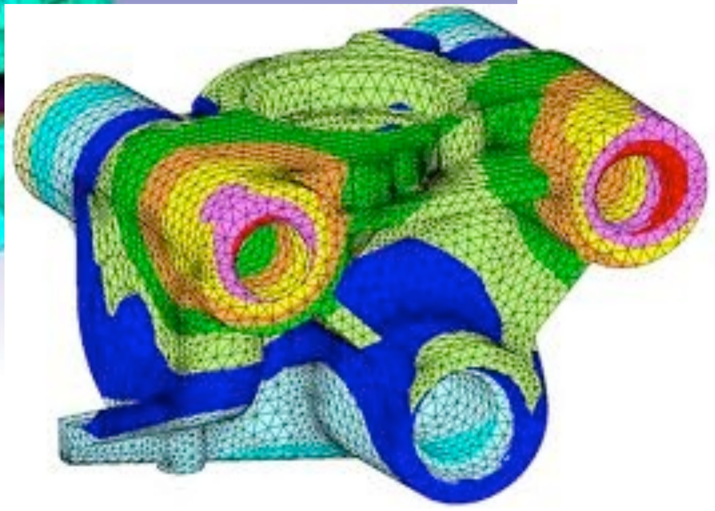


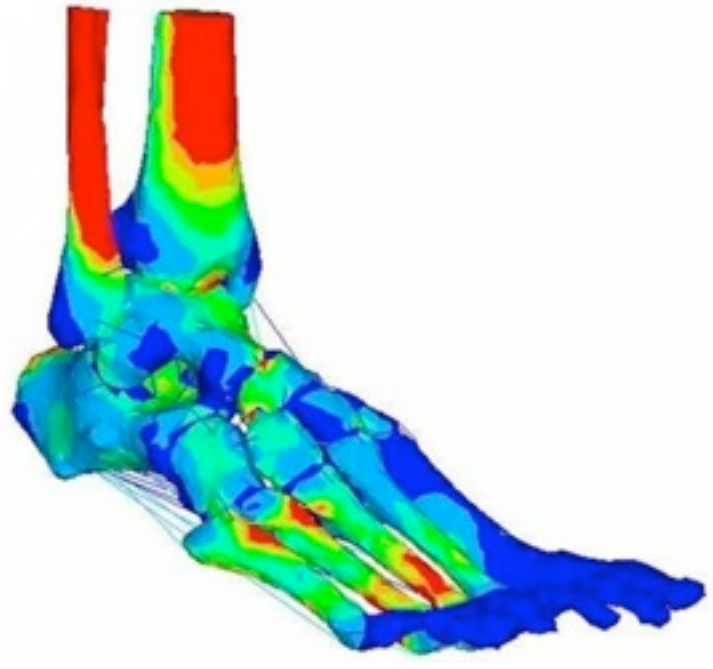
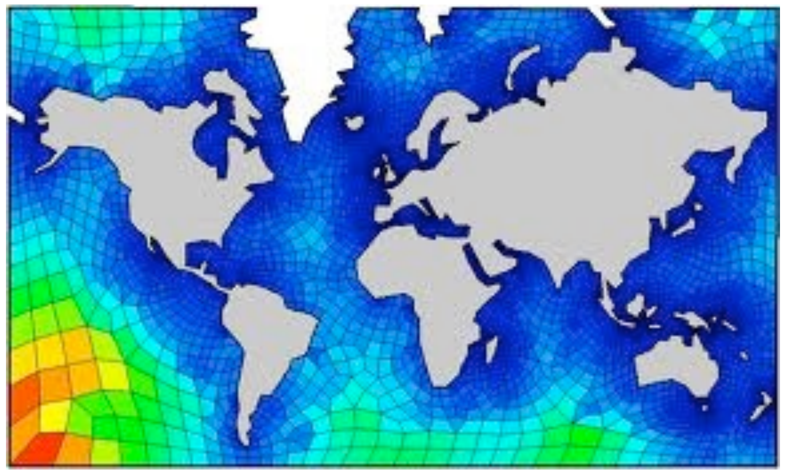


- Dissipation is applied by throwing away higher frequencies.
- Best accuracy at decreased robustness, higher implementation complexity.
- Ideally suited and routinely applied for elliptic problems in NR.

Features of Finite Elements methods

- Tailored for adaption to complicated geometries.
    - perfect for engineering
    - rarely used in NR
    - many free and commercial software packages.
    - combination with spectral methods: spectral elements.

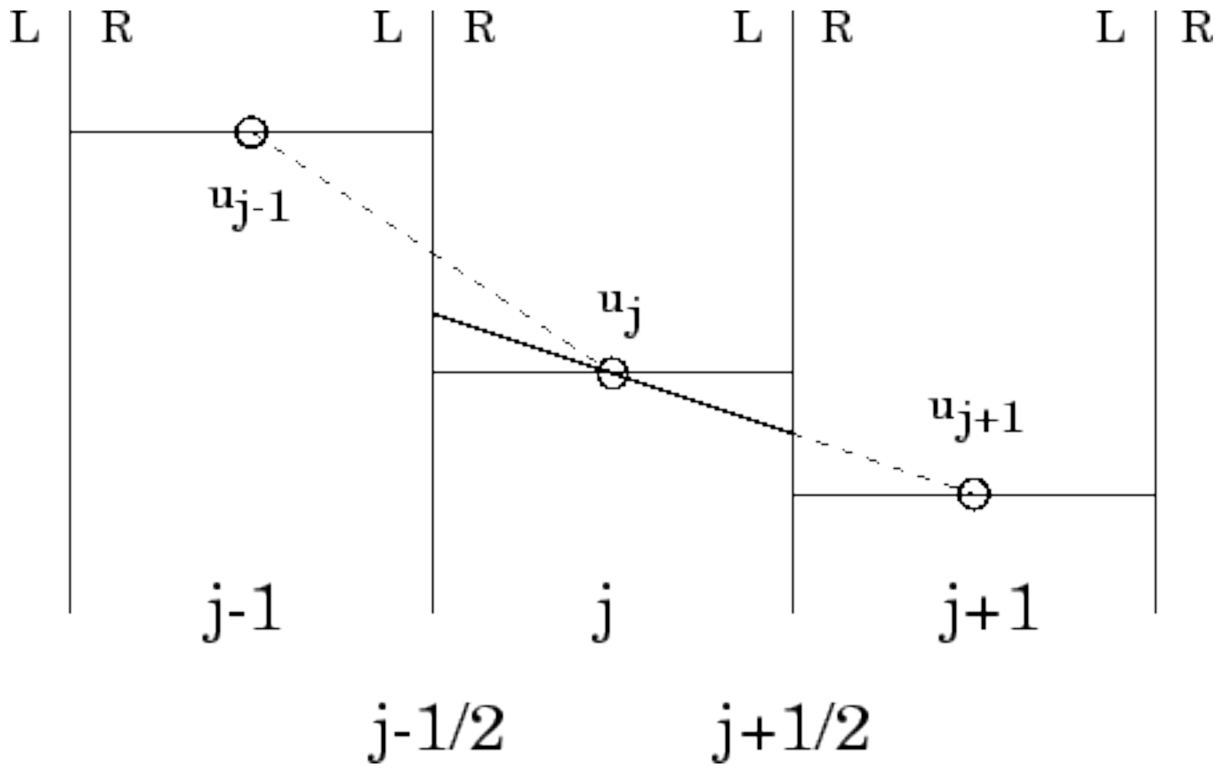
Alternatives to FD: Finite Volume Methods

- Calculate values at discrete places on a meshed geometry.
- "Finite volume": Integrate PDE over volume surrounding each node point on a mesh, and convert divergence terms to surface integrals, evaluate as fluxes at the surfaces of each finite volume.
- Flux entering a given volume is identical to that leaving the adjacent volume, these methods are “conservative”.
- Consider conservation problem: $\partial_t u + \nabla \cdot f(u) = 0$

$$\int_{v_i} \partial_t u \, dv + \int_{v_i} \nabla \cdot f(u) \, dv = 0$$

$$v_i \partial_t \, \bar{u} + \int_{S_i} f(u) \cdot n \, dS = 0$$

$$\partial_t \, \bar{u} + \frac{1}{v_i} \int_{S_i} f(u) \cdot n \, dS = 0$$



alternative hardware: GP-GPU

- General-purpose computing on graphics processing units
- Graphics cards (games ...) are very fast at doing floating point operations!
- GPU providing a functionally complete set of operations performed on arbitrary bits -> GP-GPU.
- Hundreds to thousands of cores/GPU -> “Desktop supercomputer”
- Standard hardware for scientific purposes: NVIDIA cards with Fermi architecture, fast hardware double precision.
- Dominant general-purpose GPU computing languages: OpenCL, CUDA (NVIDIA, proprietary)Stream processing paradigm: Given a set of data (a stream), a series of operations (kernel functions) is applied to each element in the stream.
- Limited memory capabilities: few GByte / card (~ MByte / compute core)
- Harder to program, but can already be used from within Matlab, Mathematica, ...
- Very successful for problems with smaller memory requirements, e.g. Teukolsky equation for BH perturbations.

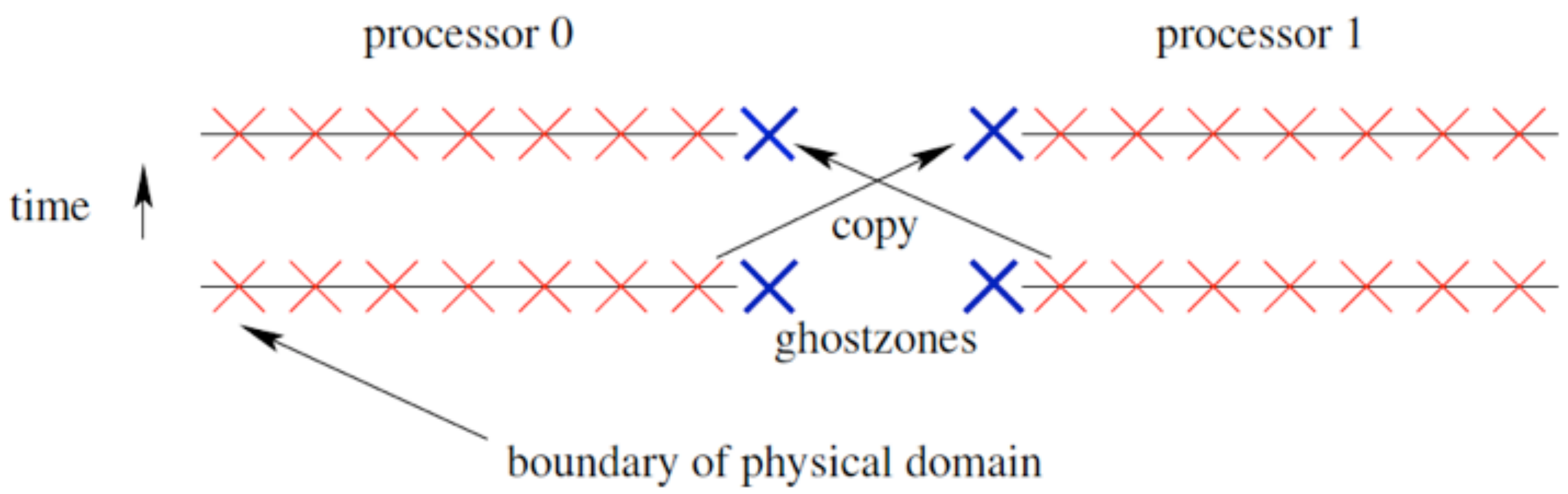
Cost & error II

- (Theoretical) computational cost scales with grid size, 3D: $\propto \Delta x^{-3} \Delta t^{-1}$
  - x 2 resolution -> x 16 computational cost
  - Real world software and hardware may behave in a more complicated way!
- To estimate the error, we need to understand convergence. n-th accurate:
  $$X(\Delta x) = X_0 + e \Delta x^n + O(\Delta x^{n+1})$$
  - 3 resolutions determine X_0, e, n
    - logic: is n consistent with what I think my algorithm is?
      Yes -> estimate X_0, estimate error e.g. as X_0 - X(best resolution).
    - break-even for 3D - n=4: x 2 resolution, 16 computational cost, error/16
    - typical n for binary black holes: 6-10.
    - spectral code: exponential convergence (SpEC, standard initial data solvers), more accurate, harder to verify.
    - mixed order schemes are common, and can lead to subtle problems, e.g. RK4 + 6th order finite differencing + ...

Cost & error: some numbers

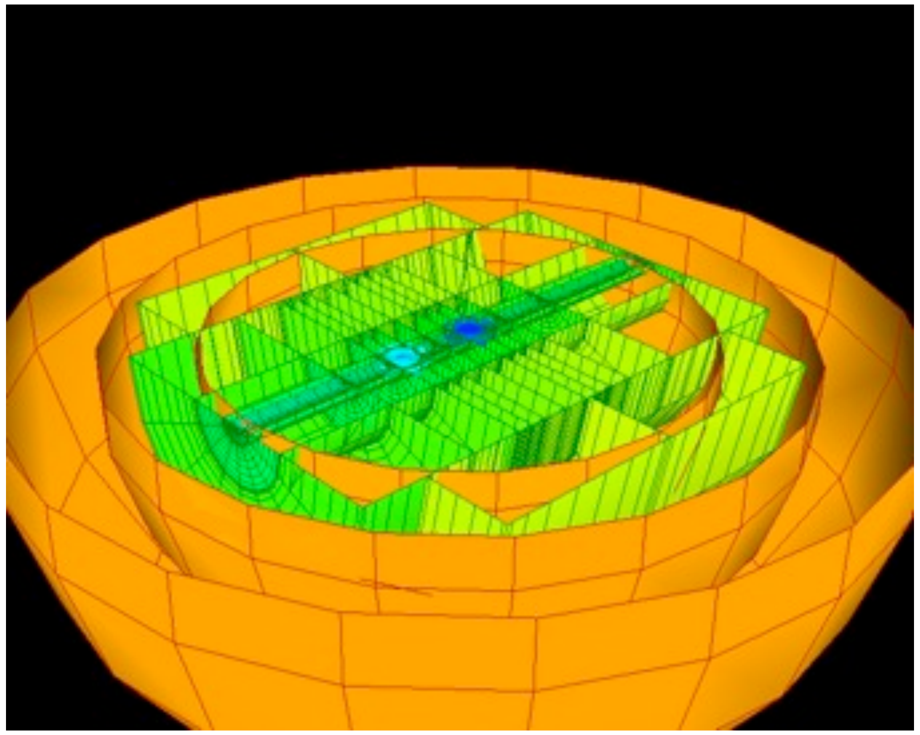
- Typical memory requirements for 3D BBH codes:
    - ~ 100 grid functions (e.g. harmonic first order, BSSN: 21 GFs, 4 time levels)
    - ≥ 100^3 boxes in 10 refinement levels: ~85 GByte.
    - Typical memory/core in a compute cluster: 2 GByte
    - standard processors ~ 12 cores => do need clusters for 3D physics
    - use domain decomposition: split computational domain over several blocks, 1 block/core. Communicate between blocks after each time step.
- Typical time requirements for 3D BBH codes:
    - (100 GFs) x (4 time levels) x (10 evaluations/GF/time level) x (100^3 points) = 4×10^9 operations/time step => > 1 second/step on 3GHz core.
    - 100 points to cover black hole region => $\Delta x \sim \Delta t = 0.01$ M
    - 100 000 iterations for 1000 M ~ 1 day.
    - cost increases for 2 BHs, 1 BH much smaller, outer regions (waves!), ...
    - standard processors ~ 12 cores => do need clusters for 3D physics

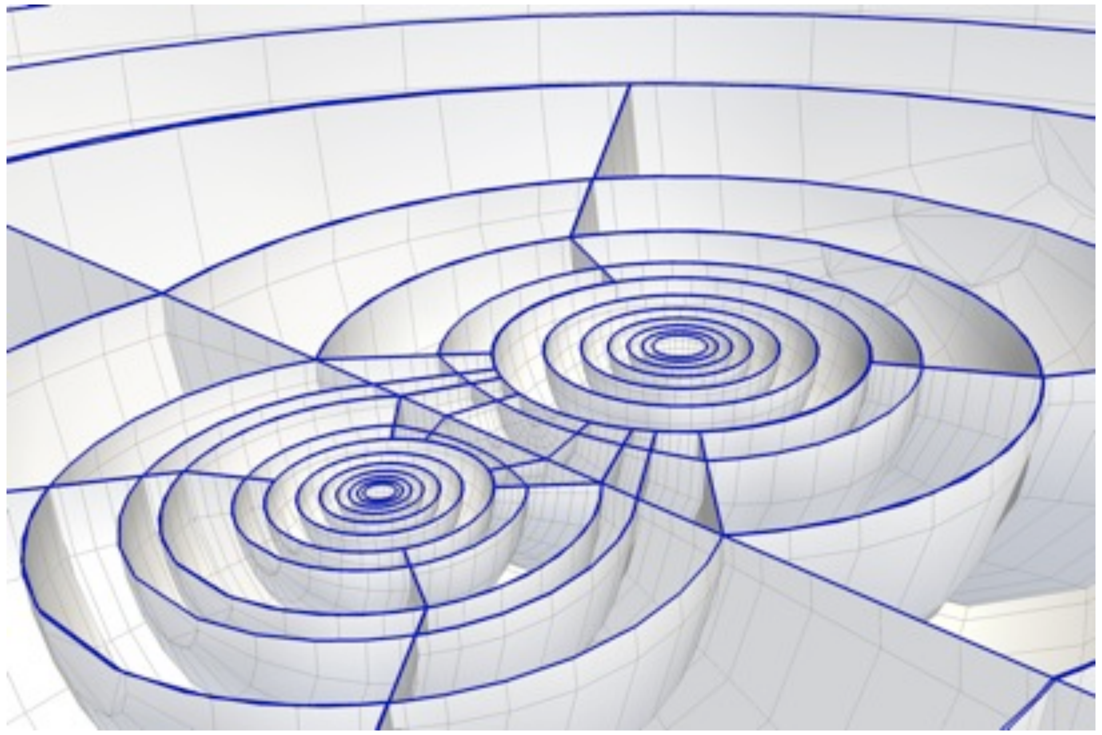
Domain decomposition with ghost zones



- Break up computational domains, e.g. into logical cubes:
  - enlarge each cube by “ghost zones” depending on size of finite difference stencil: e.g. 1 GZ for 2nd order centered, 2 GZ for 4th order centered, 4 GZ for sixth order upwind ...
  - Fill up ghost zones after each iteration by communicating between processors.
  - load balance: how to spread load evenly among processors when domains are complicated, time steps not equal, ...

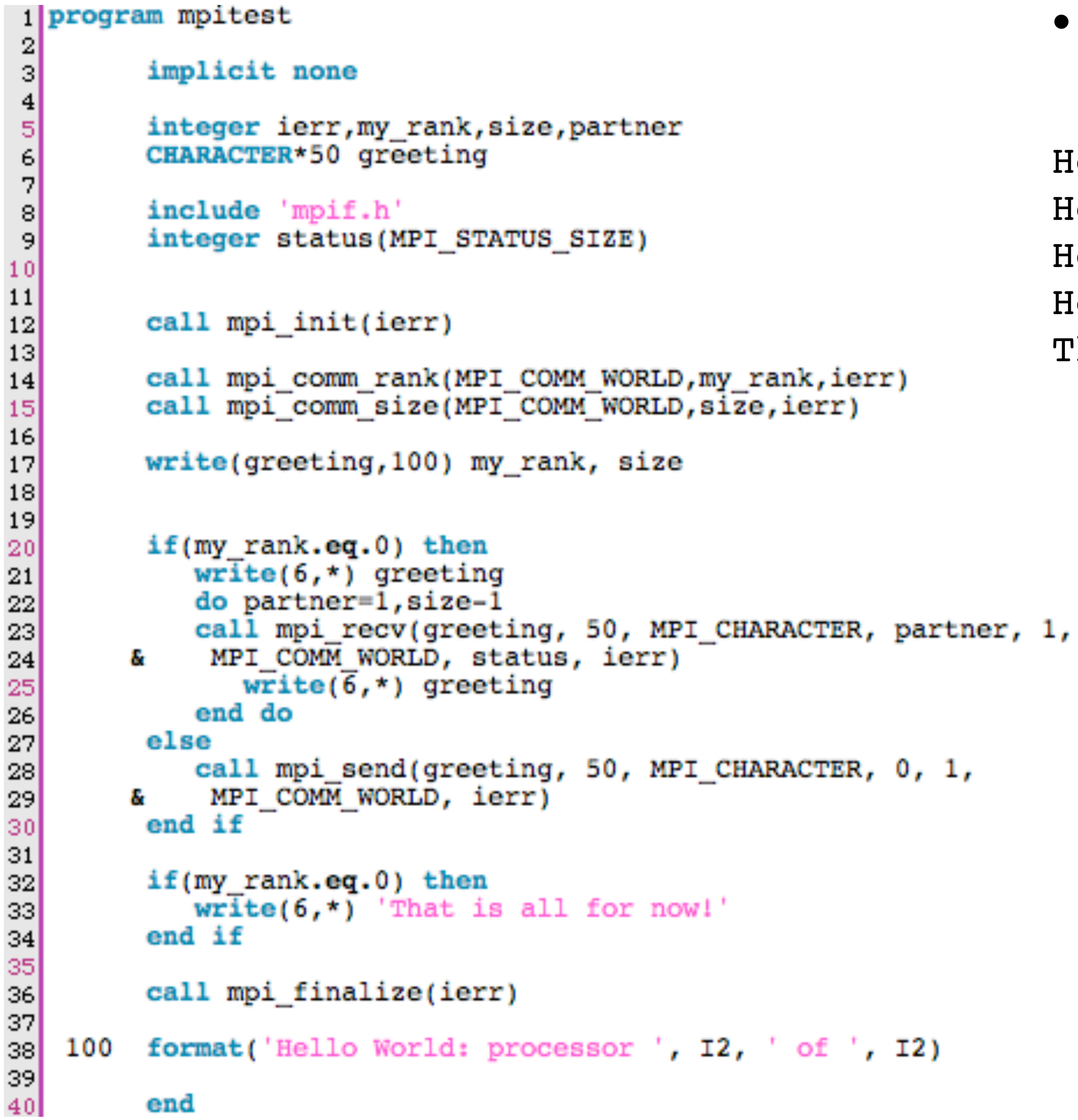
Domain Decomposition: Communication





- 2 types of algorithms:
  - overlapping grids: use interpolation
  - touching grids: can exchange characteristic information, i.e. book-keeping of the information in the hyperbolic system which travels across domain boundaries.
- 2 types of communication technology
  - Common address space: OpenMP, ...
  - Message passing: MPI (message passing interface), ...

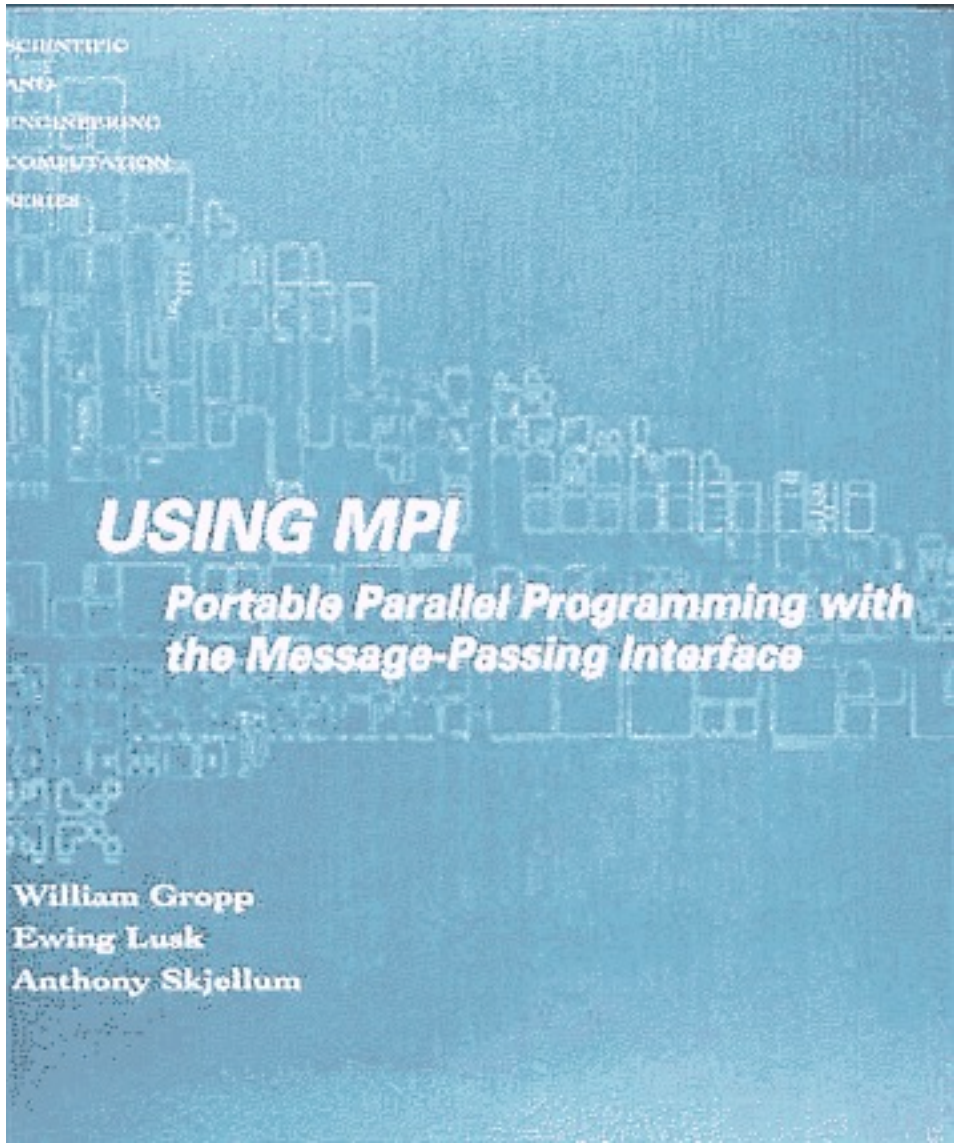
MPI



```fortran
program mpitest

      implicit none

      integer ierr,my_rank,size,partner
      CHARACTER*50 greeting

      include 'mpif.h'
      integer status(MPI_STATUS_SIZE)


      call mpi_init(ierr)

      call mpi_comm_rank(MPI_COMM_WORLD,my_rank,ierr)
      call mpi_comm_size(MPI_COMM_WORLD,size,ierr)

      write(greeting,100) my_rank, size


      if(my_rank.eq.0) then
         write(6,*) greeting
         do partner=1,size-1
         call mpi_recv(greeting, 50, MPI_CHARACTER, partner, 1,
     &    MPI_COMM_WORLD, status, ierr)
            write(6,*) greeting
         end do
      else
         call mpi_send(greeting, 50, MPI_CHARACTER, 0, 1,
     &    MPI_COMM_WORLD, ierr)
      end if

      if(my_rank.eq.0) then
         write(6,*) 'That is all for now!'
      end if

      call mpi_finalize(ierr)

 100  format('Hello World: processor ', I2, ' of ', I2)

      end
```

- Output from this code looks like:

```
Hello world: processor 0 of 4
Hello world: processor 1 of 4
Hello world: processor 2 of 4
Hello world: processor 3 of 4
That is all for now!
```

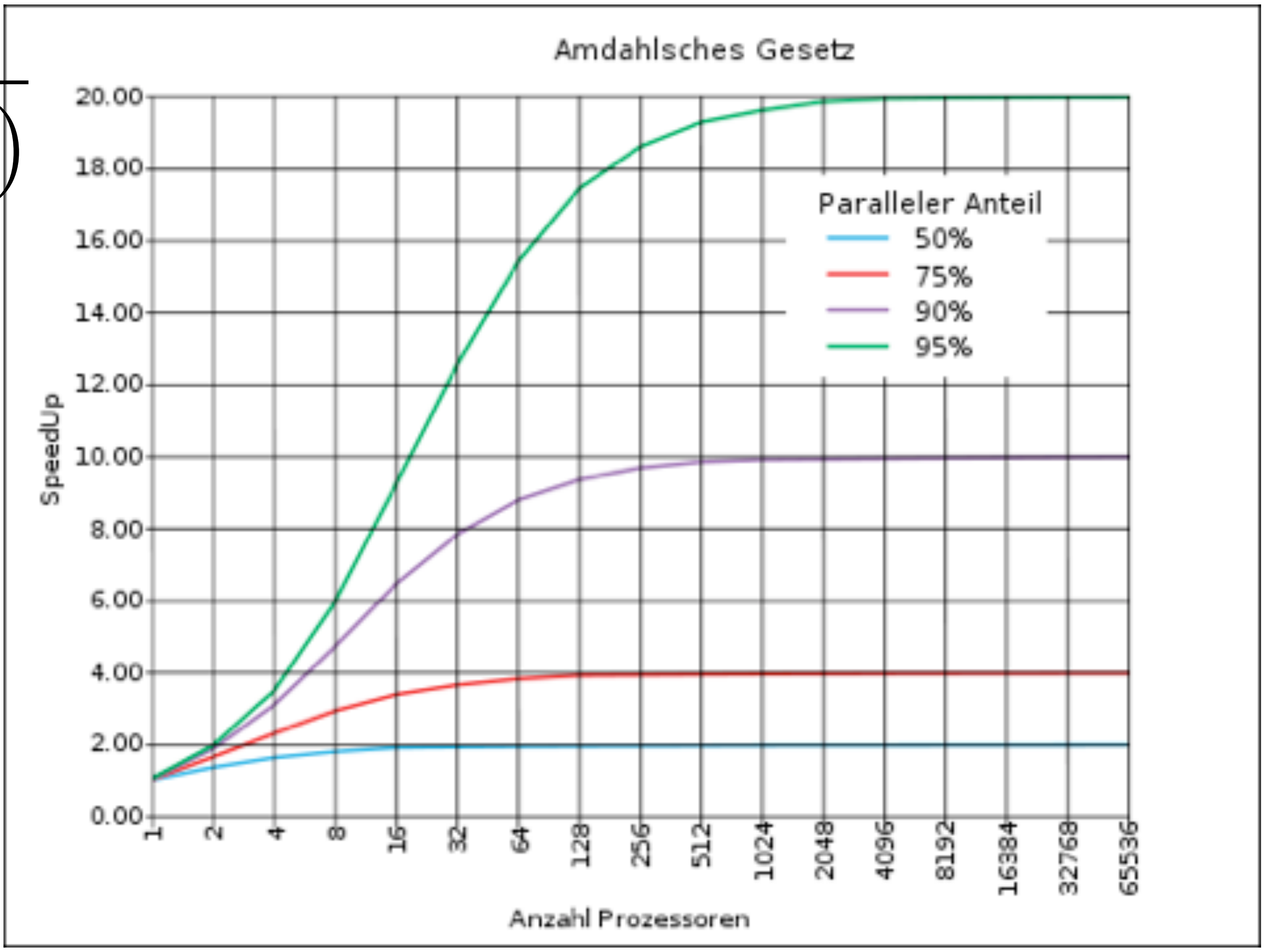

SCIENTIFIC AND ENGINEERING COMPUTATION SERIES
USING MPI
Portable Parallel Programming with the Message-Passing Interface
William Gropp
Ewing Lusk
Anthony Skjellum

Scaling

- Strong scaling: how the solution time varies with the number of processors for a fixed total problem size.
- Weak scaling: how the solution time varies with the number of processors for a fixed problem size per processor.
- Amdahl-Gunther law: (speedup S, parallel fraction of algorithm P, N cores, β coherency delay). Amdahl's law for β = 0.

$$S = \frac{1}{1 - \left(1 - \frac{1}{n}\right)p + \beta(n-1)}$$

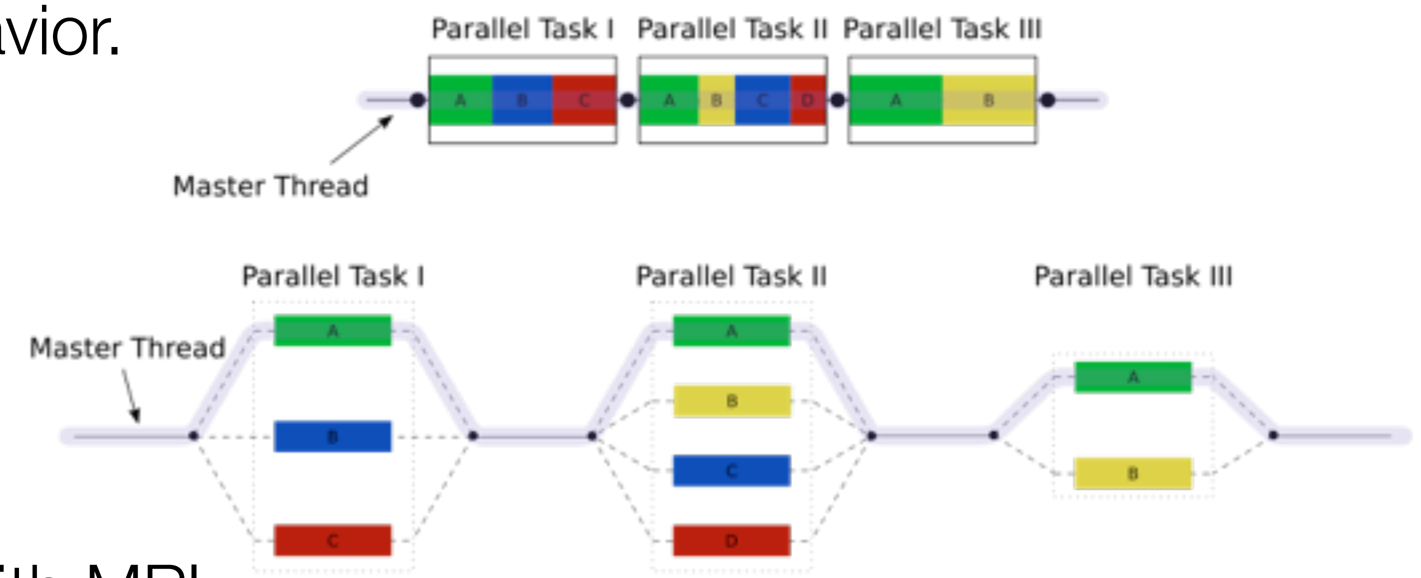
- Real life: may be more interested in keeping execution time constant and increase problem size with number of processors.
- Throughput can be more important than speed! How much science can you do with a given amount of resources?



OpenMP: multithreading

- OpenMP (Open Multiprocessing) is an API that supports multi-platform shared memory multiprocessing programming in C, C++, andFortran.

- It consists of a set of compiler directives, library routines, and environment variables that influence run-time behavior.



- Typically combine OpenMP with MPI:
  - parallelization on each "node" with Open MP
  - communication between nodes with MPI

```
#include <stdio.h>

int main(void)
{
  #pragma omp parallel
    printf("Hello, world.\n");
  return 0;
}
```

```
int main(int argc, char *argv[]) {
    const int N = 100000;
    int i, a[N];

    #pragma omp parallel for
    for (i = 0; i < N; i++)
        a[i] = 2 * i;

    return 0;
}
```

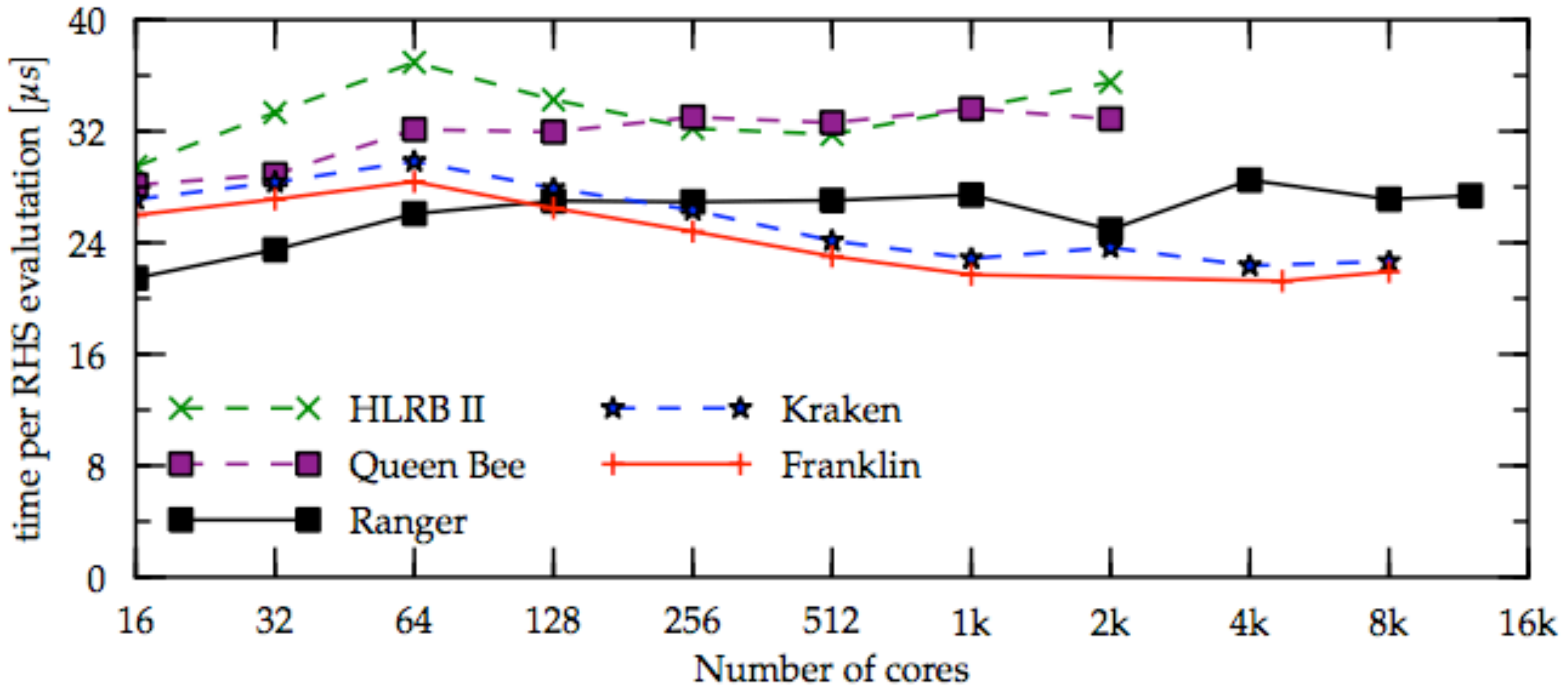


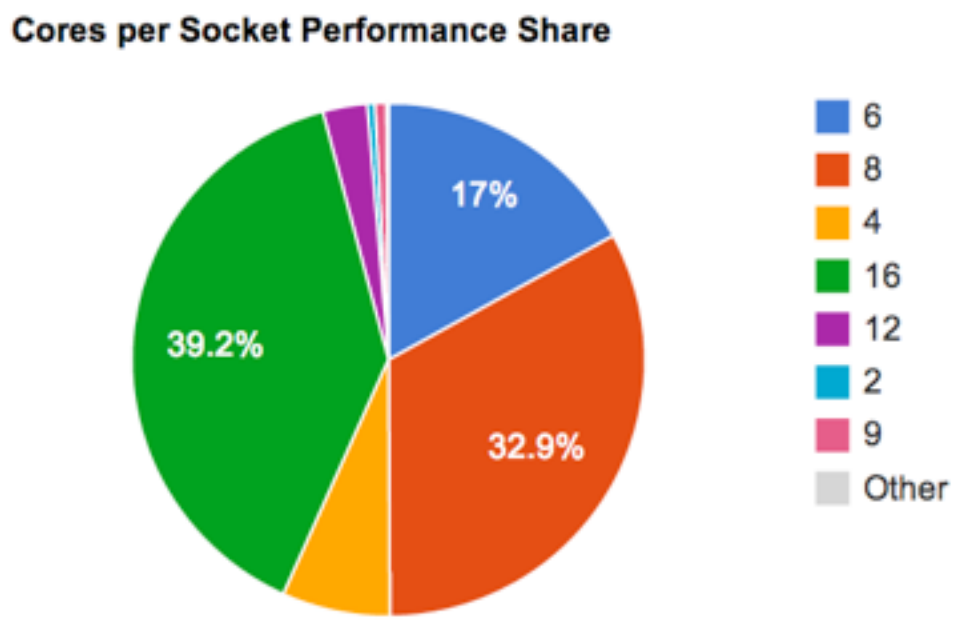
Figure 2. Results from weak scaling tests evolving the Einstein equations on a mesh refinement grid structure with nine levels. This shows the time required per grid point, where smaller numbers are better (the ideal scaling is a horizontal line). This demonstrates excellent scalability to up to more than 10,000 cores. Including hydrodynamics approximately doubles calculation times without negatively influencing scalability.

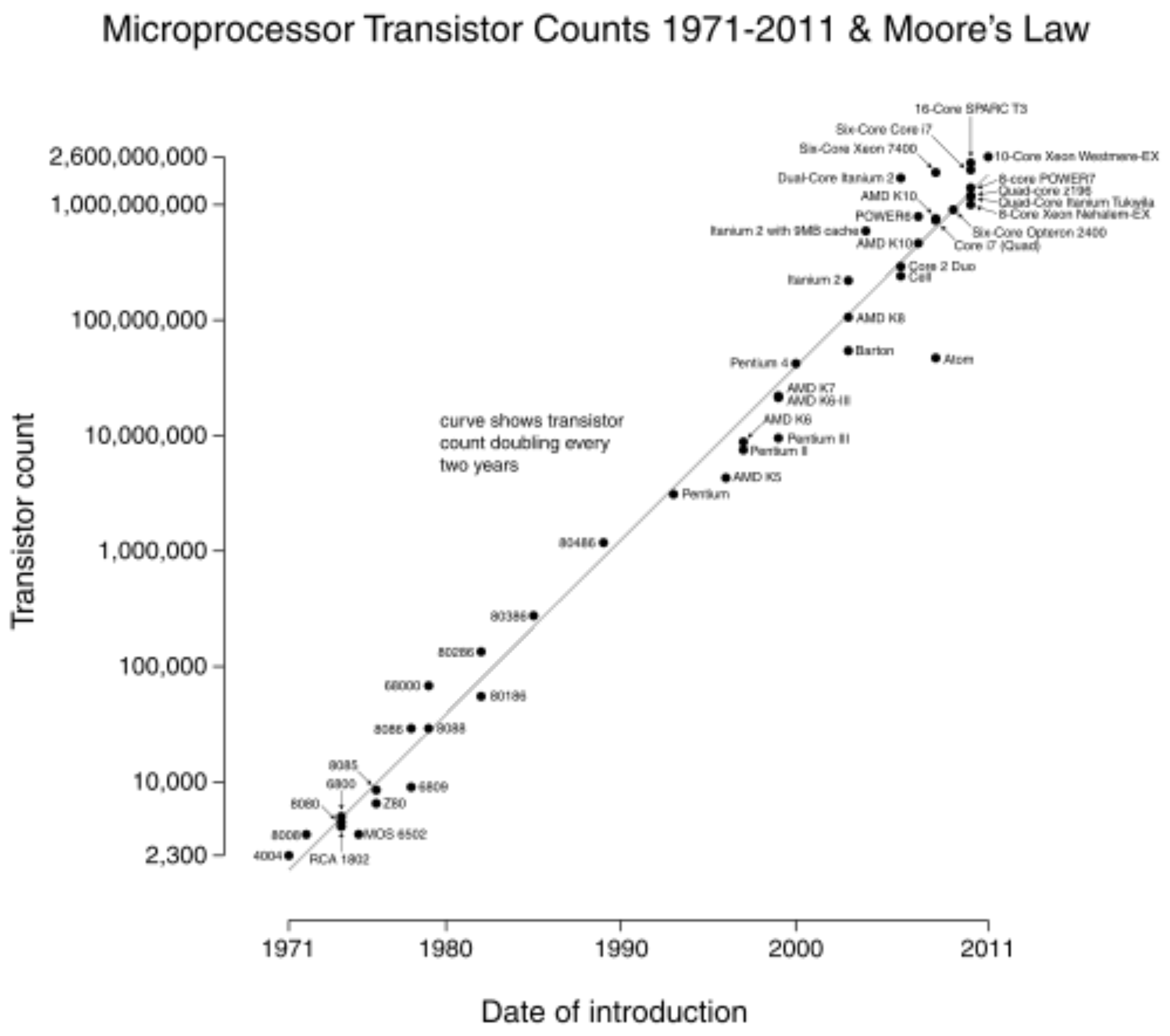
Moore’s law

- Processor performance doubles every ~ 18 months

- Modern CPUs have increasingly more cores!

- Modern supercomputers have more and more CPUs!


Cores per Socket Performance Share
Other


Microprocessor Transistor Counts 1971-2011 & Moore’s Law
Transistor count
Date of introduction
curve shows transistor count doubling every two years
16-Core SPARC T3
Six-Core Core i7
Six-Core Xeon 7400
10-Core Xeon Westmere-EX
Dual-Core Itanium 2
8-core POWER7
Quad-core z196
Quad-Core Itanium Tukwila
8-Core Xeon Nehalem-EX
AMD K10
POWER6
Itanium 2 with 9MB cache
AMD K10
Six-Core Opteron 2400
Core i7 (Quad)
Core 2 Duo
Cell
Itanium 2
AMD K8
Barton
Atom
Pentium 4
AMD K7
AMD K6-III
AMD K6
Pentium III
Pentium II
AMD K5
Pentium
Z80
MOS 6502
RCA 1802