Computational Seismology

Lecture 2: A Discrete World

January 27, 2021

University of Toronto

TABLE OF CONTENTS

Introduction

Discretization

1. To solve the WE by numerical methods: discretization of the problem (Earth model, wavefield, etc).
2. How to discretize: dimensionality (1D/2D/3D)? mesh generation? various geometries & different coordinate systems? Parallel computing?

Classification of PDEs

1D Acoustic Wave Equation (smoothly varying material properties)

$$p_{tt} - c^2 p_{xx} = 0 \tag{1}$$

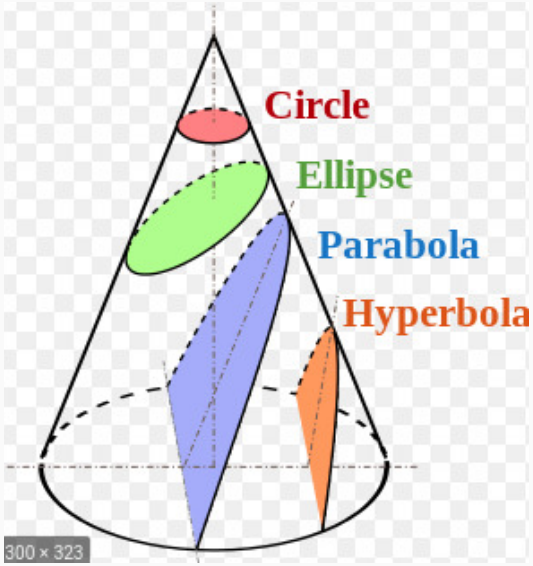
- Linear PDE: the PDE can be expressed as linear combinations of $p(x,t)$ and its spatial/temporal derivatives with coefficients independent of p (but not necessarily x) $\Rightarrow$ sometimes analytical solutions.
- *Second-order* PDE: highest-order derivatives

Classification of PDEs

Classification based on equivalent 'conic sections', or quadratic equations

$$Ap_{xx}+Bp_{xt}+Cp_{tt}+Dp_x+Ep_t+F=0 \rightarrow Ax^2+Bxt+Ctt+Dx+Et+F=0$$

$$\begin{cases} B^2-4AC=0 & \text{parabolic} \\ B^2-4AC<0 & \text{elliptic} \\ B^2-4AC>0 & \text{hyperbolic} \end{cases} \quad (2)$$



Wave equation

The wave equation is a classic **hyperbolic PDE**. The solutions to hyperbolic PDEs are **wavelike** and disturbances travel with finite propagation speeds.

In contrast, for elliptic and parabolic problems, perturbations of initial conditions or boundaries have an immediate effect everywhere.

Semi-discrete scheme

The separation of space-time derivatives

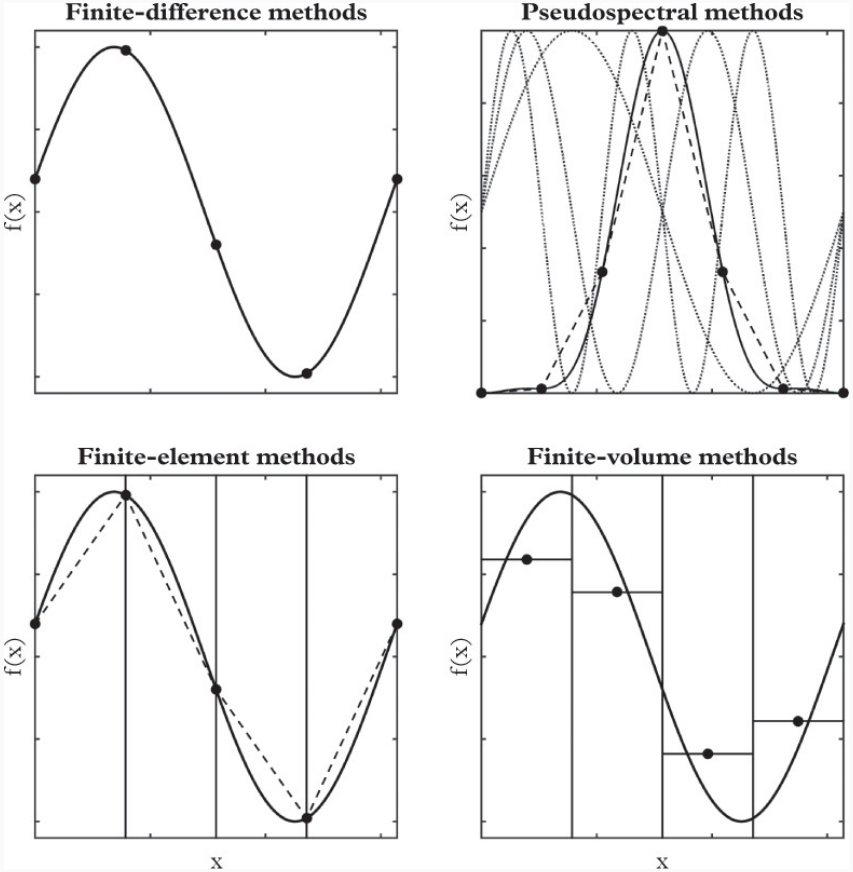
$$\partial_t^2 p(x,t) = L(p,t) \rightarrow L(p,t) = c^2(x)\partial_x^2 p(x,t) \tag{3}$$

1. This implies a scheme that advances in discrete time steps, and for every time step, displacement field can be updated through property functions such as $c(x)$ and the (discretized) spatial derivatives of the field.
2. different numerical schemes differ in how the RHS is discretized and how model is updated at each time step. The 'advance' in time is always solved by FD/discrete-integration type of scheme (Newmark, R-K).

Dicretization: grid-point vs series expansion

Two basic strategies for spatial discretization:

1. The grid-point method: approximates an arbitrary function $f(x)$ at a discrete set of points (FD)



Discretization by basis functions

2. The series-expansion method: approximate a function by a sum over a set of basis functions (local vs. global)

- Field interpolation: FS–pseudo-spectral
- finite-element method: space is divided into elements; within an element, field is approximated by polynomial function and continuous across element boundaries (linear, quadatic, Lagrange poly.)
- finite-volume method: field can be discontinuous across cell boundaries: define ‘flux’ between cells: finite-volume, discontinuous Galerkin method.

Dimensionality: 1D/2D/2.5D/3D

- Discretization/computation in the 3D world $u(x, y, z)$ or $u(r, \theta, \phi)$: most expensive $\rightarrow$ reduction to 1D/2D/2.5D
- started in 2D, $u(x, z)$ and $c(x, z)$, the solution is invariant in the y direction: point scatter $\rightarrow$ line scatter, point source $\rightarrow$ line source, therefore cannot be compared directly with obs.
- 2.5D: 1D radial Earth model, invariant along ϕ, source at $\theta = 0$ (Try instaseis)
- coordinate systems: cartesian, cylindrical, spherical coordinates $\rightarrow$ meshing

Dimensionality

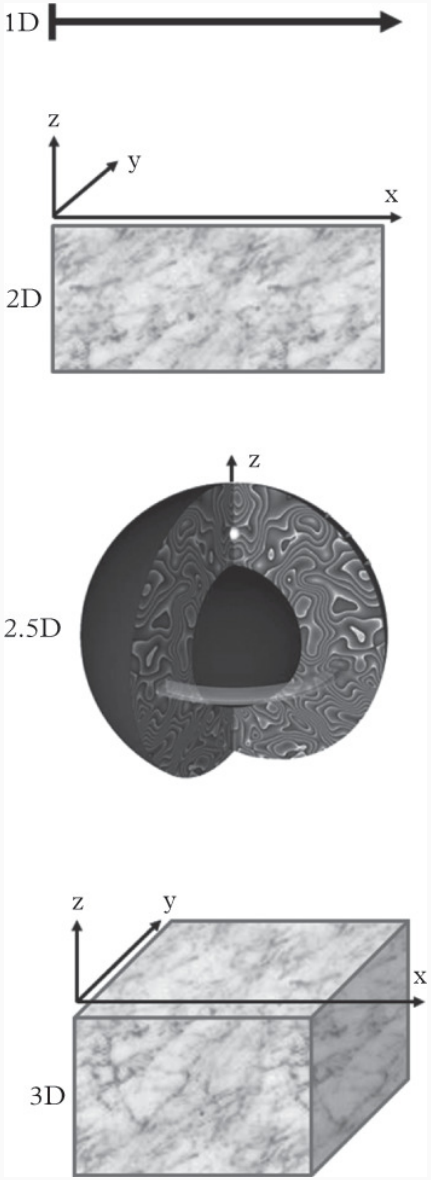
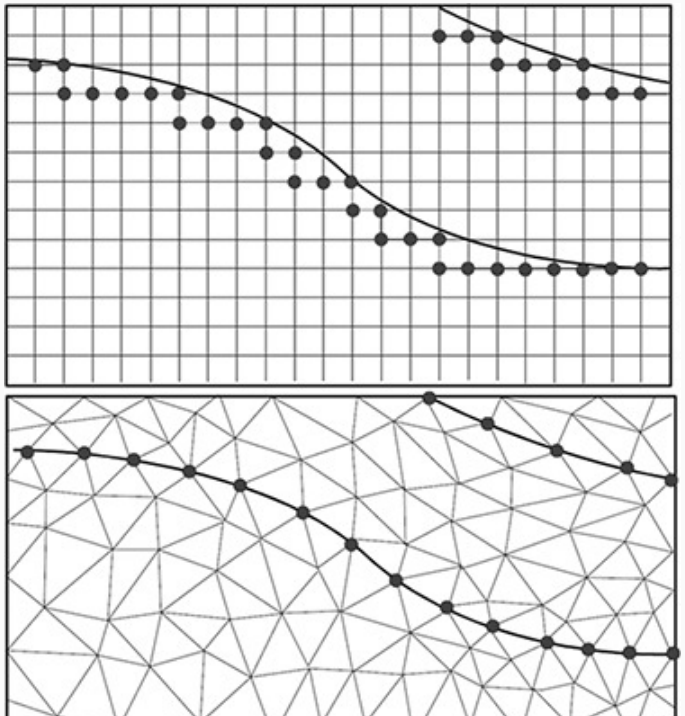


Figure 1: 1D, 2D, 2.5D, 3D

Computational Mesh

- Meshing is a field in itself and a subdomain of computational geometry
- The choice of meshing is closely related to the underlying numerical methods.
- Question: honour variations in surface topography and material interfaces? Or is the blocky representation sufficient? Difficulty in meshing/computation?



Structured (regular) mesh

Regular mesh subdivides 1D/2D/3D space by regular connectivity and can be represented by vectors in 1D and matrices in 2D and 3D, e.g., $c(x) \rightarrow c_i, i = 1, \cdots, N$.

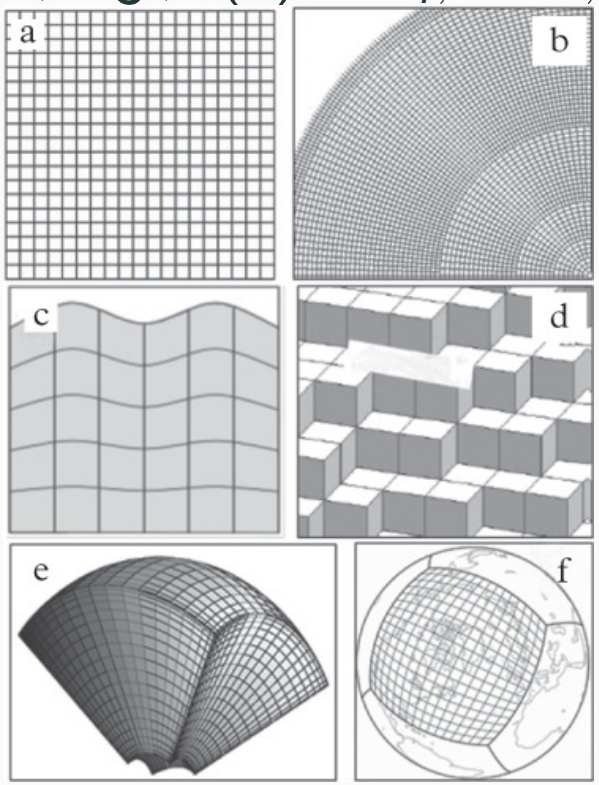
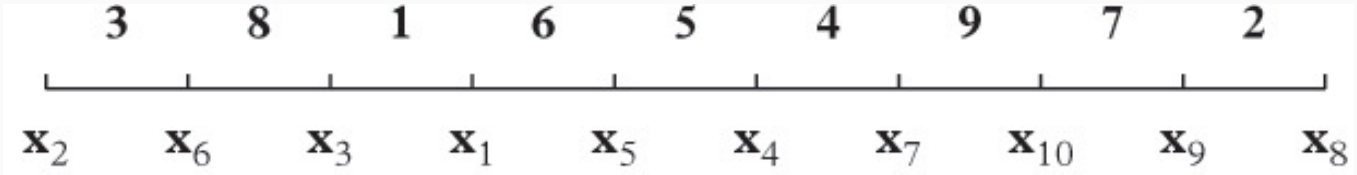


Figure 2: a: Regular, equi-spaced 2D grid in cartesian geometry; b: multi-domain regular 2D grid in spherical coordinates; c: regular, stretched grid that follows smooth surface; d: regular 3D cartesian grid with blocky topography surface. e: regular 3D grid in spherical coordinates (section) with grid points based on Chebyshev collocation points; f: regular surface grid of sphere meshed with the cubed-sphere approach.

Unstructured (irregular) mesh

They cannot be represented by vectors and matrix and require explicit definition of ‘connectivity’: list of vertices, list of elements (based on vertex #), list of neighbours (based on element #)



List of vertices	List of elements	List of neighbours
1 x_1	1 3 1	1 8 6
2 x_2	2 9 8	2 7 –
3 x_3	3 2 6	3 – 8
...	...	...
10 x_{10}	9 7 10	9 4 7

Unstructured grids in 2D and 3D

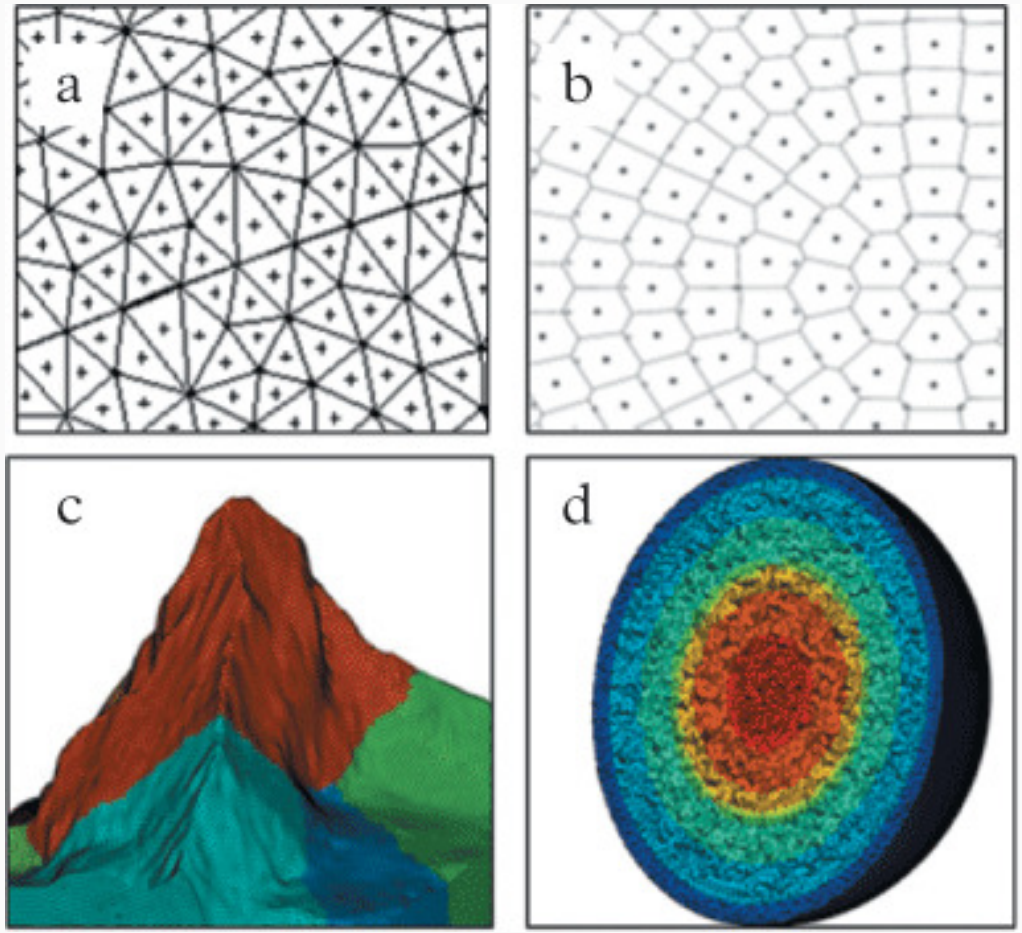
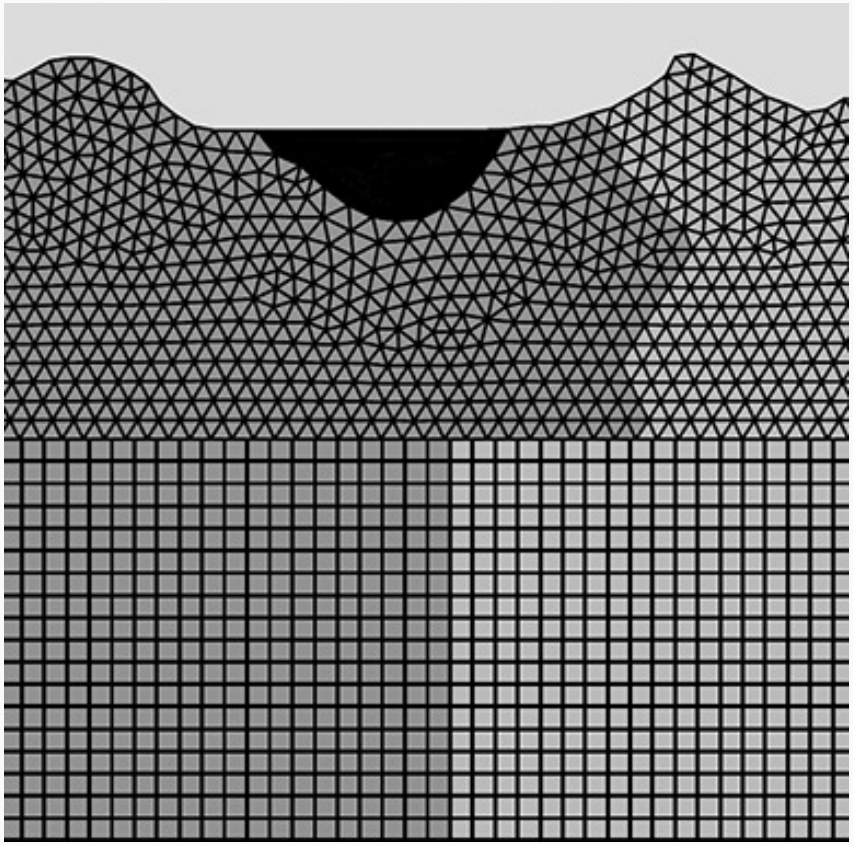


Figure 3: Unstructured grids in 2D and 3D. a: Unstructured grid based on Delauney triangulation with cross-cutting interface; b: Voronoi cells for unstructured grid; c: tetrahedral mesh for the Matterhorn (mountain in Switzerland); d: tetrahdral mesh for spherical Earth model.

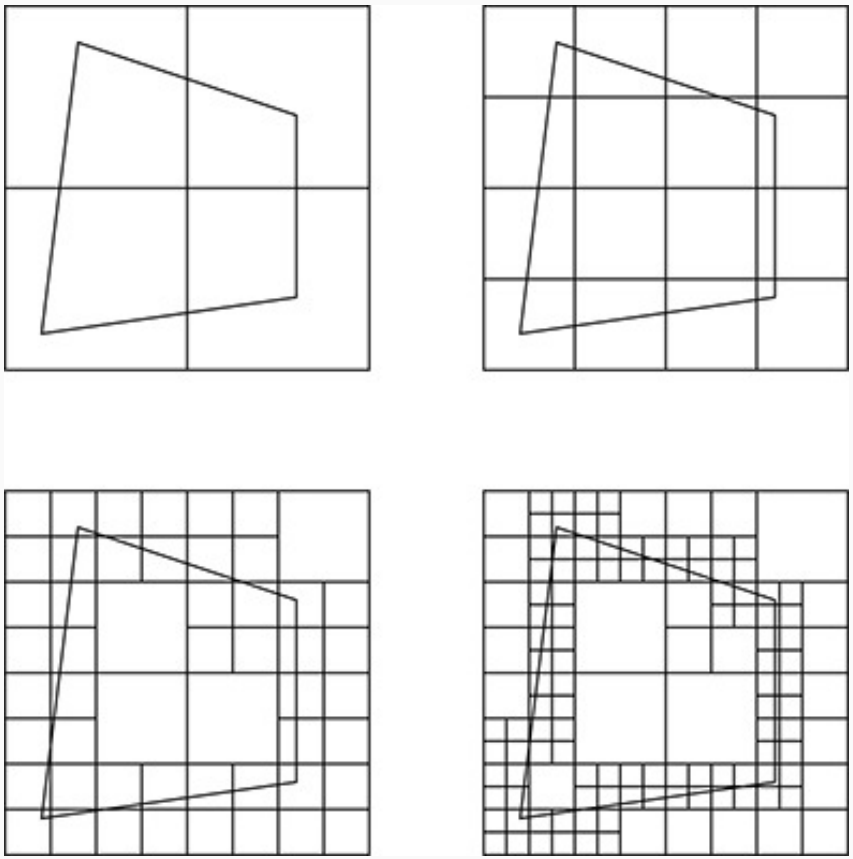
Hybrid mesh



A structured, regular grid in the lower domain with low degree of geometrical complexity is combined with an unstructured triangular grid at the top of the domain with a complicated free surface. A strong low-velocity domain (black area) is also meshed with an unstructured triangular mesh. Discontinous Galerkin method is used for wave simulation.

Adaptive-mesh refinement (AMR) and Octree

Adaptive-mesh refinement (AMR): When physical problems in large computational domains are strongly focused in space (e.g. shock waves, rupture fronts), then it might make sense to densify the grid during run time in the area where things are happening. For dynamic rupture problems, adaptive mesh refinement may be employed.



Meshing—an underestimated task

The task of meshing is often difficulty (complex geometry), time-consuming and we are not generally trained for it. Prepare a model for meshing:

- suface topography: digital elevation model (DEM), smoothed
- interface of basement map (sediment/crystalline rock interface)
- for finite-fault simulations: internal fault surfaces

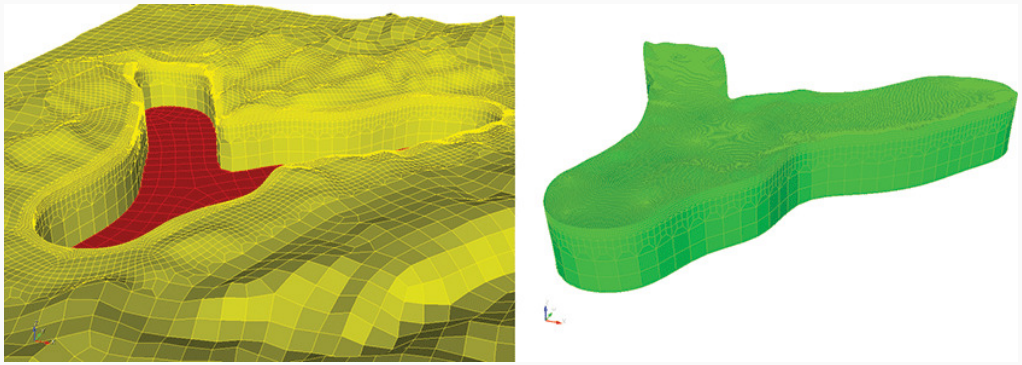
Table 3.1 Timing estimates for meshing and simulation problems

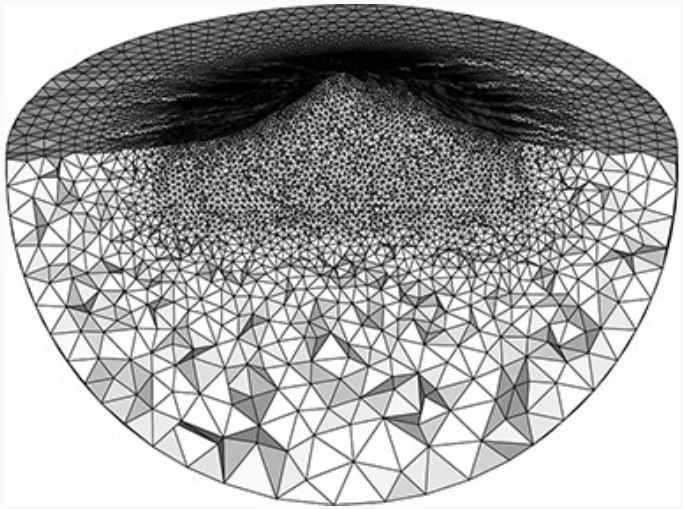
Human time	Simulation workflow	CPU time
15%	Design	0%
80% (weeks)	Geometry creation, meshing	10%
5%	Solver	90%

Source: E. Casarotti, personal communication

Steps for meshing

- geometry creation: defining surfaces bounding the mesh volume.
- mesh generation: takes the created geometry as input and subdivdes entire volume into grid cells: hexahedra (not fully automatic yet and requires manual manipulations: CUBIT/Trelis) or tetrahedra (high quality, fully automatic, but computationally more expensive)
- set geophysical parameters on the mesh through interpolation
- set boundary conditions





Left: hexahedral mesh for Gronoble Valley; Right: tetrahedral mesh for the Merapi volcano, Indonesia with refinement under the volcano summit.

Parallel Computing

Community codes for wave simulations such as *SPECFEM3D*, *SPECFEM3D_GLOBE* and *SeisSol* are often implemented in parallel.
Two computing models: serial vs parallel.

- serial: SISD (single instruction on single data) can speed up by increasing clock rate $\rightarrow$ limitation
- parallel: SIMD (single instruction on multiple data) such as matrix operations, embarrassingly parallel, GPU clusters; MIMD: multiple tasks carried out in parallel, CPU clusters.

SISD **S**ingle **I**nstruction **S**ingle **D**ata *Serial computer*	**MISD** **M**ultiple **I**nstruction **S**ingle **D**ata *Cryptographic decoding*
SIMD **S**ingle **I**nstruction **M**ultiple **D**ata *GPU cluster, CM2*	**MIMD** **M**ultiple **I**nstruction **M**ultiple **D**ata *Supercomputers PC cluster*

Wave equation: Local communications

The degree of *locality* is important for a parallel numerical algorithm.

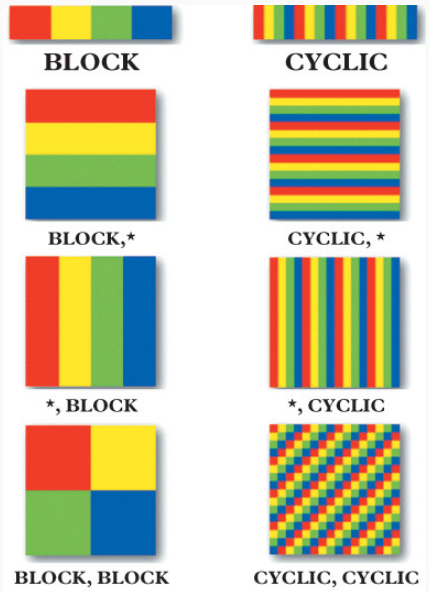
$$\partial_t^2 p = c^2 \partial_x^2 p$$

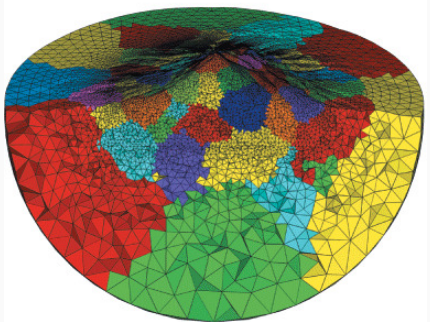
$$p(x, t + \Delta t) = \frac{c^2(x)\Delta t^2}{\Delta x^2}[p(x + \Delta x, t) - 2p(x, t) + p(x - \Delta x, t) + 2p(x, t) - p(x, t - \Delta t)]$$

The spatio-temporal interaction of elastic wave (and many other) phenomena is of a local nature $\rightarrow$ communication is only necessary between neighbouring processors $\rightarrow$ efficient parallel algorithms.

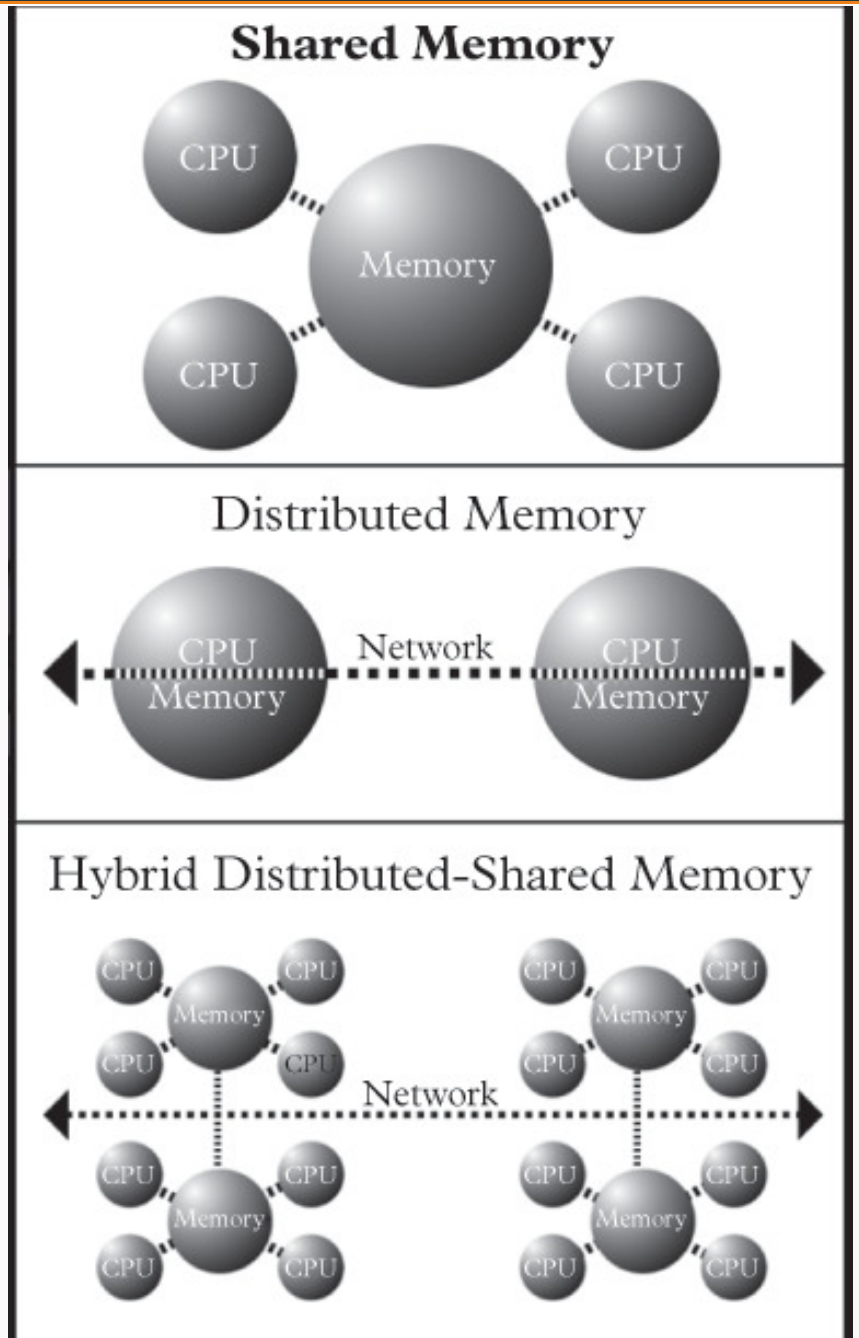
Domain decomposition

Space-dependent fields (e.g. displacements, stresses, elastic parameters) are mapped on parallel hardware by **domain decomposition** using the distributed memory concept. This means each processor only see local memory and communication occurs between processors when information needs to be exchanged. Very hard problem for unstructured mesh due to load balancing issue (SCOTCH).





Hardware: memory and CPUs



Hardware: Evolution of parallel computers

- Integrated system s(before mid 90s): homogeneous assembly of processors connected by specially designed networks
- PC cluster: (many) linux PCs + LANs
- super computers: individual computer nodes with fast network connection (e.g., the Niagara system at Scinet, Compute Canada, 2,016 nodes, each with 40 cores and 202 GB RAM, EDR infiniband network)
- GRID or cloud computing:

Hardware and software for parallel algorithms

A flexible implementation: Message-Passing Interface (MPI) which has interfaces/libraries for all major languages (Fortran, C, C++, etc)

```
! My first MPI program
program main
  use mpi
  integer  error
  integer  id
  integer  np
! Initialize MPI.
  call MPI_Init ( error )
! Get the number of processes.
  call MPI_Comm_size ( MPI_COMM_WORLD, np, error )
! Get the individual process ID.
  call MPI_Comm_rank ( MPI_COMM_WORLD, id, error )
! Print a message.
  write (*,*) "The overall number of processors is ",np
  write (*,*) "I am processor ",id
! Shut down MPI.
  call MPI_Finalize ( error )
  stop
end
```

Run with *mpirun -np 4 main.F90*

MPI output

Processor id, output in random sequence

```
The overall number of processors is 4
I am processor 0
The overall number of processors is 4
I am processor 2
The overall number of processors is 4
I am processor 3
The overall number of processors is 4
I am processor 1
```

Parallel computing: scaling and speed-up

Scalability refers to the speeding up of a program with increasing resources (e.g., cores).

$$\text{speed-up} = \frac{1}{P/n + S} \tag{4}$$

where P and S are the parallel and serial fraction of the code.

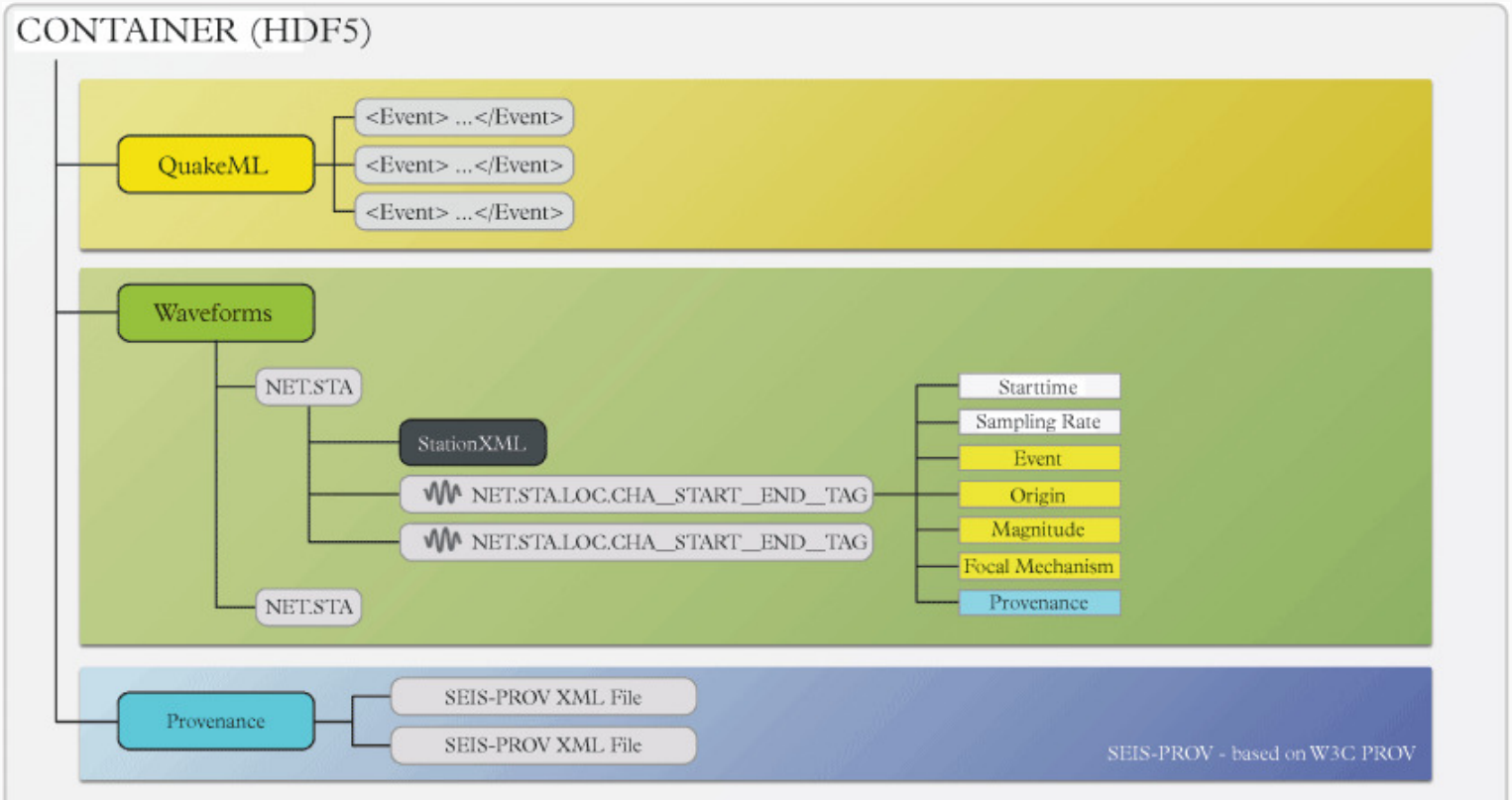
- Strong scaling: A fixed problem size, increased number of cores;
- weak scaling: increased problem size, increased number of cores
- I/O

Code optimization: When requesting resources on national or international supercomputer infrastructure you have to demonstrate that your program scales.

Parallel I/O

I/O operations, such as transfer, post-processing, etc of wavefield snapshots, can be a major bottleneck in parallel computation.

Parallel I/O with parallel storage facilities (efficient data exchange and sharing). Example: ASDF (Adaptible Seismic Data Format) based on HDF5 that contains data (waveforms), metadata, and provenance (description of its generation) and works with Obspy.



Final words

In seismological research, most leading-edge problems require larger-scale simulations working on huge data sets, requiring substantial post-processing resources (e.g. visualization, filtering, etc.) that can rarely be developed and maintained by (small) research groups.

Ongoing developments to openly distribute simulation software for Earth sciences (e.g. CIG) with sufficient documentation and training material, as well as the development of community platforms such as VERCE or EPOS.