Istituto Nazionale di Oceanografia
e di Geofisica Sperimentale, Trieste - Italy

Parallel methods
for
wave modelling codes

Géza Seriani

ICTP School for HPC applications in Earth Sciences, November 2014, Trieste

Outline

Code development main issues:

- **Problem definition**
- **Hard- & Soft- Tools**
- **MATH representation**
- **GRID representation**
- **Examples**

Problem definition

Wave modelling
for
exploration geophysics
&
engineering seismology

Modelling constraints

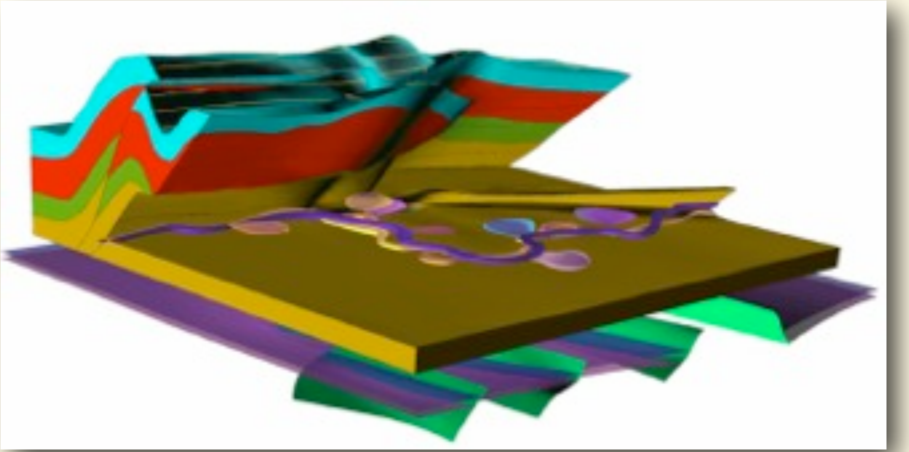
For realistic simulations:

- *heteregeneous properties*
  must be correctly reproduced,

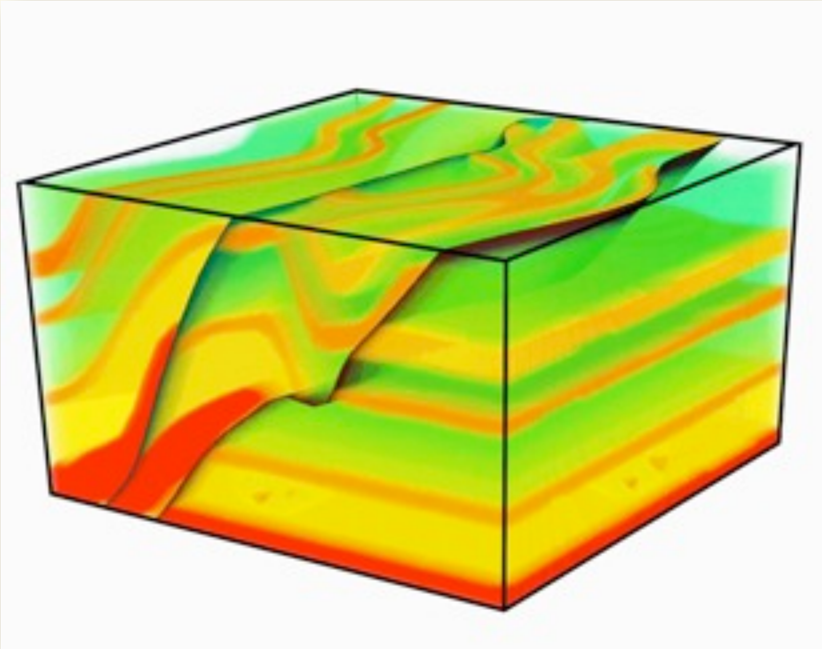
- *very complex structures*
  must be correctly modelled,

- *numerical algorithms* **must be computationally efficient.**

Space discretization:



Geo-structure complexity



Geo-structure heterogeneity

Computational issues:

The numerical solution requires

- **a** *huge computational effort,*
- **both in** *memory storage* *(from Giga to Tera nodes),*
- **&** *CPU time* *(from hours to weeks).*

Computational issues:

- a *highly accurate method* is needed for reducing storage and CPU time requirements,

- an *efficient implementation* of the algorithm is needed for reducing the total cost of the simulation.

Computational constraints:

(consequences)

Algorithms must use:

- vector/parallel platforms (clusters, massive parallel...),
- efficient hard-/soft-ware subroutines (FFT, Lapack, MPI...),
- low count of operations & of primary storage (memory).

High-order methods:

(possible choices)

- finite difference methods,
- pseudo-spectral methods,
- finite element methods,
- spectral element methods,
- finite volume methods,
- discontinous Galerkin methods.

Modelling methods

FD & Pseudo-spectral methods

3D elastic wave equation

Matrix *velocity-stress* formulation

$$\frac{\partial}{\partial t}\boldsymbol{W} = \mathrm{A}\frac{\partial}{\partial x}\boldsymbol{W} + \mathrm{B}\frac{\partial}{\partial y}\boldsymbol{W} + \mathrm{C}\frac{\partial}{\partial z}\boldsymbol{W} + \boldsymbol{f}$$

$$\boldsymbol{W} = \{v_x,\ v_y,\ v_z,\ \sigma_{xx},\ \sigma_{yy},\ \sigma_{zz},\ \sigma_{xy},\ \sigma_{xz},\ \sigma_{yz}\}^\top$$

$$\boldsymbol{f} = \{f_x/\rho,\ f_y/\rho,\ f_z/\rho,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\}^\top$$

forcing source

stress field

particle velocity

Matrices of elastic constants

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 0 & 1/\rho & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/\rho & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/\rho & 0 \\ \lambda + 2\mu & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mu & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mu & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

λ & μ Lame' elastic constants

Matrices of elastic constants

$$\mathbf{A}$$

$$\mathbf{B} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1/\rho & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/\rho & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/\rho \\ 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda+2\mu & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mu & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mu & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

λ & μ Lame' elastic constants

Matrices of elastic constants

λ & μ Lame' elastic constants

$\mathbf{A}$

$\mathbf{B}$

$$\mathbf{C} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/\rho & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/\rho \\ 0 & 0 & 0 & 0 & 0 & 1/\rho & 0 & 0 & 0 \\ 0 & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda + 2\mu & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mu & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mu & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

FD & Pseudo-spectral solution

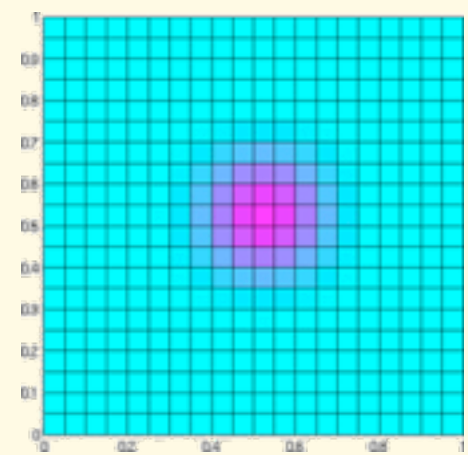
Matrix *velocity-stress* formulation

$$\frac{\partial}{\partial t}\boldsymbol{W} = \mathrm{A}\,\tilde{\mathrm{D}}_x\,\boldsymbol{W} + \mathrm{B}\,\tilde{\mathrm{D}}_y\,\boldsymbol{W} + \mathrm{C}\,\tilde{\mathrm{D}}_z\,\boldsymbol{W} + \boldsymbol{f}$$

Discrete space derivative operators

Computed by using:

Cartesian grids



FD & Pseudo-spectral solution

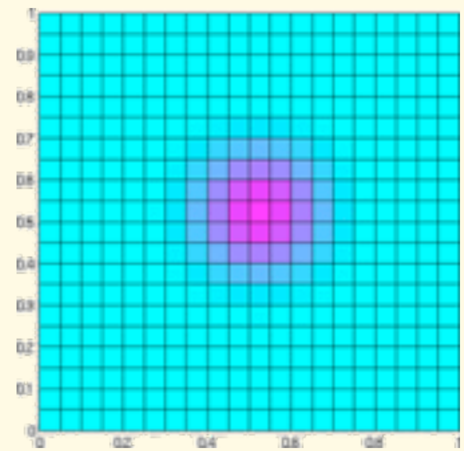
Matrix *velocity-stress* formulation

$$\frac{\partial}{\partial t}\boldsymbol{W} = \mathrm{A}\,\tilde{\mathrm{D}}_x\,\boldsymbol{W} + \mathrm{B}\,\tilde{\mathrm{D}}_y\,\boldsymbol{W} + \mathrm{C}\,\tilde{\mathrm{D}}_z\,\boldsymbol{W} + \boldsymbol{f}$$

Discrete space derivative operators

Computed by using:

- Finite differences,
- Fourier Fast Transform,
- Chebyshev Fast Transform,
- Chebyshev Derivative Matrix.



Time integration is done by time stepping methods.

Space derivative operators

Can be obtained by differentiating an assumed expansion of the solution in given basis functions:

$$u(x) \approx u_N(x) = \sum_{j=0}^{N} \hat{u}_j \, \varphi_j(x)$$

spectral coefficients

Basis functions

- Taylor expansion (finite differences),
- Trigonometric functions (pseudo-spectral periodic),
- Chebyshev or Legendre polynomials (pseudo-spectral).

Space derivative operators

Expressing the approximant with the values $u_N(x_i)$ at the collocation points:

node values

$$u_N(x) = \sum_{i=0}^{N} u_N(x_i)\ \phi_i(x)$$

Lagrange or cardinal basis

$$\partial_x u_N(x) = \sum_{i=0}^{N} u_N(x_i)\ \partial_x \phi_i(x) = \sum_{i=0}^{N} u_N^{'}(x_i)\ \phi_i(x)$$

$$u_N^{'}(x_i) = \sum_{j=0}^{N} (\tilde{\mathbf{D}}_x)_{ij}\ u_N(x_j) \quad \longleftrightarrow \quad \boxed{U_N^{'} = \tilde{\mathbf{D}}_x\ U_N}$$

Space derivative operators

Finite difference methods:

- banded matrix derivatives,
- efficient code implementation,
- low order & accuracy rely on order.

Pseudo-spectral methods:

- full matrix derivatives,
- efficient code implementation using FFT,
- high order & high accuracy.

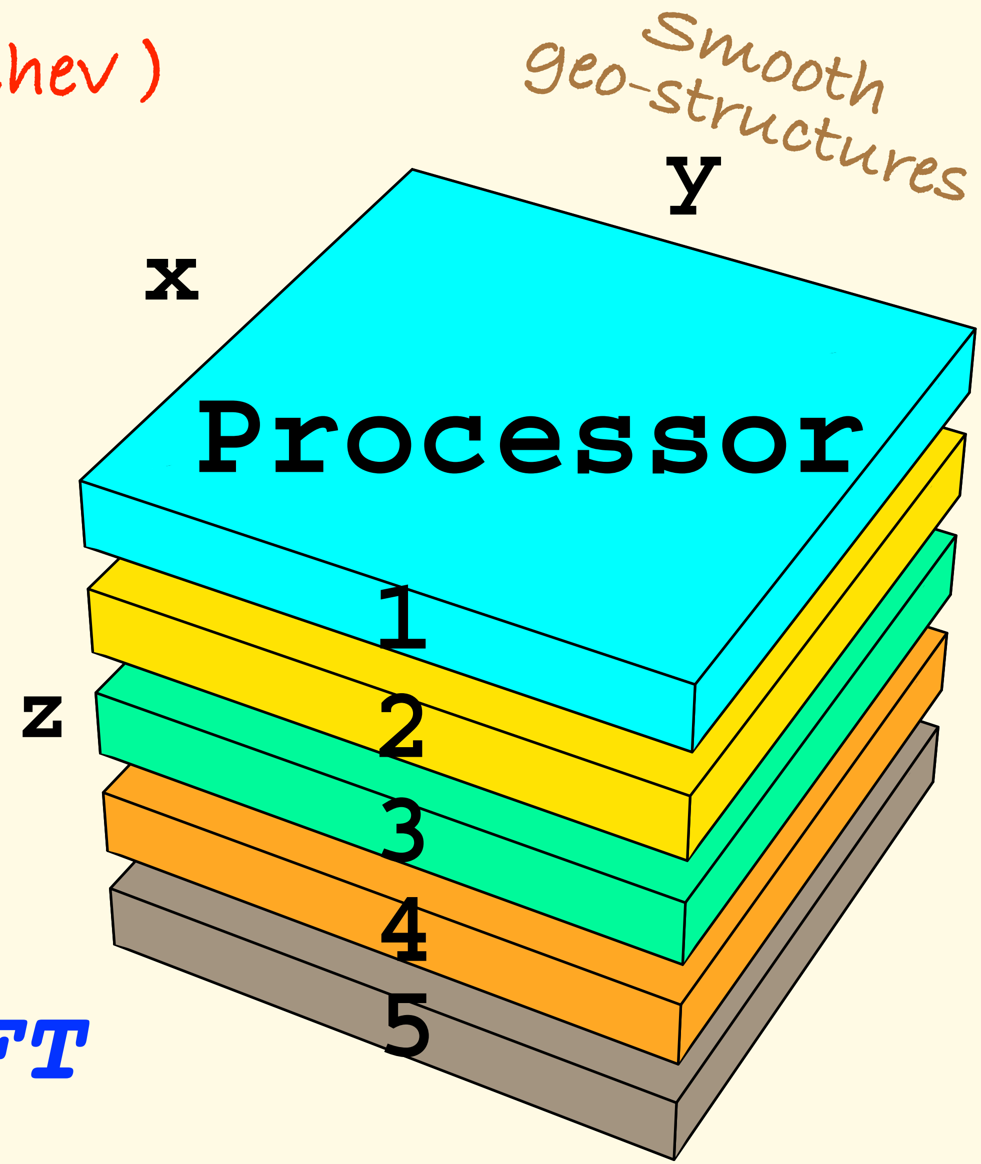
Parallel computation
Pseudo-spectral methods

(Fourier & Chebyshev)

$D_x \longleftrightarrow \partial/\partial x$

$D_y \longleftrightarrow \partial/\partial y$

D_x & D_y by *FFT*



Parallel computation
Pseudo-spectral methods

(Fourier & Chebyshev)

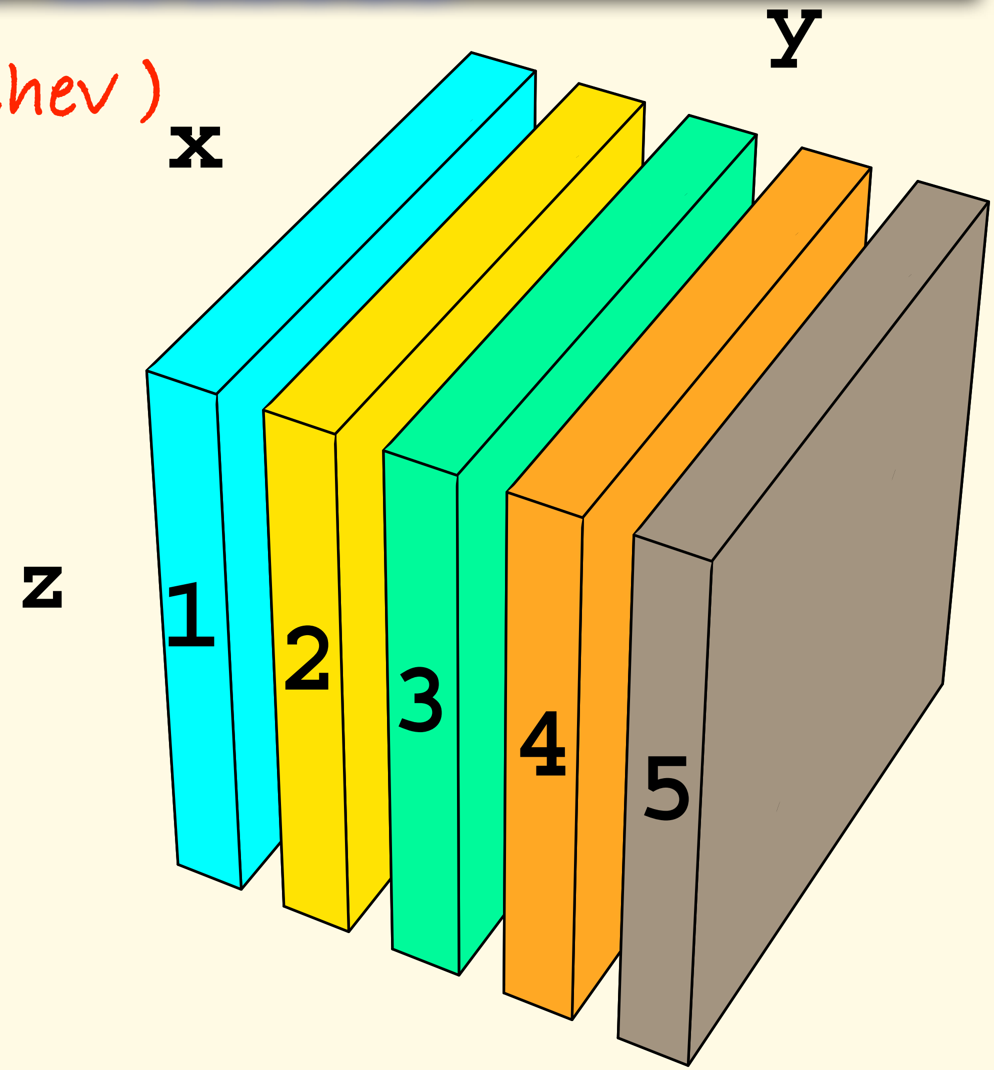
Transposition

+

$\mathbf{D}_z \longleftrightarrow \partial/\partial z$

$\mathbf{D}_z$ ***by FFT***

Global communications



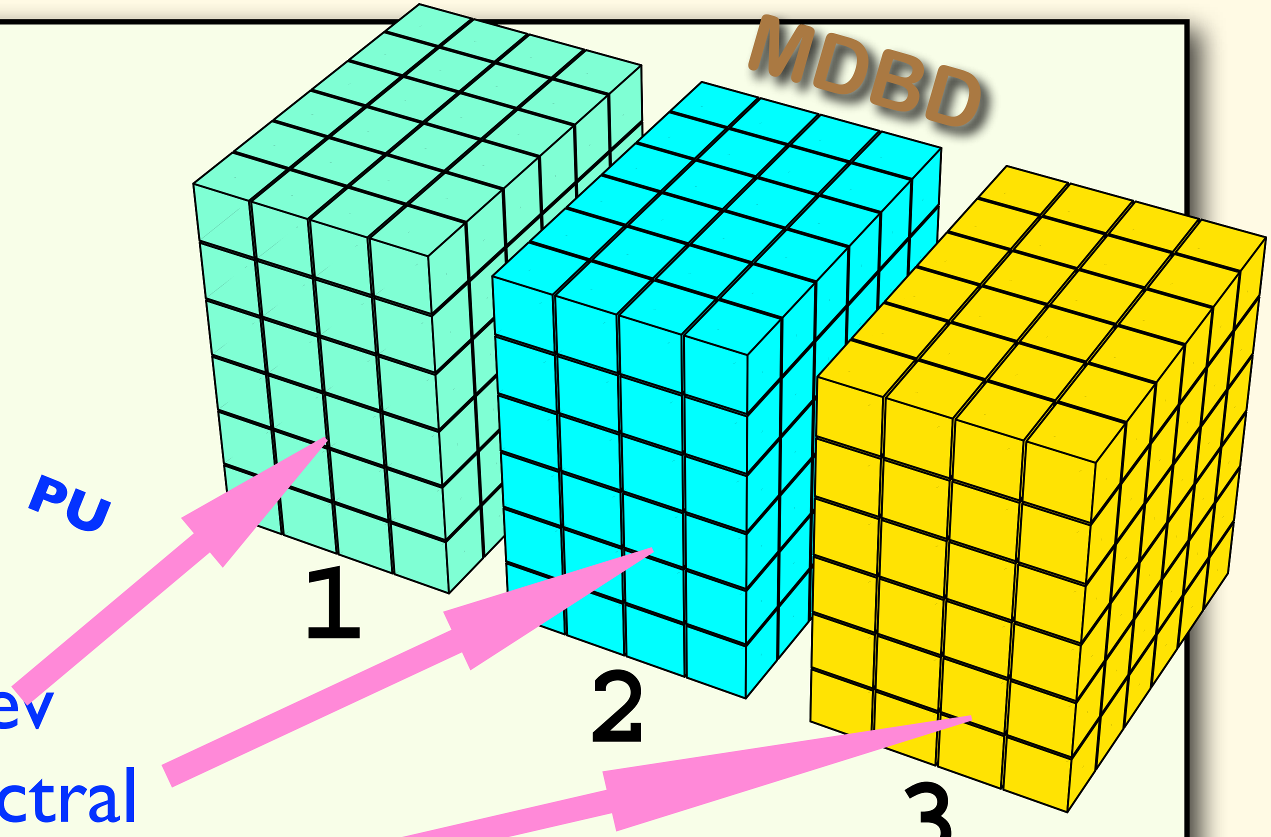
Parallel computation
Multi-Domain Block Decomposition

MDBD pseudo-spectral method:

- **Physical domain decomposed in a *hierarchical* way:**
  - **First decomposition is in *large blocks*,**
  - **Blocks decomposed in *small overlapping elements* ,**
- ***Derivatives* computed concurrently in each element by the Chebyshev pseudo-spectral method,**
- ***Continuity* across the element boundaries assured by the *overlap* of the discrete derivative operator,**
- ***Same accuracy* order at interfaces & in elements.**

Multi-Domain Block Decomposition

3D

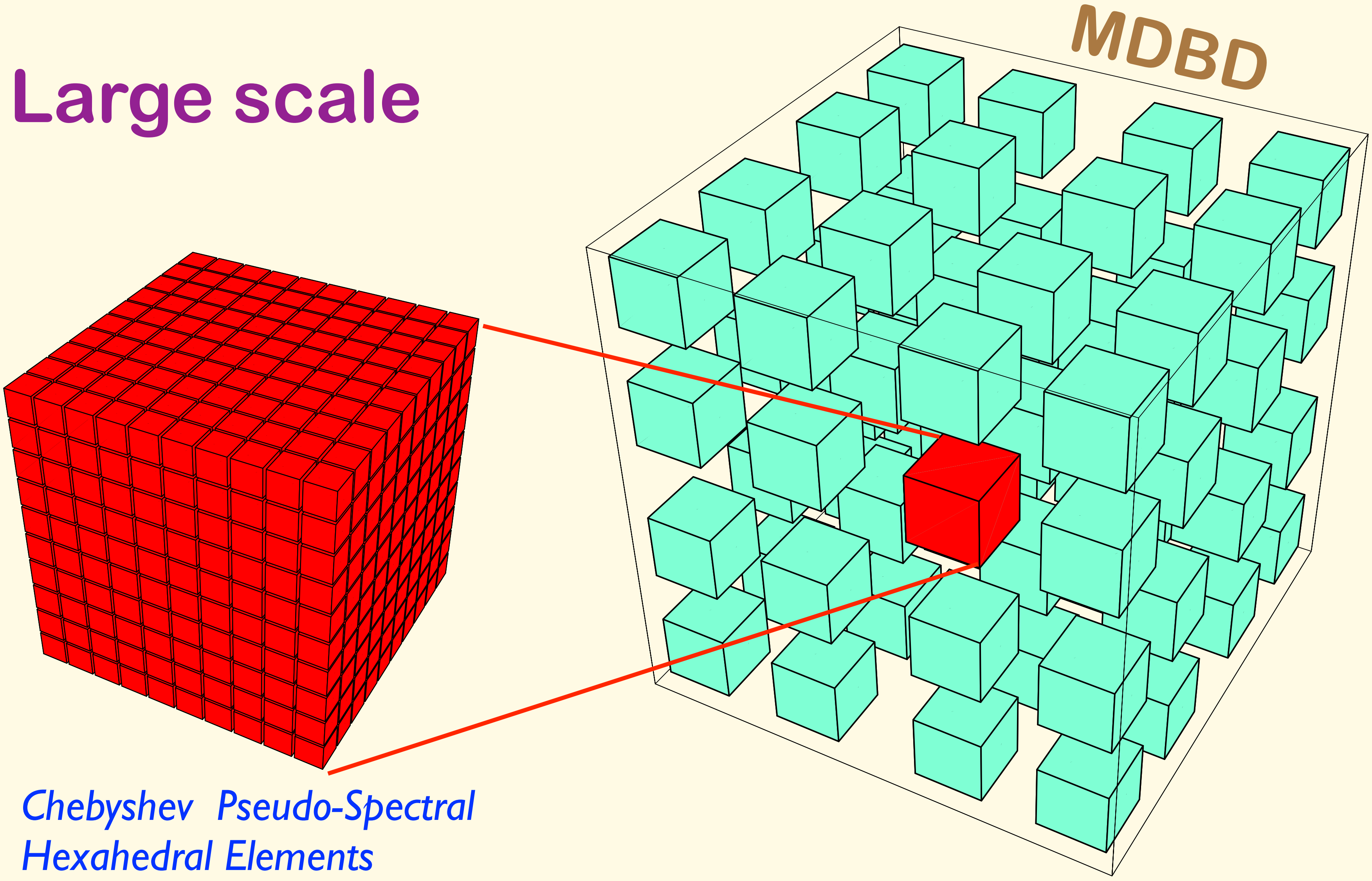


Chebyshev
Pseudo-Spectral
Hexahedral
Elements

$$u'^{e}_{i} = (\mathbf{D}\, u^{e})_{i} = \sum_{j=0}^{N} \mathbf{D}_{ij}\, u^{e}_{j}$$

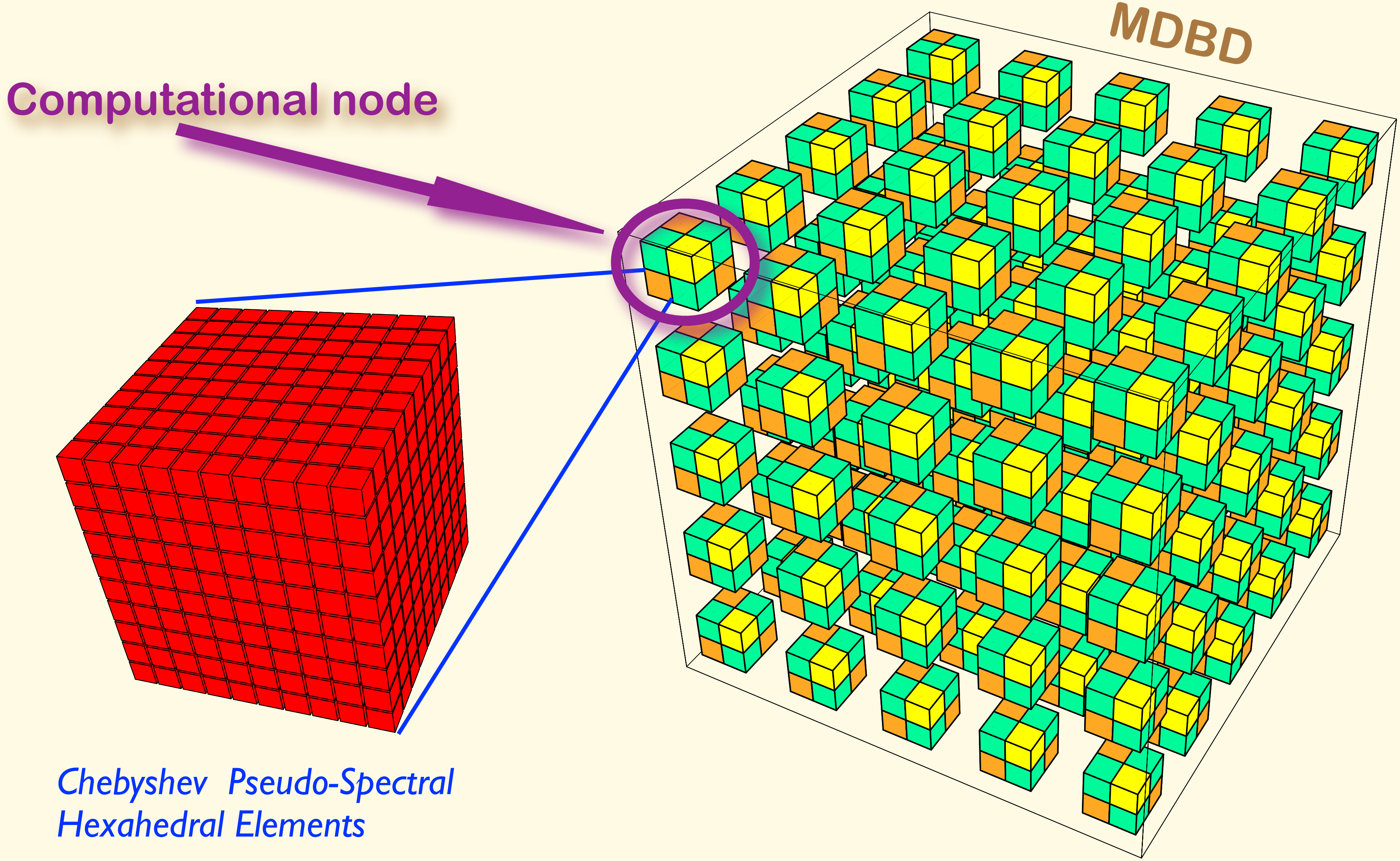
Multi-Domain Block Decomposition

Large scale



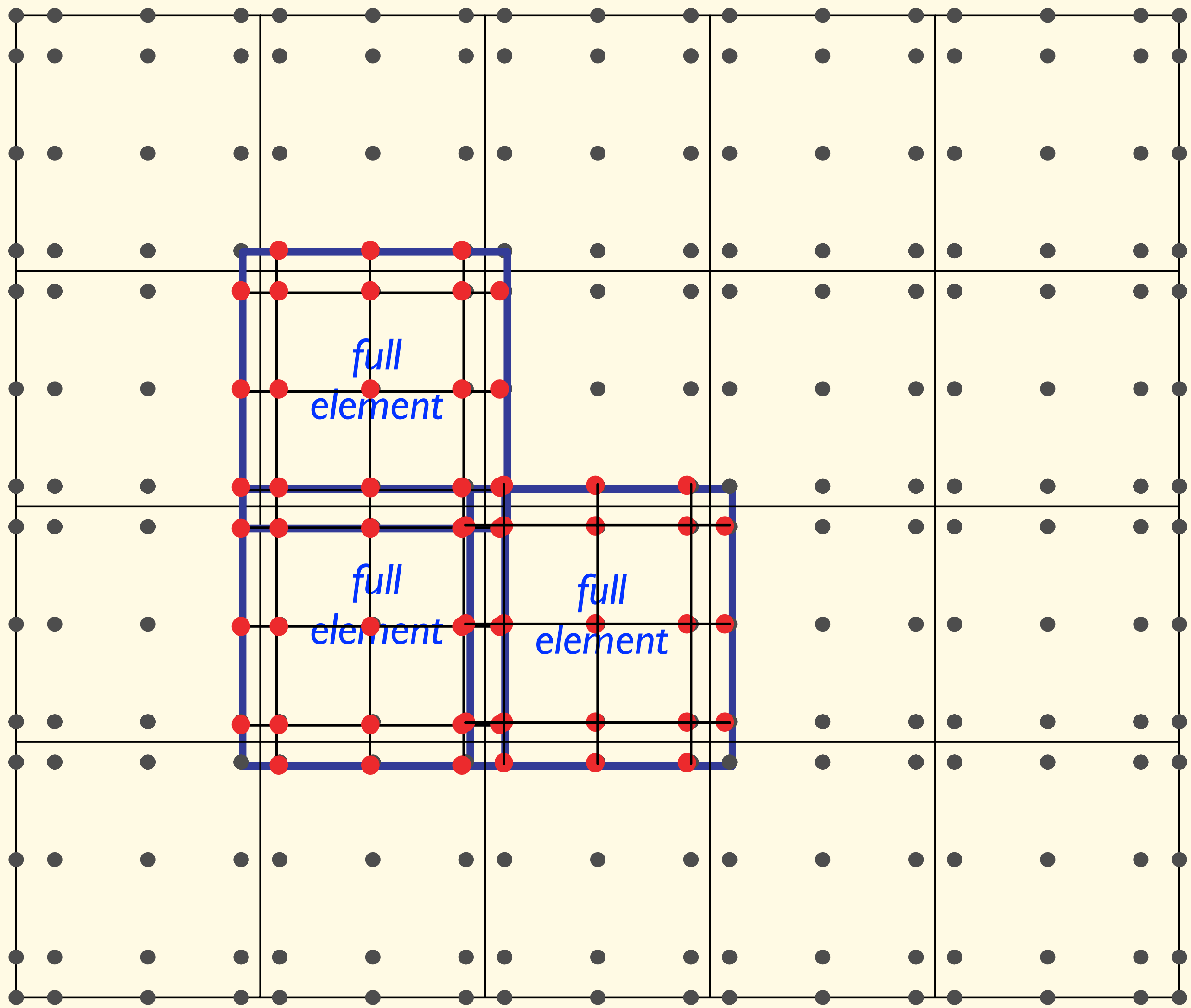
Chebyshev Pseudo-Spectral
Hexahedral Elements

Multi-Domain Block Decomposition



Chebyshev Pseudo-Spectral
Hexahedral Elements

Chebyshev Derivative & Overlap



Kronecker Product & Vec Operator

Given $\mathbf{A} = [a_{ij}]^{m \times n}$ & $\mathbf{B} = [b_{ij}]^{p \times q}$

$$\mathbf{A} \otimes \mathbf{B} \doteq \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \dots & a_{1n}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \dots & a_{2n}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & a_{m2}\mathbf{B} & \dots & a_{mn}\mathbf{B} \end{bmatrix}$$

block matrix of order (mp x nq)

$\mathbf{Vec}\ (\bullet)$ reshapes a generic rank-*K* tensor in a column vector of length

$$n_1 n_2 \cdots n_K$$

Kronecker Product & Vec Operator

- Generalized inner product:

$$\mathbf{A} \star \boldsymbol{U} \doteq \sum_{l=0}^{N} A_{il} U_{lmn}$$

contraction index

- Generalized transpose:

$$(U_{ijk})^\top \doteq U_{jki}$$

rotation

by Kronecker product properties

$$(\mathbf{A}^e \otimes \mathbf{B}^e \otimes \mathbf{C}^e)\, Vec(\boldsymbol{U}^e) = Vec(\mathbf{A}^e \star (\mathbf{B}^e \star (\mathbf{C}^e \star \boldsymbol{U}^e)^\top)^\top)^\top$$

Derivative by Kronecker Product

$$\tilde{\mathbf{D}}_x^e = \mathbf{I}_z^e \otimes \mathbf{I}_y^e \otimes \mathbf{D}_x^e$$

$$\tilde{\mathbf{D}}_y^e = \mathbf{I}_z^e \otimes \mathbf{D}_y^e \otimes \mathbf{I}_x^e$$

$$\tilde{\mathbf{D}}_z^e = \mathbf{D}_z^e \otimes \mathbf{I}_y^e \otimes \mathbf{I}_x^e$$

Unity matrix

3D

Pseudo-spectral element derivative operators

by Kronecker product properties

$$\tilde{\mathbf{D}}_x^e(\boldsymbol{U}^e) = (\mathbf{I}_z^e \star (\mathbf{I}_y^e \star (\mathbf{D}_x^e \star \boldsymbol{U}^e)^\top)^\top)^\top$$ X - axis

$$\tilde{\mathbf{D}}_y^e(\boldsymbol{U}^e) = (\mathbf{I}_z^e \star (\mathbf{D}_y^e \star (\mathbf{I}_x^e \star \boldsymbol{U}^e)^\top)^\top)^\top$$ Y - axis

$$\tilde{\mathbf{D}}_z^e(\boldsymbol{U}^e) = (\mathbf{D}_z^e \star (\mathbf{I}_y^e \star (\mathbf{I}_x^e \star \boldsymbol{U}^e)^\top)^\top)^\top$$ Z - axis

Derivative by Kronecker Product

Element wave field:

$$\bar{N} = N + 1$$

$$\boldsymbol{U}^e_{ijk} = u^e(x_i, y_j, z_k, t) = Vec\,(\boldsymbol{U}^e)_{k\bar{N}^2 + j\bar{N} + i + 1}$$

grid values

interpolation order N

Pseudo-spectral element derivatives:

$$\tilde{\mathbf{D}}^e_x(\boldsymbol{U}^e) = \mathbf{D}^e_x \star \boldsymbol{U}^e$$

X - axis

$$\tilde{\mathbf{D}}^e_y(\boldsymbol{U}^e) = (\mathbf{D}^e_y \star \boldsymbol{U}^{e\top})^{\top\top}$$

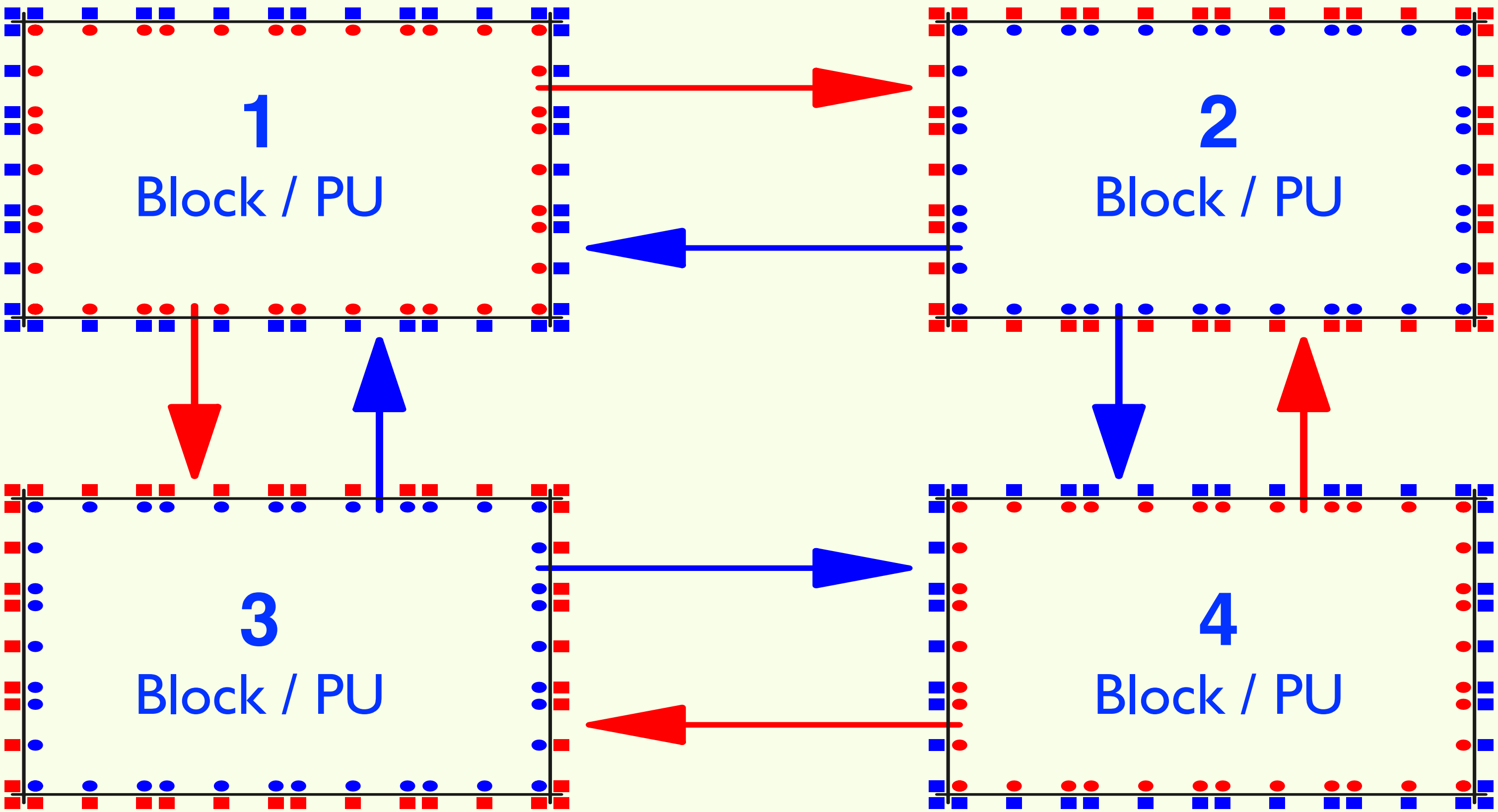
Y - axis

$$\tilde{\mathbf{D}}^e_z(\boldsymbol{U}^e) = (\mathbf{D}^e_z \star \boldsymbol{U}^{e\top\top})^{\top}$$

Z - axis

Computed by optimized *matrix-matrix* products (LAPACK)

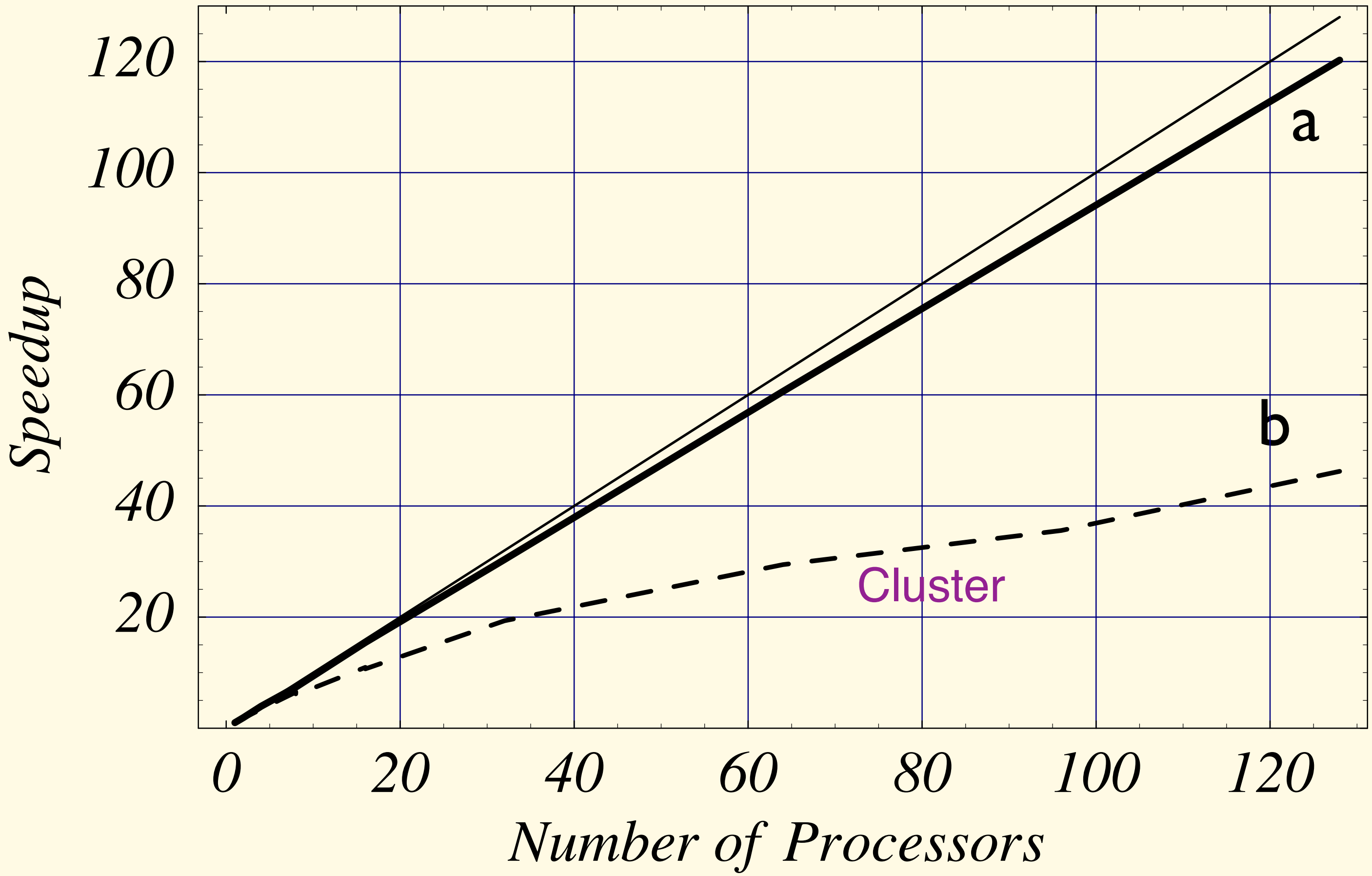
Synchronization & communications



Ghost Cells up-date

MDBD Scaled // Speed-up

	32800 elements	9 nodes/element	419400 elements
a:	230 Mw (226x226x226)		45.1 Gw (1796x898x898)
b:	400 Mw (242x242x242)		51.5 Gw (1922x962x962)
	1730 elements	22 nodes/element	221200 elements

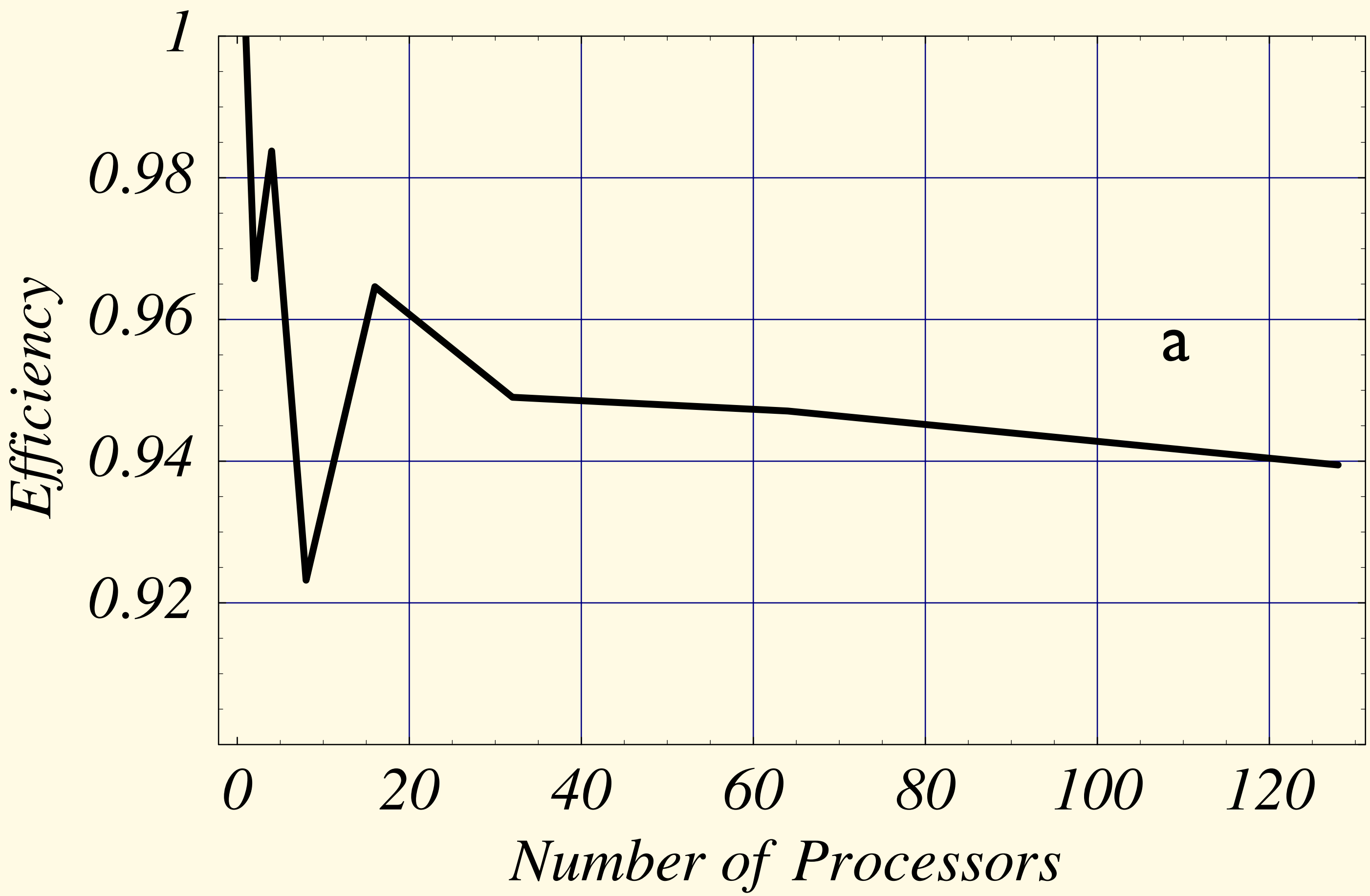


IBM
SP5
512

MDBD Scaled // Speed-up

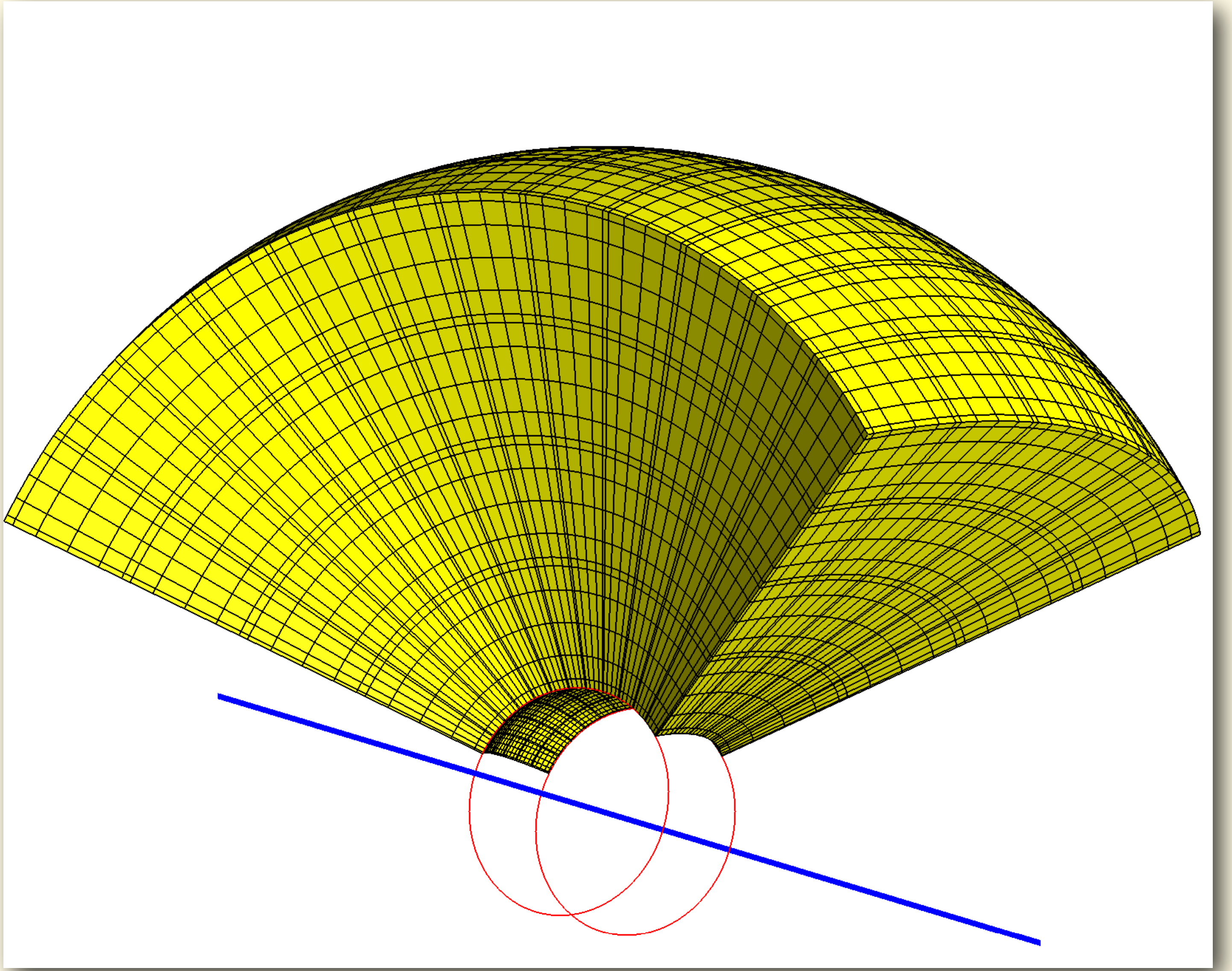
32800 elements — 9 nodes/element — 419400 elements

a: 230 Mw (226x226x226) — 45.1 Gw (1796x898x898)

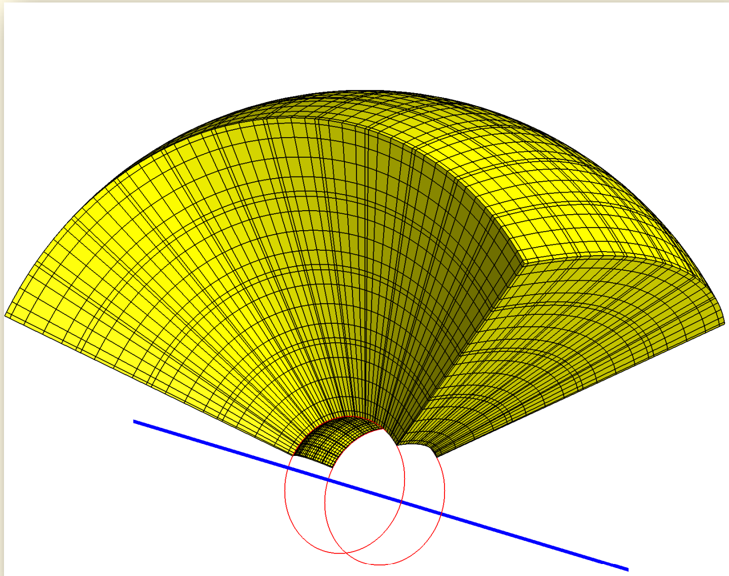


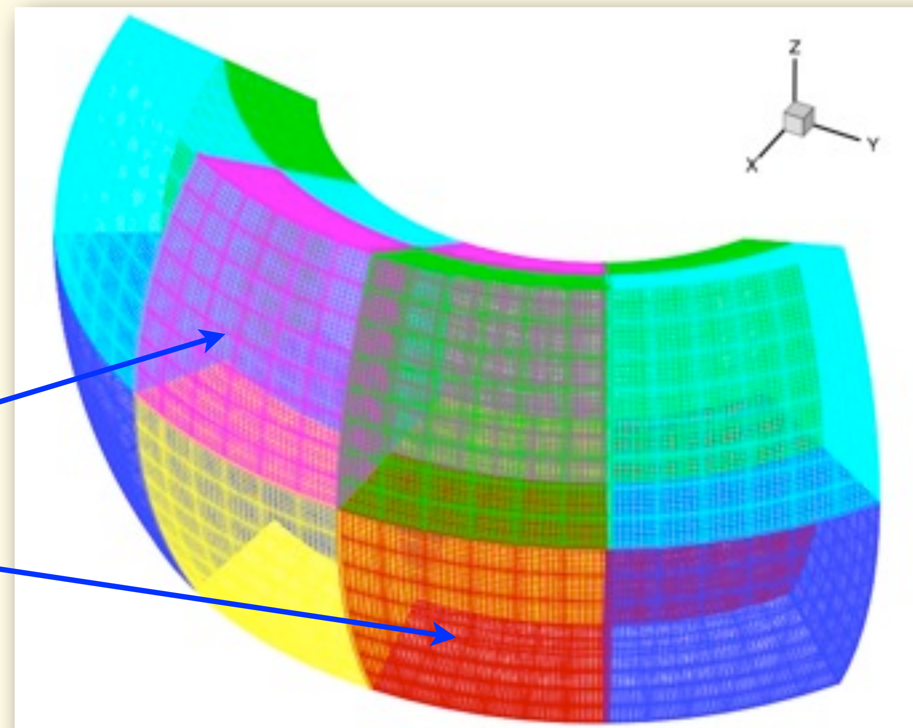
IBM
SP5
512

Spherical Coordinates



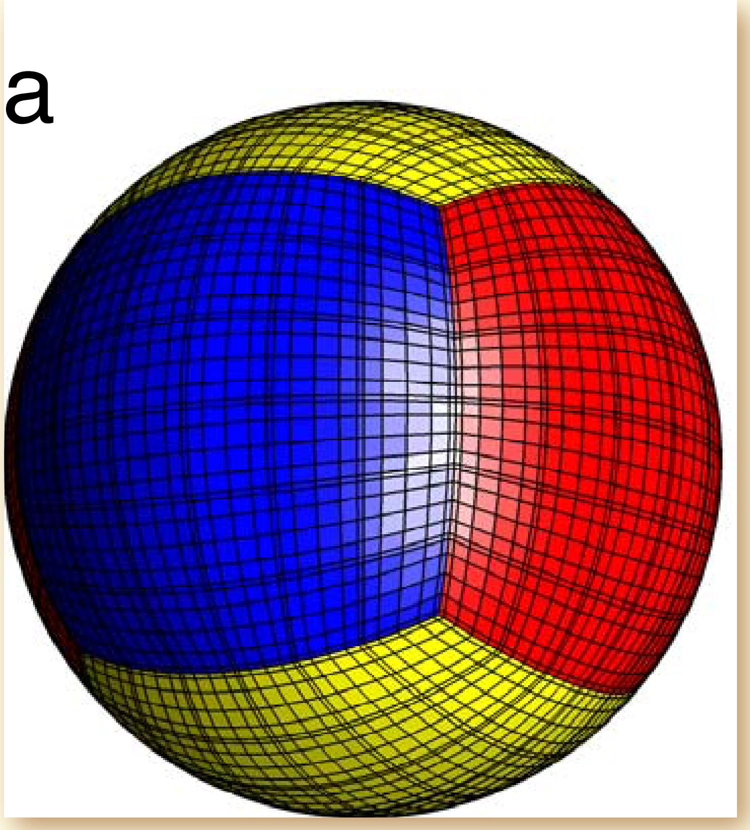
Spherical Coordinates

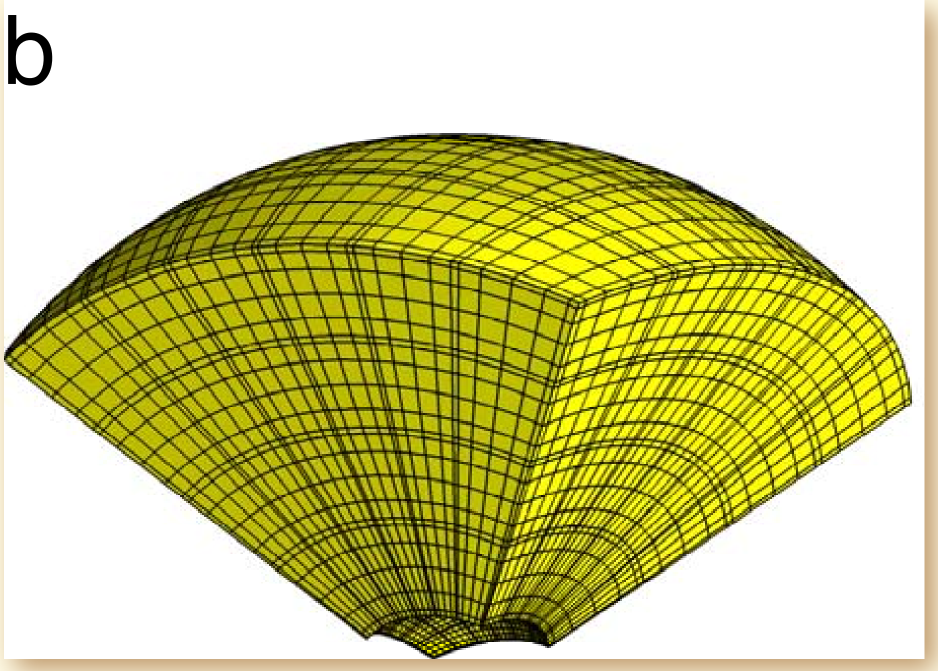


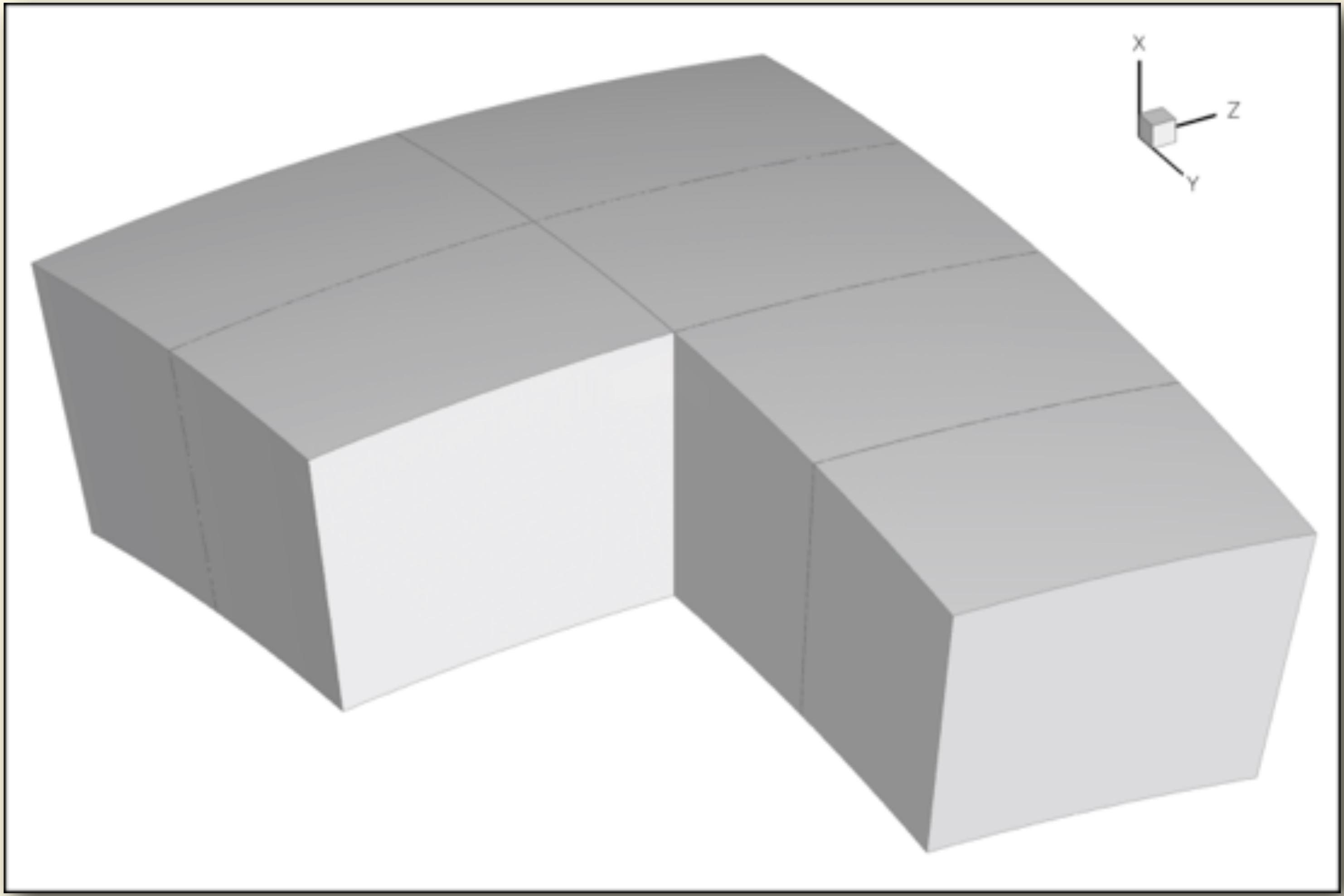


Parallel decomposition

Cubed Sphere






X
Z
Y

Modelling methods

Finite element methods
(FEM)
&
Spectral element methods
(SEM)

3D elastic wave equation

$$\rho \frac{\partial^2 \boldsymbol{u}}{\partial t^2} - \nabla[(\lambda + 2\mu)\nabla \cdot \boldsymbol{u}] - \nabla \cdot (\mu \nabla \boldsymbol{u}) = \mathbf{f}$$

$$\boxed{\rho \frac{\partial^2 \boldsymbol{u}}{\partial t^2} - \mathcal{D}^\top \boldsymbol{\sigma} = \mathbf{f}}$$

matrix formulation

$$\boldsymbol{\sigma}(\boldsymbol{u}) = \boldsymbol{C}\, \boldsymbol{\epsilon}(\boldsymbol{u})$$

stress-strain relation

$$\boldsymbol{\epsilon}(\boldsymbol{u}) = \mathcal{D}\, \boldsymbol{u}$$

strain-displacement relation

2D elastic wave equation

$$\boxed{\rho \frac{\partial^2 \boldsymbol{u}}{\partial t^2} - \mathcal{D}^\top \boldsymbol{\sigma} = \mathbf{f}}$$

in 2D:

$$\mathcal{D} = \begin{bmatrix} \partial_x & 0 \\ 0 & \partial_y \\ \partial_y & \partial_x \end{bmatrix}$$

differential operator

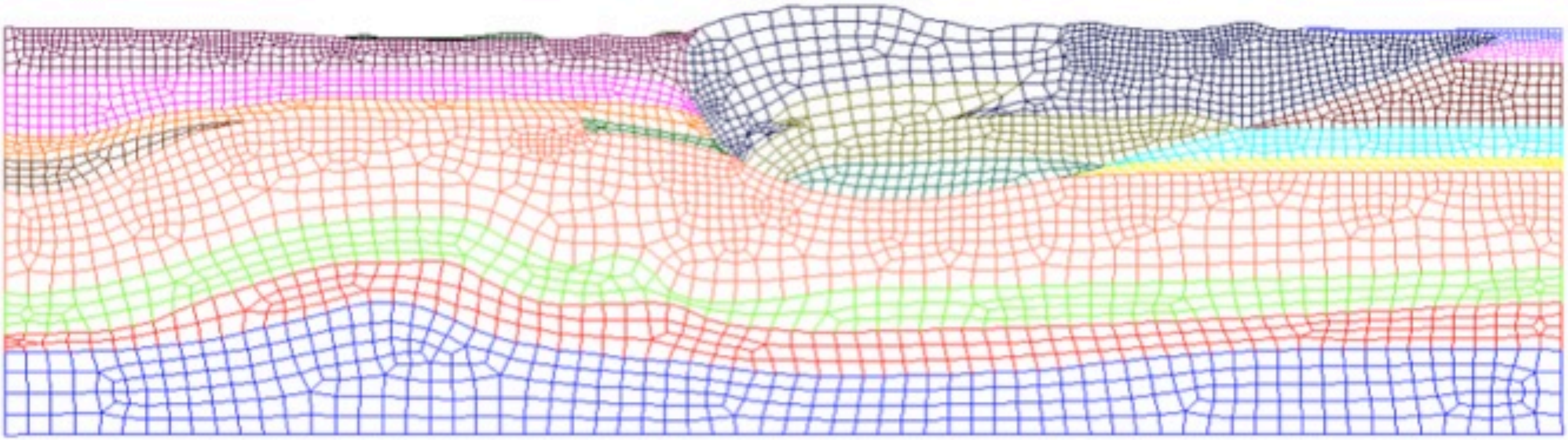
$$C = \begin{bmatrix} \lambda + 2\mu & \lambda & 0 \\ \lambda & \lambda + 2\mu & 0 \\ 0 & 0 & \mu \end{bmatrix}$$

elastic stiffness matrix

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{x}, t) = \{\sigma_{xx},\ \sigma_{yy},\ \sigma_{xy}\}^\top$$
$$\boldsymbol{\epsilon} = \boldsymbol{\epsilon}(\boldsymbol{x}, t) = \{\epsilon_{xx},\ \epsilon_{yy},\ \epsilon_{xy}\}^\top$$

stress & strain vectors

Fem/Sem mesh



$$\boldsymbol{u} \cong \tilde{\boldsymbol{u}}(x,y,z,t) \;=\; \bigcup_{e=1}^{n_e} \tilde{\boldsymbol{u}}^e(x,y,z,t)$$

shape functions

$$\tilde{\boldsymbol{u}}^e(x,y,z,t) \;=\; \sum_{i,j,k=0}^{N} \boldsymbol{u}^e_{ijk}(t)\,\Phi_{ijk}(x,y,z)$$

node values

Fem/Sem solution

$$\boxed{\rho \frac{\partial^2 \boldsymbol{u}}{\partial t^2} - \mathcal{D}^\top \boldsymbol{\sigma} = \mathbf{f}}$$

strong form

$$\frac{d^2}{dt^2} \int_\Omega \rho \, \boldsymbol{w}^\top \boldsymbol{u} \, d\Omega + \int_\Omega \boldsymbol{w}^\top \mathcal{D}^\top \mathbf{C} \, \mathcal{D} \boldsymbol{u} \, d\Omega = \int_\Omega \boldsymbol{w}^\top \mathbf{f} \, d\Omega$$

Fem/Sem solution

$$\rho \frac{\partial^2 \boldsymbol{u}}{\partial t^2} - \mathcal{D}^\top \boldsymbol{\sigma} = \mathbf{f}$$

weak form

$$\boxed{\frac{d^2}{dt^2} \int_\Omega \rho\, \boldsymbol{w}^\top \boldsymbol{u}\; d\Omega + \int_\Omega \boldsymbol{w}^\top \mathcal{D}^\top \mathbf{C}\, \mathcal{D}\, \boldsymbol{u}\; d\Omega = \int_\Omega \boldsymbol{w}^\top \mathbf{f}\; d\Omega}$$

Fem/Sem solution

$$\rho \frac{\partial^2 \boldsymbol{u}}{\partial t^2} - \mathcal{D}^\top \boldsymbol{\sigma} = \mathbf{f}$$

$$\frac{d^2}{dt^2} \int_\Omega \rho\, \boldsymbol{w}^\top \boldsymbol{u}\; d\Omega + \int_\Omega \boldsymbol{w}^\top \mathcal{D}^\top \mathbf{C}\, \mathcal{D}\, \boldsymbol{u}\; d\Omega = \int_\Omega \boldsymbol{w}^\top \mathbf{f}\; d\Omega$$

$$\tilde{\boldsymbol{u}}^e(x,y,z,t) = \sum_{i,j,k=0}^{N} \boldsymbol{u}^e_{ijk}(t)\, \Phi_{ijk}(x,y,z)$$

$$\boxed{\mathbf{M}\, \ddot{\boldsymbol{U}} + \mathbf{K}\, \boldsymbol{U} = \boldsymbol{F}}$$

Solve a system of second order ordinary differential equations

Fem/Sem solution

$$\rho \frac{\partial^2 \boldsymbol{u}}{\partial t^2} - \mathcal{D}^\top \boldsymbol{\sigma} = \mathbf{f}$$

$$\frac{d^2}{dt^2} \int_\Omega \rho\, \boldsymbol{w}^\top \boldsymbol{u}\, d\Omega + \int_\Omega \boldsymbol{w}^\top \mathcal{D}^\top \mathbf{C}\, \mathcal{D}\, \boldsymbol{u}\, d\Omega = \int_\Omega \boldsymbol{w}^\top \mathbf{f}\, d\Omega$$

$$\mathbf{M} = \sum_{\mathbf{e=1}}^{\mathbf{n_e}} \mathbf{M}^{(\mathrm{e})} \qquad \mathbf{K} = \sum_{\mathbf{e=1}}^{\mathbf{n_e}} \mathbf{K}^{(\mathrm{e})} \qquad \boldsymbol{F} = \sum_{\mathbf{e=1}}^{\mathbf{n_e}} \boldsymbol{F}^{(\mathrm{e})}$$

mass

stiffness

load

$$\boxed{\mathbf{M}\, \ddot{\boldsymbol{U}} + \mathbf{K}\, \boldsymbol{U} = \boldsymbol{F}}$$

Solve a system of second order ordinary differential equations

Fem/Sem solution

$$\rho \frac{\partial^2 \boldsymbol{u}}{\partial t^2} - \mathcal{D}^\top \boldsymbol{\sigma} = \mathbf{f}$$

$$\frac{d^2}{dt^2} \int_\Omega \rho\, \boldsymbol{w}^\top \boldsymbol{u}\; d\Omega + \int_\Omega \boldsymbol{w}^\top \mathcal{D}^\top \mathbf{C}\, \mathcal{D}\, \boldsymbol{u}\; d\Omega = \int_\Omega \boldsymbol{w}^\top \mathbf{f}\; d\Omega$$

initial conditions

$$\boxed{\mathbf{M}\, \ddot{\boldsymbol{U}} + \mathbf{K}\, \boldsymbol{U} = \boldsymbol{F}}$$

$$\boldsymbol{U}(0) = \boldsymbol{U}_0$$

$$\dot{\boldsymbol{U}}(0) = \dot{\boldsymbol{U}}_0$$

Solve a system of second order ordinary differential equations

Finite & spectral elements

Finite elements:

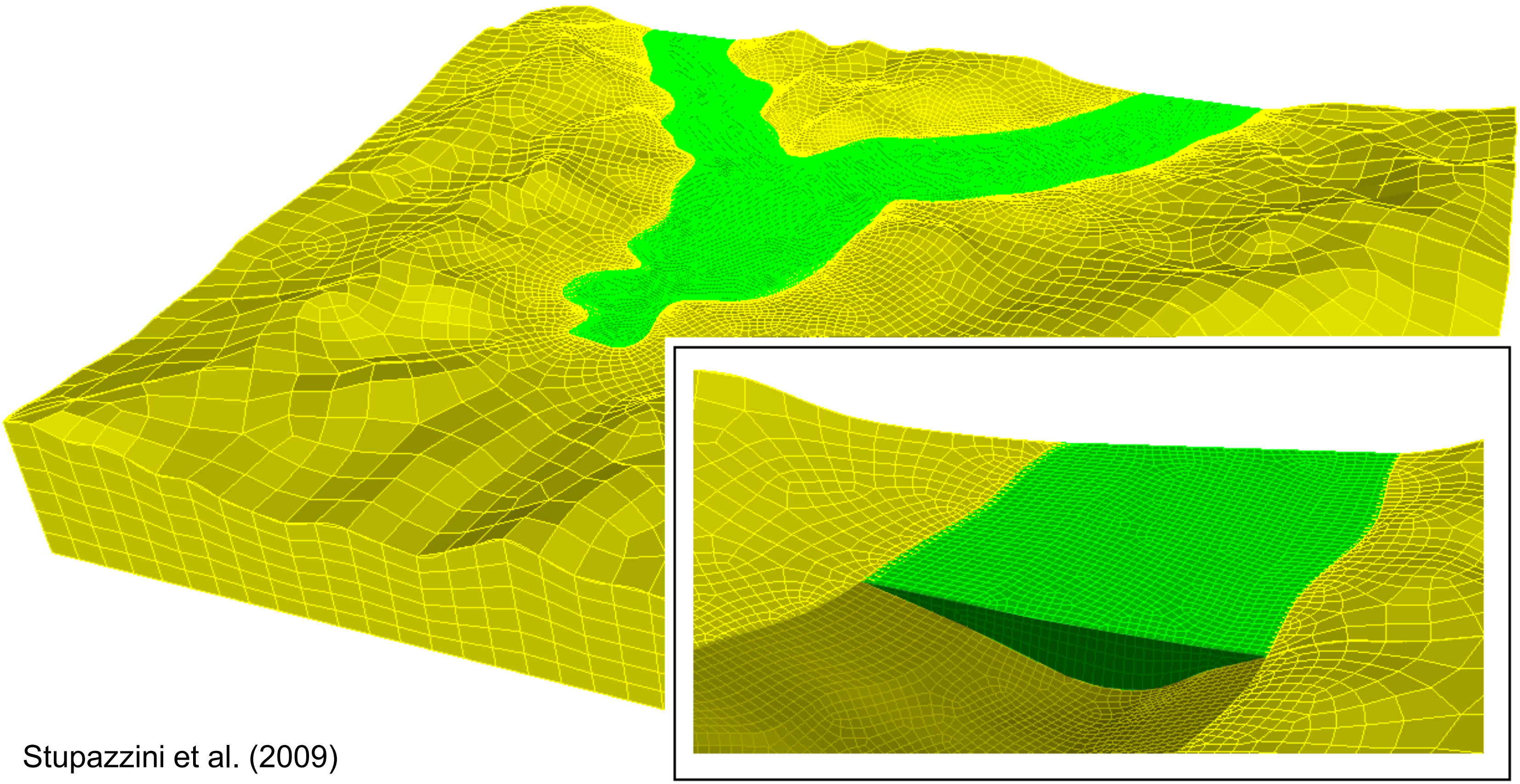
- low order & low accuracy (in general),
- shape functions by linear, quadratic and cubic polynomials,
- triangular/quadrangular or tetrahedral/hexahedral elements.

Spectral elements:

- high order & high accuracy,
- shape functions by Chebyshev and Legendre orthogonal polynomials,
- quadrangular or hexahedral elements.

SEM in 3D: Mesh design

Realistic example: **The Grenoble valley**



Stupazzini et al. (2009)

Spectral elements

Chebyshev polynomials:

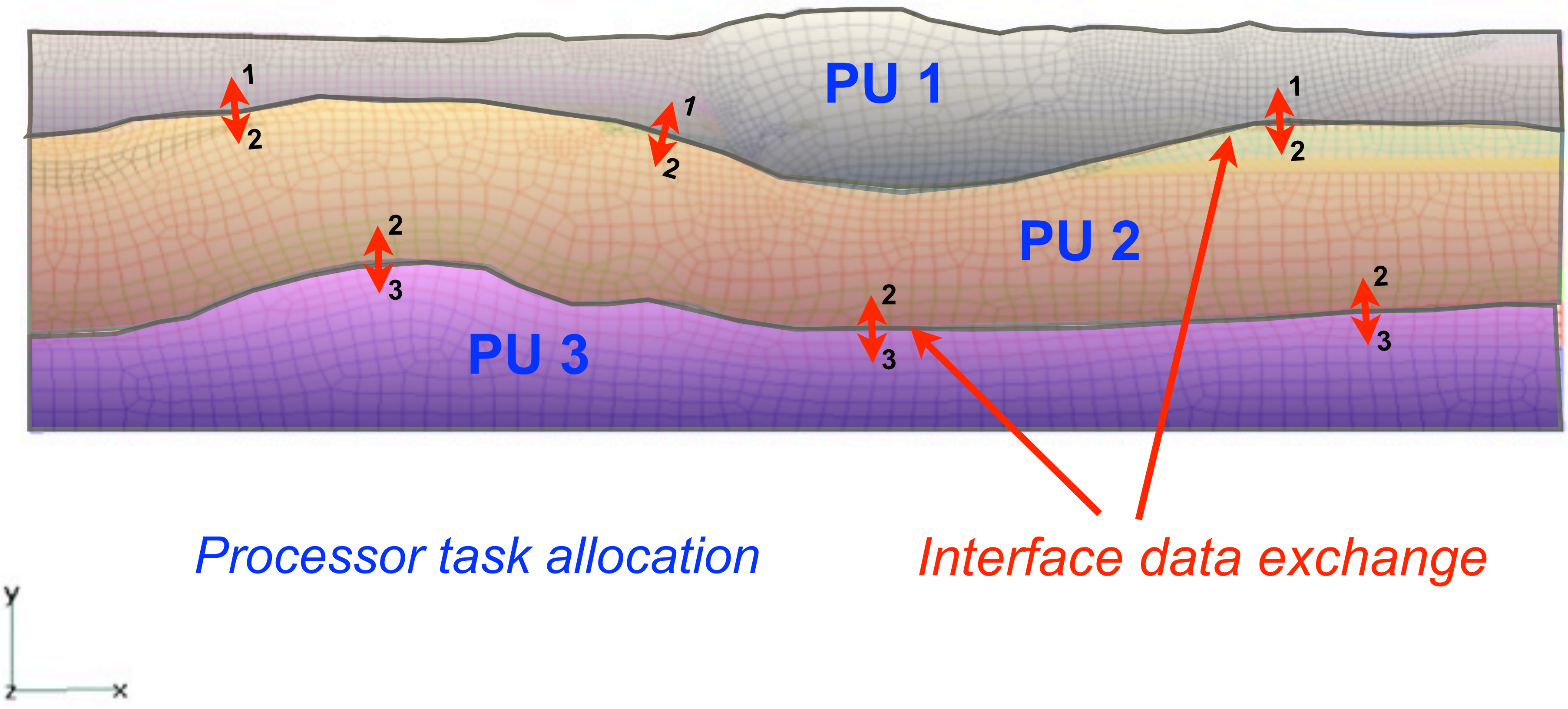
- **non-diagonal** **system matrices,**
- **implicit** **schemes** *(in time)* **=> linear systems solution** *(fast solvers, sub-structuring/EBE-iterative)*,
- **unconditionally** **stable => large time steps.**

Legendre polynomials:

- **diagonal** **system matrices,**
- **explicit** **schemes** *(in time)* **=> faster** *(in principle)*,
- **conditionally** **stable => time steps must honour CFL limit.**

Parallel computation
SEM methods

Domain partition

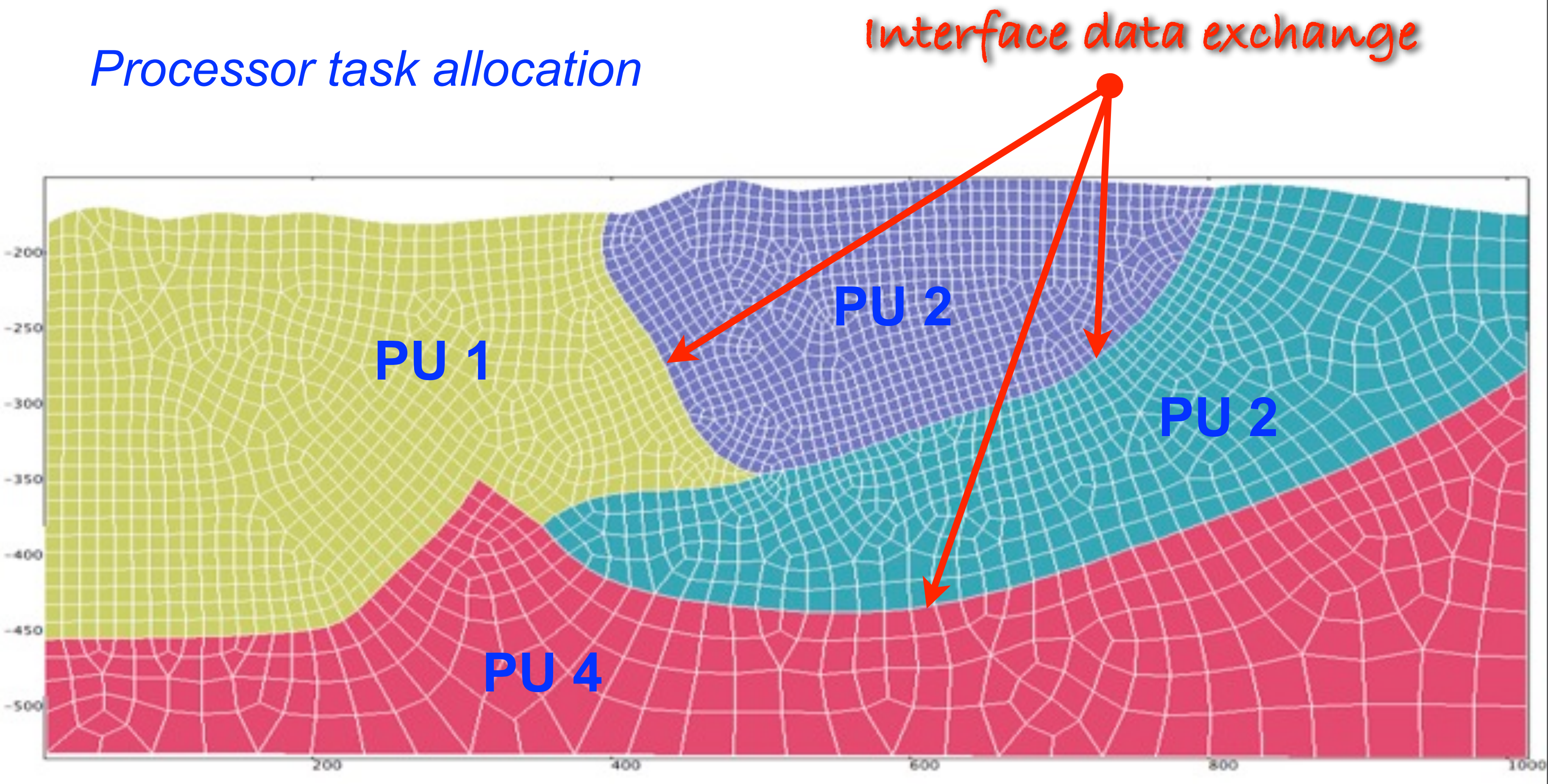


Processor task allocation

Interface data exchange

SEM partitioning

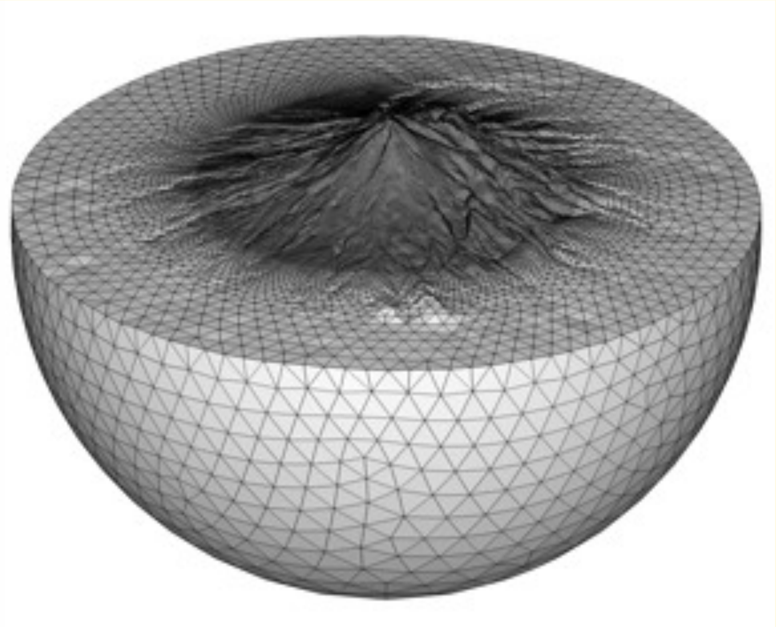
Processor task allocation

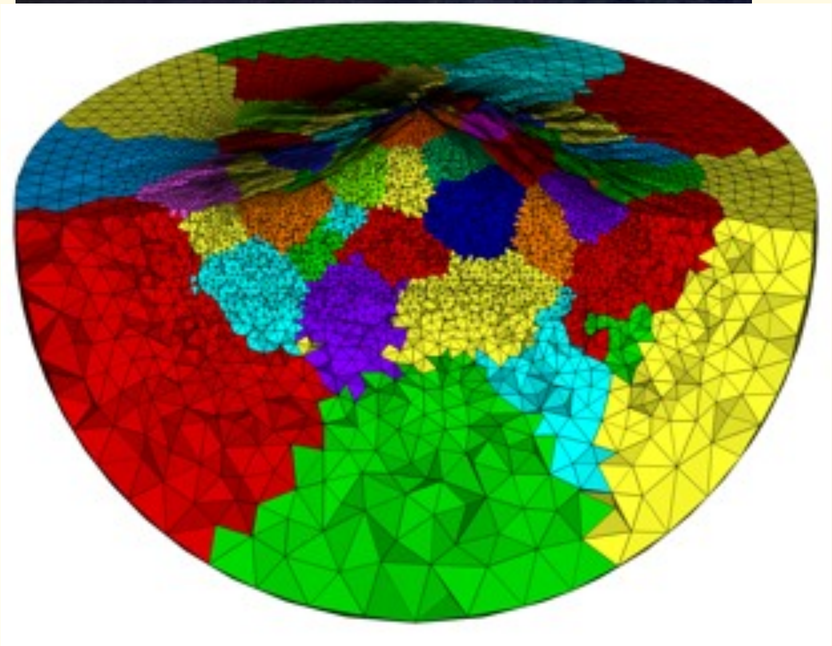


Discontinuous Galerkin(DG)Approach

Modeling of Scattered Waves in Merapi Volcano

(J. Wassermann)





- problem adapted mesh generation
- load balancing by partitioning & grouping subdomains (see Metis or Jostle software)

Parallel implementation

Very efficient parallel codes can be developed because in solving:

$$\boxed{\mathrm{M}\,\ddot{U} + \mathrm{K}\,U = F}$$

- most of calculation are based on linear algebra op.,
- efficient software exists like BLAS, LAPACK, ATLAS,
- global matrix-vector products can be easily computed at element level and back:

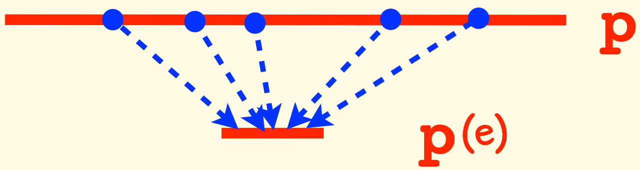
$$\mathbf{v} = \mathbf{S}\ \mathbf{p} \iff \boxed{\mathbf{v}^{(e)} = \mathbf{S}^{(e)}\mathbf{p}^{(e)}}$$

- element level op. can be done element-by-element (EBE) in any order or in parallel.

Parallel implementation

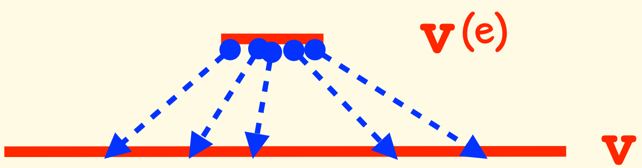
EBE algorithm

DO in parallel:



- **gather** data from global to local $\quad \mathbf{p} \Longrightarrow \mathbf{p}^{(e)}$,
- **local** matrix-vector multiply $\quad \mathbf{v}^{(e)} = \mathbf{S}^{(e)}\mathbf{p}^{(e)}$,
- **scatter** data from local to global $\quad \mathbf{v}^{(e)} \Longrightarrow \mathbf{v}$,
- **synchronize** data at subdomain interfaces by MPI call.

REPEAT for each step.



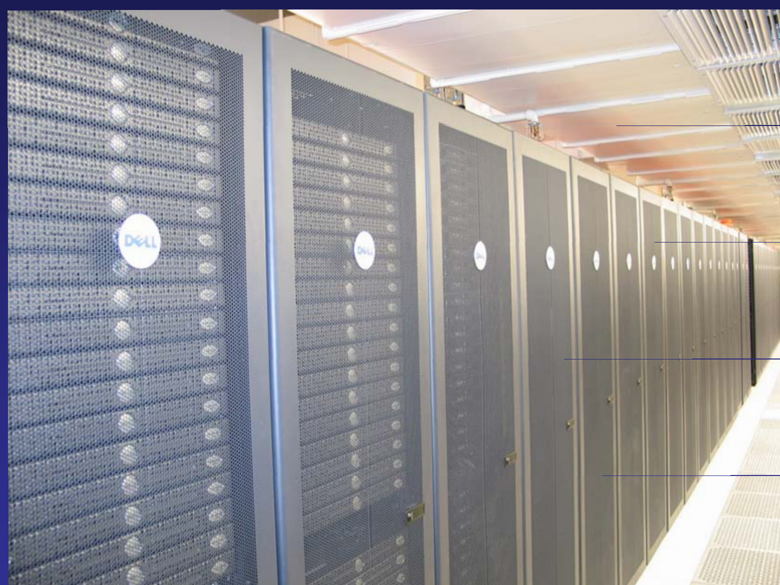
Global seismology

(D. Komatitsch, J. Tromp et al.)

SPECFEM3D

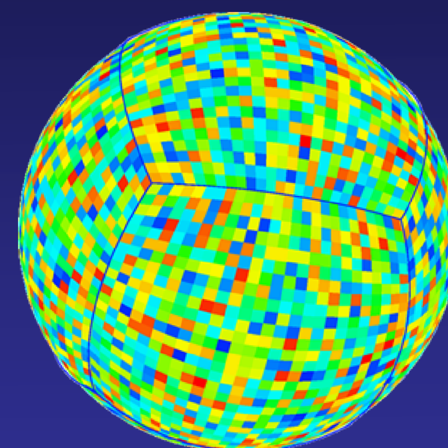
High-Performance Computing

GPS Dell cluster



“Old” machine:
- 1024 nodes/2048 processors
- 3 TB of distributed memory
- 13.1 TFLOPS

6 n^2 mesh slices



New machine:
- 512 dual-processor quad-core nodes (4096 cores)
- 6 TB of distributed memory
- 22.6 TFLOPS
- Half the cooling & power

Global seismology

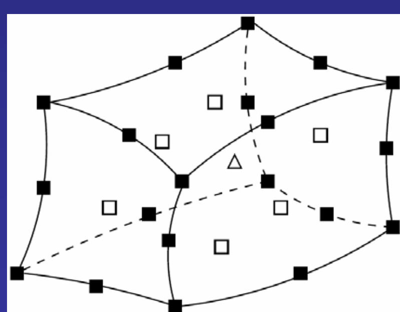
(D. Komatitsch, J. Tromp et al.)

SPECFEM3D

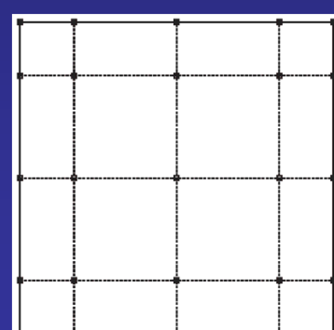
Spectral-Element Method

- Flexibility of the finite-element method
- Accuracy of a pseudospectral method
- Gauss-Lobatto-Legendre (GLL) quadrature
- Lagrange interpolants
- Diagonal mass matrix
- Explicit time integration

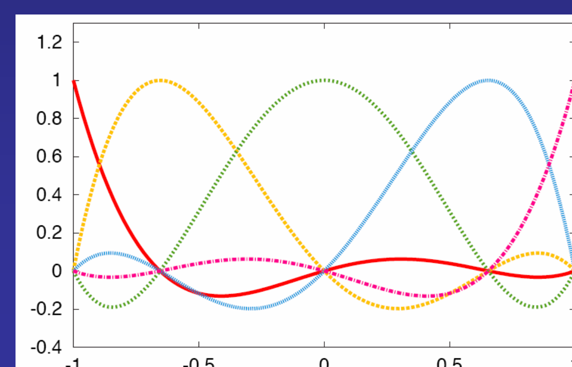
Hexahedral element



GLL integration points



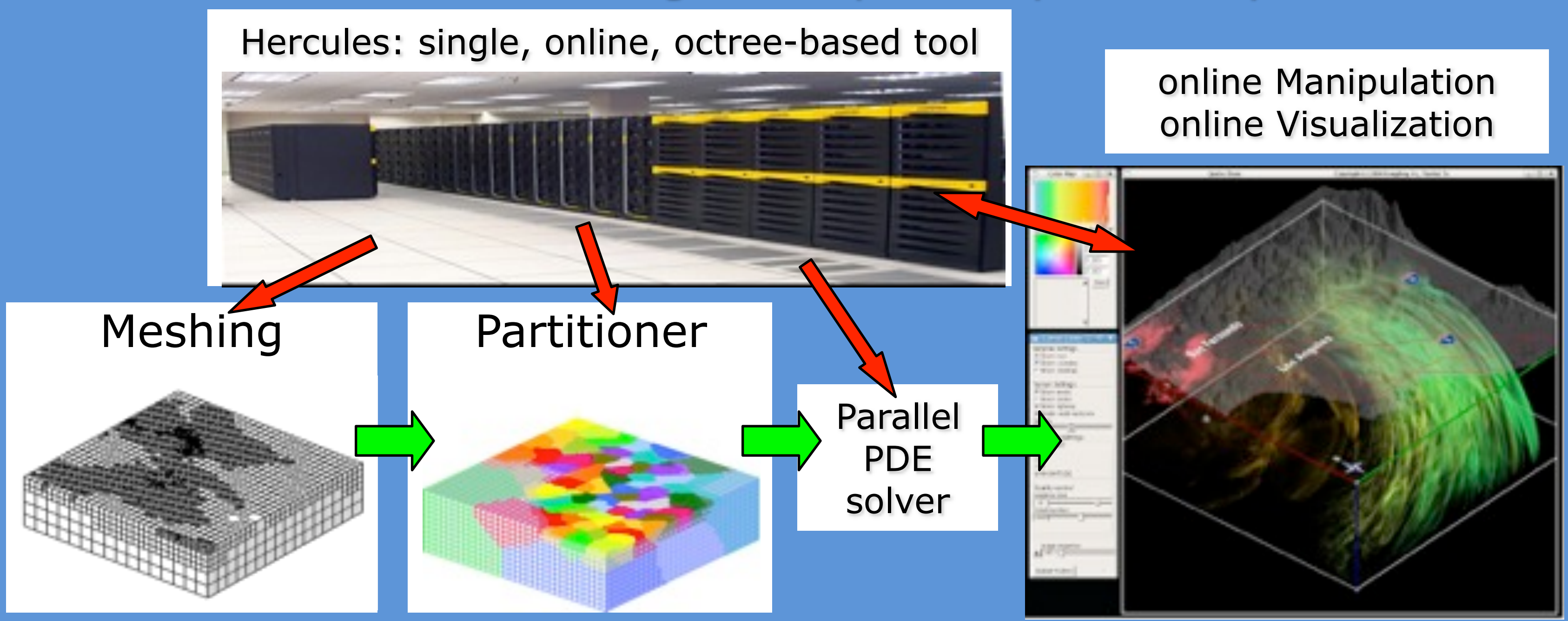
Lagrange polynomials



Octree-based scalable adaptive framework

Omar Ghattas (UT Austin), T. Tu, H. Yu (CMU)

- Avoid scalability bottlenecks (serial algorithms, file I/O)
- Integrate meshing, partitioning, simulation and visualization in a single parallel end-to-end application
- All operations are done in parallel and in-core (no I/O)
- Online remote steering of the parallel process by GUI



Prospects for Seismic Inversion on Petaflop Systems

This talk is about parallel computing.

Kilo – 1,000 Mega – 1,000,000 Giga – 1,000,000,000
Tera – 1,000,000,000,000 Peta – 1,000,000,000,000,000
Flop – "Floating Point Operation"

Example Earthquake Simulation:

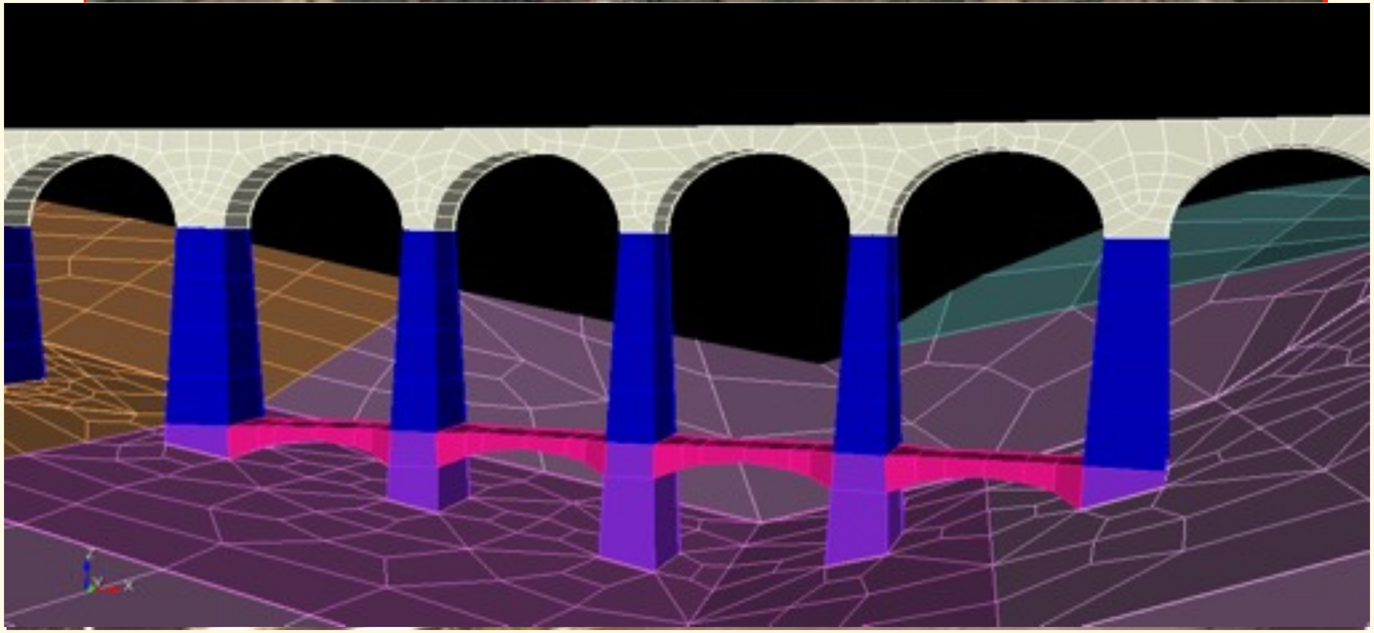
- 600km x 600km x 70km region
- Average velocity 2km/s, Max frequency 2Hz
- Wavelength 1km - 25 million grid boxes
- 10 variables per wavelength - 25 billion variables
- Simulated waves travel 400km - 4000 time steps
- Executing on average 10 arithmetic operations per variable

25 billion x 4000 x 10 = 10^15 → 1 Petaflop

Examples

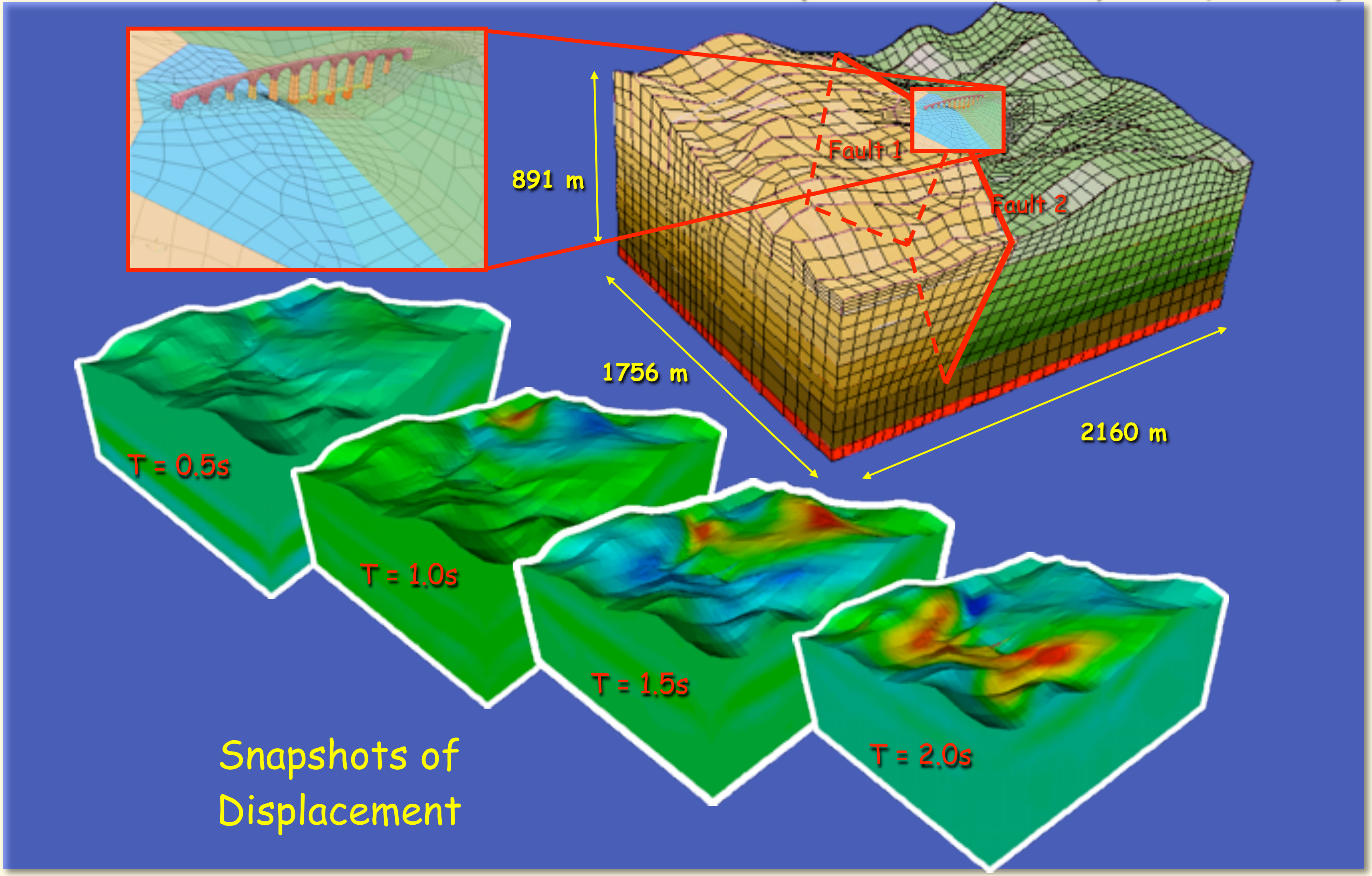
Acquansanta viaduct

(GEOElse - M. Stupazzini, Polimi)



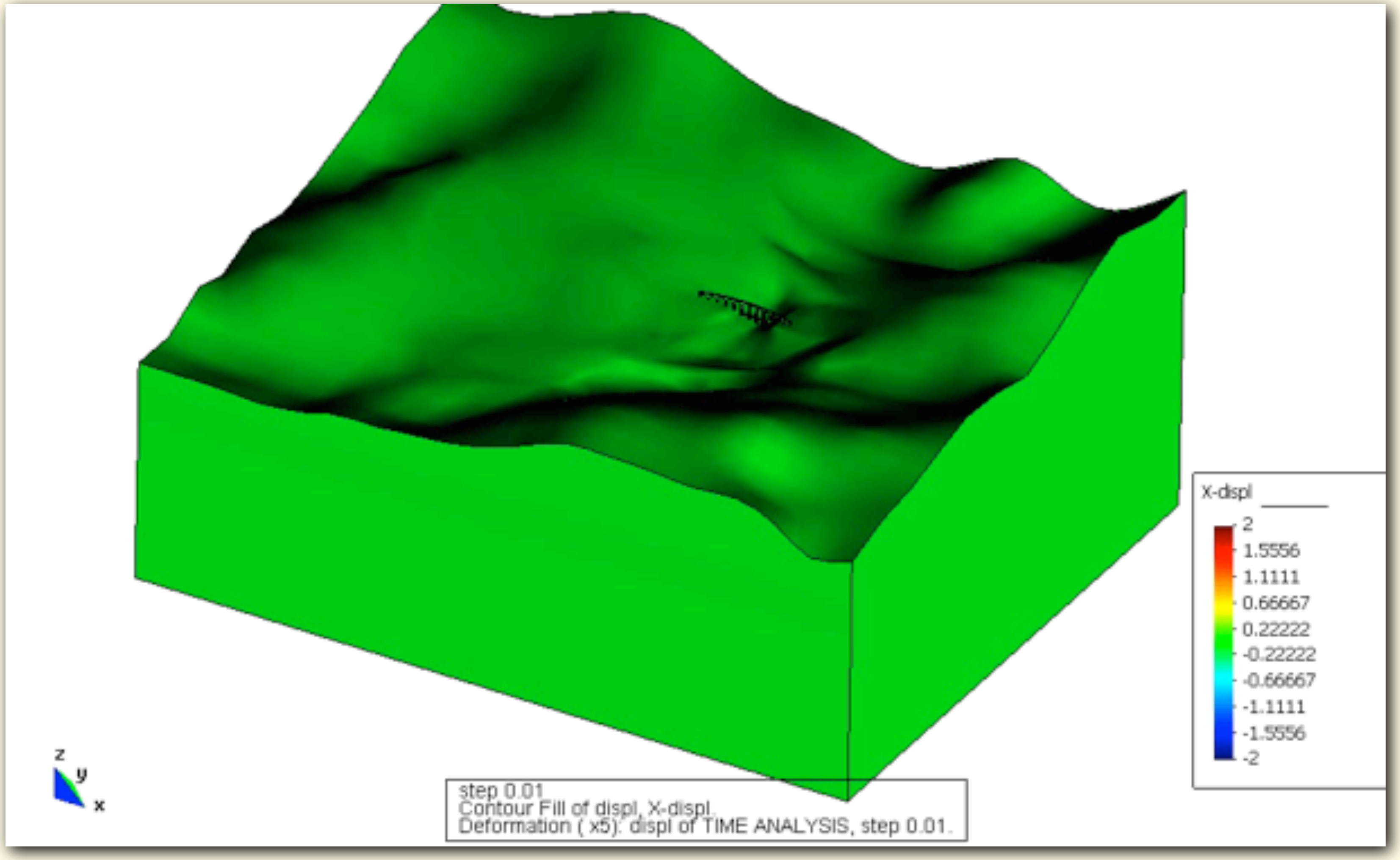
Acquansanta viaduct

(GEOElse - M. Stupazzini, Polimi)



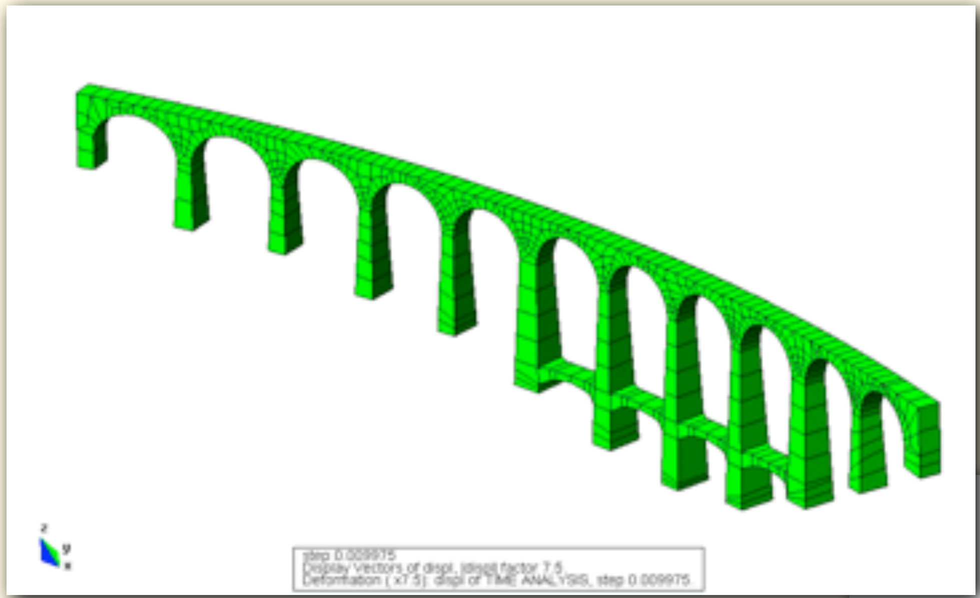
Acquansanta viaduct

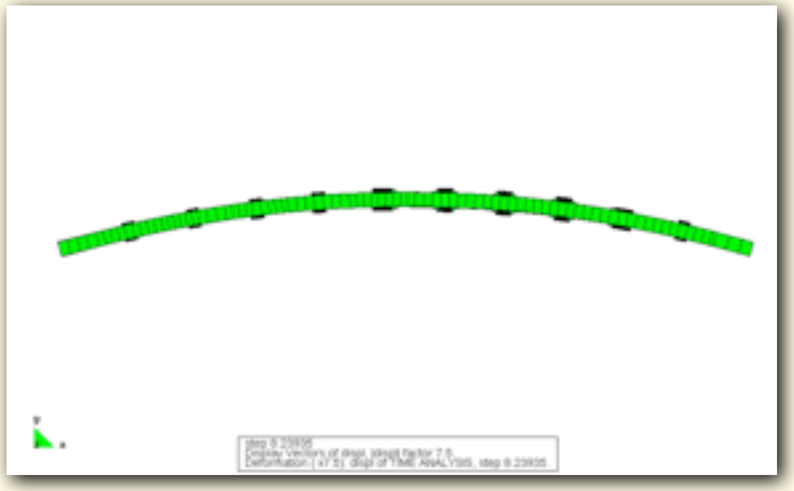
(GEOElse - M. Stupazzini, Polimi)

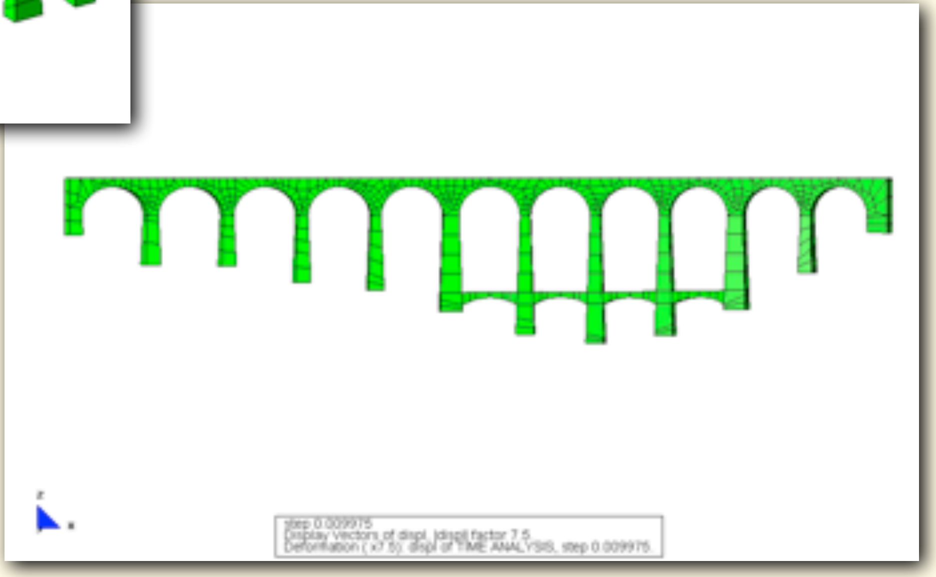


Acquansanta viaduct

(GEOElse - M. Stupazzini, Polimi)







Thank
for your
attention !