Numerical Simulation Methods in Geophysics, Part 11: Solving problems

1. MGPY+MGIN

thomas.guenther@geophysik.tu-freiberg.de

Recap

1. Types of PDEs in geophysics
2. The Finite Difference (FD) method
    - Poisson equation in 1D, look into 2D/3D
    - Heat transfer (diffusion equation) in 1D, time-stepping
3. Solving the hyperbolic (acoustic) wave equation in 1D
4. The Finite Element (FE) method
    - Poisson and diffusion equation in 1D
    - (complex) Helmholtz equation in 2D for EM problems

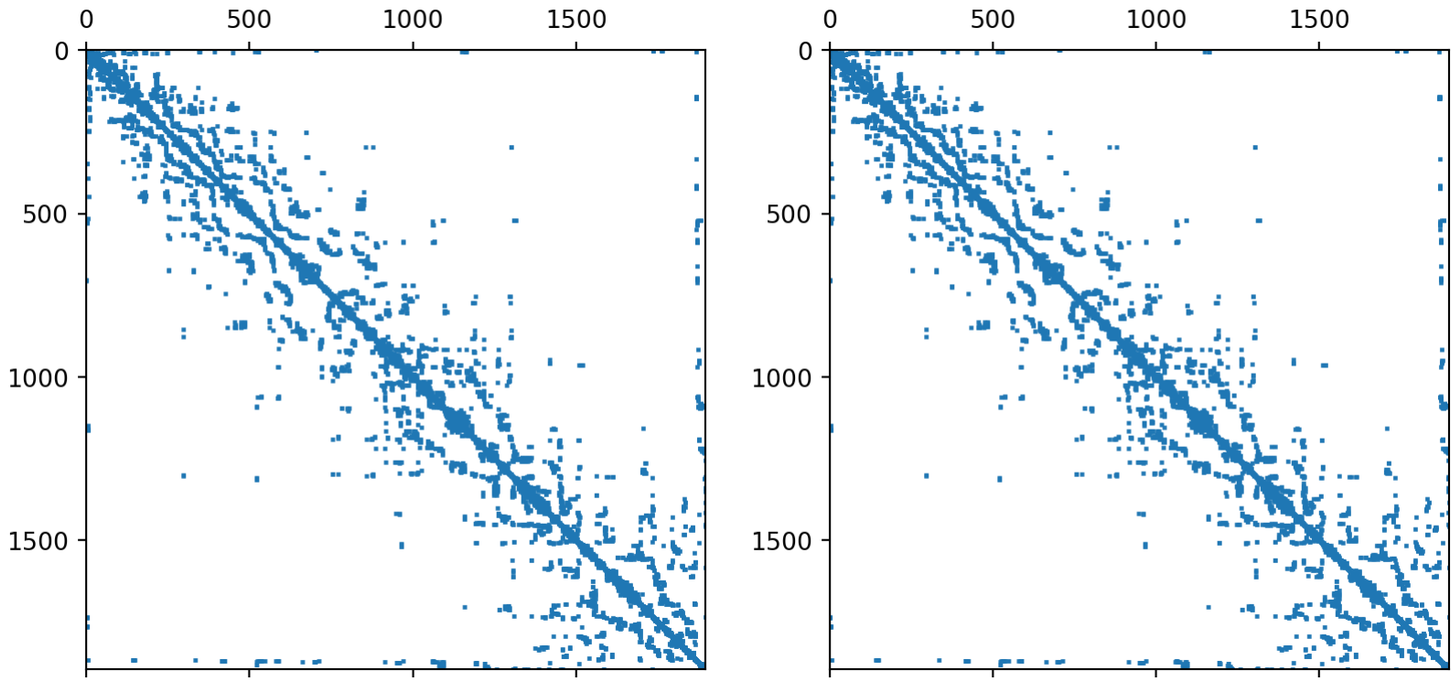
Topics yet to be covered

- solving vectorial EM problems in 2D/3D
  - staggered grid techniques 2D/3D
  - 2.5D solution techniques
  - 3D vector solution using Nedelec elements
- The Finite Volume (FV) method
- solving advection-dispersion problems
- equation solvers and high-performance computing

Three lectures (22/1, 29/1, 5/2) and exercises (23+30/1, 6/2)

The solver module

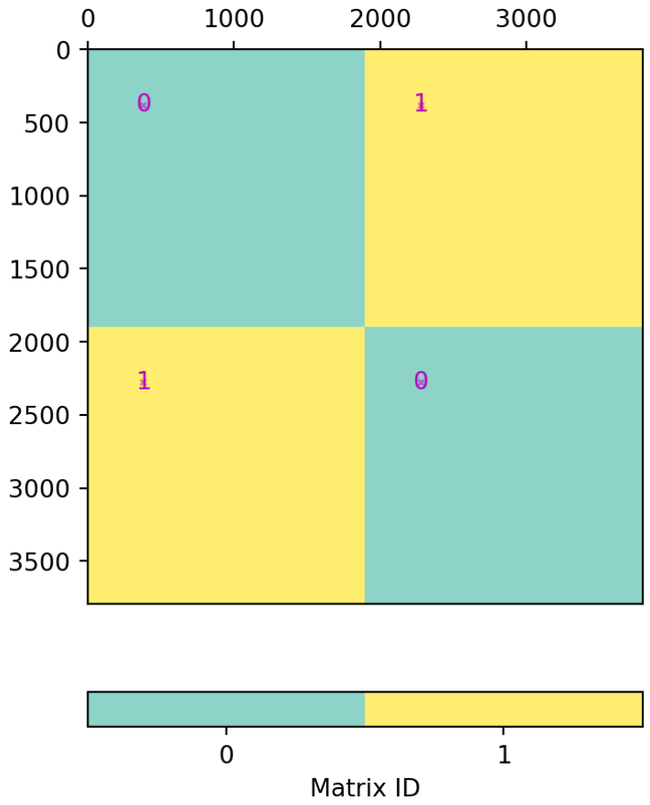
```
1  import pygimli.solver as ps
2  mesh["my"] = 4 * np.pi * 1e-7
3  A = ps.createStiffnessMatrix(mesh, a=1/mesh["my"])
4  M = ps.createMassMatrix(mesh, mesh["sigma"])
5  fig, ax = plt.subplots(ncols=2)
6  ax[0].spy(pg.utils.toCSR(A), markersize=1)
7  ax[1].spy(pg.utils.toCSR(M).todense(), markersize=1)
```



The complex problem matrix

$$\mathbf{B} = \begin{pmatrix} \mathbf{A} & -\omega\mathbf{M} \\ \omega\mathbf{M} & \mathbf{A} \end{pmatrix}$$

```
 1  w = 0.1
 2  nd = mesh.nodeCount()
 3  B = pg.BlockMatrix()
 4  B.Aid = B.addMatrix(A)
 5  B.Mid = B.addMatrix(M)
 6  B.addMatrixEntry(B.Aid, 0, 0)
 7  B.addMatrixEntry(B.Aid, nd, nd)
 8  B.addMatrixEntry(B.Mid, 0, nd, scale=-w)
 9  B.addMatrixEntry(B.Mid, nd, 0, scale=w)
10  pg.show(B)
```



Sparse matrices

Up to now: regular (dense) array: save every element including 0

Save only non-zero components (e.g. using `scipy.sparse`)

- COO - coordinate format
- CSC/CRS - compressed sparse column/row
- BSR - block sparse row format, …

Equation solvers

Solve systems of equations

direct solvers

- Gauss elimination (expensive and dense)
- Cholesky (or ILU) decomposition

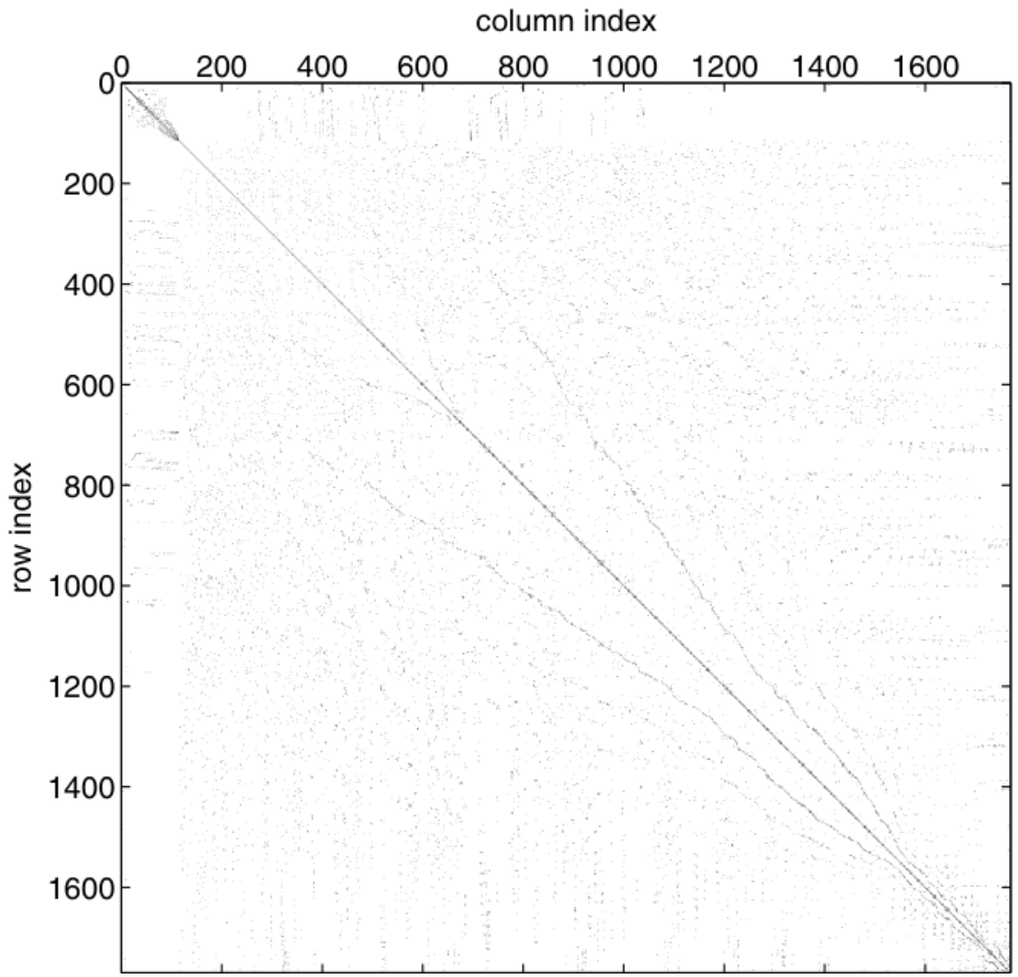
indirect solvers (fixed point iteration)

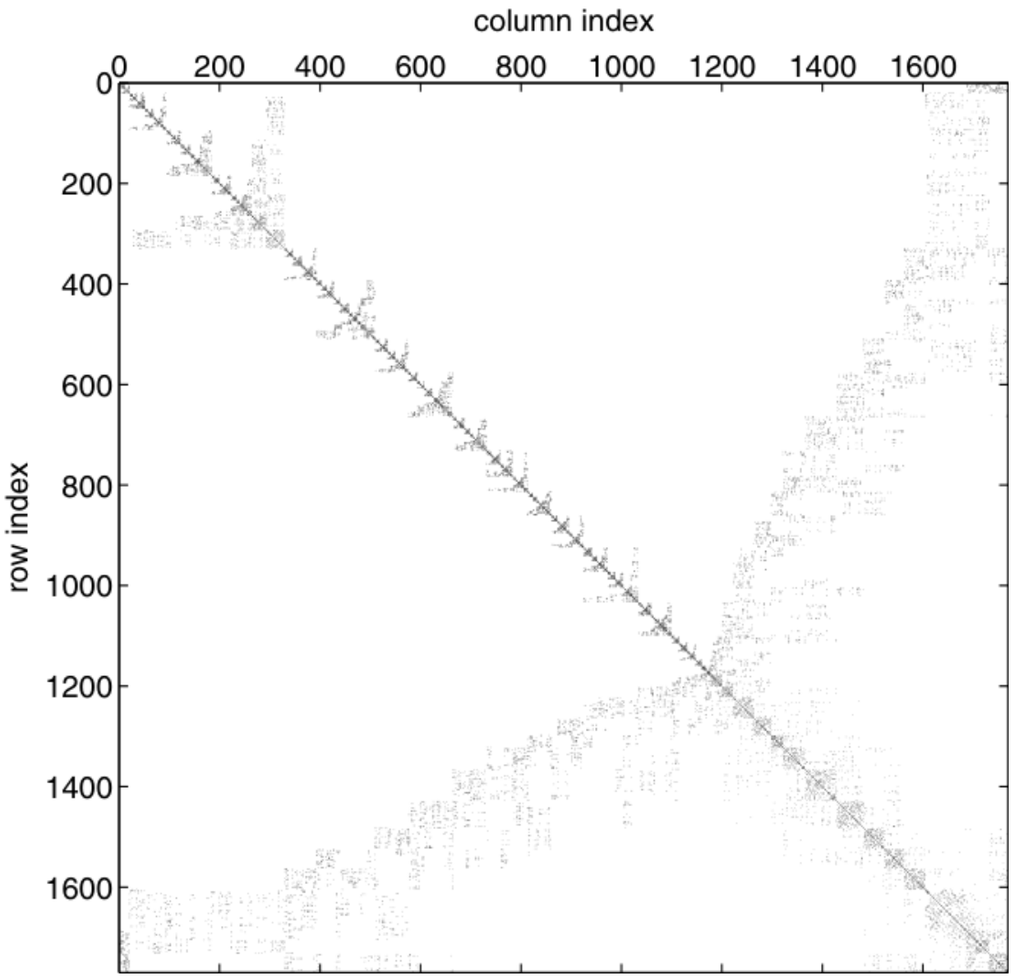
- gradient methods: steepest descent, conjugate gradients
- preconditioning: incomplete factorizations (of submatrices)

Cholesky decomposition

$$\mathbf{A} = \mathbf{L}\mathbf{L}^T \quad \text{or} \quad \mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T$$

Reordering





original (left) & reordered (right) matrix (Rücker et al. 2006)

Iterative solvers - Fixpunktverfahren

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

decompose $\mathbf{A} = \mathbf{M} - \mathbf{N}$ and solve

$$\mathbf{M}\mathbf{x} = \mathbf{N}\mathbf{x} + \mathbf{b}$$

iterativily by $\mathbf{x}^{k+1} = \mathbf{M}^{-1}\left(\mathbf{N}\mathbf{x}^{k} + \mathbf{b}\right)$

e.g. $\mathbf{M} = \mathrm{diag}(\mathbf{A})$ (Jacobi method) or
$\mathbf{M} = \mathrm{tril}(\mathbf{A})$ (Gauss-Seidel method)

Gradient methods

minimize residual $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$

Steepest descent method

$$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$$

go towards $\mathbf{r}_0$ and minimize step length α

$$\mathbf{r}_1^T \mathbf{r}_0 = 0 = \mathbf{b} - \mathbf{A}(\mathbf{x}_0 + \alpha \mathbf{r}_0)$$

$$\Rightarrow \alpha = \frac{\mathbf{r}_0^T \mathbf{r}_0}{\mathbf{r}_0 \mathbf{A} \mathbf{r}_0}$$

Steepest descent algorithm

```
for i = 0,1, ...
```

$$\mathbf{r}_i = \mathbf{b} - \mathbf{A}\mathbf{x}_i$$

$$\Rightarrow \alpha_i = \frac{\mathbf{r}_i^T \mathbf{r}_i}{\mathbf{r}_i \mathbf{A} \mathbf{r}_i}$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{r}_i$$

Conjugate directions

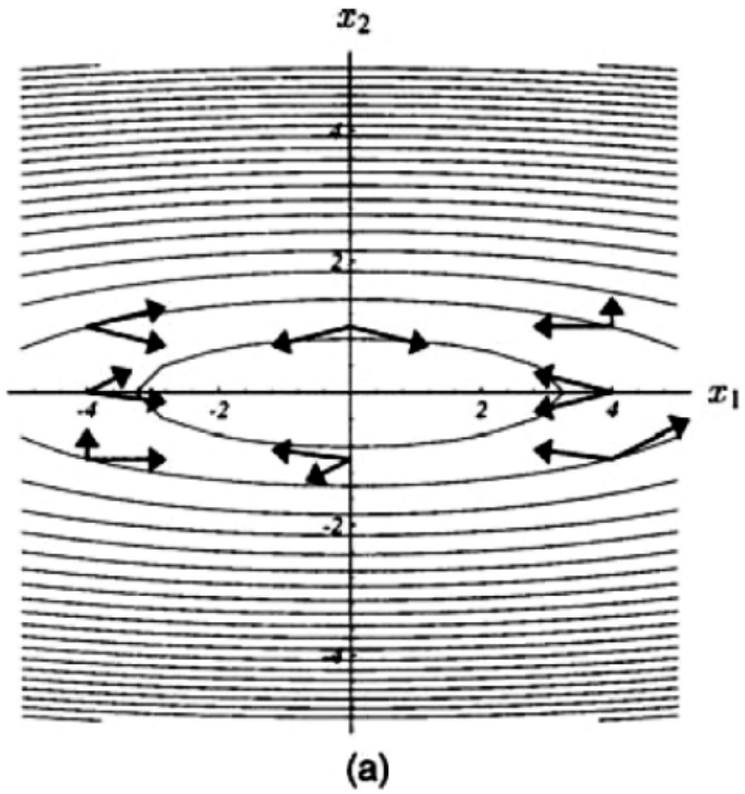
Set of orthogonalen directions $\mathbf{d}$ with

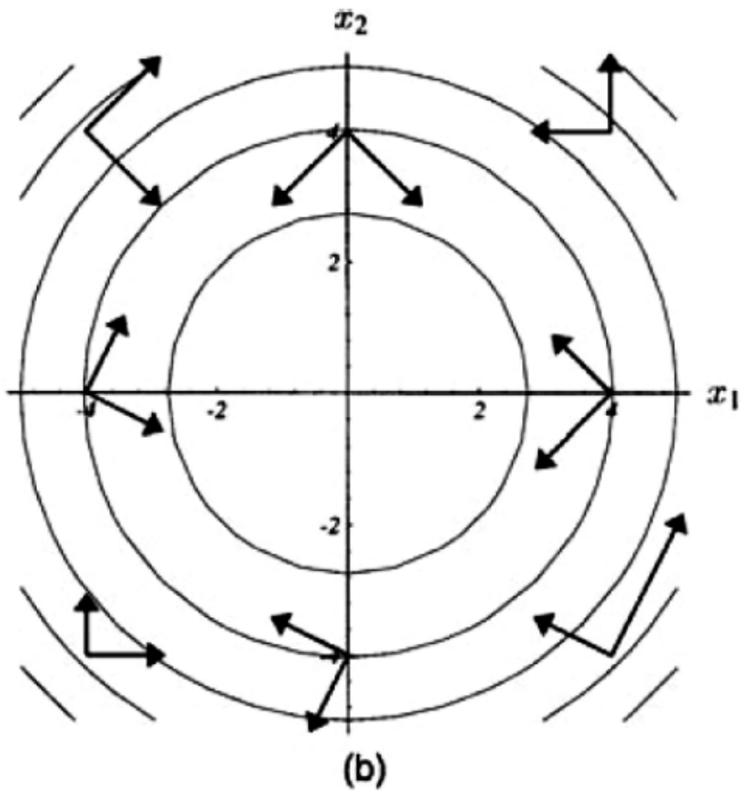
$$\mathbf{d}_i^T \mathbf{A} \mathbf{d}_j = 0 \qquad \forall i \neq j$$

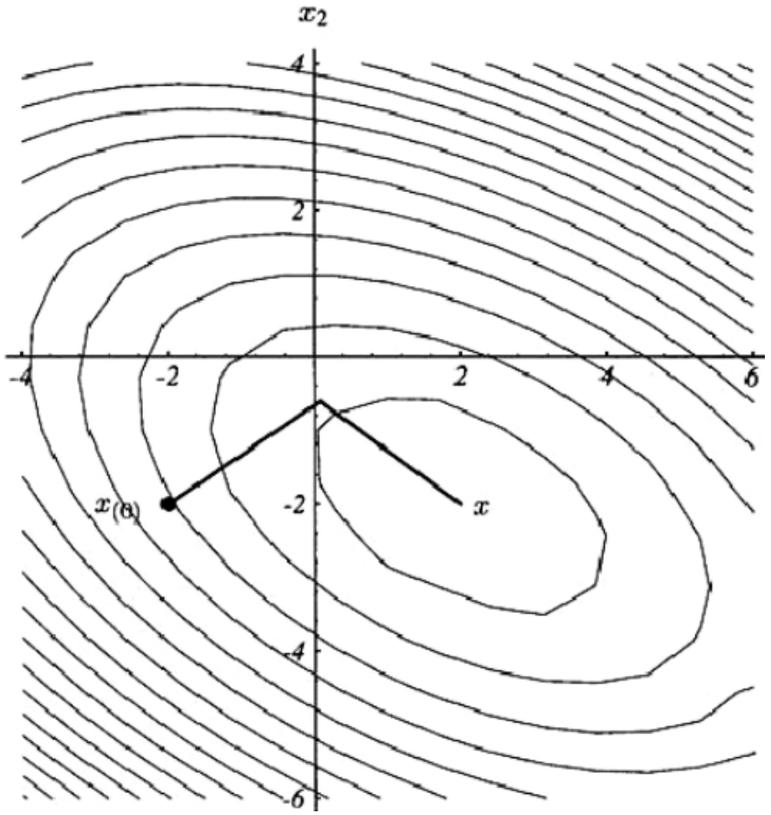
> **Note**
>
> every search direction is only used once

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{d}_i$$

Conjugate directions







Conjugate gradient (Hestenes&Stiefel, 1952)

$$\mathbf{d}_0 = \mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$$

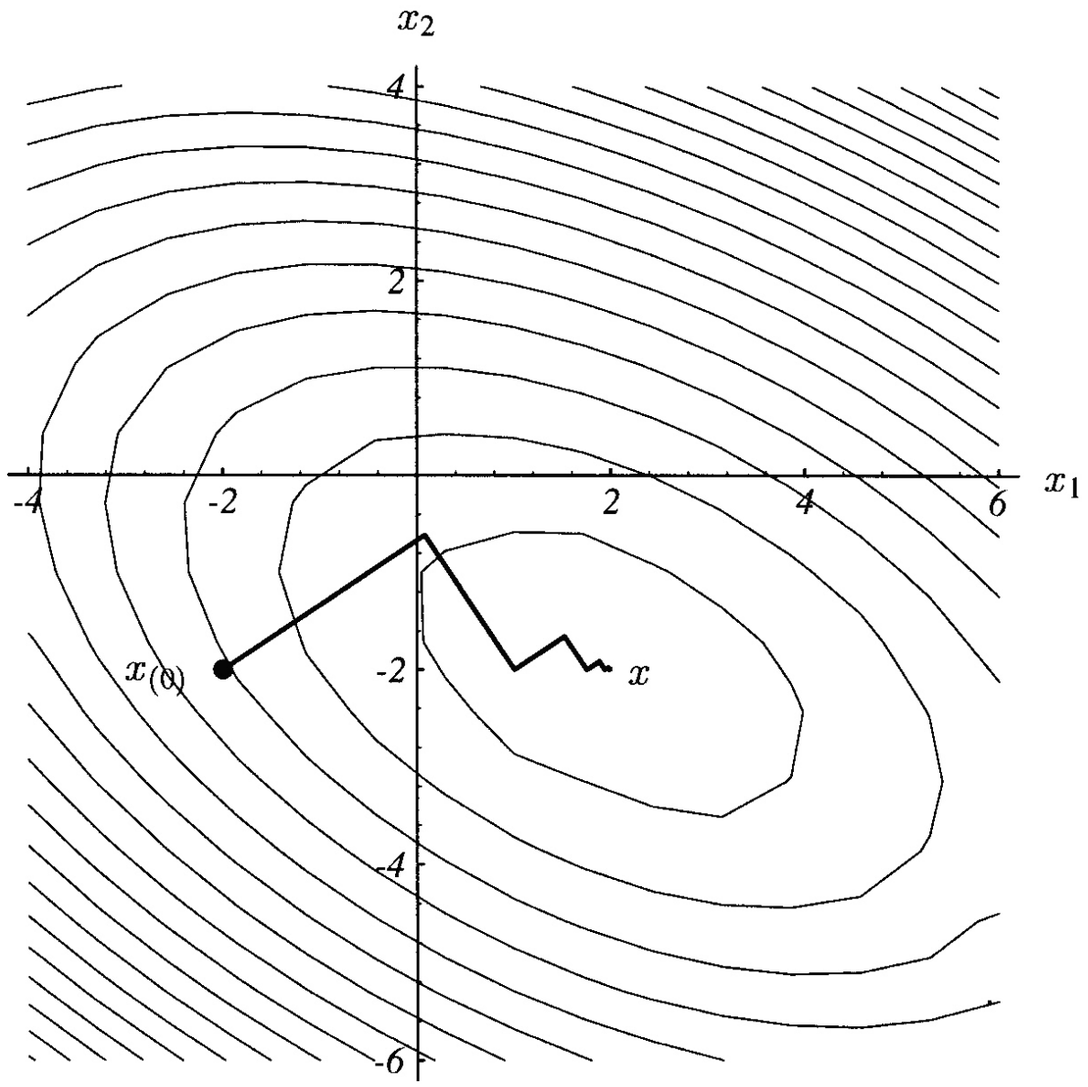
$$\alpha_i = \frac{\mathbf{r}_i^T \mathbf{r}_i}{\mathbf{d}_i \mathbf{A} \mathbf{d}_i}$$

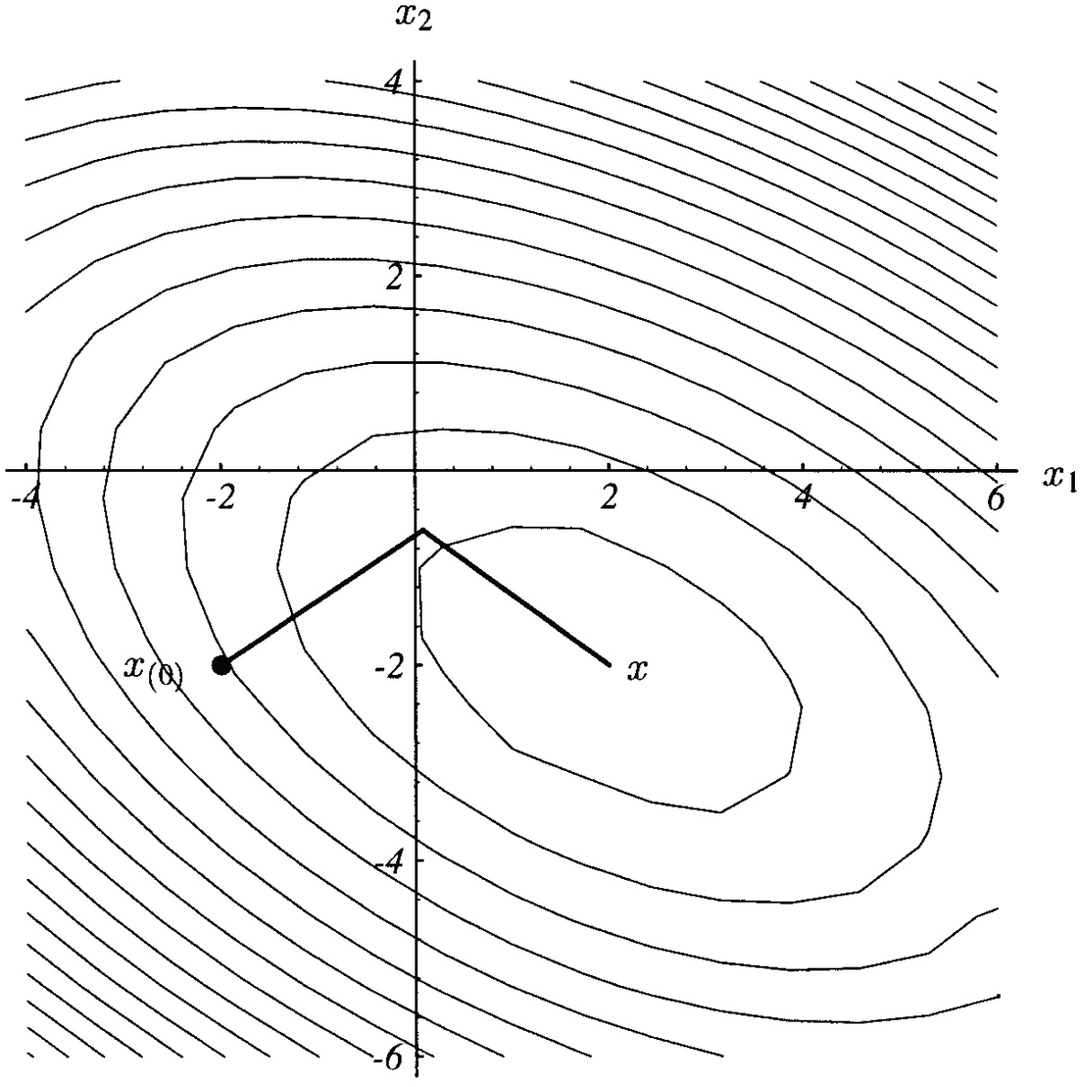
$$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{d}_i$$

$$\mathbf{r}_{i+1} = \mathbf{r}_i - \alpha_i \mathbf{A} \mathbf{d}_i$$

$$d_{i+1} = \mathbf{r}_{i+1} + \mathbf{d}_{i+1} \frac{\mathbf{r}_{i+1}^T \mathbf{r}_{i+1}}{\mathbf{r}_i^T \mathbf{r}_i}$$

Steepest descent vs. conjugate gradient





Preconditioning

$\mathbf{Ax} = \mathbf{b}$ often badly conditioned (elongated) $\Rightarrow$ slow convergence

Idea: transform (precondition) equation system by preconditioner $\mathbf{K}$

$$\mathbf{K}^{-1}\mathbf{Ax} = \mathbf{K}^{-1}\mathbf{b}$$

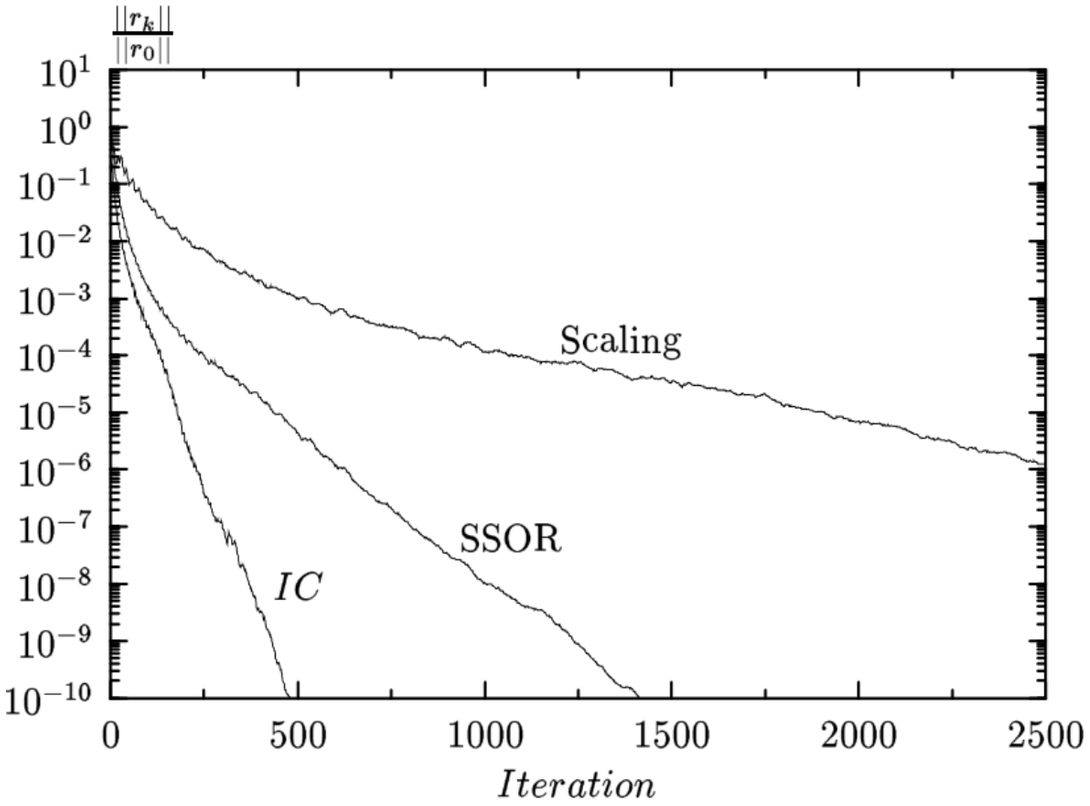
Extreme cases: 1. $\mathbf{K} = \mathbf{I}$, cheap PC but no gain

2. $\mathbf{K} = \mathbf{A}$, perfect conditioning but expensive PC
   e.g. $\mathbf{K} = \mathrm{diag}A$ or $\mathbf{K} = \epsilon\mathrm{tril}A$

Incomplete Cholesky decomposition

$$\mathbf{A} \approx \mathbf{L}\mathbf{L}^T$$

with certain accuracy or sparsity



Preconditioning

EM problem

$$\mathbf{B} = \begin{pmatrix} \mathbf{A} & -\omega\mathbf{M} \\ \omega\mathbf{M} & \mathbf{A} \end{pmatrix}$$

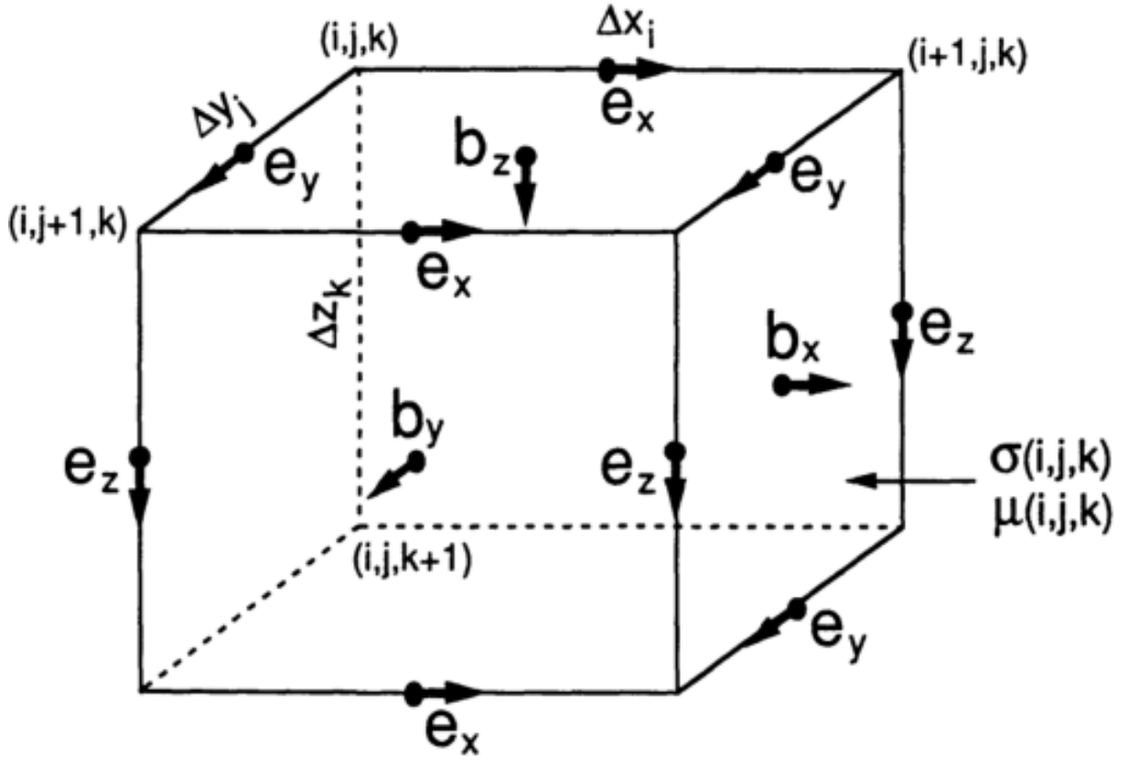
$$\mathbf{K} = \begin{pmatrix} \mathbf{A} + \omega\mathbf{M} & 0 \\ 0 & \mathbf{A} + \omega\mathbf{M} \end{pmatrix}$$

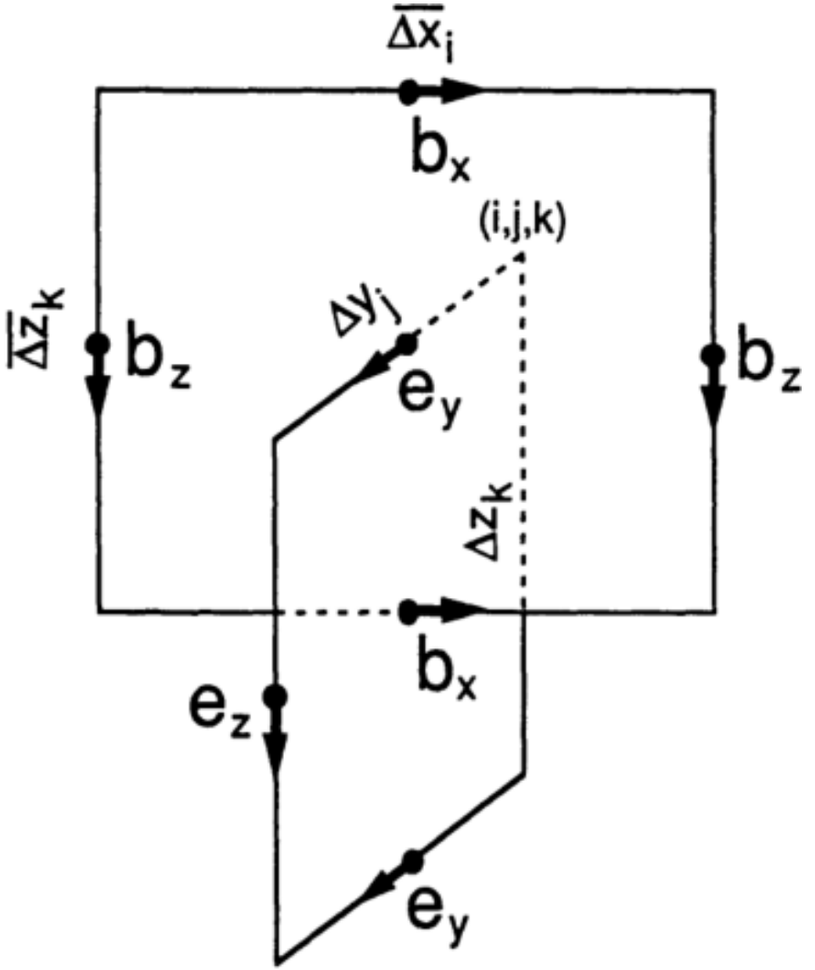
or Schur complement. E.g., hold factorization of $\mathbf{A}$ and $\mathbf{M}$ in memory

Development of EM modelling

- approaches: integral equations (IE), finite differences (FD) and elements (FE)
- decompose 3D source (2D inverse) problems in wavenumber domains
- improvement of equation solvers and preconditioners
-

Solving EM problems with staggered grid





Staggered grid cell after Yee (1966)

The way from FD to FE

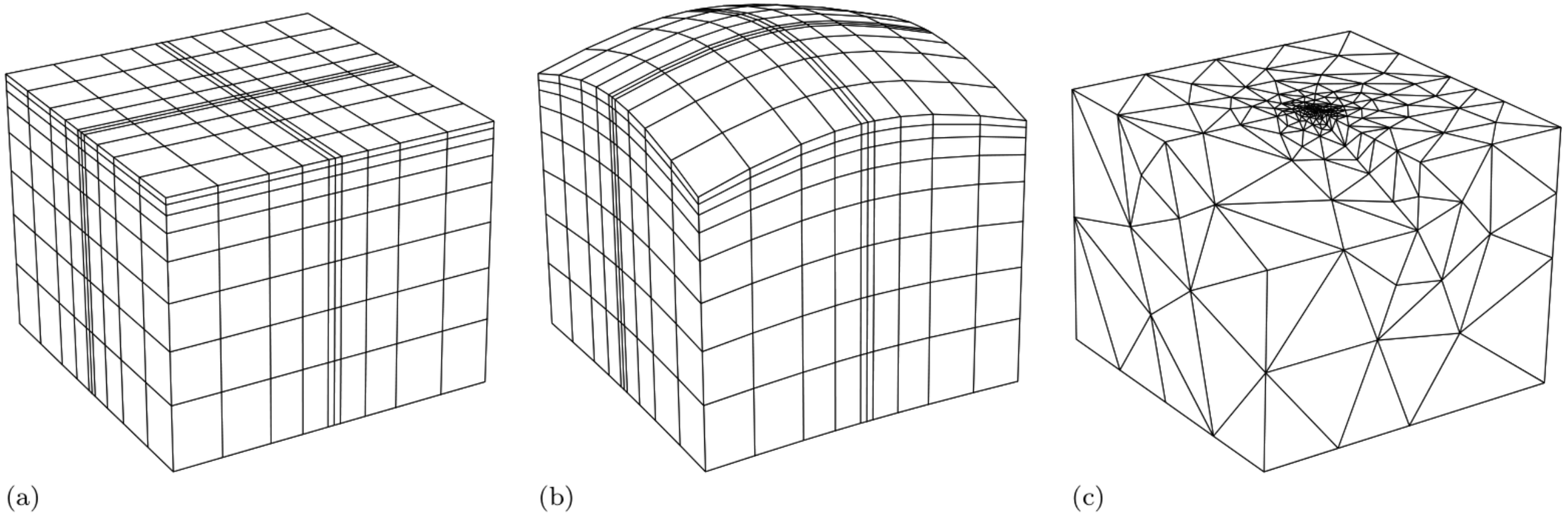
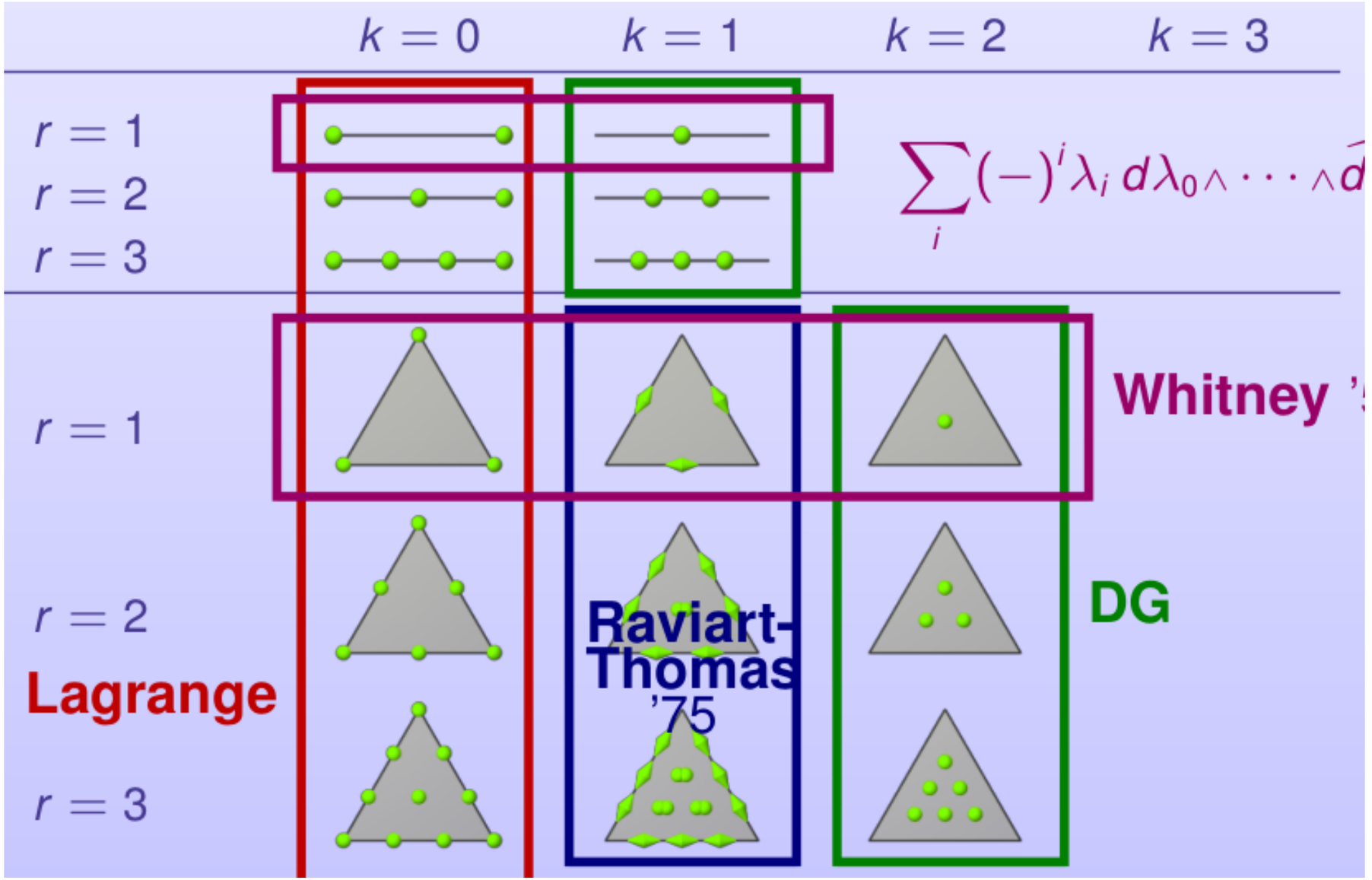


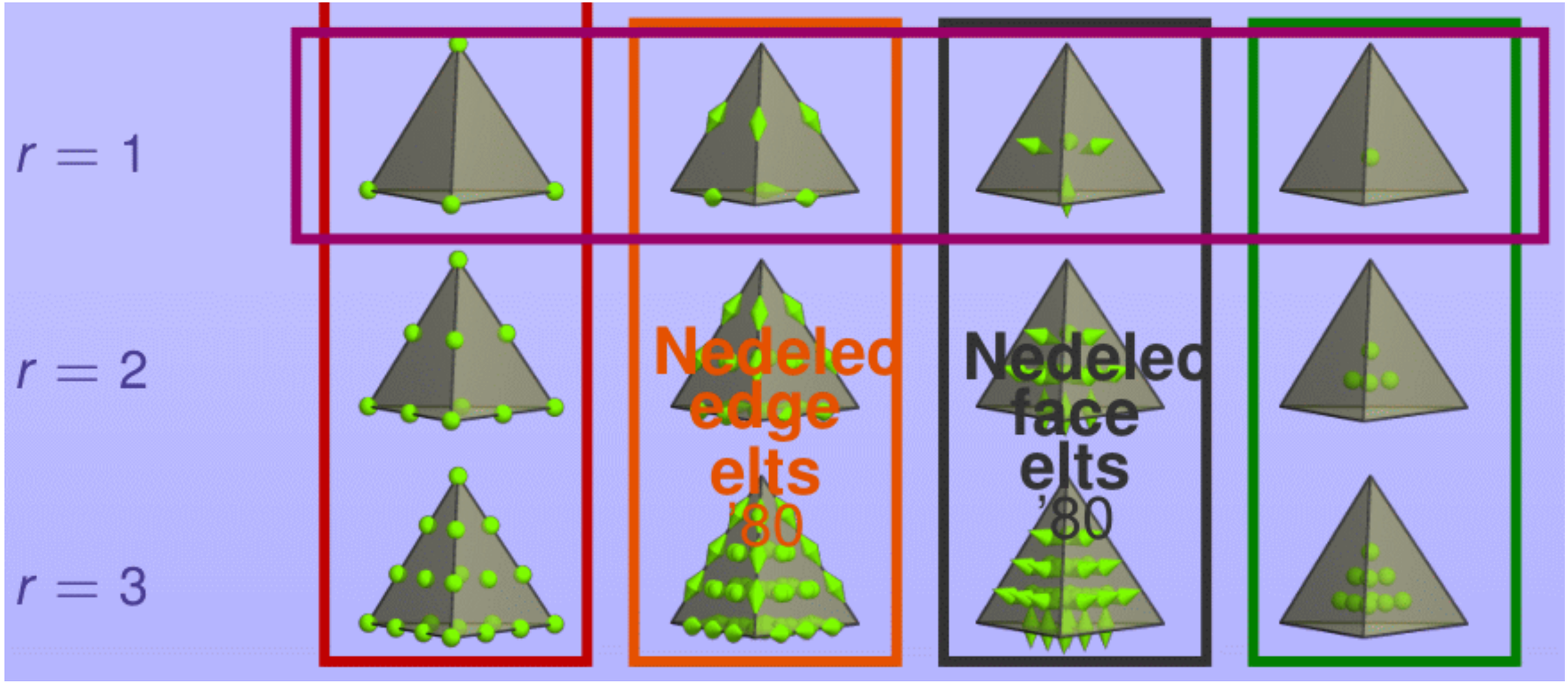
Figure 1. Different grid types: Orthogonal cuboid (a), non-orthogonal hexahedral (b) and unstructured tetrahedral (c) grid.

Orthogonal, non-orthogonal and irregular grids (Rücker et al., 2006)

The Finite Element zoo (1D & 2D)

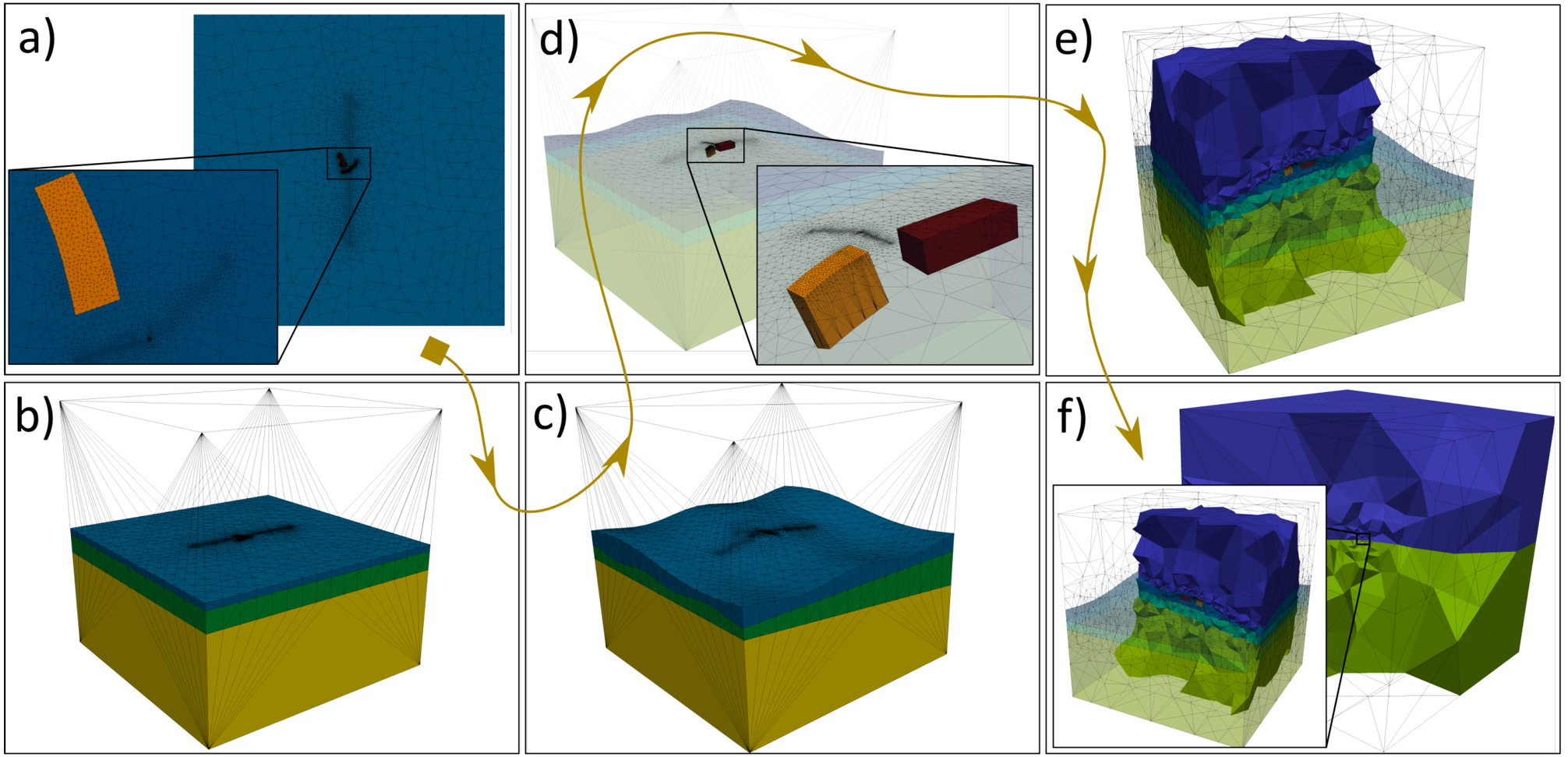


The Finite Element zoo (3D)



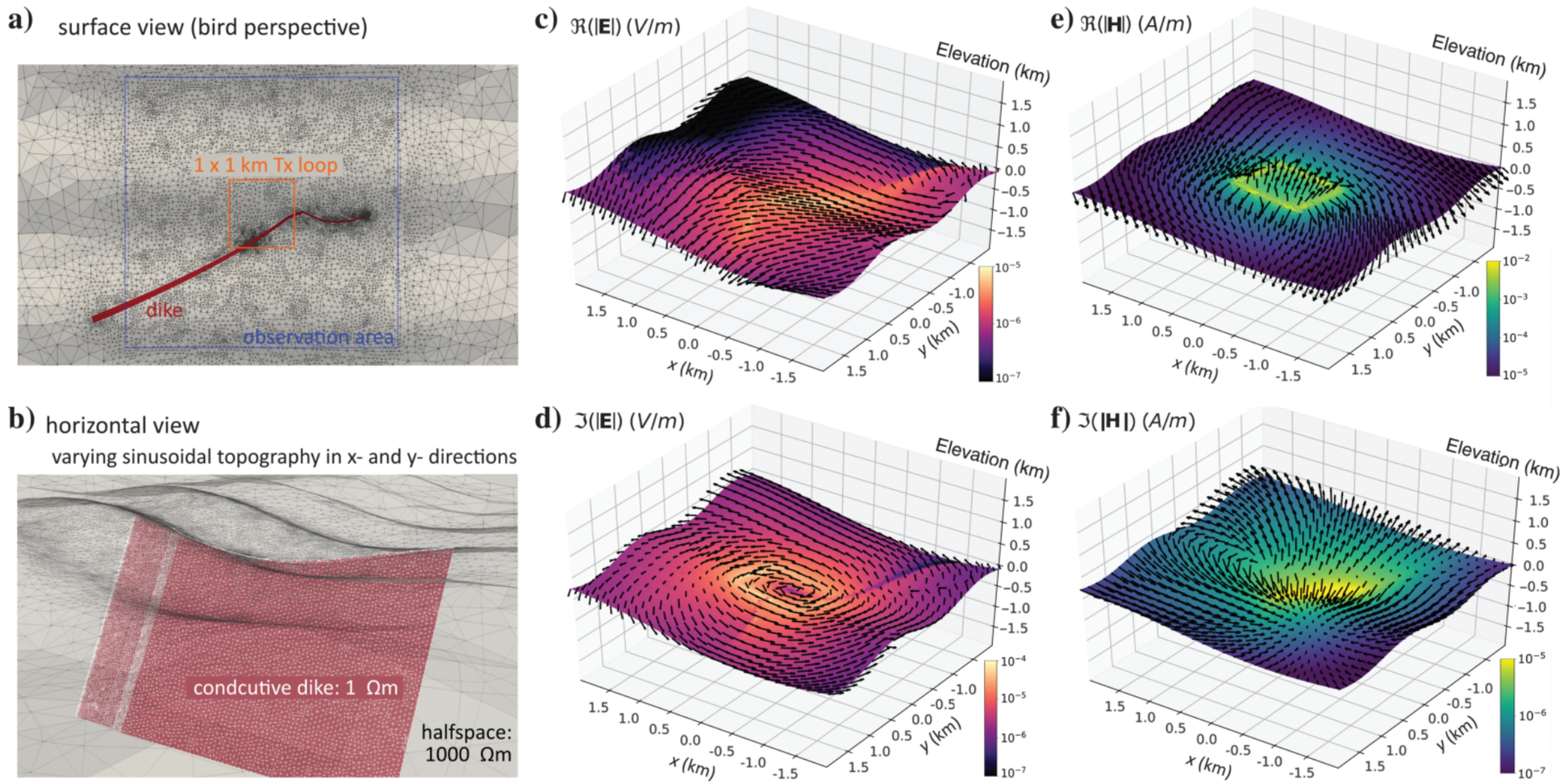
Arnold, Periodic table of elements

Meshing complicated geometries



Meshing workflow in custEM (Rochlitz et al., 2019)

Modelling example



Modelling example of a conductive dike (Rochlitz et al., 2019)

Packages

Mesh generation: TetGen (3D), GMsh (2D/3D)

FE packages: FEniCS, NETGEN/NGsolve

Equation solvers: Suitesparse, MUMPS, SciPy

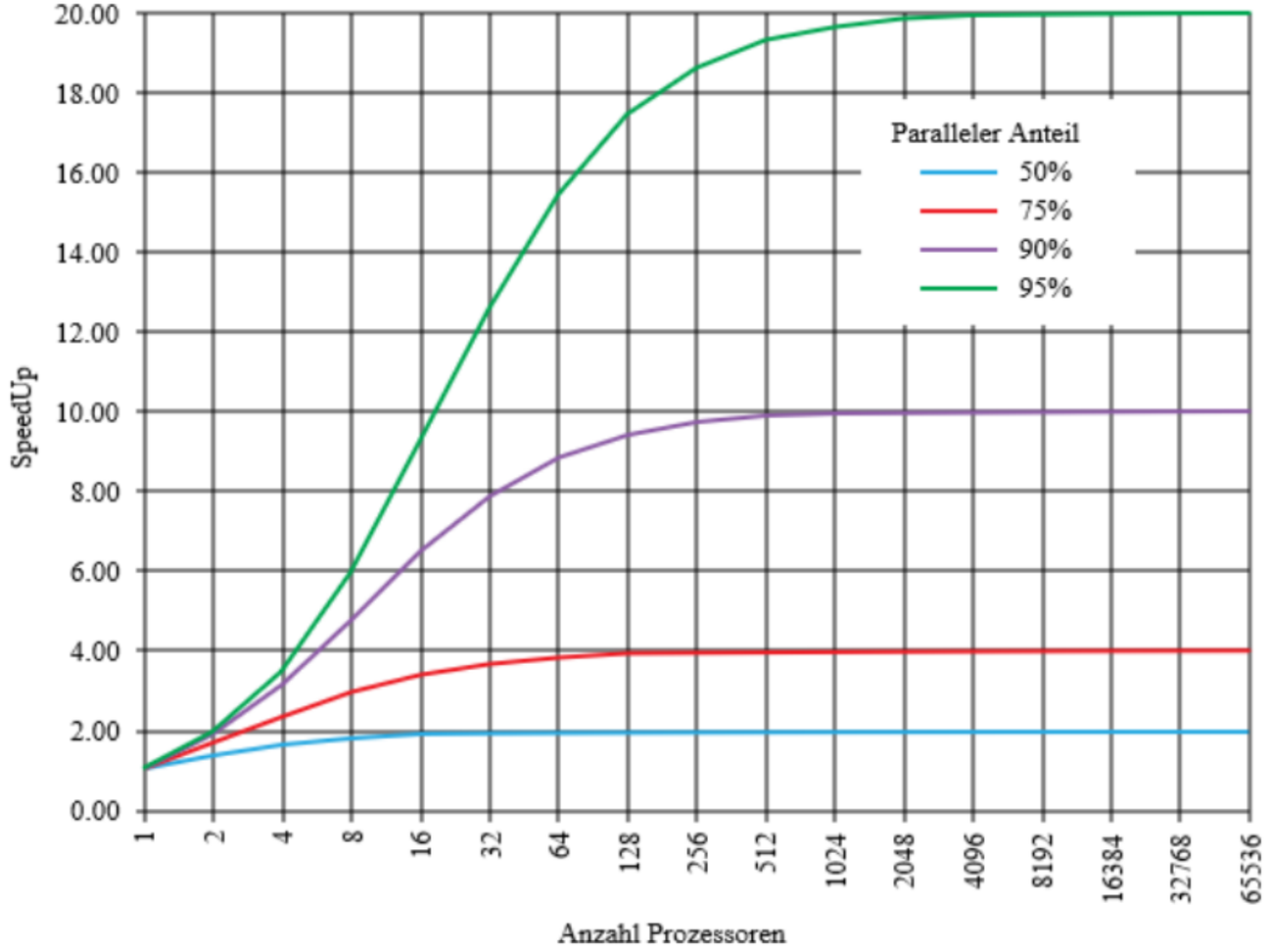
Computational frameworks: PetSc, MPI

EM modelling (and inversion) packages: Mare2DEM, emg3d, GoFEM, PETGEM, custEM, SimPEG, ModEM, FEMTIC

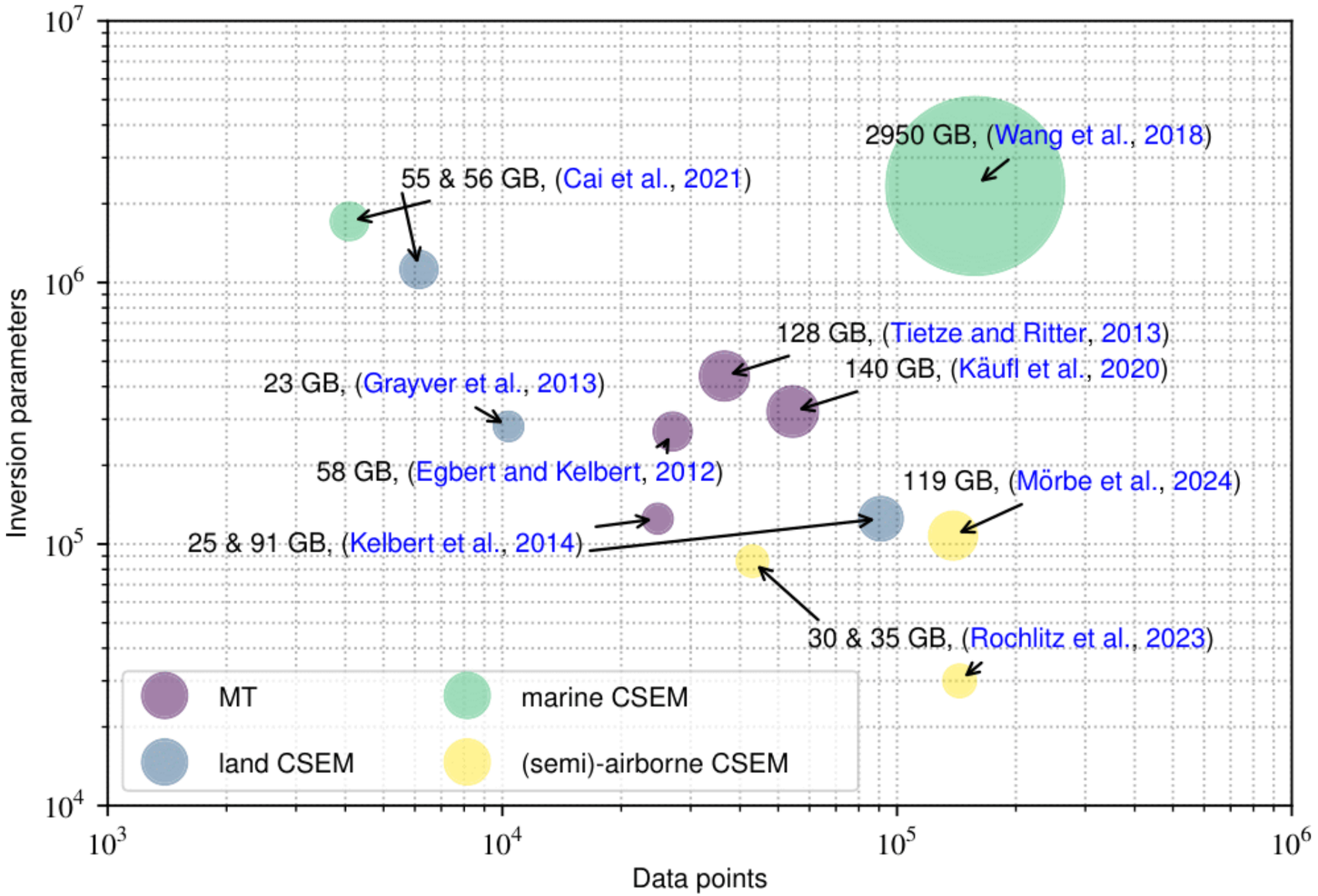
Performance on Parallel machines

Amdahlsches Gesetz

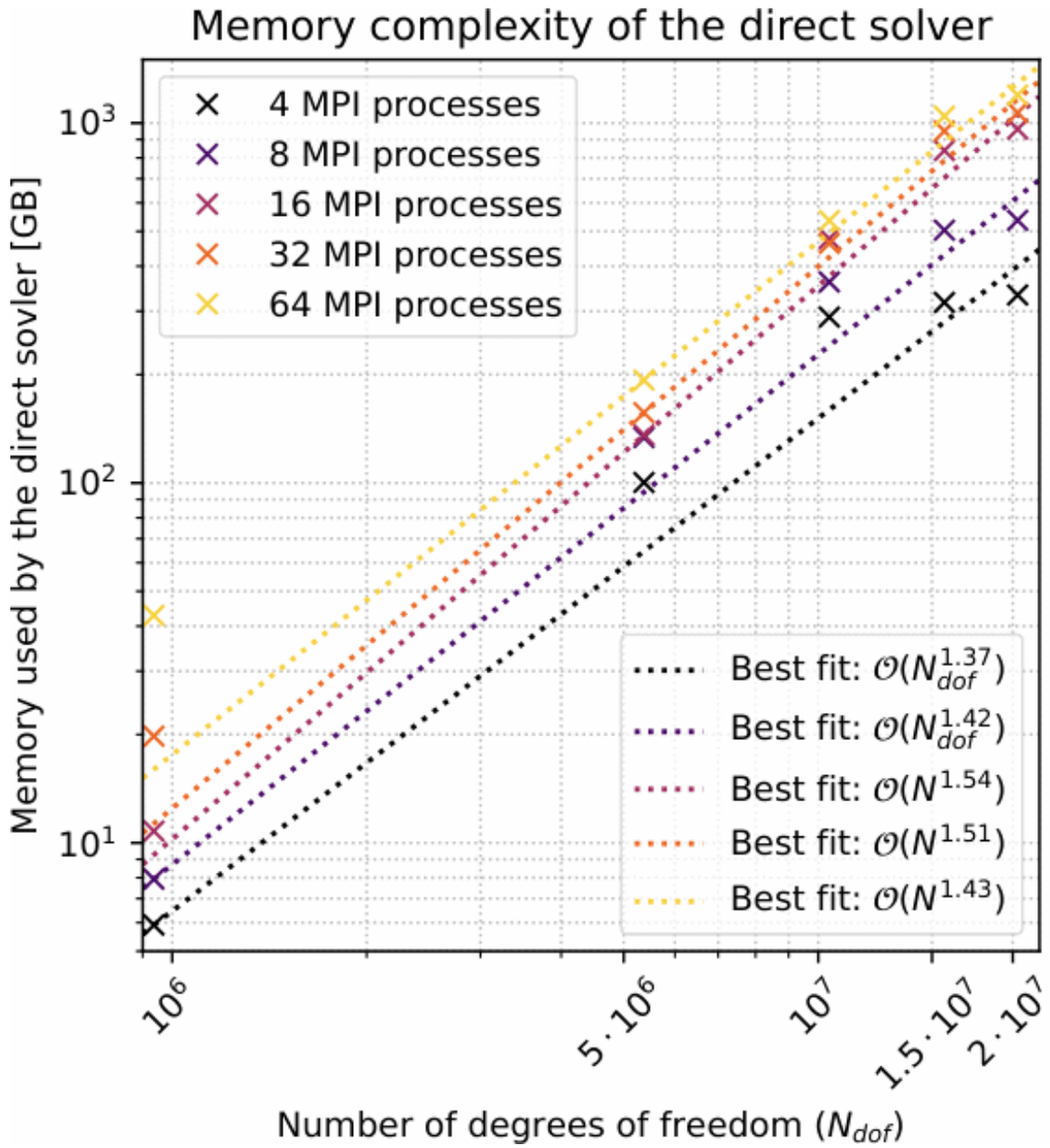
Zerlegung in seriellen und parallelen Anteil $t = t_s + t_p/N(+t_c(N))$

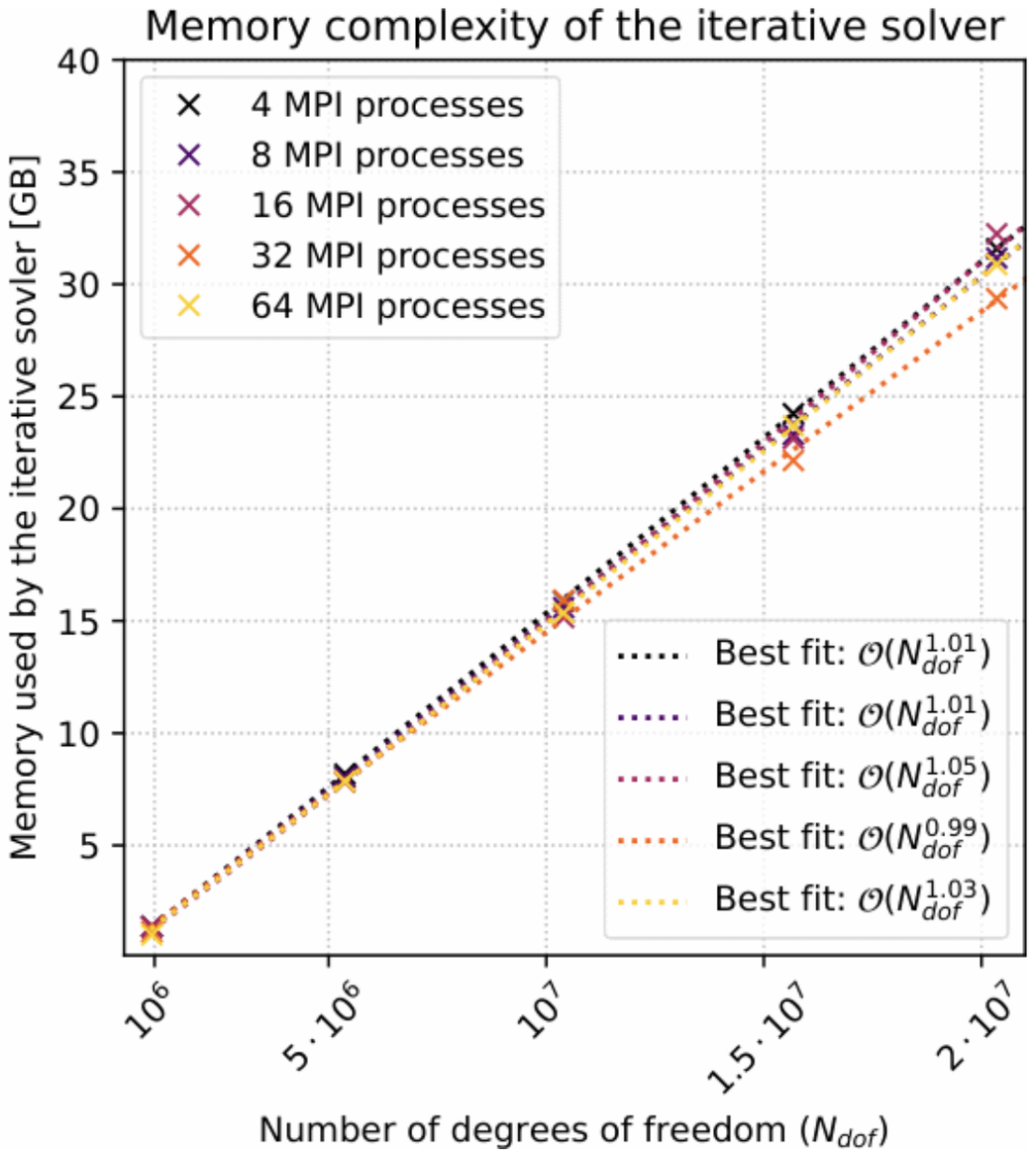


Computational size

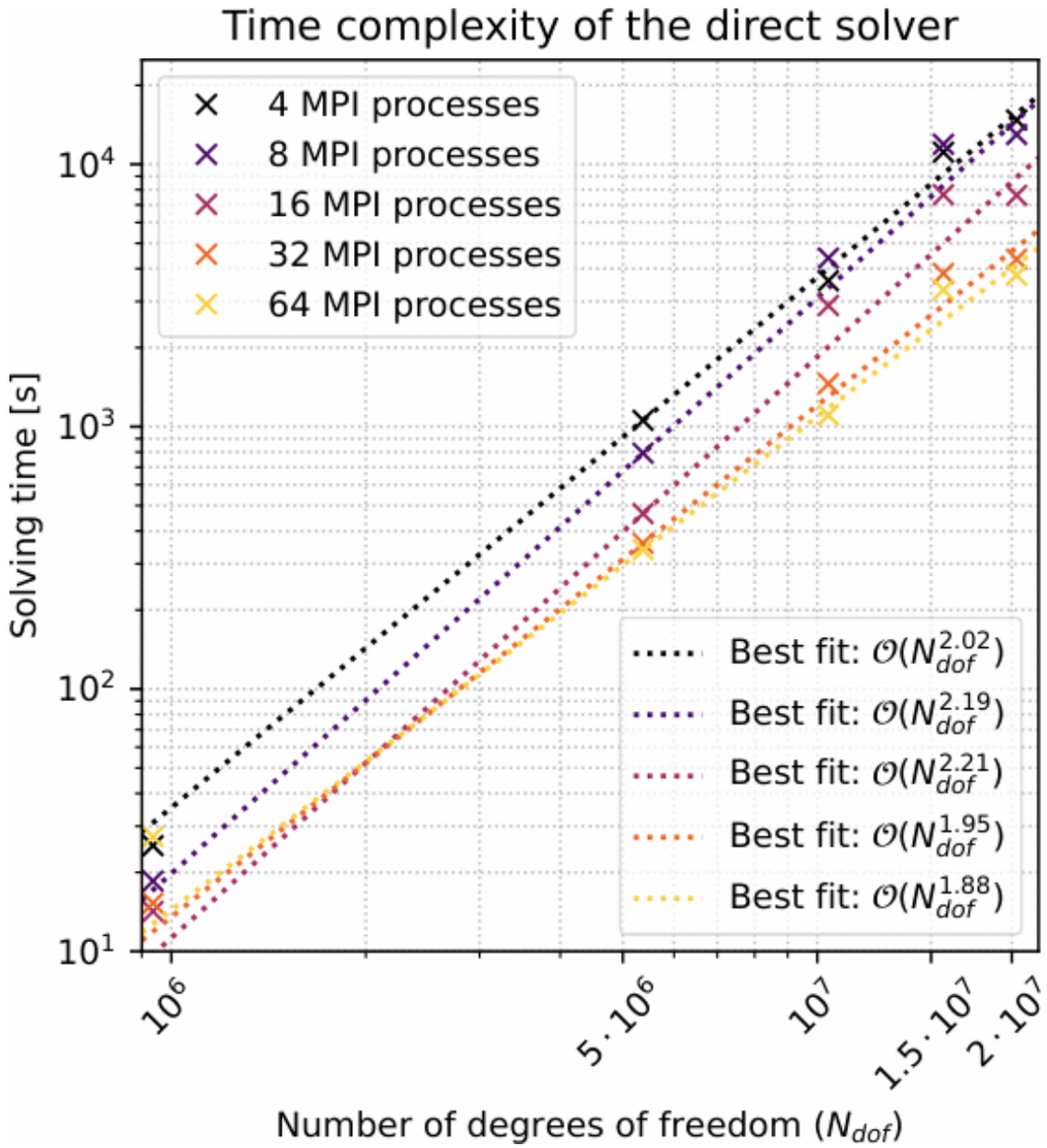


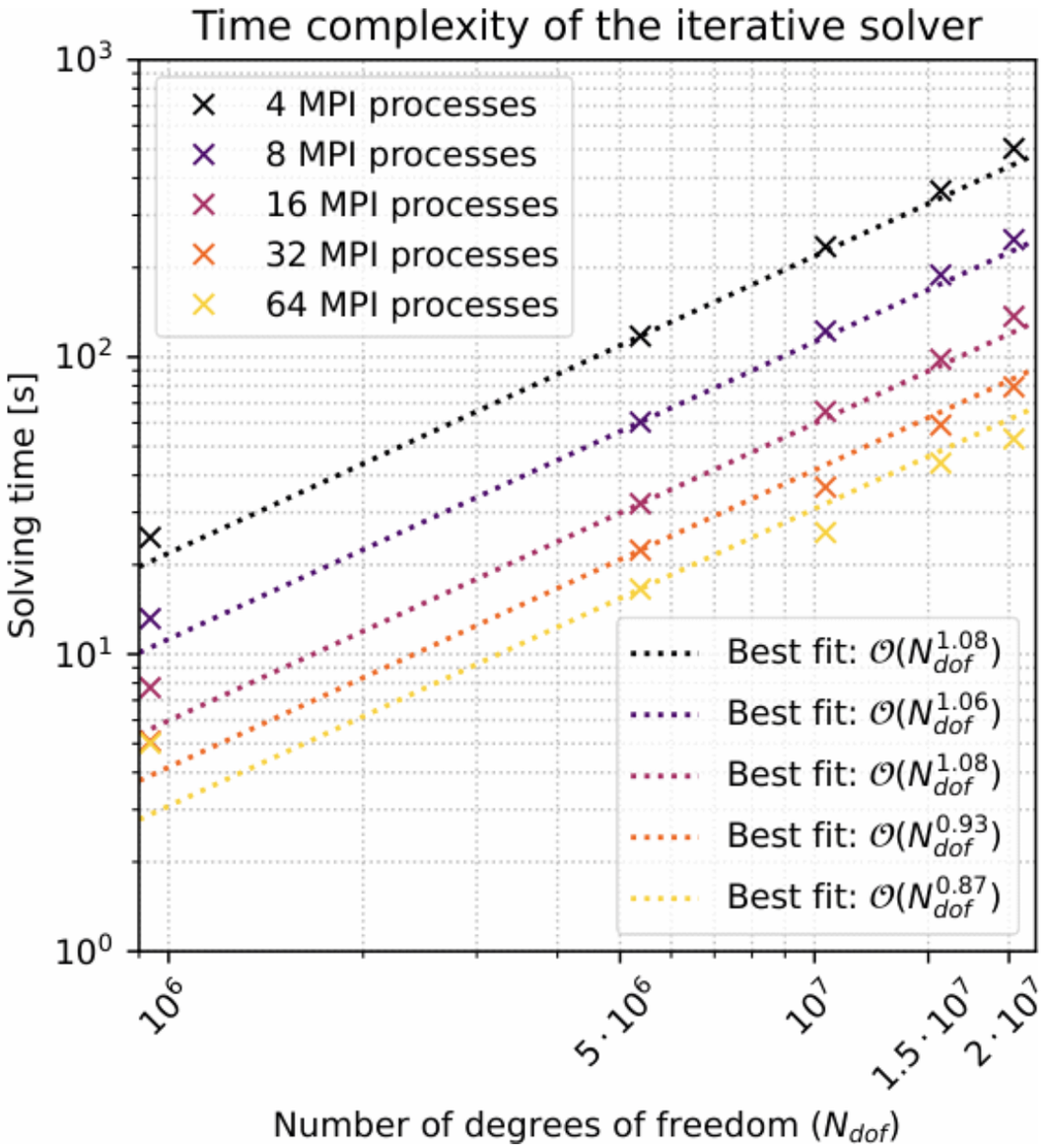
Perfomance analysis - Memory



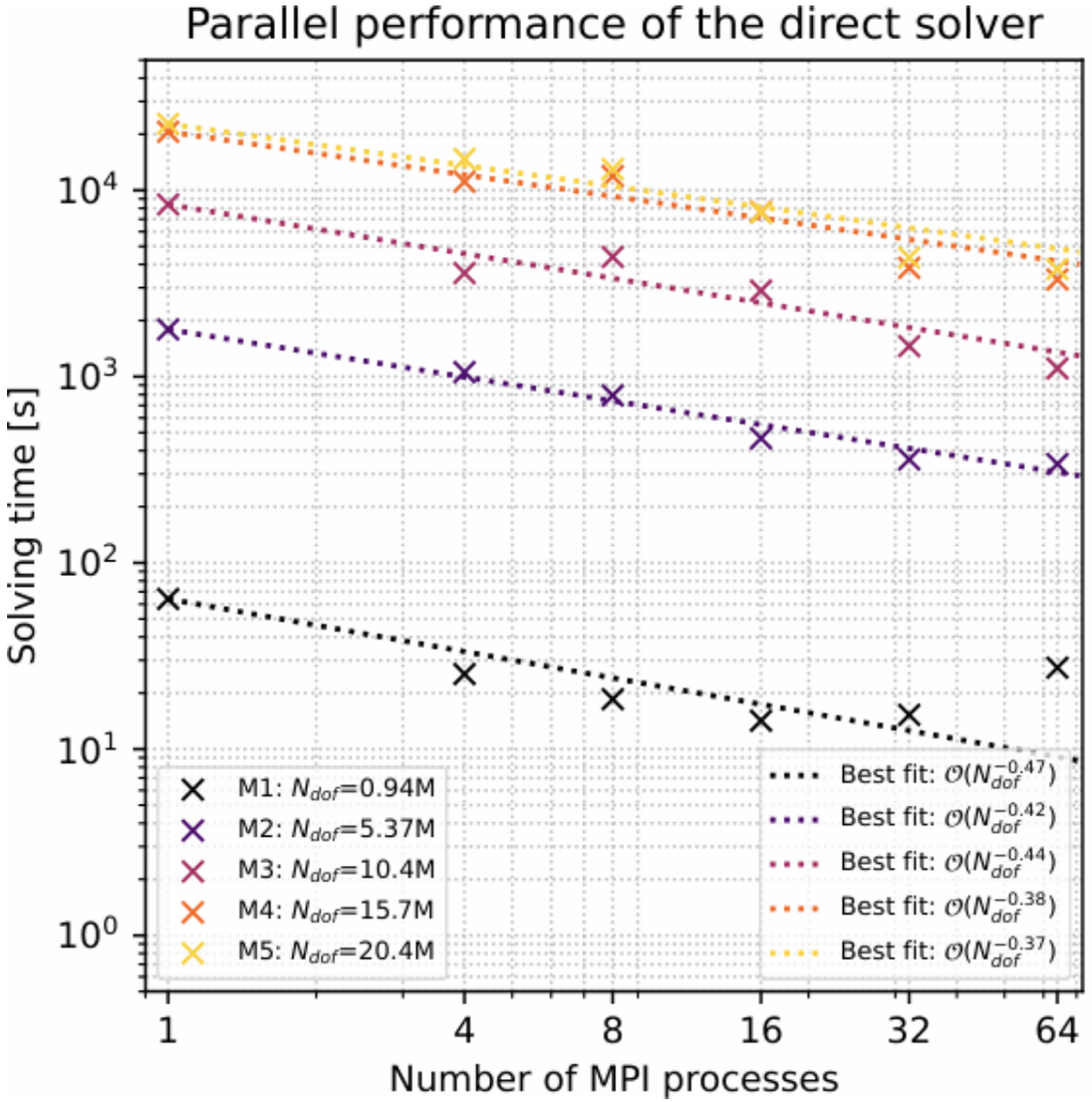


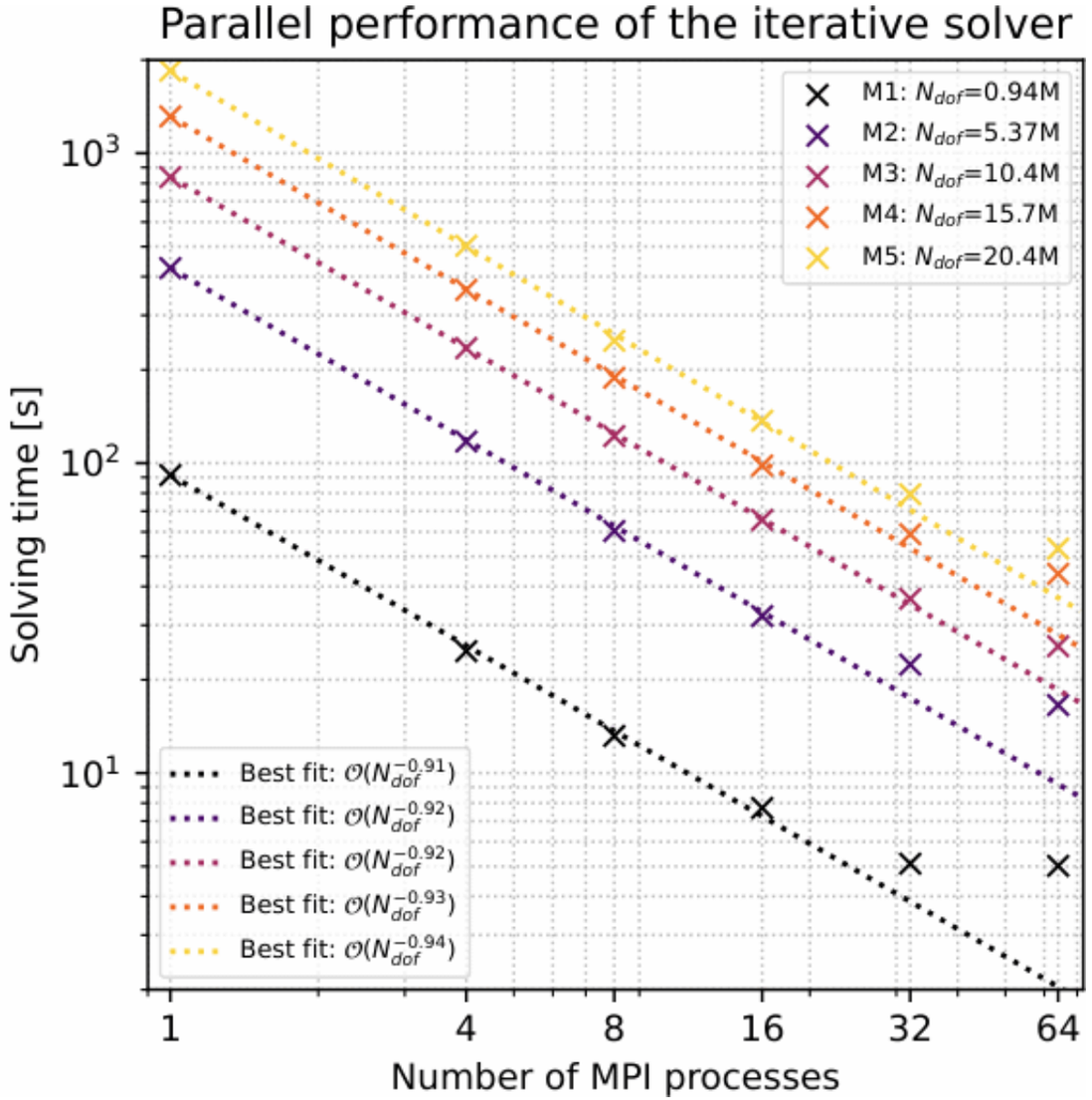
Perfomance analysis - Time



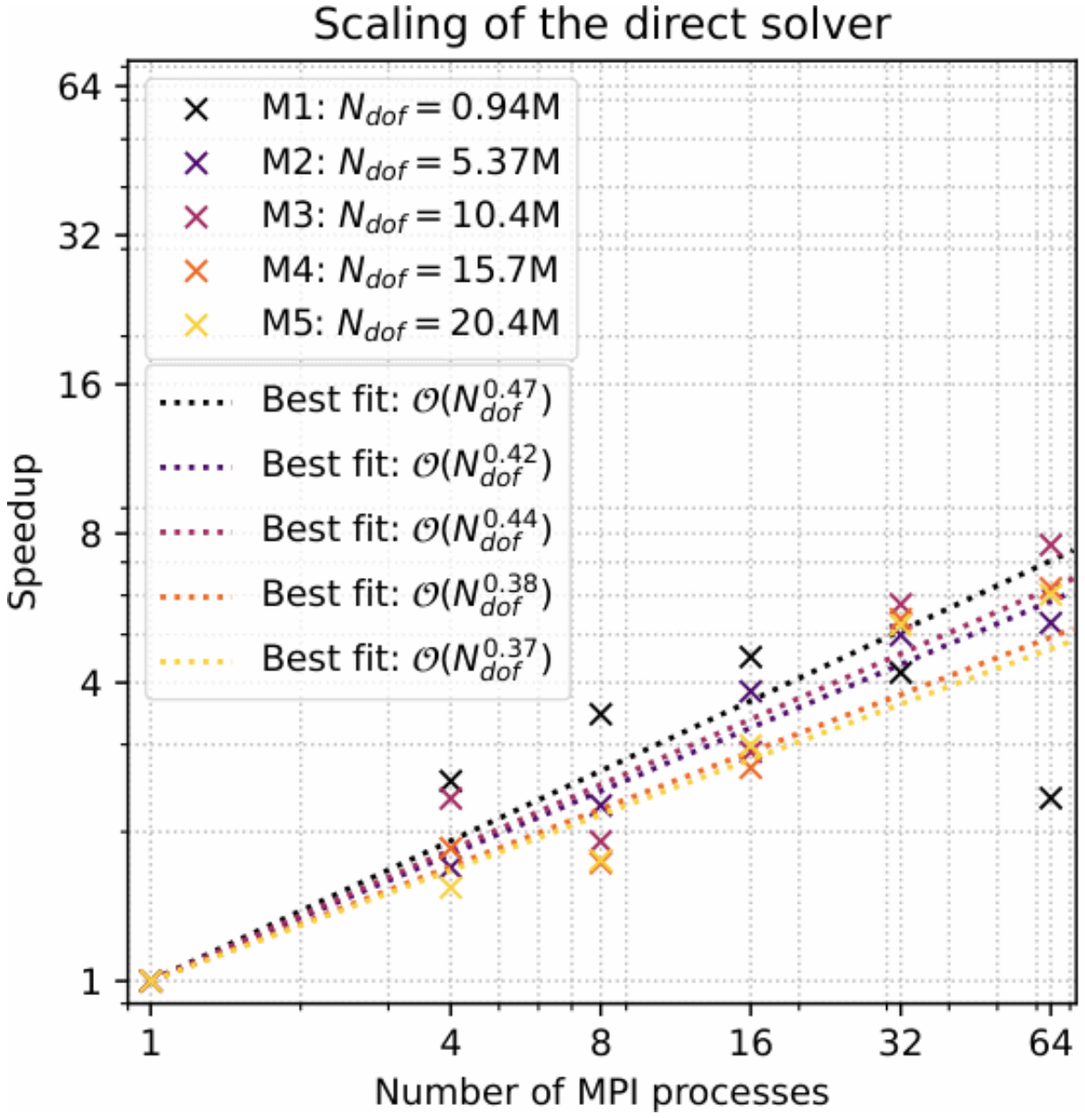


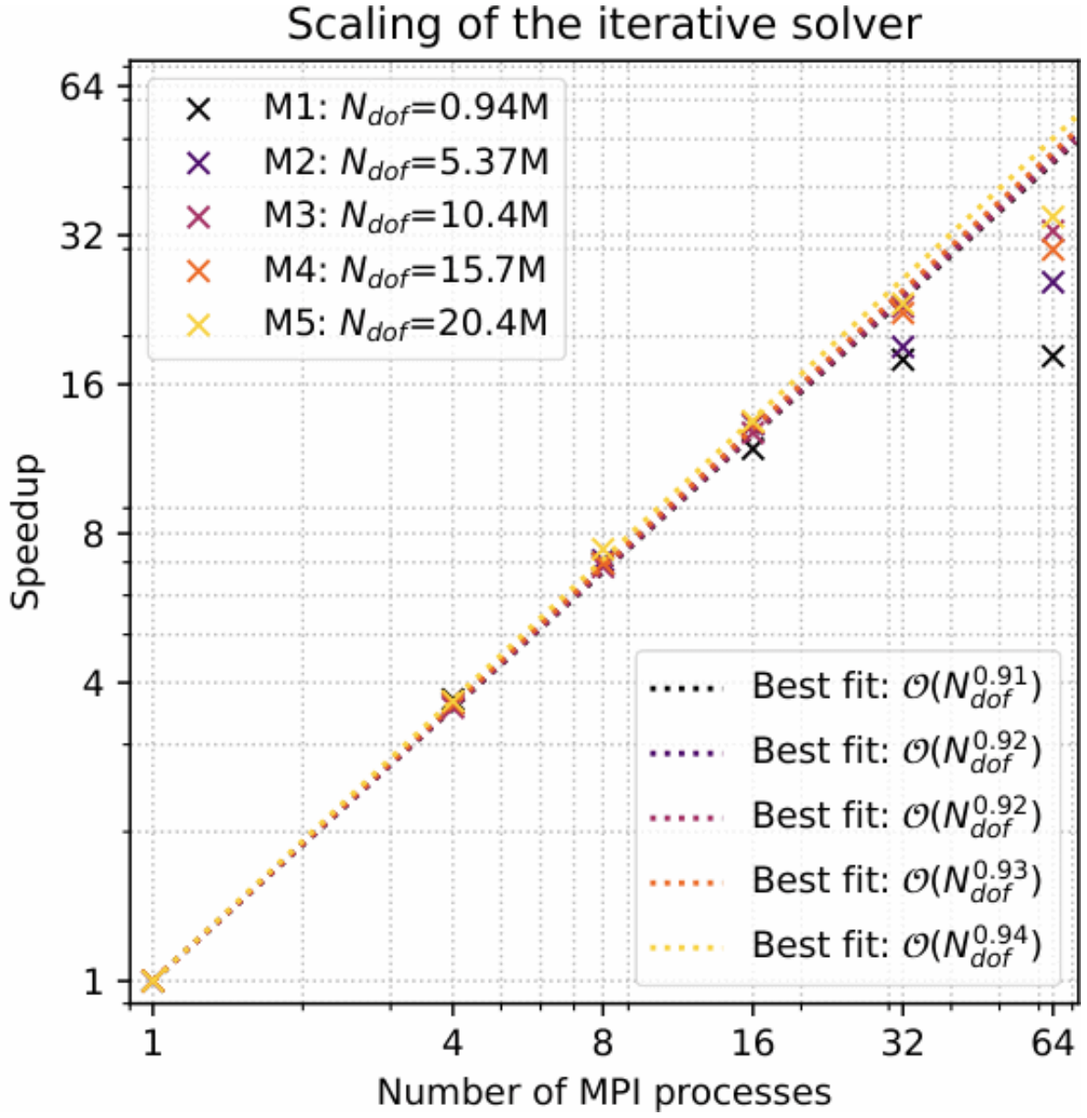
Perfomance analysis - MPI scaling





Perfomance analysis - MPI scaling





Multiple right-hand sides