Introduction to multigrid

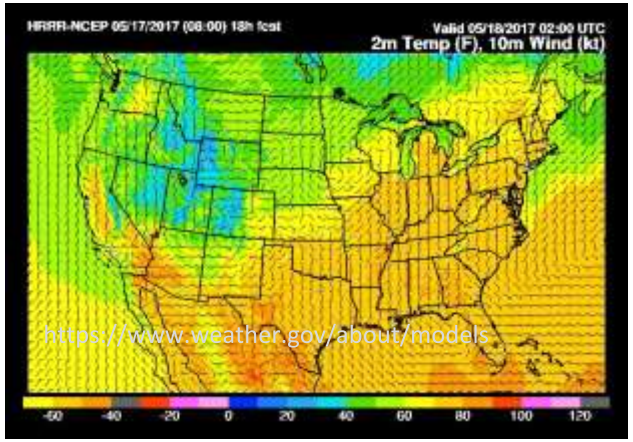
Iskander Ibragimov JGU Mainz

Mainz 4 December 2024

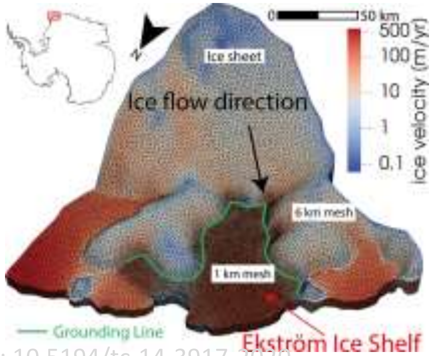
Overview

- Direct solvers
- Iterative solvers
  - Excercises
  - Advantages of iterative solver vs direct
- Steady-state diffusion equation
- Multigrid solver
  - Excercises

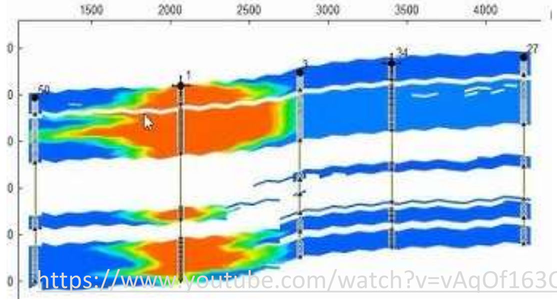
What is common?



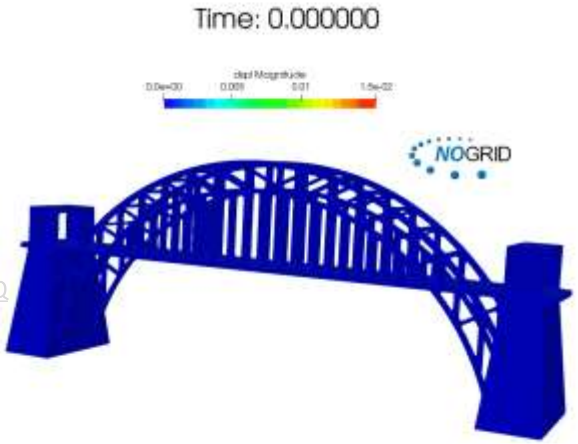
https://www.weather.gov/about/models

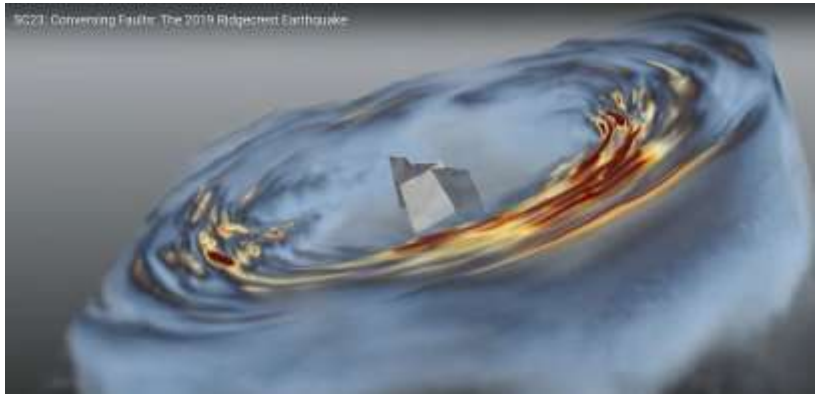


DOI: 10.5194/tc-14-3917-2020

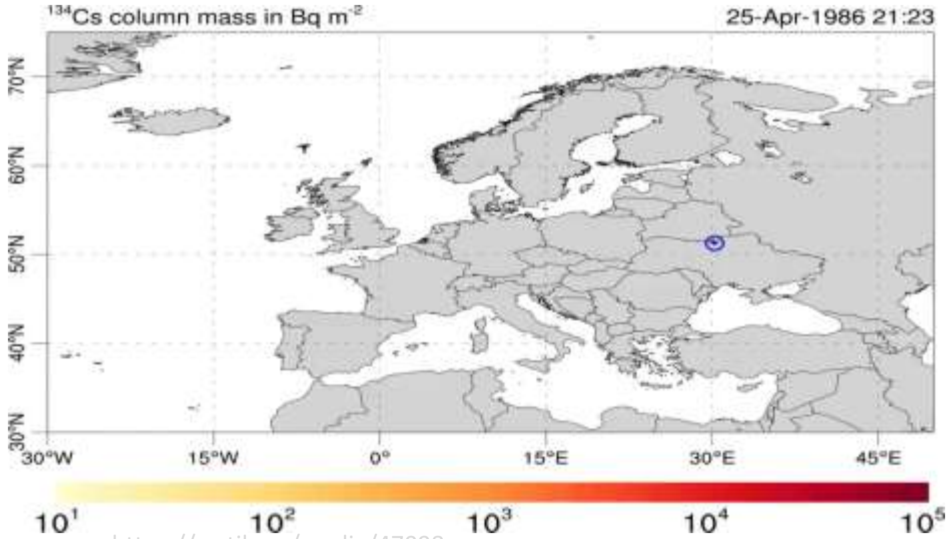


https://www.youtube.com/watch?v=vAqOf163QCQ
tNavigator

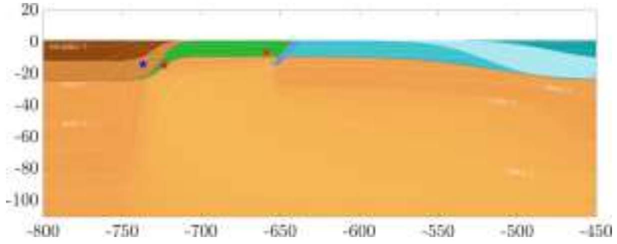




https://www.youtube.com/watch?v=JvMuQN4FuvU
https://seissol.org/



https://av.tib.eu/media/47098



System of linear equations

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

How to solve?

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

Thank you for your attention!

System of linear equations

101x + 12y – 13z = 14

21x + 201y + 23z = 24

–31x + 32y + 301z = 34

$$A = \begin{bmatrix} 101 & 12 & -13 \\ 21 & 201 & 23 \\ -31 & 32 & 301 \end{bmatrix} \quad b = \begin{bmatrix} 14 \\ 24 \\ 34 \end{bmatrix}$$

System of linear equations

$$\mathbf{101}x_1 + \mathbf{12}x_2 - \mathbf{13}x_3 = \mathbf{14}$$
$$\mathbf{21}x_1 + \mathbf{201}x_2 + \mathbf{23}x_3 = \mathbf{24}$$
$$-\mathbf{31}x_1 + \mathbf{32}x_2 + \mathbf{301}x_3 = \mathbf{34}$$

$$A = \begin{bmatrix} 101 & 12 & -13 \\ 21 & 201 & 23 \\ -31 & 32 & 301 \end{bmatrix} \quad b = \begin{bmatrix} 14 \\ 24 \\ 34 \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \boldsymbol{x}_3 \end{bmatrix}$$

Direct solve

Solve system of linear equations in code

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

OR

$$\mathbf{x} = A^{-1}\mathbf{b}$$

Direct solve

Solve system of linear equations in code

```
x = A\b
x = A^(-1)*b
x = inv(A)*b
x = linsolve(A,b)
```

Iterative solvers

Iterative solvers

$$\mathbf{101}x_1 + \mathbf{12}x_2 - \mathbf{13}x_3 = \mathbf{14}$$
$$\mathbf{21}x_1 + \mathbf{201}x_2 + \mathbf{23}x_3 = \mathbf{24}$$
$$-\mathbf{31}x_1 + \mathbf{32}x_2 + \mathbf{301}x_3 = \mathbf{34}$$

Guess x !

$$A = \begin{bmatrix} 101 & 12 & -13 \\ 21 & 201 & 23 \\ -31 & 32 & 301 \end{bmatrix} \quad b = \begin{bmatrix} 14 \\ 24 \\ 34 \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\boldsymbol{x} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

Iterative solvers

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}_{\mathbf{guess}}$$

Iterative solvers

$$\mathbf{r} = \mathbf{b} - \mathbf{Ax}$$
$$\boldsymbol{x} = \boldsymbol{x} + \boldsymbol{r}$$

Iterative solvers

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

$$\boldsymbol{x} = \boldsymbol{x} + \boldsymbol{r}$$

Check if norm of ($x - x_{old}$) is small

Go excercise

Iterative solvers

Richardson method

$$\mathbf{r} = \mathbf{w}(\mathbf{b} \; - \; \mathbf{A}\mathbf{x})$$

Iterative solvers

Richardson method

$$\mathbf{r} = \mathbf{w}(\mathbf{b} - \mathbf{Ax})$$

w = 2/ (min(eig(A)) + max(eig(A)))

Iterative solvers
Jacobi method

$$\mathbf{r} = \boldsymbol{D}^{-\mathbf{1}}(\mathbf{b} - (\mathbf{L} + \mathbf{U})\mathbf{x})$$

$$A = \begin{bmatrix} 101 & 12 & -13 \\ 21 & 201 & 23 \\ -31 & 32 & 301 \end{bmatrix}$$

$$D = \begin{bmatrix} 101 & & \\ & 201 & \\ & & 301 \end{bmatrix}$$

$$U = \begin{bmatrix} & 12 & -13 \\ & & 23 \\ & & \end{bmatrix}$$

$$L = \begin{bmatrix} & & \\ 21 & & \\ -31 & 32 & \end{bmatrix}$$

Iterative solvers

Dampened Jacobi method

$$\mathbf{r} = \mathbf{w}\mathbf{D}^{-1}(\mathbf{b} - (\mathbf{L} + \mathbf{U})\mathbf{x}) + (\mathbf{1} - \mathbf{w})\mathbf{x}$$

$$A = \begin{bmatrix} 101 & 12 & -13 \\ 21 & 201 & 23 \\ -31 & 32 & 301 \end{bmatrix}$$

$$D = \begin{bmatrix} 101 & & \\ & 201 & \\ & & 301 \end{bmatrix}$$

$$U = \begin{bmatrix} & 12 & -13 \\ & & 23 \\ & & \end{bmatrix}$$

$$L = \begin{bmatrix} & & \\ 21 & & \\ -31 & 32 & \end{bmatrix}$$

Iterative solvers

Gauss-Seidel method

$$A = \begin{bmatrix} 101 & 12 & -13 \\ 21 & 201 & 23 \\ -31 & 32 & 301 \end{bmatrix}$$

$$D = \begin{bmatrix} 101 & & \\ & 201 & \\ & & 301 \end{bmatrix}$$

$$U = \begin{bmatrix} & 12 & -13 \\ & & 23 \\ & & \end{bmatrix}$$

$$L = \begin{bmatrix} & & \\ 21 & & \\ -31 & 32 & \end{bmatrix}$$

$$\mathbf{r} = \mathbf{L}^{-1}(\mathbf{b} - \mathbf{U}\mathbf{x})$$

Iterative solvers

Successive over-relaxation method

$$\mathbf{r} = (\mathbf{1} - \mathbf{w})\mathbf{x} + \mathbf{w}\mathbf{D}^{-1}(\mathbf{b} - \mathbf{U}\mathbf{x} - \mathbf{L}\mathbf{x})$$

$$A = \begin{bmatrix} 101 & 12 & -13 \\ 21 & 201 & 23 \\ -31 & 32 & 301 \end{bmatrix}$$

$$D = \begin{bmatrix} 101 & & \\ & 201 & \\ & & 301 \end{bmatrix}$$

$$U = \begin{bmatrix} & 12 & -13 \\ & & 23 \\ & & \end{bmatrix}$$

$$L = \begin{bmatrix} & & \\ 21 & & \\ -31 & 32 & \end{bmatrix}$$

w = 0.52

Iterative solvers

Dampened Jacobi method simplified

$$A = \begin{bmatrix} 101 & 12 & -13 \\ 21 & 201 & 23 \\ -31 & 32 & 301 \end{bmatrix}$$

$$D = \begin{bmatrix} 101 & & \\ & 201 & \\ & & 301 \end{bmatrix}$$

$$U = \begin{bmatrix} & 12 & -13 \\ & & 23 \\ & & \end{bmatrix}$$

$$L = \begin{bmatrix} & & \\ 21 & & \\ -31 & 32 & \end{bmatrix}$$

$$\mathbf{r} = \mathbf{w}\mathbf{D}^{-1}(\mathbf{b} - (\mathbf{L} + \mathbf{U})\mathbf{x}) + (\mathbf{1} - \mathbf{w})\mathbf{x}$$

Dampened Jacobi method

$$\mathbf{r} = \mathbf{w}\mathbf{D}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x})$$

Simplified dampened Jacobi method

w = 2/3

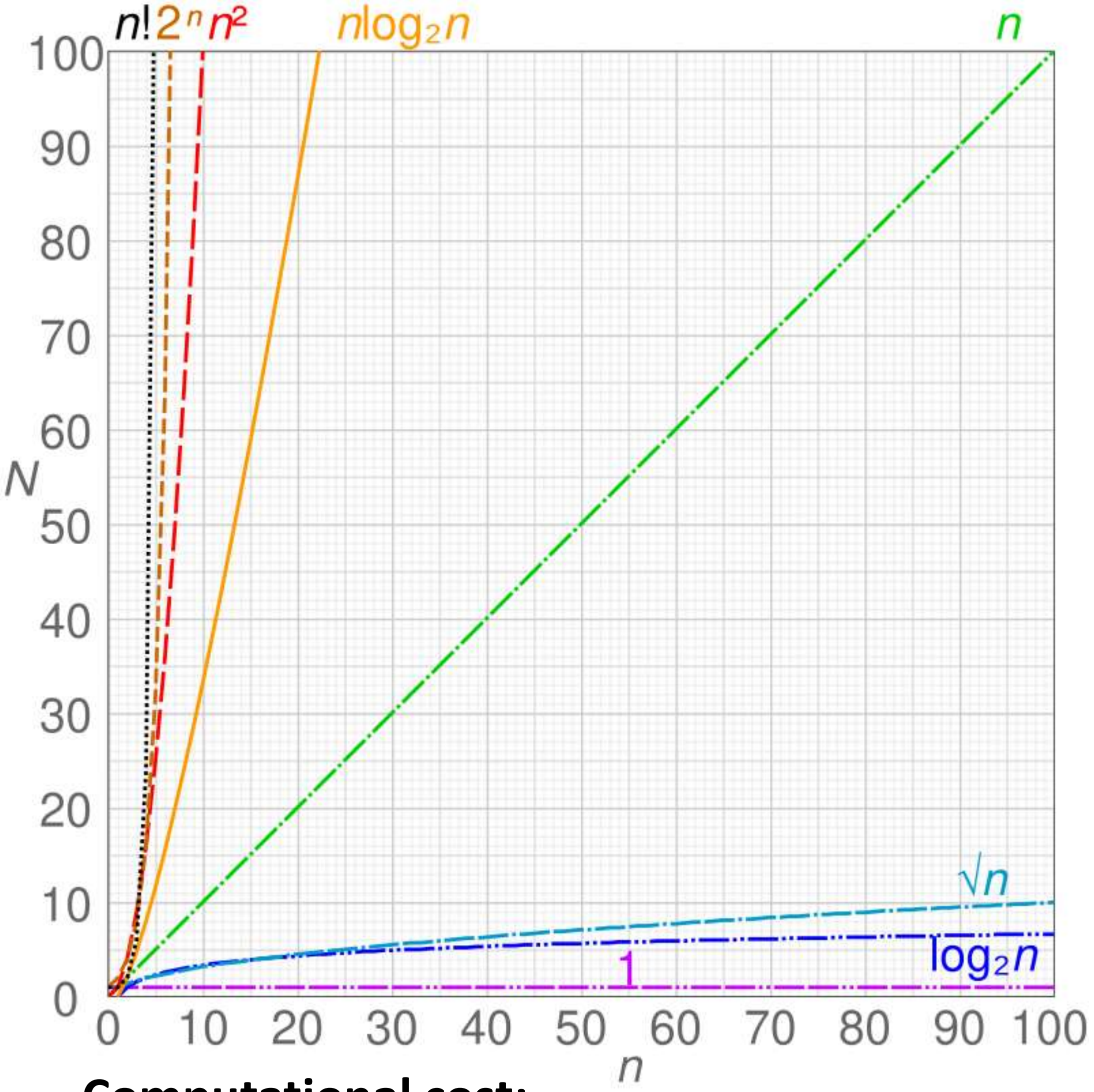
Disclaimer for iterative methods:

A Needs to be positive definite to converge = **diagonally dominant**

$$A = \begin{bmatrix} 101 & 12 & -13 \\ 21 & 201 & 23 \\ -31 & 32 & 301 \end{bmatrix}$$

Why use iterative solvers?

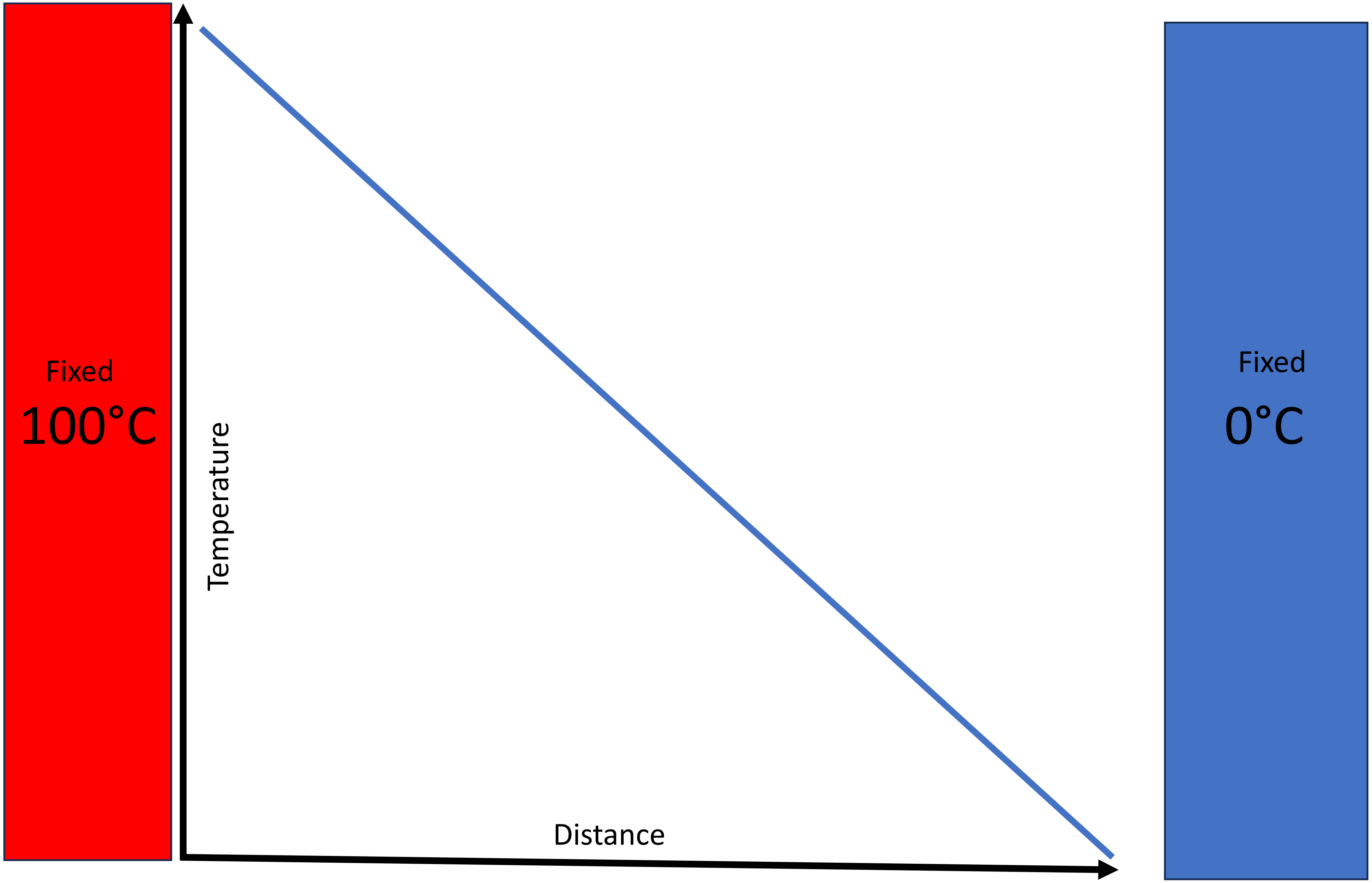
- Only matrix-vector multiplication
- Most of our systems has zeros in matrix A
- Do not need to store additional matrices for factorization
- More scalable
- Much better to parallelize
- Can use preconditioners



Computational cost:

Direct solver: $O(n^3)$

Iterative solvers (sparse): $\approx O(n^2)$



Fixed
100°C
Temperature
Distance
Fixed
0°C

Temperature equation

(When internal heat production is negligible and there is no advection of material)

$$\kappa \nabla^2 T = \frac{\partial T}{\partial t}$$

$$\kappa \ \frac{\partial^2}{\partial x^2} T = \frac{\partial T}{\partial t}$$

where $\kappa = \frac{k}{\rho C_p}$

T – temperature

t – time

κ – diffusivity constant

Steady-state

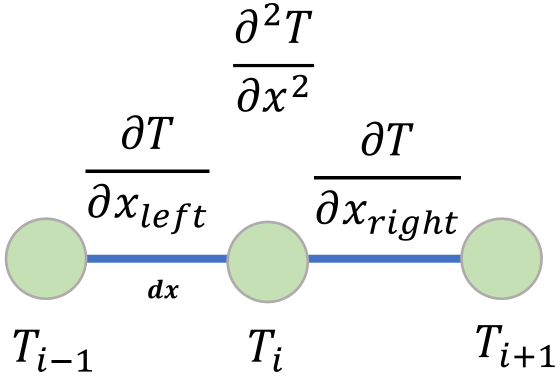
$$\frac{\partial T}{\partial t} = 0$$

$$\kappa \frac{\partial^2 T}{\partial x^2} = 0$$

- Temperature does not change in time

Finite-Difference Discretization

$$\boldsymbol{\kappa \frac{\partial^2 T}{\partial x^2} = 0}$$

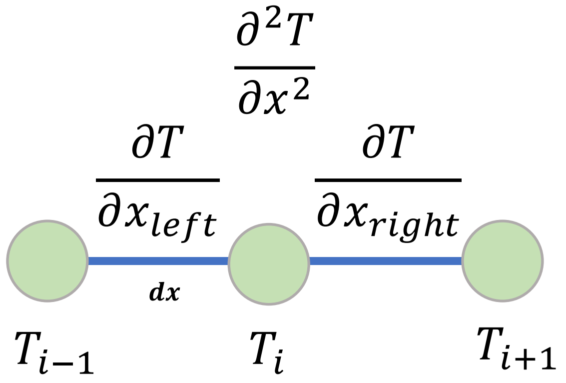


$$\frac{\partial T}{\partial x_{right}} = \frac{T_{i+1} - T_i}{dx}$$

$$\frac{\partial T}{\partial x_{left}} = \frac{T_i - T_{i-1}}{dx}$$

$$\frac{\partial^2 T_i}{\partial x^2} = \frac{\partial}{\partial x}\left(\frac{\partial T}{\partial x}\right) = \frac{\frac{\partial T}{\partial x_R} - \frac{\partial T}{\partial x_L}}{dx} = \frac{1}{dx}\left(\frac{T_{i+1} - T_i}{dx} - \frac{T_i - T_{i-1}}{dx}\right)$$

Finite-Difference Discretization



$$\boldsymbol{\kappa \frac{\partial^2 T}{\partial x^2} = 0}$$

Finite-difference approximation

$$\boldsymbol{k \frac{T_{i+1} - 2T_i + T_{i-1}}{dx^2} = 0}$$

$$\boldsymbol{\frac{k}{dx^2} T_{i+1} - 2\frac{k}{dx^2} T_i + \frac{k}{dx^2} T_{i-1} = 0}$$

$$S = \frac{k}{dx^2} \text{ — Constant}$$

$$\boldsymbol{ST_{i+1} - 2ST_i + ST_{i-1} = 0}$$

$$\frac{\partial T}{\partial x_{right}} = \frac{T_{i+1} - T_i}{dx}$$

$$\frac{\partial T}{\partial x_{left}} = \frac{T_i - T_{i-1}}{dx}$$

$$\frac{\partial^2 T_i}{\partial x^2} = \frac{1}{dx}\left(\frac{T_{i+1} - T_i}{dx} - \frac{T_i - T_{i-1}}{dx}\right)$$

Fill up matrix A and RHS

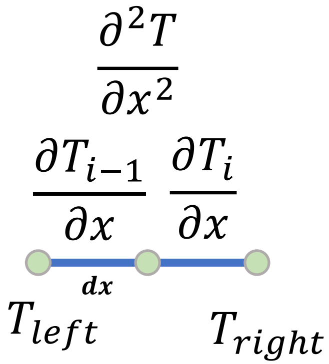
$$\boldsymbol{ST_{i+1} - 2ST_i + ST_{i-1} = 0}$$

$$A = \begin{bmatrix} -2S & 0 & 0 \\ 1 & -2S & 1 \\ 0 & 0 & -2S \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad b = \begin{bmatrix} T_{guess} \\ T_{guess} \\ T_{guess} \end{bmatrix}$$

New T

Old T/Initial Guess T

Boundary Conditions

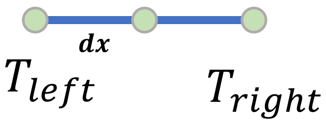


$$A = \begin{bmatrix} -2S & 0 & 0 \\ 1 & -2S & 1 \\ 0 & 0 & -2S \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad b = \begin{bmatrix} T_{guess} \\ T_{guess} \\ T_{guess} \end{bmatrix}$$

Boundary Conditions

$$\frac{\partial^2 T}{\partial x^2}$$

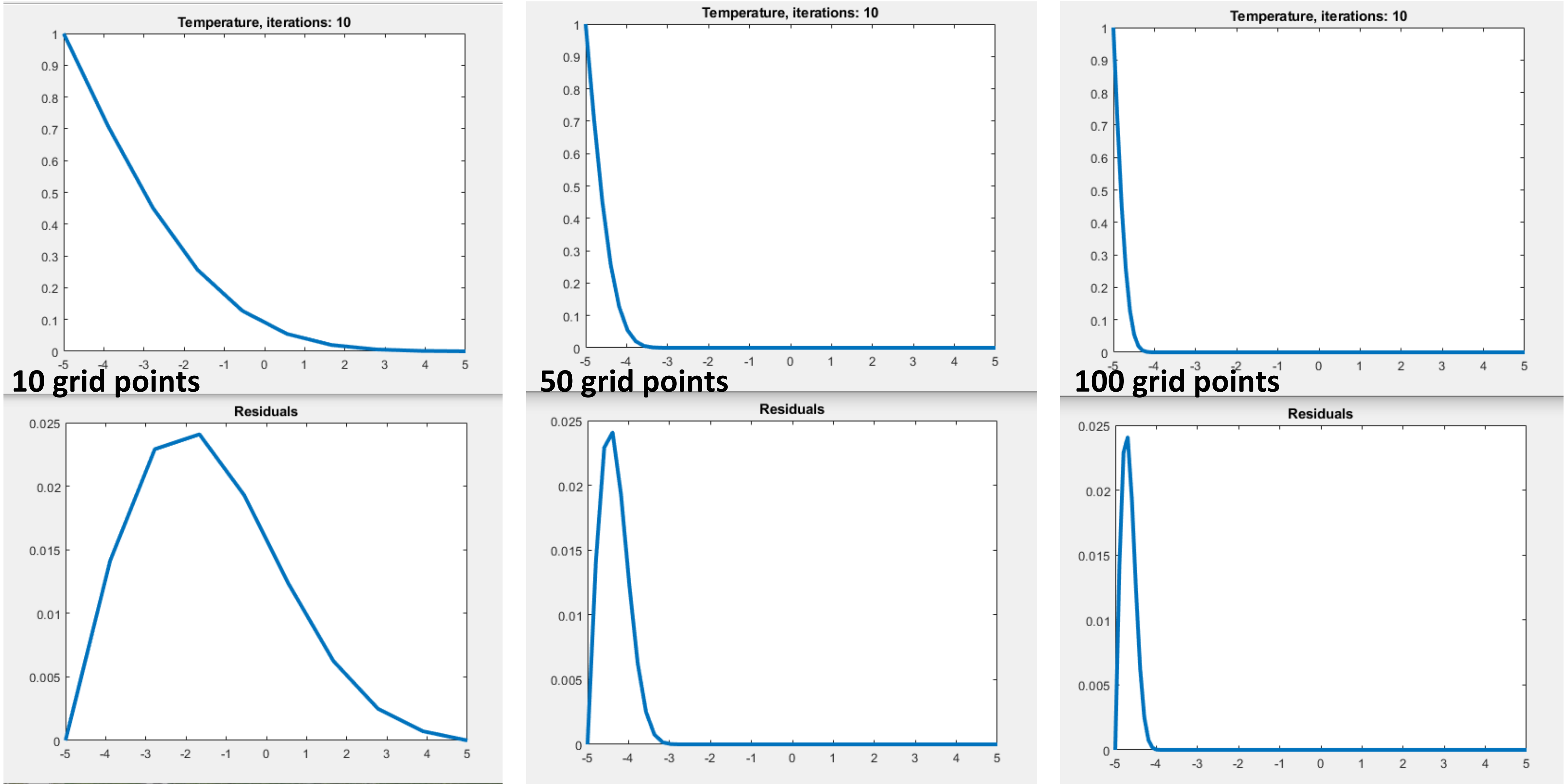
$$\frac{\partial T_{i-1}}{\partial x} \quad \frac{\partial T_i}{\partial x}$$



$$A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & -2S & 1 \\ 0 & 0 & 1 \end{bmatrix} \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad b = \begin{bmatrix} T_{left} \\ T_{guess} \\ T_{right} \end{bmatrix}$$

Correction propagation

Go excercise



Iteration cycle

Courant–Friedrichs–Lewy (CFL) condition is dependent on spatial increment

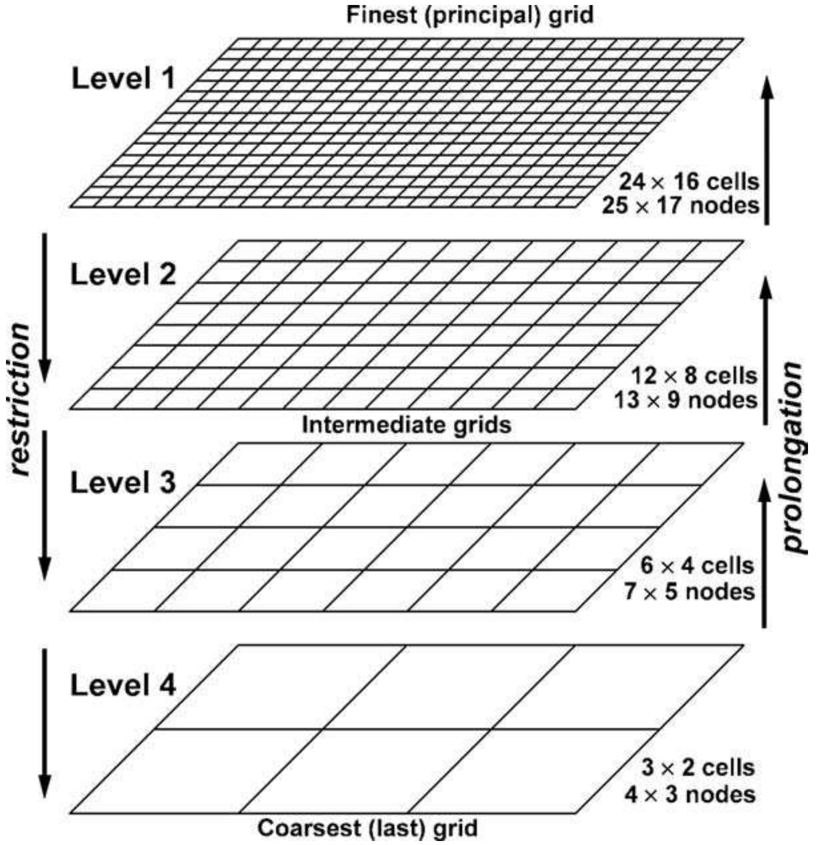
Initial model

Model with small correction

Jacobi, Gauss-seidel etc

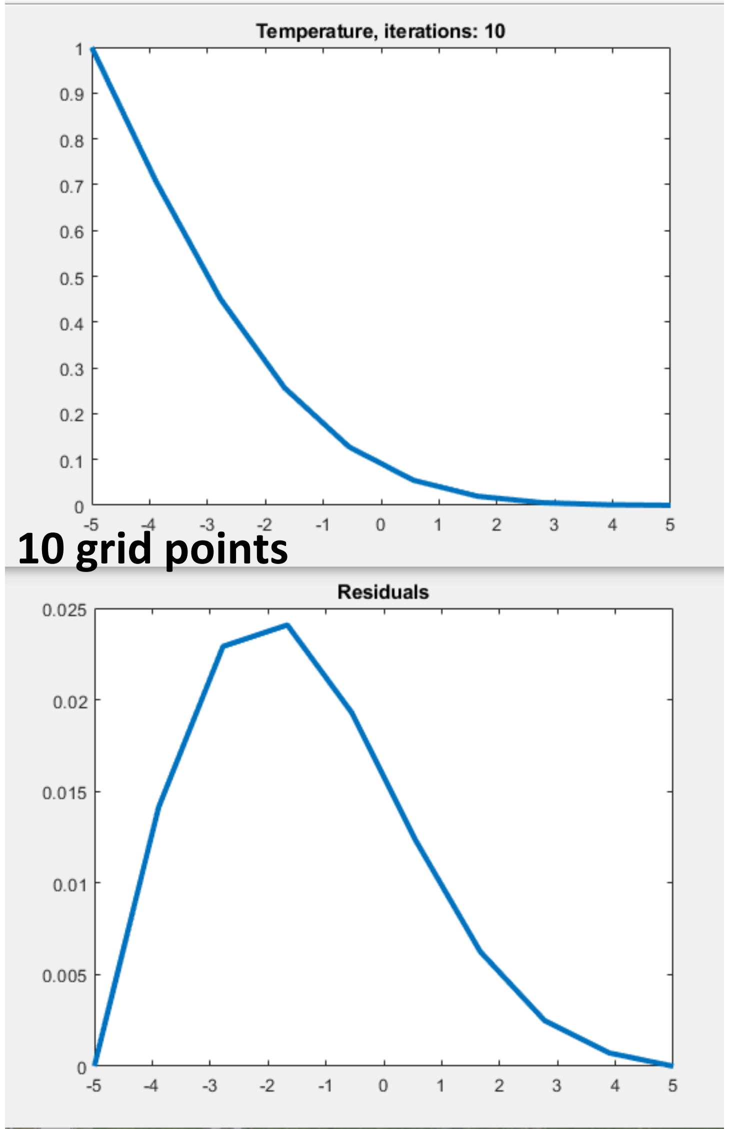
What is multigrid

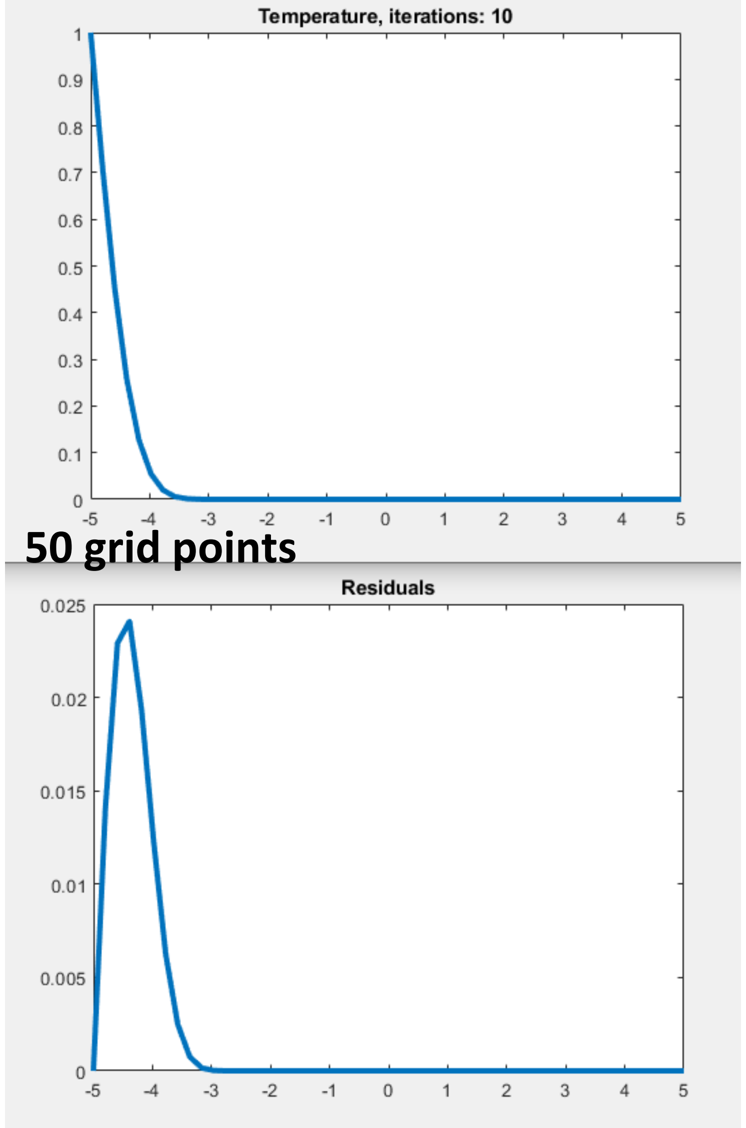
- The essential multigrid principle is to approximate the smooth (long wavelength) part of the error on coarser grids.

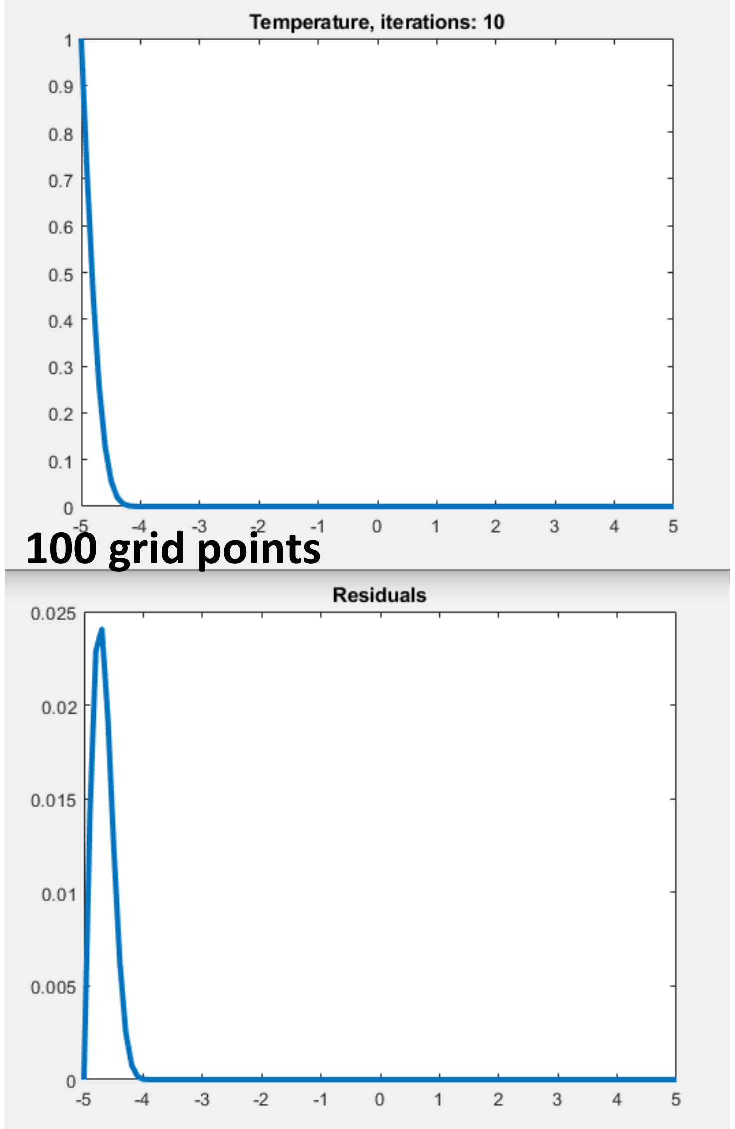


2-D multigrid
T. Gerya, 2008

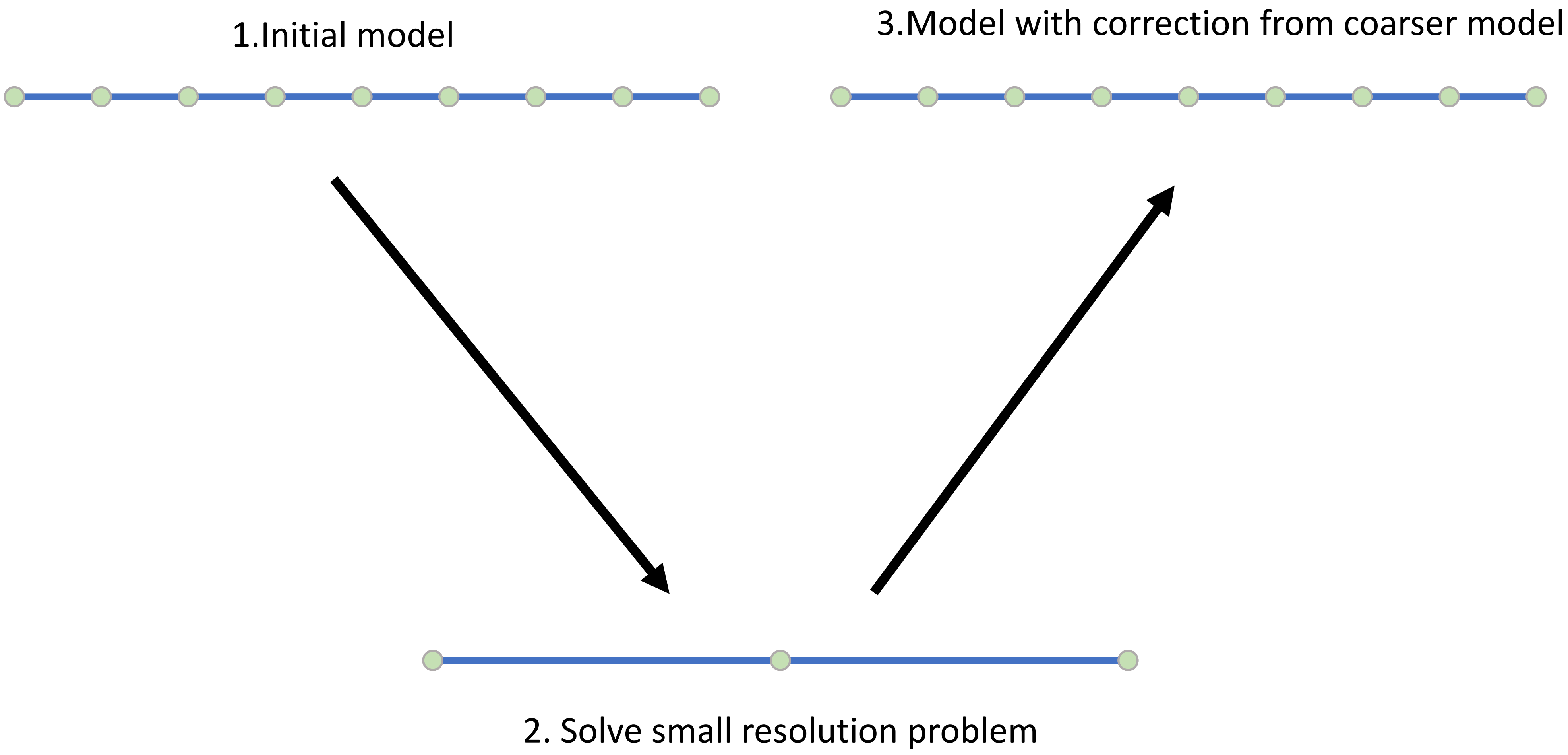
Correction propagation



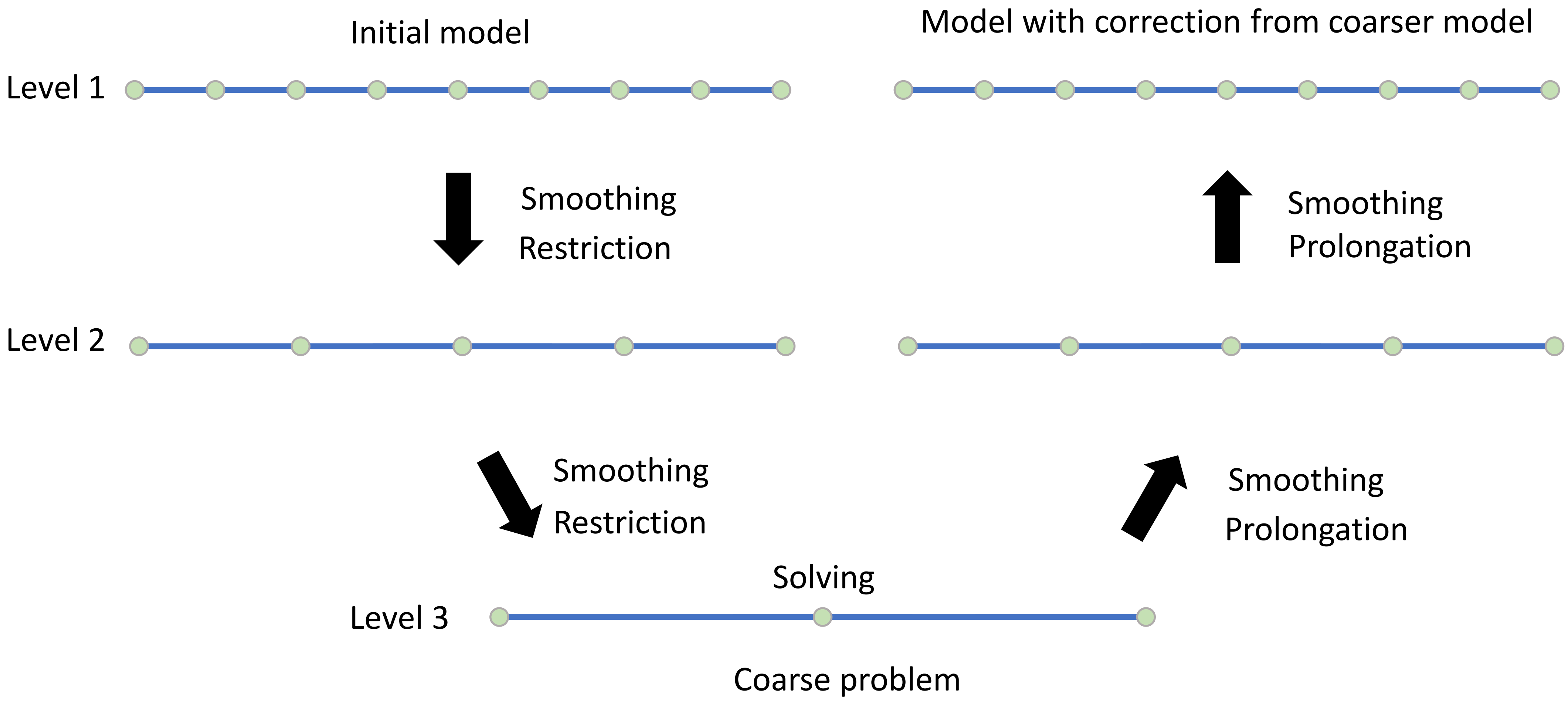




V-cycle

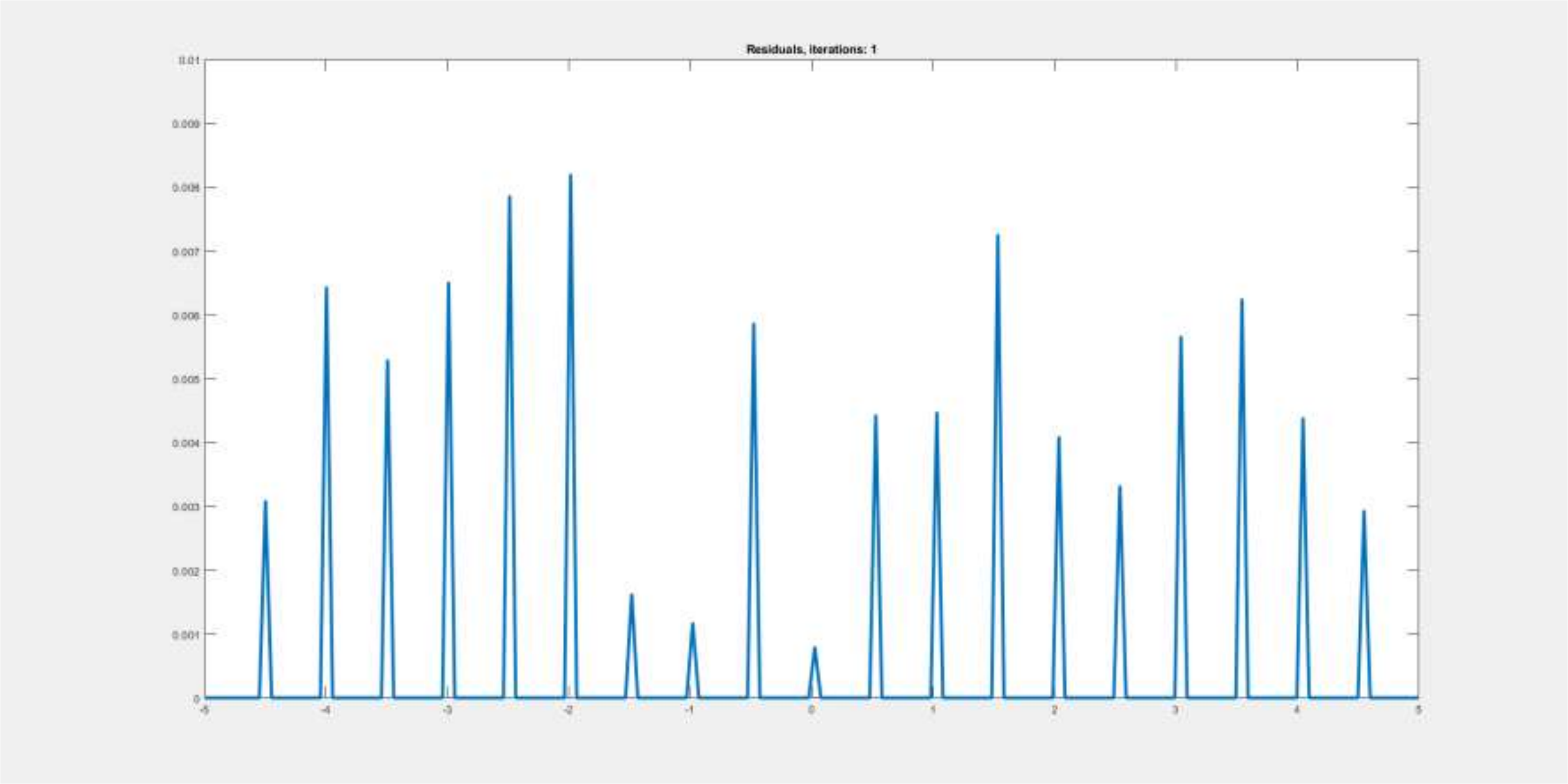

1.Initial model
3.Model with correction from coarser model
2. Solve small resolution problem

V-cycle 4 multigrid levels



Multigrid

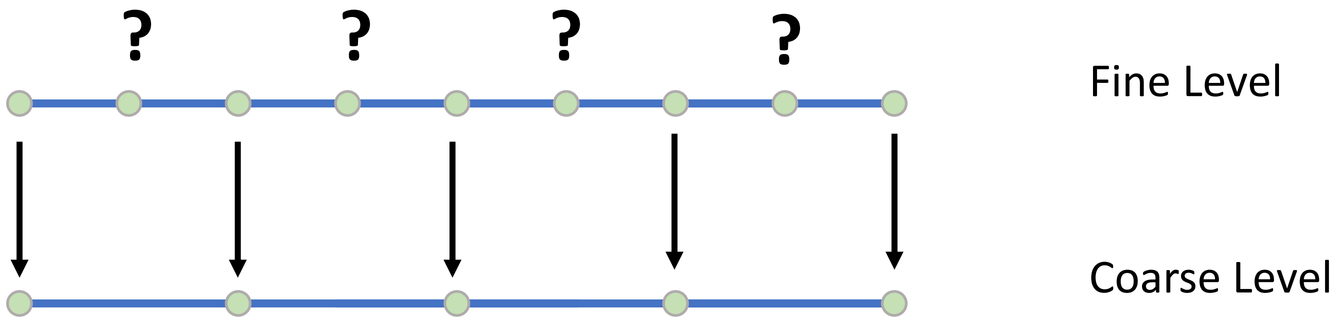
Smoothing



Residuals look spiky!

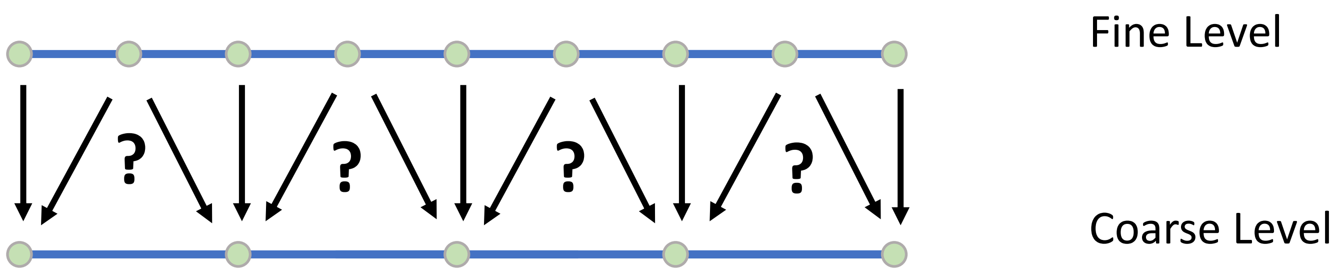
Geometric Multigrid

Restriction operation



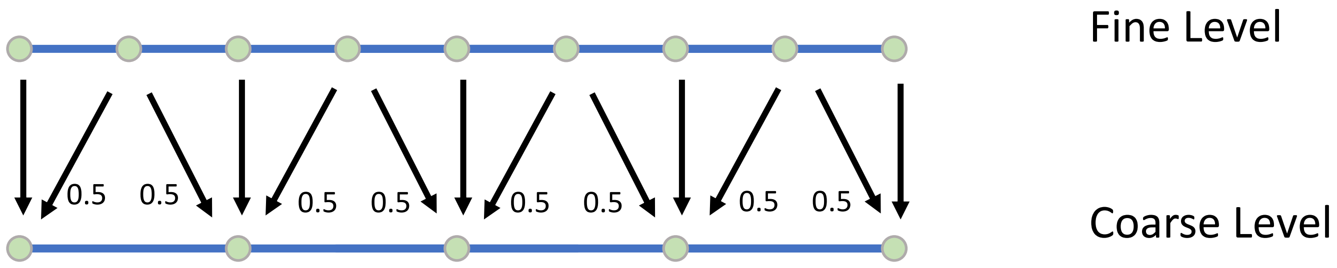
Geometric Multigrid

Restriction operation



Geometric Multigrid

Restriction operation



$$R = \begin{array}{cc} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 1 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 1 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} & \frac{\mathbf{nx}}{\mathbf{2}} \\ \text{nx} & \end{array}$$

rectangular!

Geometric Multigrid

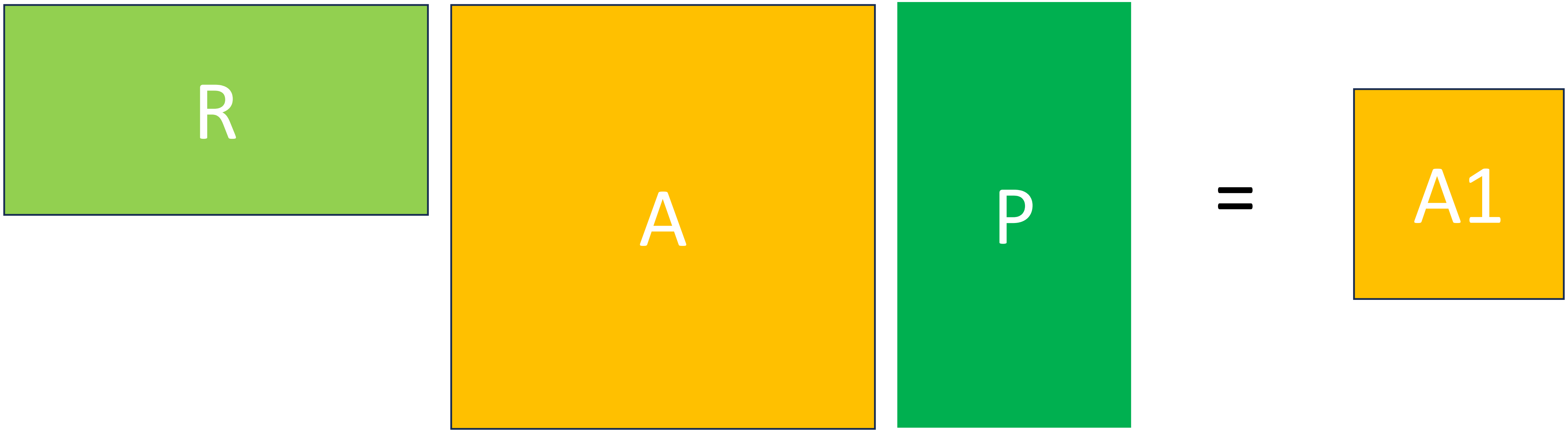
Prolongation operation

$$P = R$$

Geometric Multigrid

Galerkin Coarsening

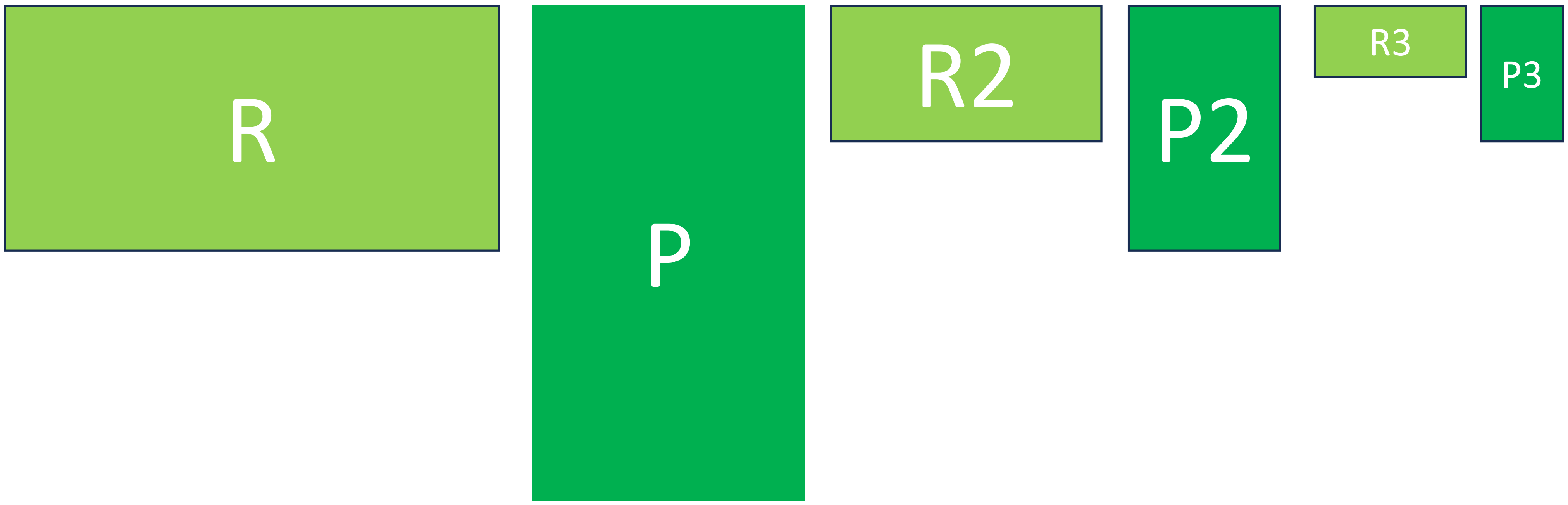
$$r1 = Rr$$
$$A1 = RAP$$



Geometric Multigrid

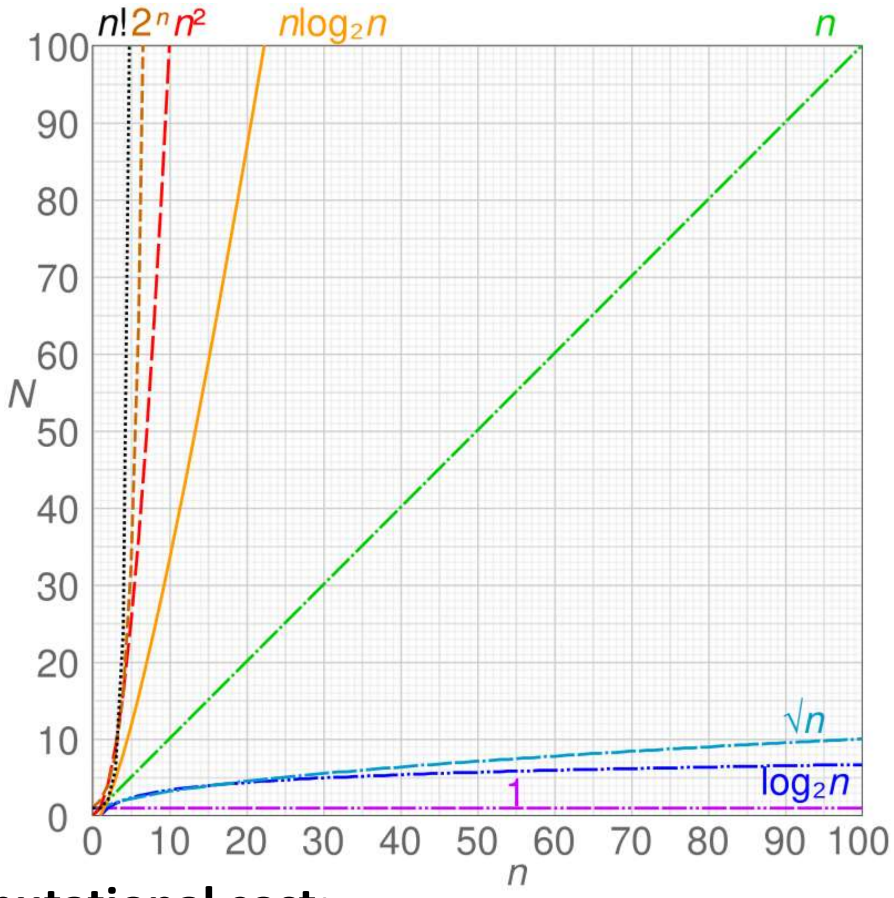
Galerkin Coarsening

You need to build all restriction operators in the beginning for geometric multigrid



What is an advantage of multigrid

- Good to parallelize
- Low memory usage
- Fast to converge
- Scalable
- No problem with sparse matrices
- Preconditioners
- Domain decomposition

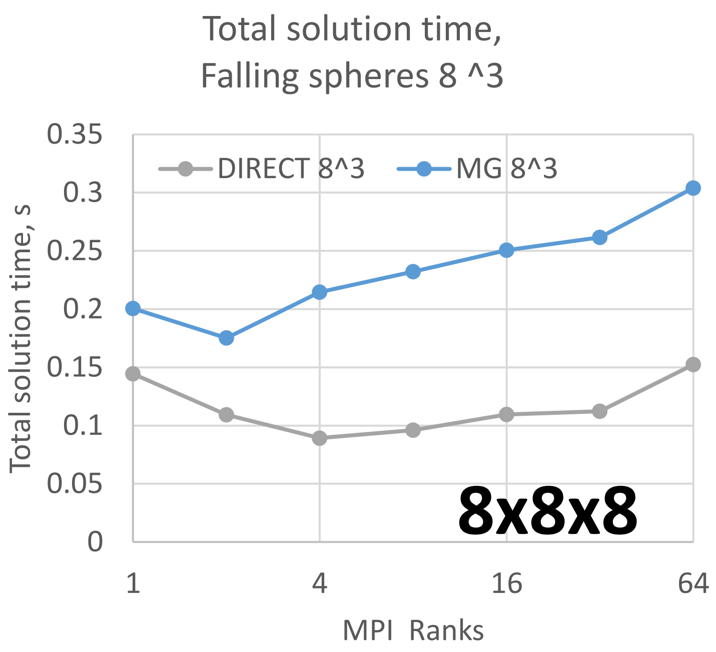


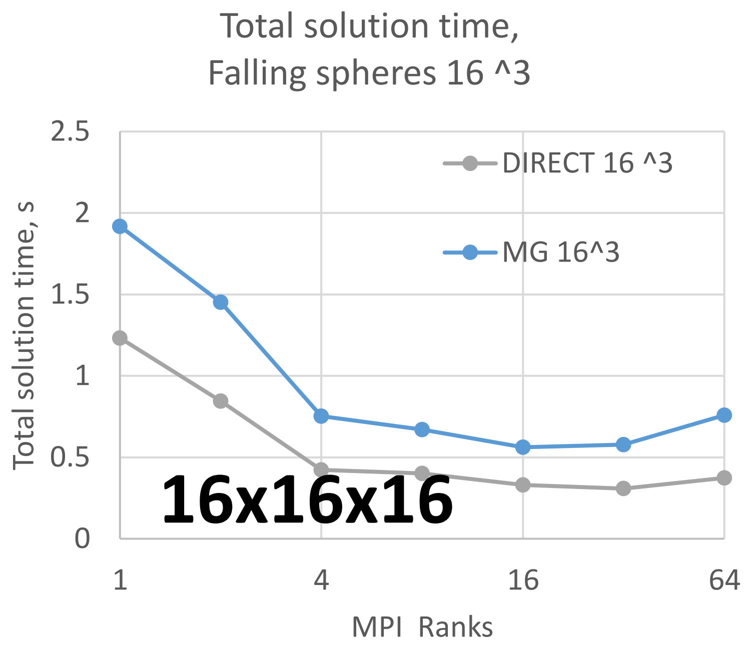
Computational cost:
Direct solver: $O(n^3)$
Iterative solvers (sparse): $O(nnz(A))+O(n) \approx O(n^2)$
Multigird: $O(n)$

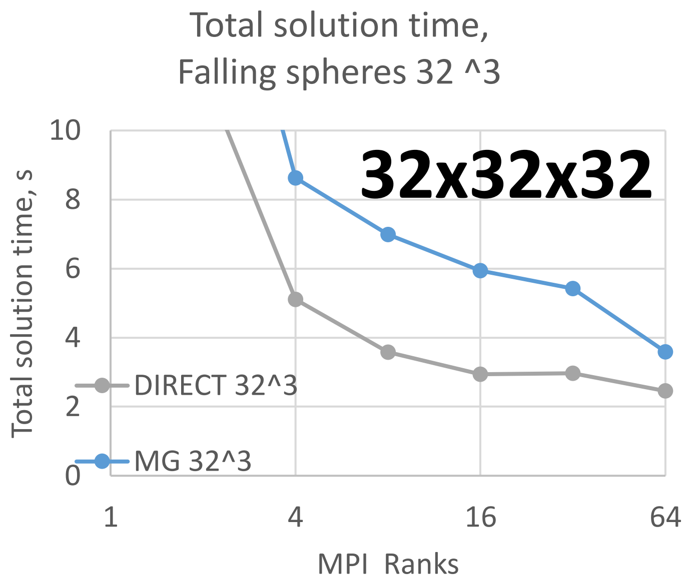
LaMEM

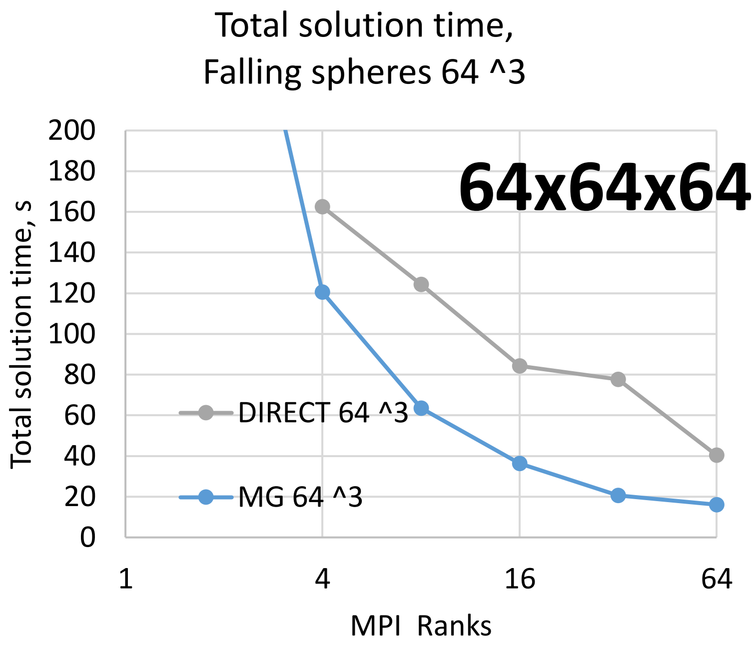
Problem size:

Resolution	Cell Number
8x8x8	512
16x16x16	4096
32x32x32	32768
64x64x64	262144
128x128x128	2097152









Acknowledgments

Anton Popov (JGU Mainz) – With multigrid code
Evangelos Moulas (JGU Mainz) – Consultation