Automatic Domain Decomposition for a Black-Box PDE Solver

Torsten Adolph and Willi Schönauer

Forschungszentrum Karlsruhe
Institute for Scientific Computing
Karlsruhe, Germany

torsten.adolph@iwr.fzk.de
willi.schoenauer@iwr.fzk.de

http://www.fzk.de/iwr
http://www.rz.uni-karlsruhe.de/rz/docs/FDEM/Literatur

Motivation

Numerical solution of non-linear systems of Partial Differential Equations (PDEs)

- **Finite Difference Method (FDM)**
- **Finite Element Method (FEM)**
- **Finite Volume Method (FVM)**

Finite Difference Element Method (FDEM)

Combination of advantages of FDM and FEM:
FDM on unstructured FEM grid

Objectives

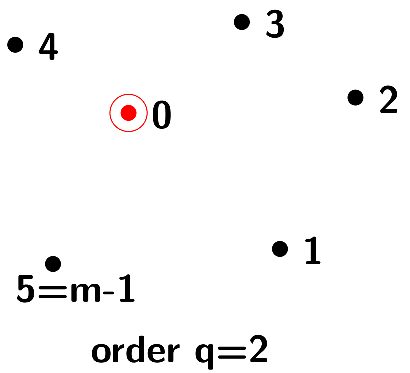
- Elliptic and parabolic non-linear systems of PDEs
- 2-D and 3-D with arbitrary geometry
- Arbitrary non-linear boundary conditions (BCs)
- Subdomains with different PDEs
- Robustness
- Black-box (PDEs/BCs and domain)
- Error estimate
- Order control/Mesh refinement
- Efficient parallelization

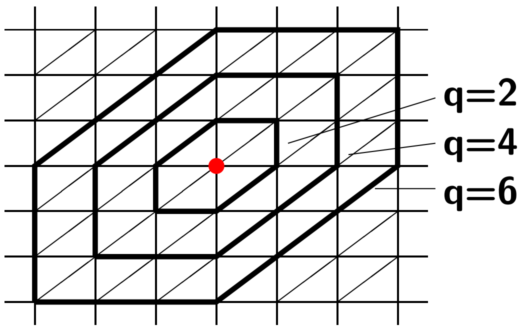
Difference formulas of order q on unstructured grid

Polynomial approach of order q (m coefficients)

2-D: m = (q+1)·(q+2)/2

3-D: m = (q+1)·(q+2)·(q+3)/6





Influence polynomial $P_{q,i} = \begin{cases} 1, & \text{node i} \\ 0, & \text{other nodes} \end{cases} \longrightarrow \quad u_d,\ u_{x,d},\ u_{y,d},\ u_{xx,d},\ u_{yy,d},\ u_{xy,d}$

Search for nodes in rings (up to order q+Δq) $\longrightarrow$ m+r nodes

Selection of m appropriate nodes by special algorithm

Discretization error estimate

e.g. for u_x: $$u_x = u_{x,d,q} + \bar{d}_{x,q} = u_{x,d,q+2} + \bar{d}_{x,q+2}$$

$$\rightarrow \quad d_{x,q} = u_{x,d,q+2} - u_{x,d,q} \left\{ + \bar{d}_{x,q+2} \right\}$$

Error equation

$$Pu \equiv P(t, x, y, u, u_t, u_x, u_y, u_{xx}, u_{yy}, u_{xy})$$

Linearization by Newton-Raphson

Discretization with error estimates d_t, d_x, ... and linearization in d_t, d_x, ...

$$\rightarrow \quad \Delta u_d = \Delta u_{Pu} + \Delta u_{D_t} + \Delta u_{D_x} + \Delta u_{D_y} + \Delta u_{D_{xy}} = \quad \text{(level of solution)}$$

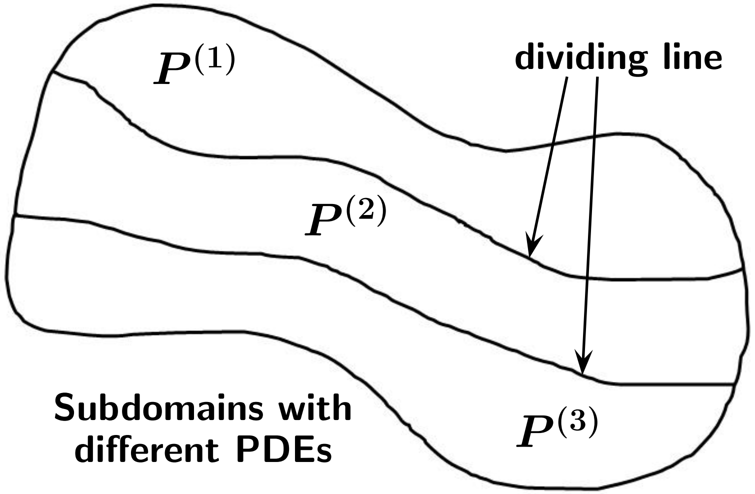
$$= Q_d^{-1} \cdot [(Pu)_d + D_t + \{D_x + D_y + D_{xy}\}] \quad \text{(level of equation)}$$

Only apply Newton correction Δu_{Pu}:

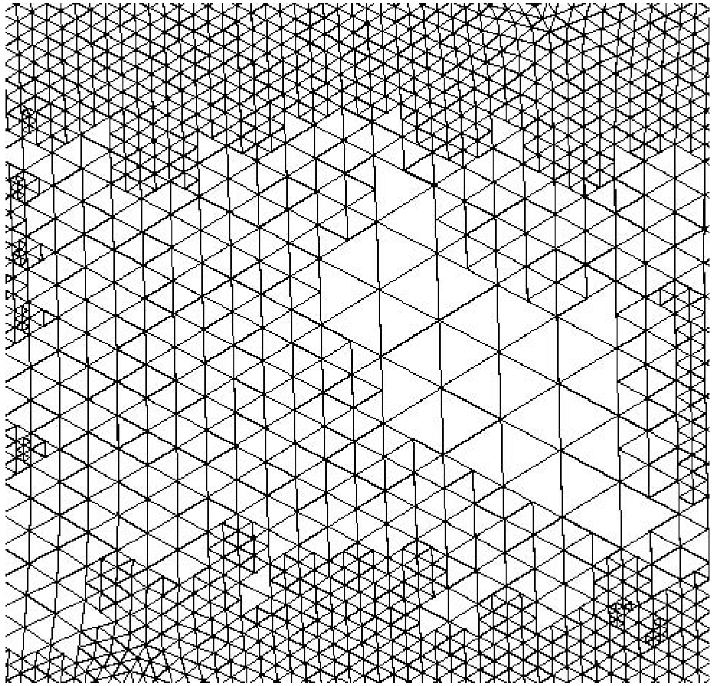
$$\rightarrow \quad Q_d \cdot \Delta u_{Pu} = (Pu)_d$$

Problem: Black-box for PDEs and domain

User input: any system of PDEs
any unstructured FEM grid
2-D and 3-D
(Sliding) dividing lines



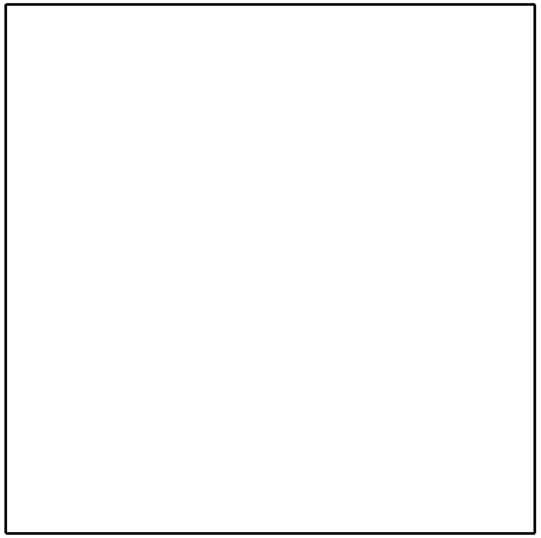
Local mesh refinement



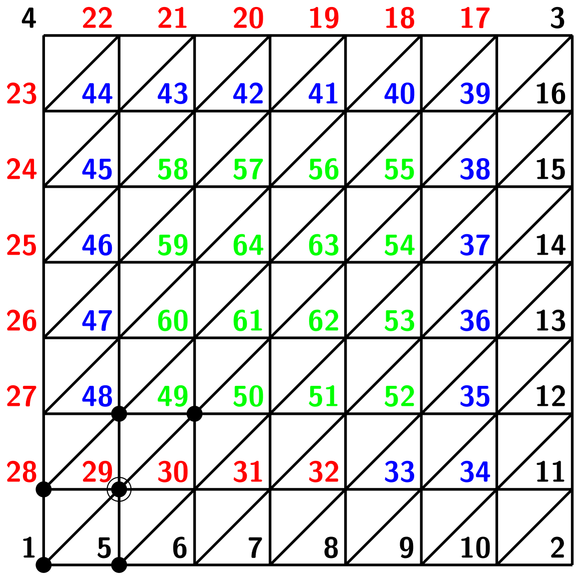
Solution: 1-D DD with overlap

Re-sorting of the nodes

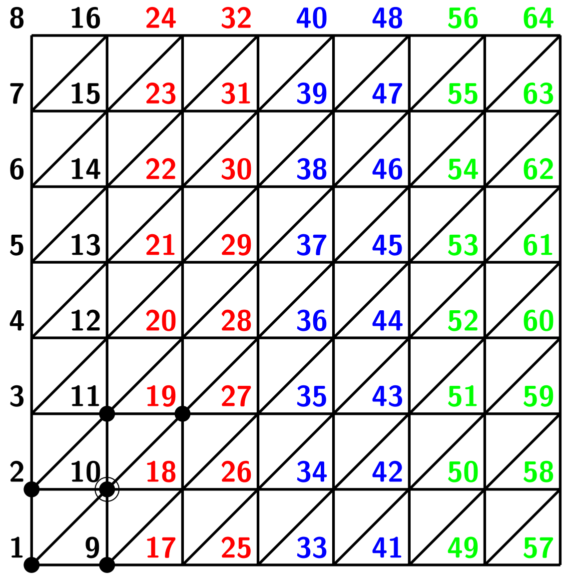
Domain



Node numbering
(from mesh generator)



Objective
(by re-sorting for x-coordinate)



Difference star: 29 1 5 28 48 49 → all 4 processors involved

10 1 2 9 11 19 → only 2 (neighboured) processors involved

Proc. 1 2 3 4

Algorithm for global sorting of the nodes I

- Needs 2·(np-1) steps on np processors
- Step i $\in \{$ 1, . . ., np-1 $\}$:
  Sorting until first sorted nodes are received by proc. np
- Step i $\in \{$ np, . . ., 2·(np-1) $\}$:
  Sorting, proc. np sends sorted nodes to processors 1 to np-1
- Always send to the right neighbour processor (except for processor np)
- Always receive from the left neighbour processor
- Up to np/2 processors active in parallel
- Communication via MPI
- Start with local sorting of the nodes (heapsort)
- Step after receiving nodes: merging (= local sorting)

Algorithm for global sorting of the nodes II

- Formal description of step i (illustration on next slide)

i odd	merge	$ip \geq \frac{i+3}{2}$	$\wedge$	$ip \leq \min(i,np)$
	send	$ip \geq \frac{i+1}{2}$	$\wedge$	$ip \leq \min(i,np)$
	receive	$ip \geq \frac{i+3}{2}$	$\wedge$	$ip \leq \min(i,np)$
		additionally: $i \in \{1, \ldots, np-1\}$: $ip = i+1$		
		$i \in \{np, \ldots, 2 \cdot (np-1)\}$: $ip = i-np+1$		
i even	merge	$ip \geq \frac{i}{2} + 1$	$\wedge$	$ip \leq \min(i,np)$
	send	$ip \geq \frac{i}{2} + 1$	$\wedge$	$ip \leq \min(i,np)$
	receive	$ip \geq \frac{i}{2} + 2$	$\wedge$	$ip \leq \min(i,np)$
		additionally: $i \in \{1, \ldots, np-1\}$: $ip = i+1$		
		$i \in \{np, \ldots, 2 \cdot (np-1)\}$: $ip = i-np+1$		

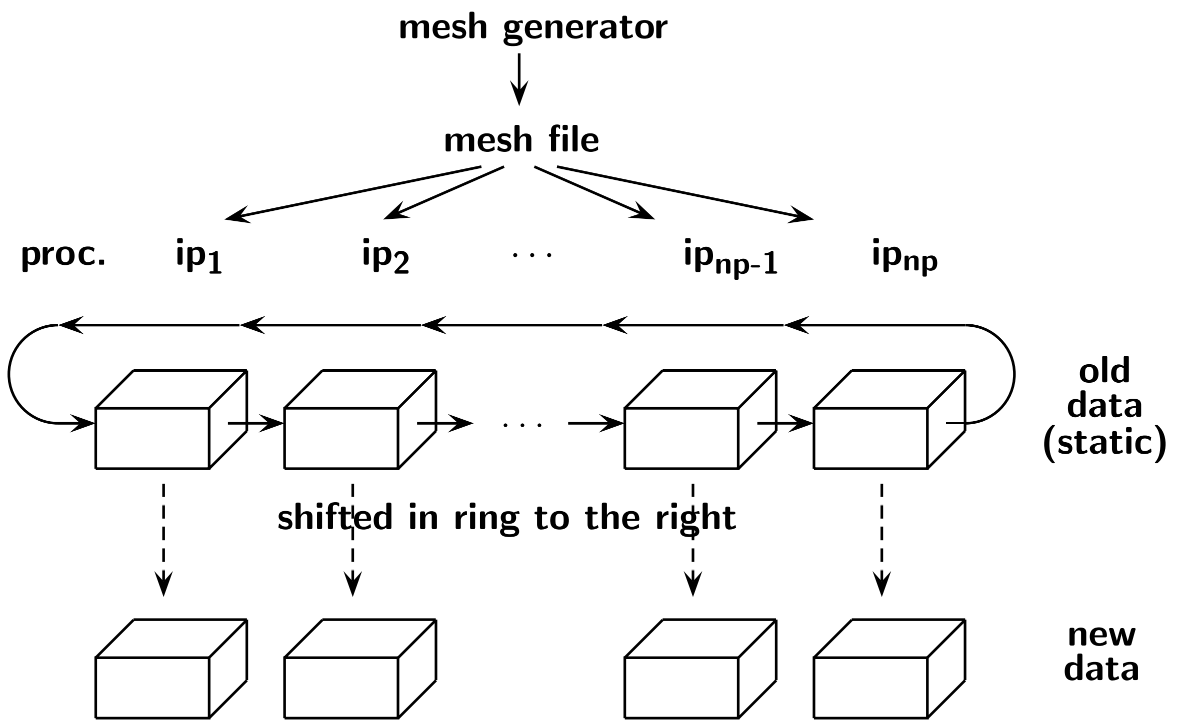
Illustration of the global re-sorting algorithm

Proc.	Step 1			2			3			4			5			6			7		
1		S																			
2			R	M	S			S													
3						R	M	S	R	M	S			S							
4									R	M	S	R	M	S	R	M	S			S	
5												R	M	S	R	M	S	R	M	S	R
6															R	M	S	R	M	S	R
7																		R	M	S	R
np=8																					R

Proc.	Step 8			9			10			11			12			13			14		
1			R																		
2						R															
3									R												
4												R									
5	M	S			S										R						
6	M	S	R	M	S	R	M	S			S							R			
7	M	S	R	M	S	R	M	S	R	M	S	R	M	S			S				R
np=8	M	S	R	M	S	R	M	S	R	M	S	R	M	S	R	M	S	R	M	S	

- **M** Merge
- **S** Send
- **R** Receive

Distribution of the elements



Processor that owns leftmost node of an element becomes element owner

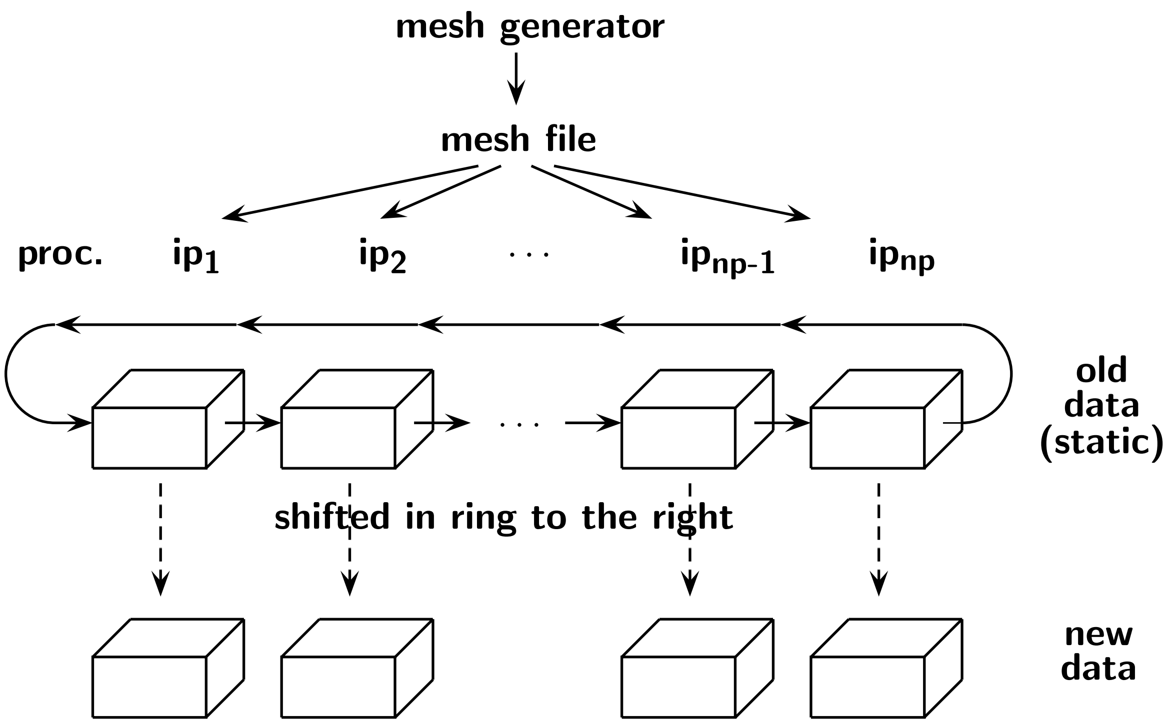
⟶ Execution of 2 ring shifts

1^{st}: Determination of owner of leftmost node

2^{nd}: Storing of element numbers on owning processors

Distribution of the boundary/DL/SDL nodes

DL: dividing line
SDL: sliding dividing line

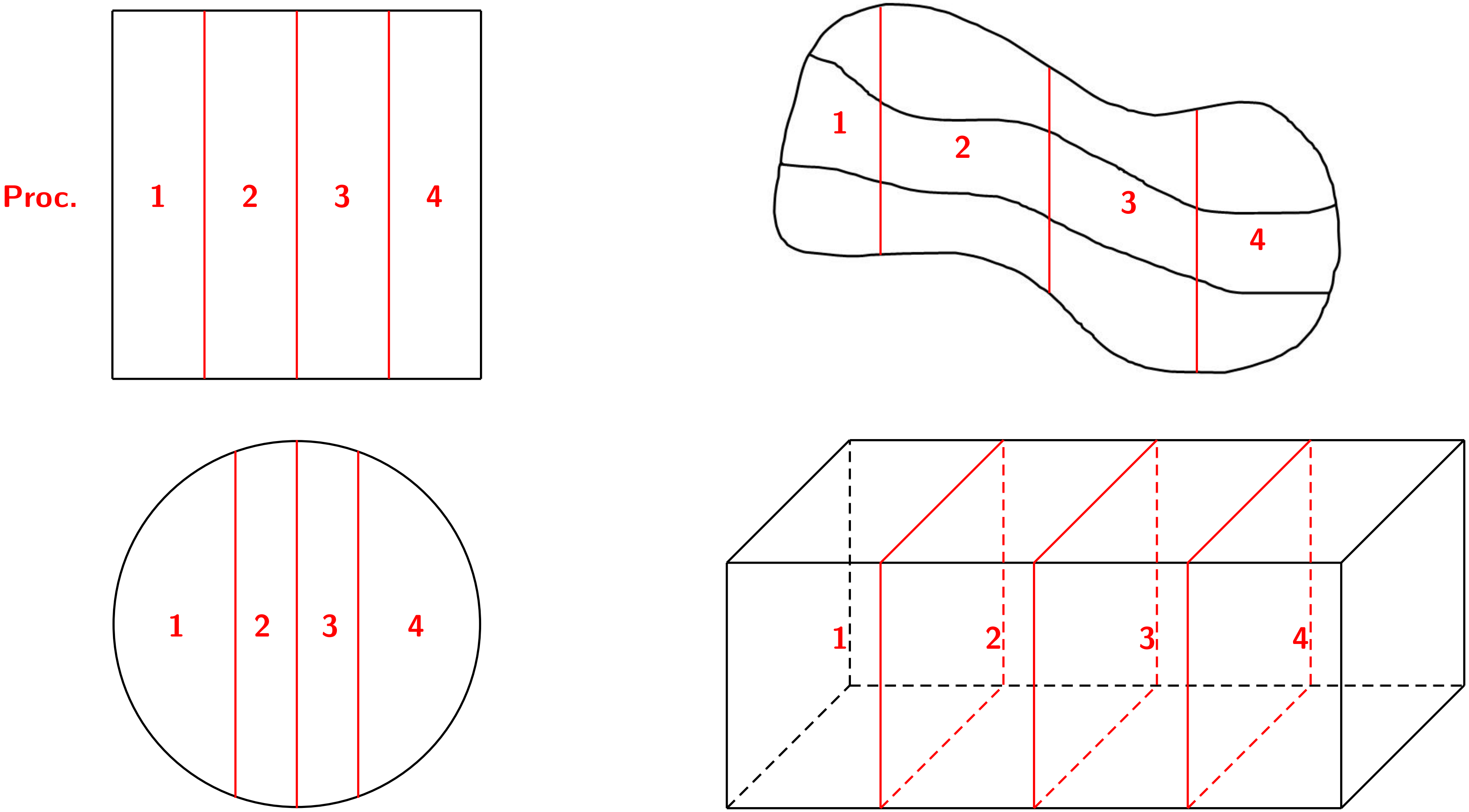


Compare received node numbers of boundary/DL/SDL nodes to node numbers of own nodes

⟶ Store matching node numbers in arrays for boundary/DL/SDL nodes

Send non-matching node numbers to right neighbour processor

Illustration of 1-D DD (np=4)



Overlap

Computation of the right hand side and of the matrix Q_d } local (without communication)

$\longrightarrow$ Store necessary nodes and elements of neighbour processors on proc. ip

Proc.	ip-1	ip	ip+1
	overlap	own	overlap

necessary nodes on proc. ip

ip-1, ip+1: overlap processors of proc. ip

Width of overlap:

Compute mean edge length h_{mean}

Choose safety factor $a_{overlap}$

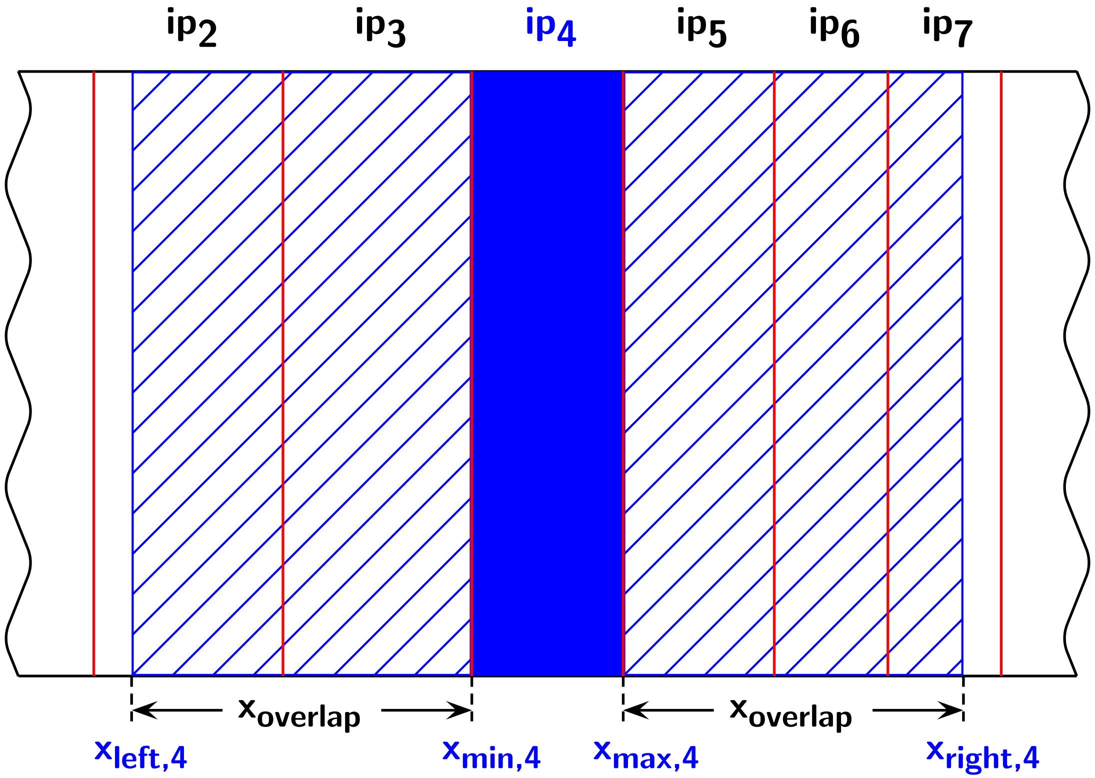
Compute $x_{overlap,1}$, $x_{overlap,2}$

1. criterion (enough nodes): $x_{overlap,1} = 0.5 \cdot a_{overlap} \cdot h_{mean} \cdot (\sqrt{m(q+\Delta q)} - 1)$
2. criterion (enough rings): $x_{overlap,2} = a_{overlap} \cdot h_{mean} \cdot (q+\Delta q)$

Compute $x_{overlap} = \max(x_{overlap,1}, x_{overlap,2})$

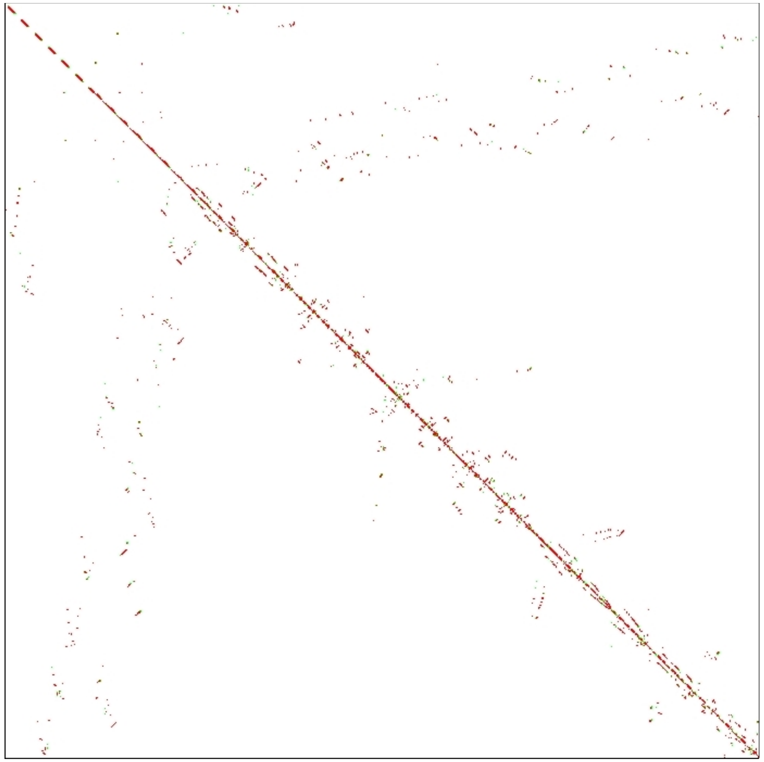
Illustration of overlap

$$x_{left} = x_{min} - x_{overlap}$$
$$x_{right} = x_{max} + x_{overlap}$$

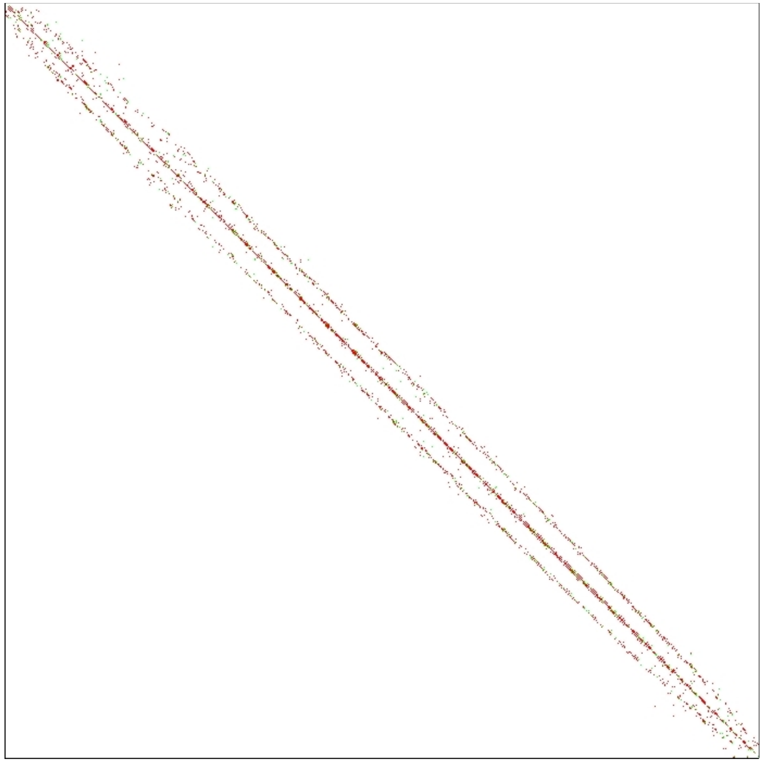


Bandwidth optimization

Before re-sorting



After re-sorting



Bandwidth:	**Full**	**2253**
	Before re-sorting	**2154**
	After re-sorting	**185**
	With SSP BO	**112**

SSP: own improved Cuthill-McKee

Summary

- **Black-box PDE solver FDEM (URL: http://www.rz.uni-karlsruhe.de/rz/docs/FDEM/Literatur)**
- **User input: any PDE system, any domain, 2-D and 3-D**
- **Global re-sorting algorithm for nodes**
- **Send elements in ring-shift to owning processors**

⟶ 1-D DD with overlap

- **Computation of linear system of equations purely local**
- **Efficient parallelization with MPI**
- **Built-in bandwidth optimizer**

This 1-D DD is simple, robust and efficient!