A User Friendly Toolbox for Parallel PDE-Solvers

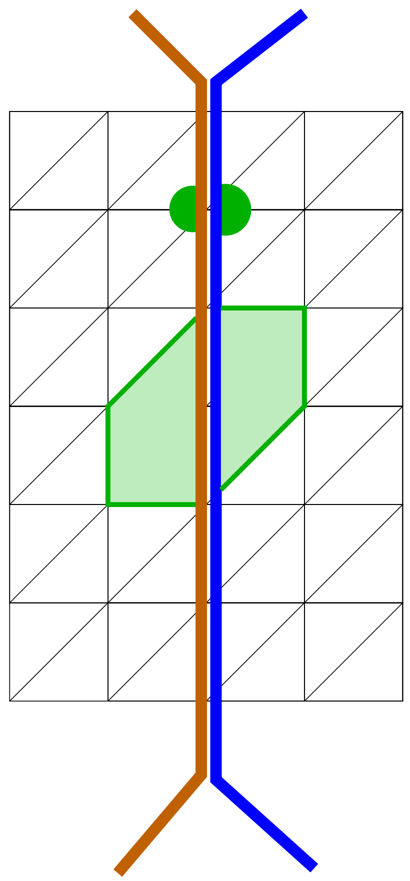
Gundolf Haase

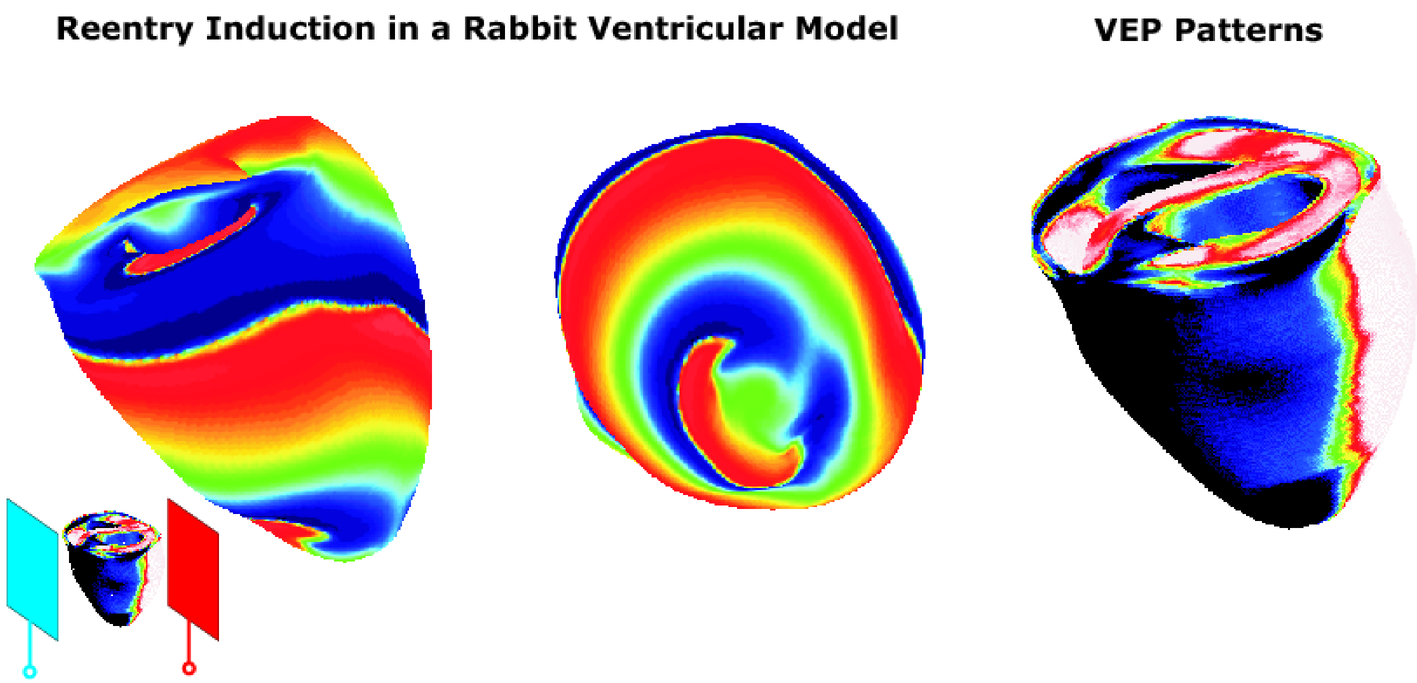
Institut for Mathematics and Scientific Computing
Karl-Franzens University of Graz

Manfred Liebmann

Mathematics in Sciences
Max-Planck-Institute Leipzig

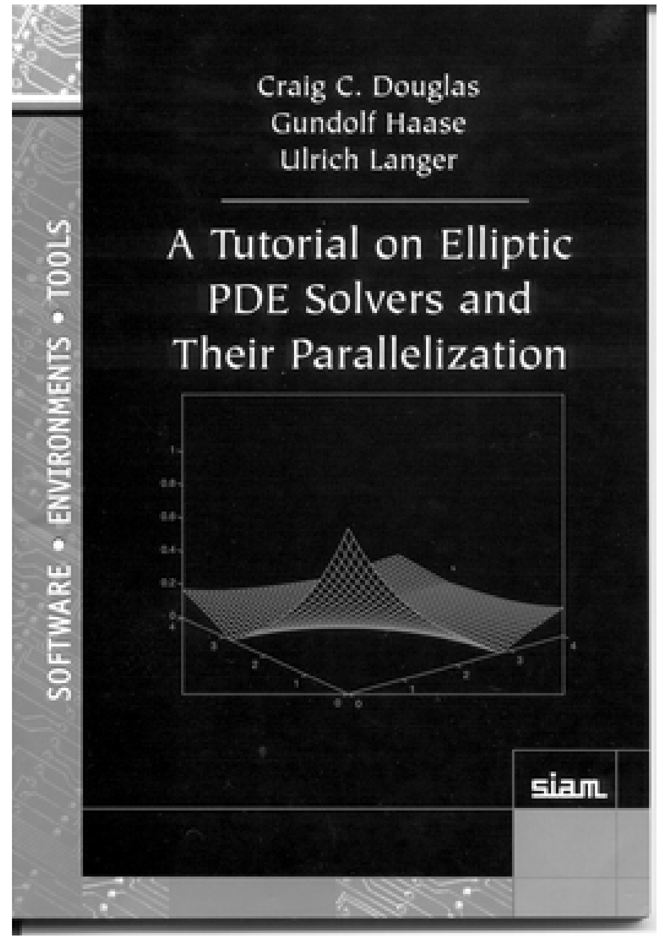
in cooperation with G. Planck [Med-Uni Graz]





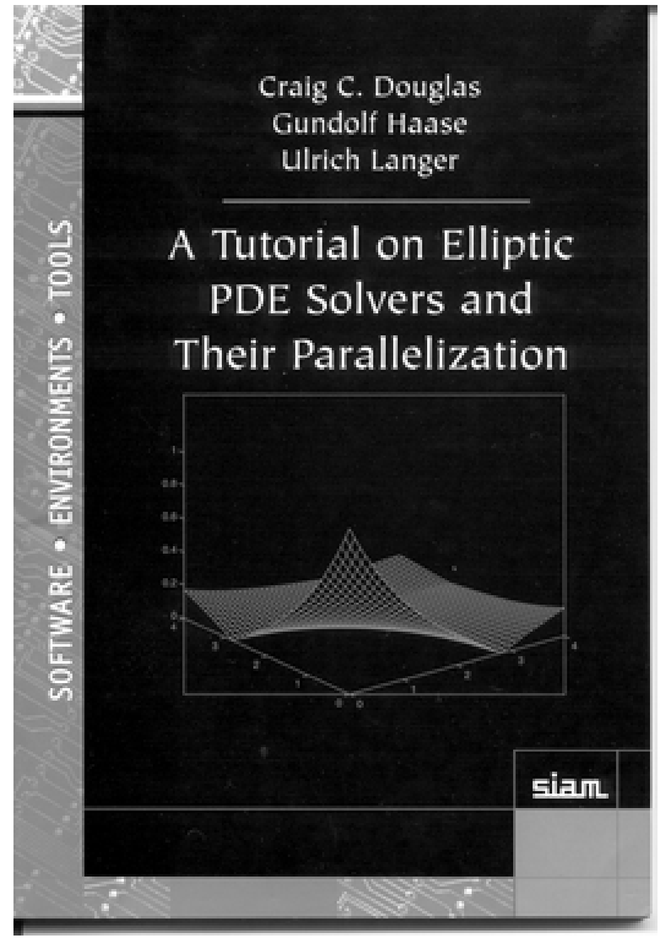
Contents

Motivation



- We have developed parallel codes for 15 years, incl. Multigrid, Multilevel, AMG, Krylov methods, ...
- We have a dozen of applications from potential problems, elasticity problems, Maxwell's equations.
- Approx. 25 licences for the parallel code PEBBLES.
- We have written a book on parallelization [Douglas/Haase/Langer].

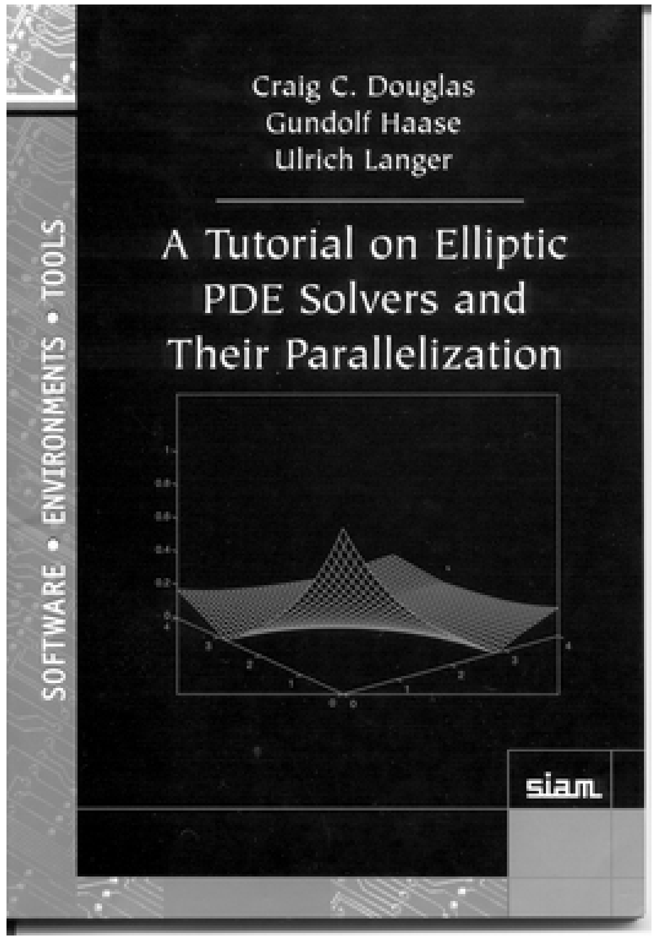
Motivation



- We have developed parallel codes for 15 years, incl. Multigrid, Multi-level, AMG, Krylov methods, ...
- We have a dozen of applications from potential problems, elasticity problems, Maxwell's equations.
- Approx. 25 licences for the parallel code PEBBLES.
- We have written a book on parallelization [Douglas/Haase/Langer].

What's wrong with our available codes?

Motivation

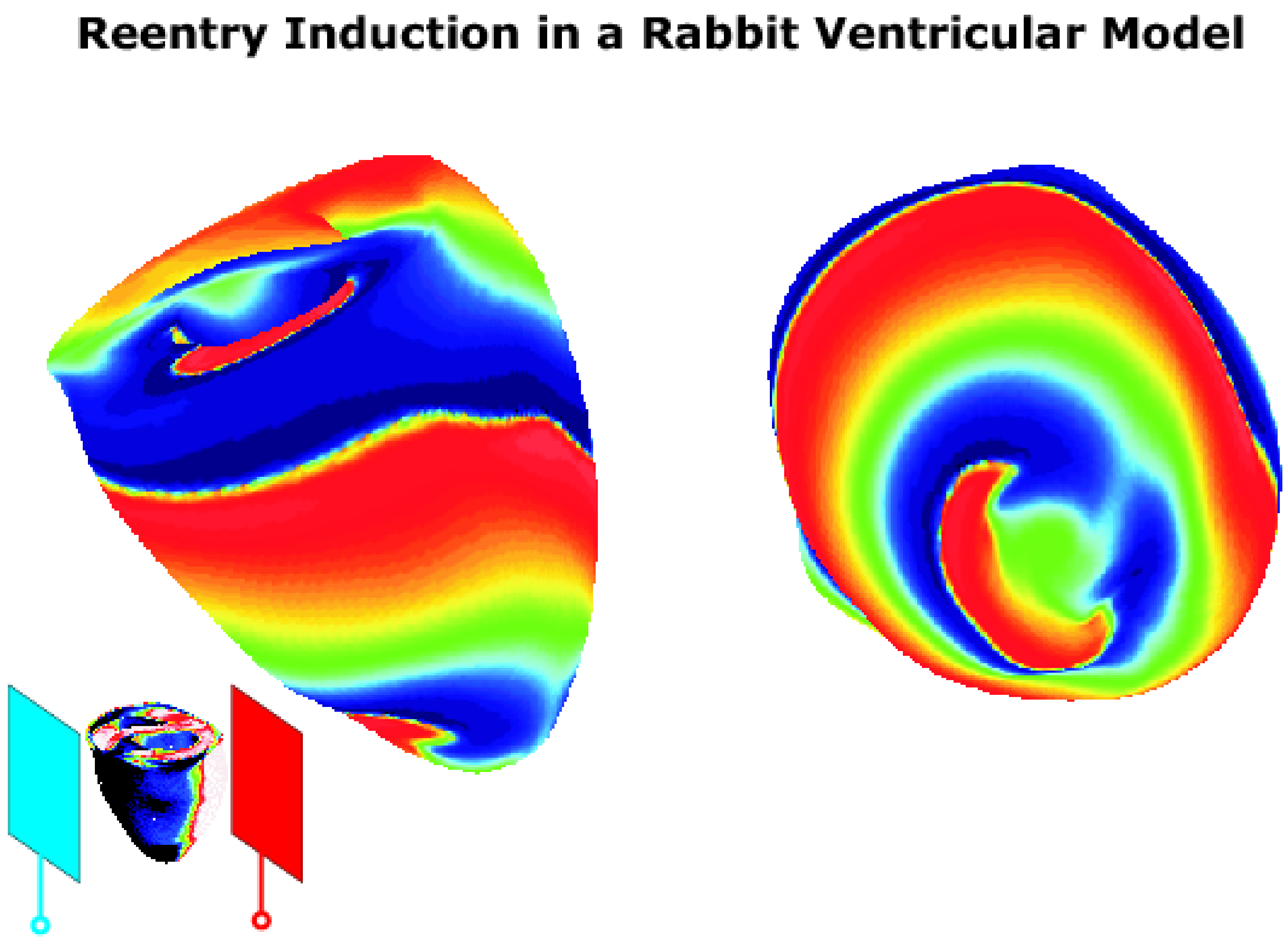


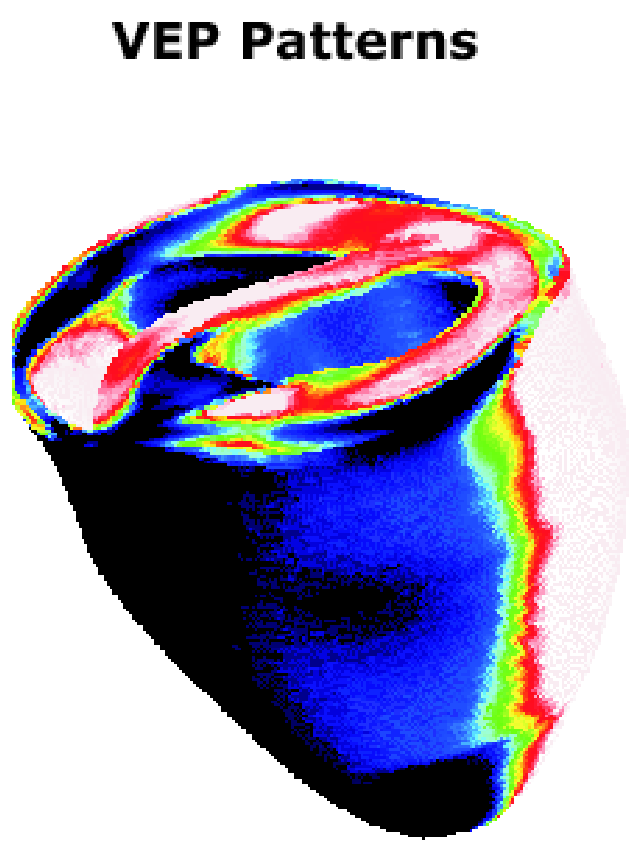
- We have developed parallel codes for 15 years, incl. Multigrid, Multilevel, AMG, Krylov methods, ...
- We have a dozen of applications from potential problems, elasticity problems, Maxwell's equations.
- Approx. 25 licences for the parallel code PEBBLES.
- We have written a book on parallelization [Douglas/Haase/Langer].

What's wrong with our available codes?

$\Longrightarrow$ Let's have a look at an example.

Rabbit Heart [G. Planck, M. Liebmann, G. Haase]





- time-dependent electrical potential, anisotropic coefficients
- 5.082.272 tetrahedrons with 862.525 FEM-nodes
- Goal: 150 Mill. tetrahedrons using parallel mesh generator Spider by F. Kickinger

PEBBLES as AMG–preconditioner in the heart problem ($\varepsilon = 10^6$)

- sequentially, 111.589 nodes, Pentium4 3GHz :

solver	solution [sec.]	Iterations
ILU/CG	12.0	211
Hypre	1.9	5
SuperLU	1.2	(but 70 sec. in setup)
PEBBLES/CG	0.8	10

- parallel, 862.515 nodes, Opteron nodes, PEBBLES:

processors	1	2	4	8
solver iterations	13	12	12	14
coarse grid	3059	4008	4850	3070
solver in sec.	10.3	9.0	5.0	3.2
	[0.3]	[0.6]	[0.7]	[0.4]
setup in sec.	37.1	22.4	17.9	11.3
	[3.7]	[7.0]	[9.7]	[5.4]

- Some obscurities wrt. *re*construction of PEBBLES.
- Our parallel data structures didn't fit into global code.
- Conclusion: It's worth to continue the development of the code. But we have to re–write the code.

Motivation for a new code

- PETSc [B. Smith] is used by cooperation partners but we need more/other information for an efficient parallel AMG.

Menu

Motivation for a new code

- PETSc [B. Smith] is used by cooperation partners but we need more/other information for an efficient parallel AMG.

⇒ We have to provide similar functionality as PETSc to convince partners to use our code.

Menu

Motivation for a new code

- PETSc [B. Smith] is used by cooperation partners but we need more/other information for an efficient parallel AMG.

$\Rightarrow$ We have to provide similar functionality as PETSc to convince partners to use our code.

- Handling of old code is too complicated, i.e., one developer has to be always available.
- The professional code cannot be used in education (too much overhead from data setup)
- Code developers left for jobs in industrie [M. Kuhn, S. Reitzinger]

Motivation for a new code

- PETSc [B. Smith] is used by cooperation partners but we need more/other information for an efficient parallel AMG.

⇒ We have to provide similar functionality as PETSc to convince partners to use our code.

- Handling of old code is too complicated, i.e., one developer has to be always available.

- The professional code cannot be used in education (too much overhead from data setup)

- Code developers left for jobs in industrie [M. Kuhn, S. Reitzinger]

⇒ Redesign of interfaces, data structures and funtionality.

Motivation for a new code

- PETSc [B. Smith] is used by cooperation partners but we need more/other information for an efficient parallel AMG.

$\Rightarrow$ We have to provide similar functionality as PETSc to convince partners to use our code.

- Handling of old code is too complicated, i.e., one developer has to be always available.
- The professional code cannot be used in education (too much overhead from data setup)
- Code developers left for jobs in industrie [M. Kuhn, S. Reitzinger]

$\Rightarrow$ Redesign of interfaces, data structures and funtionality.

Goal: Toolbox provides all needed **basic** routines for **parallel** funtionality.

Menu

Motivation for a new code

- PETSc [B. Smith] is used by cooperation partners but we need more/other information for an efficient parallel AMG.

$\Rightarrow$ We have to provide similar functionality as PETSc to convince partners to use our code.

- Handling of old code is too complicated, i.e., one developer has to be always available.
- The professional code cannot be used in education (too much overhead from data setup)
- Code developers left for jobs in industrie [M. Kuhn, S. Reitzinger]

$\Rightarrow$ Redesign of interfaces, data structures and funtionality.

Goal: Toolbox provides all needed **basic** routines for **parallel** funtionality.

Goal: Write your own parallel code by **re-using sequential** code.

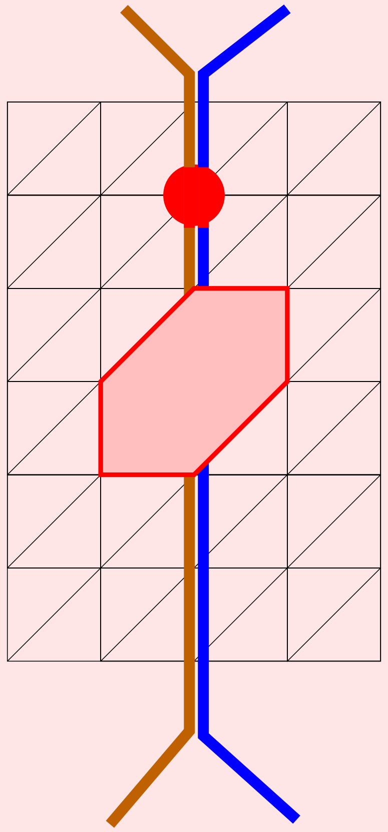
Menu

Parallel Algebra

- Concept for Parallelization
- Extended Parallelization Concept
- Parallel Multigrid
- Parallel factorization
- Parallelization of Algebraic Multigrid
- Contents

Non-overlapping Data Decomposition

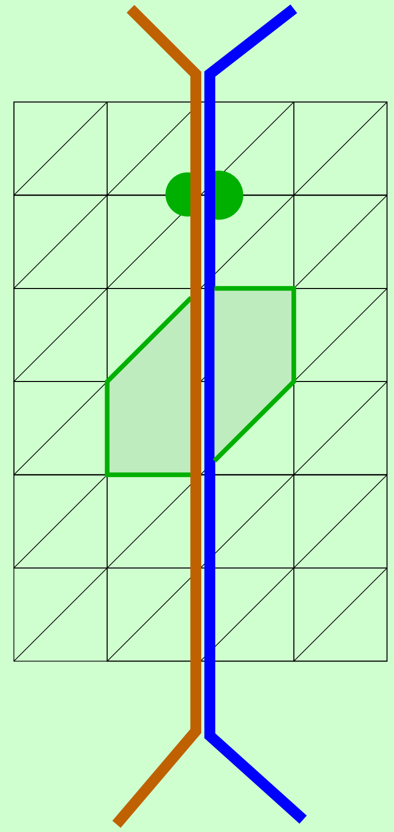
accumulated



$$\underline{\mathfrak{u}}_s = A_s \underline{\mathfrak{u}}$$

$$\mathfrak{M}_s = A_s \mathfrak{M} A_s^T$$

distributed



$$\underline{\mathsf{r}} = \sum_{s=1}^{P} A_s^T \underline{\mathsf{r}}_s$$

$$\mathsf{K} = \sum_{s=1}^{P} A_s^T \mathsf{K}_s^{\mathsf{FEM}} A_s$$

Overlapping Domain Decomposition

accumulated

$$\underline{\mathfrak{u}}_s = A_s \underline{\mathfrak{u}} \quad , \quad \mathfrak{M}_s = A_s \mathfrak{M} A_s^T$$

distributed

$$\underline{\mathsf{r}} = \sum_{s=1}^{P} A_s^T \underline{\mathsf{r}}_s \quad , \quad \mathsf{K} \stackrel{!}{=} \sum_{s=1}^{P} A_s \mathsf{K}_s A_s^T$$

Overlapping Domain Decomposition

<u>accumulated</u>

$$\underline{\mathfrak{u}}_s = A_s \underline{\mathfrak{u}} \quad , \quad \mathfrak{M}_s = A_s \mathfrak{M} A_s^T$$

<u>distributed</u>

$$\underline{\mathsf{r}} = \sum_{s=1}^{P} A_s^T \underline{\mathsf{r}}_s \quad , \quad \mathsf{K} \stackrel{!}{=} \sum_{s=1}^{P} A_s \mathsf{K}_s A_s^T$$

BUT, how to choose K_s ?

Overlapping Domain Decomposition

accumulated

$$\underline{\mathfrak{u}}_s = A_s \underline{\mathfrak{u}} \quad , \quad \mathfrak{M}_s = A_s \mathfrak{M} A_s^T$$

distributed

$$\underline{\mathsf{r}} = \sum_{s=1}^{P} A_s^T \underline{\mathsf{r}}_s \quad , \quad \mathsf{K} \stackrel{!}{=} \sum_{s=1}^{P} A_s \mathsf{K}_s A_s^T$$

BUT, how to choose K_s ?

$$\mathsf{K}_s := \sum_{\delta^{(r)} \subseteq \Omega_s} \frac{1}{W^{(r)}} \cdot K^{\mathsf{FEM},r}$$

$W^{(r)} := \# \, \Omega_s$ an element $\delta^{(r)}$ associated with..

Basic Operations

<u>without communication</u>

$$\underline{\mathsf{v}} \leftarrow \mathsf{K} \cdot \underline{\mathfrak{s}}$$

$$\underline{\mathsf{r}} \leftarrow \underline{\mathsf{f}} + \alpha \cdot \underline{\mathsf{v}}$$

$$\underline{\mathfrak{w}} \leftarrow \underline{\mathfrak{u}} + \alpha \cdot \underline{\mathfrak{s}}$$

$$\underline{\mathsf{r}} \leftarrow R^{-1} \cdot \underline{\mathfrak{w}}$$

<u>global reduce</u>

$$\alpha \Leftarrow \langle \mathfrak{w}, \mathsf{r} \rangle = \sum_{s=1}^{P} \langle \mathfrak{w}_s, \mathsf{r}_s \rangle$$

<u>next neighbor comm.</u>

$$\underline{\mathfrak{w}} \leftarrow \underline{\mathsf{r}} = \sum_{s=1}^{P} A_s^T \underline{\mathsf{r}}_s$$

with $R = \text{diag}\{R_{ii}\}_{i=1}^{N} = \text{diag}\{\#\text{ subdomains } x_i \text{ is associated with}\} = \sum_{s=1}^{P} A_s^T \cdot A_s$

and $\mathsf{R}^{-1} = \sum_{s=1}^{P} A_s^T \cdot \mathbf{I}_s \cdot A_s$ (partition of unity)

Parallel CG : PCG(K, $\underline{\mathfrak{u}}$, $\underline{\mathsf{f}}$)

repeat

$$
\begin{aligned}
\underline{\mathsf{v}} &\leftarrow \mathsf{K} \cdot \underline{\mathfrak{s}} \\
\alpha &\Leftarrow \sigma / \langle \underline{\mathfrak{s}}, \underline{\mathsf{v}} \rangle \\
\underline{\mathfrak{u}} &\leftarrow \underline{\mathfrak{u}} + \alpha \underline{\mathfrak{s}} \\
\underline{\mathsf{r}} &\leftarrow \underline{\mathsf{r}} - \alpha \underline{\mathsf{v}} \\
\underline{\mathfrak{w}} &\Leftarrow C^{-1} \cdot \underline{\mathsf{r}} \\
\sigma &\Leftarrow \langle \underline{\mathfrak{w}}, \underline{\mathsf{r}} \rangle \\
\beta &\leftarrow \sigma / \sigma_{\text{old}} \quad , \quad \sigma_{\text{old}} \leftarrow \sigma \\
\underline{\mathfrak{s}} &\leftarrow \underline{\mathfrak{w}} + \beta \underline{\mathfrak{s}}
\end{aligned}
$$

until termination

Menü

Basic Operations (revisited)

without communication

$$\underline{\mathsf{v}} \leftarrow \mathsf{K} \cdot \underline{\mathfrak{s}}$$

$$\underline{\mathsf{r}} \leftarrow \underline{\mathsf{f}} + \alpha \cdot \underline{\mathsf{v}}$$

$$\underline{\mathfrak{w}} \leftarrow \underline{\mathfrak{u}} + \alpha \cdot \underline{\mathfrak{s}}$$

$$\underline{\mathsf{r}} \leftarrow R^{-1} \cdot \underline{\mathfrak{w}}$$

global reduce

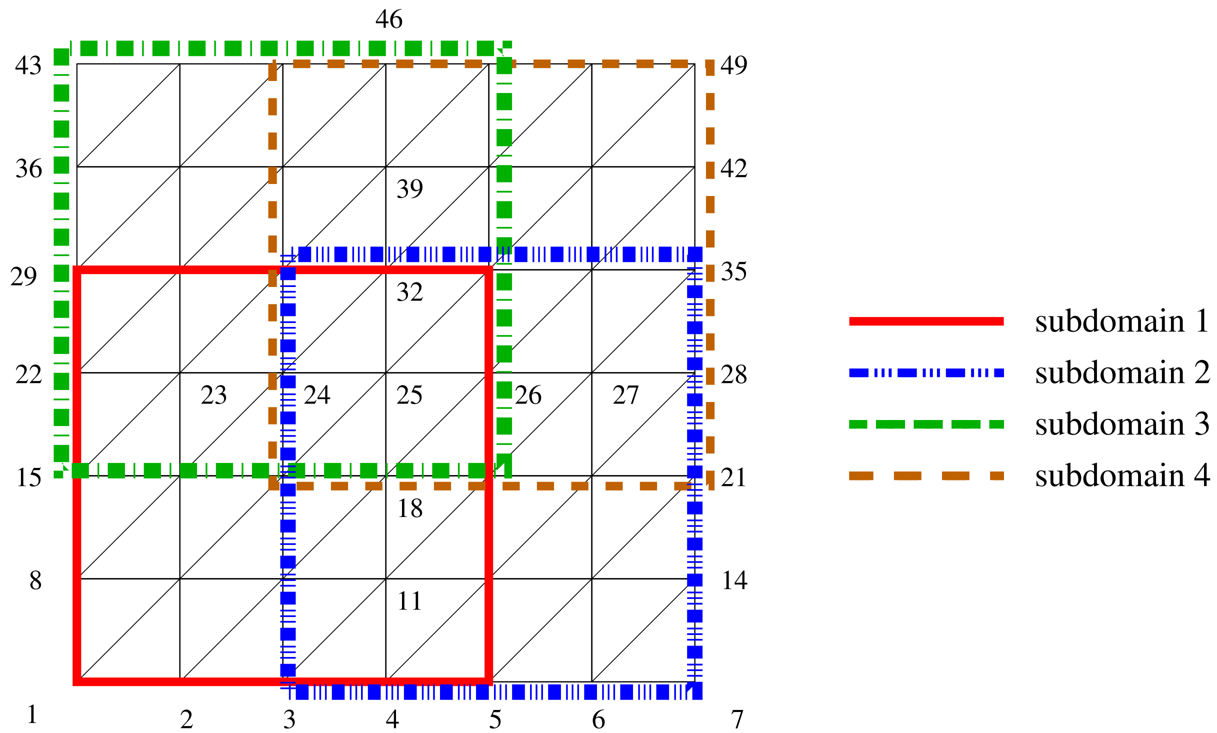
$$\alpha \Leftarrow \langle \mathfrak{w}, \mathsf{r} \rangle = \sum_{s=1}^{P} \langle \mathfrak{w}_s, \mathsf{r}_s \rangle$$

next neighbor comm.

$$\underline{\mathfrak{w}} \leftarrow \underline{\mathsf{r}} = \sum_{s=1}^{P} A_s^T \underline{\mathsf{r}}_s$$

What can be done with $\mathfrak{M} \cdot \underline{v}$?

Some Definitions



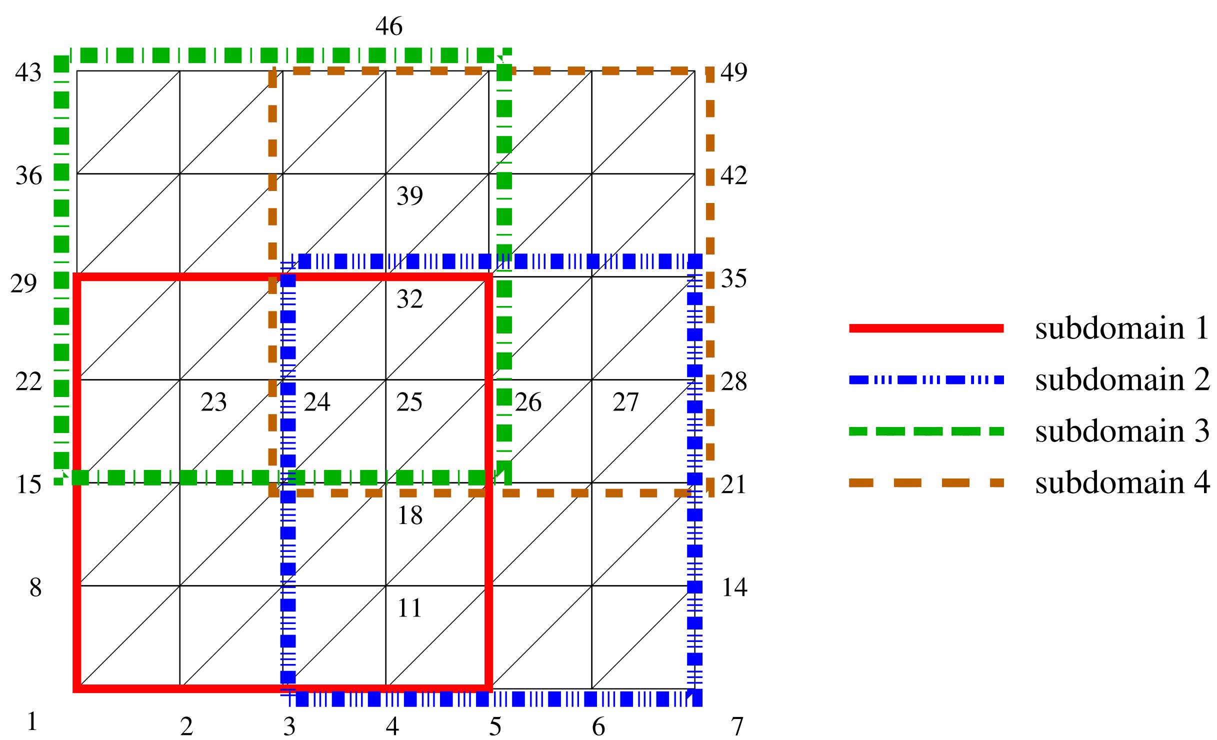
Set of subdomains :

$$\sigma^{[i]} = \{ s : x^{[i]} \in \overline{\Omega}_s \}$$

Set of indices/nodes :

$$\omega(\sigma) := \{ i \in \omega : \sigma^{[i]} = \sigma \}$$

Some Definitions



Set of subdomains :

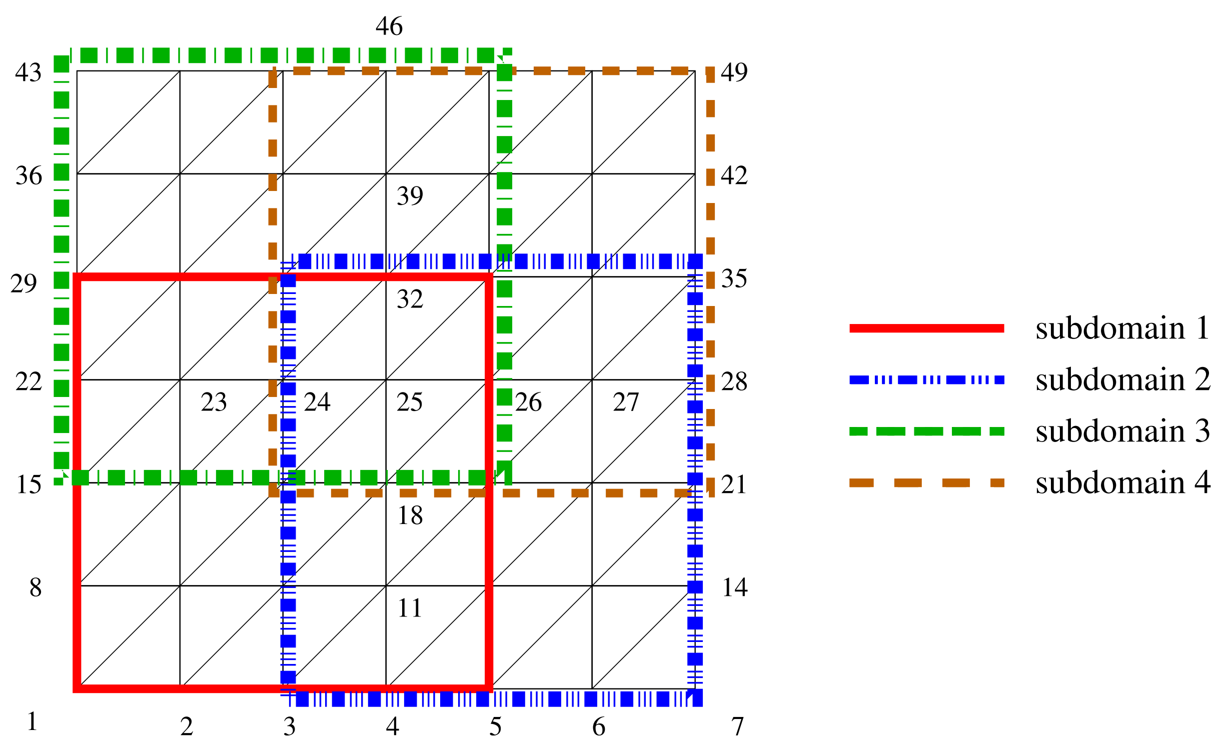
$$\sigma^{[i]} = \{ s : x^{[i]} \in \overline{\Omega}_s \}$$

Set of indices/nodes :

$$\omega(\sigma) := \{ i \in \omega : \sigma^{[i]} = \sigma \}$$

Bsp: $\sigma^{[11]} = \{1, 2\}$ $\quad \omega(\sigma^{[11]}) = \{3, 4, 5, 10, 11, 12\}$

Some Definitions



Set of subdomains :

$$\sigma^{[i]} = \{ s : x^{[i]} \in \overline{\Omega}_s \}$$

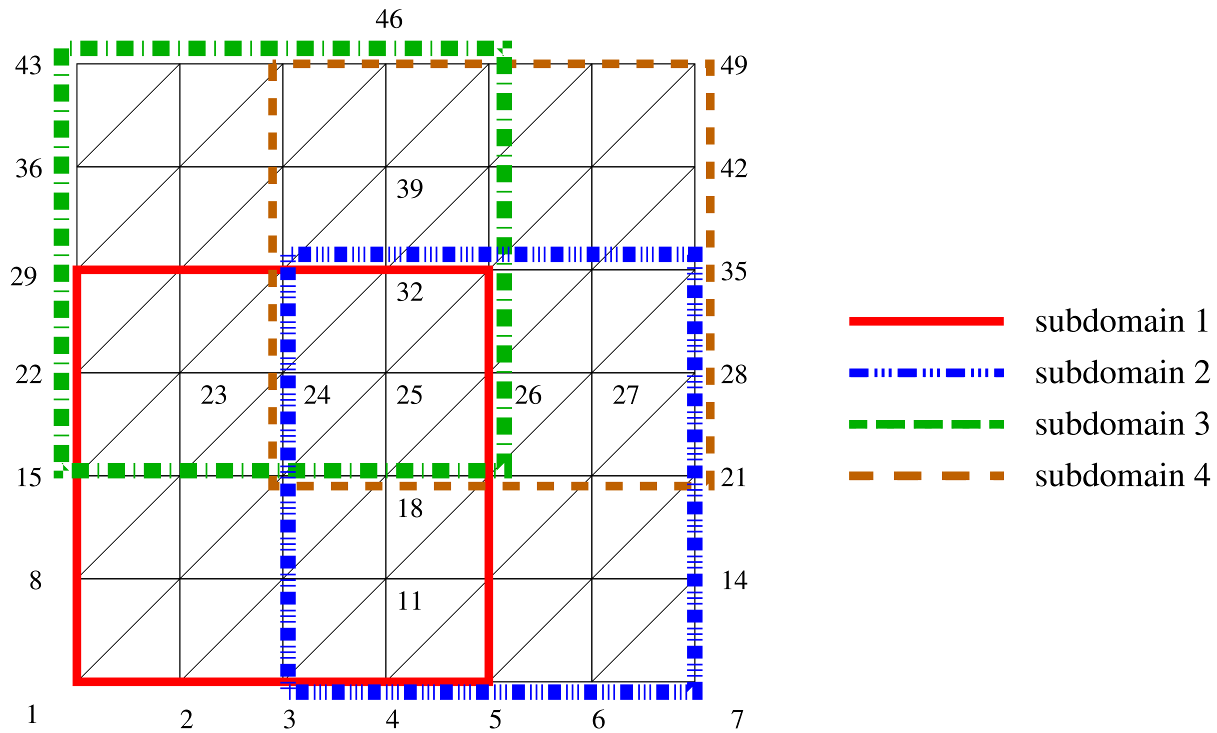
Set of indices/nodes :

$$\omega(\sigma) := \{ i \in \omega : \sigma^{[i]} = \sigma \}$$

Bsp: $\sigma^{[11]} = \{1, 2\}$ $\omega(\sigma^{[11]}) = \{3, 4, 5, 10, 11, 12\}$

$\sigma^{[27]} = \{2, 4\}$ $\omega(\sigma^{[27]}) = \{20, 21, 27, 28, 34, 35\}$

Some Definitions



Set of subdomains :

$$\sigma^{[i]} \;=\; \{s \,:\, x^{[i]} \in \overline{\Omega}_s\}$$

Set of indices/nodes :

$$\omega(\sigma) \;:=\; \{i \in \omega \,:\, \sigma^{[i]} = \sigma\}$$

Bsp: $\sigma^{[11]} = \{1, 2\}$ $\quad \omega(\sigma^{[11]}) = \{3, 4, 5, 10, 11, 12\}$

$\sigma^{[27]} = \{2, 4\}$ $\quad \omega(\sigma^{[27]}) = \{20, 21, 27, 28, 34, 35\}$

$\sigma^{[14]} = \{2\}$ $\quad \omega(\{2\}) = \{6, 7, 13, 14\}$

Matrix Patterns and their Application

The following operations can be performed in parallel without any communication:

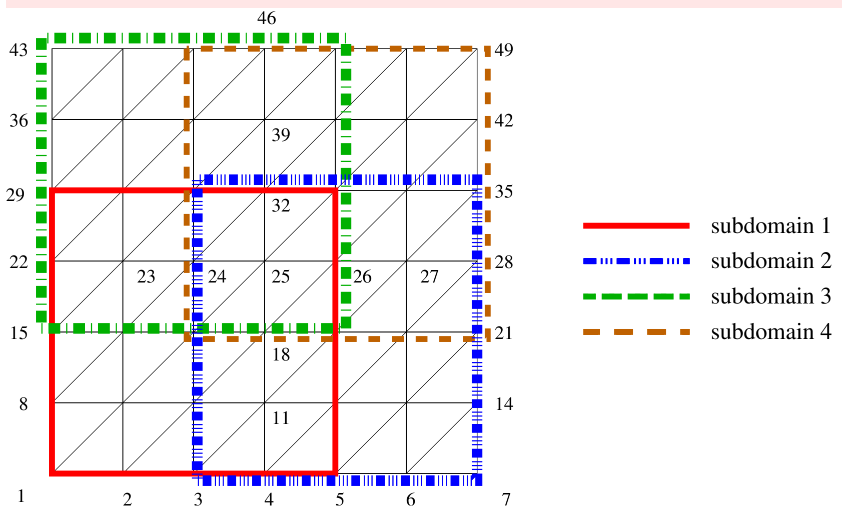
$$\underline{\mathsf{f}} = \mathsf{K} \cdot \underline{\mathfrak{u}}$$

Matrix $\mathfrak{M}$ fulfills the pattern condition: $\forall i,j \in \omega : \quad \sigma^{[i]} \not\subseteq \sigma^{[j]} \implies \mathfrak{M}^{[i,j]} = 0 \qquad i \not\longleftarrow j$

$$\underline{\mathfrak{u}} = \mathfrak{M} \cdot \underline{\mathfrak{w}}$$
$$\underline{\mathsf{f}} = \mathfrak{M}^T \cdot \underline{\mathsf{r}}$$

$$\boxed{\mathsf{K}^H = \mathfrak{M}^T \cdot \mathsf{K} \cdot \mathfrak{M}}$$

Theorems/Proofs **[Haase]**



Ex.:

Matrix Patterns and their Application

The following operations can be performed in parallel without any communication:

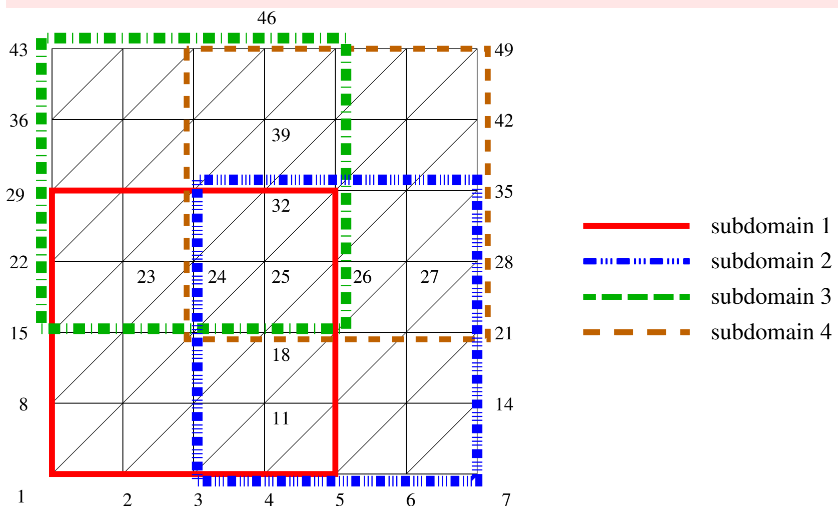
$$\underline{\mathsf{f}} = \mathsf{K} \cdot \underline{\mathfrak{u}}$$

Matrix $\mathfrak{M}$ fulfills the pattern condition: $\forall i,j \in \omega : \quad \sigma^{[i]} \not\subseteq \sigma^{[j]} \Longrightarrow \mathfrak{M}^{[i,j]} = 0 \qquad i \not\longleftarrow j$

$$\begin{aligned} \underline{\mathfrak{u}} &= \mathfrak{M} \cdot \underline{\mathfrak{w}} \\ \underline{\mathsf{f}} &= \mathfrak{M}^T \cdot \underline{\mathsf{r}} \end{aligned}$$

$$\boxed{\mathsf{K}^H = \mathfrak{M}^T \cdot \mathsf{K} \cdot \mathfrak{M}}$$

Theorems/Proofs **[Haase]**



Ex.: $\quad \sigma^{[11]} \not\subseteq \sigma^{[27]} \Longrightarrow \mathfrak{M}^{[11,27]} = 0 \qquad 12 \not\longleftarrow 20$

Matrix Patterns and their Application

The following operations can be performed in parallel without any communication:

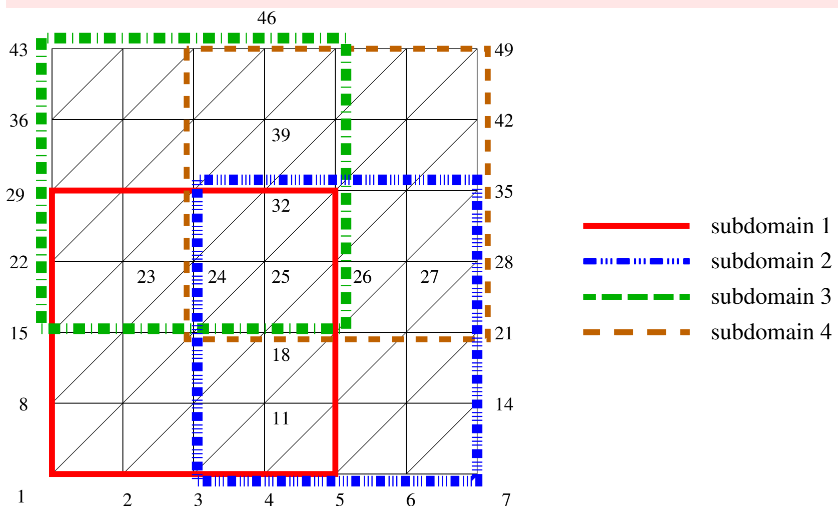
$$\underline{\mathsf{f}} \; = \; \mathsf{K} \cdot \underline{\mathfrak{u}}$$

Matrix $\mathfrak{M}$ fulfills the pattern condition: $\forall i,j \in \omega : \quad \sigma^{[i]} \not\subseteq \sigma^{[j]} \implies \mathfrak{M}^{[i,j]} = 0 \qquad i \not\longleftarrow j$

$$\begin{aligned} \underline{\mathfrak{u}} &= \mathfrak{M} \cdot \underline{\mathfrak{w}} \\ \underline{\mathsf{f}} &= \mathfrak{M}^T \cdot \underline{\mathsf{r}} \end{aligned}$$

$$\boxed{\mathsf{K}^H \; = \; \mathfrak{M}^T \cdot \mathsf{K} \cdot \mathfrak{M}}$$

Theorems/Proofs **[Haase]**



Ex.: $\sigma^{[11]} \not\subseteq \sigma^{[27]} \implies \mathfrak{M}^{[11,27]} = 0 \qquad 12 \not\longleftarrow 20$

$\sigma^{[27]} \not\subseteq \sigma^{[11]} \implies \mathfrak{M}^{[27,11]} = 0 \qquad 20 \not\longleftarrow 12$

Matrix Patterns and their Application

The following operations can be performed in parallel without any communication:

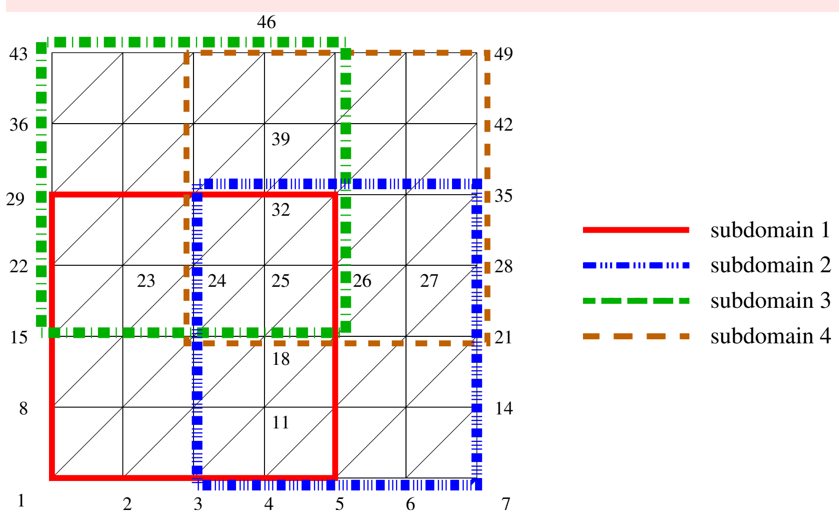
$$\underline{\mathsf{f}} \;=\; \mathsf{K} \cdot \underline{\mathfrak{u}}$$

Matrix $\mathfrak{M}$ fulfills the pattern condition: $\forall i, j \in \omega : \quad \sigma^{[i]} \not\subseteq \sigma^{[j]} \implies \mathfrak{M}^{[i,j]} = 0 \quad i \not\longleftarrow j$

$$\begin{aligned} \underline{\mathfrak{u}} &= \mathfrak{M} \cdot \underline{\mathfrak{w}} \\ \underline{\mathsf{f}} &= \mathfrak{M}^T \cdot \underline{\mathsf{r}} \end{aligned}$$

$$\boxed{\mathsf{K}^H \;=\; \mathfrak{M}^T \cdot \mathsf{K} \cdot \mathfrak{M}}$$

Theorems/Proofs **[Haase]**



Ex.:

$$\begin{aligned} \sigma^{[11]} \not\subseteq \sigma^{[27]} &\implies \mathfrak{M}^{[11,27]} = 0 \qquad 12 \not\longleftarrow 20 \\ \sigma^{[27]} \not\subseteq \sigma^{[11]} &\implies \mathfrak{M}^{[27,11]} = 0 \qquad 20 \not\longleftarrow 12 \\ \sigma^{[11]} \not\subseteq \sigma^{[14]} &\implies \mathfrak{M}^{[11,14]} = 0 \qquad 12 \not\longleftarrow 13 \end{aligned}$$

Matrix Patterns and their Application

The following operations can be performed in parallel without any communication:

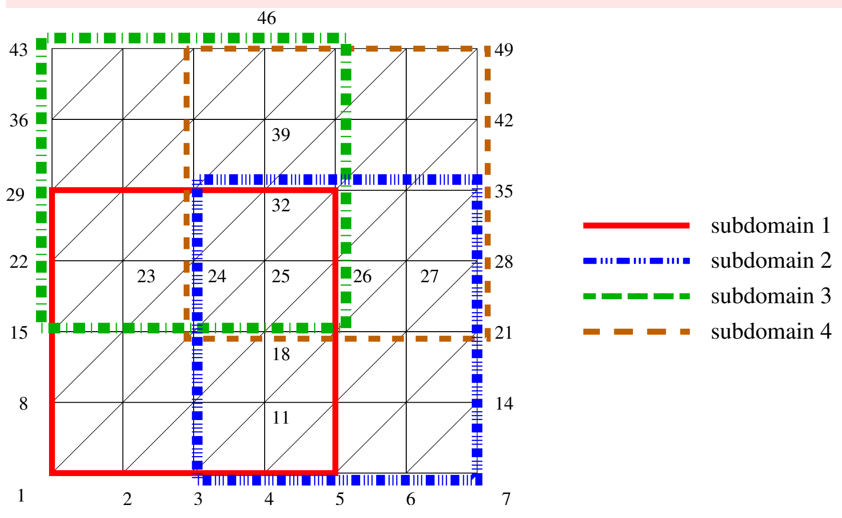
$$\underline{\mathsf{f}} = \mathsf{K} \cdot \underline{\mathfrak{u}}$$

Matrix $\mathfrak{M}$ fulfills the pattern condition: $\forall \boldsymbol{i}, \boldsymbol{j} \in \boldsymbol{\omega} : \quad \boldsymbol{\sigma}^{[i]} \not\subseteq \boldsymbol{\sigma}^{[j]} \Longrightarrow \mathfrak{M}^{[i,j]} = \mathbf{0} \quad \boldsymbol{i} \not\longleftarrow \boldsymbol{j}$

$$\underline{\mathfrak{u}} = \mathfrak{M} \cdot \underline{\mathfrak{w}}$$
$$\underline{\mathsf{f}} = \mathfrak{M}^{\boldsymbol{T}} \cdot \underline{\mathsf{r}}$$

$$\boxed{\mathsf{K}^{\boldsymbol{H}} = \mathfrak{M}^{\boldsymbol{T}} \cdot \mathsf{K} \cdot \mathfrak{M}}$$

Theorems/Proofs **[Haase]**



Ex.:

$$\boldsymbol{\sigma}^{[11]} \not\subseteq \boldsymbol{\sigma}^{[27]} \Longrightarrow \mathfrak{M}^{[11,27]} = \mathbf{0} \qquad \mathbf{12} \not\longleftarrow \mathbf{20}$$
$$\boldsymbol{\sigma}^{[27]} \not\subseteq \boldsymbol{\sigma}^{[11]} \Longrightarrow \mathfrak{M}^{[27,11]} = \mathbf{0} \qquad \mathbf{20} \not\longleftarrow \mathbf{12}$$
$$\boldsymbol{\sigma}^{[11]} \not\subseteq \boldsymbol{\sigma}^{[14]} \Longrightarrow \mathfrak{M}^{[11,14]} = \mathbf{0} \qquad \mathbf{12} \not\longleftarrow \mathbf{13}$$
$$\boldsymbol{\sigma}^{[27]} \not\subseteq \boldsymbol{\sigma}^{[14]} \Longrightarrow \mathfrak{M}^{[27,14]} = \mathbf{0} \qquad \mathbf{21} \not\longleftarrow \mathbf{14}$$

Admissible Matrix Operations

Vertex, **E**dge, **I**nner nodes

$$\mathfrak{M} = \begin{pmatrix} \mathfrak{M}_V & 0 & 0 \\ \mathfrak{M}_{EV} & \mathfrak{M}_E & 0 \\ \mathfrak{M}_{IV} & \mathfrak{M}_{IE} & \mathfrak{M}_I \end{pmatrix} = \mathfrak{M}_L + \mathfrak{M}_D \quad \Longrightarrow \quad \underline{\mathfrak{u}} = \mathfrak{M} \cdot \underline{\mathfrak{w}}$$

Pattern condition $\boldsymbol{\sigma}^{[i]} \not\subseteq \boldsymbol{\sigma}^{[j]} \implies \mathfrak{M}^{[i,j]} = \mathbf{0}$ has to be fulfilled in all submatrices!!

Allows operations as (Parallel ADI **[DouHaa]**)

$$\underline{\mathfrak{w}} = \mathfrak{M} \cdot \underline{\mathfrak{u}} := (\mathfrak{M}_L + \mathfrak{M}_D) \cdot \underline{\mathfrak{u}} + \sum_{s=1}^{P} A_s^T \mathfrak{M}_{U,s} R_s^{-1} \cdot \underline{\mathfrak{u}}_s$$

or, for $\mathfrak{M} = \mathfrak{L}^{-1} \cdot \mathfrak{U}^{-1}$ (**[Haase]**)

$$\underline{\mathfrak{w}} = \mathfrak{L}^{-1} \mathfrak{U}^{-1} \cdot \underline{\mathsf{r}} := \mathfrak{L}^{-1} \sum_{s=1}^{P} A_s^T \mathfrak{U}_s^{-1} \cdot \underline{\mathsf{r}}_s$$

Menü

Parallel Iteration Schemes to Solve $\mathsf{K} \cdot \underline{\mathfrak{u}} = \underline{\mathsf{f}}$

- Richardson iteration:

$$\underline{\mathfrak{u}}_s^{k+1} \;:=\; \underline{\mathfrak{u}}_s^{k} + \tau \sum_{q=1}^{P} \left(\underline{\mathsf{f}} - \mathsf{K} \cdot \underline{\mathfrak{u}}^{k}\right)_q$$

- Jacobi iteration with $\mathfrak{D} = \sum\limits_{s=1}^{P} \operatorname{diag}\{\boldsymbol{K_s}\}$:

$$\underline{\mathfrak{u}}_s^{k+1} \;:=\; \underline{\mathfrak{u}}_s^{k} + \omega \mathfrak{D}_s^{-1} \sum_{q=1}^{P} \left(\underline{\mathsf{f}} - \mathsf{K} \cdot \underline{\mathfrak{u}}^{k}\right)_q$$

- Incomplete factorization $\mathfrak{K} = \mathfrak{U} \cdot \mathfrak{L} + \mathfrak{R}$:

$$\underline{\mathfrak{u}}_s^{k+1} \;:=\; \underline{\mathfrak{u}}_s^{k} + \mathfrak{L}_s^{-1} \sum_{q=1}^{P} \mathfrak{U}_q^{-1} \left(\underline{\mathsf{f}} - \mathsf{K} \cdot \underline{\mathfrak{u}}^{k}\right)_q$$

Parallel Multigrid : PMG($\mathsf{K}, \underline{\mathfrak{u}}, \underline{\mathsf{f}}, \ell$)

if $\ell == 1$ **then**

LSSolve $\sum\limits_{s=1}^{P} \boldsymbol{A}_s^T \mathsf{K} \boldsymbol{A}_s \cdot \underline{\mathfrak{u}} \;=\; \underline{\mathsf{f}}$

else

$\underline{\widetilde{\mathfrak{u}}} \leftarrow$ SMOOTH($\mathsf{K}, \underline{\mathfrak{u}}, \underline{\mathsf{f}}, \boldsymbol{\nu}$)

$\underline{\mathsf{d}} \leftarrow \underline{\mathsf{f}} - \mathsf{K} \cdot \underline{\mathfrak{u}}$

$\boxed{\underline{\mathsf{d}}^{\boldsymbol{H}} \leftarrow \mathfrak{P}^{\boldsymbol{T}} \cdot \underline{\mathsf{d}}}$

$\underline{\mathfrak{w}}^{\boldsymbol{H}} \leftarrow \boldsymbol{0}$

PMG$^{\gamma}$($\mathsf{K}^{\boldsymbol{H}}, \underline{\mathfrak{w}}^{\boldsymbol{H}}, \underline{\mathsf{d}}^{\boldsymbol{H}}, \ell - 1$)

$\boxed{\underline{\mathfrak{w}} \leftarrow \mathfrak{P} \cdot \underline{\mathfrak{w}}^{\boldsymbol{H}}}$

$\underline{\widehat{\mathfrak{u}}} \leftarrow \underline{\widetilde{\mathfrak{u}}} + \underline{\mathfrak{w}}$

$\underline{\mathfrak{u}} \leftarrow$ SMOOTH($\mathsf{K}, \underline{\widehat{\mathfrak{u}}}, \underline{\mathsf{f}}, \boldsymbol{\nu}$)

end if

Menü

Application to LU-decomposition based on DD: Factorization

- Again, we have nodes which correspond to inner nodes (index **1**) and coupling nodes (index **2**) and a distributed sparse stiffness matrix $\mathsf{K} = \begin{pmatrix} \mathsf{K}_{11} & \mathsf{K}_{12} \\ \mathsf{K}_{21} & \mathsf{K}_{22} \end{pmatrix}$.
- property for set of subdomains $\sigma(\omega_1) \subset \sigma(\omega_2)$ is locally valid on all processors s.
- LU-decomposition:

$$
\begin{array}{llll}
\text{Get } \ \mathfrak{L}_{ij,s}, \mathfrak{U}_{ij,s} \ \text{ from} & \mathsf{K}_{11,s} & = & \mathfrak{K}_{11,s} = \mathfrak{L}_{11,s}\mathfrak{U}_{11,s};\, \mathsf{K}_{12,s} = \mathfrak{K}_{12,s} = \mathfrak{L}_{11,s}\mathfrak{U}_{12,s};\, \mathsf{K}_{21,s} = \mathfrak{K}_{21,s} = \mathfrak{L}_{21,s}\mathfrak{U}_{11,s}; \\
\text{Update remaining matrix} & \mathsf{K}_{22,s} & := & \mathsf{K}_{22,s} - \boxed{\mathfrak{L}_{21,s} \cdot \mathbf{I}_{22,s} \cdot \mathfrak{U}_{21,s}} \quad \text{with} \quad \mathbf{I}_{22,s} = R_{2,s}^{-1} \\
\text{Accumulate:} & \mathfrak{K}_{22} & := & \sum_{s=1}^{P} A_s \mathsf{K}_{22,s} A_s^T \\
 & \mathfrak{K}_{22,s} & := & A_s^T \mathfrak{K}_{22} A_s \\
\text{Get } \ \mathfrak{L}_{22}, \mathfrak{U}_{22} \ \text{ from} & \mathfrak{K}_{22,s} & = & \mathfrak{L}_{22,s}\mathfrak{U}_{22,s}
\end{array}
$$

- Above algorithm can be applied recursively but the lower right matrix block can be accumulated and decomposed only after the update of the remaining distributed matrix.

Application to LU-decomposition based on DD: Elimination

- Solving the (preconditioning) system: $\begin{pmatrix} \mathfrak{L}_{11} & 0 \\ \mathfrak{L}_{21} & \mathfrak{L}_{22} \end{pmatrix} \begin{pmatrix} \mathfrak{U}_{11} & \mathfrak{U}_{12} \\ 0 & \mathfrak{U}_{22} \end{pmatrix} \underline{\mathfrak{w}} = \underline{\mathsf{r}}$.
- LU-elimination:

$$
\begin{array}{rrcl}
\text{locally} & \underline{\mathsf{u}}_{1,s} & := & \mathfrak{L}_{11,s}^{-1} \underline{\mathsf{r}}_{1,s} \\
 & \underline{\mathsf{u}}_{2,s} & := & \boxed{\mathfrak{L}_{22,s}^{-1}} \left(\underline{\mathsf{r}}_{2,s} - \mathfrak{L}_{21,s} \underline{\mathsf{u}}_{1,s} \right) \\
\text{accumulate} & \underline{\mathfrak{u}} & := & \sum_{s=1}^{P} A_s \underline{\mathsf{u}} \\
\text{locally} & \underline{\mathfrak{w}}_{2,s} & := & \boxed{\mathfrak{U}_{22,s}^{-1}} \underline{\mathfrak{u}}_{2,s} \\
 & \underline{\mathfrak{w}}_{1,s} & := & \mathfrak{U}_{11,s}^{-1} \left(\underline{\mathfrak{u}}_{1,s} - \mathfrak{U}_{12,s} \underline{\mathfrak{w}}_{2,s} \right)
\end{array}
$$

- The matrices $\mathfrak{L}_{22}$ and $\mathfrak{U}_{22}$ have to fulfill the pattern condition if the boundary is assembled from pieces with different sets of subdomains σ. In this case, some entries have to be deleted in the accumulation phase of the matrix block (before the factorization of this block).
- restricted LU-dcomposition, ILU-decomposition, $\mathcal{H}$-LU-decomposition based on DD [Grasedyck]

Idea for Parallelizing AMG as realized in PEBBLES

parallel MG $\Leftrightarrow$ interpolation $\mathfrak{P}$ has to fulfill the pattern condition

$\Downarrow$

Menu

Idea for Parallelizing AMG as realized in PEBBLES

parallel MG $\Leftrightarrow$ interpolation $\mathfrak{P}$ has to fulfill the pattern condition

$\Downarrow$

$\mathsf{K}^{H} = \mathfrak{P}^{T} \cdot \mathsf{K} \cdot \mathfrak{P}$ can be used in PMG$(\mathsf{K}^{H}, \underline{\mathfrak{u}}^{H}, \underline{\mathsf{f}}^{H}, \ell - 1)$.

$\Downarrow$

Idea

Menu

Idea for Parallelizing AMG as realized in PEBBLES

parallel MG $\Leftrightarrow$ interpolation $\mathfrak{P}$ has to fulfill the pattern condition

$\Downarrow$

$\mathsf{K}^{\boldsymbol{H}} \;=\; \mathfrak{P}^{\boldsymbol{T}} \cdot \mathsf{K} \cdot \mathfrak{P}$ can be used in PMG$(\mathsf{K}^{\boldsymbol{H}}, \underline{\mathfrak{u}}^{\boldsymbol{H}}, \underline{\mathsf{f}}^{\boldsymbol{H}}, \boldsymbol{\ell} - \mathbf{1})$.

$\Downarrow$

Idea

Control of coarsening and interpolation such that the pattern condition is fulfilled for $\mathfrak{P}$.

That requires identification and access to $\omega(\sigma)$.

Menu

Library: basic operations

Provide necessary data structures and functions to hide nasty details of communication from the user.
The user should be assisted to reuse as much as possible of his sequential routines.

Library: basic operations

Provide necessary data structures and functions to hide nasty details of communication from the user.
The user should be assisted to reuse as much as possible of his sequential routines.

Goal: User concentrates on numerical algorithms not on communication/data overhead.

Library: basic operations

Provide necessary data structures and functions to hide nasty details of communication from the user.
The user should be assisted to reuse as much as possible of his sequential routines.

Goal: User concentrates on numerical algorithms not on communication/data overhead.

- Methods with communication:
  - setup of communicator(s) from **distributed f.e. mesh** information
  - inner product of vectors,
  - vector accumulation: $\underline{\mathfrak{w}} := \sum_{s=1}^{P} A_s^T \underline{\mathsf{r}}_s$,
  - matrix accumulation (blockwise, update of matrix pattern): $\mathfrak{M} := \sum_{s=1}^{P} A_s^T \mathsf{K}_s A_s$

Library: basic operations (cont.)

- Methods without communication
  - derive a distributed vector from an accumulated vector: $\underline{\mathsf{r}}_s := \mathsf{R}_s^{-1} \cdot \underline{\mathfrak{w}}_s$
  - determine subsets of nodes $\omega(\sigma)$ belonging to the same set of subdomains σ
  - derive new $\omega(\sigma)$ from the old one after refinement/coarsening
  - \# construct a local ordering of the σ-sets (subset property + unique ordering), **globally consistent!**
  - \# local node renumbering according to the ordering of the local σ-sets
  - \# renumbering of incoming and re-renumbering of outgoing data

- Data structures
  - array: local to global numbering
  - arrays: sequence of nodes belonging to σ-sets
  - array of communicators
  - vector for mult. right hand sides, blocks etc. `packed_vector<double> a(nnode, nrhs, nblock)`
    access as linear array $\Longrightarrow$ cache–aware programming
  - special intermediate sparse matrix format `[row, col, entry]`

Code example for setup in main–function

```
//  ----------------  read data from files  ---------------------------------------
string root("../Plank/TBunnyC2/");                  // directory for data
vector<int> hdr;
...
read_header(root, hdr);                             // size of arrays
read_partition(root, hdr, par);                     // partition mapping
read_connection(root, hdr, par, rcon);              // element connectivity
read_element(root, hdr, par, rele);                 // element matrices

// -----------------  setup communicator  ------------------------------------------
element_accumulation(hdr, rcon, row, col, rele);  // determine local nodes,elements
communicator<int, double> com(rcon);                // derive communicator

// -----------------  local matrix accumulation  ----------------------------------
idx_matrix<int, double> A(row, col, rele);          // intermediate matrix format

//  crs_matrix<int, double> D(cnt, col, rele);
GH_crs_matrix D(cnt, col, rele, com);               // derive user specific data format

// ------------------  call numerical algorithm  ----------------------------------
packed_vector<double> _X(_nnodes, _num, 1);
packed_vector<double> _B(_nnodes, _num, 1);

n = GS_iteration_merge(con, D, _X, _B , 1.0e-14, 512, stride);
```

Code example for applying parallel routines in preconditioned CG

```
 template<class T, class S>
 int conjugate_gradient(const matrix<T, S> &_K, const matrix<T, S> &_C,
                        packed_vector<S> &_u, const packed_vector<S> &_f,
const S _eps, const int _max,
                        communicator<T, S> &_com)
 {
   packed_vector<S> _r(_f);
   packed_vector<S> _s(_u);
   packed_vector<S> _v(_r.numnod(), _r.numrhs(), _r.numdof());

   multiply(_K, _s, _v);                          // sequ. matrix-vector
   sub_scale(_r, _v, alpha);                      // sequ. vector-vector
   multiply(_C, _r, _v);                          // parall. precond. [user]
   com.accumulate(_v);                            // parall. vector accu
   scalar_product(_v, _r, sigma);                 // sequ. vector-vector
   _com.collect(sigma);                           // parall. reduce operation
   scale_add(_s, _v, beta);                       // sequ. vector-vector
   ....
 }
```

Summary

- Toolbox works already well and efficient for one–level–methods
- Very fast communicator setup: (kepler:Infiniband, pregl/archimedes: Gigabit)

NP	kepler	pregl	archimedes
1	55.9	70.3	54.7
2	47.6	78.1	66.4
4	35.5	87.9	86.9
8	29.9	98.6	91.3
16	27.4	102.8	95.2
32	34.5	676.6	622.9

Timing for the construction of the communicator object in milliseconds.

Summary

- Toolbox works already well and efficient for one–level–methods
- Very fast communicator setup: (kepler:Infiniband, pregl/archimedes: Gigabit)

NP	kepler	pregl	archimedes
1	55.9	70.3	54.7
2	47.6	78.1	66.4
4	35.5	87.9	86.9
8	29.9	98.6	91.3
16	27.4	102.8	95.2
32	34.5	676.6	622.9

Timing for the construction of the communicator object in milliseconds.

- construction of σ-sets, renumbering/indexing until IV/2006
- multilevel support until II/2007

Summary

- Toolbox works already well and efficient for one–level–methods
- Very fast communicator setup: (kepler:Infiniband, pregl/archimedes: Gigabit)

NP	kepler	pregl	archimedes
1	55.9	70.3	54.7
2	47.6	78.1	66.4
4	35.5	87.9	86.9
8	29.9	98.6	91.3
16	27.4	102.8	95.2
32	34.5	676.6	622.9

Timing for the construction of the communicator object in milliseconds.

- construction of σ-sets, renumbering/indexing until IV/2006
- multilevel support until II/2007

- The works is supported by the **Austrian Grid** project.
- Documentation and code is available via **http://paralleltoolbox.sourceforge.net**

Summary

- Toolbox works already well and efficient for one–level–methods
- Very fast communicator setup: (kepler:Infiniband, pregl/archimedes: Gigabit)

NP	kepler	pregl	archimedes
1	55.9	70.3	54.7
2	47.6	78.1	66.4
4	35.5	87.9	86.9
8	29.9	98.6	91.3
16	27.4	102.8	95.2
32	34.5	676.6	622.9

Timing for the construction of the communicator object in milliseconds.

- construction of σ-sets, renumbering/indexing until IV/2006
- multilevel support until II/2007

- The works is supported by the **Austrian Grid** project.
- Documentation and code is available via **http://paralleltoolbox.sourceforge.net**

Thank You for Your attention!!