Fast solvers for Eulerian convection schemes

Semester Thesis FS 2010

Andreas Hiltebrand
Supervisor: Holger Heumann
Professor: Ralf Hiptmair

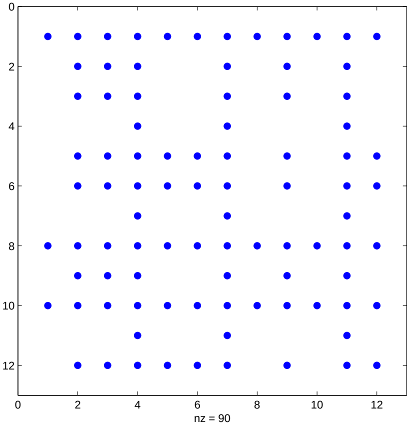
Seminar for Applied Mathematics
ETH Zürich

23.12.10

Goal and discretization

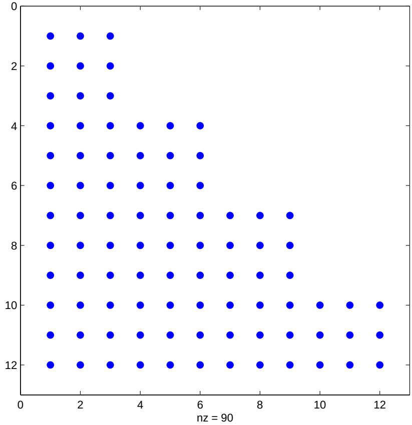
- Goal:
  solve quickly pure advection and advection dominated problems
- Discretization:
  finite elements
  discontinuous Galerkin
  upwind formulation

Permuted block triangular systems



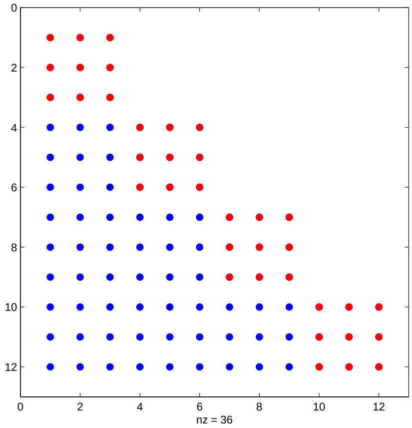
$$Au = b$$

Block triangular systems



$$\left(PAP^{\mathrm{T}}\right) Pu = Pb$$

Block triangular systems



$$\left(PAP^{\mathrm{T}}\right) Pu = Pb$$

Solution of lower block triangular systems

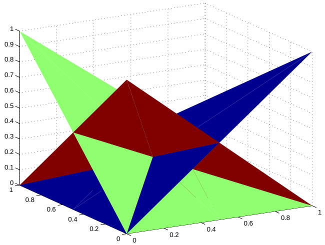
- lower block triangular systems
  $\Longrightarrow$ easily solvable
  by block-wise forward substitution
- for $i = 1, \dots, n_B$
  $\boldsymbol{u}_i^B = \left(\boldsymbol{D}_i^B\right)^{-1}\left(\boldsymbol{b}_i^B - \sum_{j=1}^{i-1} \boldsymbol{L}_{i,j}^B \boldsymbol{u}_j^B\right)$

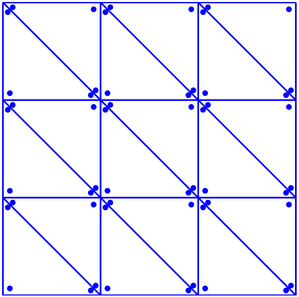
$$\begin{pmatrix} \boldsymbol{D}_1^B & \boldsymbol{0} & \cdots & \cdots & \boldsymbol{0} \\ \boldsymbol{L}_{2,1}^B & \boldsymbol{D}_2^B & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \boldsymbol{D}_{n_B-1}^B & \boldsymbol{0} \\ \boldsymbol{L}_{n_B,1}^B & \cdots & \cdots & \boldsymbol{L}_{n_B,n_B-1}^B & \boldsymbol{D}_{n_B}^B \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1^B \\ \boldsymbol{u}_2^B \\ \vdots \\ \boldsymbol{u}_{n_B-1}^B \\ \boldsymbol{u}_{n_B}^B \end{pmatrix} = \begin{pmatrix} \boldsymbol{b}_1^B \\ \boldsymbol{b}_2^B \\ \vdots \\ \boldsymbol{b}_{n_B-1}^B \\ \boldsymbol{b}_{n_B}^B \end{pmatrix}$$

Advection problems ↔ block triangular systems

- pure advection problem
  - finite elements
  - discontinuous Galerkin
  - upwind formulation

$\Longrightarrow$ permutation of block triangular system

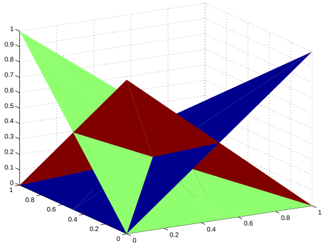


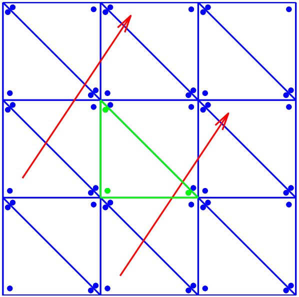


Advection problems $\leftrightarrow$ block triangular systems

- pure advection problem
  - finite elements
  - discontinuous Galerkin
  - upwind formulation

$\Longrightarrow$ permutation of block triangular system





Advection problems $\leftrightarrow$ block triangular systems

- pure advection problem
  - finite elements
  - discontinuous Galerkin
  - upwind formulation

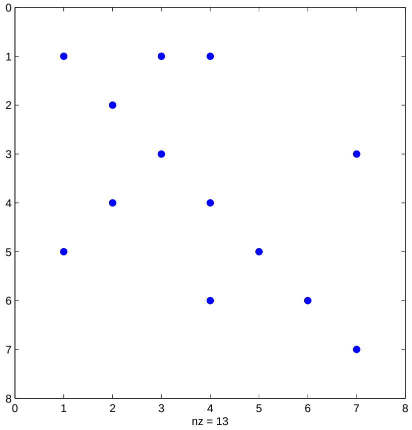
  $\Longrightarrow$ permutation of block triangular system
- advection dominated problem
  $\Longrightarrow$ permutation of almost block triangular system,
  use block Gauss-Seidel method
- construction of permutation?

Outline

1. Introduction
   - Goal and discretization
   - Solution of lower block triangular systems
   - Relationship between advection problems and block triangular systems
2. Construction of permutation
   - Matrix graph
   - Consistent ordering
   - Cycles and strongly connected components
   - Tarjan's algorithm
3. Problems and results
   - Advection-diffusion equation
4. Conclusions

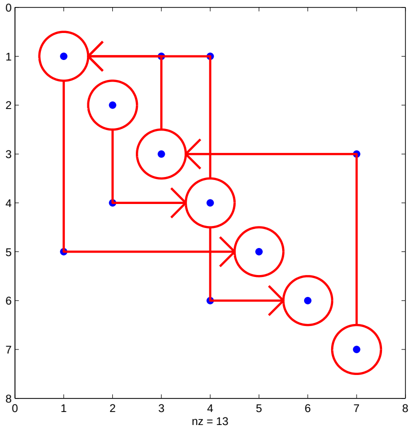
Matrix graph

capturing the dependencies $\Longrightarrow$ matrix graph



Matrix graph

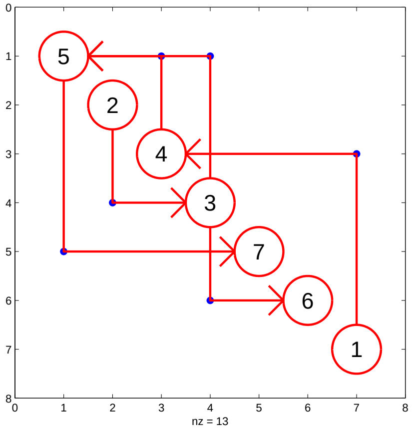
capturing the dependencies $\Longrightarrow$ matrix graph



Consistent ordering

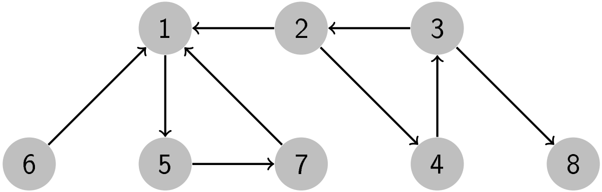
Find an ordering π such that

$$(i,j) \in E \Rightarrow \pi(i) < \pi(j) \quad \forall i,j \in V$$

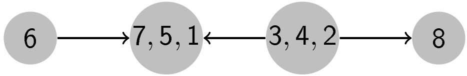


Cycles and strongly connected components

- No cycles $\Longrightarrow$ no problem (Topological sorting)
- Cycles $\Longrightarrow$ no consistent ordering



Condensate strongly connected components



$\Longrightarrow$ consistent ordering possible

Tarjan's algorithm

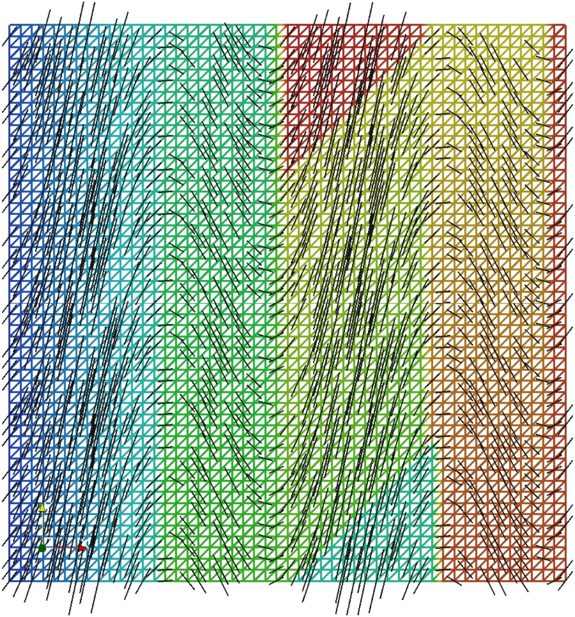
- Determination of strongly connected components: Tarjan's algorithm
- depth first search
- $\Theta(\,|V| + |E|\,)$
- here: $\Theta(n)$

$\Longrightarrow$ construction of ordering: $\Theta(n)$

Steady state advection-diffusion equation in 2D/3D

$$-\epsilon \Delta u + \boldsymbol{b} \cdot \nabla u = f$$

- on the unit square $[0,1]^2$/ unit cube $[0,1]^3$
- Dirichlet boundary conditions
- $\boldsymbol{b}$ velocity field
- f source term
- ϵ diffusivity coefficient
- u unknown scalar function

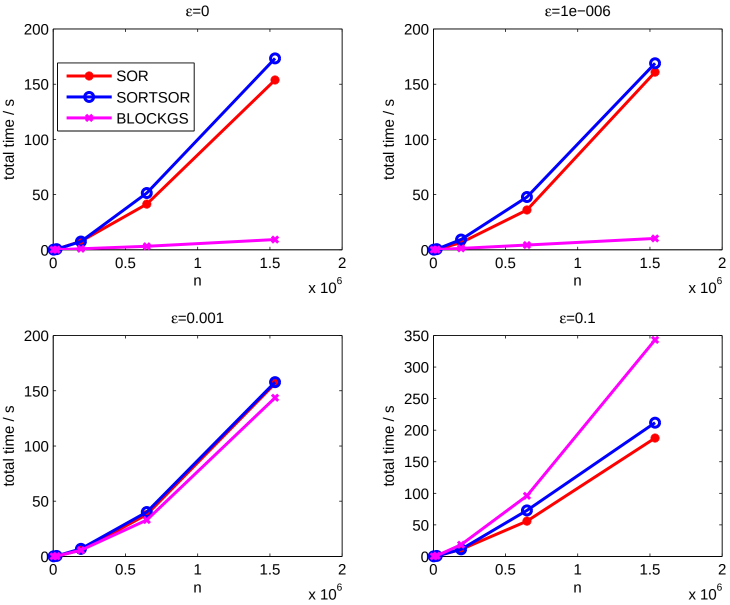


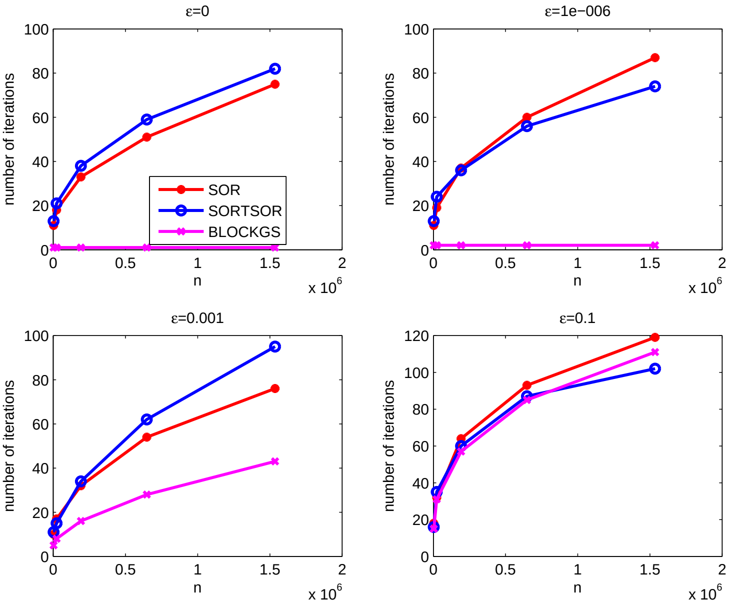
Compared methods

Krylov solver:
Biconjugate gradient stabilized method (BiCGSTAB)

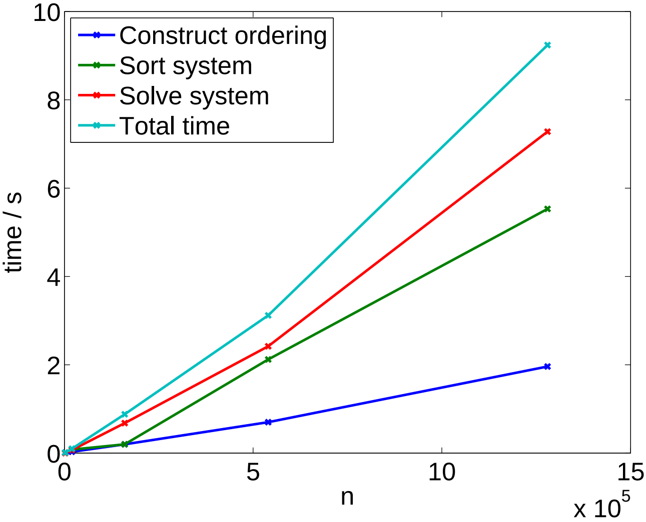
Preconditioner:

- SOR: SSOR
- SORTSOR: sorting the system and then SSOR
- BLOCKGS: implicitly sorting the system and then block Gauss-Seidel method


ε=0
ε=1e−006
ε=0.001
ε=0.1
SOR
SORTSOR
BLOCKGS
total time / s
n
x 10^6
0
0.5
1
1.5
2
0
50
100
150
200
250
300
350


ε=0
ε=1e−006
ε=0.001
ε=0.1
number of iterations
n
x 10^6
SOR
SORTSOR
BLOCKGS
0
20
40
60
80
100
120
0.5
1
1.5
2

Comparison of different parts



Conclusions

- pure advection problems (with this discretization): permuted lower block triangular system
- permutation can be found in $\Theta(n)$ using Topological sorting and Tarjan's Algorithm
- advection dominated problems (with this discretization): permuted almost lower block triangular system
- solve system with block Gauss-Seidel preconditioner: only few iterations
- the more dominating the advection the more efficient

Appendix

Topological sorting

Tarjan's algorithm

Topological sorting

Algorithm 1: Topological sorting

input : graph $G = (V, E)$
output: ordering π
for $v \in V$ **do** $attr(v) = C$
for $v \in V$ **do** SetAttr(v)
for $v \in V$ **do**
    if $attr(v) = C$ **then** $\pi(first) = v$
end

Procedure SetAttr(v)

if $attr(v) = C$ **then** SetF(v);
if $attr(v) = C$ **then** SetL(v);

Topological sorting

Procedure SetF(v)

if $\forall w \in pred(v) : attr(w) = F$ **then**
  $attr(v) = F$;
  $\pi(first) = v$;
  for $w \in succ(v)$ **do if** $attr(w) = C$ **then** SetF(w)
end

Procedure SetL(v)

if $\forall w \in succ(v) : attr(w) = L$ **then**
  $attr(v) = L$;
  $\pi(last) = v$;
  for $w \in pred(v)$ **do if** $attr(w) = C$ **then** SetL(w)
end

Tarjan's algorithm

Algorithm 2: Tarjan's Algorithm

input : graph $G = (V, E)$
output: strongly connected components *components*
$index = 1$
$S = \{\}$
$components = \{\}$
for $v \in V$ **do**
    if *index*(v) *is undefined* **then** tarjan(v)
end

Tarjan's algorithm

Procedure tarjan(v)

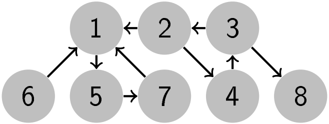
$index(v) = index$
$lowlink(v) = index$
$index = index + 1$
$S.push(v)$
for $(v, v') \in E$ **do**
  if $index(v')$ *is undefined* **then**
    tarjan(v')
    $lowlink(v) = min(lowlink(v), lowlink(v'))$
  end
  else if $v' \in S$ **then**
    $lowlink(v) = min(lowlink(v), index(v'))$
  end
end
...

Tarjan's algorithm

Procedure tarjan(v)

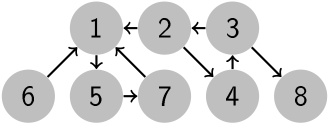
...
if $lowlink(v) = index(v)$ **then**
  $c = \{\}$
  repeat
    $v' = S.pop()$
    $c = c \cup \{v'\}$
  until $v' = v$
  $components = components \cup \{c\}$
end

Tarjan's algorithm



v	$index(v)$	$lowlink(v)$	S	c
1	1	1	$\{1\}$	
5	2	2	$\{1,5\}$	
7	3	3	$\{1,5,7\}$	
7	3	1	$\{1,5,7\}$	
5	2	1	$\{1,5,7\}$	
1	1	1	$\{1,5,7\}$	
			$\{\}$	$\{7,5,1\}$
2	4	4	$\{2\}$	
4	5	5	$\{2,4\}$	
3	6	6	$\{2,4,3\}$	
3	6	4	$\{2,4,3\}$	
8	7	7	$\{2,4,3,8\}$	
			$\{2,4,3\}$	$\{8\}$

Tarjan's algorithm



v	$index(v)$	$lowlink(v)$	S	c
4	5	4	$\{2,4,3\}$	
2	4	4	$\{2,4,3\}$	
			$\{\}$	$\{3,4,2\}$
6	8	8	$\{6\}$	
			$\{\}$	$\{6\}$