Parallel algorithms for Scientific Computations
Iterative methods and preconditioning

Maya Neytcheva

PASC, November 2020

Plan of the lecture:

- The Conjugate Gradient method
- CG in a distributed memory setting
- Recall the GMRES method
- Preconditioning techniques - ILU, approximate inverses, (A)MG
- Examples

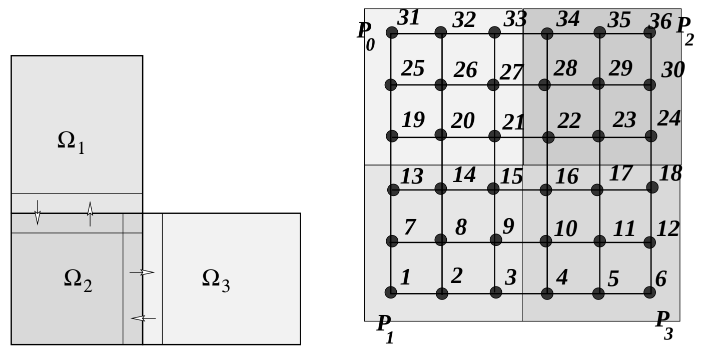
Standard CG

Given A, $\mathbf{b}$ and an initial guess $\mathbf{x}^{(0)}$.

```
        $\mathbf{r}^{(0)} = A\mathbf{x}^{(0)} - \mathbf{b}$,
        $\delta_0 = (\mathbf{r}^{(0)}, \mathbf{r}^{(0)})$
        $\mathbf{d}^{(0)} = -\mathbf{r}$
for $k = 0, 1, \cdots$ until convergence
(1)     $\mathbf{h} = A\mathbf{d}^{(k)}$
(2)     $\tau = \delta_0 / (\mathbf{h}, \mathbf{d}^{(k)})$
(3)     $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \tau\mathbf{d}^{(k)}$
(4)     $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} + \tau\mathbf{h}$,
(5)     $\delta_1 = (\mathbf{r}^{(k+1)}, \mathbf{r}^{(k+1)})$
(6)     $\beta = \delta_1 / \delta_0$,        $\delta_0 = \delta_1$
(7)     $\mathbf{d}^{(k+1)} = -\mathbf{r}^{(k+1)} + \beta\mathbf{d}^{(k)}$
endfor
```

Convergence rate, computational cost per iteration

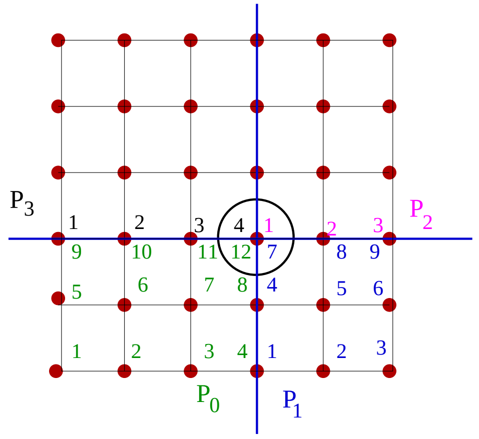
DD-parallel CG, Distributed memory



(a) (b)

Grid-wise ('pointwise non-overlapping') mapping of a discrete problem onto distributed memory computer

DD-parallel CG, Distributed memory



Grid-wise ('pointwise overlapping') mapping of a discrete problem onto distributed memory computer

DD-parallel CG

Assume A and $\mathbf{b}$ are distributed and an initial guess $\mathbf{x}^{(0)}$ is given, which is replicated.

```
        $\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$
        $\mathbf{r} = replicate(\mathbf{r}^{(0)})$
        $\mathbf{d}^{(0)} = -\mathbf{r}$
        $\delta_0 = (\mathbf{r}^{(0)}, \mathbf{r}^{(0)})$
for $k = 0, 1, \cdots$ until convergence
(1)     $\mathbf{h} = A\mathbf{d}^{(k)}$
(2)     $\tau = \delta_0/(\mathbf{h}, \mathbf{d}^{(k)})$
(3)     $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \tau\mathbf{d}^{(k)}$
(4)     $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} + \tau\mathbf{h}$
(5)     $\mathbf{r} = replicate(\mathbf{r}^{(k+1)})$
(6)     $\delta_1 = (\mathbf{r}^{(k+1)}, \mathbf{r})$
(7)     $\beta = \delta_1/\delta_0, \delta_0 = \delta_1$
(8)     $\mathbf{d}^{(k+1)} = \mathbf{r} + \beta\mathbf{d}^{(k)}$
endfor
```

FEM-DD-PCG & 'distributed' (SPAI) preconditioner

Assume A, B and $\mathbf{b}$ are distributed and the initial guess $\mathbf{x}^{(0)}$ is replicated.

```
        $\mathbf{r}^{(0)} = A\mathbf{x}^{(0)} - \mathbf{b}$,    $\mathbf{r}^{(0)} = replicate(\mathbf{r}^{(0)})$
        $\mathbf{h} = B\mathbf{r}^{(0)}$
        $\delta_0 = (\mathbf{r}^{(0)}, \mathbf{h})$    $\mathbf{h} = replicate(\mathbf{h})$
        $\mathbf{d}^{(0)} = -\mathbf{h}$
for $k = 0, 1, \cdots$ until convergence
(1)     $\mathbf{h} = A\mathbf{d}^{(k)}$
(2)     $\tau = \delta_0/(\mathbf{h}, \mathbf{d}^{(k)})$
(3)     $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \tau\mathbf{d}^{(k)}$
(4)     $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} + \tau\mathbf{h}$,    $\mathbf{r}^{(k+1)} = replicate(\mathbf{r}^{(k+1)})$
(5)     $\mathbf{h} = B\mathbf{r}^{(k+1)}$,
(6)     $\delta_1 = (\mathbf{r}^{(k+1)}, \mathbf{h})$    $\mathbf{h} = replicate(\mathbf{h})$
(7)     $\beta = \delta_1/\delta_0$,    $\delta_0 = \delta_1$
(8)     $\mathbf{d}^{(k+1)} = -\mathbf{h} + \beta\mathbf{d}^{(k)}$
endfor
```

Basic GMRES

```
     Compute $\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$, $\beta = \|\mathbf{r}^{(0)}\|_2$ and $\mathbf{v}^{(1)} = \mathbf{r}^{(0)}/\beta$
     for $k = 1, 2, \cdots, m$
(1)      $\mathbf{w}^{(k)} = A\mathbf{v}^{(k)}$
(2)      for $i = 1, 2, \cdots, k$
(3)          $h_{ik} = (\mathbf{w}^{(k)}, \mathbf{v}^{(i)})$
(4)          $\mathbf{w}^{(k)} = \mathbf{w}^{(k)} - h_{ik}\mathbf{v}^{(i)}$
(5)      endfor
(6)      $h_{k+1,k} = \|\mathbf{w}^{(k)}\|_2$; if $h_{k+1,k} = 0$, set $m = k$, goto 11
(7)      $\mathbf{v}^{(k+1)} = \mathbf{w}^{(k)}/h_{k+1,k}$
(8)  endfor
(9)  Define the $(m+1) \times m$ Hessenberg matrix $\overline{H}_m = \{h_{ik}\}$,
     $1 \leq i \leq m+1, 1 \leq k \leq m$
(10) Compute $\mathbf{y}^{(m)}$ as the minimizer of $\|\beta\mathbf{e}_1 - \overline{H}_m\mathbf{y}\|_2$ and $\mathbf{x}^{(m)} = \mathbf{x}^{(0)} + V_m\mathbf{y}^{(m)}$
```

GMRES:

- No breakdown of GMRES
- As m increases, storage and work per iteration increase fast. Remedies:
  - Restart (keep m constant)
  - Truncate the orthogonalization process
- The norm of the residual in the GMRES method is monotonically decreasing. However, the convergence may stagnate. The rate of convergence of GMRES cannot be determined so easy as that of CG.
- The convergence history depends on the initial guess.

GMRES: convergence

Theorem: Let A be diagonalizable, $A = X^{-1}\Lambda X$ where $\Lambda = diag\{\lambda_1, \cdots, \lambda_n\}$ contains the eigenvalues of A. Define

$$\epsilon^m = \min_{p\in\Pi_m^1} \max_{i=1,\cdots n} |p(\lambda_i)|.$$

Then, the residual norm at the mth step of GMRES satisfies

$$\|\mathbf{r}^{(m)}\| \le \kappa(X)\epsilon^m \|\mathbf{r}^{(0)}\|,$$

where $\kappa(X) = \|X\|\,\|X^{-1}\|$.

Preconditioners, suitability for efficient parallelization

- Classical preconditioners
- Approximate inverses, finite-element based ideas
- Multigrid, algebraic multigrid

Classical preconditioners - ILU (IC)

Conventional sparse ILU factorization

```
for i = 2 to n do
    for k = 1 to i-1 and (i,k) in S do
        a_ik = a_ik/a_kk
        for j = k+1 to n and (i,j) in S do
            a_ij = a_ij - a_ik*a_kj
        end
    end
end
```

The preconditioner is in factored form and we have to solve systems with two sparse triangular systems.

Classical preconditioners - ILU (IC)

- The preconditioner and the matrix should have the same data distribution, or?
- Always control the fill-in.
- Show existence of the ILU algorithm.

Not very recent reference:
David Hysom and Alex Pothen
A scalable parallel algorithm for incomplete factor preconditioning, SISC, 22 (2001), 2194-2215

D. Hysom and A. Pothen - the algorithm

Input: A coefficient matrix, its adjacency graph, and the number of processors *p*.

Output: The incomplete factors of the coefficient matrix.

1. Partition the adjacency graph of the matrix into p subgraphs (subdomains), and map each subgraph to a processor. The objectives of the partitioning are that the subgraphs should have roughly equal work, and there should be few edges that join the different subgraphs.
2. On each subgraph, locally order interior nodes first, and then order boundary nodes.
3. Form the subdomain intersection graph corresponding to the partition, and compute an approximate minimum vertex coloring for it. Order subdomains according to color classes.
4. Compute the incomplete factors in parallel.
   - a Factor interior rows of each subdomain.
   - b Receive sparsity patterns and numerical values of the nonzeros of the boundary rows of lower-numbered subdomains adjacent to a subdomain (if any).
   - c Factor boundary rows in each subdomain and send the sparsity patterns and numerical values to higher-numbered neighboring subdomains (if any).

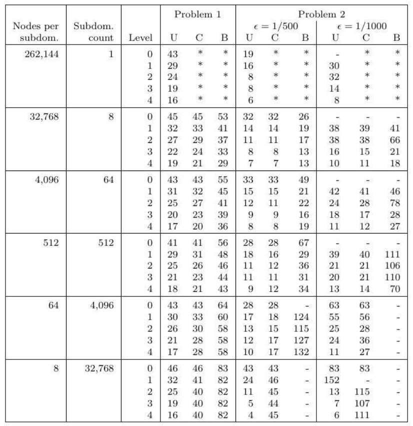
D. Hysom and A. Pothen - results

Tests on diffusion and convection-diffusion problems:
Citation: Page 2209:

Table 4.5 Iteration comparisons for the 646464 grid. U denotes unconstrained, C denotes constrained, and B denotes block Jacobi ILU(k) preconditioners. The starred entries (*) indicate that, since there is a single subdomain, the factor is structurally and numerically identical to the unconstrained PILU(k). Dashed entries (-) indicate the solutions either diverged or failed to converge after 200 iterations. For Problem 2, when $\varepsilon = 1/500$ the level zero preconditioners did not reduce the relative error in the solution by a factor of $10^{-5}5$ at termination; when $\varepsilon = 1/1000$, the level one preconditioners did not do so either.

D. Hysom and A. Pothen - results

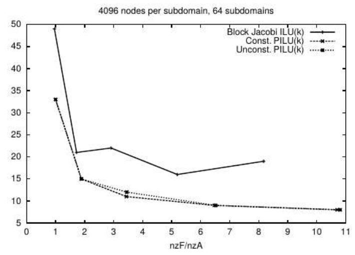
Tests on diffusion and convection-diffusion problems:
Citation: Page 2209:

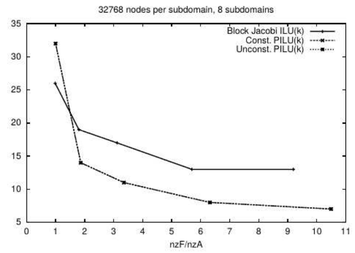


Nodes per subdom.	Subdom. count	Level	Problem 1			Problem 2					
						$\epsilon = 1/500$			$\epsilon = 1/1000$		
			U	C	B	U	C	B	U	C	B
262,144	1	0	43	*	*	19	*	*	-	*	*
		1	29	*	*	16	*	*	30	*	*
		2	24	*	*	8	*	*	32	*	*
		3	19	*	*	8	*	*	14	*	*
		4	16	*	*	6	*	*	8	*	*
32,768	8	0	45	45	53	32	32	26	-	-	-
		1	32	33	41	14	14	19	38	39	41
		2	27	29	37	11	11	17	38	38	66
		3	22	24	33	8	8	13	16	15	21
		4	19	21	29	7	7	13	10	11	18
4,096	64	0	43	43	55	33	33	49	-	-	-
		1	31	32	45	15	15	21	42	41	46
		2	25	27	41	12	11	22	24	28	78
		3	20	23	39	9	9	16	18	17	28
		4	17	20	36	8	8	19	11	12	27
512	512	0	41	41	56	28	28	67	-	-	-
		1	29	31	48	18	16	29	39	40	111
		2	25	26	46	11	12	36	21	21	106
		3	21	23	44	11	11	31	20	21	110
		4	18	21	43	9	12	34	13	14	70
64	4,096	0	43	43	64	28	28	-	63	63	-
		1	30	33	60	17	18	124	55	56	-
		2	26	30	58	13	15	115	25	28	-
		3	21	28	58	12	17	127	24	36	-
		4	17	28	58	10	17	132	11	27	-
8	32,768	0	46	46	83	43	43	-	83	83	-
		1	32	41	82	24	46	-	152	-	-
		2	25	40	82	11	45	-	13	115	-
		3	19	40	82	5	44	-	7	107	-
		4	16	40	82	4	45	-	6	111	-

D. Hysom and A. Pothen - results

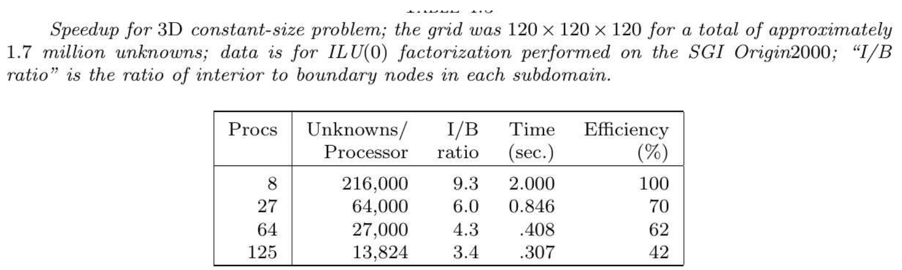
Tests on diffusion and convection-diffusion problems:
Citation: Page 2211:





D. Hysom and A. Pothen - results

Tests on diffusion and convection-diffusion problems:
Citation: Page 2207:



Speedup for 3D constant-size problem; the grid was $120 \times 120 \times 120$ for a total of approximately 1.7 million unknowns; data is for ILU(0) factorization performed on the SGI Origin2000; "I/B ratio" is the ratio of interior to boundary nodes in each subdomain.

Procs	Unknowns/ Processor	I/B ratio	Time (sec.)	Efficiency (%)
8	216,000	9.3	2.000	100
27	64,000	6.0	0.846	70
64	27,000	4.3	.408	62
125	13,824	3.4	.307	42

Classical preconditioners - ILU (IC)

An idea from 2015:
Edmond Chow and Aftab Patel
Fine-grained parallel incomplete LU factorization
SISC, 37 (2015), C169-C193

"All nonzeros in the incomplete factors can be computed in parallel and asynchronously, using one or more sweeps that iteratively improve the accuracy of the factorization. Unlike existing parallel algorithms, the amount of parallelism is large irrespective of the ordering of the matrix, and matrix ordering can be used to enhance the accuracy of the factorization rather than to increase parallelism. Numerical tests show that very few sweeps are needed to construct a factorization that is an effective preconditioner."

Classical preconditioners - ILU (IC)

Reformulation of the algorithm: not-so-far-observed constraint:

$$(LU)_{ij} = a_{ij}$$

We have also

$$l_{ij}, i > j, (i,j) \in S$$
$$u_{ij}, i \leq j, (i,j) \in S$$

The total number of unknowns is equal to the number of elements in the sparsity pattern S and using the constraint we have

$$\sum_{k}^{min(i,j)} = a_{ij}, \quad (i,j) \in S.$$

Classical preconditioners - ILU (IC)

Fine-grained sparse ILU factorization (fixed point iteration)

```
Set unknowns l_ij and u_ij to initial values
for sweep = 1,2,... until convergence do
    parallel for (i,k) in S do
        if i>j then
           l_ij = (a_ij-\sum_{k=1}^{j-1} l_ik*u_kj)
        else
           u_ij = a_ij - \sum_{k=1}^{i-1} l_ik*u_kj
        end
    end
end
```

Parallelism across all elements in S!

Show the article.

Classical preconditioners - ILU (IC)

Current developments:

Thomas Huckle
Parallel approximate LU factorizations for sparse matrices
2019, preprint
Main idea: Newton method instead of fixed point iterations

Sungwoo Kang, Long Cu Ngo, Hyounggwon Choi, Wanjin Chung, Yo-Han Yoo, Jung Yul Yoo
Performance comparison of parallel ILU preconditioners for the incompressible Navier-Stokes equations
Journal of Mechanical Science and Technology 34 (3) 2020
... show the article.

Sparse approximate inverses

Approximate inverses

- Kolotilina, L. Yu.; Yeremin, A. Yu. Factorized sparse approximate inverse preconditionings. I. Theory. SIAM J. Matrix Anal. Appl. 14 (1993), 45-58
- Kolotilina, L. Yu.; Nikishin, A. A.; Yeremin, A. Yu. Factorized sparse approximate inverse preconditionings. IV. Simple approaches to rising efficiency. Numer. Linear Algebra Appl. 6 (1999), 515-531
- Grote, Marcus J.; Huckle, Thomas Parallel preconditioning with sparse approximate inverses. SIAM J. Sci. Comput. 18 (1997), 838-853.
- Benzi, Michele; Meyer, Carl D.; Tuma, Miroslav A sparse approximate inverse preconditioner for the conjugate gradient method. SIAM J. Sci. Comput. 17 (1996), 1135-1149

Approximate inverses

Given a sparse matrix $A = [a_{ij}] \in \mathbf{R}^{n\times n}$.
Let S be a sparsity pattern. We want to compute $G \in \mathcal{S}$, such that

$$(GA)_{ij} = \delta_{ij},\ (i,j) \in \mathcal{S},$$

i.e.

$$\sum_{k:(i,k)\in S} g_{ik}a_{kj} = \delta_{ij},\ (i,j) \in \mathcal{S}.$$

Some observations:

$\oplus$ the elements in the ith row of G can be computed independently;

$\ominus$ even if A is symmetric, G is not necessarily symmetric, because g_{ij} and g_{ji} are, in general, not equal.

Approximate inverses I

Choose $\mathcal{S}$ to be the tridiagonal part of A,
$\mathcal{S} = \{(1,1),(1,2),\{(i,i-1),(i,i),(i,i+1)\}_{i=1}^{n},(n,n-1),(n,n)\}$.
Then, when computing the ith row of G we need only the entries of the matrix A, namely,

$$A^i = \begin{bmatrix} a_{i-1,i-1} & a_{i-1,i} & a_{i-1,i+1} \\ a_{i,i-1} & a_{i,i} & a_{i,i+1} \\ a_{i+1,i-1} & a_{i+1,i} & a_{i+1,i+1} \end{bmatrix}$$

Given $A \in R^{n\times n}$ and $\mathcal{S}$

Approximate inverses II

```
for i=1:n,
    Extract from $A$ the small matrix $A^i$, needed to compute
    the entries of $G(i,:)$
    Solve with $A^i$
    Store row $G(i,:)$
end
```

For all rows, the steps can be performed fully in parallel!

Approximate inverses

Frobenius norm minimization

$$\|A\|_I = \sqrt{\sum_{i=1}^{n}\sum_{i=1}^{n} a_{ij}^2} = \sqrt{tr(AA^H)}$$

Let a sparsity pattern $\mathcal{S}$ be given. Consider the functional

$$F_W(G) = \|I - GA\|_W^2 = tr(I - GA)W(I - GA)^T,$$

where the weight matrix W is spd If $W \equiv I$ then $\|I - GA\|_I$ is the Frobenius norm of $I - GA$.
Clearly $F_W(G) \geq 0$. If $G = A^{-1}$ then $F_W(G) = 0$. Hence, we want to compute the entries of G in order to minimize $F_W(G)$, i.e. to find $\hat{G} \in S$, such that

$$\|I - \hat{G}A\|_W \leq \|I - GA\|_W, \ \forall G \in S.$$

Use the following properties of $tr(\cdot)$:

Approximate inverses

$$\begin{aligned} F_W(G) &= tr(I - GA)W(I - GA)^T \\ &= tr(W - GAW - W(GA)^T + GAW(GA)^T) \qquad (1) \\ &= trW - trGAW - tr(GAW)^T + trGAWA^TG^T. \end{aligned}$$

Minimize F_W w.r.t. G, consider the entries $g_{i,j}$ as variables. The necessary condition for a minimizing point are

$$\frac{\partial F_W(G)}{\partial g_{ij}} = 0, \ (i,j) \in \mathcal{S}. \qquad (2)$$

From (1) and (2) we get $\boxed{-2(WA^T)_{ij} + 2(GAWA^T)_{ij} = 0,}$ or

$$(GAWA^T)_{ij} = (WA^T)_{ij}, \ (i,j) \in \mathcal{S}. \qquad (3)$$

The equations (3) may or may not have a solution, depending on the particular matrix A and the choice of $\mathcal{S}$ and W.

Approximate inverses in parallel

Choice 1: Let A be spd. Choose $W = A^{-1}$ which is also spd

$$\Longrightarrow (GA)_{ij} = \delta_{ij}, \ (i,j) \in S,$$

i.e. the formula for the explicit method can be seen as a special case of the more general framework for computing approximate inverses using weighted Frobenius norms.

Choice 2: Let $W = (A^TA)^{-1}$.

$$\Longrightarrow (G)_{ij} = (A^{-1})_{ij}, \ (i,j) \in S,$$

which is the formula for the implicit method. In this case the entries of G are the corresponding entries of the exact inverse.

Approximate inverses in parallel

1996 Marcus J. Grote, Thomas Huckle, Parallel Preconditioning with Sparse Approximate Inverses (SPAI), SISC

2018 H. Anztab, T.Huckle, J. Bräckle, J. Dongarra, Incomplete Sparse Approximate Inverses for Parallel Preconditioning (ISAI), Parallel Computing

2019 M. Bernaschi, M. Carrozzo, A. Franceschini, C. Janna, A Dynamic Pattern Factored Sparse Approximate Inverse Preconditioner on Graphics Processing Units, SISC (FSAIPACK)

SPAI: Finite element setting

$$A = \sum_{k=1}^{M} R_k^T A_k R_k,$$

with R_k being the Boolean matrices which prescribe the local-to-global correspondence of the numbered degrees of freedom. Is this of interest?

$$B^{-1} = \sum_{k=1}^{M} R_k^T A_k^{-1} R_k.$$

B^{-1} and A^{-1} are spectrally equivalent, namely, for some $0 < \alpha_1 < \alpha_2$ there holds

$$\alpha_1 A_{11}^{-1} \le B_{11}^{-1} \le \alpha_2 A_{11}^{-1},$$

SPAI: Finite element setting

Consider spd matrices.

$$\min_M(\lambda_{min}(A_k)) \le \lambda(A) \le p \max_M(\lambda_{max}(A_k)),$$

where p is the maximum degree of the graph representing the discretization mesh. Similarly, there holds

$$\min_M(\lambda_{min}(A_k)^{-1}) \le \lambda(B^{-1}) \le p \max_M(\lambda_{max}(A_k)^{-1}).$$

Then we obtain

$$\frac{\min(\lambda_{min}(A_k))}{\max(\lambda_{max}(A_k))} \le \frac{\mathbf{x}^T B^{-1}\mathbf{x}}{\mathbf{x}^T A^{-1}\mathbf{x}} \le \frac{\max(\lambda_{max}(A_k))}{\min(\lambda_{min}(A_k))}$$

Thus, the spectral equivalence constants do not depend on the mesh parameter h but they are in general robust neither with respect to problem and mesh-anisotropies, nor to jumps in the problem coefficients as the eigenvalues of A_k depend on those.

Finite element based approximate inverse

FEM-SPAI: Scalability figures: Constant problem size

$\#proc$	n_{fine}	$t_{B_{11}^{-1}}/t_A$	t_{repl} [s]	$t_{solution}$ [s]	$\#$ $iter$
4	197129	0.005	0.28	7.01	5
16	49408	0.180	0.07	0.29	5
64	12416	0.098	0.02	0.03	5

Problem size: 787456
Solution method: *PCG*
Relative stopping criterium: $< 10^{-6}$

FEM-SPAI: Scalability figures: Constant load per processor

$\#proc$	$t_{B_{11}^{-1}}/t_A$	t_{repl} [s]	$t_{solution}$ [s]	$\#$ $iter$
1	0.0050	-	0.17	5
4	0.0032	0.28	7.01	5
16	0.0035	0.24	4.55	5
64	0.0040	0.23	12.43	5

Local number of degrees of freedom: 197129
Solution method: *PCG*
Relative stopping criterium: $< 10^{-6}$

What is usually done in practice

- We rely on libraries.
- Need to discover some structure in the matrix, for instance, a block structure.
- Come up with a block preconditioner and call ready solvers for the blocks.
- If no structure is discovered, then rely on ready preconditioners, but we need to know something on their performance, to choose a good one.

A solver that scales (I-AMLI)

Grid size	Coarsest level No (total no. of levels)	Number of PEs					Time (sec)
		4	8	16	32	64	
256^2	10(16)	403.49 5.31	189.06 5.93	93.86 5.58	49.18 4.62	28.71 3.89	outer comm.
512^2	12(18)			632.60 629.44 14.28	304.24 302.71 12.14	154.65 153.81 10.14	total outer comm.
1024^2	12(20)				1662.73 1655.73 29.89	829.71 826.22 22.26	total outer comm.

Stokes problem: Performance results on the Cray T3E-600 computer

3D Multiphase flow simulations (PRESB)

Size	549250	4293378	33949186	270011394
No. cores	1	8	64	512
		(i1)		
$N_1/N_2/N_3$	3 / 11 / 5	3 / 10 / 6	3 / 9 / 6	3 / 9 / 6
T	35.36	91.02	145.66	275.20
R		2.57	1.60	1.89
		(i2)		
N_1/N_3	20/5	17/5	17/6	14/6
T	31.45	67.53	136.56	140.59
R		2.15	2.02	1.03

Weak scalability test, PCG with AMG preconditioner used for systems with $M+\varepsilon\sqrt{\beta}K$. Average iteration counts N_1, N_2, and N_3, average wall time T in seconds and factor R of increase of T after one refinement of the mesh

Multigrid, software packages

http://homepages.ulb.ac.be/ ynotay/AGMG/

http://mfem.org/parallel-tutorial/

http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview

Some references:

P. R. Amestoy, T. A. Davis and I. S. Duff. An approximate minimum degree ordering algorithm. *SIAM J. Matr. Anal. Appl.*, 17, 886-905, 1996.

C. Ashcraft and J.W.H. Liu. Robust ordering of sparse matrices using multisection. *SIAM J. Matrix Anal. Appl.*, 19, 816-832, 1998.

E. Cuthill, J. McKee. Reducing the bandwidth of sparse symmetric matrices. *Proc. 24th Nat. Conf. Assoc. Comput. Mach.*, 157-172, 1969.

J.W.H. Liu, A. H. Sherman. Comparative analysis of the Cuthill-McKee and the reverse Cuthill-Mckee ordering algorithms for sparse matrices. *SIAM J. Numer. Anal.*, 13, 198-213, 1975.

J. Dongarra, I. Duff, Sorensen and H. van der Vorst, *Numerical Linear Algebra for High Performance Computers*, SIAM Press.

I. Duff, Direct methods, Technical report TR/PA/98/28, July 29, 1998, CERFACS.

H.W. Berry and A. Sameh (1988), Multiprocessor schemes for solving block tridiagonal linear systems, *The International Journal of Supercomputer Applications*, 12, 37-57.

Some references:

I. S. Duff, A. M. Ersiman and J. K. Reid, *Direct Methods for Sparse Matrices*, Oxford University Press, 1986. Reprinted 1989.

I. S. Duff, R. G. Grimes and J. G. Lewis. Sparse matrix test problems. *ACM Trans. Math. Software*, 15, 1-14, 1989.

J.A. George and J.W.H. Liu. *Computer solution of large sparse positive definite systems*. Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

K.A. Gallivan, R.J. Plemmons, and A.H. Sameh (1990), Parallel algorithms for dense linear algebra computations, *SIAM Review*, 32, 54-135.

J. George and J.W.H. Liu. The evolution of the minimum degree ordering algorithm. *SIAM Rev.*, 31, 1-19, 1989.

F.-C. Lin and K.-L. Chung (1990), A cost-optimal parallel tridiagonal system solver, *Parallel Computing*, 15, 189-199.

E. Rothberg and S.C. Eisenstat. Node selection strategies for bottom-up sparse matrix ordering. *SIAM J. Matr. Anal. Appl.*, 19, 682-695, 1998.

H. van der Vorst and K. Dekker, Vectorization of linear recurrence relations, *SIAM Sci. Stat. Comp.*, 10 (1989), 27–35.