Jaca, Sept 2002

Iterative methods for solving large sparse linear systems

Gérard MEURANT

CEA

Solve

$$Ax = b$$

A large sparse non singular matrix of order n, b given, arising from PDE discretization

2 possibilities:

- direct methods (Gauss),

pros: give the "exact" solution (up to round off errors)

cons: fill–in phenomenon,

storage can be too large,

not scalable on parallel computers,

nb of flops $O(n^{\alpha}), 1.75 \leq \alpha \leq 3$

- iterative methods

pros: low storage,

flops count $O(n^{3/2})$ or even $O(n)$

cons: convergence? depends on the matrix properties

Today we want to be able to efficiently solve very large systems on distributed memory parallel computers in $O(n)$ flops

We want to solve with the same efficiency:

- "small" problems on a small number of processors (10),
- "large" problems on a large number of processors (1000)

Elapsed time must be constant when we rise proportionally the problem size and the number of processors (scalability)

This is a tough problem: most known algorithms are not scalable

State of the art for iterative methods: preconditioned Krylov algorithms

Preconditioners: domain decomposition or multilevel

Other contender: algebraic multigrid

Krylov methods

Iterative methods starting from an initial vector x^0

Initial residual $r^0 = b - Ax^0$

Krylov space $\mathcal{K}_k$ of order k based on A and r^0

$$span\{r^0, Ar^0, \ldots, A^{k-1}r^0\}$$

$$x^k \in x^0 + \mathcal{K}_k$$

If V_k is a matrix whose columns are basis vectors $v^j, j = 1, \ldots, k$ of the Krylov space,

$$x^k = x^0 + V_k z^k$$

Two basic kinds of Krylov methods:

1) Orthogonal residual (OR) methods for which

$$(r^k)^T V_k = 0$$

Examples are the Conjugate Gradient (CG) algorithm for symmetric positive definite matrices and FOM for general matrices

2) Minimum residual (MR) methods minimizes the l_2 norm of the residual $r^k = b - Ax^k$

$$(r^k)^T A V_k = 0$$

or solving a least squares problem

Examples of such methods are MINRES for symmetric indefinite matrices and GMRES for general matrices

There are strong relations between OR and MR methods

OR methods may break down for non SPD matrices

Generally for stability reasons one uses an orthogonal basis of the Krylov space: **Arnoldi** process (1951)

Basis vectors are computed recursively by **Gram–Schmidt**

$$v^1 = \frac{r^0}{\|r^0\|}$$

The algorithm for computing column j of V_k is

$$h_{i,j} = (Av^j, v^i), \quad i = 1, \ldots, j$$

$$\bar{v}^j = Av^j - \sum_{i=1}^{j} h_{i,j} v^i,$$

$$h_{j+1,j} = \|\bar{v}^j\|, \quad v^{j+1} = \frac{\bar{v}^j}{h_{j+1,j}}$$

Generally, for stability reasons one prefers using the modified Gram–Schmidt form of the algorithm

$$w^j = Av^j,$$

and for $i = 1, \ldots, j$

$$h_{i,j} = (w^j, v^i), \quad w^j = w^j - h_{i,j} v^i$$

Both algorithms are the same in exact arithmetic, but MGS is more stable

The matrix relation for V_k is the following:

$$AV_k = V_k H_k + h_{k+1,k} v^{k+1} (e^k)^T,$$

where H_k is an upper Hessenberg matrix with elements $h_{i,j}$ and e^k is the kth unit vector. This is written as

$$AV_k = V_{k+1} \tilde{H}_k,$$

with

$$\tilde{H}_k = \begin{pmatrix} H_k \\ h_{k+1,k} (e^k)^T \end{pmatrix}$$

We also have

$$V_k^T A V_k = H_k$$

In the symmetric case the matrix H_k is tridiagonal and we denote it by T_k

This implies that the basis vectors can be computed by a three–term recurrence (**Lanczos** algorithm 1952)

GMRES (Generalized Minimum RESidual, Saad and Schulz 1986) minimizes the l_2 norm of the residual vector $b - Ax^k$

$$\| \|r^0\|e^1 - \tilde{H}_k z^k\|$$

This is solved by incrementally computing a QR decomposition of $\tilde{H}_k$ using Givens rotations

In FOM we have to solve linear systems

$$H_k z^k = \|r^0\|e^1$$

This is also done using QR factorizations

Drawback: the storage grows with the iteration number since we have to store the basis vectors v^j

Solution: restart every m iterations $\rightarrow$ GMRES(m)

These methods are used with a preconditioner M solving

$$M^{-1}Ax = M^{-1}b$$

When A is SPD, H_k is tridiagonal and SPD so we can use a Cholesky factorization

By using a clever change of variables, this leads to the **Conjugate Gradient (CG)** (Hestenes and Stiefel 1952) method which is the most popular and efficient algorithm for solving large sparse SPD systems

Let x^0 be given, $r^0 = b - Ax^0$. For $k = 0, 1, \ldots$

$$Mz^k = r^k,$$
$$\beta_k = \frac{(z^k, Mz^k)}{(z^{k-1}, Mz^{k-1})}, \quad \beta_0 = 0,$$
$$p^k = z^k + \beta_k p^{k-1},$$
$$\gamma_k = \frac{(z^k, Mz^k)}{(p^k, Ap^k)},$$
$$x^{k+1} = x^k + \gamma_k p^k,$$
$$r^{k+1} = r^k - \gamma_k A p^k.$$

M is the preconditioner and must be an SPD matrix

CG convergence depends on the distribution of the eigenvalues

Others methods for non symmetric matrices

BiCG, BiCGSTAB (Van der Vorst)

QMR (Freund)

CMRH (Sadok)

Classical preconditioners

The most popular preconditioner for SPD matrices is the **Incomplete Cholesky (IC)** decomposition that can be generalized to Incomplete LU (ILU) for general sparse matrices, Meijerink and Van der Vorst 1977

This preconditioner mimics what is done in the ordinary Cholesky decomposition of the matrix except that some entries are thrown away during the process based either on their position or their absolute value

The matrix M has the form $M = LD^{-1}L^T$ where L is lower triangular and D diagonal

Problem: the computation and use of M are not parallel (triangular systems)

Other solution: **approximate inverses**

Directly compute $C = M^{-1}$, then $z^k = Cr^k$

There are two basic kinds of methods

1) Compute C to minimize $\|AC - I\|_F$

SPAI algorithm by Huckle and Grote 1994

Pb for SPD matrices

2) Compute $C = ZD^{-1}Z^T$ approximate inverse of A which involves a parameter τ used to define which elements are dropped during the factorization

AINV algorithm by Benzi and Tuma 1996 (approximate A orthogonalization)

$Z = I$

$d_1 = a_{1,1}$

for $i = 2, \ldots, n$

  drop the entries $z_{k,i}$ such that $|z_{k,i}| \leq \tau \|a_i\|_\infty$

    for $j = i, \ldots, n$

      $d_j = a_i^T z_j$

      $z_j = z_j - \frac{d_j}{d_i} z_i$

    end

end

PCG for Poisson problem $m \times m$ mesh, $\Omega =]0,1[^2$

$\varepsilon = 10^{-10}$

m	IC(0)	IC($\epsilon = 0.005$)	IC($\epsilon = 0.001$)
10	16 op=66009 str=100	9 op=44173 str=525	7 op=42573 str=758
20	27 op=446957 str=400	15 op=296841 str=2245	10 op=257393 str=3488
30	38 op=1410785 str=900	21 op=934509 str=5165	13 op=762053 str=8218
40	49 op=3230793 str=1600	26 op=2056749 str=9285	16 op=1673553 str=14948
50	60 op=6176681 str=2500	31 op=3829389 str=14605	19 op=3108893 str=23678
60	71 op=10519049 str=3600	38 op=6747485 str=21125	22 op=5185073 str=34408

PCG for an anisotropic problem

m	IC(0)	IC($\epsilon = 0.005$)	IC($\epsilon = 0.001$)
10	9 op=38373 str=100	6 op=28041 str=434	4 op=20253 str=461
20	14 op=236913 str=400	7 op=134025 str=1864	6 op=119829 str=1975
30	20 op=755081 str=900	9 op=384533 str=4294	8 op=370325 str=4995
40	26 op=1736529 str=1600	10 op=758817 str=7724	8 op=679245 str=9435
50	31 op=3227149 str=2500	12 op=1411905 str=12154	9 op=1204573 str=15275
60	37 op=5533017 str=3600	14 op=2358353 str=17584	10 op=1935861 str=22515

PCG for a discontinuous problem

m	IC(0)	IC($\epsilon = 0.005$)	IC($\epsilon = 0.001$)
10	16 op=66009 str=100	7 op=41229 str=716	5 op=35013 str=898
20	29 op=478233 str=400	10 op=247713 str=3268	7 op=228521 str=4817
30	39 op=1447213 str=900	13 op=747101 str=7951	9 op=710413 str=20541
39	49 op=3070512 str=1521	16 op=1565299 str=13835	10 op=1382077 str=22361
50	61 op=6278389 str=2500	19 op=3072973 str=23229	12 op=2777061 str=38407
59	73 op=10453792 str=3481	22 op=4962385 str=32715	13 op=4244652 str=54841

Pb : Poisson, L-shaped region, mixed b.c.

Comparison between IC and AINV (Benzi)

IC			AINV		
fill	nb. iter	time	fill	nb. iter	time
675	87	0.33	743	76	0.32
897	53	0.18	780	74	0.32
912	51	0.18	1135	54	0.26
1204	38	0.14	1208	47	0.18
1439	32	0.14	1300	40	0.21
1565	24	0.10	3654	22	0.14

Comparison between SPAI and AINV (Benzi)

Matrix	SPAI			AINV		
	Its	init	t its	Its	init	t its
3DCD	40	10.63	0.111	25	1.885	0.068
ALE3D	45	30.79	0.088	43	1.446	0.094
ORSREG1	40	3.309	0.033	33	0.550	0.031
SHERMAN1	62	0.878	0.029	43	0.201	0.021
PORES3	111	0.941	0.044	75	0.127	0.038
WATT2	377	2.590	0.384	111	0.505	0.116

1138-bus

$x^0 = 0$, b random, $\varepsilon = 10^{-10}$

prec	nb it	flops	str
diag	1120	$2.57\ 10^7$	1138
ic	163	$5.80\ 10^6$	2596+
ssor	553	$2.18\ 10^7$	0
lev 1	163	$5.80\ 10^6$	2596+
lev 2	77	$3.13\ 10^6$	3877+
lev 3	54	$2.45\ 10^6$	5025+
lev 4	40	$1.99\ 10^6$	6168+
ch 0.1	98	$3.57\ 10^6$	2807+
ch 0.05	79	$3.05\ 10^6$	3347+
ch 0.01	43	$1.96\ 10^6$	5104+
ch 0.005	36	$1.78\ 10^6$	6085+
ai 0.1	111	$5.09\ 10^6$	5725+
ai 0.05	85	$5.19\ 10^6$	9525+
ai 0.01	46	$6.92\ 10^6$	31874+
pol ls 1	818	$3.95\ 10^7$	0
pol ls 2	580	$4.27\ 10^7$	0

Anisotropic problem, $m = 40$

$x^0 = 0$, b random, $\varepsilon = 10^{-10}$

prec	nb it	flops	str
diag	288	$1.05\ 10^7$	1600
ic	26	$1.52\ 10^6$	4720+
ssor	97	$6.15\ 10^6$	0
lev 1	26	$1.52\ 10^6$	4720+
lev 2	10	$0.65\ 10^6$	6241+
lev 3	10	$0.71\ 10^6$	7723+
lev 4	10	$0.74\ 10^6$	8501+
ch 0.1	30	$1.57\ 10^6$	3160+
ch 0.05	30	$1.57\ 10^6$	3160+
ch 0.01	30	$1.57\ 10^6$	3160+
ch 0.005	10	$0.64\ 10^6$	6124+
ai 0.1	31	$3.68\ 10^6$	20520+
ai 0.05	30	$4.05\ 10^6$	24460+
ai 0.01	30	$4.76\ 10^6$	30560+
tw	161	$8.15\ 10^6$	7840+
pol ls 1	162	$1.31\ 10^7$	0
pol ls 2	113	$1.41\ 10^7$	0

Discontinuous problem, $m = 40$

$x^0 = 0$, b random, $\varepsilon = 10^{-10}$

prec	nb it	flops	str
diag	168	$6.13\ 10^6$	1600
ic	54	$3.16\ 10^6$	4720+
ssor	62	$3.93\ 10^6$	0
lev 1	54	$3.16\ 10^6$	4720+
lev 2	33	$2.13\ 10^6$	6241+
lev 3	27	$1.91\ 10^6$	7723+
lev 4	29	$2.14\ 10^6$	8501+
ch 0.1	41	$2.49\ 10^6$	5227+
ch 0.05	31	$2.00\ 10^6$	6196+
ch 0.01	19	$1.52\ 10^6$	10097+
ch 0.005	16	$1.46\ 10^6$	12878+
ai 0.1	51	$4.39\ 10^6$	12430+
ai 0.05	36	$5.21\ 10^6$	27026+
ai 0.01	18	$9.50\ 10^6$	122907+
tw	90	$4.55\ 10^6$	7840+
pol ls 1	96	$7.75\ 10^6$	0
pol ls 2	67	$8.37\ 10^6$	0

Multilevel preconditioners

◦ Algebraic methods (grid $\equiv$ (sub) set of unknowns)

◦ Algebraic MultiGrid (AMG)–like (V–cycle):

Starting from the zero vector:

0– if we are on the coarsest level, solve exactly by Gaussian elimination, otherwise

1– do ν iterations of smoothing

2– restrict the residual r to $r_c = Rr$

3– recursively solve $A_c e_c = r_c$, $A_c = RAP$, $R = P^T$

4– interpolate e_c to $e = Pe_c$

5– add the correction e to the current iterate

6– do ν iterations of smoothing

We have to define

◦ the smoother

◦ how to construct the coarse grids (coarsening alg.)

◦ the interpolation

There are many choices!

Smoothers

◦ Symmetric Gauss–Seidel (not naturally //)

◦ Incomplete Cholesky (not // either) LDL^T

- IC(0)
- IC with some fill–in on values
- IC with some fill–in on levels of fill

$$LD^{-1}L^T(x^{k+1} - x^k) = b - Ax^k$$

◦ Approximate inverse AINV from Benzi

$$M = ZD^{-1}Z^T$$

where the matrix Z is upper triangular with $1's$ on the diagonal and D is diagonal

Smoother: Richardson iteration defined as (matrix $\times$)

$$x^{k+1} = x^k + M(b - Ax^k)$$

For a two–grid algorithm using one step of pre–smoothing and one step of post–smoothing starting from $x^0 = 0$, the preconditioner $\tilde{M}_1$ is defined as

$$\tilde{M}_1 = M + M(I - AM) + (I - MA)(P(RAP)^{-1}R)(I - AM)$$

Theorem

$\tilde{M}_1$ is symmetric

We can show that $\tilde{M}_1$ is positive definite if we suppose that M is such that $\rho(I - AM) < 1$ (sufficient condition)

Moreover 1 is a multiple eigenvalue of $\tilde{M}_1 A$ and all the eigenvalues of $\tilde{M}_1 A$ are smaller or equal to 1

This occurs whatever the choice of M, R and P

The same results apply if the coarse matrix is obtained by using the same algorithm recursively, that is in the multilevel case

The influence matrix

$$\mathcal{N} = \{1, \ldots, n\}$$

$$\mathcal{N} = F \cup C$$

Influence matrix: standard AMG choice for M–matrices

$$S_i = \{j | -a_{i,j} > \theta \max_{k \neq i}(-a_{i,k}), \quad \theta < 1\}$$

This gives S (by padding with zeros)

Generalization:

$$S_i^A = \{j \mid |a_{i,j}| > \tau \max_{k \neq i} |a_{i,k}|, \quad \tau < 1\}$$

The coarsening algorithm

The "standard" algorithm is:

Weights $w_i =$ the number of points that depend on i (using S)

1- choose the first point i with maximal weight as a C point

2- assign the points that i influences (using S) as F points

3- increment by 1 the weights of the points influencing these new F points (to give more chances to be selected as C points)

4- decrease by 1 the weights of points that depends on i

repeat from step 1- until all points are labeled

◦ There are many other choices for the coarsening algorithm

◦ Remark that if we change the interpolation algorithm the number and location of coarse nodes change (on coarse levels)

The interpolation algorithm

◦ AMG uses $Ae = 0$, $i \in F, j \in C$

$$\omega_{i,j} = -\frac{a_{i,j} + \sum_{k \in D_i^S} \frac{a_{i,k} a_{k,j}}{\sum_{m \in C_i} a_{k,m}}}{a_{i,i} + \sum_{k \in D_i^W} a_{i,k}}$$

This uses $e_j \approx e_i$ for weak connections and a weighted average for F connections

Note that the given F point needs to have at least one coarse point in its neighborhood in the graph of A

The coarse matrices

The interpolation algorithm defines P and $R = P^T$

$$A_C = RAP$$

Problems

There are still sequential parts in this algorithm:

- some smoothers (IC and computation of AINV)

- the coarsening algorithm

We are now working on parallelizing this when using a domain partitioning of the graph of the matrix

There are still problems to solve on parallel computers:

- load balancing on coarse levels?

- how many levels to use?

- enough work to do on coarse levels?