

SVD, Jacobi Iteration, SLE Iterative Solver (GMRES), Pre-conditioned Matrix

(1.) For a given matrix $[A]$ where $[A]$ could be square/rectangular, symmetrical/unsymmetrical, sparse/dense, etc..., we can use SVD to factorize $[A]$ into:

$$[A] = [U] [S] [V]^T$$

orthonormal matrices \odot ----- (1)

$$[A] = [U][\Sigma][V^T]$$

orthonormal matrices \odot

- Diagonal matrix (✓)

(2.) How to efficiently solve $[A]_{N \times N} \vec{x}_{N \times 1} = \vec{b}_{N \times 1}$?? (2)
Non-singular, Non-symmetrical matrix

In principle, we can use SVD to transform Eq. (2) into "diagonal" matrix,

and can easily solve after that. However, SVD process is expensive!

and can easily solve after that. However, SVD process is expensive.
Furthermore, we only need to transform $[A]$ into "diagonal dominant" matrix
(such as $|a_{ii}| \geq \sum_{j=1, j \neq i}^N |a_{ij}|$) which then can be easier to solve by any
much by JACOBI iteration, Gauss-Seidel, etc...

furthermore, we only need to minimize $\sum_{j=1}^N |a_{ij}|$ (such as $|a_{ii}| \geq \sum_{j=1, j \neq i}^N |a_{ij}|$) which then can be easier to solve by any "non-symmetrical iterative" solver, such as GMRES, Bi-CGSTAB, etc...

In other words, we do NOT need to go "all the way" to diagonal matrix, we can stop the process prematurely whenever the new/transformed matrix $[A^*]$ becomes a diagonal dominant matrix!

(3) Derivations of Basic Algorithms (to solve unsym. linear equations) by GMRES iterative method

$$[A] \vec{x} = \vec{b}$$

$$[P_1^T] [A] \vec{x} = [P_1^T] \vec{b}$$

$$\underbrace{P_i^T A [Q_i Q_i^T]}_{[A_i]} \underbrace{\vec{x}}_{\vec{y}_i} = \underbrace{P_i^T \vec{b}}_{\vec{b}_i}$$

$$\underbrace{P_2^T [A_1] [Q_2 Q_2^T]}_{\text{matrix}} \underbrace{\vec{y}_1}_{\text{vector}} = \underbrace{P_2^T}_{\text{matrix}} \underbrace{\vec{b}_1}_{\text{vector}}$$

$$[A_2] * \vec{y}_2 = \vec{b}_2$$

Thus: $[A_2] = P_2^T [P_1^T A Q_1] Q_2$

Assuming $[A_2] = \text{"diag. dominant" matrix, then:}$
First: Solve for \vec{y}_2

First: Solve for y_2

$$\vec{T} |_{T_1} \vec{T}_1 \rightarrow \vec{u}_2 \equiv Q_2^T \{ Q_1^T \vec{x} \} \quad \text{--- (8)}$$

Hence: $\vec{x} = [Q_1][Q_2]\vec{y}_2$ Second: Solve for \vec{x} (9)

(4) Detailed Step-by-Step Computer Program Implementation

Step 1: Get (or generate) input data for $[A]_{N \times N}$ & \vec{b}
(unsym.)
from the internet (or by yourself).

Step 2: Find the largest (absolute) value of off-diagonal term
amongst those "un-diagonal dominant" rows
, and its location (n, q)

GlobalLargest = 0.00

Do 1 irow = 1, N

AbsSumPerRow = 0.00

PerRowLargest = 0.00
 $A_{ii} = \text{ABS}[A(irow, irow)]$

Do 2 jcol = 1, N

If (jcol .NE. irow) then

$A_{ij} = \text{ABS}[A(irow, jcol)]$

AbsSumPerRow = AbsSumPerRow + A_{ij}

Endif

If (A_{ij} .GT. PerRowLargest) then
PerRowLargest = A_{ij}
ip = irow
jq = jcol
Endif

2 Continue

check to see if $|A_{ii}| \geq \sum_{j \neq i} |A_{ij}|$

If (A_{ii} .GE. AbsSumPerRow) then

Go To 1

Else

If [PerRowLargest .GT. GlobalLargest] Then

GlobalLargest = PerRowLargest

irowp = ip

jcolq = jq

Endif

Endif

1 Continue

step 3: Compute and store the 2 angles $\hat{\varphi}_1$ and $\hat{\varphi}_2$,
by referring to the 2 equations:

$$A(p, q) = 0$$

$$A^k(q, p) = 0$$

Thus, we can (later on) generate matrices $[P_1^T]$ & $[Q_1]$
 $[P_2^T]$ & $[Q_2]$
etc...

Notes:

• Assuming that $[A_2]$ = diag. dominant matrix

• Then: $[A_2] \vec{y}_2 = \vec{b}_2$

or $[A^*] \vec{y} = \vec{b}^*$

(7, repeated)

(10)

where:

$$[A^*] = P_2^T [P_1^T A Q_1] Q_2 \quad (11)$$

No neglecting small off-diagonal terms
in rows # p & q and in columns p & q
during computation of $[A_1]$, $[A_2]$ etc...

• If we continue the Jacobi process (to zero out the largest value
at locations "p, q" & "q, p" in each iteration) until the end
(when $[A^*]$ = diagonal matrix), then all the eigen-values of

$[A]$ will appear on the diagonal matrix $[A^*]$. Thus, the
condition # of $[A]$ and $[A^*]$ should be the same, as

verified by numerous results, presented in Tables 1-2. However, solving
the SLE with $[A^*]$ & diag. dominant matrix should be much easier as compared to $[A]$.

• In real computer implementation with iterative solver (such as GMRES), one
needs to compute the coefficient matrix $[A^*]$ & the known vector \vec{d} ,

which should be done as:

$$[A^*] \vec{d} = P_2^T P_1^T A Q_1 Q_2 \{ \vec{d} \}$$

1st
2nd
3rd
4th
5th

- Pre-conditioned should be used in Eq(10), in conjunction with GMRES, such as

$$[M]^{-1} [A^*] \vec{y} = [M]^{-1} \{b^*\} \quad (12)$$

where $[M]$ should be \nearrow close to $[A^*]$
 \searrow cheaper to inverse (or factorize)

In MATLAB/GMRES, the user is required to provide $[A^*]$, $\{b^*\}$ and $[M]$. Inside GMRES procedure, $[M]^{-1} * \vec{d}_{\text{known}}$ operation needs be done repeatedly!

- How to compute the pre-conditioned matrix $[M]$?

Since MATLAB requires the user to provide $[M]$ (not M^{-1} !),

GMRES- the matrix $[M]$ can be explicitly computed as

$$[M] \approx [A^*] \quad (13)$$

$$[M] \approx \dots P_2^T A_1 Q_2 \dots \quad (14)$$

$$\text{and } A_1 \approx P_1^T A Q_1 \quad (15)$$

In computing Eq. (15) for $[A_1]$, or computing Eq. (14) for $[M]$, we realize that only 2 rows & 2 columns associated with the maximum (absolute) value locations (p, q) & (q, p) needs be updated. To preserve more even sparsity (than the previous iteration), any non-zero off-diagonal value (on rows p, q and on columns p, q) which is small relative to its diagonal value (on its same row) will be neglected

$$\left[\frac{(A_{ij}^{(k)})^2}{A_{ii}^{(k)} * A_{jj}^{(k)}} \right]^{1/2} \leq 10^{-5} \quad [\text{see K.J. Bathe's book, pp. 914}] \quad (16)$$

The same procedure is applied when computing the triple products, shown in Eq. (14)

$$P_2^T A, Q_2$$

Thus, matrix $[M]$ will ^{also} generally have more sparsity

not only is a good approximation of $[A^*]$, but it

than the original matrix $[A]$, as can be confirmed by Tables 1-2. Finally, the pre-conditioned matrix $[P]$, shown in Tables 1-2, can be defined

as:

$$[P] \equiv [M]^{-1} * [A^*] \quad (17)$$

Step 4 Compute/Update matrix $[M]$ as explained in Eqs. (13-16)

Step 5 Check to see if we already achieve/converge the "diagonal dominant" matrix ??

- If no, then return/goback to step 2
- If yes, then Go to step 6

Step 6 Solve Eq. (7) to get the intermediate vector \vec{y} from $[A^*] \vec{y} = \vec{b}^*$ (18)

Step 7 Solve Eq. (9) to recover the ^{original} unknown vector \vec{x}

Step 8

Print # non-zeros in $[A]$, in $[M]$. Print condition # of $[A]$, $[A^*]$, $[M]$, $[P] = [M]^{-1} * [A^*]$
 Plot $[A]$, $[M]$ by using MATLAB command `> SPY(A)`
 Use MATLAB/GMRES, record # iterations to solve $A\vec{x} = \vec{b}$; and $[A^*] \vec{y} = \vec{b}^*$

JACOBI iteration or SVD

$$[U_k^T] = [U_k^{-1}] \quad [V_k^T] = [V_k^{-1}] \quad \text{For Example:}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_2 & -s_2 & 0 \\ 0 & s_2 & c_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_2 & s_2 & 0 \\ 0 & -s_2 & c_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = [I]$$

where U_k and V_k are orthonormal matrices that both equal to the identity matrix except that four entries at the intersection of rows and columns p and q . These four entries are triangular function of rotation angles $\varphi^{(k)}$ and $\psi^{(k)}$ which can be evaluated from two equations: $a_{pq}^{(k)} = 0$ and $a_{qp}^{(k)} = 0$. When

A is symmetric matrix, U_k and V_k are in the same.

assuming this location has the largest absolute value (say, @ row p & col q)

$\hat{\varphi} \equiv \hat{\psi}$ (hence, only 1 unknown)
If $[A]$ is unsymmetrical
Then $\hat{\varphi} \neq \hat{\psi}$ (hence, 2 unknown angles)

$$A^{(k)} = U_k^T A^{(k-1)} V_k = \begin{bmatrix} 1 & & & \\ & c_1 & & \\ & -s_1 & & \\ & & & 1 \end{bmatrix} \begin{bmatrix} a_{11}^{(k-1)} & & & \\ & a_{pp}^{(k-1)} & & \\ & & a_{qq}^{(k-1)} & \\ & & & a_{nn}^{(k-1)} \end{bmatrix} \begin{bmatrix} 1 & & & \\ & c_2 & & \\ & s_2 & & \\ & & & 1 \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}^{(k-1)} & & & \\ c_1 a_{p1}^{(k-1)} + s_1 a_{q1}^{(k-1)} & & & \\ -s_1 a_{p1}^{(k-1)} + c_1 a_{q1}^{(k-1)} & & & \\ a_{n1}^{(k-1)} & & & \end{bmatrix} \begin{bmatrix} a_{1p}^{(k-1)} & & & \\ c_1 a_{pp}^{(k-1)} + s_1 a_{qp}^{(k-1)} & & & \\ -s_1 a_{pp}^{(k-1)} + c_1 a_{qp}^{(k-1)} & & & \\ a_{np}^{(k-1)} & & & \end{bmatrix} \begin{bmatrix} a_{1q}^{(k-1)} & & & \\ c_1 a_{pq}^{(k-1)} + s_1 a_{qq}^{(k-1)} & & & \\ -s_1 a_{pq}^{(k-1)} + c_1 a_{qq}^{(k-1)} & & & \\ a_{nq}^{(k-1)} & & & \end{bmatrix} \begin{bmatrix} a_{1n}^{(k-1)} & & & \\ c_1 a_{pn}^{(k-1)} + s_1 a_{qn}^{(k-1)} & & & \\ -s_1 a_{pn}^{(k-1)} + c_1 a_{qn}^{(k-1)} & & & \\ a_{nn}^{(k-1)} & & & \end{bmatrix} \begin{bmatrix} 1 & & & \\ & c_2 & & \\ & s_2 & & \\ & & & 1 \end{bmatrix}$$

$$[A^{(k)}] = \begin{bmatrix} a_{11}^{(k-1)} & & & \\ c_1 a_{p1}^{(k-1)} + s_1 a_{q1}^{(k-1)} & & & \\ -s_1 a_{p1}^{(k-1)} + c_1 a_{q1}^{(k-1)} & & & \\ a_{n1}^{(k-1)} & & & \end{bmatrix} \begin{bmatrix} a_{1p}^{(k-1)} & & & \\ c_2 a_{pp}^{(k-1)} + s_2 a_{qp}^{(k-1)} & & & \\ -s_2 a_{pp}^{(k-1)} + c_2 a_{qp}^{(k-1)} & & & \\ a_{np}^{(k-1)} & & & \end{bmatrix} \begin{bmatrix} a_{1q}^{(k-1)} & & & \\ c_2 a_{pq}^{(k-1)} + s_2 a_{qq}^{(k-1)} & & & \\ -s_2 a_{pq}^{(k-1)} + c_2 a_{qq}^{(k-1)} & & & \\ a_{nq}^{(k-1)} & & & \end{bmatrix} \begin{bmatrix} a_{1n}^{(k-1)} & & & \\ c_2 a_{pn}^{(k-1)} + s_2 a_{qn}^{(k-1)} & & & \\ -s_2 a_{pn}^{(k-1)} + c_2 a_{qn}^{(k-1)} & & & \\ a_{nn}^{(k-1)} & & & \end{bmatrix}$$

Here, $c_1 = \cos(\varphi)$, $s_1 = \sin(\varphi)$, $c_2 = \cos(\psi)$ and $s_2 = \sin(\psi)$ are triangular functions of rotation angles.

Note: matrix $[A^*]$ is the same as matrix $[A^{k-1}]$, except 2 rows ($\#p, \#q$) & 2 columns ($\#p, \#q$) are different

The sum of square of off-diagonal entries $S^{(k)} = \sum_{p \neq q} |a_{pq}^{(k)}|^2$ is decreased after each plane rotation [7].

We then proceed to generate a sequence of $A^{(k)}$ which leads to $\lim_{k \rightarrow \infty} A^{(k)} = \Sigma$.

With the help of plane rotation, Singular Value Decomposition of a real $m \times m$ general matrix A can be obtained by iteratively applying plane rotation.

$$\text{Jacobi} \quad A = U [\Sigma] V^T \quad (\text{or SVD})$$

Where the $m \times m$ matrix $U = \prod_{k=0}^{\infty} U_k$ and the $m \times m$ matrix $V = \prod_{k=0}^{\infty} V_k$ are two orthonormal matrices. The process stops in the case $S^{(k)}$ is small enough. The $\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_m)$ is a diagonal matrix and σ_i is singular value of the given matrix A .

The matrix Σ is diagonal matrix, as well as diagonally dominated matrix. We can do Monte Carlo method to calculate the solution of the following two equations instead of the previous equation:

$$A \vec{x} = \vec{b}$$

$$\text{hence } U \in V^T \vec{x} = \vec{b}$$

$$\text{hence } [U] V^T \vec{x} = U^T \vec{b}$$

where x is desired solution.

renamed as \vec{y}

$$\Sigma * y = U^T b$$

$$\{V^T\} x = y \Rightarrow \vec{x} = [V] \vec{y}$$

same as

$$\{V^{-1}\}$$

Annihilating off-diagonal elements

$$\begin{bmatrix} 1 & & & \\ & \cos(\theta) & \sin(\theta) & \\ & -\sin(\theta) & \cos(\theta) & \\ & & & 1 \end{bmatrix} \begin{bmatrix} x_{11} & x_{11} & x_{11} & x_{11} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix} \begin{bmatrix} 1 \\ \\ \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \\ & & & \\ & & & \end{bmatrix} \begin{bmatrix} 1 \\ \\ \\ 1 \end{bmatrix}$$

$\begin{bmatrix} \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & 0 & \blacksquare \\ \blacksquare & 0 & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare \end{bmatrix}$

Where

$$\begin{cases} x'_{23} = -\sin(\psi)(\cos(\theta)x_{22} + \sin(\theta)x_{32}) + \cos(\psi)(\cos(\theta)x_{23} + \sin(\theta)x_{33}) = 0 \\ x'_{32} = \cos(\psi)(-\sin(\theta)x_{22} + \cos(\theta)x_{32}) + \sin(\psi)(-\sin(\theta)x_{23} + \cos(\theta)x_{33}) = 0 \end{cases}$$

$$\cos(A+B) = \dots$$

Table 1: Symmetrical Cases

($s=0.1$ for all cases, used by H.Ji @ Yao Hang = OPV-CS dept)
 see Eq.(11.83) on page # 914 of K.J.Bathe's FE Book

Symmetry Pattern	Name	Order	NZ_A	No. IIR _s	NZ_M	SP_M	Cond_A	Cond_M	Cond_A*	Cond_P	BICG			BICGSTAB			GMRES		
											A	r.r.	P	A	r.r.	P	A	r.r.	P
	685_bus	685	3,249	145	1,623		4.2313e+05	1.4545e+04	4.2313e+05	6.6554e+03	520	8.8e-07	134	746.5	5.2e-07	107.5	271	9.6e-07	12
	bcsstk13	2,003	83,883	830	3,535		1.0955e+10	7.5290e+07	1.0955e+10	1.1588e+07	334,147	8.6e-07	1,079			2,131.5	1,901	9.6e-07	62
	TSOPF_FS_p9_c1	2,454	25,032	5561	2548		7.2272e+11	1.4280e+14	7.2272e+11	5.7104e+18									
	filter2D	1,668	10,750	295	5,000		4.9806e+04	93.6683	4.9806e+04	2.2829e+04	330	9.7e-07	155	256.5	7.3e-07	124.5	282	9.8e-07	14
	plbuckle	1,282	30,644	490	2,186		1.2833e+06	736.0084	1.2833e+06	1.0823e+04	1,966	9e-07	219	2,076.5	1.8e-07	178	865	9.5e-07	20
	meg4	5,860	25,258	54	22.644		4.1919e+10	4.1919e+10	4.1919e+10	1.0093	1,799	8.5e-07	2						
	msc01440	1,440	44,998	611	2,282		3.3059e+06	1.6669e+03	3.3059e+06	3.8036e+05	6,101	9.3e-07	572	9,583.5	7.5e-07	397.5	1,354	9e-07	36
	sts4098	4,098	72,356	1,187	12.694		2.1709e+08	6.9476e+07	2.1709e+08	3.4914e+06	29,903	9.5e-07	940	102,475.5	9.8e-07	1,327			
	flowmeter0	9,669	67,391	2,106	25.475		1.5893e+07	1.0493e+05	1.5893e+07	2.0939e+04	9,394	8.4e-07	302	39,923.5	9.1e-07	231			
	cbuckle	13,681	676,515	5,658	29.927		3.2991e+07	2.7873e+05	3.2991e+07	1.2017e+05	4,862	9.5e-07	232	10,671.5	9.9e-07	120.5	2,582	1e-07	15
	sterman1	1,000	3,750	159	1,948		1.5595e+04	2.3731e+03	1.5595e+04	7.8445e+03	458	9.9e-07	135	354.5	3.6e-07	87	321	9.6e-07	10
	SINa	5,743	198,787	2,417	12.335		507.1168	4.1711	507.1168	374.4517	167	9.9e-07	148	305	8.1e-07	245	159	9.4e-07	13

bicg or bicgstab stopped at iteration 1000000 without converging to the desired tolerance 1e-06;

bicg or bicgstab stopped without converging to the desired tolerance 1e-06, because a scalar quantity became too small or too large to continue computing;

(Warning): input tol might be obviously smaller than eps*cond(A) and may not be achieved by GMRES.

Table 2: Unsymmetrical Cases
 $S = 0.1$; used by H.Ji for all cases
 See page #914 of K.T. Balke's book

$[P] = [M]^T * [A]^T$

DUC received Max 3 pages from Yao Hany Li on Wed, Oct. 26 '2011

Asymmetric matrices

Sparsity pattern	Name	Order	NZ_A	No. IIR	NZ_M	SP_M	Cond_A	Cond_M	Cond_A*	Cond_P	BLOG				BLOGSTAB				GMRES			
											A	r.r.	P	r.r.	A	r.r.	P	r.r.	A	r.r.	P	r.r.
	Pd	8,081	13,036	3,197	8,732		2.6211e+1 1	3.6607e+1 0	2.6211e+1 1	8.8167e+1 0	119	8.3e-07	40	6.1e-07	185.5	7.7e-07	28.5	6.2e-07	*	*	2	4.9e-07
	adder_trans_0_1	1,814	14,579	9	8,782		4.9269e+0 3	4.9012e+0 3	4.9269e+0 3	274.2097	174	6.5e-07	4	3e-07	204.5	9.7e-07	2	2.8e-07	70	9.5e-07	3	5.3e-08
	sherman2	1,080	23,094	1,639	3,196		9.6432e+1 1	5.7507e+1 3	9.6432e+1 1	3.0149e+1 2	---	---	---	---	+	+	+	+	*	*	22	8.9e-07
	coupled	11,341	97,193	2,285	50,167		3.6959e+0 4	4.0454e+0 4	3.6959e+0 4	5.5365e+0 3	1,136	8.2e-07	118	4.6e-07	1,675.5	9.9e-07	80	9.9e-07	475	9.7e-07	90	8.7e-07
	circuit_2	4,510	21,199	751	8,192		1.3193e+0 5	5.7150e+0 4	1.3193e+0 5	122.1832	387	8.6e-07	57	3.4e-07	413.5	1e-06	29	9.9e-07	123	9.5e-07	38	7.6e-07
	rajat12	1,879	12,818	79	5,682		6.9136e+0 5	4.2169e+0 5	6.9136e+0 5	9.1298e+0 4	2,344	6.3e-07	84	5.9e-07	7,000.5	8.6e-07	92.5	9.3e-07	341	9.5e-07	54	8.3e-07
	heart3	2,339	680,341	1,500	3,372		2.1023e+0 6	6.9597e+0 8	2.1023e+0 6	4.9722e+0 7	167.526	8.9e-07	819	9.9e-07	---	---	5	9.4e-07	1.97	9.1e-07	12	9.9e-07
	thermal	3,456	66,528	1,152	11,624		10.8484	3.6524	10.8484	5.5831	19	6.7e-07	14	8.4e-07	10.5	8.8e-07	7	8.7e-07	18	7.5e-07	14	4.1e-07
	shermanACd	6,136	53,329	2,977	8,569		5.8427e+0 6	4.4011e+0 6	5.8427e+0 6	1.3557e+0 5	---	---	959	7.4e-07	+	+	5	9.7e-07	451	9.8e-07	17	1e-06
	dw1024	2,048	10,114	934	6,242		2.0932e+0 3	549.0212	2.0932e+0 3	4.5146e+0 4	1,242	9.5e-07	8	9.1e-07	1,527.5	8e-07	+	+	785	9.7e-07	35	9.1e-07
	memplus	17,758	99,147	2,731	40,652		1.2944e+0 5	1.2151e+0 5	1.2944e+0 5	6.7047e+0 4	1,027	9.4e-07	261	9.3e-07	1,190.5	8.7e-07	253.5	9.7e-07	685	9.4e-07	17	9.2e-07
	goodwin	7,320	324,772	13,992	10,271		1.9599e+1 0	6.7645e+1 8	1.9599e+1 0	1.0832e+2 4	---	---	---	---	+	+	---	---	*	*	8	3.4e-08

big or bigstab stopped at iteration 1000000 without converging to the desired tolerance 1e-06;






big or bigstab stopped without converging to the desired tolerance 1e-06, because a scalar quantity became too small or too large to continue computing;

(Warning): Input tol might be obviously smaller than eps*cond(A) and may not be achieved by GMRES.





Matrix is close to singular or badly scaled.

Yao Hany Li's (ODU # 683-6001 ext 5085)
 Yao Hany Li's (ODU # 683-6001 ext 5085)
 Yao Hany Li's (ODU # 683-6001 ext 5085)
 HTJ@cs.odu.edu
 ODU # 842-2632

Table 3: "Few Cases" where the condition # of $[P] = [M]^{-1} * [A^*]$ is worse than the condition # of $[A]$, and yet solving SLE with $[P]$ still took less # iterations, as compared to $[A]$!

Sparsity pattern	Name	Order	NZ_A	No.IJRs	NZ_M	SP_M	Cond_A	Cond_M	Cond_A*	Cond_P	BI-CG			BI-CGSTAB			GMRES			s [*]			
											A	r.r.	P	A	r.r.	P	A	r.r.	P		A	r.r.	P
	dw1024	2,048	10,114	934	6,242		2.0932e+03	549.0212	2.0932e+03	4.5146e+04	1.242	9.5e-07	3.568	1.527.5	8e-07	+	785	9.7e-07.	355	9.1e-07	0.1		
				1,313	6,236			2.3263e+03		1.0622e+05			1.466			+			337	9.8e-07	0.3		
				1,808	6,459			369.0040		1.6004e+04			372			1.052.5	1.3e-07		186	8.2e-07	0.5		
				3,651	8,722			445.3086		1.0752e+04			197			305.5	6.6e-07		120	8.4e-07	0.7		

16.1
[M]X = F

Sparsity pattern	Name	Order	NZ_A	No.IJRs	NZ_M	SP_M	Cond_A	Cond_M	Cond_A*	Cond_P	BI-CG			BI-CGSTAB			GMRES				
											A	r.f.	P	A	r.f.	P	A	r.f.	P		
	stiffness	2,442	84,282	987	4829		3.3898e+06	9.2824e+03	3.3898e+06	7.8408e+04	3.321	9.5e-07	474	7.6e-07	24,024	9.2e-07	905.5	9.1e-07	9.8e-07	318	9e-07
	bad_matrix	7,272	235,134	3040	12929		1.0488e+08	1.9742e+05	1.0488e+08	3.6213e+07	---	---	28,021	9e-07	---	10,721.5	7.3e-07	9.9e-07	983	1e-06	

10000
10-6
M < A

M < A

10-6

[A] - Pits

Considering the standard eigenproblem $\mathbf{K}\Phi = \lambda\Phi$, the k th iteration step defined in (11.72) reduces to

$$\mathbf{K}_{k+1} = \mathbf{P}_k^T \mathbf{K}_k \mathbf{P}_k \quad (11.77)$$

where \mathbf{P}_k is an orthogonal matrix; i.e., (11.73) gives

$$\mathbf{P}_k^T \mathbf{P}_k = \mathbf{I} \quad (11.78)$$

In the Jacobi solution the matrix \mathbf{P}_k is a rotation matrix that is selected in such way that an off-diagonal element in \mathbf{K}_k is zeroed. If element (i, j) is to be reduced to zero, the corresponding orthogonal matrix \mathbf{P}_k is

$$\mathbf{P}_k = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & \cos \theta & & -\sin \theta \\ & & & & 1 & \\ & & & & & \ddots \\ & & & \sin \theta & & \cos \theta \\ & & & & & & 1 \\ & & & & & & & \ddots \\ & & & & & & & & 1 \end{bmatrix} \quad (11.79)$$

i th j th column
 i th
 j th row

where θ is selected from the condition that element (i, j) in \mathbf{K}_{k+1} be zero. Denoting element (i, j) in \mathbf{K}_k by $k_{ij}^{(k)}$, we use

$$\tan 2\theta = \frac{2k_{ij}^{(k)}}{k_{ii}^{(k)} - k_{jj}^{(k)}} \quad \text{for } k_{ii}^{(k)} \neq k_{jj}^{(k)} \quad (11.80)$$

and

$$\theta = \frac{\pi}{4} \quad \text{for } k_{ii}^{(k)} = k_{jj}^{(k)} \quad (11.81)$$

It should be noted that the numerical evaluation of \mathbf{K}_{k+1} in (11.77) requires only the linear combination of two rows and two columns. In addition, advantage should also be taken of the fact that \mathbf{K}_k is symmetric for all k ; i.e., we should work on only the upper (or lower) triangular part of the matrix, including its diagonal elements.

An important point to emphasize is that although the transformation in (11.77) reduces an off-diagonal element in \mathbf{K}_k to zero, this element will again become nonzero during the transformations that follow. Therefore, for the design of an actual algorithm, we have to decide which element to reduce to zero. One choice is to always zero the largest off-diagonal element in \mathbf{K}_k . However, the search for the largest element is time-consuming, and it may be preferable to simply carry out the Jacobi transformations systematically, row by row or column by column, which is known as the *cyclic Jacobi procedure*. Running once over all off-diagonal elements is one *sweep*. The disadvantage of this procedure is that regardless of its size, an off-diagonal element is always zeroed; i.e., the element may already be nearly zero, and a rotation is still applied.

A procedure that has been used very effectively is the *threshold Jacobi method*, in which the off-diagonal elements are tested sequentially, namely, row by row (or column by column), and a rotation is applied only if the element is larger than the threshold for that sweep. To define an appropriate threshold we note that, physically, in the diagonalization of \mathbf{K} we want to reduce the coupling between the degrees of freedom i and j . A measure of this coupling is given by $(k_{ij}^2/k_{ii}k_{jj})^{1/2}$, and it is this factor that can be used effectively in deciding whether to apply a rotation. In addition to having a realistic threshold tolerance, it is also necessary to measure convergence. As described above, $\mathbf{K}_{k+1} \rightarrow \mathbf{A}$ as $k \rightarrow \infty$, but in the numerical computations we seek only a close enough approximation to the eigenvalues and corresponding eigenvectors. Let l be the last iteration; i.e., we have, to the precision required,

$$\mathbf{K}_{l+1} \doteq \mathbf{A} \quad (11.82)$$

Then we say that convergence to a tolerance s has been achieved provided that

$$\frac{|k_{ii}^{(l+1)} - k_{ii}^{(l)}|}{k_{ii}^{(l+1)}} \leq 10^{-s}; \quad i = 1, \dots, n \quad (11.83)$$

and

$$\left[\frac{(k_{ij}^{(l+1)})^2}{k_{ii}^{(l+1)} k_{jj}^{(l+1)}} \right]^{1/2} \leq 10^{-s}; \quad \text{all } i, j; i < j \quad (11.84)$$

The relation in (11.83) has to be satisfied because the element $k_{ii}^{(l+1)}$ is the current approximation to an eigenvalue, and the relation states that the current and last approximations to the eigenvalues did not change in the first s digits. This convergence measure is essentially the same as the one used in vector iteration in (11.20). The relation in (11.84) ensures that the off-diagonal elements are indeed small.

Having discussed the main aspects of the iteration, we may now summarize the actual solution procedure. The following steps have been used in a threshold Jacobi iteration.

1. Initialize the threshold for the sweep. Typically, the threshold used for sweep m may be 10^{-2m} .
2. For all i, j with $i < j$ calculate the coupling factor $[(k_{ij}^{(k)})^2/k_{ii}^{(k)}k_{jj}^{(k)}]^{1/2}$ and apply a transformation if the factor is larger than the current threshold.
3. Use (11.83) to check for convergence. If the relation in (11.83) is not satisfied, continue with the next sweep; i.e., go to step 1. If (11.83) is satisfied, check if (11.84) is also satisfied; if "yes," the iteration converged; if "no," continue with the next sweep.

So far we have stated the algorithm but we have not shown that convergence will indeed always occur. The proof of convergence has been given elsewhere (see J. H. Wilkinson [A]) and will not be repeated here because little additional insight into the working of the solution procedure would be gained. However, one important point should be noted — that convergence is quadratic once the off-diagonal elements are small. Since rapid convergence is obtained once the off-diagonal elements are small, little extra cost is involved in solving for the eigensystem to high accuracy when an approximate solution has been obtained. In practical solutions we use $m = 2$ and $s = 12$, and about six sweeps are required for solution of the eigensystem to high accuracy. A program used is given in the next section, when we discuss the solution of the generalized eigenproblem $\mathbf{K}\phi = \lambda \mathbf{M}\phi$.

EXAMPLE 11.9: Calculate the eigensystem of the matrix \mathbf{K} , where

$$\mathbf{K} = \begin{bmatrix} 5 & -4 & 1 & 0 \\ -4 & 6 & -4 & 1 \\ 1 & -4 & 6 & -4 \\ 0 & 1 & -4 & 5 \end{bmatrix}$$

Use the threshold Jacobi iteration described above.

To demonstrate the solution algorithm we give one sweep in detail and then the results obtained in the next sweeps.

For sweep 1 we have as a threshold 10^{-2} . We therefore obtain the following results. For

$$i = 1, j = 2:$$

$$\cos \theta = 0.7497; \quad \sin \theta = 0.6618$$

$$\mathbf{P}_1 = \begin{bmatrix} 0.7497 & -0.6618 & 0 & 0 \\ 0.6618 & 0.7497 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{P}_1^T \mathbf{K} \mathbf{P}_1 = \begin{bmatrix} -1.469 & 0 & -1.898 & 0.6618 \\ 0 & 9.531 & -3.661 & 0.7497 \\ -1.898 & -3.661 & 6 & -4 \\ 0.6618 & 0.7497 & -4 & 5 \end{bmatrix}$$

$$\text{For } i = 1, j = 3:$$

$$\cos \theta = 0.9398; \quad \sin \theta = 0.3416$$

$$\mathbf{P}_2 = \begin{bmatrix} 0.9398 & 0 & -0.3416 & 0 \\ 0 & 1 & 0 & 0 \\ 0.3416 & 0 & 0.9398 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{P}_2^T \mathbf{P}_1^T \mathbf{K} \mathbf{P}_1 \mathbf{P}_2 = \begin{bmatrix} -0.7792 & -1.250 & 0 & -0.7444 \\ -1.250 & 9.531 & -3.440 & 0.7497 \\ 0 & -3.440 & 6.690 & -3.986 \\ -0.7444 & 0.7497 & -3.986 & 5 \end{bmatrix}$$

$$\mathbf{P}_1 \mathbf{P}_2 = \begin{bmatrix} 0.7046 & -0.6618 & -0.2561 & 0 \\ 0.6220 & 0.7497 & -0.2261 & 0 \\ 0.3416 & 0 & 0.9398 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{For } i = 1, j = 4:$$

$$\cos \theta = 0.9857; \quad \sin \theta = 0.1687$$

$$\mathbf{P}_3 = \begin{bmatrix} 0.9857 & 0 & 0 & -0.1687 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0.1687 & 0 & 0 & 0.9857 \end{bmatrix}$$

$$\mathbf{P}_3^T \mathbf{P}_2^T \mathbf{P}_1^T \mathbf{K} \mathbf{P}_1 \mathbf{P}_2 \mathbf{P}_3 = \begin{bmatrix} 0.6518 & -1.106 & -0.6725 & 0 \\ -1.106 & 9.531 & -3.440 & 0.9499 \\ -0.6725 & -3.440 & 6.690 & -3.928 \\ 0 & 0.9499 & -3.928 & 5.127 \end{bmatrix}$$

$$\mathbf{P}_1 \mathbf{P}_2 \mathbf{P}_3 = \begin{bmatrix} 0.6945 & -0.6618 & -0.2561 & -0.1189 \\ 0.6131 & 0.7497 & -0.2261 & -0.1050 \\ 0.3367 & 0 & 0.9398 & -0.0576 \\ 0.1687 & 0 & 0 & 0.9857 \end{bmatrix}$$

For $i = 2, j = 3$:

$$\cos \theta = 0.8312; \quad \sin \theta = -0.5560$$

$$\mathbf{P}_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.8312 & 0.5560 & 0 \\ 0 & -0.5560 & 0.8312 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{P}_4^T \mathbf{P}_3^T \mathbf{P}_2^T \mathbf{P}_1^T \mathbf{K} \mathbf{P}_1 \mathbf{P}_2 \mathbf{P}_3 \mathbf{P}_4 = \begin{bmatrix} 0.6518 & 0.5453 & -1.174 & 0 \\ -0.5453 & 11.83 & 0 & 2.974 \\ -1.174 & 0 & 4.388 & -2.737 \\ 0 & 2.974 & -2.737 & 5.127 \end{bmatrix}$$

$$\mathbf{P}_1 \mathbf{P}_2 \mathbf{P}_3 \mathbf{P}_4 = \begin{bmatrix} 0.6945 & -0.4077 & -0.5808 & -0.1189 \\ 0.6131 & 0.7488 & 0.2289 & -0.1050 \\ 0.3367 & -0.5226 & 0.7812 & -0.0576 \\ 0.1682 & 0 & 0 & 0.9857 \end{bmatrix}$$

For $i = 2, j = 4$:

$$\cos \theta = 0.9349; \quad \sin \theta = 0.3549$$

$$\mathbf{P}_5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.9349 & 0 & -0.3549 \\ 0 & 0 & 1 & 0 \\ 0 & 0.3549 & 0 & 0.9349 \end{bmatrix}$$

$$\mathbf{P}_5^T \mathbf{P}_4^T \mathbf{P}_3^T \mathbf{P}_2^T \mathbf{P}_1^T \mathbf{K} \mathbf{P}_1 \mathbf{P}_2 \mathbf{P}_3 \mathbf{P}_4 \mathbf{P}_5 = \begin{bmatrix} 0.6518 & 0.5098 & -1.174 & 0.1935 \\ -0.5098 & 12.96 & 0.9713 & 0 \\ -1.174 & -0.9713 & 4.388 & -2.559 \\ 0.1935 & 0 & -2.559 & 3.999 \end{bmatrix}$$

$$\mathbf{P}_1 \mathbf{P}_2 \mathbf{P}_3 \mathbf{P}_4 \mathbf{P}_5 = \begin{bmatrix} 0.6945 & -0.4233 & -0.5808 & 0.0335 \\ 0.6131 & 0.6628 & 0.2289 & -0.3639 \\ 0.3367 & 0.5090 & 0.7812 & 0.1316 \\ 0.1687 & 0.3498 & 0 & 0.9213 \end{bmatrix}$$

To complete the sweep, we zero element (3, 4), using

$$\cos \theta = 0.7335; \quad \sin \theta = -0.6797$$

$$\mathbf{P}_6 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.7335 & 0.6797 \\ 0 & 0 & -0.6797 & 0.7335 \end{bmatrix}$$

and hence, the approximations obtained for \mathbf{A} and $\mathbf{\Phi}$ are

$$\mathbf{A} \doteq \mathbf{P}_6^T \dots \mathbf{P}_1^T \mathbf{K} \mathbf{P}_1 \dots \mathbf{P}_6$$

$$\text{i.e.,} \quad \mathbf{A} \doteq \begin{bmatrix} 0.6518 & -0.5098 & -0.9926 & -0.6560 \\ -0.5098 & 12.96 & -0.7124 & -0.6602 \\ -0.9926 & -0.7124 & 6.7596 & 0 \\ -0.6560 & -0.6602 & 0 & 1.6272 \end{bmatrix}$$

and

$$\mathbf{\Phi} \doteq \mathbf{P}_1 \dots \mathbf{P}_6$$

$$\text{i.e.,} \quad \mathbf{\Phi} \doteq \begin{bmatrix} 0.6945 & -0.4233 & -0.4488 & -0.3702 \\ 0.6131 & 0.6628 & 0.4152 & -0.1113 \\ 0.3367 & -0.5090 & 0.4835 & 0.6275 \\ 0.1687 & 0.3498 & -0.6264 & 0.6759 \end{bmatrix}$$

After the second sweep we obtain

$$\mathbf{A} \doteq \begin{bmatrix} 0.1563 & -0.3635 & 0.0063 & -0.0176 \\ -0.3635 & 13.08 & -0.0020 & 0 \\ 0.0063 & -0.0020 & 6.845 & 0 \\ -0.0176 & 0 & 0 & 1.910 \end{bmatrix}$$

$$\mathbf{\Phi} \doteq \begin{bmatrix} 0.3875 & -0.3612 & -0.6017 & -0.5978 \\ 0.5884 & 0.6184 & 0.3710 & -0.3657 \\ 0.6148 & -0.5843 & 0.3714 & 0.3777 \\ 0.3546 & 0.3816 & -0.6020 & 0.6052 \end{bmatrix}$$

And after the third sweep we have

$$\mathbf{A} \doteq \begin{bmatrix} 0.1459 & & & \\ & 13.09 & & \\ & & 6.854 & \\ & & & 1.910 \end{bmatrix}$$

$$\mathbf{\Phi} \doteq \begin{bmatrix} 0.3717 & -0.3717 & -0.6015 & -0.6015 \\ 0.6015 & 0.6015 & 0.3717 & -0.3717 \\ 0.6015 & -0.6015 & 0.3717 & 0.3717 \\ 0.3717 & 0.3717 & -0.6015 & 0.6015 \end{bmatrix}$$

The approximation for Λ is diagonal to the precision given, and we can use

$$\begin{aligned}\lambda_1 &\doteq 0.1459; & \Phi_1 &\doteq \begin{bmatrix} 0.3717 \\ 0.6015 \\ 0.6015 \\ 0.3717 \end{bmatrix} \\ \lambda_2 &\doteq 1.910; & \Phi_2 &\doteq \begin{bmatrix} -0.6015 \\ -0.3717 \\ 0.3717 \\ 0.6015 \end{bmatrix} \\ \lambda_3 &\doteq 6.854; & \Phi_3 &\doteq \begin{bmatrix} -0.6015 \\ 0.3717 \\ 0.3717 \\ -0.6015 \end{bmatrix} \\ \lambda_4 &\doteq 13.09; & \Phi_4 &\doteq \begin{bmatrix} -0.3717 \\ 0.6015 \\ -0.6015 \\ 0.3717 \end{bmatrix}\end{aligned}$$

It should be noted that the eigenvalues and eigenvectors did not appear in the usual order in the approximation for Λ and Φ .

In the following example we demonstrate the quadratic convergence when the off-diagonal elements are already small (see J. H. Wilkinson [B]).

EXAMPLE 11.10: Consider the Jacobi solution of the eigenproblem $\mathbf{K}\Phi = \lambda\Phi$, where

$$\mathbf{K} = \begin{bmatrix} k_{11} & o(\epsilon) & o(\epsilon) \\ o(\epsilon) & k_{22} & o(\epsilon) \\ o(\epsilon) & o(\epsilon) & k_{33} \end{bmatrix}$$

The symbol $o(\epsilon)$ signifies "of order ϵ ," where $\epsilon \ll k_{ii}$, $i = 1, 2, 3$. Show that after one complete sweep, all off-diagonal elements are of order ϵ^2 , meaning that convergence is quadratic.

Since the rotations to be applied are small, we make the assumption that $\sin \theta \approx \theta$ and $\cos \theta \approx 1$. Hence, the relation in (11.80) gives

$$\theta = \frac{k_{ij}^{(k)}}{k_{ii}^{(k)} - k_{jj}^{(k)}}$$

In one sweep we need to set to zero, in succession, all off-diagonal elements. Using \mathbf{K} , we obtain \mathbf{K}_2 by zeroing element (1, 2) in \mathbf{K}_1 ,

$$\mathbf{K}_2 = \mathbf{P}_1^T \mathbf{K}_1 \mathbf{P}_1$$

where

$$\mathbf{P}_1 = \begin{bmatrix} 1 & \frac{-o(\epsilon)}{k_{11} - k_{22}} & 0 \\ \frac{o(\epsilon)}{k_{11} - k_{22}} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Hence,

$$\mathbf{K}_2 = \begin{bmatrix} k_{11} + o(\epsilon^2) & 0 & o(\epsilon) \\ 0 & k_{22} + o(\epsilon^2) & o(\epsilon) \\ o(\epsilon) & o(\epsilon) & k_{33} \end{bmatrix}$$

Similarly, we zero element (1, 3) in \mathbf{K}_2 to obtain \mathbf{K}_3 ,

$$\mathbf{K}_3 = \begin{bmatrix} k_{11} + o(\epsilon^2) & o(\epsilon^2) & 0 \\ o(\epsilon^2) & k_{22} + o(\epsilon^2) & o(\epsilon) \\ 0 & o(\epsilon) & k_{33} + o(\epsilon^2) \end{bmatrix}$$

Finally, we zero element (2, 3) in \mathbf{K}_3 and have

$$\mathbf{K}_4 = \begin{bmatrix} k_{11} + o(\epsilon^2) & o(\epsilon^2) & o(\epsilon^2) \\ o(\epsilon^2) & k_{22} + o(\epsilon^2) & 0 \\ o(\epsilon^2) & 0 & k_{33} + o(\epsilon^2) \end{bmatrix}$$

with all off-diagonal elements at least $o(\epsilon^2)$.

11.3.2 The Generalized Jacobi Method

In the previous section we discussed the solution of the standard eigenproblem $\mathbf{K}\Phi = \lambda\Phi$ using the conventional Jacobi rotation matrices in order to reduce \mathbf{K} to diagonal form. To solve the generalized problem $\mathbf{K}\Phi = \lambda\mathbf{M}\Phi$, $\mathbf{M} \neq \mathbf{I}$, using the standard Jacobi method, it would be necessary to first transform the problem into a standard form. However, this transformation can be dispensed with by using a generalized Jacobi solution method that operates directly on \mathbf{K} and \mathbf{M} (see S. Falk and P. Langemeyer [A] and K. J. Bathe [A]). The algorithm proceeds as summarized in (11.72) to (11.76) and is a natural extension of the standard Jacobi solution scheme; i.e., the generalized method reduces to the scheme presented for the problem $\mathbf{K}\Phi = \lambda\Phi$ when \mathbf{M} is an identity matrix.

Referring to the discussion in the previous section, in the generalized Jacobi iteration we use the following matrix \mathbf{P}_k :

$$\mathbf{P}_k = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix} \quad \begin{array}{l} \text{\textit{i}th} \\ \text{\textit{j}th column} \end{array} \quad \begin{array}{l} \\ \\ \text{\textit{i}th} \\ \text{\textit{j}th row} \end{array} \quad (11.85)$$

α (between $\text{\textit{i}th}$ and $\text{\textit{j}th column}$)
 γ (between $\text{\textit{i}th}$ and $\text{\textit{j}th row}$)

where the constants α and γ are selected in such a way as to reduce to zero simultaneously elements (i, j) in \mathbf{K}_k and \mathbf{M}_k . Therefore, the values of α and γ are a function of the elements $k_{ij}^{(k)}$, $k_{ii}^{(k)}$, $k_{jj}^{(k)}$, $m_{ij}^{(k)}$, $m_{ii}^{(k)}$, and $m_{jj}^{(k)}$, where the superscript (k) indicates that the k th iteration is considered. Performing the multiplications $\mathbf{P}_k^T \mathbf{K}_k \mathbf{P}_k$ and $\mathbf{P}_k^T \mathbf{M}_k \mathbf{P}_k$ and using the condition

that $k_{ij}^{(k+1)}$ and $m_{ij}^{(k+1)}$ shall be zero, we obtain the following two equations for α and γ :

$$\alpha k_{ii}^{(k)} + (1 + \alpha\gamma)k_{ij}^{(k)} + \gamma k_{jj}^{(k)} = 0 \quad (11.86)$$

and

$$\alpha m_{ii}^{(k)} + (1 + \alpha\gamma)m_{ij}^{(k)} + \gamma m_{jj}^{(k)} = 0 \quad (11.87)$$

If

$$\frac{k_{ii}^{(k)}}{m_{ii}^{(k)}} = \frac{k_{ij}^{(k)}}{m_{ij}^{(k)}} = \frac{k_{jj}^{(k)}}{m_{jj}^{(k)}}$$

(i.e., the submatrices considered are scalar multiples, which may be regarded to be a trivial case), we use $\alpha = 0$ and $\gamma = -k_{ij}^{(k)}/k_{jj}^{(k)}$. In general, to solve for α and γ from (11.86) and (11.87), we define

$$\left. \begin{aligned} \bar{k}_{ii}^{(k)} &= k_{ii}^{(k)} m_{ij}^{(k)} - m_{ii}^{(k)} k_{ij}^{(k)} \\ \bar{k}_{jj}^{(k)} &= k_{jj}^{(k)} m_{ij}^{(k)} - m_{jj}^{(k)} k_{ij}^{(k)} \\ \bar{k}^{(k)} &= k_{ii}^{(k)} m_{jj}^{(k)} - k_{jj}^{(k)} m_{ii}^{(k)} \end{aligned} \right\} \quad (11.88)$$

and

$$\gamma = -\frac{\bar{k}_{ii}^{(k)}}{x}; \quad \alpha = \frac{\bar{k}_{jj}^{(k)}}{x} \quad (11.89)$$

The value of x needed to obtain α and γ is then to be determined using

$$x = \frac{\bar{k}^{(k)}}{2} + \text{sign}(\bar{k}^{(k)}) \sqrt{\left(\frac{\bar{k}^{(k)}}{2}\right)^2 + \bar{k}_{ii}^{(k)} \bar{k}_{jj}^{(k)}} \quad (11.90)$$

The relations for α and γ are used and have primarily been developed for the case of \mathbf{M} being a positive definite full or banded mass matrix. In that case (and, in fact, also under less restrictive conditions), we have

$$\left(\frac{\bar{k}^{(k)}}{2}\right)^2 + \bar{k}_{ii}^{(k)} \bar{k}_{jj}^{(k)} > 0$$

and hence x is always nonzero. In addition, $\det \mathbf{P}_k \neq 0$, which indeed is the necessary condition for the algorithm to work.

The generalized Jacobi solution procedure has been used a great deal in the subspace iteration method (see Section 11.6) and when a consistent mass idealization is employed. However, other situations may arise as well. Assume that \mathbf{M} is a diagonal mass matrix, $\mathbf{M} \neq \mathbf{I}$ and $m_{ii} > 0$, in which case we employ in (11.88),

$$\bar{k}_{ii}^{(k)} = -m_{ii}^{(k)} k_{ij}^{(k)}; \quad \bar{k}_{jj}^{(k)} = -m_{jj}^{(k)} k_{ij}^{(k)} \quad (11.91)$$

and otherwise (11.85) to (11.90) are used as before. However, if $\mathbf{M} = \mathbf{I}$, the relation in (11.87) yields $\alpha = -\gamma$, and we recognize that \mathbf{P}_k in (11.85) is a multiple of the rotation matrix defined in (11.79) (see Example 11.11). In addition, it should be mentioned that the solution procedure can be adapted to solve the problem $\mathbf{K}\Phi = \lambda \mathbf{M}\Phi$ when \mathbf{M} is a diagonal matrix with some zero diagonal elements.

The complete solution process is analogous to the Jacobi iteration in the solution of the problem $\mathbf{K}\Phi = \lambda \Phi$, which was presented in the preceding section. The differences are that now a mass coupling factor $[(m_{ij}^{(k)})^2 / m_{ii}^{(k)} m_{jj}^{(k)}]^{1/2}$ must also be calculated, unless \mathbf{M} is diagonal, and the transformation is applied to \mathbf{K}_k and \mathbf{M}_k . Convergence is measured by comparing successive eigenvalue approximations and by testing if all off-diagonal elements are small enough; i.e., with l being the last iteration, convergence has been achieved if

$$\frac{|\lambda_i^{(l+1)} - \lambda_i^{(l)}|}{\lambda_i^{(l)}} \leq 10^{-6}, \quad i = 1, \dots, n \quad (11.92)$$

and γ :

(11.86)

(11.87)

where

$$\lambda_i^{(l)} = \frac{k_{ii}^{(l)}}{m_{ii}^{(l)}}; \quad \lambda_i^{(l+1)} = \frac{k_{ii}^{(l+1)}}{m_{ii}^{(l+1)}} \quad (11.93)$$

and

$$\left[\frac{(k_{ij}^{(l+1)})^2}{k_{ii}^{(l+1)} k_{jj}^{(l+1)}} \right]^{1/2} \leq 10^{-s}; \quad \left[\frac{(m_{ij}^{(l+1)})^2}{m_{ii}^{(l+1)} m_{jj}^{(l+1)}} \right]^{1/2} \leq 10^{-s}; \quad \text{all } i, j; i < j \quad (11.94)$$

be a trivial
(1.86) and

(11.88)

(11.89)

(11.90)

he case of
also under

necessary

e subspace
employed.
ss matrix,

(11.91)

relation in
e rotation
ed that the
a diagonal

tion of the
es are that
I is diago-
y compar-
ments are
ed if

(11.92)

where 10^{-s} is the convergence tolerance.

Table 11.1 summarizes the solution procedure for the case of \mathbf{M} being full (or banded) and positive definite. The relations given in Table 11.1 are employed directly in the subrou-tine JACOBI, which is presented at the end of this section. Table 11.1 also gives an operation count of the solution process and the storage requirements. The total number of operations in one sweep as given in the table are an upper bound because it is assumed that both matrices are full and that all off-diagonal elements are zeroed; i.e., the threshold tolerance is never passed. Considering the number of sweeps required for solution, the same experience as with the solution of standard eigenproblems holds; i.e., with $m = 2$ and $s = 12$ in the iteration (see Section 11.3.1) about six sweeps are required for solution of the eigensystem to high accuracy.

TABLE 11.1 Summary of generalized Jacobi solution

Operation	Calculation	Number of operations	Required storage
Calculation of coupling factors	$\frac{(k_{ij}^{(k)})^2}{k_{ii}^{(k)} k_{jj}^{(k)}}; \quad \frac{(m_{ij}^{(k)})^2}{m_{ii}^{(k)} m_{jj}^{(k)}}$	6	
Transformation to zero elements (i, j)	$\bar{k}_{ii}^{(k)} = k_{ii}^{(k)} m_{ij}^{(k)} - m_{ii}^{(k)} k_{ij}^{(k)}$ $\bar{k}_{jj}^{(k)} = k_{jj}^{(k)} m_{ij}^{(k)} - m_{jj}^{(k)} k_{ij}^{(k)}$ $\bar{k}^{(k)} = k_{ii}^{(k)} m_{ij}^{(k)} - k_{ij}^{(k)} m_{ii}^{(k)}$		
	$x = \frac{\bar{k}^{(k)}}{2} + (\text{sign } \bar{k}^{(k)}) \sqrt{\left(\frac{\bar{k}^{(k)}}{2}\right)^2 + \bar{k}_{ii}^{(k)} \bar{k}_{jj}^{(k)}}$ $\gamma = -\frac{\bar{k}_{ii}^{(k)}}{x}, \alpha = \frac{\bar{k}_{jj}^{(k)}}{x}$	$4n + 12$	Using symmetry of matrices $n(n+2)$
Calculation of eigenvectors	$\mathbf{K}_{k+1} = \mathbf{P}_k^T \mathbf{K}_k \mathbf{P}_k, \quad \mathbf{M}_{k+1} = \mathbf{P}_k^T \mathbf{M}_k \mathbf{P}_k$ $(\mathbf{P}_1 \dots \mathbf{P}_{k-1}) \mathbf{P}_k$	$2n$	n^2
Total for one sweep		$3n^3 + 6n^2$	$2n^2 + 2n$

The following examples demonstrate some of the characteristics of the generalized Jacobi solution algorithm.

EXAMPLE 11.11: Prove that the generalized Jacobi method reduces to the standard technique when $\mathbf{M} = \mathbf{I}$.

For the proof we need only consider the calculation of the transformation matrices that would be used to zero typical off-diagonal elements. We want to show that the transformation matrices obtained in the standard and generalized Jacobi methods are multiples of each other; namely, in that case we could, by proper scaling, obtain the standard method from the

generalized scheme. Since each step of iteration consists of applying a rotation in the (i, j) th plane, we can without loss of generality consider the solution of the problem

$$\begin{bmatrix} k_{11} & k_{12} \\ k_{12} & k_{22} \end{bmatrix} \Phi = \lambda \Phi$$

Using (11.88) to (11.90), we thus obtain

$$\alpha = -\gamma; \quad \mathbf{P}_1 = \begin{bmatrix} 1 & -\gamma \\ \gamma & 1 \end{bmatrix} \quad (a)$$

and

$$\gamma = \frac{-k_{11} + k_{22} \pm \sqrt{(k_{11} - k_{22})^2 + 4k_{12}^2}}{2k_{12}}$$

On the other hand, in the standard Jacobi solution we use

$$\mathbf{P}_1 = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$

which may be written as

$$\mathbf{P}_1 = \cos \theta \begin{bmatrix} 1 & -\tan \theta \\ \tan \theta & 1 \end{bmatrix} \quad (b)$$

Thus, \mathbf{P}_1 in (b) would be a multiple of \mathbf{P}_1 in (a) if $\tan \theta = \gamma$. In the standard Jacobi method we obtain $\tan 2\theta$ using (11.80). In this case, we have

$$\tan 2\theta = \frac{2k_{12}}{k_{11} - k_{22}} \quad (c)$$

We also have, by simple trigonometry,

$$\tan 2\theta = \frac{2 \tan \theta}{1 - \tan^2 \theta} \quad (d)$$

Using (c) and (d), we can solve for $\tan \theta$ to be used in (b) and obtain

$$\tan \theta = \frac{-k_{11} + k_{22} \pm \sqrt{(k_{11} - k_{22})^2 + 4k_{12}^2}}{2k_{12}}$$

Hence, $\gamma = \tan \theta$ and the generalized Jacobi iteration is equivalent to the standard method when $\mathbf{M} = \mathbf{I}$.

EXAMPLE 11.12: Use the generalized Jacobi method to calculate the eigensystem of the problem $\mathbf{K}\Phi = \lambda\mathbf{M}\Phi$.

(1) In the first case let

$$\mathbf{K} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}; \quad \mathbf{M} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

We note that \mathbf{K} is singular, and hence we expect a zero eigenvalue.

(2) Then let

$$\mathbf{K} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}; \quad \mathbf{M} = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$$

in which case we have an infinite eigenvalue.

For solution, we use the relations in (11.85) to (11.90). Considering the problem in case (1), we obtain

$$\begin{aligned}\bar{k}_{11}^{(1)} &= 3; & \bar{k}_{22}^{(1)} &= 3; & \bar{k}^{(1)} &= 0 \\ x &= 3; & \gamma &= -1; & \alpha &= 1\end{aligned}$$

$$\mathbf{P}_1 = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$

Hence,
$$\mathbf{P}_1^T \mathbf{K} \mathbf{P}_1 = \begin{bmatrix} 4 & 0 \\ 0 & 0 \end{bmatrix}; \quad \mathbf{P}_1^T \mathbf{M} \mathbf{P}_1 = \begin{bmatrix} 2 & 0 \\ 0 & 6 \end{bmatrix}$$

To obtain $\mathbf{\Lambda}$ and $\mathbf{\Phi}$ we use (11.75) and (11.76) and arrange the columns in the matrices in the appropriate order. Hence,

$$\mathbf{\Lambda} = \begin{bmatrix} 0 & \\ & 2 \end{bmatrix}; \quad \mathbf{\Phi} = \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

Now consider the problem in case (2). Here we have

$$\begin{aligned}\bar{k}_{11}^{(1)} &= -2; & \bar{k}_{22}^{(1)} &= 0; & \bar{k}^{(1)} &= -4 \\ x &= -4; & \alpha &= 0; & \gamma &= -\frac{1}{2}\end{aligned}$$

$$\mathbf{P}_1 = \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & 1 \end{bmatrix}$$

Hence,
$$\mathbf{P}_1^T \mathbf{K} \mathbf{P}_1 = \begin{bmatrix} \frac{3}{2} & 0 \\ 0 & 2 \end{bmatrix}; \quad \mathbf{P}_1^T \mathbf{M} \mathbf{P}_1 = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$$

and
$$\mathbf{\Lambda} = \begin{bmatrix} \frac{3}{4} & \\ & \infty \end{bmatrix}; \quad \mathbf{\Phi}_1 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{2\sqrt{2}} \end{bmatrix}$$

The above discussion of the generalized Jacobi solution method has already indicated in some way the advantages of the solution technique. First, the transformation of the generalized eigenproblem to the standard form is avoided. This is particularly advantageous (1) when the matrices are ill-conditioned, and (2) when the off-diagonal elements in \mathbf{K} and \mathbf{M} are already small or, equivalently when there are only a few nonzero off-diagonal elements. In the first case the direct solution of $\mathbf{K}\mathbf{\Phi} = \lambda\mathbf{M}\mathbf{\Phi}$ avoids the solution of a standard eigenproblem of a matrix with very large and very small elements (see Section 10.2.5). In the second case the eigenproblem is already nearly solved, because the zeroing of small or only a few off-diagonal elements in \mathbf{K} and \mathbf{M} will not result in a large change in the diagonal elements of the matrices, the ratios of which are the eigenvalues. In addition, fast convergence can be expected when the off-diagonal elements are small (see Section 11.3.1). We will see that this case arises in the subspace iteration method described in Section 11.6, which is one reason why the generalized Jacobi method is used effectively in that technique.

$$[M]^{-1} [A^*] \vec{y} = \vec{b}^*$$

$$[M]^{-1} [A^*] \vec{y} = \vec{b}^*$$

$$[M]^{-1} [A^*] \vec{y} = \vec{b}^*$$

$$[M]^{-1} [A^*] \vec{y} = \vec{b}^*$$

A Method for Constructing Diagonally Dominant Preconditioners based on Jacobi Rotations

Jin Yun Yuan*

Plamen Y. Yalamov†

Abstract

A method is presented to make a given matrix strictly diagonally dominant as much as possible based on Jacobi rotations in this paper. The strictly diagonally dominant rows are used to build a preconditioner for some iterative method. Roundoff error analysis of the method is also given. The numerical tests illustrate that the method works very well even for very ill-conditioned linear systems.

AMS subject classification: 65F10

Key words: strictly diagonally dominant matrix, preconditioned iterative method, preconditioner, Jacobi rotation.

*Departamento de Matemática, Universidade Federal do Paraná, Centro Politécnico, CP: 19.081, CEP: 81531-990, Curitiba, Brazil, e-mail: jin@mat.ufpr.br. The work was partially supported by Grant 301039/93-8 from CNPq, Brazil.

†Center of Applied Mathematics and Informatics, University of Rousse, 7017 Rousse, Bulgaria, e-mail: yalamov@ami.ru.acad.bg. The work was finished during the visit of this author to the Federal University of Paraná. The research was partially supported by Grant MM-434/94 from the Bulgarian Ministry of Education and Science.

1 Introduction

It is well-known that strictly diagonally dominant matrices are usually well conditioned, and that the solution of linear systems with such matrices is stable (for example, see [4, p. 120]). Many iterative methods, such as the Jacobi method, the Gauss-Seidel and the SOR method, are convergent for diagonally dominant systems. Unfortunately, in practice we often have of course non-diagonally dominant matrices. Hence, iterative methods can perform badly when applied to general matrices.

We call a matrix $A \in \mathcal{R}^{n \times n}$ strictly diagonally dominant, if

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|, \quad i = 1, \dots, n.$$

For every nonsingular matrix A it follows immediately from the Singular Value Decomposition (SVD) that there exist nonsingular matrices P and Q such that PAQ is strictly diagonally dominant. A different proof of this fact is given in [8]. The author also showed that there exists a nonsingular matrix P such that PA is strictly diagonally dominant. A tridiagonal matrix P is constructed such that PA is strictly diagonally dominant for 3-cyclic matrices as an example in [8].

It is very clear that after finding the preconditioners P and Q such that PAQ is strictly diagonally dominant, we can apply iterative methods for solving

$$PAQy = Pb,$$

and

$$x = Qy$$

instead of solving

$$Ax = b. \Rightarrow$$

$$PAX = Pb$$

$$\text{Let } x = Qy$$

Then

$$PA(Qy) = Pb$$

Therefore, the key problem is to find P and Q cheaply. The simplest way for obtaining such strictly diagonally dominant matrix is the SVD, where $P = U$ and $Q = V^T$. But the price of the SVD is prohibitively high for solving linear systems. Unfortunately, at the moment there is no other cheap way to construct such preconditioners P and Q for general matrices A .

In the present work we consider some cheaper approach for the desired preconditioners P and Q . We propose a method which transforms the matrix A to either a strictly diagonally dominant matrix, or a matrix in which some of the rows are strictly diagonally dominant which can be used to build a preconditioner for some iterative method (see [1]).

The outline of the paper is as follows. The method is established in Section 2. In the last section, some numerical examples and some comments are given.

2 The method of diagonal dominance

The SVD reduces matrix A to a diagonal matrix by orthogonal transformations. The LU and the QR factorizations (also other incomplete or modified factorizations) reduce the matrix A to some triangular matrix. We must eliminate many nonzero entries with these approaches which will cause fill-in problems. Fortunately, Yuan's result in [8] shows that it is not necessary to reduce the matrix A to a triangular or diagonal matrix for solving linear systems $Ax = b$ by iterative methods. We can keep more nonzero elements for the dense case, and as much as needed for the sparse case. We just reduce the matrix A to some strictly diagonally dominant (but not triangular or diagonal) matrix by orthogonal transformations. With the new strongly diagonally dominant matrix we can solve the linear system by some iterative

methods, such as the Gauss-Seidel method and the SOR method.

The idea of the method is as follows. One of the approaches to obtain the SVD of a given matrix is to apply Jacobi iterations (see [4, p.457]). Such iterations reduce the magnitude of the off-diagonal entries until all the off-diagonal entries are small enough.

The SVD is the limit case when the matrix obtained is diagonally dominant to the maximal possible extent. The idea is to stop process when some, or all, of the rows of the matrix are diagonally dominant but not when we have all elements very small. The method can be given as follows:

Choose m (the number of Jacobi iterations to be applied), and $\delta > 0$ (the level of diagonal dominance);

for $k = 1, \dots, m$

Find $a_{i_0 j_0}$ such that $|a_{i_0 j_0}| = \max_{i \neq j} |a_{ij}|$;

$i_1 = i_0$; $j_1 = j_0$;

Compute $Q_k = \begin{pmatrix} a_{i_1 i_1} & a_{i_1 j_1} \\ a_{j_1 i_1} & a_{j_1 j_1} \end{pmatrix} = U_k \Sigma_k V_k^T$ (the SVD of matrix Q_k);

Apply the transformation U_k^T to rows i_1 and j_1 of matrix A : $A = (U^{(k)})^T A$, where the matrix $U^{(k)}$ differs from the unity matrix in the following positions: $U_{i_1 i_1}^{(k)} = (U_k)_{11}$, $U_{i_1 j_1}^{(k)} = (U_k)_{12}$, $U_{j_1 i_1}^{(k)} = (U_k)_{21}$, $U_{j_1 j_1}^{(k)} = (U_k)_{22}$;

Apply the transformation V_k to columns i_1 and j_1 of matrix A : $A = A V^{(k)}$, where the matrix $V^{(k)}$ differs from the unity matrix in the following positions: $V_{i_1 i_1}^{(k)} = (V_k)_{11}$, $V_{i_1 j_1}^{(k)} = (V_k)_{12}$, $V_{j_1 i_1}^{(k)} = (V_k)_{21}$, $V_{j_1 j_1}^{(k)} = (V_k)_{22}$;

Compute $b = (U^{(k)})^T b$;

Modify RHS also! $\rightarrow b = U^T b$

Golub & Van Loan Book!

small value

$K^{k+1} = P^T K^k P$ (Duc's book, p. 279)

$A = U^T A$

$A = (U^T A) V$

$$\text{or } |a_{ii}| \geq \sum_{j \neq i} |a_{ij}| + \delta$$

\uparrow This is NOT the original matrix $[A]$
 This is the updated matrix $[A]$

matrix X exists

$[A]$

$M =$

~~Last A~~

~~matrix~~

~~0~~

~~4/2~~

Compute $x = V^{(1)} \dots V^{(m)} y$.

In practice the number δ needs not be large because in most cases it measures the diagonal dominance in row l . The previous rows can have larger diagonal dominance. From our numerical experience we recommend the value $\delta \in [10^{-2}, 10^{-6}]$ (in double precision arithmetic with machine roundoff unit $\approx 10^{-16}$).

The preconditioner can be also chosen in other ways. For example, instead of $A(1 : l, 1 : l)$ we can take the tridiagonal part of $A(1 : l, 1 : l)$, and $M(i, i) = A(l, l), i = l + 1, \dots, n$, again.

(c) $\boxed{Ax = b}$ where $A = \begin{pmatrix} K_{BB} & K_{BI} & K_{IB} \end{pmatrix}$ will this diag. dominant idea still work/valid??

(a) π Hao's $\{m\}$ is
constructed in the ~~same~~ way
(or different) as this paper??

(b) At each ^{iteration} step (to zero the largest value) YES (or different) as this paper?!

the # NZ in $[M]$ keeps decreasing it is possible to increase the # NZ

3 Numerical experiments

In this section, we consider three examples. The experiments are done by Matlab. For simplicity, we introduce some notations in this section as follows:

FE: the relative error in ∞ -norm, i.e.,

$$FE = \frac{\|x - \tilde{x}\|_{\infty}}{\|x\|_{\infty}},$$

IT: the number of iterations of the iterative method which was applied to solve the new systems,

Cnd: the condition number of the original matrix A with respect to 2-norm,

Cndm: the condition number of the preconditioned matrix $M^{-1}A$ with respect to 2-norm.

For comparison purpose, we choose b such that $Ax = b$ has the exact solution $x = (1, \dots, 1)^T$ for all test problems.

We first apply the new method to Hilbert matrices H_n of different size. These matrices are well-known to be severely ill conditioned. For example, Gaussian elimination with partial pivoting breaks down when $n \geq 12$. We apply the bi-conjugate gradient method to solve the preconditioned system, and choose a tridiagonal preconditioner as it is described in Section 2. Even the matrices are not good test matrices, we can use the results to verify the improvement of the condition number of the new method.

The results for $\delta = 1e - 6$ and different choices of the matrix size n are given in Table 1 with $m = n^2$. We see that the number of iterations is not large when the matrix becomes larger. The error is quite satisfactory taking into account the behavior of the Gaussian elimination. It is not necessary to take $m = n^2$ for this example by our numerical tests.

relative error norm (see previous page)

??

n	10	20	30	40	50
FE	3.30e-4	4.20e-4	5.03e-5	8.29e-6	1.86e-5
IT	11	8	6	7	5
$m=n^2$	100	400	900	1600	2500

Table 1: The error and the number of iterations for the Bi-CG method with H_n when $\delta = 1e-6$

δ	1e-1	1e-2	1e-3	1e-4	1e-5	1e-6
FE	1.86e-5	1.86e-5	1.86e-5	1.86e-5	1.86e-5	1.86e-5
IT	23	18	15	11	9	5

Table 2: The error and the number of iterations for $n = 50$ and different choices of δ with H_n

In Table 2 the case when $n = 50$ is shown for different choices of δ for $m = n^2$. We see that the number of iterations reduces essentially when δ is small. This is because the preconditioner includes entries from more rows of the matrix $A^{(1)}$ for smaller δ . In this way we get a better preconditioner, and the number of iterations is less. Very small values for δ are not recommended because then the preconditioner becomes very ill conditioned.

We then apply the method to matrix $A_n = (a_{ij})$ defined by

$$a_{ij} = \begin{cases} a & \text{if } i \neq j \\ a + 1 & \text{if } i = j \text{ and } i \text{ is odd,} \\ a - 1 & \text{if } i = j \text{ and } i \text{ is even.} \end{cases}$$

off-diagonal terms

diagonal terms

The numerical results with $m \leq 3n$ for $a = 1e7$ are given in Table 3. We also test different δ for $n = 60$ and $a = 1e8$. the results for all $\delta \in (1e-1, 1e-6)$ are $IT = 1$, $m = 180$, $FE = 5.0715e-6$, $Cnd = 2.453e25$ and $Cndm = 4.9062e9$, where the stop criterion is $\|r_k\|/\|b\| < 1e-12$.

The method is also applied for nonsymmetric ill-conditioned matrices

n	10	20	30	40	50
Cnd	9.43e15	3.20e16	5.88e16	4.85e16	1.19e17
Cndm	9.62e7	1.89e8	3.03e8	3.93e8	4.76e8
FE	2.52e-8	8.35e-8	1.65e-7	2.74e-7	3.16e-7
IT	1	1	1	1	1
m	25	55	85	110	130

Table 3: The error and the number of iterations for the Bi-CG method with A_n when $\delta = 1e - 6$

$Y_n = (a_{ij})$ defined by

$$a_{ij} = \begin{cases} 1/(i+j-1) & \text{if } i \leq j, \\ (i-j)^k/(i+j-1), \quad k \geq 5 & \text{if } i > j \end{cases}$$

For example, the condition number of A is 7.2173e14 when $k = 7$ and $n = 20$. The stop criterion of the Bi-CG method is $\|r_k\|/\|b\| \leq \epsilon$. The Bi-CG method without preconditioning obtains the solution with relative error $1.9778e - 6$ under the tolerance $\epsilon = 1e - 11$ in 96 iterations. The Bi-CG method does not converge to the solution if $\epsilon < 1e - 11$. In order to compare with other preconditioners, we consider only the LU preconditioner and incomplete LU preconditioner for the Bi-CG method here. We also compare the Bi-CG method with our preconditioner to the Gaussian Elimination (GE) in Table 4. We use Matlab code to find the LU preconditioner and ILU preconditioner here with drop $1e - 6$. The LU Bi-CG (Bi-CG with a complete LU preconditioner) and ILU Bi-CG (Bi-CG with an incomplete preconditioner) do not converge to the solution after 200 iterations because the preconditioners are very ill-conditioned. The numerical results of our method are given in Table 4. Note that our method is competitive with the Gaussian elimination

for our tested ill-conditioned systems.

We finally tested Riemann system $Ax = b$ where the matrix A is 100×100 Riemann matrix. In this case, the condition number of A is 480.5192. The Gaussian elimination obtains the solution with relative error $4.441e - 16$. The Bi-CG method without preconditioner obtains the approximate solution with tolerance $1e - 12$ at 177 steps. Preconditioned Bi-CG method with LU preconditioner given by Matlab does not obtain the solution after 200 steps because the preconditioner is very ill-conditioned. The Bi-CG method with our preconditioner obtains the solution with tolerance $1e - 12$ at 53 steps where $\delta = 1e - 6$ and $m = 10$. By our experiments, the value of m might not be big for well-conditioned systems.

In fact, our numerical tests illustrated that the value of m can be reduced to cn but not necessary cn^2 where $1 \leq c < n$ is an integer. This means the multiplication of finding the preconditioner is not very big. In general, $m \leq 5n$ is enough for well-conditioned matrices, and m is between $15n$ and $25n$ for ill-conditioned matrices.

All these results illustrated that our method is suitable to find good preconditioner for very ill-conditioned matrices. We do not present numerical tests for sparse matrices because the implementation of our method is much more complicated than the dense case. This is a topic for our further research.

Acknowledgement: The first author likes to give his sincere thanks to Professor Gene H. Golub and Professor Van der Vorst for their very helpful discussions and constructive suggestions on this topic during the Winter School in Hong Kong in 1995. The authors also thank Professor Golub and Professor Plemmons for introducing them to the references [2], [3] and [5]. We would like to thank and Professor Howard and Elman referees for their very helpful suggestions concerning the numerical tests, specially for compar-

n	k	Cnd	Cndm	FE	IT	m	GE
12	6	2.3827e10	2.6023e4	1.062e-8	5	120	2.757e-8
12	6	2.3827e10	7.5857e4	1.374e-8	7	100	
12	6	2.3827e10	2.8434e4	1.539e-8	9	90	
20	7	7.2173e14	3.3646e10	1.647e-6	86	120	1.740e-6
20	7	7.2173e14	1.6210e9	1.225e-6	40	180	
20	7	7.2173e14	2.9267e8	2.927e-5	29	220	
20	7	7.2173e14	1.6592e8	1.480e-6	28	300	
20	7	7.2173e14	2.5634e7	1.479e-6	20	350	
20	7	7.2173e14	1.0920e7	1.479e-6	16	400	

Table 4: The error and the number of iterations for the Bi-CG method with Y_n when $\delta = 1e - 6$

ison with the Gaussian elimination. Their comments and suggestions helped us to improve the quality of the paper.

References

- [1] O. Axelsson, Iterative Solution Methods, Cambridge University Press, 1996.
- [2] A. Berman and R.J. Plemmons, Nonnegative Matrices in the Mathematical Sciences, SIAM, Philadelphia, 1994.
- [3] M. Fiedler and V. Ptak, On matrices with nonpositive off-diagonal elements and positive principal minors, Czech. Math. J. 12(1962) 382-400
- ~~[4]~~ G. H. Golub and C. F. Van Loan, Matrix Computations, John Hopkins University Press, Baltimore, 1996.

- [5] M. Neumann, A note on generalizations of strict diagonal dominance for real matrices, *Linear Algebra and its Applications*, 26(1979) 3-14.
- [6] V. V. Voevodin, *Computational Processes in Linear Algebra*, Nauka, Moscow, 1977. (In Russian)
- [7] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.
- [8] J.Y. Yuan, Preconditioned Diagonal Dominant Matrices, *Applied Mathematics and Computation*, 114(2000) 255-262.

Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods¹

Richard Barrett², Michael Berry³, Tony F. Chan⁴,
James Demmel⁵, June M. Donato⁶, Jack Dongarra^{3,2},
Victor Eijkhout⁷, Roldan Pozo⁸, Charles Romine⁹,
and Henk Van der Vorst¹⁰

This document is the electronic version of the 2nd edition of the Templates book,
which is available for purchase from the Society for Industrial and Applied
Mathematics (<http://www.siam.org/books>).

¹This work was supported in part by DARPA and ARO under contract number DAAL03-91-C-0047, the National Science Foundation Science and Technology Center Cooperative Agreement No. CCR-8809615, the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under Contract DE-AC05-84OR21400, and the Stichting Nationale Computer Faciliteit (NCF) by Grant CRG 92.03.

²Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37830-6173.

³Department of Computer Science, University of Tennessee, Knoxville, TN 37996.

⁴Applied Mathematics Department, University of California, Los Angeles, CA 90024-1555.

⁵Computer Science Division and Mathematics Department, University of California, Berkeley, CA 94720.

⁶Science Applications International Corporation, Oak Ridge, TN 37831

⁷Texas Advanced Computing Center, The University of Texas at Austin, Austin, TX 78758

⁸National Institute of Standards and Technology, Gaithersburg, MD

⁹Office of Science and Technology Policy, Executive Office of the President

¹⁰Department of Mathematics, Utrecht University, Utrecht, the Netherlands.

5. Conjugate Gradient (CG)
6. Minimal Residual (MINRES) and Symmetric LQ (SYMMLQ)
7. Conjugate Gradients on the Normal Equations (CGNE and CGNR)
8. Generalized Minimal Residual (GMRES)
9. Biconjugate Gradient (BiCG)
10. Quasi-Minimal Residual (QMR)
11. Conjugate Gradient Squared (CGS)
12. Biconjugate Gradient Stabilized (Bi-CGSTAB)
13. Chebyshev Iteration

For each method we present a general description, including a discussion of the history of the method and numerous references to the literature. We also give the mathematical conditions for selecting a given method.

We do not intend to write a "cookbook", and have deliberately avoided the words "numerical recipes", because these phrases imply that our algorithms can be used blindly without knowledge of the system of equations. The state of the art in iterative methods does not permit this: some knowledge about the linear system is needed to guarantee convergence of these algorithms, and generally the more that is known the more the algorithm can be tuned. Thus, we have chosen to present an algorithmic outline, with guidelines for choosing a method and implementing it on particular kinds of high-performance machines. We also discuss the use of preconditioners and relevant data storage issues.

A^*, b^*, M

$\textcircled{A}^* y = b^*$

$M^T A$

$x^{(0)}$ is an initial guess

for $j = 1, 2, \dots$

Solve r from $Mr = b - Ax^{(0)}$

$v^{(1)} = r / \|r\|_2$

$s := \|r\|_2 e_1$

for $i = 1, 2, \dots, m$

Solve w from $Mw = Av^{(i)}$

for $k = 1, \dots, i$

$h_{k,i} = (w, v^{(k)})$

$w = w - h_{k,i} v^{(k)}$

end

$h_{i+1,i} = \|w\|_2$

$v^{(i+1)} = w / h_{i+1,i}$

apply J_1, \dots, J_{i-1} on $(h_{1,i}, \dots, h_{i+1,i})$

construct J_i , acting on i th and $(i+1)$ st component of $h_{\cdot,i}$, such that $(i+1)$ st component of $J_i h_{\cdot,i}$ is 0

$s := J_i s$

if $s(i+1)$ is small enough then (UPDATE(\tilde{x}, i) and quit)

end

UPDATE(\tilde{x}, m)

end

In this scheme UPDATE(\tilde{x}, i)

replaces the following computations:

Compute y as the solution of $Hy = \tilde{s}$, in which the upper $i \times i$ triangular part of H has $h_{i,j}$ as its elements (in least squares sense if H is singular), \tilde{s} represents the first i components of s

$\tilde{x} = x^{(0)} + y_1 v^{(1)} + y_2 v^{(2)} + \dots + y_i v^{(i)}$

$s^{(i+1)} = \|b - A\tilde{x}\|_2$

if \tilde{x} is an accurate enough approximation then quit

else $x^{(0)} = \tilde{x}$

Figure 2.6: The Preconditioned GMRES(m) Method

preferred, giving up some stability for better parallelization properties (see Demmel, Heath and Van der Vorst [66]). Here we adopt the Modified Gram-Schmidt approach.

The major drawback to GMRES is that the amount of work and storage required per iteration rises linearly with the iteration count. Unless one is fortunate enough to obtain extremely fast convergence, the cost will rapidly become prohibitive. The usual way to overcome this limitation is by restarting the iteration. After a chosen number (m) of iterations, the accumulated data are cleared and the intermediate results are used as the initial data for the next m iterations. This procedure is repeated until convergence is achieved. The difficulty is in choosing an appropriate value for m . If m is "too small", GMRES(m) may be slow to converge, or fail to converge entirely. A value of m that is larger than necessary involves excessive work (and uses more storage). Unfortunately, there are no definite rules governing the choice of m —choosing when to restart is a matter of experience.

For a discussion of GMRES for vector and shared memory computers see Dongarra *et al.* [70];