AISM: A factorized sparse approximate inverse preconditioner [1]

R. Bru J. Cerdán J. Marín J. Mas

Departament de Matemàtica Aplicada
Universitat Politècnica de València
{rbru,jcerdan,jmarinma,jmasm}@mat.upv.es

[1]Supported by spanish DGI grant BFM2001-2641.

Outline

- Introduction
- Factorization of $A_0^{-1} - A^{-1}$ using the Sherman-Morrison formula
- Approximate Inverse Preconditioner
- Numerical experiments
- Conclusions and future work

Introduction

Problem: solve $Ax = b$ using a preconditioned Krylov method.

Left preconditioned linear system

$$MAx = Mb\ ,$$

where $M \approx A^{-1}$.

Other approximate inverse preconditioners:

- Non Factorized: SPAI (Grothe and Huckle).
- Factorized: AINV (Benzi, Meyer and Tuma), FSAI (Kolotilina and Yeremin).

Purpose: A factorized approximate inverse preconditioner using the Sherman-Morrison formula.

Uses of the Sherman-Morrison formula

- Update linear models by using least-squares in statistics
- In networks and structures analysis to compute a new solution when the system is modified
- To update a factorization of a matrix
- To invert a matrix
- $\dots$

We show how the matrix $A_0^{-1} - A^{-1}$ is factored in the form $U\Omega V^T$.

Factorization of $A_0^{-1} - A^{-1}$ using the Sherman-Morrison formula

B $n \times n$ nonsingular matrix, $x, y \in \mathbf{R}^n$: $r = 1 + y^T B^{-1} x \neq 0$, then $A = B + xy^T$ is nonsingular, and $A^{-1} = B^{-1} - r^{-1} B^{-1} x y^T B^{-1}$.

To compute A^{-1}:

- A_0 nonsingular $n \times n$ easy to invert.
- $\{x_k\}_{k=1}^n, \{y_k\}_{k=1}^n$ two sets of $\mathbf{R}^n$ such that:

$$A = A_0 + \sum_{k=1}^{n} x_k y_k^T \,. \tag{1}$$

- $A_k := A_0 + \sum_{i=1}^k x_i y_i^T = A_{k-1} + x_k y_k^T$, $k = 1, \ldots, n$. (Note $A_n = A$).
- $r_k = 1 + y_k^T A_{k-1}^{-1} x_k \neq 0$

$$A_k^{-1} = A_{k-1}^{-1} - \frac{1}{r_k} A_{k-1}^{-1} x_k y_k^T A_{k-1}^{-1} \quad k = 1, \ldots, n. \quad (A_n^{-1} = A^{-1})$$

This process can be written as follows

$$A^{-1} = A_0^{-1} - \sum_{k=1}^{n} \frac{1}{r_k} A_{k-1}^{-1} x_k y_k^T A_{k-1}^{-1}$$

or

$$A_0^{-1} - A^{-1} = \sum_{k=1}^{n} \frac{1}{r_k} A_{k-1}^{-1} x_k \,. y_k^T A_{k-1}^{-1}$$

In matrix notation

$$A_0^{-1} - A^{-1} = \Phi \Omega^{-1} \Psi^T \,,$$

where

$$\Phi = \left[\begin{array}{cccc} A_0^{-1} x_1 & A_1^{-1} x_2 & \cdots & A_{n-1}^{-1} x_n \end{array} \right],$$

$$\Omega^{-1} = \left[\begin{array}{cccc} r_1^{-1} & & & \\ & r_2^{-1} & & \\ & & \ddots & \\ & & & r_n^{-1} \end{array} \right], \quad \text{and} \quad \Psi^T = \left[\begin{array}{c} y_1^T A_0^{-1} \\ y_2^T A_1^{-1} \\ \vdots \\ y_n^T A_{n-1}^{-1} \end{array} \right].$$

This process can be computed without the matrices $\{A_k\}_{k=1}^n$ explicitly appearing.

- $u_1 := x_1, v_1 := y_1$.

$$\begin{aligned} A_1^{-1} x_2 &= (A_0^{-1} - \tfrac{1}{r_1} A_0^{-1} x_1 y_1^T A_0^{-1}) x_2 \\ &= A_0^{-1} x_2 - \tfrac{y_1^T A_0^{-1} x_2}{r_1} A_0^{-1} x_1 \\ &= A_0^{-1} (x_2 - \tfrac{y_1^T A_0^{-1} x_2}{r_1} x_1) = A_0^{-1} u_2 \,. \end{aligned}$$

$$\begin{aligned} y_2^T A_1^{-1} &= y_2^T (A_0^{-1} - \tfrac{1}{r_1} A_0^{-1} x_1 y_1^T A_0^{-1}) \\ &= y_2^T A_0^{-1} - \tfrac{y_2^T A_0^{-1} x_1}{r_1} y_1^T A_0^{-1} \\ &= (y_2^T - \tfrac{y_2^T A_0^{-1} x_1}{r_1} y_1^T) A_0^{-1} = v_2^T A_0^{-1} \,. \end{aligned}$$

- $u_2 := x_2 - \frac{y_1^T A_0^{-1} x_2}{r_1} x_1 = x_2 - \frac{v_1^T A_0^{-1} x_2}{r_1} u_1$.

- $v_2 := y_2 - \frac{y_2^T A_0^{-1} x_1}{r_1} y_1 = y_2 - \frac{y_2^T A_0^{-1} u_1}{r_1} v_1$.

Theorem 1 *Let A and A_0 be two nonsingular matrices, and let $\{x_k\}_{k=1}^n$ and $\{y_k\}_{k=1}^n$ be two set of vectors such that $A = A_0 + \sum_{k=1}^n x_k y_k^T$. Moreover, suppose that $r_k = 1 + y_k^T A_{k-1}^{-1} x_k \neq 0$ for $k = 1, \ldots, n$, where $A_{k-1} = A_0 + \sum_{i=1}^{k-1} x_i y_i^T$. Then*

$$u_k := x_k - \sum_{i=1}^{k-1} \frac{v_i^T A_0^{-1} x_k}{r_i} u_i \tag{2}$$

and

$$v_k := y_k - \sum_{i=1}^{k-1} \frac{y_k^T A_0^{-1} u_i}{r_i} v_i \tag{3}$$

are well defined for $k = 1, \ldots, n$. In addition, the relations

$$\begin{aligned} A_{k-1}^{-1} x_k &= A_0^{-1} u_k \\ y_k^T A_{k-1}^{-1} &= v_k^T A_0^{-1} \end{aligned}$$

and $r_k = 1 + y_k^T A_0^{-1} u_k = 1 + v_k^T A_0^{-1} x_k$ are satisfied.

Corollary 1

$$U := \begin{bmatrix} u_1 & u_2 & \cdots & u_n \end{bmatrix} \quad \textit{and} \quad V := \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix},$$

$$X := \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \quad \textit{and} \quad Y := \begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix}.$$

Then

$$A_0^{-1} - A^{-1} = A_0^{-1} U \Omega^{-1} V^T A_0^{-1} \tag{4}$$

where $\Omega = \mathrm{diag}(r_1, r_2, \ldots, r_n)$. *Moreover,*

$$X = UT_X \qquad , \qquad Y = VT_Y$$

where

$$T_X = \begin{bmatrix} 1 & t_{12} & & & t_{1n} \\ 0 & 1 & & & t_{2n} \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & & 1 & t_{n-1,n} \\ 0 & 0 & \cdots & & 1 \end{bmatrix}, \qquad t_{ij} = \frac{v_i^T A_0^{-1} x_j}{r_i}, \quad i < j,$$

and

$$T_Y = \begin{bmatrix} 1 & \bar{t}_{12} & & & \bar{t}_{1n} \\ 0 & 1 & & & \bar{t}_{2n} \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & & 1 & \bar{t}_{n-1,n} \\ 0 & 0 & \cdots & & 1 \end{bmatrix}, \qquad \bar{t}_{ij} = \frac{y_j^T A_0^{-1} u_i}{r_i}, \quad i < j,$$

Approximate Inverse Preconditioner

Choice of x_k, y_k and A_0

$$\begin{aligned} &A_0 = sI_n,\ s > 0 \\ &x_k = e_k, && X = I_n \\ &y_k = (a^k - (a_0)^k)^T, && Y = (A - sI_n)^T \end{aligned}$$

a^k denotes the kth row A, e_k kth of I_n. From (4)

$$A_0^{-1} U \Omega^{-1} V^T A_0^{-1} = A_0^{-1} - A^{-1}$$

it follows

$$s^{-1} U \Omega^{-1} V^T s^{-1} = s^{-1} I_n - A^{-1}. \tag{5}$$

Note:

- $X = I_n \Longrightarrow U = T_X^{-1}$ is a nonsingular unit upper triangular matrix.
- $V = Y T_Y^{-1}$ nonsingular $\Longleftrightarrow s \notin \sigma(A)$.

Factorization Algorithm

(1) **Set** $x_k = e_k$**,** $y_k = (a^k)^T - s\, e_k$**,** $\quad (k = 1, \ldots, n)$

(2) **for** $k = 1, \ldots, n$

$u_k = x_k$

$v_k = y_k$

for $i = 1, \ldots, k-1$

$$u_k = u_k - \frac{(v_i)_k}{s\, r_i} u_i$$

$$v_k = v_k - \frac{y_k^T u_i}{s\, r_i} v_i$$

end for

$r_k = 1 + (v_k)_k / s$

end for

(3) **Return** $U = [u_1,\, u_2,\, \cdots\, ,\, u_n]$**,** $V = [v_1,\, v_2,\, \cdots\, ,\, v_n]$ **and** $\Omega = \mathrm{diag}(r_1,\, r_2,\, \ldots,\, r_n)$**.**

Approximate Inverse Preconditioner (II)

Existence for M-matrices

Theorem 2 *Let* u_k*,* v_k *and* r_k *be the vectors and pivots computed by the complete factorization algorithm, and let* u_k^**,* v_k^* *and* r_k^* *be the vectors and pivots computed by the complete factorization algorithm for* $s = 1$*, respectively. Then,*

$$u_k \;=\; u_k^* \tag{6}$$

$$v_k \;=\; v_k^* - (s-1)w_k \quad \text{where} \quad w_k = x_k - \sum_{i=1}^{k-1} \frac{(y_i^*)^T u_k^*}{r_i^*} w_i \tag{7}$$

$$r_k \;=\; r_k^*/s \tag{8}$$

Lemma 1 *Let* A *be a nonsingular* M*-matrix. Then, the pivots* r_k *computed by the complete factorization Algorithm are positive.*

Theorem 3 *Let* A *be a nonsingular* M*-matrix. The pivots* r_k *computed by the complete factorization Algorithm and the pivots* $\bar{r}_k$ *computed by the incomplete factorization algorithm verify* $\bar{r}_k \geq r_k > 0$.

Preconditioning possibilities

Applying a dropping rule to U and V factors:

$$s^{-1}\bar{U}\bar{\Omega}^{-1}\bar{V}^T s^{-1} \approx s^{-1}I_n - A^{-1}, \tag{9}$$

1. $M_1 = s^{-1}I_n - s^{-2}\bar{U}\bar{\Omega}^{-1}\bar{V}^T \approx A^{-1}$

$$\sigma(M_1 A) \approx \sigma(I_n)$$

2. $M_2 = s^{-2}\bar{U}\bar{\Omega}^{-1}\bar{V}^T \approx s^{-1}I_n - A^{-1}$

$$\sigma(M_2 A) \approx \frac{\sigma(A)}{s} - 1$$

3. $M_3 = \bar{U}\bar{\Omega}^{-1}\bar{V}^T \approx s^2(s^{-1}I_n - A^{-1})$

$$\sigma(M_3 A) \approx s^2\left(\frac{\sigma(A)}{s} - 1\right)$$

Relations between preconditioners

$$\begin{aligned} M_2 &= s^{-1}I_n - M_1, \\ M_3 &= s^2 M_2 = sI_n - s^2 M_1. \end{aligned} \tag{10}$$

In practice $s > \rho(A) \Rightarrow \sigma(M_2 A), \sigma(M_3 A)$ are contained in the left half complex plane.

Numerical Results

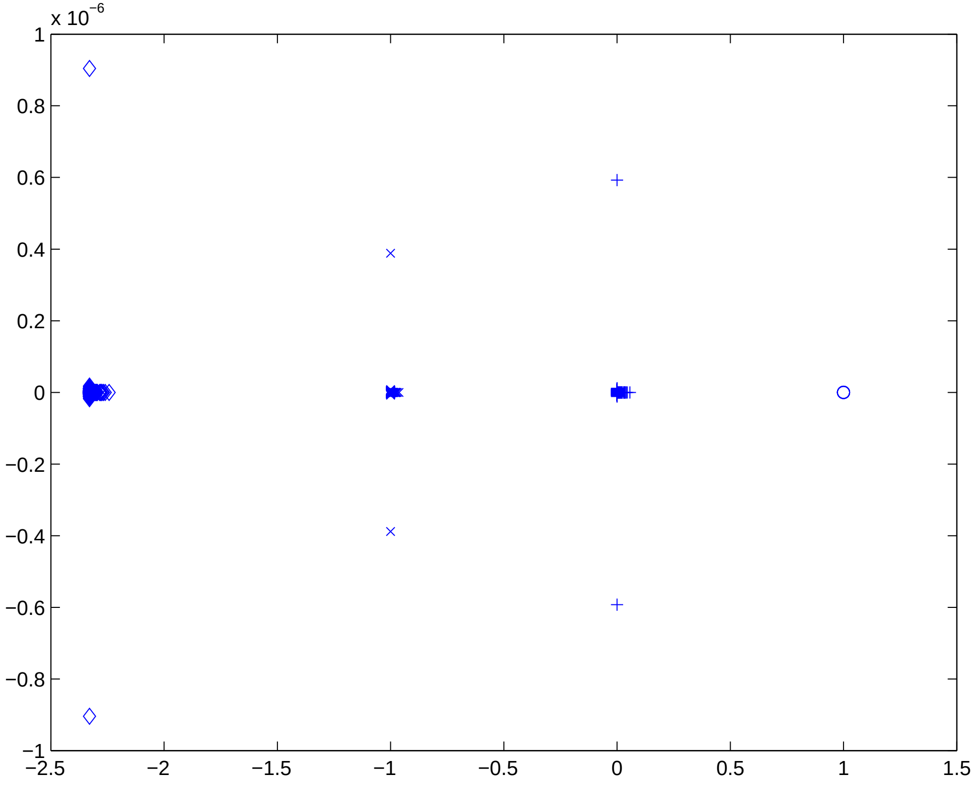
- Fortran and MatLab code
- IFC (Intel Fortran Compiler) on a Linux Pentium III
- BiCGSTAB method.
- Matrices rescaled by $\max |a_{ij}|$
- Right-hand side $b = A(1, 1, \ldots, 1)^T$
- Error test $\|r_k\| \leq 10^{-8}\|r_0\|$
- Comparison with AINV preconditioner (left-looking version)
- Test matrices from Harwell Boeing and Tim Davis' collection

Test matrices

Matrix	n	nnz	Description
ADD20	2395	17319	Circuit simulation
FS5414	541	4285	Chemical kinetics
HOR131	434	4710	Network flow
ORSIRR1	1030	6858	Reservoir simulation
ORSIRR2	886	5970	Reservoir simulation
ORSREG1	2205	14133	Reservoir simulation
SAYLR3	1000	3750	Reservoir simulation
SAYLR4	3564	22316	Reservoir simulation
SHERMAN1	1000	3750	Reservoir simulation
WATT1	1856	11360	Petroleum engineering
WATT2	1865	11550	Petroleum engineering

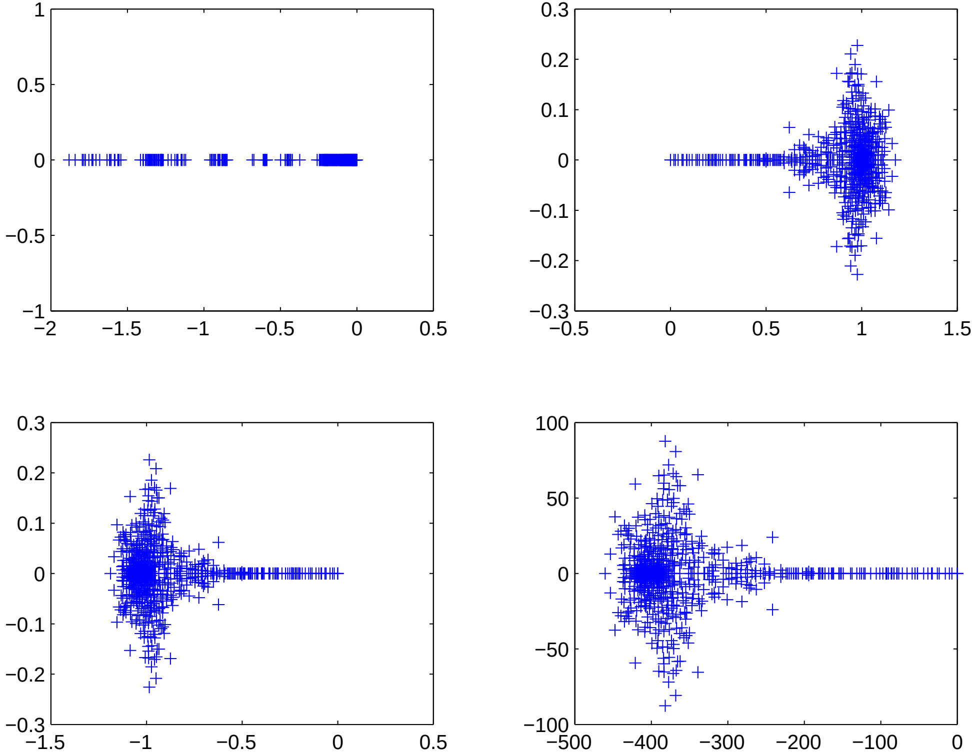
Comparative study of M_1, M_2 and M_3

Matrix FS5414. $\sigma(A)$ - $+$, $\sigma(M_1 A)$ - $\circ$, $\sigma(M_2 A)$ -$\times$, $\sigma(M_3 A)$ - $\diamond$



Comparative study of M_1, M_2 and M_3

Matrix SAYLR3, $s = 10\|A\|_\infty$, Tol=0.1



Experimental results for the matrix FS5414.

Tol	$s/\|A\|_\infty$	nnz($\bar{U}$)	nnz($\bar{V}$)	nnz	iter. M_1	iter. M_2	iter. M_3
0.1	1	547	708	1256	448	55	43
0.1	1.5	547	751	1298	†	107	42
0.1	2	547	768	1315	279	111	41
0.1	5	547	871	1418	243	182	39
0.1	10	547	1047	1594	239	292	39
0.1	100	547	1589	2136	206	209	38
0.0001	1	1418	3420	4838	44	20	16
0.0001	1.5	1418	3584	5002	50	31	18
0.00001	1	2598	5520	8118	9	5	5
0.00001	1.5	2598	5707	8305	8	6	5

Experimental results for the matrix SAYLR3.

Tol	$s/\|A\|_\infty$	nnz($\bar{U}$)	nnz($\bar{V}$)	nnz	iter. M_1	iter. M_2	iter. M_3
0.1	1	1314	5066	6380	205	105	39
0.1	1.5	1314	5968	7282	174	118	34
0.1	2	1314	6695	8009	198	76	35
0.1	10	1314	11041	12355	158	148	32

- Preconditioning with M_3 (AISM) is better.
- $s \approx 1.5\ \|A\|_\infty$ good choice.
- Larger s implies:
  1. Less iterations to converge.
  2. More fill-in on $\bar{V}$.
  3. Smaller pivots (breakdown).

Comparative study with AINV

Matrix	No Prec.		AISM ($s/\|A\|_\infty = 1.5$)				AINV(0.1)		
	Iter.	Time	Tol.	nnz	Iter.	Time	nnz	Iter.	Time
ADD20	315	1.56	0.01	14880	7	0.09	9990	7	0.05
FS5414	863	0.56	0.1	1298	45	0.03	4104	81	0.07
HOR131	†		0.05	6077	38	0.06	8129	26	0.04
ORSIRR1	†		0.01	11668	35	0.12	6381	38	0.07
ORSIRR2	1196	1.39	0.01	11154	34	0.09	5488	40	0.06
ORSREG1	304	1.17	0.1	17101	41	0.4	13230	40	0.36
SAYLR3	374	0.37	0.1	7282	36	0.10	6690	32	0.06
SAYLR4	†		0.5	54253	46	1.19	51926	45	1.18
SHERMAN1	†		0.1	7282	36	0.09	6690	32	0.08
WATT1	†		0.1	8829	2	0.02	14807	1	0.02
WATT2	†		0.1	12075	56	0.37	15488	115	0.97

- AISM performs similarly to AINV
- Both methods are robust

Conclusions and future work

Conclusions

- AISM shifts the spectrum of A to the left half complex plane.
- AISM is breakdown-free for M-matrices (and possibly for H-matrices).
- AISM shows good performance with very sparse factors.
- AISM performs similar to AINV for the tested problems.
- $s =$1.5 $\|A\|_\infty$, and tol=0.1 or 0.01 are good choices

Future work

- AISM for symmetric matrices.
- Parallel computation of AISM.
- Effect of reorderings.
- Block version of AISM (Sherman-Morrison-Woodbury formula).
- ...