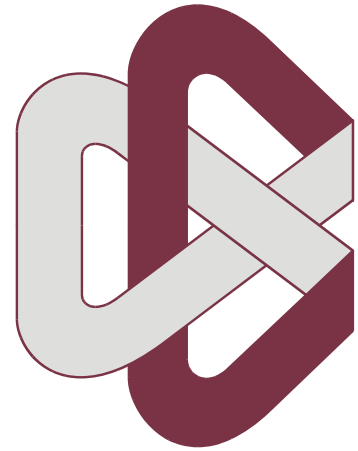


CIMAT

Precondicionamiento con inversas aproximadas

Miguel Vargas-Félix

miguelvargas@cimat.mx
http://www.cimat.mx/~miguelvargas

Número de condición

El número de condición κ de una matriz $\mathbf{A}$ no singular, para una norma $\|\cdot\|$ está dado por

$$\kappa(\mathbf{A}) = \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|.$$

Para la norma $\|\cdot\|_2$,

$$\kappa(\mathbf{A}) = \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\| = \frac{\sigma_{max}(\mathbf{A})}{\sigma_{min}(\mathbf{A})},$$

donde σ son los valores singulares de la matriz.

Para una matriz $\mathbf{A}$ simétrica positiva definida,

$$\kappa(\mathbf{A}) = \frac{\lambda_{max}(\mathbf{A})}{\lambda_{min}(\mathbf{A})},$$

donde λ son los eigenvalores de $\mathbf{A}$.

El **número de condición** indica que tan sensible es una función a pequeños cambios en la entrada.

Así, matrices con un número de condición cercano a 1 se dicen que están bien condicionadas.

Un sistema de ecuaciones $\mathbf{A}\,\mathbf{x} = \mathbf{b}$ es considerado

bien condicionado si un pequeño cambio en los valores de $\mathbf{A}$
o un pequeño cambio en $\mathbf{b}$ resulta en un pequeño cambio en $\mathbf{x}$.

Un sistema de ecuaciones $\mathbf{A}\,\mathbf{x} = \mathbf{b}$ es considerado

mal condicionado si un pequeño cambio en los valores de $\mathbf{A}$
o un pequeño cambio en $\mathbf{b}$ resulta en un cambio grande en $\mathbf{x}$.

Al reducir el número de condición de un sistema, se puede incrementar la velocidad de convergencia del gradiente conjugado.

Gradiente conjugado precondicionado

Entonces, en vez de resolver el problema

$$\mathbf{A}\,\mathbf{x}-\mathbf{b}=0,$$

se resuelve el problema

$$\mathbf{M}^{-1}\left(\mathbf{A}\,\mathbf{x}-\mathbf{b}\right)=0,$$

con $\mathbf{M}^{-1}$ una matriz cuadrada, la cual recibe el nombre de **precondicionador**.

El mejor precondicionador sería claro $\mathbf{M}^{-1}=\mathbf{A}^{-1}$, así $\mathbf{x}=\mathbf{M}^{-1}\mathbf{b}$, y el gradiente conjugado convergiría en un paso.

Al igual que la matriz $\mathbf{A}$, el precondicionador $\mathbf{M}^{-1}$ tiene que ser simétrico positivo definido.

Hay dos tipos de precondicionadores, implícitos $\mathbf{M}$ y explícitos $\mathbf{M}^{-1}$.

En algunos casos es costoso calcular $\mathbf{M}^{-1}$, en general se utilizan precondicionadores con inversas fáciles de calcular o precondicionadores implicitos que se puedan factorizar.

El algoritmo es el siguiente:

entrada: $\mathbf{A}$, $\mathbf{x}_0$, $\mathbf{b}$,ε

$\mathbf{r}_0 \leftarrow \mathbf{A}\mathbf{x}_0 - \mathbf{b}$

$\mathbf{q}_0 \leftarrow \mathbf{M}^{-1}\mathbf{r}_0$

$\mathbf{p}_0 \leftarrow -\mathbf{q}_0$

$k \leftarrow 0$

mientras $\left\|\mathbf{r}_k\right\| > \varepsilon$

$\mathbf{w} \leftarrow \mathbf{A}\mathbf{p}_k$

$\alpha_k \leftarrow \dfrac{\mathbf{r}_k^{\mathrm{T}}\mathbf{q}_k}{\mathbf{p}_k^{\mathrm{T}}\mathbf{w}}$

$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{p}_k$

$\mathbf{r}_{k+1} \leftarrow \mathbf{r}_k + \alpha \mathbf{w}$

$\mathbf{q}_{k+1} \leftarrow \mathbf{M}^{-1}\mathbf{r}_{k+1}$, o resolver $\mathbf{M}\,\mathbf{q}_{k+1} \leftarrow \mathbf{r}_{k+1}$

$\beta_k \leftarrow \dfrac{\mathbf{r}_{k+1}^{\mathrm{T}}\mathbf{q}_{k+1}}{\mathbf{r}_k^{\mathrm{T}}\mathbf{q}_k}$

$\mathbf{p}_{k+1} \leftarrow -\mathbf{q}_{k+1} + \beta_{k+1}\mathbf{p}_k$

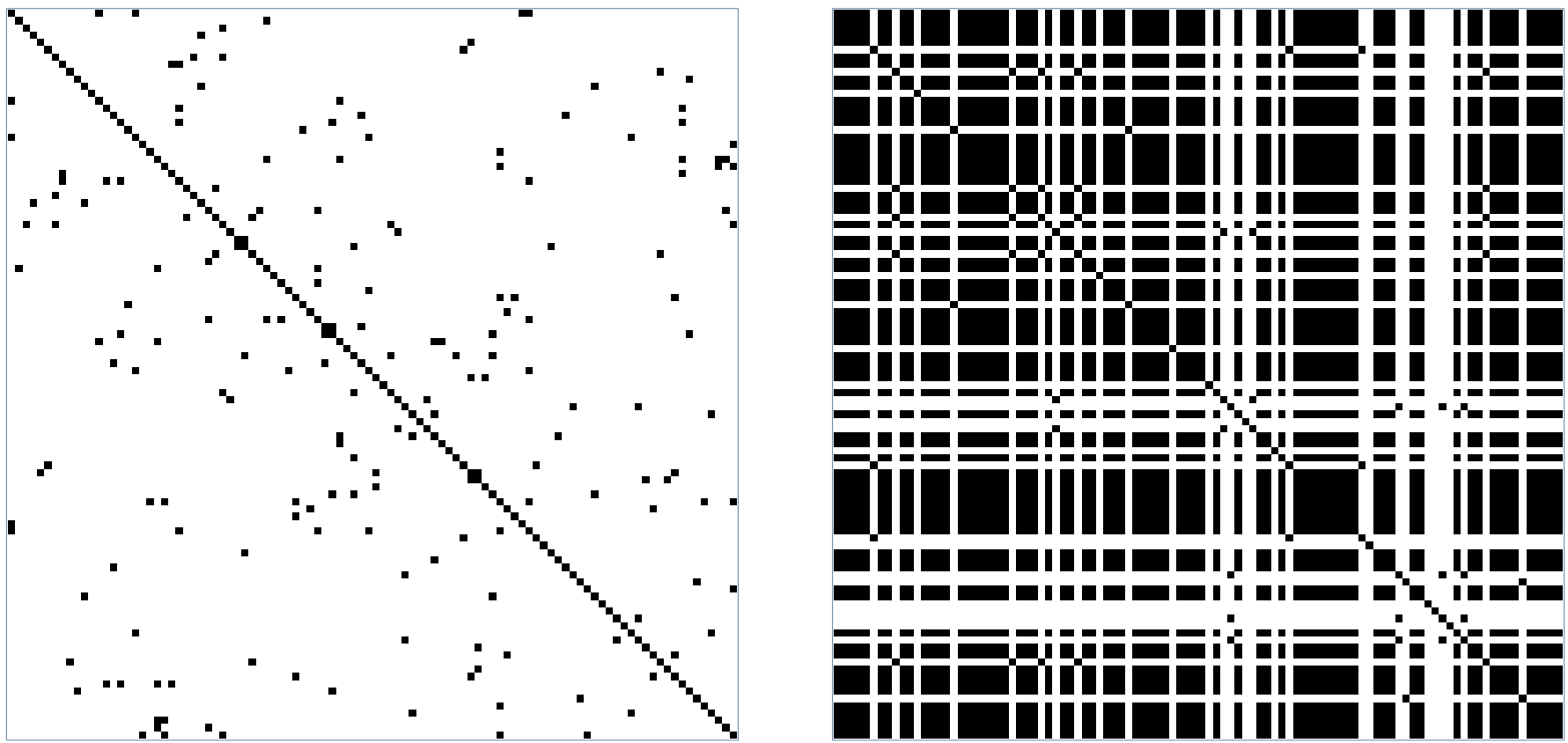
$k \leftarrow k+1$

Nótese que ahora el algoritmo require aplicar el precondicionador en cada paso.

Precondicionadores con inversa aproximada dispersa

Vamos a describir algorimos que permitan generar una matriz $\mathbf{M}^{-1}$ se sea una aproximación a la inversa de $\mathbf{A}$, que además tenga la propiedad de ser dispersa.

El problema es que a partir de una matriz dispersa su inversa no es necesariamente dispersa.



Ejemplo de las estructuras de una matriz dispersa y de su inversa

Una forma de buscar la inversa aproximada se basa en **minimizar** la norma de Frobenius del residuo $\mathbf{I}-\mathbf{A}\,\mathbf{M}^{-1}$

$$F\left(\mathbf{M}^{-1}\right)=\left\|\mathbf{I}-\mathbf{A}\,\mathbf{M}^{-1}\right\|_{\mathrm{F}}^{2} \tag{1}$$

La norma de Frobenius se define como

$$\|\mathbf{A}\|_{\mathrm{F}}=\sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}\left|a_{ij}\right|^{2}}=\sqrt{\mathrm{tr}\left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)}.$$

Podemos descomponer (1) en sumas desacolpladas de las 2-normas de las columnas individuales del residuo $\mathbf{I}-\mathbf{A}\,\mathbf{M}^{-1}$, es decir

$$F\left(\mathbf{M}^{-1}\right)=\left\|\mathbf{I}-\mathbf{A}\,\mathbf{M}^{-1}\right\|_{\mathrm{F}}^{2}=\sum_{j=1}^{n}\left\|\mathbf{e}_{j}-\mathbf{A}\,\mathbf{m}_{j}\right\|_{2}^{2},$$

donde $\mathbf{e}_j$ es la j-ésima columna de $\mathbf{I}$ y $\mathbf{m}_j$ es la j-ésima columna de $\mathbf{M}^{-1}$.

Podemos así paralelizar la construcción del precondicionador.

El problema ahora es: ¿como elegir las entradas de $\mathbf{M}^{-1}$ que son significativas para construir la inversa aproximada?

Inversa aproximada dispersa factorizada

Los precondicionadores para el gradiente conjugado requiere que al igual que la matriz $\mathbf{A}$, el precondicionador $\mathbf{M}^{-1}$ sea simétrico positivo definido.

El siguiente método [Chow01] genera un precondicionador que cumple que $\mathbf{M}^{-1}$ sea positivo definido y además es posible calcularlo en paralelo.

Sea $\tilde{\mathbf{A}}$ que una matriz binaria construida a partir de $\mathbf{A}$

$$\tilde{A}_{ij} = \begin{cases} 1 & \text{si } i = j \text{ o } \left| \left(\mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2} \right)_{ij} \right| > umbral \\ 0 & \text{otro caso} \end{cases},$$

el umbral es un valor no negativo y la matriz diagonal $\mathbf{D}$ es

$$D_{ii} = \begin{cases} |A_{ii}| & \text{si } |A_{ii}| > 0 \\ 1 & \text{otro caso} \end{cases}.$$

La matriz diagonal $\mathbf{D}$ funciona como un escalamiento.

Notese que la estructura de $\tilde{\mathbf{A}}$ es más dispersa que la estructura de $\mathbf{A}$.

Buscamos construir un precondicionador

$$\mathbf{M}^{-1} = \mathbf{G}^{\mathrm{T}} \mathbf{G},$$

donde $\mathbf{G}$ es una matriz triangular inferior tal que

$$\mathbf{G} \approx \mathbf{L}^{-1},$$

donde $\mathbf{L}$ es el factor Cholesky de $\mathbf{A}$.

Sea $\boldsymbol{S}_L$ la estructura de $\mathbf{G}$, ésta se construirá de tal forma que tenga la misma estructura que la parte triangular inferior de $\tilde{\mathbf{A}}$.

Buscamos entonces minimizar $\left\| \mathbf{I} - \mathbf{G}\,\mathbf{L} \right\|_{\mathrm{F}}^2$, lo que puede lograrse sin conocer el factor $\mathbf{L}$, resolviendo las ecuaciones

$$\left(\mathbf{G}\,\mathbf{L}\,\mathbf{L}^{\mathrm{T}} \right)_{i\,j} = \left(\mathbf{L}^{\mathrm{T}} \right)_{i\,j}, \left(i\,,\, j \right) \in \boldsymbol{S}_L, \tag{2}$$

donde $\boldsymbol{S}_L$ contiene la estructura de $\mathbf{G}$.

Sea $\widetilde{\mathbf{D}}$ la diagonal de $\mathbf{L}$, y sea $\widetilde{\mathbf{G}} = \widetilde{\mathbf{D}}^{-1} \mathbf{G}$. Multiplicando (2) por $\widetilde{\mathbf{D}}^{-1}$, tenemos

$$\left(\widetilde{\mathbf{G}}\,\mathbf{A} \right)_{i\,j} = \left(\boldsymbol{I} \right)_{i\,j}, \left(i\,,\, j \right) \in \boldsymbol{S}_L.$$

Las entradas de $\tilde{\mathbf{G}}$ se calculan por renglones (tiene la misma estructura que $\mathbf{G}$). Éstas se pueden calcular en paralelo.

El algoritmo para calcular las entradas de $\tilde{\mathbf{G}}$ es el siguiente:

Sea $\boldsymbol{S}_L$ la estructura de $\boldsymbol{G}$
para $i \leftarrow 1 \ldots n$
  para $\forall (i, j) \in \boldsymbol{S}_L$
    resolver $\left(\boldsymbol{A}\tilde{\boldsymbol{G}}_i\right)_j = \delta_{ij}$
  fin_para
fin_para

Resolver $\left(\mathbf{A}\tilde{\mathbf{G}}_i\right)_j = \delta_{ij}$ significa que, si $m = \eta\left(\tilde{\mathbf{G}}_i\right)$ el número de entradas no cero del renglón i de $\tilde{\mathbf{G}}$, entonces hay que resolver un sistema SPD local pequeño de tamaño $m \times m$.

$$\left(\mathbf{A}\tilde{\mathbf{G}}_i\right)_j = \delta_{ij}, (i, j) \in \boldsymbol{S}_L,$$

Ejemplo:

Si la estructura $(i, j) \in \boldsymbol{S}_L$ del renglón 9 de $\tilde{\mathbf{G}}$ contiene 4, 7, 8, 9, entonces, para calcular los valores de $\tilde{\mathbf{G}}_9$ hay que resolver el sistema:

$$\begin{pmatrix} a_{44} & a_{47} & a_{48} & a_{49} \\ a_{74} & a_{77} & a_{78} & a_{79} \\ a_{84} & a_{87} & a_{88} & a_{89} \\ a_{94} & a_{97} & a_{98} & a_{99} \end{pmatrix} \begin{pmatrix} \tilde{g}_{94} \\ \tilde{g}_{97} \\ \tilde{g}_{98} \\ \tilde{g}_{99} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

Como es un sistema pequeño y simétrico positivo definido, podemos resolver estos sistemas utilizando factorización Cholesky.

Es posible eficientar la solución de estos sistemas haciendo casos particulares para matrices de 1×1, 2×2, hasta 3×3, y el caso general para matrices más grandes.

Finalmente, para obtener $\mathbf{G}$, $\mathbf{G} = \tilde{\mathbf{G}}\,\tilde{\mathbf{D}}$,

(no confundir esta matriz diagonal $\tilde{\mathbf{D}}$ con la utilizada para calcular la estructura de $\tilde{\mathbf{A}}$), su valor se detemina como

$$\tilde{\mathbf{D}} = \operatorname{diag}\left(\tilde{\mathbf{G}}\right)^{-1/2},$$

es decir,

$$\tilde{D}_{ii} = \frac{1}{\sqrt{\tilde{G}_{ii}}}.$$

La secuencia del cálculo de las matrices es:

A

D

$$D_{ii} = \begin{cases} |A_{ii}| & \text{si } |A_{ii}| > 0 \\ 1 & \text{otro caso} \end{cases}$$

$\tilde{\mathbf{A}}$

$$\tilde{A}_{ij} = \begin{cases} 1 & \text{si } i = j \text{ o } \left|\left(\mathbf{D}^{-1/2}\,\mathbf{A}\,\mathbf{D}^{-1/2}\right)_{ij}\right| > umbral \\ 0 & \text{otro caso} \end{cases}$$

$\tilde{\mathbf{G}}$

$$\left(\mathbf{A}\,\tilde{\mathbf{G}}_i\right)_j = \delta_{ij}$$

Su estructura es la parte triangular inferior de $\tilde{\mathbf{A}}$

$\tilde{\mathbf{D}}$

$$\tilde{D}_{ii} = \frac{1}{\sqrt{\tilde{G}_{ii}}}$$

G

$$\mathbf{G} = \tilde{\mathbf{G}}\,\tilde{\mathbf{D}}$$

Este precondicionador $\mathbf{M}^{-1}=\mathbf{G}^{\mathrm{T}}\mathbf{G}$ se puede aplicar entonces al gradiente conjugado precondicionado:

- Este precondicionador es simétrico positivo definido si no hay ceros en la diagonal de $\mathbf{G}_l$.
- El algoritmo para construir el precondicionador es paralelizable.
- El algoritmo es estable si $\mathbf{A}$ es simétrica positiva definida.

E. Chow. Parallel implementation and practical use of sparse approximate inverse preconditioners with a priori sparsity patterns. International Journal of High Performance Computing, Vol 15. pp 56-74, 2001.

El algoritmo es el siguiente

entrada: $\mathbf{A}$, $\mathbf{x}_0$, $\mathbf{b}$,ε

$\mathbf{r}_0 \leftarrow \mathbf{A}\mathbf{x}_0 - \mathbf{b}$

$\mathbf{q}_0 \leftarrow \mathbf{M}^{-1}\mathbf{r}_0$

$\mathbf{p}_0 \leftarrow -\mathbf{q}_0$

$k \leftarrow 0$

mientras $\|\mathbf{r}_k\| > \varepsilon$

$\mathbf{w} \leftarrow \mathbf{A}\mathbf{p}_k$

$\alpha_k \leftarrow \dfrac{\mathbf{r}_k^{\mathrm{T}}\mathbf{q}_k}{\mathbf{p}_k^{\mathrm{T}}\mathbf{w}}$

$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{p}_k$

$\mathbf{r}_{k+1} \leftarrow \mathbf{r}_k + \alpha \mathbf{w}$

$\mathbf{q}_{k+1} \leftarrow \mathbf{M}^{-1}\mathbf{r}_{k+1}$

$\beta_k \leftarrow \dfrac{\mathbf{r}_{k+1}^{\mathrm{T}}\mathbf{q}_{k+1}}{\mathbf{r}_k^{\mathrm{T}}\mathbf{q}_k}$

$\mathbf{p}_{k+1} \leftarrow -\mathbf{q}_{k+1} + \beta_{k+1}\mathbf{p}_k$

$k \leftarrow k+1$

$\mathbf{M}^{-1}\mathbf{r}_{k+1}$ se calcula con dos multiplicaciones matriz-triangular*vector,

$$\mathbf{M}^{-1}\mathbf{r}_{k+1} = \mathbf{G}_l^{\mathrm{T}}\left(\mathbf{G}_l \mathbf{r}_{k+1}\right).$$

No hay que resolver un sistema de ecuaciones en cada paso.

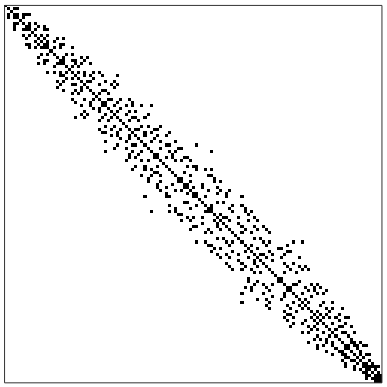
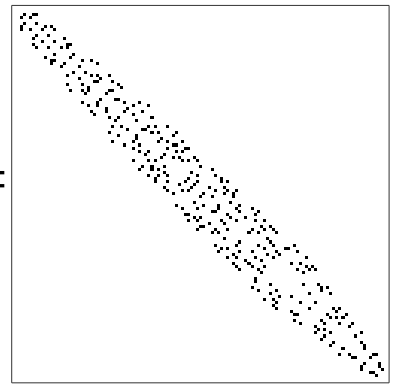
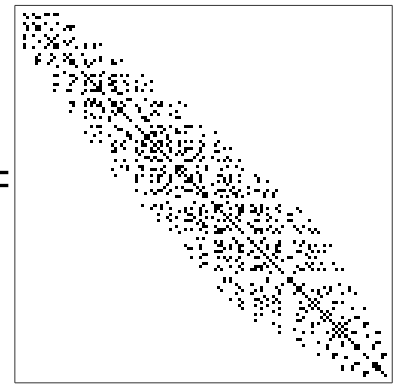
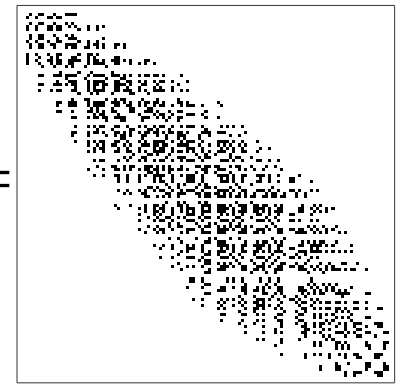
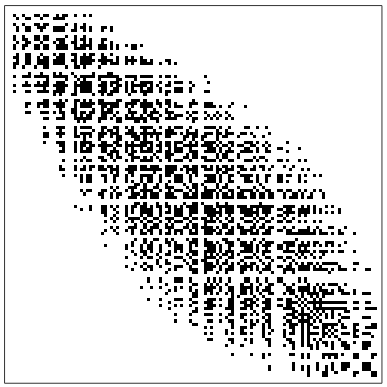
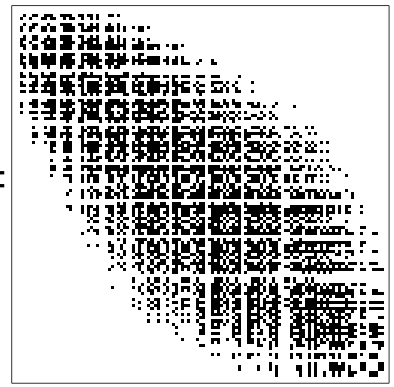
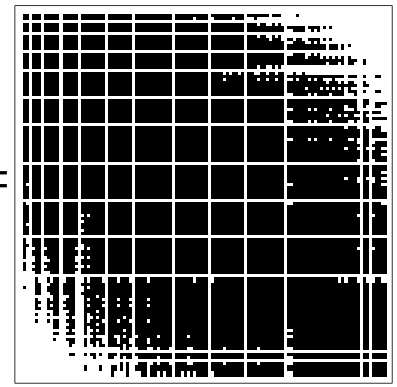
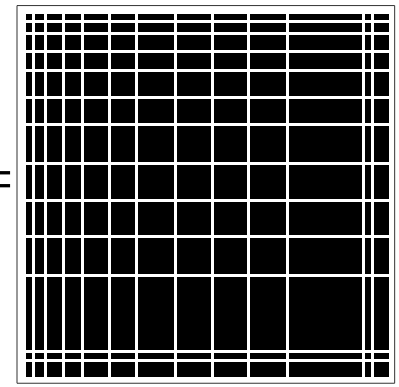
Es posible aumentar la cantidad de entradas del precondicionador y mejorar el número del condición del problema. Con el costo de aumentar la cantidad de tiempo utilizada en construir el precondicionador.

El método se conoce como potencias de matrices dispersas.

Se puede construir $\boldsymbol{S}_L$ es a través de las estructuras formadas por

$$\tilde{\mathbf{A}}, \tilde{\mathbf{A}}^2, \tilde{\mathbf{A}}^3, ..., \tilde{\mathbf{A}}^l,$$

Así, por ejemplo:

$\mathbf{A} =$ 
$\tilde{\mathbf{A}} =$ 
$\tilde{\mathbf{A}}^2 =$ 
$\tilde{\mathbf{A}}^3 =$ 
$\tilde{\mathbf{A}}^4 =$ 
$\tilde{\mathbf{A}}^5 =$ 
$\tilde{\mathbf{A}}^{10} =$ 
$\tilde{\mathbf{A}}^{20} =$ 

Las potencias de $\tilde{\mathbf{A}}$ pueden ser calculadas combinando los renglones de $\tilde{\mathbf{A}}$. Denotemos el k-ésimo renglón de $\tilde{\mathbf{A}}^l$ por $\tilde{\mathbf{A}}^l_{k,:}$, así

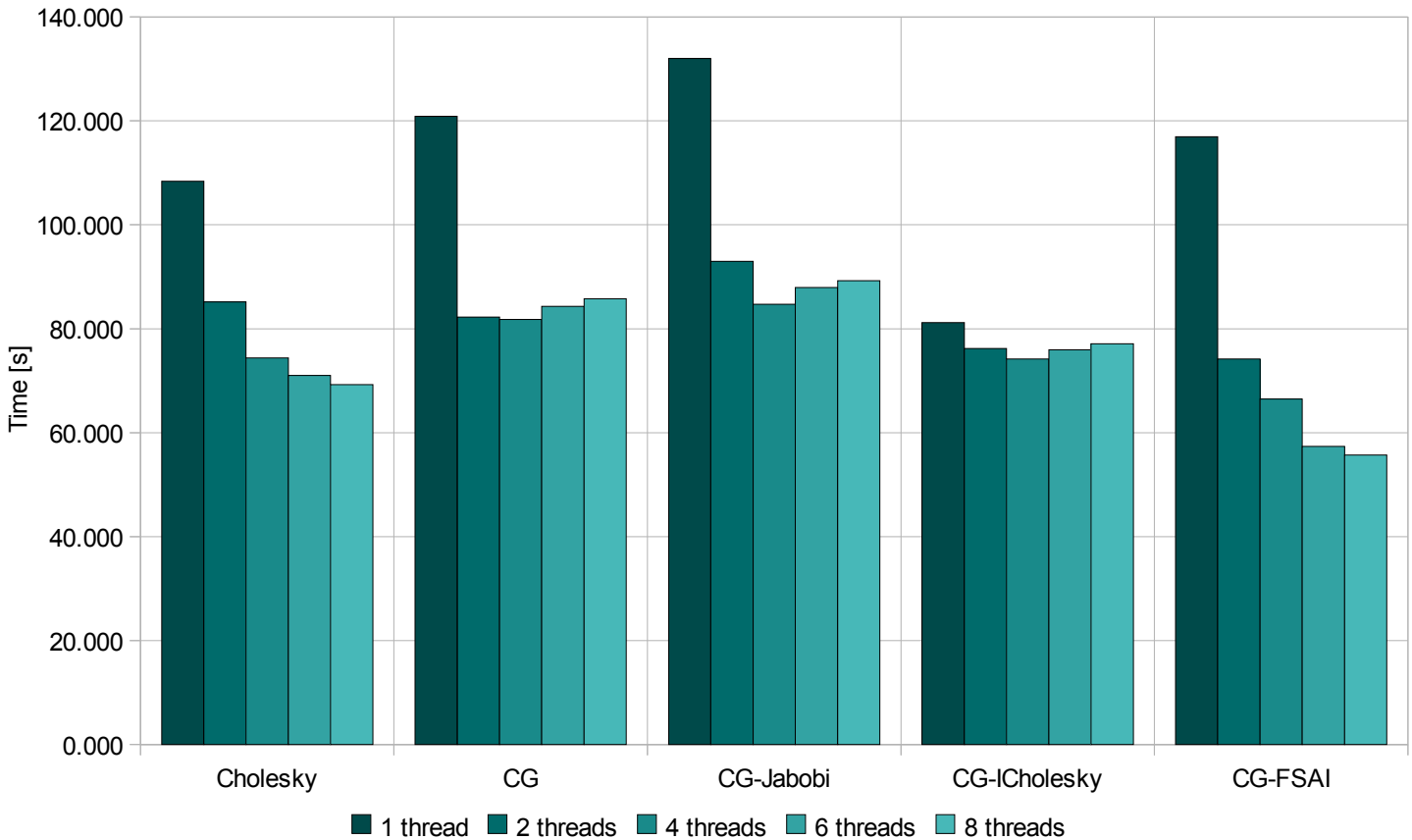
$$\tilde{\mathbf{A}}^l_{k,:} = \tilde{\mathbf{A}}^{l-1}_{k,:}\,\tilde{\mathbf{A}}.$$

La estructura $\boldsymbol{S}_L$ será entonces la parte triangular inferior de $\tilde{\mathbf{A}}^l$.

Algunos resultados numéricos

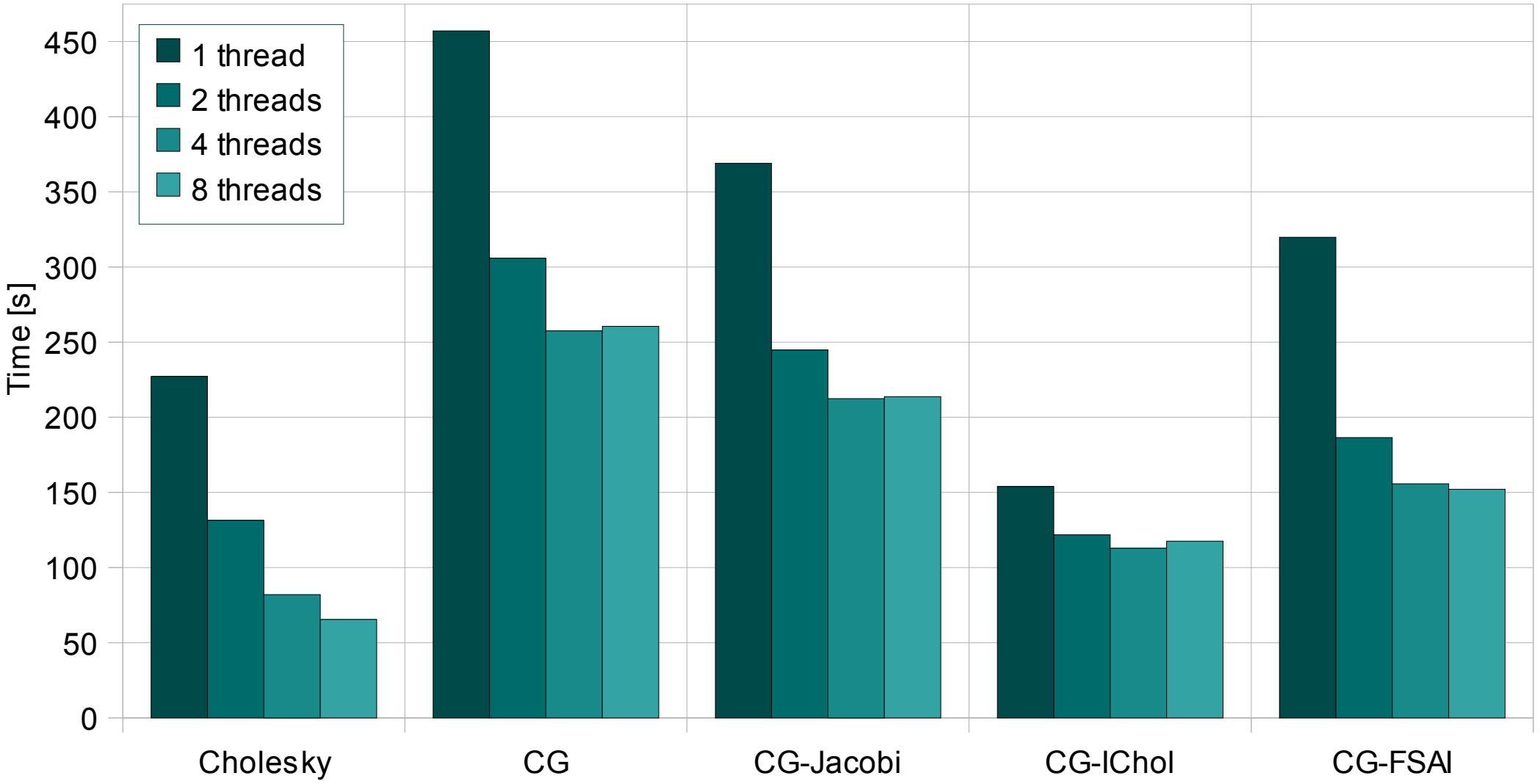
Ecuación de calor en 2D, 1’813,582 elementos, con 1,005,362 variables, $\eta(\mathbf{A}) = 6\,'355{,}782$.

Solver	1 thread [s]	2 threads [s]	4 threads [s]	6 threads [s]	8 threads [s]	Steps
Cholesky	108.4	85.1	74.4	71.0	69.276	
CG	120.8	82.2	81.8	84.3	85.769	3,718
CG-Jabobi	132.0	92.9	84.6	87.9	89.251	3,504
CG-ICholesky	81.1	76.2	74.2	75.9	77.112	978
CG-FSAI	116.9	74.2	66.4	57.3	55.707	1,288



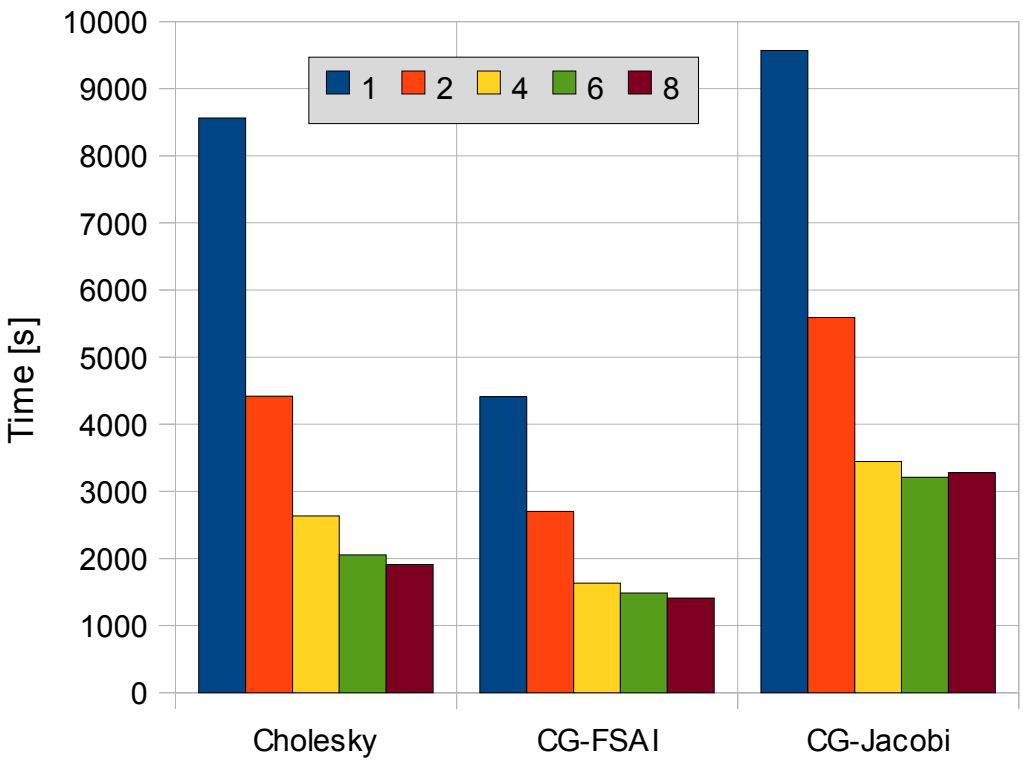
Deformación de sólidos en 2D, 501,264 elementos, con 909,540 variables, $\eta(\mathbf{A}) = 18'062{,}500$.

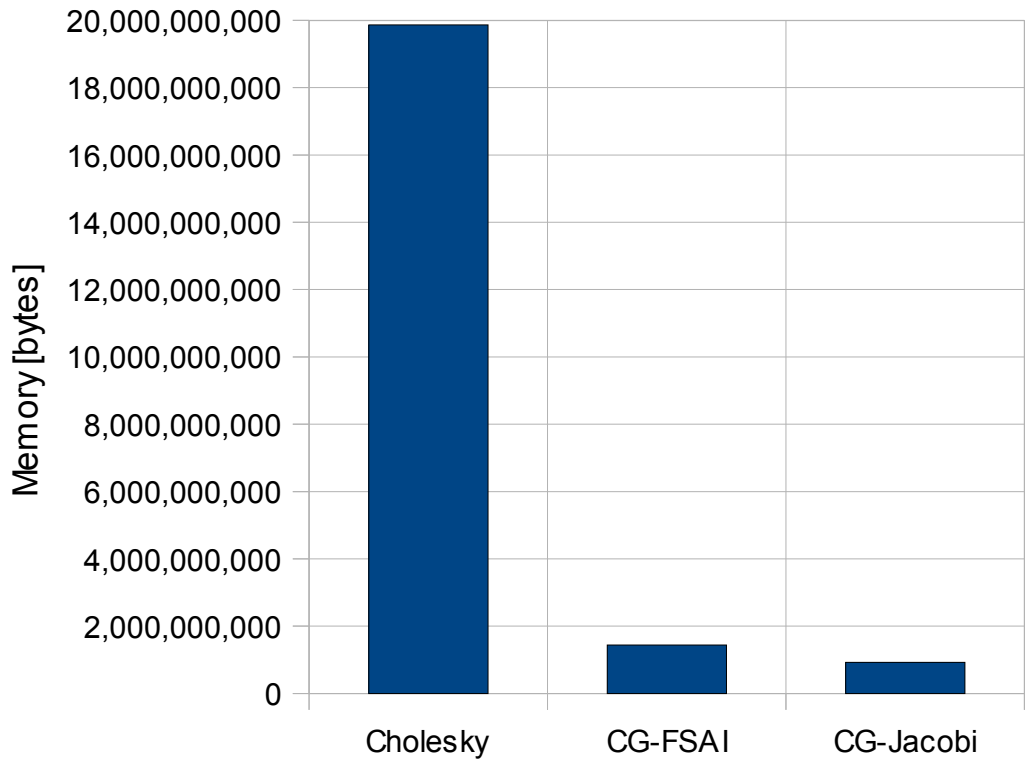
Solver	1 thread [s]	2 threads [s]	4 threads [s]	8 threads [s]	Steps	Memory
Cholesky	227.2	131.4	81.9	65.4		3,051,144,550
CG	457.0	305.8	257.5	260.4	9,251	317,929,450
CG-Jacobi	369.0	244.7	212.4	213.7	6,895	325,972,366
CG-IChol	153.9	121.9	112.8	117.6	1,384	586,380,322
CG-FSAI	319.8	186.5	155.7	152.1	3,953	430,291,930



Deformación de sólidos en 3D, 264,250 elementos, con 978,684 variables, $\eta(\mathbf{A}) = 69'255{,}522$.

Solver	1 thread [s]	2 threads [s]	4 threads [s]	6 threads [s]	8 threads [s]	Memory
Cholesky	8,562	4,418	2,635	2,055	1,911	19,864,132,056
CG-FSAI	4,410	2,703	1,631	1,484	1,410	1,440,239,572
CG-Jacobi	9,568	5,591	3,444	3,207	3,278	923,360,936





¿Preguntas?

miguelvargas@cimat.mx

Referencias

[Saad03] Y. Saad. Iterative Methods for Sparse Linear Systems. SIAM, 2003.

[Chow01] E. Chow. Parallel implementation and practical use of sparse approximate inverse preconditioners with a priori sparsity patterns. International Journal of High Performance Computing, Vol 15. pp 56-74, 2001.