On Kaczmarz's projection iteration as a direct solver for linear least squares problems

Constantin Popa

"Ovidius" University, Constanta, Romania

joint work with Harald Köstler, Tobias Preclik, Ulrich Rüde
"Friedrich-Alexander" University, Erlangen, Germany

Talk at ALA 2010 Conference, Novi Sad
Thursday May 27, 2010

Details in Tech. Rep. 10-6, IMMD10 Erlangen
www10.informatik.uni-erlangen.de/Publications/TechnicalReports/

Directional projections

$A : m \times n,\ b \in \mathcal{R}(A) \subset \boldsymbol{R}^m;\ Ax = b,\ S(A; b) = \mathcal{N}(A) + x_{LS}$

$A_i \neq 0,\ A^j \neq 0,\ \ H_i = \{x \in \boldsymbol{R}^n, \langle A_i, x\rangle = b_i\},$

$d \in \boldsymbol{R}^n, \langle d, A_i\rangle \neq 0;$ "directional projection" $P_{H_i}^{(d)}$

$$P_{H_i}^{(d)}(x) = x - \frac{\langle x, A_i\rangle - b_i}{\langle d, A_i\rangle} d,$$

i.e. projection onto H_i parallel with d

The General Row Projection algorithm

Algorithm GRP. (C. Brezinski, book 1997)

Step 1 (Initialization): a set of vectors $\{z^1, \ldots, z^m\} \subset \boldsymbol{R}^n$ such that

$$\langle z^i, A_i \rangle \neq 0, \ \ i = 1, \ldots, m$$

and an initial approximation $x^0 \in \boldsymbol{R}^n$

Step 2 (Successive projections):
for $i = 1, \ldots, m$

$$x^i = P_{H_i}^{(z^i)}(x^{i-1}) = x^{i-1} - \frac{\langle x^{i-1}, A_i \rangle - b_i}{\langle z^i, A_i \rangle} z^i$$

end

Step 3 (Ending procedure):
if $x^m = x^*$ STOP
else set $x^0 = x^m$ and go to Step 2 until convergence

Examples

Iterative methods

- for $m = n$ and $z^i = e^i$: Gauss-Seidel iteration
- for $z^i = A_i$: Kaczmarz's successive projections iteration

Direct methods

- E. Purcell (1953), T. Pietrzykowski (1960)
- Null space algorithm (M. Benzi, C. Mayer. M. Tuma - 1993, 1995, 1996); Generalized Null Space (consistent rectangular systems)
- A direct projection algorithm and its extension to overdetermined full-column rank least squares problems (F. Sloboda, 1978)
- ABS algorithm (Abaffy J., Broyden C., Spedicato E. - 1984; a more general method; connections with CG, etc)

Kaczmarz's iterative method

$$f_i(b;x) = P_i(x) + \frac{b_i}{\|A_i\|^2} A_i, \quad P_i(x) = x - \frac{\langle x, A_i \rangle}{\|A_i\|^2} A_i$$

Algorithm Kaczmarz (K). *Initialization:* $x^0 \in \boldsymbol{R}^n$
Iterative step:

$$x^{k+1} = F(b;x^k) = (f_1 \circ \cdots \circ f_m)(b;x^k) = Qx^k + Rb.$$

where

$$Q : n \times n, Q = P_1 P_2 \dots P_m, \; R : n \times m, R = col(R^1, \dots, R^m),$$

$$R^1 = \frac{1}{\|A_1\|^2} A_1, \; R^i = \frac{1}{\|A_i\|^2} P_1 P_2 \dots P_{i-1}(A_i), i = 2, \dots, m$$

R. (K. Tanabe, 1971) *For any* $x^0 \in \boldsymbol{R}^n$, $(x^k)_{k \geq 0}$ *converges and*

$$\lim_{k \to \infty} x^k = P_{\mathcal{N}(A)}(x^0) + x_{LS} \in S(A;b)$$

.

Kaczmarz as a direct solver

R. (P., 2002) *If the matrix* $Q = P_1 P_2 \dots P_m$ *satisfies*

$$Q(A_i) = 0, \forall\ i = 1, \dots, m \quad (*),$$

then, for any $x^0 \in \boldsymbol{R}^n$, *the first approximation,* x^1 *generated with algorithm* **K** *is given by*

$$x^1 = P_{\mathcal{N}(A)}(x^0) + x_{LS} \in S(A; b),$$

i.e. the algorithm **K** *becomes a direct solver.*

Note. If the matrix A has mutually orthogonal rows then $(*)$ holds.

Supplementary directions for projection - 1

$P_{H_i}^{(d)}(x) = x - \frac{\langle x, A_i\rangle - b_i}{\langle d, A_i\rangle} d$ projection onto H_i parallel with d is replaced by

$P^d(x) = x - \frac{\langle x, d\rangle - b(d)}{\langle d, d\rangle} d$, i.e. the orthogonal projection onto the

hyperplane $H_d = \{z, \langle z, d\rangle = b(d)\}$ (if $d = 0$, $P^d(x) = x$)

$$\begin{cases} d_{m-1} = & P_m(A_{m-1}) \\ d_{m-2} = & P_{m-1}P^{d_{m-1}}P_m(A_{m-2}) \\ \dots & \dots \\ d_1 = & P_2P^{d_2}\dots P_{m-1}P^{d_{m-1}}P_m(A_1) \end{cases}$$

$$d_{m-k} = A_{m-k} + \alpha_{m-k+1}A_{m-k+1} + \dots + \alpha_m A_m$$

$$b(d_{m-k}) = b_{m-k} + \alpha_{m-k+1}b_{m-k+1} + \dots + \alpha_m b_m$$

Supplementary directions for projection - 2

Formally we get an "extended" system:
$\widetilde{A} : (2m-1) \times n, \tilde{b} \in \boldsymbol{R}^{2m-1}$

$$\widetilde{A}x = \widetilde{b} \Leftrightarrow \begin{bmatrix} A_1^T \\ d_1^T \\ A_2^T \\ \dots \\ A_{m-1}^T \\ d_{m-1}^T \\ A_m^T \end{bmatrix} x = \begin{bmatrix} b_1 \\ b(d_1) \\ b_2 \\ \dots \\ b_{m-1} \\ b(d_{m-1}) \\ A_m^T \end{bmatrix}$$

which is still consistent because of the definition of $b(d_{m-k})$, w.r.t. the construction of d_{m-k} (i.e. for $x \in S(A; b)$, $d_{m-k}^T x = \langle d_{m-k}, x \rangle = b(d_{m-k})$

Direct Kaczmarz algorithm

K applied to the extended system $\widetilde{A}x = \widetilde{b}$ gives

Algorithm Direct Kaczmarz (DK) *Initialization:* $x^0 \in \boldsymbol{R}^n$
Iterative step: $x^{k+1} = \bar{F}(b; x^k)$

$$\text{where } f^{d_i}(b;x) = \begin{cases} x - \frac{\langle x, d_i\rangle - b(d_i)}{\langle d_i, d_i\rangle} d_i, & d_i \neq 0, \\ x, & d_i = 0. \end{cases}$$

$$\bar{F}(b;x) = f_1 \circ f^{d_1} \circ f_2 \circ f^{d_2} \circ \cdots \circ f_{m-1} \circ f^{d_{m-1}} \circ f_m(b;x)$$

R. (P., 2002) *If $b \in \mathcal{R}(A)$, then for any $x^0 \in \boldsymbol{R}^n$ the algorithm DK produces in one iteration the vector*

$$x^1 = P_{\mathcal{N}(A)}(x^0) + x_{LS} \in S(A;b)$$

R. (P. et al., 2010) *For any $i = 1, \ldots, m-1$*

$d_{m-i} = 0$ if and only if $A_{m-i} \in sp\{A_m, A_{m-1}, A_{m-i+1}\}$

Block-row Kaczmarz algorithm

$$A = \begin{bmatrix} B_1^T \\ B_2^T \\ \dots \\ B_p^T \end{bmatrix}, B_i^T = \begin{bmatrix} A_{m_{i-1}+1}^T \\ A_{m_{i-1}+2}^T \\ \dots \\ A_{m_i}^T \end{bmatrix}, b = \begin{bmatrix} b^1 \\ b^2 \\ \dots \\ b^p \end{bmatrix}, b^i = \begin{bmatrix} b_{m_{i-1}+1} \\ b_{m_{i-1}+2} \\ \dots \\ b_{m_i} \end{bmatrix},$$

$$\Pi_i = I - (B_i^T)^+ B_i^T, \mathcal{F}_i(b; x) = \Pi_i(x) + (B_i^T)^+ b^i, i = 1, \dots, p$$

$$\mathcal{F}(b; x) = (\mathcal{F}_1 \circ \dots \circ \mathcal{F}_p)(b; x), \quad x \in \boldsymbol{R}^n$$

Algorithm Block Kaczmarz (BK). *Initialization:* $x^0 \in \boldsymbol{R}^n$
Iterative step: $x^{k+1} = \mathcal{F}(b; x^k)$

R. (T. Elfving, 1980) *For any* $x^0 \in \mathcal{R}(A)$,

$$\exists \lim_{k \to \infty} x^k = x_{LS} \in S(A; b)$$

Supplementary block-row directions

Note. $(B_i^T)^+ = B_i(B_i^T B_i)^+ \Rightarrow \Pi_i = I - B_i(B_i^T B_i)^+ B_i^T$

If $D = B_{p-1} - B_p(B_p^T B_p)^+ B_p^T B_{p-1}$, we have for $x \in S(A; b)$

$$D^T x = B_{p-1}^T x - B_{p-1}^T B_p (B_p^T B_p)^+ B_p^T x = b^{p-1} - B_{p-1}^T B_p (B_p^T B_p)^+ b^p$$

thus, we define the block right hand side $b(D)$ by

$$b(D) = b^{p-1} - B_{p-1}^T B_p (B_p^T B_p)^+ b^p = b^{p-1} - B_{p-1}^T (B_p^T)^+ b^p$$

$$\begin{cases} D_{p-1} = & \Pi_p(B_{p-1}) \\ D_{p-2} = & \Pi_{p-1}\Pi^{D_{p-1}}\Pi_p(B_{p-2}) \\ \dots & \dots \\ D_1 = & \Pi_2\Pi^{D_2}\dots\Pi_{p-1}\Pi^{D_{p-1}}\Pi_p(B_1), \end{cases}$$

$$b(D_{p-1}), \quad b(D_{p-2}) \quad , \dots , \quad b(D_p)$$

Direct block Kaczmarz algorithm

$$\mathcal{F}_i(b;x) = \Pi_i(x) + (B_i^T)^+ b^i, \quad \mathcal{F}^D(b;x) = \Pi^D(x) + (D^T)^+ b(D)$$

where $\Pi^D = I - (D^T)^+ D^T$

$$\widetilde{\mathcal{F}}(b;x) = \mathcal{F}_1 \circ \mathcal{F}^{D_1} \circ \mathcal{F}_2 \circ \mathcal{F}^{D_2} \circ \cdots \circ \mathcal{F}_{p-1} \circ \mathcal{F}^{D_{p-1}} \circ \mathcal{F}_p(b;x).$$

Algorithm Direct Block Kaczmarz (DBK)
Initialization: $x^0 \in \boldsymbol{R}^n$
Iterative step: $x^{k+1} = \widetilde{\mathcal{F}}(b;x^k)$

R. (P. et al., 2010) *For any $x^0 \in \boldsymbol{R}^n$, the algorithm DBK produces in one iteration $x^1 = P_{\mathcal{N}(A)}(x^0) + x_{LS} \in S(A;b)$.*

Blocks with mutually orthogonal rows

R. (P., 2002) Let $B_i^T = \begin{bmatrix} A_{m_{i-1}+1}^T \\ A_{m_{i-1}+2}^T \\ \dots \\ A_{m_i}^T \end{bmatrix}$, $P_i(x) = x - \frac{\langle x, A_i \rangle}{\|A_i\|^2} A_i$, $\Pi_i = I - (B_i^T)^+ B_i^T$. If the rows $A_{m_{i-1}+1}, \dots, A_{m_i}$ are mutually orthogonal then

$$\Pi_i x = \sum_{j=m_{i-1}+1}^{m_i} P_j x - (m_i - 1)x$$

Some reduction of the computational cost

$$\widetilde{\mathcal{F}}(b; x) = \mathcal{F}_1 \circ \mathcal{F}^{D_1} \circ \mathcal{F}_2 \circ \mathcal{F}^{D_2} \circ \dots \circ \mathcal{F}_{p-1} \circ \mathcal{F}^{D_{p-1}} \circ \mathcal{F}_p(b; x)$$

$$\left(\mathcal{F}_i(b; x) = \Pi_i(x) + (B_i^T)^+ b^i, \quad \mathcal{F}^D(b; x) = \Pi^D(x) + (D^T)^+ b(D)\right)$$

Extended Kaczmarz algorithm

$$\varphi_j(y) = y - \frac{\langle y, A^j \rangle}{\|A^j\|^2} A^j, \qquad \Phi(y) = (\varphi_1 \circ \cdots \circ \varphi_n)(y)$$

Algorithm Extended Kaczmarz (EK).
Initialization: $x^0 \in \boldsymbol{R}^n, y^0 = b$
Iterative step:

$$y^{k+1} = \Phi(y^k),\ b^{k+1} = b - y^{k+1},\ x^{k+1} = F(b^{k+1}; x^k)$$

R. (P., 1995) *For any vector $b \in \boldsymbol{R}^m$ and any $x^0 \in \boldsymbol{R}^n$ the sequence $(x^k)_{k\geq 0}$ generated with the algorithm EK converges, and*

$$\lim_{k\to\infty} x^k = P_{\mathcal{N}(A)}(x^0) + x_{LS} \in LSS(A; b)$$

Direct Extended Kaczmarz algorithm

$$\begin{cases} d_{m-1} = & P_m(A_{m-1}) \\ d_{m-2} = & P_{m-1}P^{d_{m-1}}P_m(A_{m-2}) \\ \dots & \dots \\ d_1 = & P_2P^{d_2}\dots P_{m-1}P^{d_{m-1}}P_m(A_1) \end{cases} \begin{cases} \delta_{n-1} = & \varphi_n(A^{n-1}) \\ \delta_{n-2} = & \varphi_{n-1}\varphi^{\delta_{n-1}}\varphi_n(A^{n-2}) \\ \dots & \dots \\ \delta_1 = & \varphi_2\varphi^{\delta_2}\dots\varphi_{n-1}\varphi^{\delta_{n-1}}\varphi_n(A^1). \end{cases}$$

$\bar{F}(b;x) = f_1 \circ f^{d_1} \circ f_2 \circ f^{d_2} \circ \dots \circ f_{m-1} \circ f^{d_{m-1}} \circ f_m(b;x)$
$\bar{\Phi} = \varphi_1\varphi^{\delta_1}\varphi_2\varphi^{\delta_2}\dots\varphi_{n-1}\varphi^{\delta_{n-1}}\varphi_n$

Algorithm Direct Extended Kaczmarz (DEK). *Initialization:*
$y^0 = b, x^0 \in \boldsymbol{R}^n$
Iterative step:

$$y^{k+1} = \bar{\Phi}(y^k), b^{k+1} = b - y^{k+1}, x^{k+1} = \bar{F}(b^{k+1}; x^k).$$

R. (P. et al., 2010) *For any $b \in \boldsymbol{R}^m$ and any $x^0 \in \boldsymbol{R}^n$, DEK gives in one iteration the vector $x^1 = P_{\mathcal{N}(A)}(x^0) + x_{LS} \in LSS(A;b)$*

Direct Block Extended Kaczmarz algorithm

$$\begin{cases} D_{p-1} = & \Pi_p(B_{p-1}) \\ D_{p-2} = & \Pi_{p-1}\Pi^{D_{p-1}}\Pi_p(B_{p-2}) \\ \dots & \dots \\ D_1 = & \Pi_2\Pi^{D_2}\dots\Pi_{p-1}\Pi^{D_{p-1}}\Pi_p(B_1) \end{cases} \quad \begin{cases} \Delta_{q-1} = & \Gamma_q(C_{q-1}) \\ \Delta_{q-2} = & \Gamma_{q-1}\Gamma^{\Delta_{q-1}}\Gamma_q(C_{q-2}) \\ \dots & \dots \\ \Delta_1 = & \Gamma_2\Gamma^{\Delta_2}\dots\Gamma_{q-1}\Gamma^{\Delta_{q-1}}\Gamma_q(C_1) \end{cases}$$

$$\widetilde{\mathcal{F}}(b;x) = \mathcal{F}_1 \circ \mathcal{F}^{D_1} \circ \mathcal{F}_2 \circ \mathcal{F}^{D_2} \circ \dots \circ \mathcal{F}_{p-1} \circ \mathcal{F}^{D_{p-1}} \circ \mathcal{F}_p(b;x)$$

$$\widetilde{\Phi}(y) = \Gamma_1 \circ \Gamma^{\Delta_1} \circ \Gamma_2 \circ \Gamma^{\Delta_2} \circ \dots \circ \Gamma_{q-1} \circ \Gamma^{\Delta_{q-1}} \circ \Gamma_q(y)$$

Algorithm Direct Block Extended Kaczmarz (DBEK).

Initialization: $y^0 = b, x^0 \in \boldsymbol{R}^n$

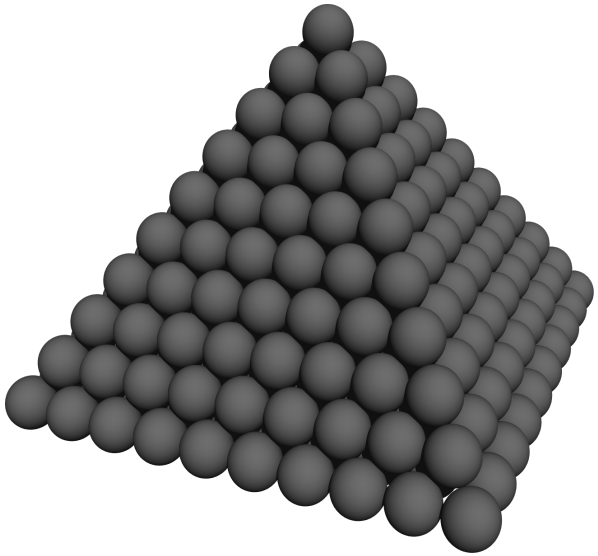
Iterative step:

$$y^{k+1} = \widetilde{\Phi}(y^k), b^{k+1} = b - y^{k+1}, x^{k+1} = \widetilde{\mathcal{F}}(b^{k+1}; x^k).$$

R. (P. et al., 2010) *For any $b \in \boldsymbol{R}^m$ and any $x^0 \in \boldsymbol{R}^n$ the algorithm DBEK produces in one iteration the vector* $x^1 = P_{\mathcal{N}(A)}(x^0) + x_{LS} \in LSS(A;b)$

Numerical experiments - Rigid multibody problems

(a) Well



(b) Pyramid

(c) Mobile

Numerical experiments - construction of blocks

- the row and column blocks (without the last (smallest) ones) contain mutually orthogonal rows or columns; they have been generated with the algorithm from the paper [P., *B I T*, **39(2)**(1999), 323-338]
- the computational times for the Direct Block versions of the algorithms do not include the generation of row or column blocks

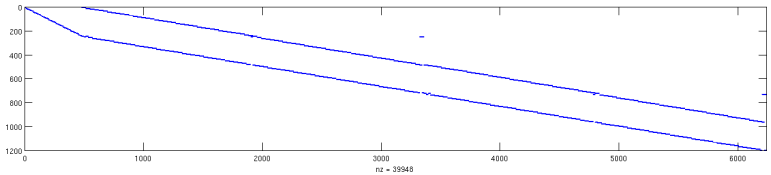
Well problem - computational times

$A : 1200 \times 6240$; $rank(A) = 1200$; consistent system

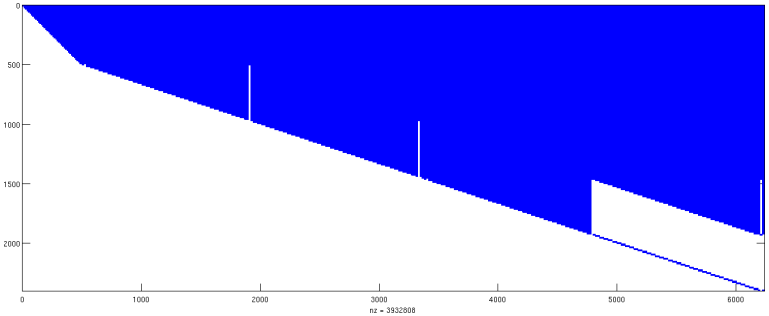
Algorithm	Runtime in s	Error	Residual
DK	438.78	$1.18 \cdot 10^{-14}$	$2.86 \cdot 10^{-15}$
DBK	9.65	$1.24 \cdot 10^{-14}$	$5.79 \cdot 10^{-15}$
GNS	874.43	4.44	$1.01 \cdot 10^{-15}$

Table: Numerical results for DK, DBK and GNS methods on the well test problem.

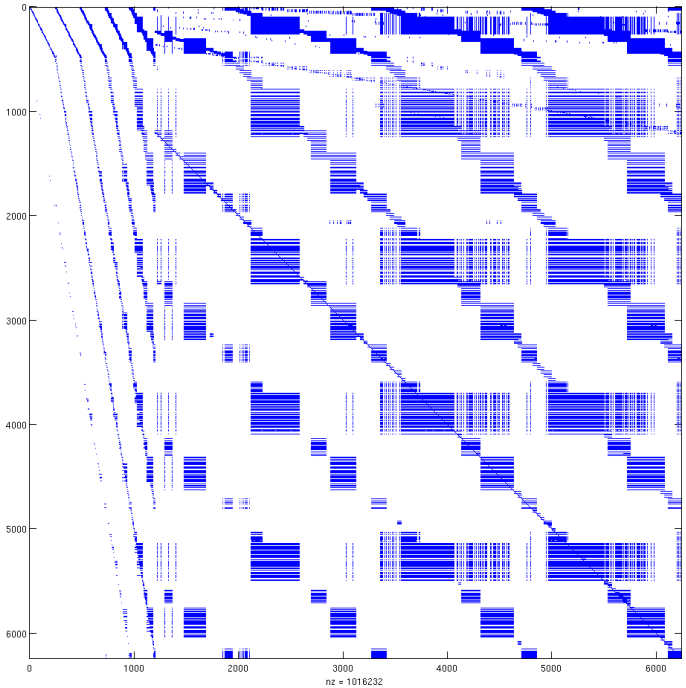
Well problem - fill-in process



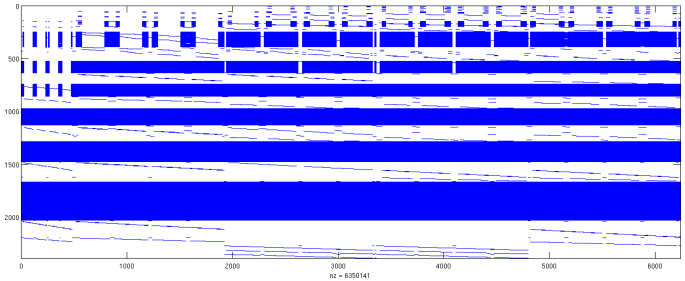
(a) Original



(b) DK



(c) GNS



(d) DBK

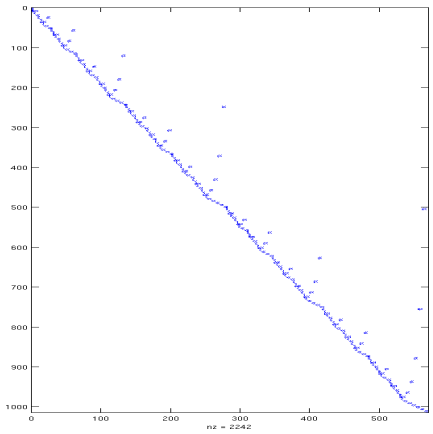
Mobile problem - computational times

$A : 1013 \times 570$; $rank(A) = 570$; inconsistent system

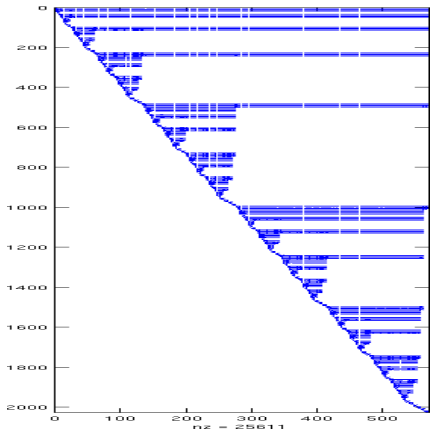
Algorithm	Runtime in s	Error	Residual
DEK	48.90	$1.39 \cdot 10^{-15}$	$2.42 \cdot 10^{-13}$
DBEK	1.59	$8.09 \cdot 10^{-16}$	$4.40 \cdot 10^{-16}$
GS	165.65	$5.40 \cdot 10^{-15}$	$5.15 \cdot 10^{-15}$

Table: Numerical results for DEK, DBEK and GS methods on the mobile test problem.

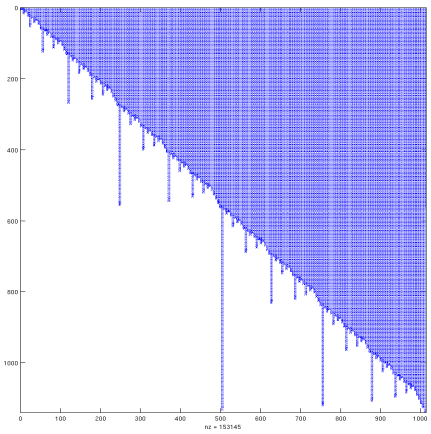
Mobile problem - fill-in process



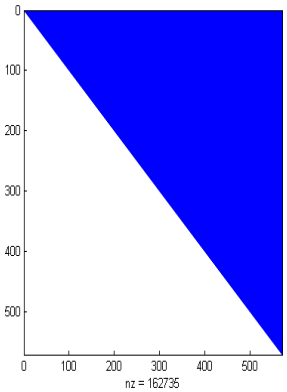
(a) Original



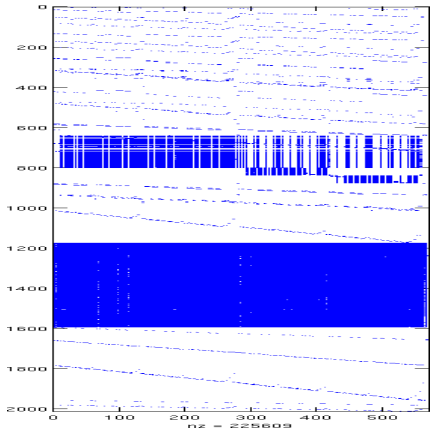
(b) DEK (A)



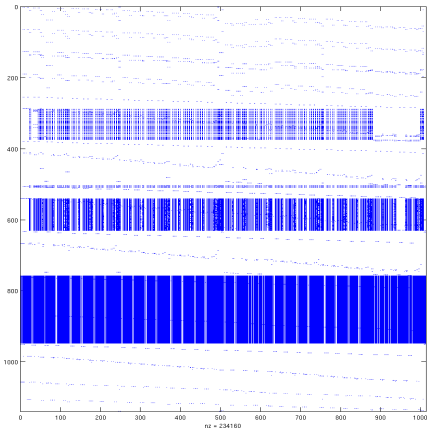
(c) DEK (A^T)



(d) GS



(e) DBEK (A)



(f) DBEK (A^T)

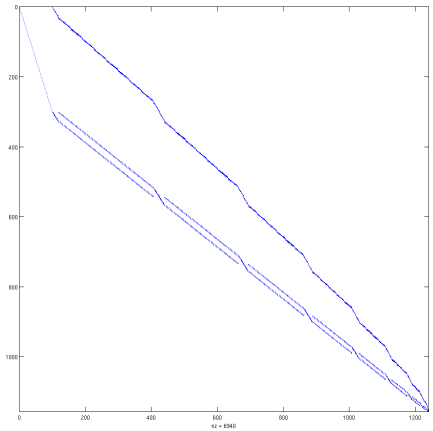
Pyramid problem - computational times

$A : 1155 \times 1240$; $rank(A) = 1100$; inconsistent system

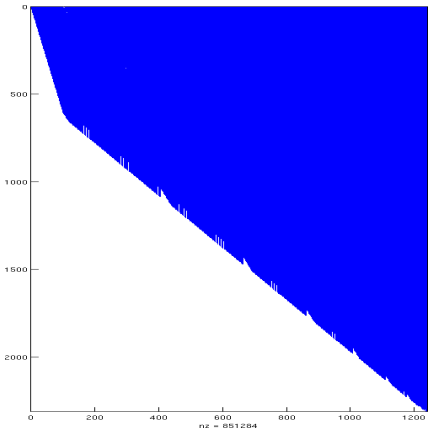
Algorithm	Runtime in s	Error	Residual
DEK	194.05	$1.83 \cdot 10^{-14}$	$2.01 \cdot 10^{-14}$
DBEK	7.21	$9.44 \cdot 10^{-16}$	$7.03 \cdot 10^{-16}$

Table: Numerical results for DEK and DBEK methods on the pyramid test problem.

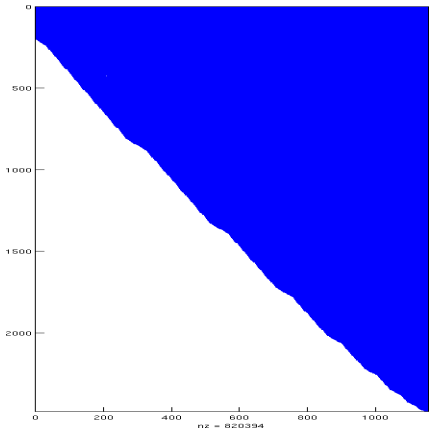
Pyramid problem - fill-in process



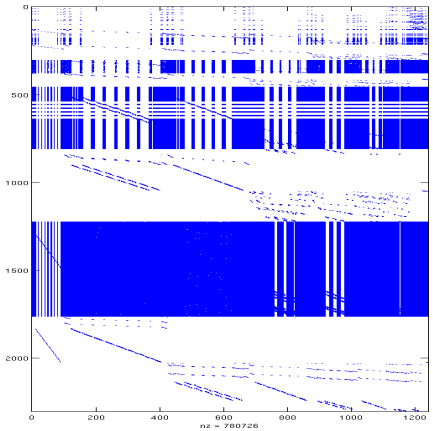
(a) Original



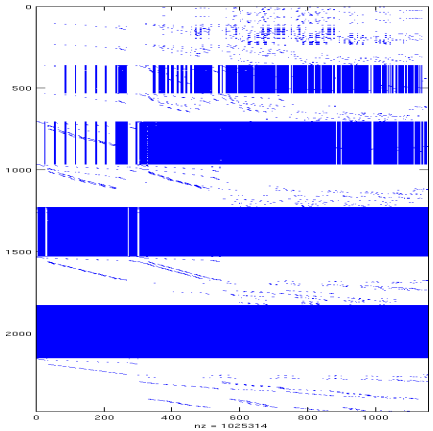
(b) DEK (A)



(c) DEK (A^T)



(d) DBEK (A)



(e) DBEK (A^T)

Thanks for your attention !