FRIEDRICH-ALEXANDER-UNIVERSITÄT ERLANGEN-NÜRNBERG
INSTITUT FÜR INFORMATIK (MATHEMATISCHE MASCHINEN UND DATENVERARBEITUNG)

Lehrstuhl für Informatik 10 (Systemsimulation)

Some projection-based direct solvers for general linear systems of equations

C. Popa, T. Preclik, H. Köstler, U. Rüde

Technical Report 10-6

Some projection-based direct solvers for general linear systems of equations

C. Popa*, T. Preclik†, H. Köstler‡, U. Rüde§

May 4, 2010

Abstract. The Direct Projection-based solvers are direct methods for solving linear systems of equations in which an initial set of vectors are projected onto the hyperplanes of the system by using projections parallel with some specific directions also constructed during the development of the algorithm. They have been initially designed for square nonsingular systems, but further developments were produced also for more general ones (see an overview of these algorithms in [1], [6] and references therein). The main scope of our paper is to introduce and theoretically analyse a new class of such kind of direct solvers. Starting from some preliminary methods and results proposed by one of the authors in [15] we extend the theoretical analysis for one of them and design new block row and column projection versions. In the last section of the paper we use these algorithms and compare them with some classical direct projection-based ones for some numerical experiments on linear systems arising from rigid body dynamics problems.

1 Introduction

Let $A : n \times n$ be a nonsingular matrix and $b \in I\!R^n$. We consider the system

$$Ax^* = b,\ x^* = A^{-1}b \text{ its unique solution.} \tag{1}$$

We shall denote by $\langle \cdot, \cdot \rangle, \| \cdot \|$ the Euclidean scalar product and norm, and by

$$S_i = \{x \in I\!R^n, \langle A_i, x \rangle = 0\}, \quad H_i = \{x \in I\!R^n, \langle A_i, x \rangle = b_i\} = S_i + \frac{b_i}{\| A_i \|^2} A_i, \tag{2}$$

the vector subspace and hyperplane, respectively associated to the i-th equation of the system (1), where $A_i, i = 1, \ldots, n$ are the rows of A. For $d \in I\!R^n$ such that $\langle d, A_i \rangle \neq 0$ and any $x \in I\!R^n$ there exists the projections of x parallel with d onto S_i and H_i (also called "directional projections"), defined by

$$P^d_{S_i}(x) = x - \frac{\langle x, A_i \rangle}{\langle d, A_i \rangle} d, \quad P^d_{H_i}(x) = x - \frac{\langle x, A_i \rangle - b_i}{\langle d, A_i \rangle} d. \tag{3}$$

We shall consider the General Row Projection (GRP) algorithm for solving (1) (see [6], page 123).
Algorithm GRP.

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

$$\langle z^i, A_i \rangle \neq 0,\ i = 1, \ldots, n \tag{4}$$

and an initial approximation $x^0 \in I\!R^n$

*Ovidius University of Constanta, Faculty of Mathematics and Computer Science, cpopa@univ-ovidius.ro
†University of Erlangen-Nürnberg, Computer Science 10, System Simulation, tobias.preclik@informatik.uni-erlangen.de
‡University of Erlangen-Nürnberg, Computer Science 10, System Simulation, harald.koestler@informatik.uni-erlangen.de
§University of Erlangen-Nürnberg, Computer Science 10, System Simulation, ulrich.ruede@informatik.uni-erlangen.de

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

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

end

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

Two well known particular cases of the above algorithm are the following.

- $z^i = e^i$ (the vectors from the canonical basis of $\mathbb{R}^n$) which gives us the Gauss-Seidel iteration (see e.g. [19])
- $z^i = A_i$ which gives us Kaczmarz's projection iteration (see [8] and section 3.1 of this paper)

Both these methods are iterative. For convergence Gauss-Seidel needs supplementary assumptions on the matrix A, whereas Kaczmarz converges for any nonsingular A (see e.g. [19]). But, in the present paper we are interested in direct methods of the form GRP, i.e for which in Step 3 we get $x^n = x^*$. The basic idea for this would be that the approximations x^i satisfy

$$x^i \in H_1 \cap H_2 \cap \cdots \cap H_i, \tag{5}$$

$\forall i = 1, \ldots, n$, thus

$$x^n \in H_1 \cap H_2 \cap \cdots \cap H_n = \{x^*\}. \tag{6}$$

A sufficent condition for this is given in the following result (see also [6]).

Lemma 1 *If the vectors z^i satisfy the relations*

$$z^i \in S_1 \cap S_2 \cap \cdots \cap S_{i-1}, \tag{7}$$

$\forall i = 2, \ldots, n$ then the assumption (5) holds.

Proof. Let us suppose that we are in the hypothesis (7); we firstly observe that $x^1 \in H_1$ according to the construction in Step 2 of GRP. Then, we shall use the mathematical induction: let $n - 1 \geq i \geq 2$ be a fixed index such that (5) holds for it. By again using Step 2 of GRP we get

$$x^{i+1} = P_{H_{i+1}}^{z^{i+1}}(x^i) = x^i - \frac{\langle x^i, A_{i+1}\rangle - b_{i+1}}{\langle z^{i+1}, A_{i+1}\rangle} z^{i+1} \in H_{i+1}. \tag{8}$$

From (5) (i.e $x^i \in H_i, j = 1, \ldots, i$), (7) (i.e. $z^{i+1} \in S_j, j = 1, \ldots, i$), the last equality in (2) and (8) (i.e. $x^{i+1} = x^i + \alpha_i z^{i+1}$ ($\alpha_i \in \mathbb{R}$ is the scalar from (8)) we get $x^{i+1} \in H_i, j = 1, \ldots, i$, which together with (8) completes the induction step and the proof. ♠

2 Square Nonsingular Systems

2.1 Slododa's Algorithm

Let $x_0^{(0)}, x_0^{(1)}, \ldots, x_0^{(n)} \in \mathbb{R}^n$ such that the difference vectors

$$x_0^{(1)} - x_0^{(0)}, x_0^{(2)} - x_0^{(0)}, \ldots, x_0^{(n)} - x_0^{(0)} \text{ are linearly independent.} \tag{9}$$

Example 1

$$x_0^{(0)} = 0,\ x_0^{(i)} = e^i,\ i = 1, \ldots, n, \tag{10}$$

where $\{e^1, \ldots, e^n\}$ *is the canonical basis in* $\mathbb{R}^n$.

In [17] F. Sloboda proposed the following algorithm for solving the system (1).

Algorithm Sloboda (S). Let $x_0^{(0)}, x_0^{(1)}, \ldots, x_0^{(n)} \in \mathbb{R}^n$ be as before; recursively define the vectors $x_i^{(k)} \in \mathbb{R}^n$, $i = 1, \ldots, n,\ k = i, i+1, \ldots, n$ by

$$x_i^{(k)} = P_{H_i}^{v_{i-1}^{(i)}}(x_{i-1}^{(k)}) = x_{i-1}^{(k)} - \frac{\langle x_{i-1}^{(k)}, A_i\rangle - b_i}{\langle v_{i-1}^{(i)}, A_i\rangle} v_{i-1}^{(i)}, \tag{11}$$

where

$$v_{i-1}^{(i)} = x_{i-1}^{(i)} - x_{i-1}^{(i-1)}. \tag{12}$$

Theorem 1 *([17]) Let us suppose that for the above algorithm we have*

$$\langle v_{i-1}^{(i)}, A_i\rangle \neq 0, \forall i = 1, \ldots, n. \tag{13}$$

Then $x_n^{(n)} = x^*$, *where* x^* *is the (unique) solution of the system (1).*

Unfortunately, Sloboda's algorithm does not work for any nonsingular system. This is related to the assumption (13) which is not always satisfied, as you can see from the Example 2 below.

Example 2 *Let*

$$A = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad b = (3, 1, 1)^T,$$

with (see (1)) $x^* = (1, 1, 1)^T$. *By applying the algorithm S we get* $\langle v_0^{(1)}, A_1\rangle = 1$, *but* $\langle v_1^{(2)}, A_2\rangle = 0$, *and the algorithm will stop before finding the solution.*

A class of matrices for which (13) always hold is given in the following result (for the proof see e.g. [7]).

Proposition 1 *If the matrix A is strictly regular, i.e.*

$$\det\left(\begin{bmatrix} A_{11} & \ldots & A_{1k} \\ \ldots & \ldots & \ldots \\ A_{k1} & \ldots & A_{kk} \end{bmatrix}\right) \neq 0,\ \forall k = 1, \ldots, n, \tag{14}$$

and if we start with the vectors from Example 1, then the assumption (13) holds.

Remark 1 *The assumption (14) also ensures the existence of an LU decomposition of A. Irreducible diagonally dominant matrices or symmetric and positive definite ones satisfy (14) (see e.g. [7]).*

2.2 The Null Space Algorithm

If $A^{(k)} : k \times n, k = 1, \ldots, n$ is the submatrix formed with its first k rows, i.e.

$$A = \begin{bmatrix} A_1^T \\ A_2^T \\ \ldots \\ A_n^T \end{bmatrix}, \quad A^{(k)} = \begin{bmatrix} A_1^T \\ A_2^T \\ \ldots \\ A_k^T \end{bmatrix}, \tag{15}$$

and we shall denote by $\mathcal{N}_k$ the null space of $A^{(k)}$ (with the notations from (2) we have $\mathcal{N}_k = S_1 \cap \cdots \cap S_k$). The following relations are obvious

$$\{0\} = \mathcal{N}_n \subset \mathcal{N}_{n-1} \subset \cdots \subset \mathcal{N}_2 \subset \mathcal{N}_1 \tag{16}$$

and

$$dim(\mathcal{N}_k) = n - k, k = 1, \ldots, n. \tag{17}$$

In [3] (see also [4]) the author considers a set of *null vectors*

$$\mathcal{N} = \{z^2, z^3, \ldots, z^n\} \subset I\!R^n \tag{18}$$

such that (compare with (7))

$$z^k \neq 0, z^k \in \mathcal{N}_{k-1}, z^k \notin \mathcal{N}_k, \forall k = 2, \ldots, n, \tag{19}$$

i.e.

$$\langle z^k, A_j \rangle = 0, \forall j = 1, \ldots, k-1 \text{ and } \langle z^k, A_k \rangle \neq 0. \tag{20}$$

If such a set of null vectors (18)-(20) is available, we can compute x^* from (1) by the following direct solver (DS).

Algorithm DS.

Step 0 (Initialization): any vector $x \in H_1$

Step 1 (Successive projections):
for $i = 2 : n$

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

end

Remark 2 *We shall denote by $e^1, e^2, \ldots, e^n$ the cannonical basis in $I\!R^n$. Then, a possible choice for x in the above Step 0 can be (see [4])*

$$x = \frac{b_1}{A_{1k_0}} e^{k_0}, \text{ where } A_{1k_0} \neq 0. \tag{21}$$

From (19) and Lemma 1 we obtain the following result from [3].

Proposition 2 *The algorithm DS gives us the unique solution x^* from (1).*

In order to construct the above set of null vectors, the authors proposed in [4] the following Null Vectors (NV) algorithm.

Algorithm NV.

Step 1 (Initialization): any basis $\mathcal{Z}_1 = \{z_2^{(1)}, \ldots, z_n^{(1)}\}$ in $\mathcal{N}_1$

Step 2 Because $dim(\mathcal{N}_2) = n - 2$ we must have $\langle A_2, z_i^{(1)} \rangle \neq 0$, for some $i \in \{2, 3, \ldots, n-1, n\}$, so we can permute the elements of $\mathcal{Z}_1$ such that $\langle A_2, z_2^{(1)} \rangle \neq 0$. Moreover, for the numerical stability of the algorithm we may choose

$$0 < |\langle A_2, z_2^{(1)} \rangle| = \max_{2 \leq i \leq n} |\langle A_2, z_i^{(1)} \rangle|. \tag{22}$$

Then, we produce the new set $\mathcal{Z}_2 = \{z_3^{(2)}, \ldots, z_n^{(2)}\}$ by using directional projections

$$z_j^{(2)} = P_{H_2}^{z_2^{(1)}}(z_j^{(1)}) = z_j^{(1)} - \frac{\langle z_j^{(1)}, A_2 \rangle}{\langle z_2^{(1)}, A_2 \rangle} z_2^{(1)}, \ j = 3, \ldots, n. \tag{23}$$

$$\vdots$$

Step n Because $dim(\mathcal{N}_{n-1}) = 1$ we must have $\langle A_{n-1}, z_i^{(n-2)} \rangle \neq 0$, for some $i \in \{n-1, n\}$, so we can permute the elements of $\mathcal{Z}_{n-2} = \{z_{n-1}^{(n-2)}, z_n^{(n-2)}\}$ such that $\langle A_{n-1}, z_{n-1}^{(n-2)} \rangle \neq 0$. Moreover, for the numerical stability of the algorithm we may choose

$$0 < |\langle A_{n-1}, z_{n-1}^{(n-2)} \rangle| = \max_{n-1 \leq i \leq n} |\langle A_{n-1}, z_i^{(n-2)} \rangle|. \tag{24}$$

Then, we produce the last set $\mathcal{Z}_{n-1} = \{z_n^{(n-1)}\}$ by

$$z_n^{(n-1)} = P_{H_{n-1}}^{z_{n-1}^{(n-2)}}(z_n^{(n-2)}) = z_n^{(n-2)} - \frac{\langle_n^{(n-2)}, A_{n-1} \rangle}{\langle z_{n-1}^{(n-2)}, A_{n-1} \rangle} z_{n-1}^{(n-2)}. \tag{25}$$

Step n+1 The set $\mathcal{Z}$ of null vectors is given by a last directional projection

$$\mathcal{Z} = \{z_2^{(1)}, z_3^{(2)}, \ldots, z_n^{(n-1)}\}. \tag{26}$$

Remark 3 *Because we start with a basis $\mathcal{Z}_1$ in $\mathcal{N}_1$, it can be proved that the vectors from $\mathcal{Z}_2, \ldots, \mathcal{Z}_{n-1}$ are also basis in $\mathcal{N}_2, \ldots, \mathcal{N}_{n-1}$, respectively (for the proof see [4]).*

Remark 4 *Following the ideas from [4], a basis $\mathcal{Z}_1 = \{z_2^{(1)}, \ldots, z_n^{(1)}\}$ in $\mathcal{N}_1$ can be constructed as indicated in the following Matlab code.*

```
Given A: n x n;
I=eye(n);z=zeros(n);
[A1k,k]=max(abs(A(1,:)));z(:,1)=I(:,k);
if (k==1)
   for i=1:n-1
    z(:,i+1)=I(:,i+1)-(A(1,i+1)/A(1,k))*I(:,k);
   end
else
    for i=1:k-1
    z(:,i+1)=I(:,i)-(A(1,i)/A(1,k))*I(:,k);
    end
   for i=k:n-1
    z(:,i+1)=I(:,i+1)-(A(1,i+1)/A(1,k))*I(:,k);
   end
end
```

We shall call the above NV algorithm together with the solution part DS the Null Space algorithm (NS). It can be written in the following Matlab-like form.
Algorithm NS.

Step 1 (Initialization): any basis $\mathcal{Z}_1 = \{z^2, \ldots, z^n\}$ in $\mathcal{N}_1$

Step 2 for i=2:n-1

$maxim = \max_{i \leq j \leq n} |\langle A_i, z^j \rangle| = |\langle A_i, z^{j^*} \rangle|$.
$p(i) = \langle A_i, z^{j^*} \rangle$

if ($j^* > i$) interchange the vectors z^i and z^{j^*}

for j=i+1:n

$z^j = P_{H_i}^{z^i}(z^j) = z^j - \frac{\langle z^j, A_i\rangle}{p(i)} z^i$

end

end

Step 3 (Initialization): any vector $x \in H_1$

Step 4 (Successive projections):

for $i = 2 : n$

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

end

3 Consistent Rectangular Systems

For A a general $m \times n$ matrix and $b \in \mathcal{R}(A) \subset I\!R^m$ we shall consider the consistent linear system of equations

$$Ax = b, \tag{27}$$

and denote by $S(A;b), x_{LS}$ the set of its solutions and the (unique) minimal norm one. Moreover, we shall suppose that the rows A_i and columns A^j of A satisfy the assumptions

$$A_i \neq 0,\ i = 1, \ldots, m,\ A^j \neq 0,\ j = 1, \ldots, n. \tag{28}$$

3.1 The Direct Kaczmarz Algorithm

Let $f_i(b;\cdot), F(b;\cdot), P_i, Q : I\!R^n \to I\!R^n$, be the applications defined by

$$f_i(b;x) = x - \frac{\langle x, A_i\rangle - b_i}{\| A_i \|^2} A_i,\ F(b;x) = (f_1 \circ \cdots \circ f_m)(b;x), \tag{29}$$

$$P_i = I - \frac{1}{\| A_i \|^2} A_i A_i^T, \quad Q = P_1 P_2 \ldots P_m, \tag{30}$$

and R the real $n \times m$ matrix of which the i-th column is

$$R^i = \frac{1}{\| A_i \|^2} P_1 P_2 \ldots P_{i-1}(A_i),\ i = 1, \ldots, m, \tag{31}$$

with P_0= identity.

Algorithm Kaczmarz (K). *Initialization:* $x^0 \in I\!R^n$

Iterative step:

$$x^{k+1} = F(b;x^k) = (f_1 \circ \cdots \circ f_m)(b;x^k) = Qx^k + Rb. \tag{32}$$

The following result gives information about its convergence properties (see [18]).

Proposition 3 *For any matrix A satisfying (28) and any vector $b \in I\!R^m$ the following are true :*
(i) the sequence $(x^k)_{k\geq 0}$ generated with the algorithm K converges and

$$\lim_{k\to\infty} x^k = P_{\mathcal{N}(A)}(x^0) + x^*, \tag{33}$$

with $x^* \in \mathbb{R}^n$ independent on x^0.
(ii) If $b \in \mathcal{R}(A)$, then

$$x^* = x_{LS} \text{ and } \lim_{k\to\infty} x^k \in S(A;b) = \{P_{\mathcal{N}(A)}(x^0) + x_{LS}, x^0 \in \mathbb{R}^n\}. \tag{34}$$

In spite of its iterative behavior, Kaczmarz's algorithm becomes a direct solver under specific assumptions on the system matrix, as pointed-out in the result below.

Proposition 4 *If the matrix Q from (30) satisfies*

$$Q(A_i) = 0, \forall\ i = 1, \ldots, m, \tag{35}$$

then, for any $x^0 \in \mathbb{R}^n$, the first approximation, x^1 generated with (32) is given by

$$x^1 = P_{\mathcal{N}(A)}(x^0) + x_{LS} \in S(A;b), \tag{36}$$

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

Proof. Any $z \in \mathcal{R}(A^T)$ can be written as $z = \sum_{i=1}^m \alpha_i A_i$, with some $\alpha_i \in \mathbb{R}$; thus, from (35) we get $Qz = 0$. In particular we obtain $Qx_{LS} = 0$, and by also using the identity $I = Q + RA$ and the consistency of (16), which gives us $Ax_{LS} = b$, we get

$$Rb = RAx_{LS} = (I - Q)x_{LS} = x_{LS}. \tag{37}$$

From the property $x \in \mathcal{N}(A) \Rightarrow Qx = x$ (see e.g. [18]) it results for any $x^0 \in \mathbb{R}^n$, $Qx^0 = QP_{\mathcal{N}(A^T)}(x^0) = P_{\mathcal{N}(A)}(x^0)$. Using these properties, (37) and (32) for $k = 0$ we obtain

$$x^1 = Qx^0 + Rb = P_{\mathcal{N}(A)}(x^0) + Rb = P_{\mathcal{N}(A)}(x^0) + x_{LS} \in S(A;b), \tag{38}$$

and the proof is complete. ♠

Remark 5 *The property (36) was observed in [18] for a particular class of matrices; indeed, if the rows of A are mutually orthogonal, i.e.*

$$\langle A_i, A_j \rangle = 0, \forall i \neq j, \tag{39}$$

according to (30) we obtain

$$P_i(A_j) = A_j, \forall i \neq j, \tag{40}$$

which combined with the equalities

$$P_i(A_i) = 0, \forall i \tag{41}$$

and with the definition of Q in (30) gives us (35).

It is clear that conditions like (39) are almost impossible to be fulfilled for a given matrix A coming from a practical problem. Then, we shall try to directly obtain (35) by introducing some new hyperplanes for projection. This approach is different than the one used in GRP algorithm from Section 1, in which the projections are made on the initial system hyperplanes H_i, but parallel with some new constructed directions z^i. The main idea of this procedure is described below. For $d \in \mathbb{R}^n, d \neq 0$ a given direction, we consider the projection onto the vector subspace $S_d = \{z \in \mathbb{R}^n, \langle z, d\rangle = 0\}$ given by (see (30))

$$P^d(x) = x - \frac{\langle x, d\rangle}{\langle d, d\rangle} d. \tag{42}$$

Proposition 5 *([15]) Let d and $Q^d : I\!R^n \longrightarrow I\!R^n$ be defined by*

$$d = P_m(A_{m-1}), \quad Q^d = P_1 \dots P_{m-1} P^d P_m, \tag{43}$$

where

$$P^d(x) = \begin{cases} x - \frac{\langle x,d\rangle}{\langle d,d\rangle} d, & d \neq 0 \\ x, & d = 0. \end{cases} \tag{44}$$

Then

$$Q^d(x) = 0, \forall x \in \{A_m, d, A_{m-1}\}. \tag{45}$$

Proof. From (44) we get $P^d(d) = 0$, thus

$$Q^d(A_m) = 0. \tag{46}$$

Then, using (43) and again $P^d(d) = 0$ we obtain

$$Q^d(A_{m-1}) = P_1 \dots P_{m-1} P^d(P_m(A_{m-1})) = P_1 \dots P_{m-1} P^d(d) = 0. \tag{47}$$

If $d \neq 0$, from (43) and (30) we have

$$d = A_{m-1} - \frac{\langle A_{m-1}, A_m\rangle}{\langle A_m, A_m\rangle} A_m, \tag{48}$$

which together with (46), (47) and because Q^d is a linear application give us $Q^d(d) = 0$. The case $d = 0$ is obvious. ♠

Remark 6 *We have to observe that such a result is no more true for the ω - Kaczmarz algorithm from [9]. Indeed, in this algorithm we use instead of P_i and Q from (30) the matrices P_i^ω, Q^ω defined by*

$$P_i^\omega(x) = x - \omega \frac{\langle x, A_i\rangle}{\| A_i \|^2} A_i, \quad Q^\omega = P_1^\omega \circ P_2^\omega \circ \dots \circ P_m^\omega. \tag{49}$$

The properties from Remark 5 are no more true because, for two rows satisfying (39) we get

$$P_i^\omega(A_j) = \begin{cases} (1-\omega)A_i, & j = i \\ A_j, & j \neq i. \end{cases} \tag{50}$$

Moreover, if we extend the definition in (42) to

$$P^{d,\omega}(x) = x - \omega \frac{\langle x, d\rangle}{\langle d, d\rangle} d, \tag{51}$$

and define $d^\omega, Q^{d,\omega}$ by (see (53) and (43)) $d = P_m^\omega(A_{m-1})$, $Q^{d,\omega} = P_1^\omega \dots P_{m-1}^\omega P^{d,\omega} P_m^\omega$ we do no more obtain equalities as in (45) because

$$P_m^\omega(A_m) = (1-\omega)A_m, \quad P_i^{d,\omega} P_m^\omega(A_{m-1}) = \begin{cases} 0, & \omega = 1 \\ (1-\omega)d, & \omega \neq 1. \end{cases} \tag{52}$$

Proposition 6 *Let $d_1, \dots, d_{m-1} \in I\!R^n$, $i = 1, \dots, m-1$ be recursively defined by*

$$\begin{cases} d_{m-1} = & P_m(A_{m-1}) \\ d_{m-2} = & P_{m-1} P^{d_{m-1}} P_m(A_{m-2}) \\ \dots & \dots \\ d_1 = & P_2 P^{d_2} \dots P_{m-1} P^{d_{m-1}} P_m(A_1) \end{cases} \tag{53}$$

If $\bar{Q} : I\!R^n \longrightarrow I\!R^n$ *is the operator*

$$\bar{Q} = P_1 P^{d_1} P_2 P^{d_2} \dots P_{m-1} P^{d_{m-1}} P_m, \tag{54}$$

where $P^{d_i} = I$ *if* $d_i = 0$ *(see (44)), we have*

$$\bar{Q}(A_i) = 0, i = 1, \dots, m, \ \bar{Q}(d_i) = 0, i = 1, \dots, m-1. \tag{55}$$

Proof. For d_{m-1} defined in (53) we get from Proposition 5 that

$$\bar{Q}(A_m) = \bar{Q}(d_{m-1}) = \bar{Q}(A_{m-1}) = 0. \tag{56}$$

Now, from the definition of d_{m-2} in (53), and (54) we obtain

$$\bar{Q}(A_{m-2}) = [\dots] P_{m-1} P^{d_{m-1}} P_m (A_{m-2}) = [\dots] P^{d_{m-2}} (d_{m-2}) = 0. \tag{57}$$

If $d_{m-2} = 0$, then $\bar{Q}(d_{m-2}) = 0$ because $\bar{Q}$ is linear. If $d_{m-2} \neq 0$, from (53), (42) and (30) we get that it is a linear combination of the directions involved in its definition, i.e.

$$d_{m-2} = A_{m-2} + \alpha A_{m-1} + \beta A_m + \gamma d_{m-1}. \tag{58}$$

Thus, from (56) - (58) and again the linearity of $\bar{Q}$ we get $\bar{Q}(d_{m-2}) = 0$, i.e. finally

$$\bar{Q}(A_m) = \bar{Q}(d_{m-1}) = \bar{Q}(A_{m-1}) = \bar{Q}(d_{m-2}) = \bar{Q}(A_{m-2}) = 0.$$

A recursive argument will complete the proof. ♠

Remark 7 *An initial version of the above proof, but without considering the (possible in practice; see Proposition 7 below) case* $d_{m-i} = 0$ *was given in [15].*

Proposition 7 *Let the index* $i \in \{1, \dots, m-1\}$ *be arbitrary fixed. Then*

$$d_{m-i} = 0 \text{ if and only if } A_{m-i} \in sp\{A_m, A_{m-1}, A_{m-i+1}\}, \tag{59}$$

i.e., for a "low rank" matrix A many of the new directions will vanish.

Proof. The "if" part. For $i = 1, A_{m-1} \in sp\{A_m\}$ implies that $A_{m-1} = c_m A_m$, for some $c_m \in I\!R$. We then get that $d_{m-1} = 0$ from (41) and (53).
For $i = 2, A_{m-2} \in sp\{A_m, A_{m-1}\}$ implies that $A_{m-2} = c_m A_m + c_{m-1} A_{m-1}$, for some $c_m, c_{m-1} \in I\!R$. Then, from (41), (53), (42) and the linearity of the projection operators P_i we successively get

$$d_{m-2} = P_{m-1} P^{d_{m-1}} P_m (c_m A_m + c_{m-1} A_{m-1}) = c_{m-1} P_{m-1} P^{d_{m-1}} P_m (A_{m-1}) =$$

$$c_{m-1} P_{m-1} P^{d_{m-1}} (d_{m-1}) = 0.$$

In general, if we denote by $\mathcal{P}_{m-i}$ the operator in (53), i.e.

$$d_{m-i} = \mathcal{P}_{m-i}(A_{m-i}) = P_{m-i+1} P^{d_{m-i+1}} P_{m-i+2} P^{d_{m-i+2}} \dots P_{m-1} P^{d_{m-1}} P_m (A_{m-i}), \tag{60}$$

a simple recursive argument gives us

$$\mathcal{P}_{m-i}(A_j) = 0, \ \forall j = m-i+1, \dots, m \tag{61}$$

and completes this part of the proof.
The "only if" part. It results directly from (53) that the direction d_{m-i} has the form

$$d_{m-i} = A_{m-i} + c_{m-i+1}A_{m-i+1} + \cdots + c_m A_m, \tag{62}$$

for some $c_{m-i+1}, \ldots, c_m \in I\!R$. Thus, for $d_{m-i} = 0$, A_{m-i} will be a linear combination of the others rows $A_{m-i+1}, \ldots, A_m$ which completes the proof. ♠

From (62) it results that the new directions $d_{m-1}, d_{m-2}, \ldots, d_1$ are linear combinations of some of the rows of A,

$$d_{m-k} = A_{m-k} + \alpha_{m-k+1}A_{m-k+1} + \cdots + \alpha_m A_m,\ k = 1, 2, \ldots, m-1. \tag{63}$$

Then, let $b(d_{m-k}) \in I\!R$ be given by

$$b(d_{m-k}) = b_{m-k} + \alpha_{m-k+1}b_{m-k+1} + \cdots + \alpha_m b_m,\ k = 1, 2, \ldots, m-1. \tag{64}$$

In particular, for a consistent system (27) we get

$$d_{m-k} = 0 \quad \Rightarrow \quad b(d_{m-k}) = 0. \tag{65}$$

If we define the new projections $f^{d_i}(b;\cdot) : I\!R^n \longrightarrow I\!R^n, i = 1, \ldots, m-1$ by (see (29))

$$f^{d_i}(b;x) = \begin{cases} x - \frac{\langle x, d_i\rangle - b(d_i)}{\langle d_i, d_i\rangle} d_i, & d_i \neq 0, \\ x, & d_i = 0. \end{cases} \tag{66}$$

we may extend $F(b;\cdot)$ from (29) to $\bar{F}(b;\cdot) : I\!R^n \longrightarrow I\!R^n$ as

$$\bar{F}(b;x) = f_1 \circ f^{d_1} \circ f_2 \circ f^{d_2} \circ \cdots \circ f_{m-1} \circ f^{d_{m-1}} \circ f_m(b;x), \tag{67}$$

and we obtain the Direct version of the algorithm K.
Algorithm Direct Kaczmarz (DK) *Initialization:* $x^0 \in I\!R^n$
Iterative step:

$$x^{k+1} = \bar{F}(b;x^k). \tag{68}$$

Theorem 2 *For any matrix A satisfying (28), if $b \in \mathcal{R}(A)$, then for any $x^0 \in I\!R^n$ the algorithm DK produces in one iteration the vector $x^1 = P_{\mathcal{N}(A)}(x^0) + x_{LS} \in S(A;b)$.*

Remark 8 *The coefficients in the linear combination giving $b(d_i)$ (see (64)) and the application of the algorithm DK can be performed in the same time with the construction of the new directions $d_i, i = m-1, \ldots, 1$ (see the Matlab code below).*

```
[m n]=size(A);
mm=2*m-1 = the total number of directions ($m$ old and $m - 1$ new)
B=zeros(mm,n);c=zeros(mm,1): auxiliary matrices
B(1,:)=A(m,:);c(1)=b(m);xkacz=zeros(n,1);al=B(1,:);norlin=al*al';al=B(1,:);
s=(al*xkacz - c(1))/norlin;xkacz=xkacz-s*al';
for ii=1:m-1
    i=m-ii;j=2*(m-i)-1;d=A(i,:);beta=b(i);
    for q=1:j
       s=(d*B(q,:)')/(B(q,:)*B(q,:)');d=d-s*B(q,:); beta=beta-s*c(q);
    end
    %%%%%%   Two new directions
```

```
    B(2*(m-i),:)=d; B(2*(m-i)+1,:)=A(i,:);
    %%%%%%    The corresponding two new components of the right hand side
    c(2*(m-i))=beta; c(2*(m-i)+1)=b(i);
    %%%%%   Kaczmarz projections w.r.t. the two new directions
    for k1=1:2
        k=k1-1; al=B(2*(m-i)+k,:);norlin=al*al';bl=c(2*(m-i)+k);
       s=(al*xkacz - bl)/norlin;xkacz=xkacz-s*al';
    end
end
```

Remark 9 *What happens if we have*

$$d_{m-1} = d_{m-2} = \cdots = d_1 = 0 \quad ? \tag{69}$$

First of all, because the system (27) is consistent and from (63)-(64) we get

$$b(d_{m-i}) = 0, i = 1, \ldots, m-1. \tag{70}$$

Then, according to (48), (58), (28) and by a recursive argument using (53) we obtain

$$A_{m-i} = c_{m-i}A_m, c_{m-i} \in I\!R, c_{m-i} \neq 0, \quad i = 1, \ldots, m-1, \tag{71}$$

i.e. $rank(A) = 1$. Thus, using again the consistency of (27)

$$b(d_{m-i}) = c_{m-i}b_m, i = 1, \ldots, m-1. \tag{72}$$

From (66) it results

$$\bar{F}(b; x) = f_1 \circ f_2 \circ \cdots \circ f_{m-1} \circ f_m(b; x) = F(b; x), \quad x \in I\!R^n \tag{73}$$

and from (29), (71) and (72)

$$f_{m-i}(b; x) = x - \frac{\langle A_{m-i}, x\rangle - b_{m-i}}{\langle A_{m-i}, A_{m-i}\rangle} A_{m-i} = f_m(b; x) = P_{H_m}(x), \quad i = 1, \ldots, m-1. \tag{74}$$

Thus, from (73) and (74) we get the equalities

$$\bar{F}(b; x) = f_1 \circ f_2 \circ \cdots \circ f_{m-1} \circ f_m(b; x) = f_m(b; x) = P_{H_m}(x). \tag{75}$$

Now, (71)-(70) give us

$$S(A; b) = \{x \in I\!R^n, \langle x, A_m\rangle = b_m\} = H_m, \tag{76}$$

and

$$P_{H_m}(x) = \left(x - \frac{\langle A_m, x\rangle}{\langle A_m, A_m\rangle} A_m\right) + \frac{b_m}{\langle A_m, A_m\rangle} A_m = P_{N(A)}(x) + x_{LS}, \tag{77}$$

i.e. the result from Theorem 2 is still valid.

Remark 10 *The algorithm DK for computing x_{LS} (together with the construction of the new directions in (53)) in the case $b \in \mathcal{R}(A)$ needs $\mathcal{O}(nm^2)$ arithmetic operations. These values are of the same order as those presented in ([10]) (page 175, Table 6.5-1 and page 177, Table 6.5-2) for some classes of direct methods based on QR or Singular Value Decomposition. More than that, we must observe that our algorithm is independent on the fact that A has or not full-rank (which is an essential problem for the above mentioned methods) and is easier to be programmed (this because it does not require the whole matrix in the memory, but successively its rows, as any row action algorithm; see e.g. [8]).*

3.2 Generalized Null Space Algorithm

We propose for the numerical solution of the system (27) the following extension of the algorithm Null Space from section 2.2.

Algorithm General Null-Space (GNS)

Step 1 (Initialization): any basis $\mathcal{Z}_1 = \{z^2, \ldots, z^n\}$ in $\mathcal{N}_1$; $\tau = 2$

Step 2 for i=2:m

if $(maxim \neq 0)$

$maxim = \max_{\tau \leq j \leq n} |\langle A_i, z^j \rangle| = |\langle A_i, z^{j^*} \rangle|$.

$p(i) = \langle A_i, z^{j^*} \rangle$

if $(j^* > \tau)$ interchange the vectors z^τ and z^{j^*}

for j=τ+1:n

$z^j = P^{z^\tau}_{H_i}(z^j) = z^j - \frac{\langle z^j, A_i \rangle}{p(i)} z^\tau$

end

$\tau = \tau + 1$

else $p(i) = 0$

end

Step 3 (Initialization): any vector $x \in H_1$; set $\tau = 2$

Step 4 for i=2:m

if $(p(i) \neq 0)$

$x = P^{z^\tau}_{H_i}(x) = x - \frac{\langle x, A_i \rangle - b_i}{p(i)} z^\tau$

$\tau = \tau + 1$

end

end

We shall analyse in what follows the properties of the algorithm GDPM. Let

$$r = rank(A) \geq 1 \text{ and } 1 = i_1 < i_2 < \ldots i_r \leq m \tag{78}$$

indices of a set of linearly independent rows $A_{i_1}, \ldots, A_{i_r}$, i.e.

for $i \in [i_1 + 1, i_2 - 1]$, A_i depends linearly on A_{i_1}

for $i \in [i_2 + 1, i_3 - 1]$, A_i depends linearly on A_{i_1}, A_{i_2}

. . . . . .

for $i \in [i_{r-1} + 1, i_r - 1]$, A_i depends linearly on $A_{i_1}, \ldots A_{i_{r-1}}$

for $i \in [i_r + 1, m]$, A_i depends linearly on $A_{i_1}, \ldots A_{i_r}$

We shall use in what follows the notation $z^{(k)}_q$ for the vectors generated during the algorithm GNS (as in NS, section 2.2). The "evolution" of these vectors during the algorithm is as follows.

for $i \in [i_1 + 1, i_2 - 1]$, $\{z^{(1)}_2, \ldots, z^{(1)}_n\} \perp A_{i_1} = A_1$

for $i \in [i_2, i_3 - 1]$, $\{z_3^{(2)}, \ldots, z_n^{(2)}\} \perp A_{i_1}, A_{i_2}$

. . . . . .

for $i \in [i_{r-1}, i_r - 1]$, $\{z_r^{(r-1)}, \ldots, z_n^{(r-1)}\} \perp A_{i_1}, A_{i_2}, \ldots, A_{i_{r-1}}$

for $i \in [i_r, m]$, $\{z_{r+1}^{(r)}, \ldots, z_n^{(r)}\} \perp A_{i_1}, A_{i_2}, \ldots, A_{i_r} \Leftrightarrow$

$$\{z_{r+1}^{(r)}, \ldots, z_n^{(r)}\} \perp A_1, A_2, \ldots, A_m \Leftrightarrow \{z_{r+1}^{(r)}, \ldots, z_n^{(r)}\} \subset \mathcal{N}(A) \tag{79}$$

Moreover, it is clear from the algorithm GNS that

$$p(i) \neq 0 \;\Leftrightarrow\; i \in \{i_1, i_2, \ldots, i_r\} \tag{80}$$

and

$$z_{k+1}^{(k)} \perp A_{i_1}, \ldots, A_{i_k},\ k = 1, \ldots, r-1. \tag{81}$$

From all the above considerations we derive the following result for GNS.

Proposition 8 *(i) The vectors $\{z_{r+1}^{(r)}, \ldots, z_n^{(r)}\}$ form a basis in $\mathcal{N}(A)$.*
(ii) In the steps 3 and 4 of GNS we obtain a solution of (27) (which generally differs from x_{LS}.)

Proof. (i) From (78) we get that $dim(\mathcal{N}(A)) = n - r$. As in [4] we obtain that, during the construction from the algorithm GNS, the vectors $z_{k+1}^k, \ldots, z_n^k$ are linearly independent, $\forall k = 1, \ldots, r$. In particular, the $n - r$ vectors in the last obtained set $\{z_{r+1}^{(r)}, \ldots, z_n^{(r)}\}$ will form a basis $\mathcal{N}(A)$.
(ii) According to (78) the consistent problem (27) is equivalent with

$$\langle A_{i_k}, x \rangle = b_{i_k},\quad \forall k = 1, \ldots, r. \tag{82}$$

Moreover, from all the above notations and considerations, it results that Steps 3 and 4 in the algorithm GNS can be (formally) written as

Step 3-1 (Initialization): any vector $x \in H_1$

Step 4-1 for k=2:r

$i = i_k$

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

end

end

By using (82), the definition of H_i in (2), and the arguments in [4] page 1162, we obtain that the resulting vector x in the Step 4-1 will satisfy

$$x \in H_{i_k},\ \forall k = 1, \ldots, r, \tag{83}$$

i.e., according to (82) $x \in S(A; b)$. ♠

4 Linear Least Squares Problems

For $A : m \times n$ and $b \in I\!R^m$ we shall consider the linear least-squares problem: find $x^* \in I\!R^n$ such that

$$\| Ax^* - b \| = \inf\{\| Ax - b \|, x \in I\!R^n\}. \tag{84}$$

Let $LSS(A; b)$ be the set of all its solutions and x_{LS} the (unique) solution of minimal norm. If $b = b_A + b_A^*$, with

$$b_A = P_{\mathcal{R}(A)}(b), \qquad b_A^* = P_{\mathcal{N}(A^T)(b)} \tag{85}$$

then the following equality is well known

$$LSS(A; b) = S(A; b_A). \tag{86}$$

We shall keep the assumptions (28) concerning the rows and columns of A.

4.1 Generalized Sloboda's Algorithm

In the same paper [17] Sloboda presents an extension of the above algorithm for a particular case of the problem (84). If B is an $n \times n$ symmetric and positive definite matrix and $c \in I\!R^n$ a given vector, he considers the global minimization problem for the quadratic functional

$$f : I\!R^n \longrightarrow I\!R^n, f(x) = \langle Bx, x\rangle - 2\langle c, x\rangle, \tag{87}$$

for which let $\hat{x}$ be the unique solution.

Algorithm Generalized Sloboda (GS). Let $x_0^{(0)}, x_0^{(1)}, \ldots, x_0^{(n)} \in I\!R^n$ as in (9) and recursively define the vectors $x_i^{(k)} \in I\!R^n$, $i = 1, \ldots, n,\ k = i, i+1, \ldots, n$ by

$$v_{i-1}^{(i)} = x_{i-1}^{(i)} - x_{i-1}^{(i-1)}, x_i^{(k)} = x_{i-1}^{(k)} + \alpha_{i-1}^{(k)} v_{i-1}^{(i)}, \tag{88}$$

where $\alpha_{i-1}^{(k)}$ is the (unique) solution of the one dimensional minimization problem

$$f(x_{i-1}^{(k)} + \alpha_{i-1}^{(k)} v_{i-1}^{(i)}) = min\{f(x_{i-1}^{(k)} + \alpha v_{i-1}^{(i)}),\ \alpha \in I\!R\}. \tag{89}$$

Remark 11 ([17]) *If the vectors $x_0^{(0)}, x_0^{(1)}, \ldots, x_0^{(n)} \in I\!R^n$ are as in Example 1, then $\alpha_{i-1}^{(k)}$ from (89) is given by*

$$\alpha_{i-1}^{(k)} = \frac{\langle b_i - x_{i-1}^{(k)}, B_i\rangle}{\langle v_{i-1}^{(i)}, B_i\rangle}. \tag{90}$$

If $n \leq m$, the matrix A in (84) has full (column) rank, and if we define $B = A^T A$, $c = A^T b$ it is well known that the global minimization problem for f from (87) is equivalent with the normal equation $A^T Ax = A^T b$ (i.e. with (84)) and $\hat{x} = x_{LS}$. In this case, because for a given $z \in I\!R^n$ we have $\langle \left(A^T A\right)_i, z\rangle = \langle A^i, Az\rangle$, the scalars $\alpha_{i-1}^{(k)}$ from (90) are given by

$$\alpha_{i-1}^{(k)} = \frac{\langle A^i, b\rangle - \langle Ax_{i-1}^{(k)}, A^i\rangle}{\langle Av_{i-1}^{(i)}, A^i\rangle}. \tag{91}$$

Moreover, the assumption (13) becomes $\langle v_{i-1}^{(i)}, (A^T A)_i\rangle = \langle Av_{i-1}^{(i)}, A^i\rangle \neq 0, \forall i = 1, \ldots, n$ and are satisfied because $A^T A$ is symmetric and positive definite.

4.2 Direct Extended Kaczmarz

Let $\varphi_j, \Phi : I\!R^m \longrightarrow I\!R^m$ be the applications defined by

$$\varphi_j(y) = y - \frac{\langle y, A^j \rangle}{\| A^j \|^2} A^j, \ \Phi(y) = (\varphi_1 \circ \cdots \circ \varphi_n)(y). \tag{92}$$

An extension to inconsistent problems like (84) of Kaczmarz's algorithm (32) has been proposed in [13].
Algorithm Extended Kaczmarz (EK).
Initialization: $x^0 \in I\!R^n, y^0 = b$
Iterative step:

$$y^{k+1} = \Phi(y^k), \ b^{k+1} = b - y^{k+1}, \ x^{k+1} = F(b^{k+1}; x^k). \tag{93}$$

The following result gives information about the convergence properties of the algorithm EK.

Proposition 9 *For any matrix A satisfying (28), any vector $b \in I\!R^m$ and any $x^0 \in I\!R^n$ the sequence $(x^k)_{k\geq 0}$ generated with the algorithm EK converges, and*

$$\lim_{k\to\infty} x^k = P_{\mathcal{N}(A)}(x^0) + x_{LS} \in LSS(A; b) = \{P_{\mathcal{N}(A)}(x^0) + x_{LS}, x^0 \in I\!R^n\}. \tag{94}$$

Proposition 10 *Let us suppose that in addition to (35) we have the properties*

$$\Phi(A^j) = 0, \ (\forall) j = 1, \ldots, n. \tag{95}$$

Then, for any $x^0 \in I\!R^n$, the first approximation, x^1 generated with the algorithm EK is given by

$$x^1 = P_{\mathcal{N}(A)}(x^0) + x_{LS} \in LSS(A; b), \tag{96}$$

i.e. EK becomes a direct solver.

Proof. According to the definition of b_A and b_A^* in (85) we get: $b_A = \sum_{j=1}^n \beta_j A^j$, with some $\beta^j \in I\!R$; thus, from (95) we get $\Phi(b_A) = 0$; moreover $\langle b_A^*, A^j \rangle = 0, \ \forall j = 1, \ldots, n$, thus by also using (92) we obtain $\varphi_j(b_A^*) = b_A^*, \ \forall j = 1, \ldots, n$, thus $\Phi(b_A^*) = b_A^*$ and $y^1 = \Phi(y^0) = \Phi(b) = b_A^*$. This gives us in the second step of (93) $b^1 = b - b_A^* = b_A$. Thus, the third step becomes Kaczmarz iteration (32) applied to the consistent system $Ax = b_A$. Then, by using Proposition 4 and the characterization (94) we get (96) and the proof is complete. ♠

Remark 12 *If also the columns of A are mutually orthogonal, i.e.*

$$\langle A^q, A^l \rangle = 0, \forall q \neq l \tag{97}$$

then, according to (92) we obtain

$$\varphi_q(A^l) = A^l, \forall q \neq l, \tag{98}$$

which combined with

$$\varphi_q(A^q) = 0, \forall q \tag{99}$$

and with the definition of Φ in (92) gives us (95).

For $\delta \in I\!R^m, \delta \neq 0$ a given directions, we consider the projection (see (42))

$$\varphi^\delta(y) = y - \frac{\langle y, \delta \rangle}{\langle \delta, \delta \rangle} \delta. \tag{100}$$

The next two results can be proved as Propositions 5 and 6 from section 3.1.

Proposition 11 *Let δ be given by*

$$\delta = \varphi_n(A^{n-1}),\ \delta \neq 0 \tag{101}$$

and $\Phi^\delta : I\!R^m \longrightarrow I\!R^m$ *be defined by*

$$\Phi^\delta = \varphi_1 \dots \varphi_{n-1} \varphi^\delta \varphi_n. \tag{102}$$

where

$$\varphi^\delta(y) = \begin{cases} y - \frac{\langle y,\delta\rangle}{\langle \delta,\delta\rangle}\delta, & \delta \neq 0 \\ y, & \delta = 0. \end{cases} \tag{103}$$

Then we have

$$\Phi^\delta(y) = 0, y \in \{A^n, \delta, A^{n-1}\}. \tag{104}$$

Proposition 12 *Let $\delta_1, \dots, \delta_{n-1} \in I\!R^m$, $j = 1, \dots, n-1$ be recursively defined by*

$$\begin{cases} \delta^{n-1} = & \varphi_n(A^{n-1}) \\ \delta^{n-2} = & \varphi_{n-1}\varphi^{\delta_{n-1}}\varphi_n(A^{n-2}) \\ \dots & \dots \\ \delta^1 = & \varphi_2\varphi^{\delta_2} \dots \varphi_{n-1}\varphi^{\delta_{n-1}}\varphi_n(A^1). \end{cases} \tag{105}$$

If $\bar{\Phi} : I\!R^m \longrightarrow I\!R^m$ is the operator

$$\bar{\Phi} = \varphi_1\varphi^{\delta_1}\varphi_2\varphi^{\delta_2} \dots \varphi_{n-1}\varphi^{\delta_{n-1}}\varphi_n, \tag{106}$$

where $\varphi^{\delta_j} = I$ if $\delta_j = 0$ (see (103)) we have

$$\bar{\Phi}(A^j) = 0, j = 1, \dots, n,\ \bar{\Phi}(\delta^j) = 0, j = 1, \dots, n-1. \tag{107}$$

With the above elements and those from section 3.1 we can construct a Direct version of EK algorithm as presented below.

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

$$y^{k+1} = \bar{\Phi}(y^k), b^{k+1} = b - y^{k+1}, x^{k+1} = \bar{F}(b^{k+1}; x^k). \tag{108}$$

Theorem 3 *For any matrix A satisfying (28), any $b \in I\!R^m$ and any $x^0 \in I\!R^n$ the algorithm DEK produces in one iteration the vector $x^1 = P_{\mathcal{N}(A)}(x^0) + x_{LS} \in LSS(A; b)$.*

Proof. It results by direct application of Propositions 10 and 12. ♠

Remark 13 *The algorithm DEK for computing x_{LS} (together with the construction of the new directions (53) and (105) needs $\mathcal{O}(nm^2 + mn^2)$ arithmetic operations (see also Remark 10).*

5 Block Versions

For

$$0 = m_0 < m_1 < m_2 < \dots < m_p = m, \tag{109}$$

we consider the following block-rows decompositions of A and b ($m_0 = 0$)

$$A = \begin{bmatrix} B_1^T \\ B_2^T \\ \dots \\ B_p^T \end{bmatrix}, B_i^T = \begin{bmatrix} A_{m_{i-1}+1}^T \\ A_{m_{i-1}+2}^T \\ \dots \\ A_{m_i}^T \end{bmatrix}, b = \begin{bmatrix} b^1 \\ b^2 \\ \dots \\ b^p \end{bmatrix}, b^i = \begin{bmatrix} b_{m_{i-1}+1} \\ b_{m_{i-1}+2} \\ \dots \\ b_{m_i} \end{bmatrix}, \tag{110}$$

with $A : m \times n, b \in I\!R^m$,

$$B_i^T : (m_i - m_{i-1}) \times n, b^i \in I\!R^{m_i - m_{i-1}}, i = 1, \dots, p, \tag{111}$$

and

$$A^T = blcol(B_1 B_2 \dots B_p). \tag{112}$$

We define the applications $\Pi_i, \mathcal{F}_i(b; \cdot), i = 1, \dots, p, \mathcal{F}(b; \cdot) : I\!R^n \longrightarrow I\!R^n$, by

$$\begin{aligned} \Pi_i &= I - B_i (B_i^T B_i)^+ B_i^T, & (113) \\ \mathcal{F}_i(b; x) &= \Pi_i(x) + B_i (B_i^T B_i)^+ b^i, & (114) \\ \mathcal{F}(b; x) &= (\mathcal{F}_1 \circ \dots \circ \mathcal{F}_p)(b; x), \quad x \in I\!R^n, & (115) \end{aligned}$$

where G^+ will denote the Moore-Penrose pseudoinverse of a matrix G (see [2]), and consider the following block version of algorithm Kaczmarz (32).

Algorithm Block Kaczmarz (BK). *Initialization:* $x^0 \in I\!R^n$
Iterative step:

$$x^{k+1} = \mathcal{F}(b; x^k). \tag{116}$$

Lemma 2 *Let $Q_i, i = 0, \dots, p-1$, Q and R be defined by*

$$Q_0 = I, Q_i = \Pi_1 \dots \Pi_i, i = 1, \dots, p-1, Q = \Pi_1 \dots \Pi_p, \tag{117}$$

$$R = blcol\left(Q_0 B_1 (B_1^T B_1)^+, \dots, Q_{p-1} B_p (B_p^T B_p)^+\right). \tag{118}$$

Then, the following equalities hold.

$$x^{k+1} = Qx^k + Rb, \; I = Q + RA, \; \mathcal{R}(R) = \mathcal{R}(A^T). \tag{119}$$

Remark 14 *A simple computation using a singular value decomposition of B_i gives us the equality*

$$B_i (B_i^T B_i)^+ = (B_i^T)^+. \tag{120}$$

Thus, the BK algorithm (116) is a particular case of the block-row SOR method (2.2) from [9] and according to Theorem 1, page 4 in this paper, if $b \in \mathcal{R}(A)$ then for any $x^0 \in \mathcal{R}(A^T)$ the sequence generated by the BK algorithm converges to x_{LS}.

Lemma 3 *(i) We have*

$$\Pi_i B_i = 0, \forall i = 1, \dots, p. \tag{121}$$

(ii) If the rows $A_{m_{i-1}+1}, \dots, A_{m_i}$ are mutually orthogonal then

$$\Pi_i x = \sum_{j=m_{i-1}+1}^{m_i} P_j x - (m_i - 1)x, \tag{122}$$

with P_j from (30).

Proof. (i) Let us denote by B one of the matrices B_i, by Π the operator $\Pi = I - B(B^TB)^+B^T$, $r = rank(B)$, $0 \leq r \leq p$ and let

$$U^TBV = \Sigma = diag(\sigma_1, \ldots, \sigma_r, 0, \ldots, 0) \tag{123}$$

a singular value decomposition of B. Then (see e.g. [2])

$$(B^TB)^+ = Vdiag(\frac{1}{\sigma_1^2}, \ldots, \frac{1}{\sigma_r^2}, 0, \ldots, 0)V^T. \tag{124}$$

From (123) and (124) we get

$$(B^TB)^+B^TB = Vdiag(1, \ldots, 1, 0, \ldots, 0)V^T, \tag{125}$$

thus

$$\Pi B = B - B(B^TB)^+B^TB = U\Sigma V^T - U\Sigma diag(1, \ldots, 1, 0, \ldots, 0)V^T =$$
$$U\Sigma V^T - U\Sigma V^T = 0$$

and the proof of the first part is complete.
(ii) From our hypothesis we obtain

$$B_i^TB_i = \begin{bmatrix} A_{m_{i-1}+1} \\ A_{m_{i-1}+2} \\ \cdots \\ A_{m_i} \end{bmatrix} col[A_{m_{i-1}+1}, \ldots, A_{m_i}] =$$

$$diag(\| A_{m_{i-1}+1} \|^2, \| A_{m_{i-1}+2} \|^2, \ldots, \| A_{m_i} \|^2).$$

Then

$$\Pi_i x = x - B_i diag(\| A_{m_{i-1}+1} \|^{-2}, \| A_{m_{i-1}+2} \|^{-2}, \ldots, \| A_{m_i} \|^{-2})B_i^Tx =$$

$$x - col[A_{m_{i-1}+1}, \ldots, A_{m_i}] \begin{bmatrix} \frac{A_{m_{i-1}+1}^T}{\|A_{m_{i-1}+1}^T\|^2} \\ \frac{A_{m_{i-1}+2}^T}{\|A_{m_{i-1}+2}^T\|^2} \\ \cdots \\ \frac{A_{m_i}^T}{\|A_{m_i}^T\|^2} \end{bmatrix} = x - \sum_{j=m_{i-1}+1}^{m_i} \frac{\langle A_j, x \rangle}{\| A_j \|^2} A_j$$

from which (122) holds by using again (30). ♠

Lemma 4 *Let us suppose that $Ax = b$ is consistent and Q from (117) satisfies*

$$QB_i = 0, i = 1, \ldots, p. \tag{126}$$

Then, the approximation x^1 generated in (116) is given by

$$x^1 = P_{\mathcal{N}(A)}(x^0) + x_{LS} \in S(A; b). \tag{127}$$

Proof. From [13], Lemmas 3.2 and 6.2 we get

$$\mathcal{N}(A) = \{x \in I\!R^n, \Pi_i x = x, \forall i = 1, \ldots, p\} \tag{128}$$

and

$$A^T = blcol(B_1, \ldots, B_p) \;\Rightarrow\; \mathcal{R}(A^T) = \mathcal{R}(B_1) + \cdots + \mathcal{R}(B_p). \tag{129}$$

From Lemma 2(ii), Lemma 4 and (126) we then obtain $x_{LS} \in \mathcal{R}(A^T)$, i.e. $x_{LS} = \sum_{j=1}^{p} B_j z_j$, $z_j \in I\!R$, thus $Rb = RAx_{LS} = (I - Q)x_{LS} = x_{LS} - \sum_{j=1}^{p} QB_j z_j = x_{LS}$. Then, from (117), (128) and (129) we get

$$Qx^0 = QP_{\mathcal{N}(A)}(x^0) + QP_{\mathcal{R}(A^T)}(x^0) = P_{\mathcal{N}(A)}(x^0),$$

thus $x^1 = Qx^0 + Rb = P_{\mathcal{N}(A)}(x^0) + x_{LS}$ and the proof is complete. ♠

Lemma 5 *Let*

$$D = \Pi_p B_{p-1}, \Pi^D = I - D(D^T D)^+ D^T, Q^D = \Pi_1 \ldots \Pi_{p-1} \Pi^D \Pi_p. \tag{130}$$

Then

$$Q^D B_p = 0,\; Q^D D = 0,\; Q^D B_{p-1} = 0. \tag{131}$$

Proof. From Lemma 3(i) we get $\Pi_p B_p = 0$, thus $Q^D B_p = 0$. Then, as in the proof of Lemma 3(i) we get $\Pi^D D = 0$ and then

$$Q^D B_{p-1} = \Pi_1 \ldots \Pi_{p-1} \Pi^D \Pi B_{p-1} = \Pi_1 \ldots \Pi_{p-1} \Pi^D D = 0.$$

Now, from (130) and the above equalities we obtain

$$Q^D D = Q^D \left[B_{p-1} - B_p (B_p^T B_p)^+ B_p^T \right] = Q^D B_{p-1} - (Q^D B_p)(B_p^T B_p)^+ B_p^T = 0$$

and the proof is complete. ♠

For a new direction D as in (130), i.e.

$$D = B_{p-1} - B_p (B_p^T B_p)^+ B_p^T B_{p-1}, \tag{132}$$

we define a corresponding right hand side $b(D)$ by

$$b(D) = b^{p-1} - \left[(B_p^T B_p)^+ B_p^T B_{p-1} \right]^T b^p \tag{133}$$

and a new application (see (114))

$$\mathcal{F}^D(b; x) = \Pi^D(x) + D(D^T D)^+ b(D). \tag{134}$$

Remark 15 *The above construction for the new component $b(D)$ in (133) was possible because of the equality (120) used in the construction of the new direction D from (132).*

Lemma 6 *The constructions (132)-(134) do not change the set of solutions $S(A; b)$ of the consistent system (27).*

Proof. According to (111) the new rows block D from (132) has dimensions $n \times (m_{p-1} - m_{p-2})$ and $b(D) \in I\!R^{m_{p-1} - m_{p-2}}$. We define an extended $(m + m_{p-1} - m_{p-2}) \times n$ matrix $\hat{A}$ and right-hand side $\hat{b} \in I\!R^{m + m_{p-1} - m_{p-2}}$ by

$$\hat{A} = \begin{bmatrix} B_1^T \\ \ldots \\ B_{p-1}^T \\ D^T \\ B_p^T \end{bmatrix}, \hat{b} = \begin{bmatrix} b^1 \\ \ldots \\ b^{p-1} \\ b(D) \\ b^p \end{bmatrix}, \tag{135}$$

and consider the extended system

$$\hat{A}x = \hat{b}. \tag{136}$$

We shall prove that (136) is consistent and

$$S(A;b) = S(\hat{A};\hat{b}). \tag{137}$$

The original system $Ax = b$ can be written as

$$B_i^T x = b^i,\ i = 1, \ldots, p. \tag{138}$$

Then, from (132), (138) and (133) we obtain

$$D^T x = B_{p-1}^T x - \left[(B_p^T B_p)^+ B_p^T B_{p-1}\right]^T B_p^T x = b^{p-1} - \left[(B_p^T B_p)^+ B_p^T B_{p-1}\right]^T b^p = b(D)$$

which completes the proof. ♠

Then, by analogy with (53) we construct a complete set of new block directions

$$\begin{cases} D_{p-1} = & \Pi_p(B_{p-1}) \\ D_{p-2} = & \Pi_{p-1}\Pi^{D_{p-1}}\Pi_p(B_{p-2}) \\ \ldots & \ldots \\ D_1 = & \Pi_2\Pi^{D_2}\ldots\Pi_{p-1}\Pi^{D_{p-1}}\Pi_p(B_1), \end{cases} \tag{139}$$

the right hand side components $b(D_i)$ (according to (139), (133) and (130)), and a new application $\widetilde{Q} : I\!R^n \longrightarrow I\!R^n$

$$\widetilde{Q} = \Pi_1\Pi^{D_1}\Pi_2\Pi^{D_2}\ldots\Pi_{p-1}\Pi^{D_{p-1}}\Pi_p. \tag{140}$$

Proposition 13 *The above application $\widetilde{Q}$ satisfies*

$$\widetilde{Q}(B_i) = 0, i = 1, \ldots, p,\ \widetilde{Q}(D_i) = 0, i = 1, \ldots, p-1. \tag{141}$$

Proof. We use the results in Lemmas 3 (i) and 5, and similar steps as in the proof of Proposition 6. ♠

Let now $\widetilde{\mathcal{F}}(b;\cdot) : I\!R^n \longrightarrow I\!R^n$ be defined by ($\mathcal{F}^D(b;x)$) from (134) and $\mathcal{F}_i(b;x)$) from (114))

$$\widetilde{\mathcal{F}}(b;x) = \mathcal{F}_1 \circ \mathcal{F}^{D_1} \circ \mathcal{F}_2 \circ \mathcal{F}^{D_2} \circ \cdots \circ \mathcal{F}_{p-1} \circ \mathcal{F}^{D_{p-1}} \circ \mathcal{F}_p(b;x). \tag{142}$$

We obtain the following direct version of the algorithm BK for the consistent system (27).
Algorithm Direct Block Kaczmarz (DBK) *Initialization: $x^0 \in I\!R^n$*
Iterative step:

$$x^{k+1} = \widetilde{\mathcal{F}}(b;x^k). \tag{143}$$

Theorem 4 *For any $x^0 \in I\!R^n$, the algorithm DBK produces in one iteration the vector $x^1 = P_{\mathcal{N}(A)}(x^0) + x_{LS} \in S(A;b)$.*

Proof. It results directly from Lemma 4 and Proposition 13. ♠

A similar block version can now be constructed for the Extended Kaczmarz algorithm (93). In this respect we consider a block row decomposition of A^T of the form (see (109) and (110))

$$0 = n_0 < n_1 < n_2 < \cdots < n_q = n, \tag{144}$$

$$A^T = \left[\begin{array}{c} C_1^T \\ C_2^T \\ \cdots \\ C_q^T \end{array} \right], C_i^T = \left[\begin{array}{c} (A^{n_{i-1}+1})^T \\ (A^{n_{i-1}+2})^T \\ \cdots \\ (A^{n_i})^T \end{array} \right], \tag{145}$$

with $C_i^T : (n_i - n_{i-1}) \times m, i = 1, \ldots, q$, and we define the applications $\Phi, \Gamma_i : I\!R^m \longrightarrow I\!R^m, i = 1, \ldots, q$ by

$$\Phi = \Gamma_1 \circ \cdots \circ \Gamma_q, \quad \Gamma_i = I - C_i (C_i^T C_i)^+ C_i^T. \tag{146}$$

Algorithm Block Extended Kaczmarz (BEK).
Initialization: $y^0 = b, x^0 \in I\!R^n$
Iterative step:

$$y^{k+1} = \Phi(y^k), \quad b^{k+1} = b - y^{k+1}, \quad x^{k+1} = \mathcal{F}(b^{k+1}; x^k), \tag{147}$$

with $\mathcal{F}$ from (115).

Remark 16 *In [13] we considered a particular case of the above BEK algorithm, in which the Moore-Penrose pseudoinverse was replaced with the classical inverse, under the supplementary assumptions*

$$det(B_i^T B_i) \neq 0, \quad i = 1, \ldots, p, \; det(C_j^T C_j) \neq 0, \quad j = 1, \ldots, q. \tag{148}$$

Under these assumptions we proved that for any $x^0 \in I\!R^n$, the sequence $(x^k)_{k \geq 0}$ generated by the algorithm BEK converges and $\lim_{k \to \infty} x^k = P_{\mathcal{N}(A)}(x^0) + x_{LS}$. The fact that this result remains valid in the above general case of the algorithm BEK is an open problem.

A direct version can now be constructed for the Block Extended Kaczmarz algorithm (147). For a new column direction $\delta : r \times m$ we shall define the application (similar with Γ_i)

$$\Gamma^{\Delta} = I - \Delta (\Delta^T \Delta)^+ \Delta^T. \tag{149}$$

The complete set of column directions will be (see (139))

$$\left\{ \begin{array}{ll} \Delta_{q-1} = & \Gamma_q(C_{q-1}) \\ \Delta_{q-2} = & \Gamma_{q-1} \Gamma^{\Delta_{q-1}} \Gamma_q(C_{q-2}) \\ \cdots & \cdots \\ \Delta_1 = & \Gamma_2 \Gamma^{\Delta_2} \ldots \Gamma_{q-1} \Gamma^{\Delta_{q-1}} \Gamma_q(C_1), \end{array} \right. , \tag{150}$$

and the corresponding column Kaczmarz application $\widetilde{\Phi} : I\!R^m \longrightarrow I\!R^m$ by (see (92))

$$\widetilde{\Phi}(y) = \Gamma_1 \circ \Gamma^{\Delta_1} \circ \Gamma_2 \circ \Gamma^{\Delta_2} \circ \cdots \circ \Gamma_{q-1} \circ \Gamma^{\Delta_{q-1}} \circ \Gamma_q(y). \tag{151}$$

Algorithm Direct Block Extended Kaczmarz (DBEK).
Initialization: $y^0 = b, x^0 \in I\!R^n$
Iterative step:

$$y^{k+1} = \widetilde{\Phi}(y^k), b^{k+1} = b - y^{k+1}, x^{k+1} = \widetilde{\mathcal{F}}(b^{k+1}; x^k). \tag{152}$$

Theorem 5 *For any $x^0 \in I\!R^n$ the algorithm DBEK produces in one iteration the vector $x^1 = P_{\mathcal{N}(A)}(x^0) + x_{LS} \in LSS(A; b)$.*

Proof. If $\widetilde{\Phi}$ is constructed as in (150)-(151) the result from Proposition 13 applies for it and we get

$$\widetilde{\Phi}(C_j) = 0, j = 1, \ldots, q, \; \widetilde{\Phi}(\Delta_j) = 0, j = 1, \ldots, q-1. \tag{153}$$

Then, according to Proposition (10) it suffices to go back at the algorithm BKE and prove that, if Q from (117) satisfies (126), and Φ from (146) satisfies

$$\Phi(C_j) = 0, j = 1, \ldots, q, \tag{154}$$

then the algorithm BKE gives us in only one iteration a solution $x^1 \in LSS(A; b)$ of the form (127). For this , as in the proof of Proposition 10 we observe that

$$b_A = \sum_{j=1}^{n} \beta_j A^j, \quad \beta_j \in I\!R \text{ and } \langle b_A^*, A^j \rangle = 0, \forall j = 1, \ldots, n. \tag{155}$$

But, for an arbitrary fixed $j \in \{1, \ldots, q\}$, from (154) and (111) we get

$$0 = \Phi(C_j) = \Phi\left[col(A^{n_{j-1}+1}, \ldots, A^{n_j})\right] = col\left(\Phi(A^{n_{j-1}+1}), \ldots, \Phi(A^{n_j})\right),$$

thus

$$\Phi(A^q) = 0, \forall q = 1, \ldots, n \tag{156}$$

which gives us

$$\Phi(b_A) = 0, \quad \Phi(b_A^*) = b_A^*, \quad y^1 = \Phi(b) = b_A^*, \quad b^1 = b - y^1 = b_A. \tag{157}$$

Thus, after the first two steps in (147) we arrive at the consistent system $Ax = b_A$ to which the BK algorithm is applied. The results will then hold by applying Lemma 4 and the proof is complete. ♠

6 Numerical Experiments

The algorithms for rectangular systems presented above (consistent or not) were implemented and applied to rigid multibody problems. Such problems consist of a set of rigid bodies which move according to the differential equations of motion. However their motions are restricted by numerous constraint equations. These constraints can be expressed in terms of the system Jacobian J and the velocities v

$$Jv = 0. \tag{158}$$

The time discretization of the equations of motions leads to

$$v = M^{-1} J^T \lambda + c, \tag{159}$$

where M is the positive-definite and block-diagonal mass matrix, λ are the unknown Lagrange multipliers and c is a constant. Inserting (159) into (158) results in a square positive-semidefinite linear system of equations

$$JM^{-1}J^T\lambda + Jc = 0. \tag{160}$$

The system (160) is consistent and thus has at least one solution. The consistency can be easily demonstrated by decomposing the inverse mass matrix into it's Cholesky factors L and L^T. Then the system can be rewritten in the following form

$$(JL)(JL)^T\lambda + (JL)L^T Mc = 0. \tag{161}$$

Hence they are the normal equations of the following system

$$(JL)^T\lambda + L^T Mc = 0. \tag{162}$$

Figure 1: A rendering of the well setup.

It is well known that the normal equations are always consistent. However system (162) can be consistent or inconsistent and the system matrix can have full column rank or not. The circumstances under which (162) is consistent can be better understood by reformulating (162) in terms of the velocities v

$$(JL)^T\lambda + L^TMc = L^TMv = 0. \tag{163}$$

Since L^TM is positive-definite, $L^TMv = 0 \leftrightarrow v = 0$. Thus (162) is consistent if the Lagrange multipliers λ exist so that any motion is prevented.

Remark 17 *In each numerical experiment we generated a block row or block column decomposition of A by using the algorithm from [14]. In this respect the blocks obtained (without possible the last (smallest) one contain mutually orthogonal rows.*

6.1 Well Test

As a consistent and rectangular test setup we chose a well structure. The well consists of 5 layers of bricks. Each layer consists of 40 bricks lined up in a circle. The bricks in subsequent layers are displaced so that the brick centers are above the gaps of the layer below. Figure 1 shows a rendering of the scene. The relative motion of neighboring bricks is constrained in the following way: The polygonal contact area is determined and at each corner of that area a joint is created. The joint constrains the relative translational motion of the neighboring bricks in all three spatial dimensions. The system is heavily overconstrained, that is there are many redundant constraints and the solution is underdetermined. However, the system is consistent which is intuitively clear since the constraints are sufficient to prevent any motion in the system. The gravity acts towards the ground opposite to the ground normal. The resulting system matrix has 1200 rows, that is because M^{-1} as well as its Cholesky factors have 6 rows and columns for each of the 200 bodies. 3 for the translational degrees of freedom and 3 for the rotational ones. The number of columns are dictated by the number of columns of J^T, which is equal to the number of Lagrange multipliers. For each joint one Lagrange multiplier is needed for constraining the translational movement in one spatial direction. Since there are 2080 joints in the well the system matrix has $3 \cdot 2080 = 6240$ columns. Figure 2 visualizes the sparsity pattern of the system matrix.

The system was solved with the Direct Kaczmarz, the Direct Block Kaczmarz and the Generalized

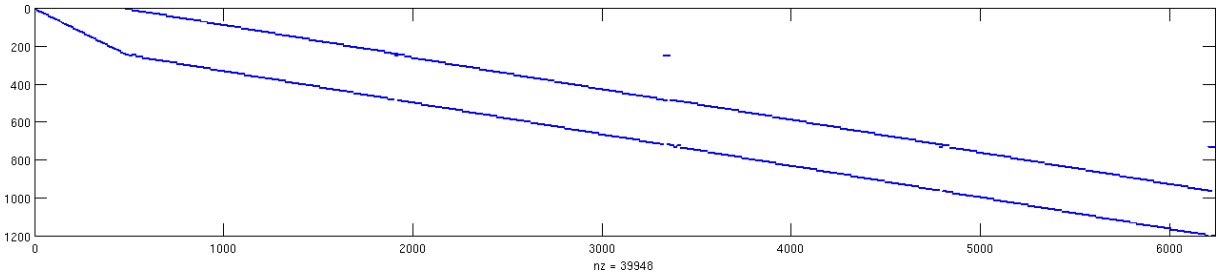


Figure 2: A visualization of the well's system matrix sparsity pattern.

Algorithm	Runtime in s	Error	Residual
Direct Kaczmarz	438.78	$1.18 \cdot 10^{-14}$	$2.86 \cdot 10^{-15}$
Direct Block Kaczmarz	9.65	$1.24 \cdot 10^{-14}$	$5.79 \cdot 10^{-15}$
Generalized Null Space	874.43	4.44	$1.01 \cdot 10^{-15}$

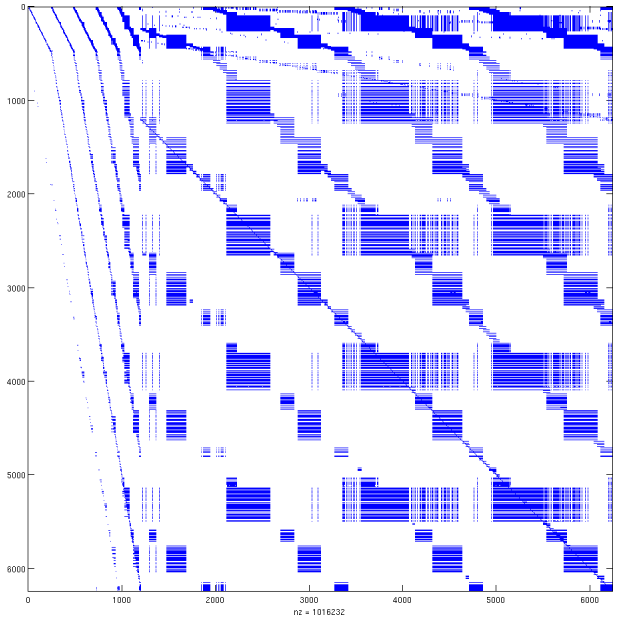
Table 1: Numerical results of the Direct Kaczmarz and Generalized Null Space methods on the well test problem.

Null Space methods. Table 1 summarizes the results. The runtimes were measured by executing the implementations in Matlab R2009b on a single core at 2.67 GHz of an Q6700 Intel® Core™2 Quad CPU. The error values are the maximum norm of the difference between the computed solution by the algorithm and the Moore-Penrose pseudoinverse, as computed by Matlab's pinv function, applied to the right-hand side. The latter is one possible minimum norm solution. Thus the Kaczmarz algorithms obtain the same minimum norm solution (except for machine precision) whereas the Generalized Null Space method gives us a different solution (see Proposition 8 (ii)). The residual column corresponds to the maximum norm of the residuals of the normal equations (161). However Direct Kaczmarz needs roughly half the runtime of the Generalized Null Space method and the block variant hugely further improves the runtime.

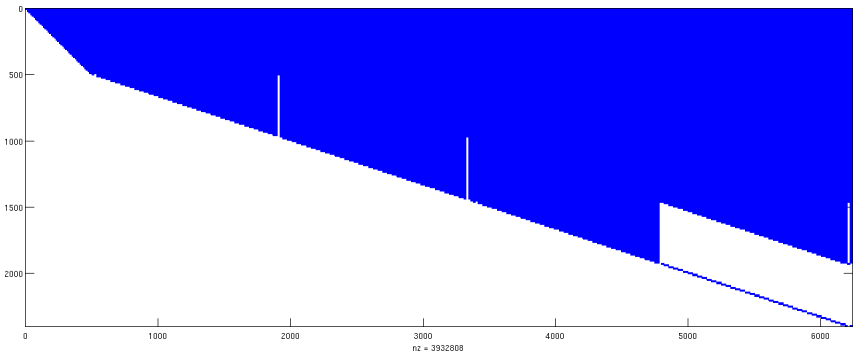
Remark 18 *Note that the measured runtime of Direct Block Kaczmarz does not include the determination of orthogonal rows here. In the same time (see Remark 17 before) in case of blocks with mutually orthogonal rows (or columns), the computations were made according to the general formulas (113) and (139), without using the much simplified version from Lemma 3 (122).*

6.2 Mobile

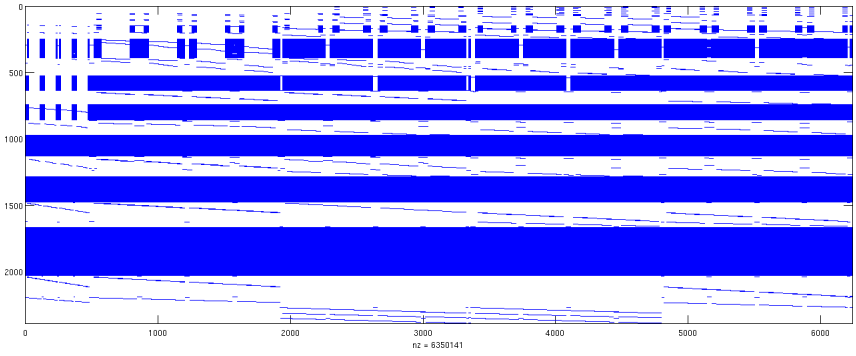
To construct a problem with full column rank the constraints must be linearly independent in order for the columns of $L^T J^T$ to be linearly independent. Note that each time constraints close a loop degrees of freedom are potentially removed from the system leading to rank deficiency. Thus we will choose a mobile hanging from the ceiling where no loop is closed as depicted in Figure 4. We modeled it solely with capsules all residing in a two dimensional plane. Our mobile has a recursion depth of 6 and thus has $1 + \sum_{i=0}^{5} 3 \cdot 2^i = 1 + 3 \cdot (2^6 - 1) = 190$ capsules. Since several zero rows have been removed from the system, the actual system matrix has only 1013 instead of $6 \cdot 190 = 1140$ rows. Joints are created at all points where capsules contact each other and additionally the topmost capsule is fixed to a ceiling. Each joint again constrains the relative translational motion of the neighboring objects in all three spatial dimensions. Thus the system matrix has $3 \cdot 190 = 570$ columns corresponding to three Lagrange multipliers per joint. Figure 5(a) visualizes the sparsity pattern of the system matrix.



(a) Sparsity pattern of the Generalized Null Space's modified well system matrix.



(b) Sparsity pattern of the Direct Kaczmarz's modified well system matrix.



(c) Sparsity pattern of the Direct Block Kaczmarz's modified well system matrix.

Figure 3: Visualizations of the sparsity patterns involved in solving the well problem.

Figure 4: A rendering of the mobile setup.

Algorithm	Runtime in s	Error	Residual
Direct Extended Kaczmarz	48.90	$1.39 \cdot 10^{-15}$	$2.42 \cdot 10^{-13}$
Direct Block Extended Kaczmarz	1.59	$8.09 \cdot 10^{-16}$	$4.40 \cdot 10^{-16}$
Generalized Sloboda	165.65	$5.40 \cdot 10^{-15}$	$5.15 \cdot 10^{-15}$

Table 2: Numerical results of the Direct Extended Kaczmarz and Generalized Sloboda methods on the mobile test problem.

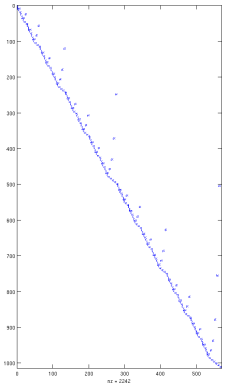
In order to ensure that the system is inconsistent motion must set in. This is *not* the case if the gravity points in the normal direction of the ceiling. Hence we chose the gravity to act in diagonal direction. The system was solved with the Generalized Sloboda, Direct Extended Kaczmarz and the Direct Block Extended Kaczmarz methods. Table 2 summarizes the results. The Direct Extended Kaczmarz finishes more than three times faster as opposed to the Generalized Sloboda solver and again the block version of Kaczmarz easily outperforms the others. All solvers attain the same minimum norm solution as Matlab and thus the error and residual norms are in the range of the machine precision.

6.3 Pyramid

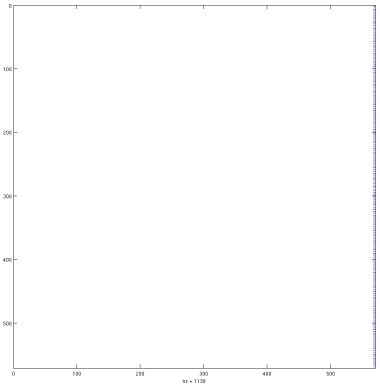
For the last test we need a setup where motion sets in so that the system is inconsistent and furthermore redundant constraints so that the system matrix is rank deficient. The following setup fulfills these requirements: We build up a pyramid of spheres with 10 layers, where the lowest layer consists of 10×10 spheres arranged in a grid. The next layer is comprised of 9×9 spheres all shifted so that they fit in between four spheres below them. If we would create joints at all points where the spheres contact, then the pyramid would remain at rest. Instead we neglect all contact points between spheres on the same layer. Furthermore we skip all joint equations, which constrain the relative translational movement in tangential directions of each contact. Thus we only have one constraint equation left per joint, which prevents the concerned spheres from changing their relative distance in the normal direction of the contact, that is penetrating each other or breaking contact. The resulting system is similar to a corresponding frictionless contact problem, which unfortunately is subject to complementarity constraints instead of equations. Gravity is assumed to point towards the ground opposite to the ground normal. Figure 6 shows a rendering of the described setup.

The system consists of $\sum_{i=1}^{10} i^2 = 385$ spheres with $10^2 + 4 \cdot \sum_{i=1}^{9} i^2 = 1240$ constraint equations. There are 1155 rows in the system, much lower than the expected $6 \cdot 385$ rows. This is again because zero rows have been removed from the system. Analysing the rank with Matlab yields a column rank of 1100. Figure 7 visualizes the sparsity pattern of the system matrix. The only solvers presented above which can cope with such a general system, are the Direct Extended Kaczmarz and Direct Block Extended Kaczmarz methods. The numerical results are summarized in Table 3. Both methods clearly return the minimum norm solution. Again the block version comes out far ahead.

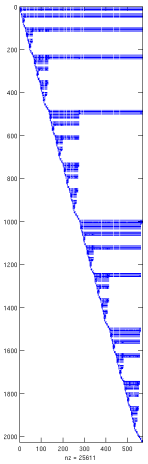
Remark 19 *Unfortunately all the above Direct Kaczmarz-like algorithms create a fill-in in the corresponding modified matrices (property which is specific for direct methods). And, although this aspect is not so unpleasant in some cases (see e.g. Figure 5(f) (c) for the Direct Kaczmarz algorithm, in which the*



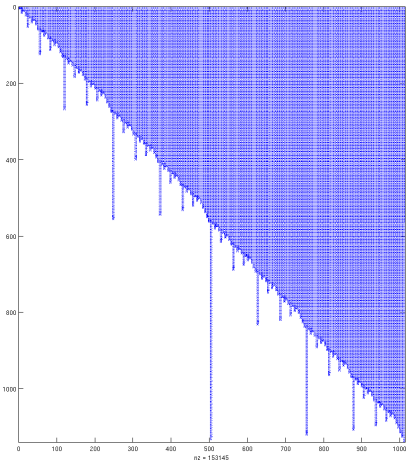
(a) Sparsity pattern of mobile's system matrix.



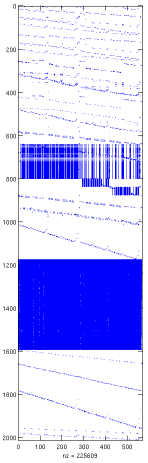
(b) Sparsity pattern of the Generalized Sloboda's modified mobile system matrix.



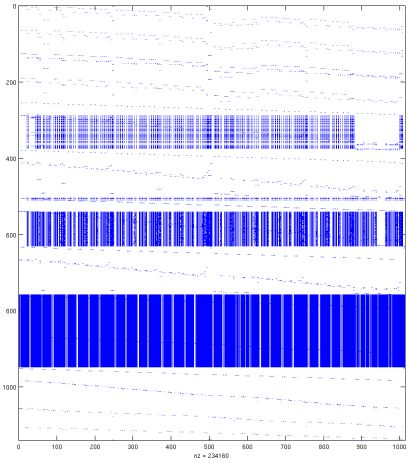
(c) Sparsity pattern of the first Direct Extended Kaczmarz's modified mobile system matrix.



(d) Sparsity pattern of the second Direct Extended Kaczmarz's modified mobile system matrix.



(e) Sparsity pattern of the first Direct Extended Block Kaczmarz's modified mobile system matrix.



(f) Sparsity pattern of the second Direct Extended Block Kaczmarz's modified mobile system matrix.

Figure 5: Visualizations of the sparsity patterns involved in solving the mobile problem.

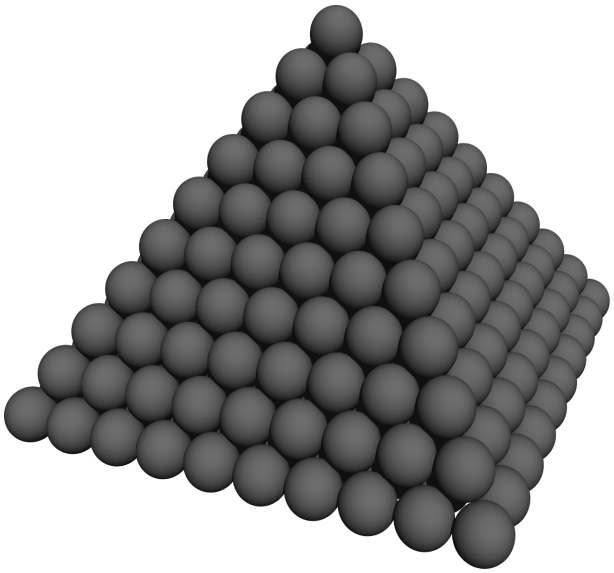


Figure 6: A rendering of the pyramid setup.

Algorithm	Runtime in s	Error	Residual
Direct Extended Kaczmarz	194.05	$1.83 \cdot 10^{-14}$	$2.01 \cdot 10^{-14}$
Direct Block Extended Kaczmarz	7.21	$9.44 \cdot 10^{-16}$	$7.03 \cdot 10^{-16}$

Table 3: Numerical results of the Direct Extended Kaczmarz methods on the pyramid test problem.

result in Proposition p-r2 applies), in other cases things dramatically change (see e.g. Figure 8(d) (a) - (d)). We can try to eliminate this bad aspect by considering an "approximate" (more sparse) construction of the new directions in (53) or block directions in (139) or (150) and use it as a preconditioner for the corresponding Kaczmarz-like algorithms applied to the extended matrix so obtained. Work is in progress in this direction.

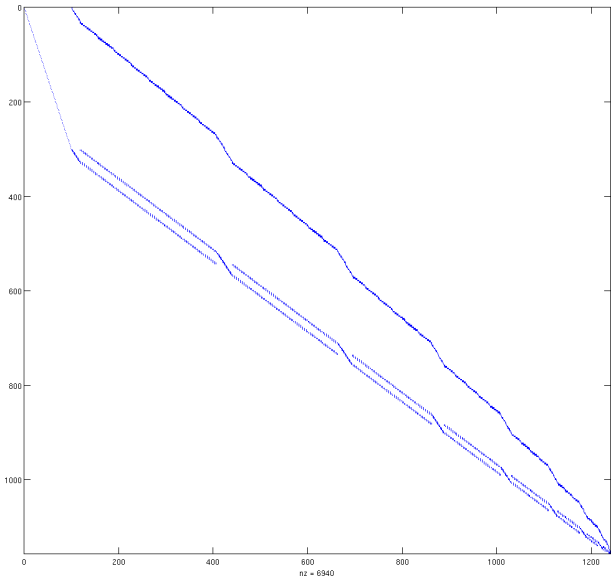


Figure 7: A visualization of the pyramid's system matrix sparsity pattern.

References

[1] Abaffy J., Broyden C., Spedicato E., *A class of direct methods for linear equations*, Numer. Math. 45(1984), 361-376.

[2] Björk A., *Numerical methods for least squares problems*, SIAM Philadelphia, 1996.

[3] Benzi M., *A direct row-projection method for sparse linear systems*, Ph.D. Thesis, Department of Mathematics, North Carolina State University, Raleigh, NC, 1993.

[4] Benzi M., Mayer C., *A direct projection method for sparse linear systems*, SIAM J. Sci. Comput., 16(5)(1995), 1159-1176.

[5] Benzi M., Mayer C., Tuma M., *A sparse approximate inverse preconditioner for the conjugate gradient method*, SIAM J. Sci. Comput., 17(5)(1996), 1135-1149.

[6] Brezinski C., *Projection methods for systems of equations*, Elsevier, Amsterdam, 1997.

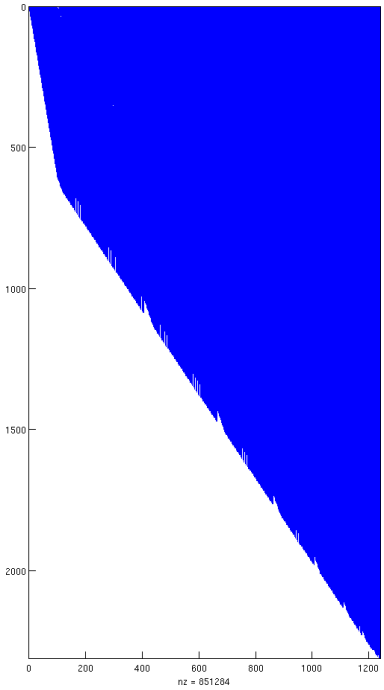
[7] Burden R.L., Faires J.D., Reynolds A.C., *Numerical analysis - second edition*, Prindle, Weber and Schmidt, Boston, Massachusetts, 1981.

[8] Censor Y., Stavros A. Z. *Parallel optimization: theory, algorithms and applications*, "Numer. Math. and Sci. Comp." Series, Oxford Univ. Press, New York, 1997.

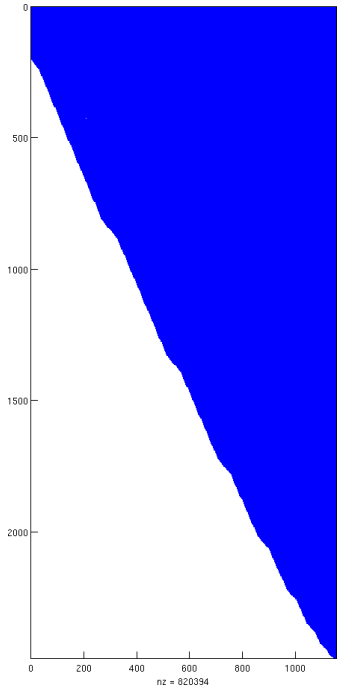
[9] Elfving T., *Block-Iterative Methods for Consistent and Inconsistent Linear Equations*, Numer. Math., **35** (1980), 1-12.

[10] Golub, G.H., van Loan, C.F., *Matrix computations*, The John's Hopkins University Press, Baltimore, 1983.

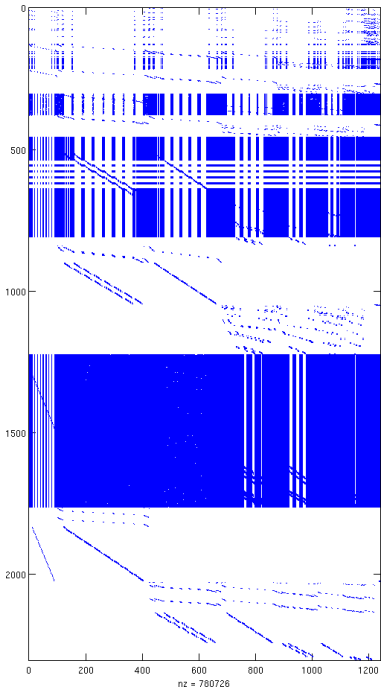
[11] Natterer F., *The Mathematics of Computerized Tomography*, John Wiley and Sons, New York, 1986.



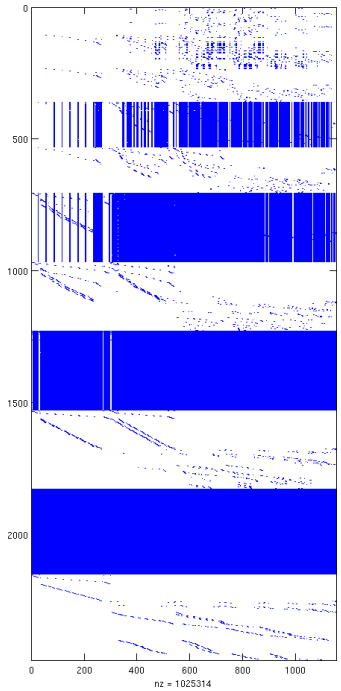
(a) Sparsity pattern of the first Direct Extended Kaczmarz's modified pyramid system matrix.



(b) Sparsity pattern of the second Direct Extended Kaczmarz's modified pyramid system matrix.



(c) Sparsity pattern of the first Direct Extended Block Kaczmarz's modified pyramid system matrix.



(d) Sparsity pattern of the second Direct Extended Block Kaczmarz's modified pyramid system matrix.

Figure 8: Visualizations of the sparsity patterns involved in solving the pyramid problem.

[12] Pietrzykowski T., *Projection method*, Zaktadu Aparatow Matematycznych Polskiej Akad. nauk Praca, A8 (1960).

[13] Popa C., *Extensions of block-projections methods with relaxation parameters to inconsistent and rank-deficient least-squares problems*;
B I T, **38(1)** (1998), 151-176.

[14] Popa C., *Block-projections algorithms with blocks containing mutually orthogonal rows and columns*; **39(2)**(1999), 323-338.

[15] Popa C., *Direct and iterative Kaczmarz-like solvers*; Annals of the West University of Timisoara, Series Mathematics, **XL(2)**(2002), 107-125.

[16] Purcell E., *The vector method of solving simultaneous linear equations*, J. Math. Phys., 32 (1953), 180-183.

[17] Sloboda F., *A parallel projection method for linear algebraic systems*, Apl. mat. Ceskosl. Akad. Ved. 23 (1978), 185-198.

[18] Tanabe K., *Projection Method for Solving a Singular System of Linear Equations and its Applications*, Numer. Math., 17(1971), 203-214.

[19] Young D.M., *Iterative solution of large linear systems*, Academic Press, New York, 1971.