AN OVERLAPPED GROUP ITERATIVE METHOD FOR SOLVING LINEAR SYSTEMS

José M. Bioucas-Dias

Instituto Superior Técnico

Instituto de Telecomunicações

Torre Norte, Piso 10

Av. Rovisco Pais, 1049-001 Lisboa

Email: bioucas@lx.it.pt

2001

Abstract

We propose a new iterative method for the numerical computation of the solution of linear systems of equations. The method is suited to problems exhibiting local dependencies, such as those resulting from the discretization of partial differential equations or from Markov random fields in image processing. The technique can be envisaged as a generalization of the Gauss-Seidel *point iterative* method being able, however, to achieve greater convergence rates. On each iteration, a group of variables is treated as unknown while the others are assumed known; the equations associated to the mentioned group are then solved in order to the unknown variables. The following iterations do the same, choosing other groups of variables. The successive groups overlap one another, departing definitely from the *group iterative* perspective, since in the latter case groups are disjoint. The *overlapped group* (OG) method, herein introduced, is shown to converge for two classes of problems: (1) symmetric positive definite systems; (2) systems in which any principal submatrix is nonsingular and whose inverse matrix elements are null above (below) some upper (lower) diagonal. In class (2) the exact solution is reached in just one step. A set of experiments comparing OG convergence rate and computational complexity with those of other popular iterative methods, illustrates the effectiveness of the proposed scheme.

1 Introduction

The need for solving $N \times N$ symmetric positive definite (SPD) linear systems

$$Ax = b, \quad x \in \mathbb{R}^N, \quad \text{and} \quad A = A^T, \tag{1}$$

arises in several applications, namely in the solution of integral and partial differential equations, image processing, time series analysis, statistics, control theory, etc. The algorithms for finding the solution $x^* = A^{-1}b$ can be classified as *direct* or *iterative* [1], [2], [3], [4].

1.1 Direct Methods

Direct methods, by definition, reach the exact solution within a finite number of operations. For matrices without any special structure, the *complexity* of direct methods is $O(N^3)$, meaning that the number of floating point operations[1] is of the order of N^3. When the system is large, a complexity of is $O(N^3)$ unbearable. There are, however, classes of systems to which there exits faster direct algorithms. A relevant example is the Toeplitz systems which can be solved with the Levinson recursion formula [5], [6], [4] with $O(N^2)$ complexity, or with the algorithms proposed in [7], [8] with complexity $O(N \ln^2 N)$. If, besides Toeplitz, A is generated by a rational function of order (p, q), methods proposed in [9], [10], and [11] have $\max(p, q)\, O(N)$ complexity.

The *conjugate gradient* (CG) method, introduced in [12], is another direct[2] method for solving SPD linear systems. The convergence rate of the CG method is determined by the spectrum pattern of matrix A [13]. Roughly, it converges faster if the eigenvalues of A are clustered. The class of *preconditioned conjugate gradient* (PCG) methods introduces a preconditioning step through a matrix P on the CG method. From the spectral clustering point of view it is the matrix $P^{-1}A$ that matters. The preconditioner P should be, somehow, close to matrix A. Preconditioners for Toeplitz systems are studied in [14], [15], and [16]. A common feature to all these preconditioners is that they can be easily inverted (by means of fast transforms such as the fast Fourier transform, cosine transform, or sine transform), leading to superfast algorithms with complexity $O(N \ln N)$. Another advantage of the PCG methods is that they can be parallelized, whenever the transform used can be parallelized. Parallelization leads to a complexity $O(\log N)$, when N processors are used. For non-Toeplitz matrices, there is no general approach concerning the conditioner design.

[1] Real additions and real multiplications.

[2] The CG method is a direct method because it finds the solution after no more than N iterations. However, it can also be classified as an iterative method, since it generally needs less than n iterations to achieve a solution with acceptable precision. This is more evident in large systems.

1.2 Iterative Methods

Linear systems resulting from many signal and specially image problems are very large. A usual feature of such huge systems is that the involved matrices are sparse; the interactions between variables are usually confined to a small neighborhood. This can be found, for example, in linear systems resulting from discretization of partial differential equations, where the interaction length depends on the order of the highest derivative [1]. Also, in the field of image restoration, using regularization principles or stochastic paradigms such as Markov random fields, the neighborhood order is, frequently, much smaller than the line and column sizes [17], [18], [19]. Assuming Toeplitz systems, the PCG method can still be applied. However, even for matrices with sparse structure, the PCG method still leads to the same $O(N \ln N)$ complexity, since the inverse of a sparse matrix is not, necessarily, sparse. Thus, the preconditioning step of PCG methods has still $O(N \ln N)$ complexity.

For large systems, iterative methods are preferred to direct methods [2]; despite the infinite time they generally need to find the exact solution, they often yield a solution within acceptable error with fewer operations than direct methods. Moreover, round off errors[3] (or any other error), are dumped out as the process evolves [2], [1], [20], [21].

For a wide class of sparse matrices (e.g. systems resulting from discretization of partial differential equations or systems resulting from Gauss-Markov descriptions in signal and image processing), the iteration complexity is $O(N)$. Consequently, if an acceptable solution is reached in t_0 iterations, with t_0 independent of N, then the overall complexity is still $O(N)$. Another important and sometimes determinant attribute of iterative methods is the mild storage requisites normally needed in the case of sparse matrices.

Among iterative techniques, the Jacobi (J), the Gauss-Seidel (GS), and the successive overrelaxation (SOR) are well known and widely used methods. They belong to the class of *linear stationary iterative methods of first degree* [1], [3]. They are also classified as *point iterative*, since each iteration can be implemented by solving simple equations for each system component.

Group iterative methods [1] resemble point iterative ones, replacing each individual component by a group, such that each component belongs to one and only one group [1]. If the groups form a *partitioning* of the set $S = \{1, \cdots, N\}$, the resulting method is known as a *block method* [20], [22]. Each of the above referred point methods has a correspondent block method; namely, the block Jacobi (BJ), the block Gauss-Seidel (BGS), and the block successive overrelaxation (BSOR).

Block iterative methods were developed with the purpose of increasing the convergence rate of the

[3]For large and/or ill-conditioned systems, rounding errors due to floating-point arithmetic are, frequently, the main problem of direct methods. Rounding errors can severely degrade the solutions found.

respective point methods. Assuming that A is a *M-matrix*, the BJ and BGS converge at least as rapidly as the respective *point* counterparts [1]. On the other hand, if A is *Stieltjes* and π-*consistently ordered*, then the BSOR method, implemented with the optimum relaxation factor, converges faster than the SOR [1], [3]. It should be stressed that determining the exact (or approximate) relaxation factor, necessary to the SOR and the BSOR methods, frequently has such a high cost that the method is impracticable. A remarkable exception occurs whenever A has the socalled property A [1], [3]. In this case it is possible to establish a relation between the eigenvalues of A (equivalently the eginvalues of Jacobi iteration matrix) and the optimum relaxation factor. This procedure is very effective, for example, in problems resulting from the discretization of *elliptic* differential equations [1], [23].

1.3 Rationale of the Proposed Iterative Method

A shortcoming of the block methods, at least for those systems describing local interactions, is that the error of each component tends to be larger on the block boundaries than on its interior[4] This pattern of behavior is illustrated in Fig. 4: the curve BSG-8 shows the error after the first iteration of the BSG method with blocks of size 8. Notice the larger errors at the block boundaries. We will back in more detail to this experiment in Section 5

The *overlapped group* (OG) iterative method, herein introduced, operates (as group methods do) on groups of components. However, contrary to block methods, in the OG scheme groups are not disjoint. By overlapping the groups in a proper manner, the distance between the components being updated and the ones already updated is kept constant (or above a minimum positive number D). Thus, by choosing the updated components with smaller error, it is expectable that the proposed method converges faster than the block methods.

The OG method is a descent procedure that can be interpreted in terms of the evolution of the quadratic function

$$F(x) = \frac{1}{2}x^T Ax - b^T x. \tag{2}$$

If A is symmetric and positive definite, then $F(x)$ is strictly convex; therefore, it has a unique minimum x^*, satisfying $\nabla F(x^*) = Ax^* - b = 0$. Minimizing $F(x)$ or solving the system $Ax = b$ are equivalent problems. In the i-th OG iteration, function $F(x)$ is minimized with respect to the components x_j for $j \in S_i$, where S_i is a set of indices, keeping the components x_k for $k \notin S_i$ constant. Iteration $(i + 1)$ resembles the i-th one, replacing the set S_i by S_{i+1} which partially overlaps the former. The overlapping of sets S_i is crucial concerning the achievement of larger convergence rates.

[4]Block boundaries depend on the problem and on the way that indexes are assigned to each site.

This behavior is illustrated by curves BSG-8 and OG-8 in Fig. 4: both methods use blocks of size 8; however the blocks in OG-8 have an overlaping of 7, whereas in BSG-8 they do not overlap. Notice the smaller error of OG-8 after the first iteration.

On the other hand, a suitable implementation of the proposed strategy demands, approximately, the same computational burden as the Gauss-Seidel (which corresponds to the choice $S_i = \{i\}$) applied on the original system.

According to the rationale just presented, it is expectable that the sequence $\{x(n)\}$ produced by the OG method converges to $x^* = A^{-1}b$, whenever A is a SPD matrix; this result is shown in Section 4. Besides SPD systems, convergence is also studied for systems whose inverse matrix elements are null above (below) some upper (lower) diagonal. For these matrices, there are choices of the sets S_i such that the OG method takes only one iteration to converge.

The paper is organized as follows. Section 2 introduces the OG method formally. A comparison with the multisplitting methods and a reinterpretation under the light of the multigrid concept is also provided. Section 3 proposes a filter-like implementation, makes considerations about complexity, and shows that there exists an equivalent system from the Gauss-Seidel iteration point of view. Section 4 presents convergence results. Section 5 shows results of a series of experiments and makes comparisons with other iterative methods. Section 6 ends the paper by presenting some concluding remarks. The proofs of the theorems on convergence are given in the Appendix.

2 Overlapped Group Method

The first step to implement the OG method is the definition of the iteration groups[5]:

Definition 1 *An* ordered covering *g of $S = \{1, \ldots, N\}$ is an ordered collection of subsets $S_i \subset S$, with $i = 1, \ldots, n_g$ such that $\cup_{i=1}^{i=n_g} S_i = S$. Two ordered coverings g and g' given by $S_1, \ldots, S_{n_g}$ and $S'_1, \ldots, S'_{n'_g}$, respectively, are identical if $n_g = n'_g$ and if $S_1 = S'_1$,. . .,$S_{n_g} = S'_{n'_g}$.*

Definition 1 differs from an *ordered grouping* [1], in that sets S_i in the latter are necessarily disjoint.

A one-dimensional (1D) oriented covering is $g_1(D) \equiv \{S_1, \ldots, S_{N-D+1}\}$, *where*

$$S_i = \{i, i+1, \ldots, i+D-1\}, \quad i = 1, \ldots, N-D+1. \tag{3}$$

The Covering (3) is suited to system matrices with its significant elements close to the main diagonal. This is quite often the picture in 1D processing problems.

[5]The designation of group is not to be understood in the usual mathematical sense.

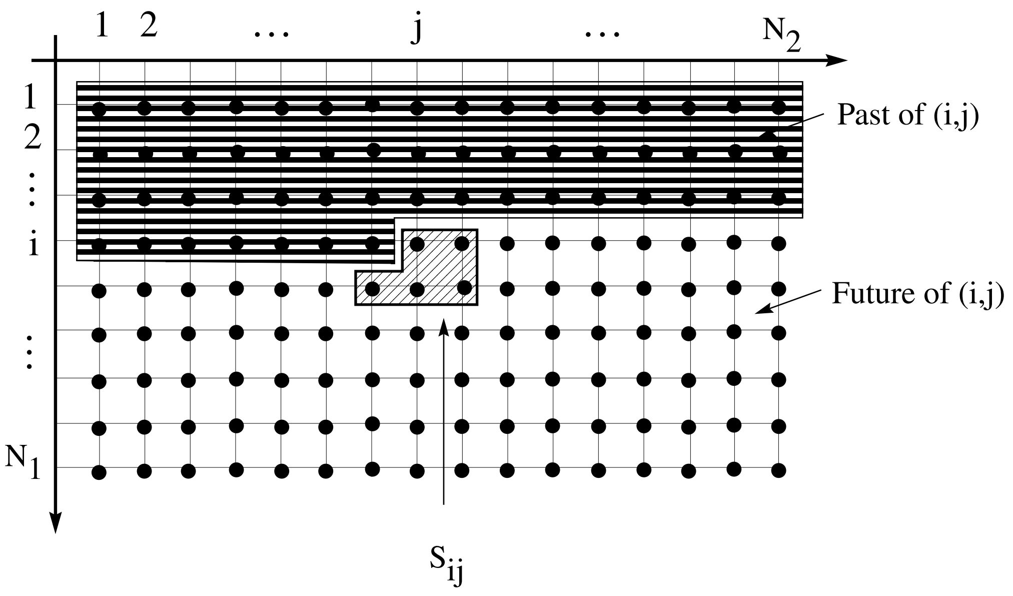

Figure 1: Illustration of a set S_{ij} which is an element of the 2D oriented covering $g_2(2,2)$. The regions named *past* and *future* denote those sites $(k,l) \in \mathcal{L}$ such that $(k,l) < (i,j)$, and $(k,l) \geq (i,j)$, respectively.

Two-dimensional (2D) problems are frequently defined on regular lattices of the form $\mathcal{L} = \{(i,j) |\ 1 \leq i \leq N_1,\ \ 1 \leq j \leq N_2\}$. Using the order relation on $\mathcal{L}$ induced by the lexicographic ordering (by row), a 2D oriented covering is $g_2(D_1, D_2) \equiv \{S_{i,j},\ i = 1, \ldots, N_1 - D_1 + 1\, j = 1, \ldots, N_2 - D_2 + 1\}$, where

$$S_{ij} = \{(k,l) \mid (k,l) \geq (i,j),\ 0 \leq |k-i| \leq D_1 - 1,\ 0 \leq |l-j| \leq D_2 - 1\}. \tag{4}$$

Fig. 1 schematizes a set S_{ij} of the covering $g_2(2,2)$. The shadowed region named *past* represents those sites $(k,l) \in \mathcal{L}$ such that $(k,l) < (i,j)$, while the unshadowed region named *future* represents those sites $(k,l) \in \mathcal{L}$ such that $(k,l) \geq (i,j)$.

We introduce now a set of matrices and vectors depending on A, g, x, and b.

Definition 2 *Given an ordered covering g, define the diagonal matrix $D_i = diag\,(1_{S_i}(k), k = 1, \ldots, N)$, where 1_{S_i} is the indicator function of set S_i and $diag(\mathbf{a})$ is a diagonal matrix with diagonal entries given by the respective components of vector $\mathbf{a}$. Define also $\overline{D}_i = I - D_i$. Given A, let $Q_i = D_i A D_i$ and $\overline{Q}_i = D_i A \overline{D}_i$. Further, define matrices $G_i = Q_i + \overline{D}_i$ and $H_i = -\overline{Q}_i + \overline{D}_i$.*

As an example of the matrices defined above, consider the covering (3) with $D = 2$, and $N = 10$. The set $S_3 = \{3, 4\}$ induces

$$Q_3 = \left[\begin{array}{c|c|c} 0_{2,2} & \multicolumn{2}{c}{0_{2,8}} \\ \hline 0_{2,2} & [2 \times 2] & 0_{2,6} \\ \hline \multicolumn{2}{c|}{0_{2,8}} & 0_{2,2} \end{array}\right] \qquad G_3 = \left[\begin{array}{c|c|c} I_2 & \multicolumn{2}{c}{0_{2,8}} \\ \hline 0_{2,2} & [2 \times 2] & 0_{2,6} \\ \hline \multicolumn{2}{c|}{0_{2,8}} & I_6 \end{array}\right]$$

and

$$\overline{Q}_3 = \left[\begin{array}{c|c|c} 0_{2,2} & \multicolumn{2}{c}{0_{2,8}} \\ \hline [2 \times 2] & 0_{2,2} & [2 \times 6] \\ \hline \multicolumn{2}{c|}{0_{2,8}} & 0_{2,2} \end{array}\right] \qquad H_3 = \left[\begin{array}{c|c|c} I_2 & \multicolumn{2}{c}{0_{2,8}} \\ \hline -[2 \times 2] & 0_{2,2} & -[2 \times 6] \\ \hline \multicolumn{2}{c|}{0_{2,6}} & I_6 \end{array}\right].$$

The notation $[i \times j]$ stands for the correspondent block of A spanning i lines and j columns.

The OG method is now formally introduced:

OG Method *Given the linear system $Ax = b$, the ordered covering g, the cyclic schedule $i = saw(n, n_g) \equiv (n-1) \bmod n_g + 1$ with $n = 1, 2, \ldots$, and the starting vector $x(0)$, the OG method generates the sequence $\{x(n)\}$ according to*

$$x(n) = (G_i^{-1} H_i) x(n-1) + G_i^{-1} D_i b, \qquad n = 1, 2, \ldots \tag{5}$$

Implicit in (5) is the assumption that G_i^{-1} exists, for $i = 1, \ldots, n_g$.

A sufficient condition for the existence of G_i^{-1}, for $i = 1, \ldots, n_g$, is that A is positive definite. However, the OG method applies to a wider class of matrices as we will see in Section 4.

Assume that matrix A is SPD. In order to evaluate the quadratic function $F(x) = \frac{1}{2} x^T A x - b^T x$ along the successive iterations, define the column vector s_i such that its j-th component is the derivative of $F(x)$ with respect to x_j if $j \in S_i$ and zero otherwise. Having in mind the structure of matrices Q_i, $\overline{Q}_i$, and D_i, one can write

$$s_i = Q_i x + \overline{Q}_i x - D_i b. \tag{6}$$

Let $x(n-1)$ and $x(n)$ be vectors with components $x_j(n-1) = x_j(n)$ if $j \notin S_i$. This constraint is expressed by

$$\overline{D}_i x(n) = \overline{D}_i x(n-1). \tag{7}$$

Furthermore, assume that the components $x_j(n)$ for $j \in S_i$ lead to to $s_i = 0$. This is equivalent to

$$Q_i x(n) = -\overline{Q}_i x(n) + D_i b = \overline{Q}_i x(n-1) + D_i b. \tag{8}$$

The last equality is an immediate consequence of $\overline{Q}_i$ having null columns c, for $c \in S_i$. Summing equation (7) and (8), one obtains

$$(Q_i + \overline{D}_i) x(n) = (-\overline{Q}_i + \overline{D}_i) x(n-1) + D_i b, \tag{9}$$

which, reminding that $G_i = Q_i + \overline{D}_i$ and $H_i = -\overline{Q}_i + \overline{D}_i$, and assuming that G_i^{-1} exists, has the solution (5). Thus, the OG method is a descent method that, in each iteration, minimizes $F(x)$ with respect to $\{x_j | j \in S_i\}$.

Successively applying the iteration (5) with $i = \mathrm{saw}(n, n_g) = 1, \ldots, n_g$, and $n = (t-1)\, n_g + 1, \ldots, t\, n_g$ one obtains

$$x(t\, n_g) = M x\big((t-1)\, n_g\big) + Nb, \quad t = 1, 2, \ldots \tag{10}$$

where

$$M = \prod_{i=n_g}^{i=1} G_i^{-1} H_i, \tag{11}$$

$$N = \left[\sum_{i=1}^{n_g - 1} \left(\prod_{k=n_g}^{k=i+1} (G_k^{-1} H_k) \right) G_i^{-1} D_i \right] + G_{n_g}^{-1} D_{n_g}. \tag{12}$$

We now introduce the subsequence $x(n = t\, n_g)$, with $t \in \mathbb{N}$ (the set of naturals), which is the output of OG iteration after t full sweeps of index i in (5). The index t in $x(t)$ and the index n in $x(n)$ will distinguish both sequences[6] . Using this notation, equation (10) becomes

$$x(t) = M x(t-1) + Nb \quad t = 1, 2, \ldots \tag{13}$$

which is a *linear stationary iterative method of first degree* [1]. Solving the recursion (13), one is led to

$$x(t) = M^t x(0) + \sum_{i=0}^{i=t-1} M^i N b. \tag{14}$$

If N^{-1} exists, then, the sequence $\{x(t)\}$ is generated by the splitting $A = B - C$, with $B = N^{-1}$, and $C = N^{-1} M$.

Assume that matrix M is convergent (*i.e.*, its spectral radius satisfies $\rho(M) < 1$). This implies that (see [1])

$$x_\infty = \lim_{t \to \infty} x(t) = \left(\sum_{i=0}^{\infty} M^i \right) Nb = (I - M)^{-1} Nb. \tag{15}$$

Defining $e(t) = x(t) - x_\infty$, and after some manipulation involving (13) and (15), one is led to

$$e(t) = M^t e(0). \tag{16}$$

Thus, if M is convergent, the *distance* between $x(t)$ and x_∞ decays geometrically. Conversely, if M is not convergent, there exist starting points $x(0)$, for which the recursion (13) diverges [1]. Of course, the only interesting case is $x_\infty = x^* = A^{-1} b$. In Section 4, two classes of systems exhibiting the latter property are studied.

[6]This notation is of course incorrect since we are naming different functions with basis on their arguments. We have adopted it however for the sake of lightness of the exposition.

2.1 Connection with Multisplitting

At this point a reference to multisplitting methods [24], [25], [26], [27] is worthy. They are parallel algorithms for solving linear systems of equations. A multisplitting is a sequence of splittings $A = B_k - C_k$, for $k = 1, \dots, n_K$ (matrix B_k is invertible by definition). The iterative scheme

$$x^k(t) = B_k^{-1} C_k x(t-1) + B_k^{-1} b \quad t = 1, 2, \dots \tag{17}$$

is associated to each k. Vectors $x^k(t)$ can be computed concurrently and linearly combined to produce

$$x(t) = \sum_{k=1}^{n_K} U_k x^k(t), \tag{18}$$

where $U_k \geq 0$ is a diagonal matrix, and $\sum_{k=1}^{k=n_K} U_k = I$. Normally, each splitting is built to privilege a group of variables x_i, $i \in S_k$, concerning their error in each iteration. In order to select only the components x_j, $j \in S_k$, matrix U_k is set to zero except for the j-th principal diagonal element which is set to one if $j \in S_k$. Consequently, only a small part of B_k^{-1} and $B_k^{-1} C_k$, need to be computed.

A comparison between multisplitting and the OG method leads to the following conclusions:

1. Multispllitting methods update groups of variables concurrently, whereas OG updates groups sequentially

2. For SPD systems, OG is a descent method: each update of a variable group minimizes the quadratic function associated to the system, keeping constant the remaining variables. In contrast, even if the meaning of each multisplitting update was clear from a descent point of view, the result of a linear update combination is not clear.

This considerations bring to mind the idea of a parallel implementation of the OG method itself; given an ordered covering and an associated set of matrices U_k having the same meaning as in (18), the following parallel/iterative scheme can be implemented:

$$x^k(t) = G_k^{-1} H_k x(t-1) + G_k^{-1} b \tag{19}$$

$$x(t) = \sum_{i=1}^{n_K} U_k x^k(t), \tag{20}$$

where G_k and H_k are the matrices associated to the group S_k, from Definition 2. We stress that $A \neq G_k - H_k$, and thus this scheme is not a multisplitting.

The work [28] proposes, implicitly, an iterative algorithm of the type (19)-(20), which was conceived to efficiently explore parallel architectures. The adequation of the algorithm to the parallel hardware determines the choice of sets S_k. Concerning sets U_k, its i-th diagonal element is given by $u_i^k =$

$[\sum_{k=1}^{n_g} I_{S_k}(i)]^{-1}$. Thus, in the t-th iteration, the new variable value $x_j(t)$ is given by the mean of $x_j^k(t)$ for $j \in S_k$, and $k = 1, \ldots, n_g$. It should be stressed that the scheme proposed in [28] results more from hardware adequation than from an intentional implementation of scheme (19)-(20).

Given an arbitrary starting vector $x(0)$, the sequence $\{x(t)\}$ generated by (19)-(20) is convergent if and only if matrix

$$M = \sum_{k=1}^{n_g} U_k G_k^{-1} H_k \tag{21}$$

is convergent. It would be worthy to compare $\rho(M)$ with $\rho(M_J)$, where $M_J = D_A^{-1}(L_A + U_A)$ is the Jacobi iteration matrix, D_A is the diagonal of A, and L_A and U_A are the negative strictly lower and the negative strictly upper triangular parts of A, respectively. It would also be interesting to compare convergence rates of matrix M with convergence rates of multisplitting schemes. However, this is out of the scope of this work.

2.2 Connection with Multigrid

Let x^* be the solution of $Ax = b$ (by hypothesis A^{-1} exists). Frequently, x_i^* depends on *far* data variables b_k: $|k - i| \gg 0$. This fact penalizes the convergence rate of iterative methods, and fostered the multigrid approach [29]. The underlying idea is that one should firstly determine the long distance (low frequency) components of the solution, by means of some subsampling scheme, and next determine the short distance (high frequency) components of the solution, by means of some interpolation scheme. The OG method, under covering $g_1(D)$, produces a local inversion of size D. In this way, the component $x_i(t+1)$ receives information from data components b_k up to distance $k = i + D - 1$. Moreover, if x_i^* depends only on b_k: $k < i + B$ (which is equivalent to having A^{-1} elements null above the upper B-th diagonal) and $D \geq B + 1$, then the OG method finds the solution in one step (a proof of this result is given in section 4). Thus, the OG method embodies, in this sense, the multigrid philosophy. A similar set of ideas applies equally to 2D-problems.

3 Implementation

Assuming segmentation $g_1(D)$ defined in (**??**) and $D = 1$, the OG method reduces to the GS method. Otherwise ($D > 1$), if equation (5) was directly implemented, it would involve a complexity per iteration greater than the GS scheme and even greater than block methods (considering blocks of size D). However, it is possible to compute $x(t)$ with fewer operations. This is going to be illustrated for segmentation $g_1(D)$.

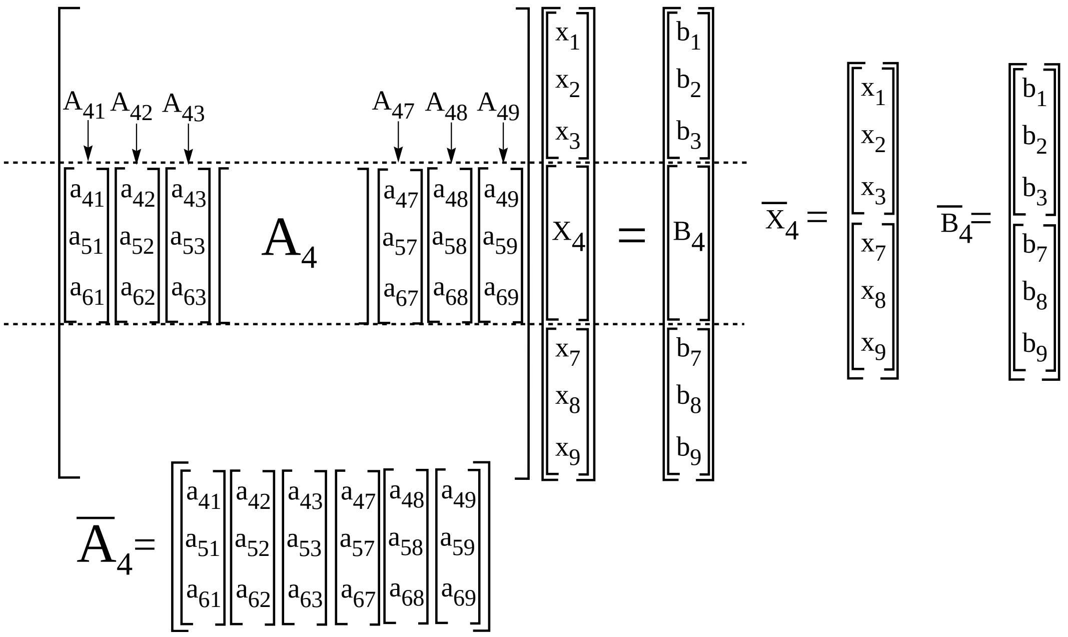

Figure 2: Schematization of submatrices introduced in Definition 3, for $N = 9$, $D = 3$, and $S_4 = \{4, 5, 6\}$.

Let $A[S_i|S_j]$, for $S_i, S_j \in g$, denote the submatrix of A obtained by removing from A all lines and columns whose indexes are not in S_i and S_j, respectively. In the same fashion define $b[S_i]$ as the subvector of column vector b obtained by removing from b all components whose indexes are not in S_i.

At this point it is convenient to introduce the following submatrixes:

Definition 3 *Given the sets $S_i \in g$ and $\overline{S}_i = S_i^c$ (complement of S_i with respect to $\{1, 2, \ldots, N\}$), the matrix A, and the vectors b and x we introduce: $A_i = A[S_i|S_i]$, $\overline{A}_i = A[S_i|\overline{S}_i]$, $A_{ij} = A[S_i|\{j\}]$, for $j = 1, \ldots, N$, $X_i = x[S_i]$, $\overline{X}_i = x[\overline{S}_i]$, $B_i = b[S_i]$, and $\overline{B}_i = x[\overline{b}_i]$.*

Figure 2 depicts an instance of the entities above defined for segmentation $g_1(3)$, $N = 9$, and $S_4 = \{4, 5, 6\}$. Notice that matrices A_i and $\overline{A}_i$ are obtained from Q_i and $\overline{Q}_i$ by deleting their null row and columns.

Consider the n-th iteration (5) such that $i = \mathrm{saw}(n, n_g)$ and $i + D \leq N$: although vector X_i has dimension D, only its first component x_i needs to be computed. This is so because: (1) the components x_j, $j = i + 1, \ldots, i + D - 1$ will be updated in the next iteration; (2) the new values of the variables in the group S_{i+1} depend on $x_j(t)$ for $j = i + D + 1, \ldots, N$, and on $x_j(t + 1)$, for $j = 1, \ldots, i$. Thus, in the n-th iteration, only $x_i(t + 1)$ needs to be computed, being wasteful to process the complete block. Only if $i = N - D + 1$, it becomes necessary to compute all components of X_{N-D+1}.

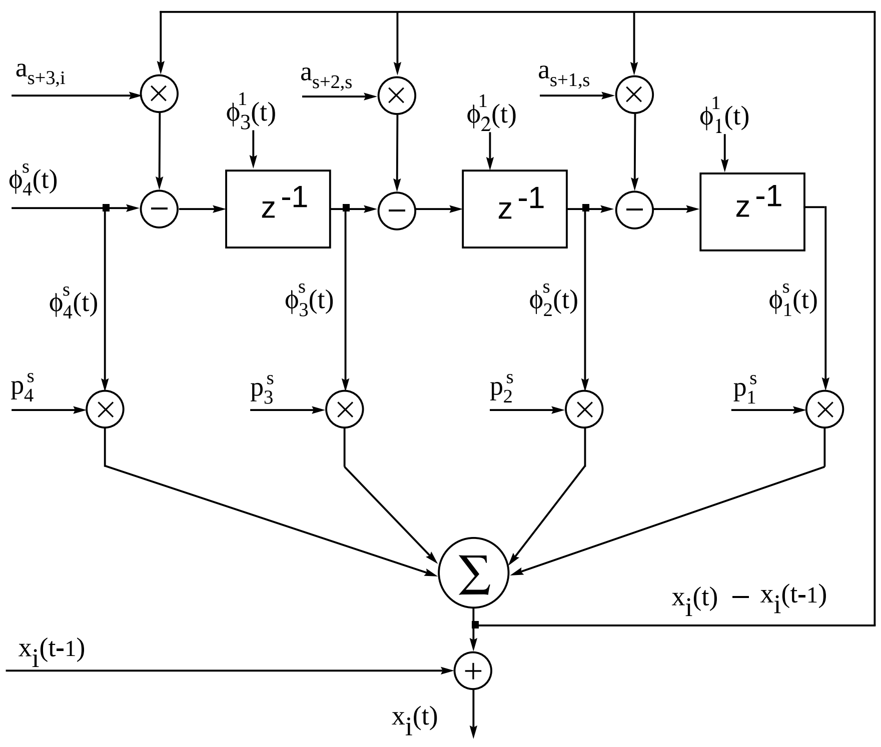

Figure 3: Filter like structure implementing a OG iteration for $D = 4$.

For 2D problems defined on regular lattices, segmentations $g_2(N_1, D_2)$ (group of lines) have the property described above, but now applied to lines (recall that the lattice has size $N_1 \times N_2$). The same applies to columns using segmentation $g_2(D_1, N_2)$. If $D_1 \neq N_1$ and $D_2 \neq N_2$, components x_{ij}, $x_{i+1,j-D_2+1}, \ldots, x_{i+D_1-1,j-D_2+1}$ (with the suitable adjustments on the boundary) need to be computed in each iteration. Hence, D_1 variables have to be updated.

Define the functions $r(i)$ and $s(i)$ on the set $i = 1, \ldots, N$ as

$$r(i) = \begin{cases} 1 & i \leq N-D \\ i-(N-D) & i > N-D \end{cases} \quad \text{and} \quad s(i) = i - r(i) + 1, \tag{22}$$

and the column vector

$$\boldsymbol{p}^i = [A_i^{-1}]_{r(i)}, \tag{23}$$

where the notation $[A]_k$ stands for the k-th row of matrix A. The iterative scheme (5) under the segmentation $g_1(D)$ is, then, equivalent to

$$x_i(n) \;=\; \boldsymbol{p}^i\big(B_{s(i)} - \overline{A}_{s(i)}\overline{X}_{s(i)}(n-1)\big) \quad i = 1, \ldots, N. \tag{24}$$

Apart from $\boldsymbol{p}^i$, vector $\overline{A}_{s(i)}\overline{X}_{s(i)}(n-1)$ in (24) is by far the most computational demanding. For fully populated matrices, it needs $(N-D) \times D$ real multiplications. However, for $D > 1$ a major component of (24) can be determined recursively. Noting that $\boldsymbol{p}^i A_{s(i)} = [\delta(k - r(i)),\ k = 1, \ldots, D]$, where δ is the kronecker symbol, one can write (recall that $x(t) = x(n = t\, n_g)$)

$$x_i(t) = x_i(t-1) + \boldsymbol{p}^i \boldsymbol{\phi}^{s(i)}(t) \quad \text{with} \quad i = 1, \ldots, N, \tag{25}$$

with

$$\boldsymbol{\phi}^s(t) = B_s - \sum_{k=1}^{k=s-1} A_{sk}\, x_k(t) - \sum_{k=s}^{k=N} A_{sk}\, x_k(t-1) \qquad s \leq N - D + 1. \tag{26}$$

The components of $\boldsymbol{\phi}^s(t) = [\phi_1^s(t), \ldots, \phi_D^s(t)]^T$ satisfy

$$\left\{ \begin{array}{lcll} \phi_k^{s+1}(t) & = & \phi_{k+1}^s(t) - a_{(s+k)s}[x_s(t) - x_s(t-1)] & s \leq N - D \\ \phi_D^s(t) & = & b_{s+D-1} - \displaystyle\sum_{k=1}^{k=s-1} a_{(s+D-1)k}\, x_k(t) - \sum_{k=s}^{k=N} a_{(s+D-1)k}\, x_k(t-1) & s \leq N - D + 1, \end{array} \right. \tag{27}$$

for $k = 1, \ldots, D-1$. Fig. 3 depicts a filter-like structure implementing an iteration ($s = s(i)$, $i = 1, \ldots, N$, $\boldsymbol{p}^s = [p_1^s, \ldots, p_D^s]$, and t constant) described by equations (25) and (27). Terms $\boldsymbol{\phi}_1^1(t), \ldots, \boldsymbol{\phi}_1^{D-1}(t)$ on top of the delay taps represents the initial values, and are given by (26).

The implementation herein proposed applies to the segmentation $g_1(D)$. For segmentations $g_2(N_1, D2)$, the same concepts and ideas still apply with the convenient adjustments (replacing each one-dimensional element by N_1-dimensional elements). However, for other segmentations with some degree of nonregularity, it is no longer possible to implement the OG method with equations (25) and (27). Indeed, the lack of regularity implies that some (or a lot of) pairs S_i, S_{i+1} do not overlap (or overlap irregularly) determining the computation of an irregular number of components of $X_i(n)$; this leads to a higher computational burden.

3.1 A Gauss-Seidel iteration point of view

Introducing definition (26) in equation (25), one obtains

$$x_i(t) = x_i(t-1) + \left(\boldsymbol{p}^i B_{s(i)} - \sum_{k=1}^{k=i-1} \boldsymbol{p}^i A_{s(i)k}\, x_k(t) - \sum_{k=i}^{k=N} \boldsymbol{p}^i A_{s(i)k}\, x_k(t-1) \right), \tag{28}$$

valid for $i = 1, \ldots, N$ and $t = 1, 2, \ldots$ Noting that $\boldsymbol{p}^i A_{s(i)i} = 1$, the iteration (28), can be rewritten as

$$x_i(t) = \left(b_i' - \sum_{k=1}^{k=i-1} a_{ik}'\, x_k(t) - \sum_{k=i+1}^{k=N} a_{ik}'\, x_k(t-1) \right), \tag{29}$$

where

$$\left\{ \begin{array}{ll} b_i' = \boldsymbol{p}^i B_{s(i)} & i = 1, \ldots, N \\ a_{ik}' = \boldsymbol{p}^i A_{s(i)k} & i, k = 1, \ldots, N. \end{array} \right. \tag{30}$$

Equation (29) defines a GS iteration applied to the system $A'x = b'$, with $A' = TA$, $b' = Tb$, and

$$T = \left[\begin{array}{cccc|cccc} p_1^1 & p_2^1 & \cdots & p_D^1 & 0 & \cdots & \cdots & 0 \\ 0 & p_1^2 & p_2^2 & \cdots & p_D^2 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & p_1^{N-D} & p_2^{N-D} & \cdots & p_D^{N-D} & 0 \\ \hline 0 & \cdots & \cdots & 0 & & & & \\ \vdots & \cdots & \cdots & \vdots & & A_{N-D+1}^{-1} & & \\ 0 & \cdots & \cdots & 0 & & & & \end{array}\right]. \tag{31}$$

If matrix T is nonsingular, systems $Ax = b$ and $A'x = b'$ are equivalent. This happens if and only if $p_1^1 \neq 0, \cdots, p_1^{N-D} \neq 0$ and A_{N-D+1}^{-1} exists (in the next section this is shown to be true for PD systems). Matrix $A' = TA$ has the following structure:

$$A' = \left[\begin{array}{cccc|cccc} 1 & 0 & \cdots & 0 & * & \cdots & \cdots & * \\ * & 1 & 0 & \cdots & 0 & * & \cdots & * \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ * & \cdots & * & 1 & 0 & \cdots & 0 & * \\ \hline * & \cdots & \cdots & * & & & & \\ \vdots & \cdots & \cdots & \vdots & & I_D & & \\ * & \cdots & \cdots & * & & & & \end{array}\right], \tag{32}$$

where symbol $*$ denotes an arbitrary real number. Defining the splitting $A' = G' - H'$, where G' is the lower triangular part of A', the GS sequence (29) is also given by

$$G'x(t) = H'x(t-1) + b'. \tag{33}$$

As D increases, A' tends to a lower triangular and H' to the null matrix. If H' is a null matrix, the solution of $A'x = b$ will be found in just one step; each element $x_i(1)$ is recursively determined from $x_j(1)$ for $j = 1, \ldots, i-1$. However, if $H' \neq 0$, sequence (33), can be thought of as a balance between *recursiveness* and *iterativeness.*

As D increases, the eigenvalues of A' tend to be closer to one (in magnitude) than those of A. Thus, the OG method implicitly implements a *preconditioning* step on matrix A [30]. This technique is applied to problems exhibiting slow convergence. For sparse matrices, preconditioning has normally the disadvantage of increasing the number of non-null elements. Therefore, the reduction in the number of iterations must compensate for the extra computations per iteration. Given that the OG method keeps the number of non-null diagonals, convergence is speeded up without increasing the computational effort.

3.2 Computational complexity

Two different procedures implementing the OG method were presented. The first, schematized in Fig. 3 and given by equations (25) and (27), is implemented directly on the system $Ax = b$. The second implements a GS iterative scheme on the system $A'x = b'$ requiring the previous computation of $A' = TA$ and $b' = Tb$, with T given by (31).

In order to study the complexity of the above implementations, we define N_T as the number of operations necessary to compute the matrix T and $N_{A'}$ as the number of operations necessary to compute matrix A'. After t iterations each method takes the following number of operations $N_{op}(t)$:

- **Filter-type implementation**

  1. Fully populated matrices

  $$N_{op}(t) = t \times N\big(2N + 4D\big) + N_T. \tag{34}$$

  The i-th update, takes $2N + 4D$ real multiplications and real additions: term $2N$ corresponds to $\phi_D^s(t)$ with $s \leq N - D + 1$ plus the initial conditions $\phi_k^1(t)$ with $k = 1, \cdots, D-1$; term $4D$ corresponds to the operations schematized in Fig. 3

  2. B-banded matrices

  $$N_{op}(t) = t \times N\big(2(2B+1) + 4D\big) + N_T. \tag{35}$$

- **GS equivalent implementation**

  1. Fully populated matrices

  $$\begin{aligned} N_{op}(t) &= t \times N\big(2(N-D)\big) + N_T + N_{A'} + 2ND \\ &\simeq t \times N\big(2(N-D)\big) + N_T + N_{A'}. \end{aligned} \tag{36}$$

  The i-th update takes $2(N-D)$ real multiplications and real additions the OG method introduces $D-1$ zeros in each line and that $a'_{ii} = 1$). Term $2ND$ corresponds to the operations for computing vector b'.

  2. B-banded matrices

  $$\begin{aligned} N_{op}(t) &= t \times N\big(2(2B)\big) + N_T + N_{A'} + 2ND \\ &= (t + \frac{D}{2B}) \times N\big(2(2B)\big) + N_T + N_{A'}. \end{aligned} \tag{37}$$

Apart from term $N_{A'} + 2ND$, which has vanishing relative weight as t increases, the GS equivalent implementation has less complexity then the filter-type implementation. However, they are both of the same order: $O(N^2)$ for fully populated matrices and $O(N)$ for banded matrices. Concerning $N_{A'}$, it takes $D \times N\big(2(N-D)\big)$ operations for fully populated matrices and $D \times N\big(2B\big)$ for B-banded matrices; this corresponds to removing $N_{A'}$ from (36) and (37), and replacing t by $(t+D)$ in the same expressions. On the other hand, for Toeplitz or *quasi-Toeplitz* the term $N_{A'}$ is negligible, even compared with the complexity of one single iteration.

In conclusion, although having an overhead necessary to compute A' and b', the GS equivalent implementation takes less operations per iteration than the filter-like implementation. Hence, choosing the implementation with less complexity depends on the convergence rate: for convergence rates close to zero, use implementation depicted in Fig. 3; for convergence rates close to one determine A', b', and apply the GS iteration (29).

Compared with GS, the OG method implemented according to (29) and after t iterations takes the number of operation to compute matrix T plus, roughly, the complexity of $(D+t)$ iterations of the GS algorithm. This overhead, is almost always compensated by the greater convergence rate of the OG method.

4 Convergence Analysis

We have, so far, introduced the OG method, proposed two possible implementations, and studied the computational complexity of these implementations. This section is devoted to convergence analysis; the main results are: (1) if A is SPD, the method converges for the exact solution regardless of the covering; (2) if $A^{-1} = [r_{ij}]$ satisfy $r_{ij} = 0$ for $j - i \geq B$, the covering $g_1(D)$ with $D \geq B$ leads to a null iteration matrix ($M = 0$). Therefore, the solution x^* is found in just one iteration.

Assume that A is SPD. Matrices G_i introduced in Definition 2 are a key element in the proposed iterative scheme. Namely, they have to be invertible. Reminding that $G_i = Q_i + \overline{D}_i$, $D_i + \overline{D}_i = I$, $Q_i = D_i A D_i$, one can write

$$\begin{aligned} x^T G_i x &= x^T \overline{D}_i x + x^T Q_i x && (38) \\ &= \|x_1\|_2^2 + x_2^T A x_2, && (39) \end{aligned}$$

where $x_1 = \overline{D}_i x$ and $x_2 = D_i x$. If $x \neq 0$, then $x_1 \neq 0$ or $x_2 \neq 0$, or both. If $x_1 \neq 0$ the right hand side of (38) is positive; if $x_1 = 0$ then $x_2 \neq 0$ and $x_2^T A x_2 > 0$ as A is positive definite. Therefore, if A is SPD, matrices G_i are nonsingular and the OG method introduced in section 2 is well defined.

Notice that matrices A_i introduced in Definition 3 are also positive definite, since they are principal submatrices of a SPD matrix.

In Section 2 it was shown that the OG method, for SPD systems, is descent, i.e. the convex cost function $F(x) = \frac{1}{2}\, x^T Ax - b^T x$ satisfies $F\big(x(n)\big) \leq F\big(x(n-1)\big)$. By exploring this feature we now state that the method is convergent. The idea is that any non-null update $\Delta(n) = x(n) - x(n-1)$ decreases $F(x)$. More precisely, we show that there is a $\theta > 0$ such that $F\big(x(n)\big) - F\big(x(n-1)\big) \leq -\theta \|\Delta(n)\|_2^2$. Since $F(x)$ is bounded below, the magnitude of $\|\Delta(n)\|$ converges to zero. Computing $\nabla F\big(x(n)\big)$, we conclude that $\nabla F\big(x_\infty = \lim_{n\to\infty} x(n)\big) = 0$ which, together with the convex nature of $F(x)$, means that x_∞ is its unique minimum. On the other hand, the unique minimum of $F(x)$ satisfies $Ax^* = b$. Given that A^{-1} exists, thus, $x^* = x_\infty$ is the desired solution. The following result, shown in the Appendix, provides sufficient conditions of convergence:

Theorem 1 *Let $Ax = b$ be a linear system where matrix A is symmetric and positive definite. Let $x^* = A^{-1}b$ be the solution of the system. Then, given an arbitrary initial vector $x(0) \in \Re^N$, the sequence $\{x(t)\}$ generated by the OG method converges to x^*.*

The proof given in the Appendix considers the following generalization of the OG method:

$$\begin{aligned} x(n) &= (1-\gamma)x(n-1) + \gamma[G_i^{-1}H_i x(n-1) + G_i^{-1}D_i b] && (40)\\ &= [\gamma G_i^{-1}H_i + (1-\gamma)I]x(n-1) + \gamma G_i^{-1}D_i b, && (41)\end{aligned}$$

with $\gamma \in (0,2)$ and $i = \text{saw}(n, n_g)$. Notice that (41) is the OG method when $\gamma = 1$. If, instead of a fixed γ, a set of γ_i, with $i = 1, \cdots, n_g$, was considered, such that $\gamma_i \in (0,2)$, the convergence of $\{x(n)\}$ to the solution x^*, would still hold. Iterative scheme (41), introduces in the OG setting the *successive overrelaxation* principle, able to achieve greater convergence rates by choosing convenient values for parameter γ (or parameters γ_i). Under the covering $g_1(D)$, iteration (41) can be written as

$$x(t) = (I - \gamma L')^{-1}\Big[\big((1-\gamma)I + \gamma U\big)x(t-1) + b'\Big], \quad t = 1, 2, \cdots, \tag{42}$$

where $-L'$ and $-U'$ are, respectively, the strictly lower triangular and the strictly upper triangular parts of matrix A', introduced in (32). The main line of the argument resembles the one given in Section 3.3.1, concerning the GS equivalent iterative scheme. It should be pointed out that $x(t)$ given by (42) can also be implemented, with minor changes, using the scheme developed in Section 3. The same set of concepts applies equally to 2D oriented coverings.

Once convergence is proved, one would like to have some insight on the convergence rate. In Section 2, it was shown that the OG method is equivalent to *to the linear stationary iterative method* *of first degree*

$$x(t) = Mx(t-1) + Nb, \quad t = 1, 2, \ldots,$$

with M given by (11) and N by (12). Assuming SPD systems, Theorem 1 implies that $\rho(M) < 1$. As a result, the error $e(t) = x(t) - x^*$ is given by $e(t) = M^t e(0)$ (see equation (16)). This considerations leads to the following result:

Proposition 1 *Let $Ax = b$ be a linear system where A is symmetric and positive definite. Then, for any $x(0) \in \Re^N$, the sequence $\{x(t)\}$ generated by the OG method, converges to $x^* = A^{-1}b$ at a geometric rate. In particular, for every $\varepsilon > 0$ there is a norm $\|\cdot\|$, such that*

$$\|x(t) - x^*\| \leq [\rho(M) + \varepsilon]^t \, \|x(0) - x^*\|, \quad t = 1, 2, \cdots \tag{43}$$

where $\rho(M)$ is the spectral radius of matrix M.

Proof 1 *Proposition 1 is a immediate consequence of the geometric nature of the error $e(t)$ and of the properties of matrix norms.* **Q.E.D.**

The OG algorithm has been introduced as a generalization of the GS method, able to achieve greater convergence rates, while having, roughly, the same complexity. The main line of argument is that the OG method is able to *see* D variables farther ahead, and use this information in order to update the present variable in a more accurate fashion. However, this does not mean that the convergence rate always increases as D increases. As in block methods [1], also OG method can exhibit slower converge rates than the GS method. The following example confirms this fact: consider the matrix $A = BB^T + C$, with $B = [\exp(-|\tau|)]$, $C = [2(-1/2)^{|\tau|}]$ being Toeplitz. For $N = 32$, the spectral radius[7] $\rho(M_{OG-D})$ is $\rho(M_{OG-1}) = 0.15677$, and $\rho(M_{OG-2}) = 0.158342$, i.e. $\rho(M_{OG-2}) > \rho(M_{OG-1})$. Nevertheless, for $D = 3, 4, 5$, the spectral radius is $\rho(M_{OG-3}) = 0.00282$, $\rho(M_{OG-4}) = 0.00122$, and $\rho(M_{OG-5}) = 0.00014$, respectively.

Choosing a covering is the first step towards the implementation of the OG method. We do not know any expedite way to obtain a well suited covering for a given matrix. However, the following result, shown in the Appendix, can help and give hints concerning this matter:

[7]Whenever the covering $g_1(D)$ is under assumption, we denote by M_{OG-D} the respective iteration matrix. The symbol M_{BGS-D} denotes the BGS iteration matrix having blocks of size D. Iteration matrices of GS and SOR methods are denoted by M_{GS} and M_{SOR}, respectively. Notice that $M_{GS} = M_{OG-1} = M_{BGS-1}$.

Theorem 2 *Let $Ax = b$ be a linear system, such that any principal submatrix of A in nonsingular. Moreover, the elements of $A^{-1} = [r_{ij}]$ verify $r_{ij} = 0$ for $j - i \geq B$. Given the covering $g_1(D)$ such that $D \geq B$, the OG iteration matrix M is null.*

This result gives credence to the rationale underlying the OG method sketched in Section 1. The less x_i depends on b_{i+k}, for $k > D$, the faster the method converges to the true solution.

Theorem 2 considers matrices A^{-1} having zero elements in its upper right part. In the case of A^{-1} having zero elements in its lower left part ($r_{ij} = 0$ for $j - i \leq -D$), then the OG method under the covering

$$\overline{g}_1(D): \; S_i = \{N - i + 1, \ldots, N - i - D + 2\}, \quad i = 1, \ldots, N - D + 1 \tag{44}$$

finds the solution x^* in one step. This is, of course, a minor modification of Theorem 2.

5 Numerical Results

This section presents four numerical examples comparing the OG method with other methods. Two classes of system matrices $A = [a_{ij}]$ are studied: (1) Toeplitz matrices; (2) quasi-regular matrices (those with a high degree of regularity, except for a few elements).

covering $g_1(D)$ is used in all examples. We denote the convergence rate of method $X \in \{OG, GS, BGS, SOR$
by $R(X) = \ln_{10}(\rho(X))$, where $\rho(X)$ denotes the spectral radius of matrix M_X. Whenever a block method is being considered the letter D in $X - D$ denotes the block size.

5.1 Toeplitz matrices

Example 1: $a_{ij} = \exp\left[-\left(\frac{j-i}{a}\right)^2\right]$.

Table 1 displays results for $a = \sqrt{3}$ and $N = 64$ (for $N = 128$ the results are equal up to the third digit). The *condition number* (given by $\kappa(A) = \sigma_n(A)/\sigma_1(A)$, where $\sigma_n(A)$ and $\sigma_1(A)$ are the largest and the smallest singular values of A, respectively) of matrix A, for $a = \sqrt{3}$, is $\kappa(A) \simeq 800$, accounting for an ill conditioned matrix.

Elements of $A^{-1} = [r_{ij}]$ verify $|r_{ij}/r_{ii}| \ll 1$, for $|\tau| = |j - i| > 10$. Thus, the high convergence rate of the OG method for $D = 10$ is essentially in accordance with Theorem 2, concerning one-sided banded A^{-1} matrices. The superior performance of the OG method, compared with BGS, is evident for $D \geq 1$. Compared with the SOR method, the OG has much higher convergence rate for $D \geq 3$.

Assume now that $a = 1$. Although smaller, spectral radii $\rho(M_{GL-D})$, $\rho(M_{BGS-D})$, and $\rho(M_{SOR})$ exhibit the same behavior evidenced by Table 1. Fig. 4 plots the error components $|x_i(1) - x_i^*|$, for

D	$\rho(M_{OG-D})$	$\dfrac{R(\text{OG})}{R(\text{GS})}$	$\rho(M_{BGS-D})$	$\dfrac{R(\text{BGS})}{R(\text{GS})}$	$\rho(M_{SOR})$	$\dfrac{R(\text{SOR})}{R(\text{GS})}$
GS≡ 1	0.99227	1	–	–	0.93666	8.4
2	0.95354	6.13	0.97307	3.52	–	
3	0.85930	19.54	–	–	–	
4	0.71047	44.06	0.95525	5.90	–	
5	0.53687	80.15	–	–	–	
10	0.05264	379.42	0.92107	10.60	–	

Table 1: Spectral radii and convergence ratios of Example1.

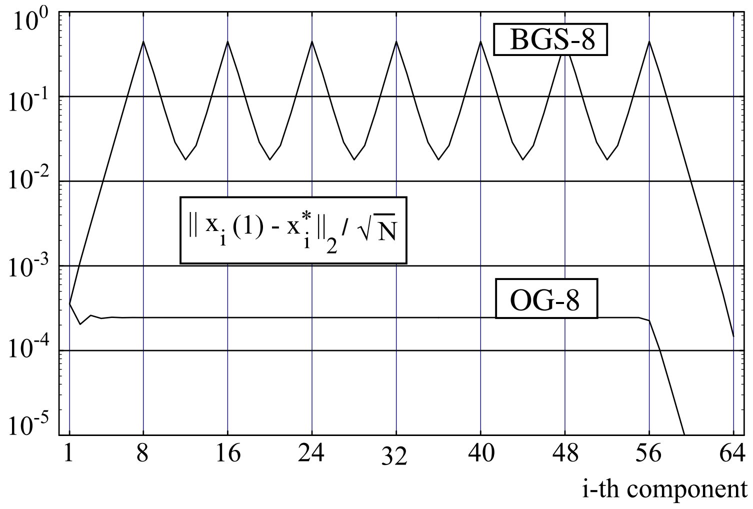

Figure 4: Error components for OG and BGS methods, after the first iteration.

$i = 1, \cdots, 64$, with $b = 0$, $x(0) = [1, \cdots, 1]^T$, and $D = 8$, after the first iteration, for the BGS and the OG methods, respectively. The BGS method displays errors much larger tan the the OG error, for any component. This was to be expected given the large magnitude of $\rho(M_{BGS-8})/\rho(M_{OG-8}) \simeq (0.17)/(2.72\,10^{-6})$. The BGS error exhibits a saw-type shape with maxima at $i = 8, 16, 24, 32, 40, 48$, and 56, and minima at $i = 1, 12, 20, 28, 36, 44$, and 52. A crude justification[8] is the following: the error of each variable inside each block increases with the errors of variables outside the block and decreases as the distance to the nearest boundary grows. In contrast, the OG method keeps the distance to the variable with larger error at a constant value of 8; the exception is the last block, this not being a problem (for both methods) if the variables in the last group do not depend on distant groups.

The *preconditioned conjugate gradient*(PCG) method, in the case of Toeplitz systems, can be ap-

[8]This argument is valid for problems displaying local interactions.

plied with great success (PCG) [16]. Namely, if the matrix is generated by a rational sequence (which is always the case of banded matrices), then the PCG complexity is $O(N \log N)$ [16], [31]. To be more precise, considering the preconditioner K_1 proposed in [16], the PCG method takes $N\big(2(2B+1)+6+6\ln_2 N\big)$ real additions and multiplications[9] per iteration. The preconditioning step takes $4.5\,N\ln_2 N$ addictions plus $1.5\,N\ln_2 N$ multiplications. On the other hand, the OG method (implemented by its GS equivalent) takes $N\big(2(2B)\big)$ operations per iteration plus a negligible overhead necessary to compute A' (recall that matrix A' is quasi-Toeplitz) and b'. Thus, if $\lim_{N\to\infty}\rho_{GS} < 1$, the OG method has, at least asymptotically, less complexity than the PCG method. On the other hand, for a given N, it can happen that the OG method produces a solution with acceptable error, with fewer operations than the PCG. This is illustrated in the next example.

Example 2: $a_{ij} = (3/5)^{|j-i|} + (-1/2)^{|j-i|},\ \ j-i \le 5$ and $a_{ij}=0$ for $\ j-i \ge 6$.

D	$\rho(M_{GL-D})$	$\frac{R(\text{OG})}{R(\text{GS})}$	$\rho(M_{BGS-D})$	$\frac{R(\text{BGS})}{R(\text{GS})}$	$\rho(M_{SOR})$	$\frac{R(\text{SOR})}{R(\text{GS})}$
GS≡ 1	0.29837	1	–	–	0.29197	1.018
2	0.27067	1.08	0.28569	1.04	–	
3	0.02351	3.10	–	–	–	
4	0.00855	3.93	0.18485	1.34	–	
5	0.00822	3.97	–	–	–	
10	0.00004	8.37	0.11790	1.77	–	

Table 2: Convergence ratios for Example 2.

Results displayed in Table 2 were computed using $N = 32$. As a general remark, the convergence rates exhibit the same pattern as the ones of Table 1. Again, we would like to recall the attention to the spectral radius $\rho(M_{OG-4}) = 0.00855$. The elements of matrix $A^{-1} = [r_{ij}]$ satisfy $|r_{ij}/r_{ii}| \ll 1$ if $|j-i| \ge 4$. Thus, one can say that $\rho(M_{GL-4}) = 0.00855$ is in accordance with Theorem 2. Fig. 5 plots the evolution of the Euclidian error $\|x(t) - x^*\|_2$ for GS, CG, OG, and PCG methods. The horizontal axis is graduated with a scale representing the ratio between the the number of operations per iteration N_{op} (of the different methods) and the number of operations per iteration of the OG method. For $N = 32$, $B = 5$ and $D = 4$ that ratio takes the value 2.9. The number of operations

[9]The term $6N\ln_2 N$ account for the preconditioning step; term $N(2(2B+1))$ account for the matrix-vector product; term $6N$ accounts the remaining vector-vector product. Notice that, in the case of banded matrices, its is better compute directly matrix-vector products than embedding it in a circular convolution an using FFT techniques to do it. Indeed, this last procedure would lead to a complexity $O(N\ln N)$ instead of the actual $N(2(2B+1))$.

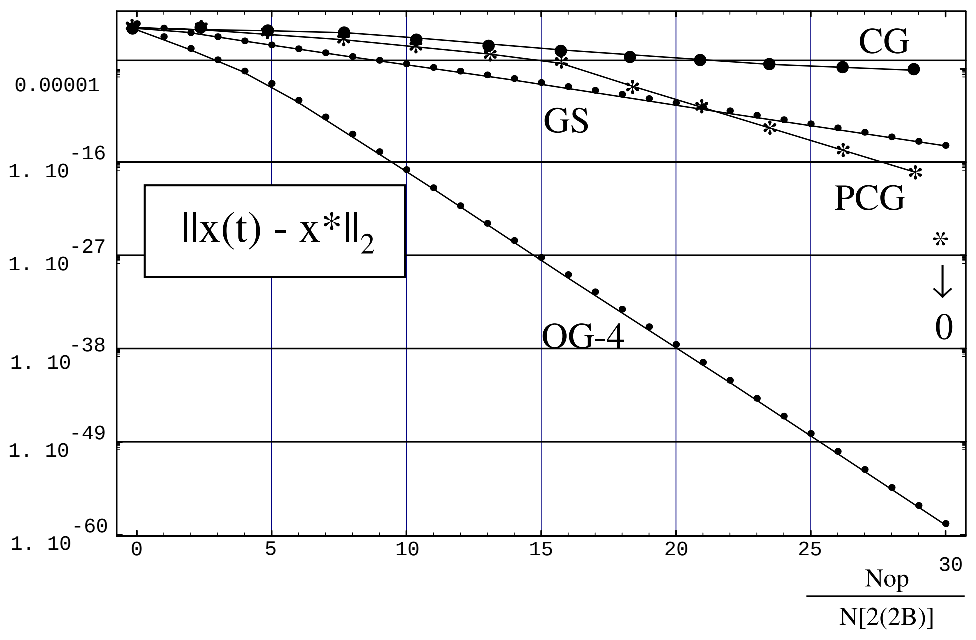

Figure 5: Euclidian error of the OG, GS, CG, and PCG methods, for successive iterations.

per iteration, for the GS and OG methods, are taken to be equal. The same is done with CG and PCG methods. Since the CG method takes $N\big(4(2B+1)\big)+6\big)$ operations per iteration, the true CG convergence rate is even worse than the one plotted in Fig. 5 .

The PCG method finds the exact solution at 11-th iteration. This result is in accordance with [16]: matrix A is generated by a rational sequence with $(p=5, q=0)$. Thus, the spectrum of $K_1^{-1}A$ has at most $2\max(p,q)$ eigenvalues different from one, and, consequently, the solution is found, at most, in $1+2\max(p,q)$ iterations. However, after the number of operations that PCG method needs to reach the 11-th iteration, the OG method outputs a solution with an error smaller that 10^{-50}.

We also ran this example with $N=128$. The ratio $\rho(M_{OG-4}, N=64)/\rho(M_{OG-4}, N=32)$ is 1.035. On the other hand, PCG method still converges in $1+2\max(p,q)$ iterations; however, it now takes 3.5 times more operations than the OG method. This would allow an error attenuation better than 10^{-70} if the OG method was run with the same number of operations.

It should be stressed that, given a system with dimension N, the comparison between PCG and OG method is not always as favorable to the latter. In fact, in order to achieve a competitive convergence ratio, it might be necessary such a high D that applying the OG method is no longer effective.

5.2 Quasi-regular matrices

Typically, in restoration problems a degraded version (the *observed signal/image*) of the *original signal/image* x is *filtered* in order to produce an *estimate* of the latter. Assuming that the data has zero mean, is observed under a linear operator B, and is contaminated with independent Gaussian zero mean noise, then, different approaches such as *Wiener filtering* [32], *constrained least square* [17], *Bayesian methods* [17], the *maximum entropy* principle [32], or the regularization approach [17], [18] demand the solution of the linear system $Ax = b$, where x is the estimate, b depends on the observed data, and A is of the form

$$A = BB^T + \eta P. \tag{45}$$

The meaning of η and P depends on the underlying paradigm. In the Bayesian framework matrix P models the *a priory* knowledge, and η depends on the noise to signal ratio [17]. On the other hand, in the regularization approach η is the regularization parameter and P is a matrix expressing constraints on unknown vector x [18].

In the one-dimensional example we are about to present, P express the constraints on pairs of components (x_i, x_{i+1}): either (x_{i+1}, x_i) is *continuous* (the difference $x_{i+1} - x_i$ must be *small*), or (x_{i+1}, x_i) is *discontinuous* (the difference $x_{i+1} - x_i$ can take arbitrary values). The i-th row of P express the continuity or discontinuity constraint as follows:

1. $[\ldots, 0, -1, 2, -1, 0, \ldots]$, if components $(x_{i-1}, x_i, x_{i+1}$ are *continuous*

2. $[\ldots, 0, 0, 1, -1, 0, \ldots]$, if components (x_i, x_{i+1}) are *continuous*, and components (x_{i-1}, x_i) are discontinuous

3. $[\ldots, 0, -1, 1, 0, 0, \ldots]$, if components (x_i, x_{i+1}) are *discontinuous*, and components (x_{i-1}, x_i) are continuous

4. $[\ldots, 0, 0, 0, 0, 0, \ldots]$, if components (x_i, x_{i+1}) are (x_{i-1}, x_i) are discontinuous

Model (45) with P just presented is related with the *weak string* model [33]. This model aims at the joint estimation of the discontinuities and the vector x. The exact solution, (carried out by maximizing a proper objective function), is analytically and computationally very demanding. Iterative algorithms achieving suboptimal solutions [33], [34], [35] have been proposed. A common procedure to all this algorithms is that, in each iteration, they need to compute the inverse of a matrix of the form (45). Matrix A is ill-conditioned for low signal to noise ratios, or ill-conditioned matrices BB^T, this leading to a high computational burden, concerning A^{-1} computation.

The following one-dimensional instance of (45) is going to be considered:

Example 3: $B = [b_{ij}]$, with $b_{ij} = \exp[-(\frac{|j-i|}{a})^2]$ (Gaussian blur). We assume that discontinuities occur at $i = 9, 14, 16, 19, 20, 30, 31$, and that $a = 3$ and $\eta = 0.1$.

D	$\rho(M_{GL-D})$	$\frac{R(\text{OG})}{R(\text{GS})}$	$\rho(M_{BGS-D})$	$\frac{R(\text{BGS})}{R(\text{GS})}$	$\rho(M_{SOR})$	$\frac{R(\text{SOR})}{R(\text{GS})}$
GS≡ 1	0.99043	1	–	–	0.97815	2.30
2	0.87747	13.59	0.99167	0.87	–	–
3	0.85908	15.80	–	–	–	–
4	0.68044	40.04	0.98123	1.97	–	–
5	0.63302	47.55	–	–	–	–
10	0.16127	189.75	0.97884	2.22	–	–

Table 3: Spectral radii and convergence ratios for Example 3.

For the present setting, the condition number of A is $\kappa(A) = 1742.52$, reflecting a severely ill conditioned matrix. We call the attention for the very low converge rate of the GS method: $R(\text{GS}) = 0.00417$. Notice that, with this figures, an attenuation of 10^{-6} over the initial error would take, approximately 1450 GS iterations. The ratio $R(\text{OG})/R(\text{GS})$ for $D = 4$ is 40. Hence, the same attenuation of 10^{-6} would take, approximately, 36 OG iterations. Gains of SOR and BGS methods (even for $D = 10$) over GS are, by far, smaller than the OG ones.

By varying parameter η in (45) from zero to ∞ one can give more importance to BB^T or, instead, to P. Table 4 displays the ratio $R(\text{OG})/R(\text{GS})$ for $D = 4$, $a = 3$, and $\eta = 10^i$ with $i = -2, \dots, 2$. Table 5 displays the ratio $R(OG)/R(GS)$ for $D = 4$, $\eta = 0.1$, and $a = 10^i$ with $i = -1, 0, 1$. Results, displayed in Tables 4 and 5 have the same behavior: they grow monotonically with the condition number $\kappa(A)$. Thus, the more ill-conditioned the system matrix, the greater the OG method convergence rate compared with that of the GS method.

$(a = 3)$ η	$\kappa(A)$	$\frac{R(\text{OG})}{R(\text{GS})}$
10^{-2}	8525.05	234.83
10^{-1}	1742.52	40.04
10^{0}	512.83	10.58
10^{1}	305.16	8.87
10^{2}	1886.16	27.12

Table 4: Convergence ratios of Example 3, for $a = 3$, $D = 4$, and η variable.

Example 4: Similar to Example 3, but with $N = 64$ and the discontinuities occuring at sites $9, 14, 16, 19, 20, 30,$ $31, 41, 46, 48, 52, 53, 62, 63$. Notice that for $i \geq 32$, the string is broken at sites displaced of a value 32 relative to Example 1.

$(\eta = 0.1)\quad a$	$\kappa(A)$	$\dfrac{R(\text{OG})}{R(\text{GS})}$
10^{-1}	1.39	4.93
10^{0}	11.58	13.52
10^{1}	1.2010^{6}	49.65

Table 5: Convergence ratios of Example 1, for $\eta = 0.1$, $D = 4$, and a variable.

The results from Example 3, displayed in Tables 3, 4, and 5, are similar of those of Example 5, up to the third digit.

6 Concluding Remarks

In this paper, we proposed an iterative algorithm for solving large systems of linear equations. Although applicable to a wide class of system matrices, it was firstly thought for symmetric and positive definite ones. The i-th and $(i+1)$-th iterations minimize the quadratic function $F(x) = \frac{1}{2}x^T Ax - b^T x$ with respect to the overlapped groups of variables S_i and S_{i+1}, respectively. Hence the name *overlapped group* (OG).

By overlapping groups in a suitable manner the following properties of the OG method emerge:

1. In the i-th iteration only a few components of the set S_i need to be computed. By exploring this feature the method can be implemented with a complexity of the order of the block iterative (e.g. the block Gauss-Seidel algorithm) methods complexity.

2. The *distance* between the component $x_i(t)$ being updated in iteration t and the components $x_j(t-1)$ from which $x_i(t)$ depends, is kept constant, or above of a minimum positive number D. Consequently, the less $x_i(t)$ depends on $x_j(t-1)$ the faster the OG method converges.

   The characteristic just emphasized is not presented in the block iterative methods; given a block of size D, there are variables whose distance in the sense above defined, ranges in the interval $1 < d(i,j) < D$. This fact is a major shortcoming of block iterative methods, concerning its convergence rate.

The choice of the iteration groups, adapted to each type of matrix, plays a central role in the OG method. Concerning this matter, we introduced the concept of *ordered covering*, which is a choice and ordering of the iteration groups. Two types of coverings were then proposed: (1) the one-dimension oriented covering $g_1(D)$; (2) the two-dimension oriented covering $g_2(D_1, D_2)$. In the case of covering $g_1(D)$, only one component need to be computed per iteration; in the case of covering $g_2(D_1, D_2)$,

D_2 components have to be computed per iteration. In 2D problems, if D_1 has the size of a line, each line can be treated as if it was a single component (the same is true for columns).

Assuming covering $g_1(D)$, two implementations were proposed: (1) based on the system $Ax = b$; (2) based on a Gauss-Seidel equivalent system $A'x = b'$, with $A' = TA$, $b' = Tb$, and T being an upper triangular D-banded matrix. The complexities of both implementations are of the same order; the first is suited for high convergence rates, and the second for slow convergence rates. The analysis of matrix A' led to the conclusion that the OG method can be thought of as a balance between *recursiveness* and *iterativeness.*

The product TA can also be viewed as *preconditioning* step on matrix A. Given that the OG method preserves the number of non-null diagonals, convergence is speeded up without increasing the computational effort in each iteration, which is unlike in classical preconditioning.

It was shown that the method converges for two classes of problems: (1) symmetric positive definite systems; (2) systems in which any subsystem is nonsingular and whose inverse matrix elements are null above (below) some upper (lower) diagonal. In class (2) the exact solution is reached in just one step. If the hypotheses of item (2) are not fulfilled, but instead elements of $A^{-1} = [r_{ij}]$ verify $|r_{ii}/r_{ij}| \ll 1$ for $j - i \geq D$, then it is predictable that the covering $g_1(D)$ leads to a high convergence rate. This was put in evidence by means of numerical examples.

A Appendix

A.1 Proof of Theorem 1

Let $\{x(n)\}$ and $x^* = A^{-1}b$ be, respectively, the sequence generated by the OG iteration defined in (5) and the solution of the SPD system $Ax = b$. The quadratic function $F(x) = \frac{1}{2}x^T Ax - b^T x$ computed at $x(t) = e(t) + x^*$ is given by $\frac{1}{2}(e^T(t)Ae(t) - b^T x^*)$. Thus, we assume that vector b is zero, since it only represents a shift on $F(x)$, and show that $F\big(x(t)\big) = \frac{1}{2}x(t)^T Ax(t) \to 0$ as $t \to \infty$, for an arbitary $x(0)$. Since A^{-1} exists, this is equivalent to showing that $x(t) \to 0$, for any $x(0)$.

Recall that the components being updated at the n-th iteration are those whose indices in the set S_i, with $i = \mathrm{saw}(n, n_g)$ (see OG definition in page 7). Formula (5) is equivalent to

$$x(n+1) = x(n) + \gamma s(n), \qquad n = 0, 1, \ldots \tag{46}$$

where $\gamma = 1$ for the OG method, and, since $b = 0$,

$$s(n) = (G_i^{-1}H_i - I)x(n). \tag{47}$$

Expanding $F(x)$ about $x(n)$ yields

$$F\big(x(n) + \gamma s(n)\big) = F\big(x(n)\big) + \gamma s(n)^T Ax(n) + \frac{\gamma^2}{2} s(n)^T As(n). \tag{48}$$

Having in attention that $s(n) = D_i s(n)$ and that $D_i A = (Q_i + \overline{Q}_i)$, one has successively

$$\begin{aligned} s(n)^T Ax(n) &= s(n)^T D_i Ax(n) && (49)\\ &= s(n)^T (Q_i + \overline{Q}_i)x(n) && (50)\\ &= s(n)^T (G_i - H_i)x(n) && (51)\\ &= -s(n)^T G_i s(n). && (52) \end{aligned}$$

Noting that $D_i AD_i = D_i G_i D_i$, then

$$\begin{aligned} s(n)^T As(n) &= s(n)^T D_i^T AD_i s(n) && (53)\\ &= s(n)^T D_i AD_i s(n) && (54)\\ &= s(n)^T G_i s(n). && (55) \end{aligned}$$

Hence, equation (48) becomes

$$F\big(x(n) + \gamma s(n)\big) = F\big(x(n)\big) - \gamma \left(1 - \frac{\gamma}{2}\right) s(n)^T G_i s(n). \tag{56}$$

Assuming that $\gamma \in (0,2)$, and defining $\beta = \min_i \lambda_1^i > 0$, with λ_1^i being the smallest eigenvalue of G_i, one can write the following inequality

$$F\big(x(n) + \gamma s(n)\big) \leq F\big(x(n)\big) - \gamma \left(1 - \frac{\gamma}{2}\right) \beta \, \|s(n)\|_2^2, \tag{57}$$

or

$$0 \leq F\big(x(n) + \gamma s(n)\big) \leq F\big(x(0)\big) - \gamma \left(1 - \frac{\gamma}{2}\right) \beta \sum_{i=0}^{n} \|s(i)\|_2^2, \tag{58}$$

which determines that

$$\sum_{i=0}^{i=\infty} \|s(i)\|_2^2 \leq \frac{1}{\beta} \frac{F\big(x(0)\big)}{\gamma\left(1 - \frac{\gamma}{2}\right)} < \infty. \tag{59}$$

Equation (59) implies that $\lim_{n\to\infty} s(n) = 0$. Thus, it follows from (46) that $\lim_{n\to\infty}[x(n+1) - x(n)] = 0$, which is equivalent to $\lim_{n\to\infty} A[x(n+1) - x(n)] = 0$. It is clear that $G_i s(n) = -D_i A x(n)$. Given that G_i is nonsingular, another consequence of having $\lim_{n\to\infty} s(n) = 0$ is that $\lim_{n\to\infty} D_i A x(n) = 0$, with $i = \text{saw}(n, n_g)$.

Consider the sum

$$u(n) = \sum_{i=1}^{n_g} D_i A x(n + i - 1), \tag{60}$$

which has null limit: $\lim_{n\to\infty} u(n) = 0$. Since $\lim_{n\to\infty} A[x(n+1) - x(n)] = 0$, terms $Ax(n+k)$, with $0 \leq k < n_g$ can recursively be replaced by $Ax(n) + \varepsilon_k(n)$, where

$$\varepsilon_k(n) = A \sum_{i=1}^{k} x(n+i) - x(n+i-1). \tag{61}$$

Notice that

$$\lim_{n\to\infty} \varepsilon_k(n) = \sum_{i=1}^{k} \lim_{n\to\infty} A[x(n+i) - x(n+i-1)] = 0. \tag{62}$$

Therefore, sum (60) becomes

$$0 = \lim_{n\to\infty} u(n) = \lim_{n\to\infty} \sum_{i=1}^{n_g} D_i[Ax(n) + \varepsilon_{i-1}(n)] \tag{63}$$

$$= \lim_{n\to\infty} \left(\sum_{i=1}^{n_g} D_i\right) Ax(n) + \lim_{n\to\infty} \sum_{i=1}^{n_g} D_i \varepsilon_{i-1}(n). \tag{64}$$

Given that $\sum_{i=1}^{n_g} D_i$ has rank N, and that $\lim_{n\to\infty} \varepsilon_i(n) = 0$, then $\lim_{n\to\infty} x(n) = 0$. **Q.E.D.**

A.2 Proof of Theorem 2

The iteration Matrix M is given by

$$M = \prod_{k=n_g}^{k=1} M_k, \text{ with } M_k = G_k^{-1} H_k. \tag{65}$$

Matrices G_k^{-1} with $k = 1, \cdots, n_g$ exist, since their determinants are those of submatrices obtained from A by choosing rows r and columns c, such that $r, c \in S_k$. But these submatrices have, by hypothesis, non-null determinant.

Denote the i-th row of matrix M_k by $[M_k]_i$. Having in account Definition 2 (matrices G_k and H_k), and remembering that segmentation $g_1(D)$ is under assumption, it follows that the first $k-1$ rows verify

$$[M_k]_i = [\delta(j-i),\ j = 1, \ldots, N] = e_i, \quad j = 1, \ldots, k-1. \tag{66}$$

Suppose that

$$[M_k]_k = [\underbrace{*, \cdots, *}_{k-1}, 0, \cdots, 0], \tag{67}$$

where the symbol $*$ denotes an arbitrary real number. An immediate consequence of (66) and (67) is that

$$[M_k M_{k-1} \cdots, M_1]_j = 0, \quad 1 \leq j \leq k, \quad k < n_g. \tag{68}$$

This can be checked by finite induction. On the other hand, segmentation $g_1(D)$ implies that

$$[M_{n_g}]_j = [\underbrace{*, \cdots, *}_{N-D}, 0, \cdots, 0], \ \ n_g \leq j \leq N. \tag{69}$$

Thus, it follows that $M = 0$.

To finish the proof, assumption (67) has to be proved. Since A^{-1} and G_k^{-1}, $k = 1, \ldots, n_g$ exist, then

$$\begin{aligned} D_k A &= (G_k - H_k), & (70)\\ D_k &= (G_k - H_k)A^{-1}, & (71)\\ G_k^{-1} D_k &= (I - G_k^{-1} H_k)A^{-1}, & (72)\\ [G_k^{-1} D_k]_k &= [A^{-1}]_k - [G_k^{-1} H_k]_k A^{-1}. & (73) \end{aligned}$$

Consider matrix M_1; rows $[G_1^{-1} D_1]_1$ and $[A^{-1}]_1$ have the following pattern:

$$\begin{aligned} [G_1^{-1} D_1]_1 &= [\underbrace{*, \cdots, *}_{D}, 0, \cdots, 0], & (74)\\ [A^{-1}]_1 &= [\underbrace{*, \cdots, *}_{B}, 0, \cdots, 0], & (75) \end{aligned}$$

where $D \geq B$ by hypothesis. Thus, one must have, from (73)

$$[G_1^{-1} H_k]_1 A^{-1} = [\underbrace{*, \cdots, *}_{D}, 0, \cdots, 0]. \tag{76}$$

Writing $[M_1]_1$ as

$$[M_1]_1 = [\underbrace{0, \cdots, 0}_{D}, \alpha_{D+1}, \ldots, \alpha_N], \tag{77}$$

equality (76) is equivalent to

$$[\alpha_{D+1}, \ldots, \alpha_N] A_1^{-1} = 0, \tag{78}$$

where A_1^{-1} stands for the submatrix of A^{-1} containing all rows r and all column c, such that $r \notin S_1$ and $c \notin S_1$. It is straighforward verifying that A_1^{-1} is non-singular, given that, by hypothesis, the correspondent submatrix defined in A is non-singular. Then vector $[\alpha_{D+1}, \ldots, \alpha_N] = 0$. This shows that $[M_1]_1 = 0$. For the remaining lines, the argument follows similar steps. **Q.E.D**

References

[1] D.M. Young, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.

[2] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation. Numerical Methods*, Prentice Hall, New Jersey, 1989.

[3] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, New York, 1996.

[4] G. H. Golub and C. F. Loan, *Matrix Computations*, Johns Hopkins University Press, 1983.

[5] J. Durbin, "The fitting of the time-series models," *Rev. Inst. Int. de Stat.*, vol. 28, pp. 233–344, 1960.

[6] N. Levinson, "The Wiener (root mean square) error criterion in filter design and prediction," *J. Math. Phys.*, vol. 25, pp. 261–278, 1947.

[7] G. S. Ammar and W. P. Gragg, "Superfast solution of real positive definite Toeplitz systems," *SIAM J. Matrix Anal. Appl.*, vol. 9, pp. 61–76, 1988.

[8] F. D. Hoog, "A new algorithm for solving Toeplitz systems of equations," *Lin. Algeb. Appl.*, vol. 88/89, pp. 123–138, 1987.

[9] B. W. Dinkinson, "Efficient solution of linear equations with banded Toeplitz matrices," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 27, pp. 421–422, 1979.

[10] W. E. Trench, "Solution of systems with Toeplitz matrices generated by rational functions," *Linear Algebra Applications*, vol. 74, pp. 191–211, 1986.

[11] W. E. Trench, "Toeplitz systems associated with the product of a formal Laurent series and a Laurent polynomial," *SIAM Journal on Matrix Analysis and Applications*, vol. 9, pp. 181–193, 1988.

[12] M. R. Hestenes and E. L. Stiefel, "Methods of conjugate gradients for solving linear systems," *J. Res. Nat. Bur. Standards Sect. 5*, vol. 49, pp. 409–436, 1952.

[13] D. Luenberger, *Linear and Nonlinear Programming*, Addison Wesley Publishing Company, Reading, Massachusetts, 1984, 2nd Edition.

[14] G. Strang, "A proposal for Toepliz matrix calculations," *Stud. Appl. Math.*, vol. 74, pp. 171–176, 1986.

[15] T. F. Chan, "An optimal circular preconditioner for Toeplitz systems," *SIAM J. Sci. Stat. Comput.*, vol. 9, pp. 766–771, 1988.

[16] T. Ku and C. Kuo, "Design and analysis of optimal Toeplitz preconditioners," *IEEE Transactions on Signal Processing*, vol. 40, pp. 129–141, 1992.

[17] H. C. Andrews and B. R. Hunt, *Digital Image Restoration*, Prentice Hall, New Jersey, 1977.

[18] A. Tikhonov, A. Goncharsky, and V. Stepanov, "Inverse problems in image processing," in *Ill-Posed Problems in the Natural Sciences*, A. Tikhonov and A. Goncharsky, Eds., pp. 220–232. Mir Publishers, Moscow, 1987.

[19] T. Chan and J. Shen, *Image Processing and Analysis: Variational, PDE, Wavelet, and Stochastic Methods*, SIAM, 2005.

[20] R. S. Varga, *Matrix Iterative Analysis*, Prentice-Hall, Englehood Cliffs, NG, 1962.

[21] R. Beauwens, "Iterative solutions methods," *Appl. Numer. Math.*, vol. 51, pp. 437–450, 2004.

[22] A. M. Ostrowski, "Iterative solution of linear systems of functional equations," *J. Math. Anal. App.*, vol. 2, pp. 351–369, 1961.

[23] L. Lapidus and G. F. Pinder, *Numerical Solution of Partial Differential Equations in Science and Engineering*, John Wiley & Sons, New York, 1982.

[24] D. O'Leary and R. E. White, "Multi-splittings of matrices and parallel solutions of linear systems," *SIAM J. Algebric Discrete Math.*, vol. 6,, pp. 630–640, 1985.

[25] O. A. Mcbryan and E. Van de Velde, “Parallel algorithms for elliptic equations,” *Comm. Pure Appl. Math.*, vol. 38, pp. 769–795, 1985.

[26] M. Newmann and R. J. Plemmons, “Convergence of parallel multisplittings and iterative methods for M-matrices,” *Linear algebra Appl.*, vol. 88/89, pp. 559–573, 1987.

[27] R. E. White, “Multisplittings of a symmetric positive definite matrix,” *SIAM J. Matrix Anal. Appl.*, vol. 11, pp. 69–82, 1990.

[28] L. J. Hayes, “A vectorized matrix-vector multiply and overlapping block iterative method,” in *Supercomputer Applications*, R.W. Numrich, Ed., pp. 91–100. Plenum Press, New York, 1984.

[29] W. Hackbush, *Multi-Grid Methods and Applications*, Springer-Verlag, New York, 1985.

[30] D. J. Evans, *Preconditioning Methods: Analysis and Applications*, Gordon and Breach, New York, 1983.

[31] T. Ku and C. Kuo, “Spectral properties of preconditioned rational Toeplitz matrices,” *SIAM Journal on Matrix Analysis and Application*, vol. 14, pp. 146–165, 1993.

[32] A. Jain, *Fundamentals of Digital Image Processing*, Prentice Hall, Englewood Cliffs, 1989.

[33] A. Blake and A. Zisserman, *Visual Reconstruction*, MIT Press, Cambridge, M.A., 1987.

[34] D. Geiger and F. Girosi, “Parallel and deterministic algorithms from MRF’s: Surface reconstruction,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-13, no. 5, pp. 401–412, May 1991.

[35] M. Figueiredo and J. Leitão, “Simulated tearing: an algorithm for discontinuity preserving visual suraface reconstruction,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition – CVPR’93*, New York, June 1993, pp. 28–33.