"Mircea cel Batran" Naval Academy Scientific Bulletin, Volume XIV – 2011 – Issue 2
Published by "Mircea cel Batran" Naval Academy Press, Constanta, Romania

PARALLEL ITERATIVE ALGORITHM FOR SOLVING LARGE LINEAR SYSTEMS

IOAN POPOVICIU[1]
[1]Senior Lecturer Ph.D., Naval Academy "Mircea cel Bătrân", Constanța

Abstract: *Solving large linear systems Ax = b is the optimal control problems can be reduced directly to solving a linear system, are reduced to quadratic programming problems in which the central place is occupied by solving a linear system. Also, such systems from mesh differential equations with values at risk, such as for example Poisson equation and Laplace equation. One of iterative methods for solving linear systems is iterative SOR method (success Overrelaxation). The paper proposes a parallel algorithm SOR method for solving Poisson's equation and a generalization of this algorithm for solving a certain system Ax = b. The generalization of the algorithm is based on partitioning the domain, resulting a parallel algorithm for solving a large linear system, together with the analyse of the convergence.*
Keywords: *parallel, iterative, mesh, processor*

1. INTRODUCTION

Let the system of linear equations

$$Ax = b, \quad A \in K^{I \times I}, \quad b \in K^I \tag{1.1}$$

The system has the solution for any $b \in K^I$ if matrix A is regular (nesingulară).

Definition An iterative method is a function (linear or nonlinear)

$$\phi : K^I \times K^I \to K^I$$

characterized the relations:

$$x^{(0)}(y,b) = y; \quad x^{(m+1)}(y,b) = \phi(x^{(m)}(y,b),b), m \geq 0 \tag{1.2}$$

where x^0 is the initial value of the string of iterations.

Definition $x^* = x^*(b)$ is called a **fixed point** of the iterative method ϕ if $x^* = \phi(x^*, b)$.

If the iteration $\{x^{(m)}\}$ is convergent, we have the following lemma:

Lemma 1.1. Let ϕ be one iteration. If $x^* = \lim_{m\to\infty} x^{(m)}(y,b)$ exists, then x * is fixed point of ϕ.

Definition Iterative method ϕ is called **consistent** for the system of equations $Ax = b$, if for any $b \in K^I$, any solution of $Ax = b$ is fixed point for ϕ.

Definition An iterative method ϕ is **convergent** if for any $b \in K^I$ there is a limit $x^*(b)$ of iterations (1.2) independent of initial value, $x^0 = y \in K^I$.

Theorem 1.1.[3] Let ϕ be a continuous function in the first argument. Then ϕ is consistent and convergent if:

i) *A* is nesingulară
ii) $Ax = b$ is true for any fixed point x of ϕ
iii) $\lim_{m\to\infty} x^{(m)}(y,b)$ exist for any $y, b \in K^I$.

Definition An iterative method ϕ is called linear if $\phi(x, b)$ is linear in *x* and *b*, adică exist the matrices *M* and *N* such that

$$\phi(x,b) = Mx + Nb \tag{1.3}$$

Theorem 1.2. An linear iteration $\phi(x,b) = Mx + Nb$ can be represented as:

$$x^{(m)}(x^{(0)},b) = M^m x^{(0)} + \sum_{k=0}^{m-1} M^k Nb, \quad \text{for m}\geq 0$$

Proof : The demonstration is by induction. For m=0 expression of the theorem is written: $x^{(0)}(x^{(0)}, b)=x^{(0)}$.

If the relationship is true for m-1 result: $x^{(m)}(x^{(0)},b) = Mx^{(m-1)} + Nb = M\left(M^{m-1}x^{(0)} + \sum_{k=0}^{m-2} M^k Nb\right) + Nb =$

$$M^m x^{(0)} + \sum_{k=1}^{m-1} M^k Nb + Nb$$

The matrix M is called the **iteration matrix** of ϕ. An iteration of the form (1.3) represented as:

$$x^{(m+1)} = Mx^{(m)} + Nb \quad (1.4)$$

is called the **first normal form** or **normal form one**.

For a consistent iteration ϕ, each solution of $Ax = b$ is a fixed point of ϕ, that is $x = Mx + Nb$. Then $x = Mx + NAx$ involving

$$M + NA = I$$

Then we have the following theorem:

Theorem 1.3. [18] A linear iteration ϕ is consistent if and only if the iteration matrix *M* is given by

$$M = I - NA \quad (1.5)$$

Moreover if A is nesingulară then

$$N = (I - M)A^{-1} \quad (1.6)$$

With the above representation

$$x^{(m+1)} = x^{(m)} - N(Ax^{(m)} - b), m \geq 0 \quad (1.7)$$

is called **secondary normal form**, and N is called **secondary normal matrix**.

Equation (1.7) shows that că $x^{(m+1)}$ is obtained from *x** with a time correction of the **defect (residual)** $Ax^{(m)} - b$) multiplied by the matrix *N*.

Observation: Secondary normal form (1.7) with $N \in K^{I \times I}$ arbitrary, represent all the linear and consistent iterations.

Third normal form is an iteration of the form:

$$W(x^{(m)} - x^{(m+1)}) = Ax^{(m)} - b \quad (1.8)$$

where W is called the matrix of the third normal form.

Observation: If W is nesingulară, iteration (1.8) coincides with secondary normal form (1.7), where $N = W^{-1}$.

If *x* is the solution of the system $Ax = b$ then:

$e^{(m)} = x^{(m)} - x$ is called **error** of the iteration $x^{(m)}$. (1.9)

From $x^{(m+1)} = Mx^{(m)} + Nb$ and $x = Mx + Nb$ by subtraction we obtain:

$$e^{(m+1)} = Me^{(m)}, \ m \geq 0 \text{ şi } e^{(0)} = x^{(0)} - x. \quad (1.10)$$

Result

$$e^{(m+1)} = Me^{(m)}e^{(0)}, \ m \geq 0 \quad (1.11)$$

Note the **defect (residual)** $Ax^{(m)} - b$ with

$$d^{(m)} = Ax^{(m)} - b \quad (1.12)$$

Observation: a) The defect $\bar{d} = A\bar{x} - b$ and the error $\bar{e} = \bar{x} - x$ satisfy the equation

$$A\bar{e} = \bar{d} \quad (1.13)$$

b) The defect satisfy the equations:

$$d^{(m+1)} = AMA^{-1}d^{(m)}, \quad d^{(0)} = Ax^{(0)} - b, \quad d^{(m)} = \left(AMA^{-1}\right)^m d^{(0)} \quad (1.14)$$

The convergence of iterative methods is given by the following necessary and sufficient condition:

Theorem 1.4. [10] A linear iterative method $\phi(x, b) = Mx + Nb$ is convergent if and only if $\delta(M) < 1$ where $\delta(M)$ is the spectral range of the matrix A. $\delta(M)$ is called the **convergence rate** of iterations ϕ.

2. ITERATIVE METHOD SOR (SUCCESIVE OVERRELAXATION)

SOR method is an iterative method in which an iteration is defined by:

$$x^{(m+1)} = M_\omega^{\text{SOR}} x^{(m)} + N_\omega^{\text{SOR}}$$

where

$$M_\omega^{\text{SOR}} = (I - \omega L)^{-1}[(1-\omega)I + \omega U] = (D - \omega F)^{-1}[(1-\omega)D + \omega F]$$

$$N_\omega^{\text{SOR}} = \omega(I - \omega L)^{-1}D^{-1} = \omega(D - \omega E)^{-1}$$

$$L = D^{-1}E, \ U = D^{-1}F, \ A = D - E - F$$

D = diag {*A*}
E = strictly lower triangular matrix
F = strictly upper triangular matrix
ω is a amortization factor.

The matrix of the normal form is:

$$W_{\omega}^{\mathrm{SOR}} = \frac{1}{\omega}\left(D - \omega E\right) = \frac{1}{\omega}D - E$$

Observation:

a) If $0 < \omega < 1$ the method is called underrelaxation method;

b) If $\omega = 1$ the SOR method coincides with method Gauss – Seidel;

c) If $\omega > 1$ the method is called overrelaxation method;

The SOR method represented on components is defined by:

$$x_i^{(m+1)} = \left[\omega\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(m+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(m)}\right) + (1-\omega)a_{ii}x_i^{(m)}\right] / a_{ii}, i = 1,2,...,n$$

In matrix terms SOR method is defined by:

$$x^{(m+1)} = \left(D - \omega L\right)^{-1}\left(\omega U + (1-\omega)D\right)x^{(m)} + \omega\left(D - \omega L\right)^{-1}b$$

The string vector x (0), x (1), x (2), ... build with the normal form SOR converges to solution x if:

$$\max_i \frac{1}{|a_{ii}|}\sum_{\substack{j=1 \\ j\neq i}}^{n}|a_{ij}| < 1 \text{ şi } 0<\omega<2$$

SOR method has a sequential update of the form:

$$x_n^{(k)} \text{ cere } x_{n-1}^{(k)}\ldots$$
$$x_{n-1}^{(k)} \text{ cere } x_{n-2}^{(k)}$$
$$\vdots$$
$$x_2^{(k)} \text{ cere } x_1^{(k)}, \text{etc.}$$

For parallelization SOR method we propose a solution based on partitioning the domain, resulting a new method which we call iterative method PSOR.

3. PSOR ITERATIVE METHOD FOR POISSON EQUATION

We consider the model problem (Poisson equation) in the space of two dimensions, defined by:

$$-\Delta u(x,y) = f(x,y) \text{ for } x,y \in \Omega = (0,1)\times(0,1)$$
$$u(x,y) = \varphi(x,y) \text{ on } \Gamma = \partial\Omega$$

For simplicity we consider the case $u(x,y) = 0$ on Γ. Discretizing differential equatione, the domain Ω is covered with a grid $(n+1)\times(n+1)$ with grid size $h = 1/(n+1)$. Each point of the grid has coordinates $x = ih, y = jh \quad (0 \le i,j \le n+1)$. If $u(i,j)$ is solution approximated in $x = ih$, $y = jh$, then an approximation of the Poisson equation is given by the 5 points formula:

$$4u(i,j) - u(i-1,j) - u(i+1,j) - u(i,j-1) - u(i,j+1) = b(i.j)$$

where $b(i,j) = -f(ih,jh)h^2$

Linear equation above is true for $1 \le i,j \le n$ and so we have $N = n^2$ equations with *N* unknown number of interior points of the grid.

Poisson equation discretized with 5-points formula lead to the following linear algebraic system:

$$4u_{ij} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1} = h^2 f_{ij} \text{ în } \Omega_h$$
$$u_{ij} = 0 \text{ pe } \partial\Omega_h$$

which can be written as matrix

$$AU = F$$

SOR method using natural ordering on line with an initial value $u_{ij}^{(0)}$ and a real number $\omega \in (0,2)$ is defined by a sequence of form:

$$u_{ij}^{(k+1)} = (1-\omega)u_{ij}^{(k)} + \frac{\omega}{4}\left(h^2 f_{ij} + u_{i-1,j}^{(k+1)} + u_{i,j-1}^{(k+1)} + u_{i+1,j}^{(k)} + u_{i,j+1}^{(k)}\right)$$

SOR method with red-black ordering, called method R / B, defines first the iterations in the red points by:

$$u(i,j)^{(k+1)} = (1-\omega)u(i,j)^{(k)} + \omega\left[u(i-1,j)^{(k)} + u(i+1,j)^{(k)} + u(i,j-1)^{(k)} + u(i,j+1)^{(k)} + h^2 f_{ij}\right]/4.$$

Then iterations in black points by:

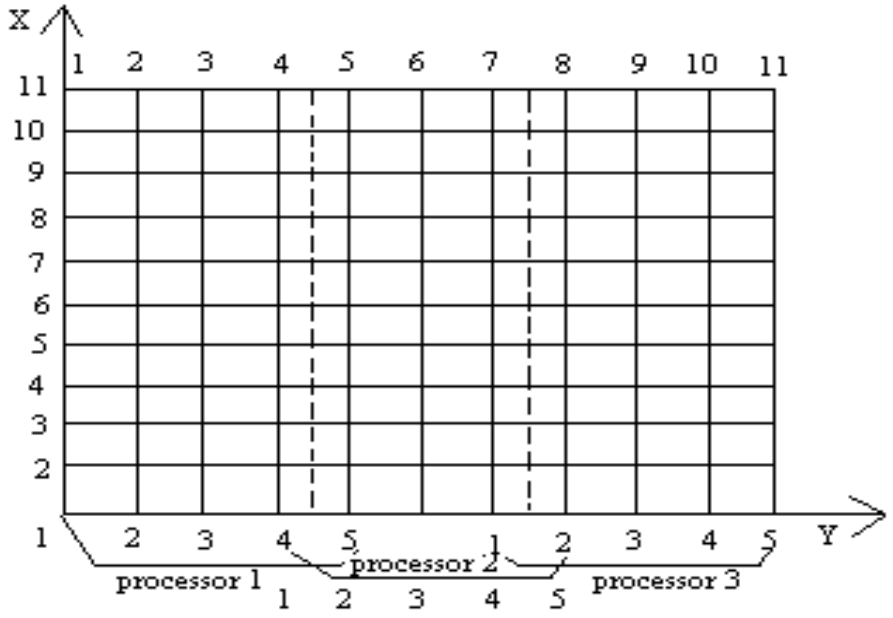
$$u(i,j)^{(k+1)} = (1-\omega)u(i,j)^{(k+1)} +$$
$$\omega\left[u(i-1,j)^{(k+1)} + u(i+1,j)^{(k+1)} + u(i,j-1)^{(k+1)} + u(i,j+1)^{(k+1)} + h^2 f_{ij}\right]/4$$

To implement the PSOR method on *p* processors, the mesh Ω_h is decomposed into *p* disjoint subgrid meshes $\Omega_{h,\nu}$ such that

$$\Omega_h = \bigcup_{\nu=1}^{p} \Omega_{h,\nu}.$$

For simplicity, each subgrid mesh $\Omega_{h,\nu}$ *is assumed to be a strip comprising the node points on* $(n-2)/p$ *consecutive horizontal (or vertical) grid lines, where* n – 2 *is assumed to be divisible by the positive integer* p*. The nodes adjacent to each strip are used as the boundary nodes of the strip and thus the reference to a strip will be made to strip together with boundary nodes and will be called extended strip.*

As each strip has one or more common grid lines with extended neighbor strip. Each extended strip will be assigned on one processor. The lines common grid at two extende strips represent the communication between 2 processors. Thus if the strip ν is assigned to the processor ν, the new iterations associated with points on the first line of the grid domain $\Omega_{h,\nu}$ are send to the processor $\nu-1$ by the processor ν, $2 \le \nu \le p$, as soon as they are calculated. Iterations send by the processor ν, to processor $\nu-1$ used by the processor $\nu-1$ in iterations associated with points on the last line of the strip associated with him.
A exemple with *p* = 3 and *n*=11 is illustrated by next figure:



Each strip comprises three grid lines, and has two interior boundary grid lines. Thus, each extended strip consists of five grid lines, wich have been locally numbered in Y-axis. For example, strip 2 comprises grid lines 5, 6, and 7, and grid lines 4 and 8 are its interior boundary. So, the second extended strip comprises five grid lines; grid lines 4,5,6,7 and 8. Grid lines 5 and 7 in strip 2 also serve as the right boundary of strip 1 and the left boundary of strip 3, respectively.Hence, when the iterates on grid line 5 are updated and immediately sent to processor 1, wich is equivalent to updating the right boundary condition of strip 1,they will be in updating the iterates on grid line 4 at processor 1.

To get the iteration expressionof the PSOR method for Poisson equation, we need to decompose each strip $\Omega_{h,\nu}$ into three sub-strips $\Omega_{h,\nu} = \Omega_{h,\nu}^1 \cup \Omega_{h,\nu}^2 \cup \Omega_{h,\nu}^3$, where $\Omega_{h,\nu}^1$ and $\Omega_{h,\nu}^3$ contain the mesh points on the first and the last grid lines of the strip $\Omega_{h,\nu}$, respectively, and $\Omega_{h,\nu}^2$ contain the remaining part. Then the PSOR iterations are defined by:

$$u_{ij}^{(k+1)} = \frac{\omega}{4}\left(h^2 f_{ij} + u_{i-1,j}^{(k)} + u_{i,j-1}^{(k+1)} + u_{i+1,j}^{(k)} + u_{i,j+1}^{(k)}\right) + (1-\omega)u_{ij}^{(k)} \text{ pe } \Omega_{h,\nu}^1$$

$$u_{ij}^{(k+1)} = \frac{\omega}{4}\left(h^2 f_{ij} + u_{i-1,j}^{(k+1)} + u_{i,j-1}^{(k+1)} + u_{i+1,j}^{(k)} + u_{i,j+1}^{(k)}\right) + (1-\omega)u_{ij}^{(k)} \text{ pe } \Omega_{h,\nu}^2$$

$$u_{ij}^{(k+1)} = \frac{\omega}{4}\left(h^2 f_{ij} + u_{i-1,j}^{(k+1)} + u_{i,j-1}^{(k+1)} + u_{i+1,j}^{(k)} + u_{i,j+1}^{(k+1)}\right) + (1-\omega)u_{ij}^{(k)} \text{ pe } \Omega_{h,\nu}^3$$

where $\nu = 1,2,\mathrm{K}, p$ and $u_{ij}^{(k)}$ represents the k-th iterate at mesh point (x_i, y_i).
Each of the three subprobleme is resolved with SOR method by each processor.

4. THE GENERALIZATION PSOR METHOD
Let the linear system

$$Au = f \qquad (4.1)$$

where $A \equiv (a_{ij})_{n\times n}$ is a sparse symmetric positive definite matrix, *f* is a given real column vector of order *n* and *u* is a unknown column vector. We assume that (4.1) arises from a finite element or a finite different discretization of an elliptic boundary value.

Let Ω_h denote the set of mesh points on wich the unknown vector *u* is defined. Mesh point μ is said to be coupled to mesh point ν provided that the corresponding entry $a_{\mu\nu}$ of matrix *A* is not zero. Since (4.1) arises from a finite element or a finite different discretization, the mesh domain Ω_h can be partitioned into *p* disjoint subgrids $\Omega_{h,i}$ $(1 \le i \le p)$, and each $\Omega_{h,i}$ can further be divided into three disjoint parts $\Omega_{h,i}^1, \Omega_{h,i}^2, \Omega_{h,i}^3$, such that $\Omega_h = \bigcup_{i=1}^{p} \Omega_{h,i}$ şi

$$\Omega_{h,i} = \Omega_{h,i}^1 \cup \Omega_{h,i}^2 \cup \Omega_{h,i}^3$$

where $\Omega_{h,i}^1$ and $\Omega_{h,i}^3$ are nonempty subsets of $\Omega_{h,i}$ and comprise the mesh points of $\Omega_{h,i}$ that are coupled to the mesh points of $\Omega_{h,j}$ with i<j, and j>*i*, respectively, and $\Omega_{h,i}^2 = \Omega_{h,i} - \Omega_{h,i}^1 - \cup\Omega_{h,i}^3$ wich may be empty.

Based on this mesh partition, we assume that every *n* - vector *u* is decomposed into subvectors:

$$u = (U_1, U_2, \ldots, U_p)^T \qquad (4.2)$$

and each sub-vector U_i is further decomposed as:

$$U_i = \left(U_i^1, U_i^2, U_i^3\right)^T, \quad i = 1,\ldots,p \qquad (4.3)$$

where U_i^1, U_i^2, U_i^3 comprise the components of *u* associated with the mesh points on the subgrid $\Omega_{h,i}^1, \Omega_{h,i}^2$ and $\Omega_{h,i}^3$, respectively.

Partitions (4.2) and (4.3) induce a block partition $A = (A_{ij})$ in:

$$A_{i.j} = \begin{bmatrix} A_{ij}^{11} & A_{ij}^{12} & A_{ij}^{13} \\ A_{ij}^{21} & A_{ij}^{22} & A_{ij}^{23} \\ A_{ij}^{31} & A_{ij}^{32} & A_{ij}^{33} \end{bmatrix}, \quad 1 \le i, j \le p$$

For $A_{ij} \neq 0$ with $i > j$ we have

$$A_{ij} = \begin{bmatrix} A_{ij}^{11} & A_{ij}^{12} & A_{ij}^{13} \\ A_{ij}^{21} & A_{ij}^{22} & A_{ij}^{23} \\ A_{ij}^{31} & A_{ij}^{32} & A_{ij}^{33} \end{bmatrix}, \qquad (4.4)$$

where $A_{ij}^{13} \neq 0$.

In fact, let $a_{\mu\nu}$ be an nonzero entry of A_{ij} with $j < i$. Mesh point μ, wich is in $\Omega_{h,i}$ is then coupled to the mesh point ν in $\Omega_{h,j}$. Thus, the mesh point μ belongs to $\Omega_{h,i}^{1}$. Similary, noting that the symmetry of *A* gives $a_{\mu\nu} = a_{\mu\nu} \neq 0$, we can show the mesh point ν is in $\Omega_{h,i}^{3}$. Hence, $a_{\mu\nu} \in A_{ij}^{13}$. Because all of the nonzero entries of A_{ij} are contained in A_{ij}^{13} *with* $j < i$, the other matrices $A_{ij}^{\mu\nu}$ must be zero.

Similary, for $A_{ij} \neq 0$ with $i < j$

$$A_{ij} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ A_{ij}^{31} & 0 & 0 \end{bmatrix}, \quad j > i \qquad (4.5)$$

where $A_{ij}^{31} \neq 0$.

Using (4.2) – (4.6) we can write the linear system (4.1) into a block structure as follows:

$$\begin{bmatrix} A_{ii}^{11} & A_{ii}^{12} & A_{ii}^{13} \\ A_{ii}^{21} & A_{ii}^{22} & A_{ii}^{23} \\ A_{ii}^{31} & A_{ii}^{32} & A_{ii}^{33} \end{bmatrix} \begin{bmatrix} U_i^1 \\ U_i^2 \\ U_i^3 \end{bmatrix} = \begin{bmatrix} F_i^1 - \sum_{j-1}^{i-1} A_{ij}^{13} U_j^3 \\ F_i^2 \\ F_i^3 - \sum_{j-i+1}^{p} A_{ij}^{31} U_j^1 \end{bmatrix}, 1 \leq i \leq p \qquad (4.6)$$

According to the above block form of the linear system (4.1), the *k*-th iterate of PSOR method can be denoted $u^{(k)} = \left(U_1^{(k)}, U_2^{(k)}, \ldots, U_p^{(k)}\right)^T$ where $U_i^{(k)} = \left(U_i^{1,(k)}, U_i^{2,(k)}, U_i^{3,(k)}\right)$. We assume that the subproblem (4.6) on $\Omega_{h,i}$ is implemented on processor $i, 1 \leq i \leq p$. Then, the PSOR method is defined by the following algorithm..

The algorithm PSOR

The *k*-th iterate of the PSOR method $u^{(k)} = \left(U_1^{(k)}, U_2^{(k)}, \ldots, U_p^{(k)}\right)^T$ with $U_i^{(k)} = \left(U_i^{1,(k)}, U_i^{2,(k)}, U_i^{3,(k)}\right)$ is defined by:

1) For $i = 1,2,..., p$ in parallel: $U_i^{1,(k+1)}$ is obtained by applying one SOR sweep to the subproblem on $\Omega_{h,i}^{1}$

$$A_{ii}^{11} U_i^1 = F_i^1 - A_{ii}^{12} U_i^{2,(k)} - A_{ii}^{13} U_i^{3,(k)} - \sum_{j=1}^{i-1} A_{ij}^{13} U_j^{3,(k)}$$

2) Communicate $U_i^{1,(k+1)}$ to other processors as needed.

3) For $i = 1,2,..., p$ in parallel: $U_i^{2,(k+1)}$ *and* $U_i^{3,(k+1)}$ are obtained by applying one SOR sweep to the subproblem

$$A_{ii}^{22} U_i^2 = F_i^2 - A_{ii}^{21} U_i^{1,(k+1)} - A_{ii}^{23} U_i^{3,(k)} \; pe \, \Omega_{h,i}^2$$

and the problem

$$A_{ii}^{33} U_i^3 = F_i^3 - A_{ii}^{31} U_i^{1,(k+1)} - A_{ii}^{32} U_i^{2,(k+1)} - \sum_{j=i+1}^{p} A_{ij}^{31} U_j^{1,(k+1)} \; pe \, \Omega_{h,i}^3$$

respectively.

4) Communicate $U_i^{3,(k+1)}$ to other processors as needed.

To get a matrix expression of the PSOR method, we split matrix $D^{-1}A$ into

$$D^{-1}A = I - L - U \tag{4.7}$$

where $D = \text{diag}\{a_{11}, a_{22}, \text{K}, a_{nn}\}$, L and U are strictly lower and upper triangular $n \times n$ matrices, whose entries are the negatives of the entries of $D^{-1}A$, respectively, below and above the main diagonal of A.

Similary, we split submatrix $D_i^{-1}A_{ii}$ into

$$D_i^{-1}A_{ii} = I_i - L_i - U_i \tag{4.8}$$

where $D_i = \text{diag}\{A_{ii}\}$. Further, we split both *L* and *U* into

L=B+N and U=C+M (4.9)

where

$$B = \begin{bmatrix} L_1 & 0 & ... & 0 \\ 0 & L_2 & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & \text{L} & L_p \end{bmatrix}, \qquad C = \begin{bmatrix} U_1 & 0 & ... & 0 \\ 0 & U_2 & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & U_p \end{bmatrix}$$

$$N = -D^{-1}\begin{bmatrix} 0 & 0 & 0 & ... & 0 \\ A_{21} & 0 & 0 & ... & 0 \\ A_{31} & A_{32} & 0 & ... & 0 \\ ... & ... & ... & ... & ... \\ A_{p1} & A_{p2} & ... & A_{p,p-1} & 0 \end{bmatrix} \quad M = -D^{-1}\begin{bmatrix} 0 & A_{12} & A_{13} & ... & A_{1p} \\ 0 & 0 & A_{23} & ... & A_{2p} \\ ... & ... & ... & ... & ... \\ 0 & 0 & ... & 0 & A_{p-1,p} \\ 0 & 0 & 0 & ... & 0 \end{bmatrix}$$

With this notations, the PSOR method, defined by PSOR algorithm, can be expressed by:

$$u^{(k+1)} = M_{PSOR}(\omega)u^{(k)} + F(\omega),\ k = 0,1,2,... \tag{4.10}$$

where

$$M_{\text{PSOR}}(\omega) = \left[I - \omega(B+M)\right]^{-1}\left[(1-\omega)I + \omega(C+N)\right] \tag{4.11}$$

$$F(\omega) = \omega\left[I - \omega(B+M)\right]^{-1}D^{-1}f$$

$M_{\text{PSOR}}(\omega)$ is called the PSOR iteration matrix.

5. The convergence of the PSOR method

A necessary condition for the convergence of the PSOR method is given in the following theorem.

Theorem 5.1 **Let $M_{\text{PSOR}}(\omega)$ be the PSOR iteration matrix. If $\delta(M_{\text{PSOR}}(\omega)) < 1$, i.e. the PSOR method is convergent, then**

$$0 < \omega < 2.$$

Proof Let *det(A)* be the determinant of a matrix A. By a well-know theorem of linear algebra, we know that det(A) is equal to the product of the eigenvalues of A. With the definitions of *B, C, M* and *N* in (4.9), we have

$$I - \omega(B+M) = \begin{bmatrix} I_1 - \omega L_1 & \omega A_{12} & \cdots & \omega A_{1p} \\ 0 & I_2 - \omega L_2 & \cdots & \omega A_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & I_p - \omega L_p \end{bmatrix}$$

and

$$(1-\omega)I+\omega(C+N)=\begin{bmatrix}(1-\omega)I_1+\omega U_1 & 0 & \cdots & 0\\ -\omega A_{21} & (1-\omega)I_2+\omega U_2 & \cdots & 0\\ \vdots & & \ddots & \\ -\omega A_{p1} & -\omega A_{2p} & \cdots & (1-\omega)I_p+\omega U_p\end{bmatrix}$$

Thus

$$\det(I-\omega(B+M))=\prod_{i=1}^{p}\det(I_i-\omega L_i)=1 \qquad (5.1)$$

and

$$\det(M_{PSOR}(\omega))=\det(I-\omega(B+M))\det(M_{PSOR}(\omega))=$$
$$\det[(I-\omega(B+M))M_{PSOR}(\omega)]=\det((1-\omega)I+\omega(C+N))=$$
$$\prod_{i=1}^{p}\det((1-\omega)I_i+\omega U_i)=(1-\omega)^n$$

Therefore

$$|1-\omega|\le\delta(M_{PSOR}(\omega))<1 \qquad (5.2)$$

From witch it follows $0<\omega<2.$

A sufficient condition for the PSOR method is given in the following theorem.

Theorem 5.2. Let matrix A be symmetric positive definite. If Dacă $0<\omega<2$, then $\delta(M_{PSOR}(\omega))<1$, i.e., the PSOR method converges.

Proof Let x be an eigenvector of $M_{PSOR}(\omega)$ and λ be its corresponding eigenvalue such that:

$$M_{PSOR}(\omega)x=\lambda x$$

By (4.11), the above equality is written as:

$$[(1-\omega)I+\omega(C+N)]x=\lambda[I-\omega(B+M)]x \qquad (5.3)$$

Multiplying x^TD to the both sides of (5.3), we find

$$\lambda=\frac{(1-\omega)x^TDx+\omega x^TDCx+\omega x^TDNx}{x^TDx-\omega x^TDBx-\omega x^TDMx} \qquad (5.4)$$

Set $q=x^TDx>0,\ x^TDBx=\alpha+i\beta$ and $x^TDNx=r_1+ir_2,\ \alpha,\beta,r_1,r_2\in R$. Clearly, the symmetry of matrix A gives that $x^TDCx=\alpha-i\beta$ and $x^TDMx=r_1r_2$. By these notations, (5.4) becomes

$$\lambda=\frac{(1-\omega)q+\omega(\alpha+r_1)+i\omega(r_2-\beta)}{q-\omega(\alpha+r_1)+i\omega(r_2-\beta)}$$

Thus,

$$|\lambda|^2=\frac{[(1-\omega)q+\omega(\alpha+r_1)]^2+\omega^2(\beta-r_2)^2}{[q-\omega(\alpha+r_1)]^2+\omega^2(\beta-r_2)^2}$$

By the positive definiteness of A, we have that for any $x\neq 0.$

$$0<x^TAx=x^T(D-DB-DC-DN-DM)x=q-2(\alpha+r_1)$$

So, when $0<\omega<2,$

$$[(1-\omega)q+\omega(\alpha+r_1)]^2-[q-\omega(\alpha+r_1)]^2=(2-\omega)\omega[2(\alpha+r_1)-q]q<0.$$

Hence, $|\lambda|<1.$ This follows that $\delta(M_{PSOR}(\omega))<1.$

6. NUMERICAL EXAMPLE

In parallel algorithm is needed to use operations: scalar product, rare matrix-vector product. Using library functions and macros Message Passing Interface (MPI), to solve linear $Ax=b$, with a matrix N×N=960×960, with 8402 nonzero elements, which gave the following results:

Număr procesoare	Număr iterații	Timp calcul	Timp comunicare între procesoare	Timp pentru alocarea memoriei	Timp pentru operații cu vectori	Timp total
1	205	3.074	0.027	0.002	0.281	3.384
2	205	2.090	0.341	0.002	0.136	2.568

- time is given in minutes.

Applying parallel algorithm PSOR in solving Poisson's equation on a grid of 200×200 was obtained:
a) Cost calculation / iteration: close to 0(N).
b) communication cost: $0(N^{1/2})$.

Number of processors	**Tolerance error**	**Number iterations**	Execution time
1	**6E-3**	216	5.7
2	**6E-3**	216	4.0

7. CONCLUSION

In parallel algotihm each processor execute *k* iterations of algorithm in parallel.
Defining:

b=block size of matrix *A* and vectors $x, r = b - Ax$ on each node

p=number of processors
T_{comp1p}=total time to update blocks of vectors on each processor
T_{comp2p}=total time to compute and comunicate *A* and (*r*, *r*)
T_{comp3p}=total time for the computation of the inner products and global communication
T_{comp4p}=total time to compute scalars
Here T_{comp1p} is the total time for 3 computation to update the vectors. It is observed that when matrix *A* is very sparse (density less than 5 percents), time exceeds the computation time. Thus, T_{comp2p} is taken equal to t_{comm}, the time to communicate a block of size *b* across *p* processors. T_{comp3p} involves time for global communication and computation. It is t_{glb}. Then:

$$T_{par} = T_{comp1p} + T_{comp2p} + T_{comp3p} + T_{comp4p}$$

where:

$$T_{comp1p} = 3 * b * k * t_{comp}$$
$$T_{comp2p} = t_{comm}$$
$$T_{comp3p} = 2 * b * k * t_{comp} + t_{\mathrm{glb}}$$
$$T_{comp4p} = 2 * k * t_{comp\alpha}$$

REFERENCES:

[1] Alefeld G. On the convergence of the symmetric SOR method for matrices with red - block. Numer. Math. 39 [1982].
[2] Axelsson O. Solution of liniar systems of equations: iterative methods. In: Barker [1] 1-51.
[3] Braess D. Finite element. Springer-Verlag 1992.
[4] Chan T, Glawinski J. Domain decomposition methods. Houston. SIAM. Philadelphia 1990.
[5] Eiermann M, Marek J. On the solution of singular systems of algebraic equations by semiiterative methods. Numer. Math. 53. (1988).
[6] Eiermann M, Niethammer W. Optimal succesive averrelaxation iterative methods for p-cyclic matrices. Numer. Math. 57. (1990).
[7] George J. Solution of liniar systems of equations: Direct methods for finite element problems. In Barker [1] 52-101.
[8] Golub G, Van Loan. Matrix computations. Notrh Oxford Academic, Oxford 1983.
[9] Haase G., Hommel Th.,Meyer A., Pester M. Biblioteken zur Entwicklung parallel Algorithmen.Preprint SPC 95_20, Technische Universitat Chemnitz-Zwickau, Fakultat fur Mathematik, 1995
[10] Hackbusch W, Trotienberg. Multi-grid methods. Bonn. 1990.
[11] Heise B,Jung M. Comparison of parallel solvers for nonlinear elliptic problems based on domain decomposition ideas.Institutbericht Nr.494,Johannes Kepler Universitat Linz,Institut fur Mathematik,1995
[12] Ichim I, Marinescu G. Metode de aproximare numerică. Ed. Acad. Bucureşti 1986.
[13] Kuznetsev A. Algebraic multigrid domain decomposition methods. Anal. and Math. 1989.
[14] Mandel J. On block diagonal and Schur complement preconditioning. Numer. Math. 58 (1990).
[15] McCormick S. Multigrid methods. SIAM, Philadelphia, 1987.
[16] Oden J, Reddy J. An introduction to the mathematical theory of finite elements. New Tork, 1976.
[17] Ortega J.M.,Voigt R.G. Solution of partial differential equation on vector and parallel computers, SIAM Rev 27(1985),149-270
[18] Varga R., Matrix Iterative Analysis, Prentice Hall, Englewood Cliff, New Jersey, 1962.
[19] Yong D.M. Iterative solution of large linear systems, Academic Press, New York, 1971.
[20] ---- An Introduction to MPI for C Programmers, The University of Texax at Austin, 1999.
[21] ---- A User's Guide to MPI, Departament of Mathematics, University of San Francisco, 1998.
[22] ---- MPI: A Message-Passing Interface Standard,The University of Tennessee,2000
[23] ---- MPICH.NT Version 1.2.0.4, 2000.