"HENRI COANDA"
AIR FORCE ACADEMY
ROMANIA

"GENERAL M.R. STEFANIK"
ARMED FORCES ACADEMY
SLOVAK REPUBLIC

INTERNATIONAL CONFERENCE of SCIENTIFIC PAPER
AFASES 2013
Brasov, 23-25 May 2013

PARALLEL ITERATIVE ALGORITHM FOR SOLVING POISSON'S EQUATION

Ioan POPOVICIU

Naval Academy "Mircea cel Batran", Constantza, Roumania

Abstract: *One of iterative methods for solving linear systems Ax=b is iterative SOR method (success Overrelaxation). The paper proposes a new algorithm of the SOR method for solving Poisson's equation based on partitioning the domain, resulting a parallel algorithm for solving a large linear system, algorithm with superior performances serial SOR method.*

Keywords: *parallel, iterative, mesh, processor*

1. INTRODUCTION

Let the system of linear equations

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

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

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

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

characterized the relations:

$$x^{(0)}(y,b) = y;$$

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

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

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

$$\phi(x,b) = Mx + Nb \qquad (1.3)$$

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

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

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

Theorem 1.1. [5] 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

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

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

$$i = 1,2,...,n$$

where:

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

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

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

3. PARALLEL 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 equation, 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: $4u_{ij} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1} = h^2 f_{ij}$ în Ω_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:

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

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

$$u(i,j)^{(k+1)} = (1-\omega)u(i,j)^{(k)} + \omega\begin{bmatrix} u(i-1,j)^{(k)} + u(i+1,j)^{(k)} + u(i,j-1)^{(k)} \\ + u(i,j+1)^{(k)} + h^2 f_{ij} \end{bmatrix}\Big/4$$

.

Then iterations in black points by:

$$u(i,j)^{(k+1)} = (1-\omega)u(i,j)^{(k+1)} + \omega\begin{bmatrix} u(i-1,j)^{(k+1)} + u(i+1,j)^{(k+1)} \\ + u(i,j-1)^{(k+1)} + u(i,j+1)^{(k+1)} + h^2 f_{ij} \end{bmatrix}\Big/4$$

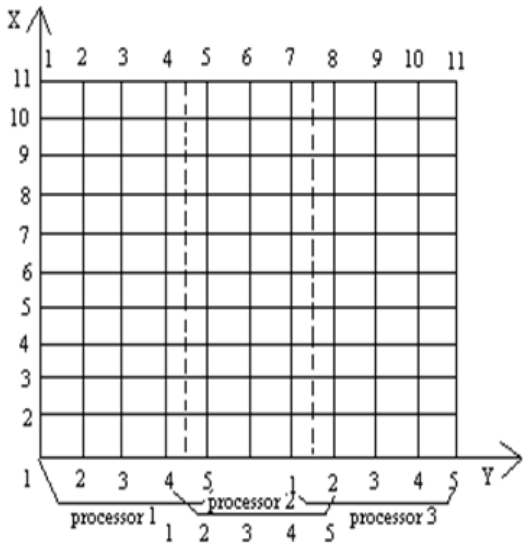
To implement the parallel method on p processors, the mesh Ω_h is decomposed into p disjoint subgrid meshes $\Omega_{h,\nu}$ such that $\Omega_h = \bigcup_{\nu=1}^{p} \Omega_{h,\nu}$.

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

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

pe $\Omega^1_{h,\nu}$

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

pe $\Omega^2_{h,\nu}$

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

pe $\Omega^3_{h,\nu}$

where $\nu = 1,2,\mathsf{K}\,, p$ 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. NUMERICAL EXAMPLE

Applying the parallel algorithm 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
4	6E-3	216	2.7

5. CONCLUSION

In the parallel algorithm 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 - black.* 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. Golub G, Van Loan. *Matrix computations*. Notrh Oxford Academic, Oxford 1983.

5. Hackbusch W, Trotienberg. *Multi-grid methods.* Bonn. 1990.

6. Ichim I, Marinescu G. *Metode de aproximare numerică*. Ed. Acad. Bucureşti 1986.

7. Kuznetsev A. *Algebraic multigrid domain decomposition methods*. Anal. and Math. 1989.

8. ---- *An Introduction to MPI for C Programmers*, The University of Texax at Austin, 1999.

9. ---- *A User's Guide to MPI*, Departament of Mathematics, University of San Francisco, 1998.

10. ---- *MPI: A Message-Passing Interface Standard,*The University of Tennessee,2000

[23] ---- MPICH.NT Version 1.2.0.4, 2000.