Joint International Topical Meeting on Mathematics & Computation and Supercomputing in Nuclear Applications (M&C + SNA 2007)
Monterey, California, April 15-19, 2007, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2007)

SOME EXPERIMENTS USING PRECONDITIONING TECHNIQUES AND SECOND-ORDER ITERATIVE METHODS FOR THE RESOLUTION OF THE 3D TRANSIENT NEUTRON DIFFUSION EQUATION

[†‡]Flores O. and [‡]Vidal V.
Department of Information Systems and Computing (DSIC)
[‡] Universidad Politécnica de Valencia, Camino de Vera S/N, 46021 Valencia
[†‡]Instituto Tecnológico de Tuxtepec, Calz. Dr. Víctor Bravo Ahuja S/N, Tuxtepec, Oaxaca, México.
{oflores, vvidal}@dsic.upv.es

Verdú G. and Miró R.
Department of Chemistry and Nuclear Engineering
Universidad politécnica de Valencia, Camino de Vera S/N, 46021, Valencia
{gverdu, rmiro}@iqn.upv.es

ABSTRACT

We present a numerical study about the application of two versions of a second-degree iterative method for the solution of the sparse linear systems arising in the discretisation of the 3D multi-group time-dependent Neutron Diffusion Equation. In addition, we propose some modifications to them, as well as a study of well-known preconditioning techniques in order to improve their convergence and accuracy when they are applied to a sequence of solutions in time of a real nuclear core transient. This is important for studies of stability and security of nuclear reactors.

Key Words: Second-Order Iterative Methods, Transient Neutron Diffusion Equation, Large ODE Systems, Preconditioning Techniques, Parallel Computing.

1. INTRODUCTION

Traditionally, direct methods [1][2] have been used for solving linear systems of equations due to their robustness and predictable behaviour. However iterative methods [3] have shown a good competency when they are combined with preconditioning techniques and Krylov subspace iterations, giving rise to efficient and simple general purpose procedures.

For design and safety reasons, nuclear power plants need fast and accurate plant simulators. The centre point of concern in the simulation of a nuclear power plant is the reactor core. Since it is the source of the energy that is produced in the reactor, a very accurate model of its constituent processes is needed. The neutron population into the reactor core is modeled using the Boltzmann transport equation. This three-dimensional problem is modeled by a complicated system of coupled partial differential equations, the multigroup neutron diffusion equation [4][5] that have been discretised using a nodal collocation method in space and one-step Backward-Difference Method in time. The solution of these equations can involve very intensive computing. Therefore, it is necessary to find effective algorithms for the solution of the three-dimensional model. Also, the progress in the area of multiprocessor technology suggests the application of High-Performance Computing to this problem.

Bru et al in [6] apply two Second-Degree methods [7] to solve the linear system of equations related to a 2D Neutron-Diffusion equation case. Thus, the main goal of this paper is the application of those methods and some modifications that we have proposed to decrease the computational work, but applied to a 3D real test case.

The outline of the paper is as follows. The mathematical model of the Time-dependent Neutron Diffusion Equation and its discretisation are described in Section 2. Section 3 describes the hardware and software platform used. The test case is presented in Section 4. The second-degree iterative methods are introduced at Section 5. The numerical results are presented in Section 6. Finally, we will draw some conclusions in Section 7.

2. PROBLEM DESCRIPTION

Plant simulators mainly consist of two different modules which account for the basic physical phenomena taking place in the plant: a neutronic module which simulates the neutron balance in the reactor core, and the evaporation and condensation processes. In this paper, we will focus on the neutronic module. The balance of neutrons in the reactor core can be approximately modelled by the time-dependent two energy group neutron diffusion equation, which is written using standard matrix notation as follows [4]:

$$[v^{-1}]\dot{\phi} + \mathcal{L}\phi = (1-\beta)\mathcal{M}\phi + \chi\sum_{k=1}^{K}\lambda_k\mathcal{C}_k \tag{1}$$

$$\dot{\mathcal{C}_k} = \beta_k[\nu\Sigma_{f_1}\nu\Sigma_{f_2}]\phi - \lambda_k\mathcal{C}_k,\ \ k = 1,\ldots,K \tag{2}$$

where

$$\mathcal{L} = \begin{bmatrix} -\vec{\nabla}\cdot(D_1\vec{\nabla}) + \sum_{a1} + \sum_{12} & 0 \\ -\sum_{12} & -\vec{\nabla}\cdot(D_2\vec{\nabla}) + \sum_{a2} \end{bmatrix}, [v^{-1}] = \begin{bmatrix} \frac{1}{v_1} & 0 \\ 0 & \frac{1}{v_2} \end{bmatrix},$$

and

$$\mathcal{M} = \begin{bmatrix} \nu\Sigma_{f1} & \nu\Sigma_{f2} \\ 0 & 0 \end{bmatrix}, \phi = \begin{bmatrix} \phi_f \\ \phi_t \end{bmatrix}, \chi = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

where

- ϕ is the neutron flux on each point of the reactor; so, it is a function of time and position.
- $\mathcal{C}_k$ is the concentration of the k-th neutron precursor on each point of the reactor (it is as well a function of time and position). $\lambda_k\mathcal{C}_k$ is the decay rate of the k-th neutron precursor.
- K is the number of neutron precursors. β_k is the proportion of fission neutrons given by transformation of the k-th neutron precursor; $\beta = \sum_{k=1}^{K}\beta_k$.
- $\mathcal{L}$ models the diffusion ($-\vec{\nabla}\cdot(D_1\vec{\nabla})$), absorption ($\sum_{a1}$, $\sum_{a2}$) and transfer from fast group to thermal group ($\sum_{12}$).
- $\mathcal{M}$ models the generation of neutrons by fission.
- $\nu\sum_{fg}$ gives the amount of neutrons obtained by fission in group g.
- v^{-1} gives the time constants of each group.

To study rapid transients of neutronic power and other space and time phenomena related to neutron flux variations, fast codes for solving these equations are needed. The first step to obtain a numerical solution of these equations consists of choosing a spatial discretization for Eq. (1). For this , the reactor is divided in cells or nodes and a nodal collocation method is applied [8][9]. In this collocation method, neutron flux is expressed as a series of Legendre Polynomials.

After a relatively standard process (setting boundary conditions, making use of the orthonormality conditions, using continuity conditions between cells) we obtain the following systems of ordinary differential equations:

$$[v^{-1}]\dot{\psi} + L\psi = (1-\beta)M\psi + X\sum_{k=1}^{K}\lambda_k C_k, \tag{3}$$

$$\dot{C}_k = \beta_k[M_{11}M_{12}]\psi - \lambda_k C_k, \qquad k = 1,\ldots,K, \tag{4}$$

where unknowns ψ and C_k are vectors whose components are the Legendre coefficients of ϕ and $\mathcal{C}_k$ in each cell, and L, M, $[v^{-1}]$ are matrices with the following block structure:

$$L = \begin{bmatrix} L_{11} & 0 \\ -L_{21} & L_{22} \end{bmatrix}, M = \begin{bmatrix} M_{11} & M_{12} \\ 0 & 0 \end{bmatrix}, v^{-1} = \begin{bmatrix} v^{-1} & 0 \\ 0 & v^{-1} \end{bmatrix}, X = \begin{bmatrix} I \\ 0 \end{bmatrix}.$$

Depending on flux continuity conditions imposed among the discretisation cells of the nuclear reactor, the blocks L_{11} and L_{22} can be symmetric or not, while blocks L_{21}, M_{11} and M_{12} are diagonal.

The next step consists of integrating the above ordinary differential equations over a series of time interval, $[t_n, t_{n+1}]$. Eq. (4) is integrated under the assumption that the term $[M_{11}M_{12}]\psi$ varies linearly from t_n to t_{n+1}, obtaining the solution C_k at t_{n+1} expressed as

$$C_k^{n+1} = C_k^n e^{\lambda_k h} + \beta_k(a_k[M_{11}M_{12}]^n\psi^n + b_k[M_{11}M_{12}]^{n+1}\psi^{n+1} \tag{5}$$

where $h = t_{n+1} - t_n$ is a fixed time step size, and the coefficients a_k and b_k are given by

$$a_k = \frac{(1+\lambda_k h)(1-e^{\lambda_k h})}{\lambda_k^2 h} - \frac{1}{\lambda_k}, b_k = \frac{\lambda_k h - 1 + e^{\lambda_k h}}{\lambda_k^2 h}.$$

To integrate Eq. (3), we must take into account that it constitutes a system of stiff differential equations, mainly due to the elements of the diagonal matrix $[v^{-1}]$. Hence, for its integration, it is convenient to use an implicit backward difference formula (BDF). A stable one-step BDF to integrate Eq. (3) is given by

$$\frac{[v^{-1}]}{h}(\psi^{n+1} - \psi^n) + L^{n+1}\psi^{n+1} = (1-\beta)M^{n+1}\psi^{n+1} + X\sum_{k=1}^{K}\lambda_k C_k^{n+1} \tag{6}$$

Taking into account Eq. (5) and the structure of matrices L and M, we rewrite Eq. (6) as the system of linear equations

$$\begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix}\begin{bmatrix} \psi_1^{n+1} \\ \psi_2^{n+1} \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix}\begin{bmatrix} \psi_1^{n} \\ \psi_2^{n} \end{bmatrix} + \sum_{k=1}^{K}\lambda_k e^{-\lambda_k h}\begin{bmatrix} C_k^n \\ 0 \end{bmatrix}, \tag{7}$$

where

$$
\begin{aligned}
T_{11} &= \frac{1}{h}v_1^{-1} + L_{11}^{n+1} - (1-\beta)M_{11}^{n+1} - \sum_{k=1}^{K}\lambda_k\beta_k b_k M_{11}^{n+1}, \\
T_{21} &= -L_{21}^{n+1}, \\
T_{12} &= -(1-\beta)M_{12}^{n+1} - \sum_{k=1}^{K}\lambda_k\beta_k b_k M_{12}^{n+1}, \\
T_{22} &= \frac{1}{h}v_2^{-1} + L_{22}^{n+1}, \\
R_{11} &= \frac{1}{h}v_1^{-1} + \sum_{k=1}^{K}\lambda_k\beta_k a_k M_{11}^{n}, \\
R_{12} &= \sum_{k=1}^{K}\lambda_k\beta_k a_k M_{12}^{n}, \\
R_{22} &= \frac{1}{h}v_2^{-1}.
\end{aligned}
$$

Thus, for each time step it is necessary to solve a large and sparse system of linear equations, with the following block structure:

$$
\begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix} = \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} \tag{8}
$$

where the right-hand side depends on both the solution in previous time steps and the backward difference method used. Usually, the coefficients matrix of system represented by Eq. (8) has similar properties as the matrices L and M in Eq. (3), namely blocks T_{11}, T_{22} are very-large sparse matrices with symmetric structure, and blocks T_{12}, T_{21} are singular diagonal matrices.

In [10], for example, (8) is solved using a GMRES method combined with a SOR preconditioner.

The transient study with more than two energy groups makes difficult to solve the respective system of equations. For this reason, it would be appropiated to use iterative methods that take advantage of the block properties that form the system (8), such as *second-order iterative methods* [11][12].

3. HARDWARE AND SOFTWARE PLATFORM

Sequential and parallel experiments have been performed on a 12-node biprocessor cluster of the Grupo de Redes y Computación de Altas Prestaciones (GRyCAP) at the Polytechnic University of Valencia. Each CPU is a 2 GHz Intel Xeon processor and has 1 GB of RAM memory. All nodes are connected by a SCI network with a Torus 2D topology in a 4x5 mesh with Red Hat 8.0 operating system.

The Portable, Extensible Toolkit for Scientific Computation (PETSc) [13][14],is a suite of data structures and routines for the implementation of large-scale application codes on parallel (and serial) computers.

Among the most popular Krylov subspace iterative methods contained in PETSc are *Conjugate Gradient, Bi-Conjugate Gradient, Stabilized BCG, Transpose Free Quasi-Minimal Residual, Generalized-Minimal Residual* and so on [3]. PETSc offers preconditioners as *Additive Schwarz, Block Jacobi, Jacobi, ILU, ICC*, etc.

In order to quantify the performance obtained for each method and to carry out a comparison among them, the sequential and parallel execution times are presented in *relative units of time*.

4. TEST CASE

The test case chosen is the commercial reactor of Leibstadt [15], which has been discretised in a 3D form. The spatial discretisation has 32*32*27 cells, so that the total number of equations and cells is quite large: 157248 equations and 796080 nonzero elements in the Jacobian matrix (see Figure 1). Due to the continuity conditions imposed on the flux between cells, the blocks T_{11} and T_{22} are symmetric positive-definite matrices.

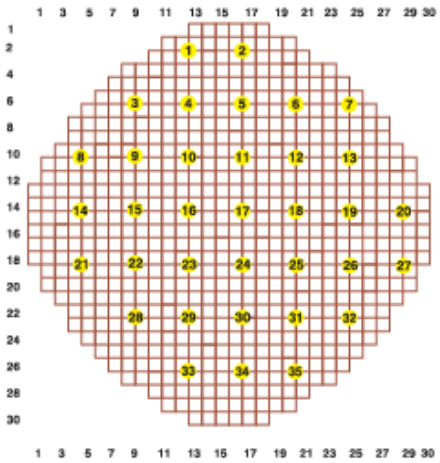

Figure 1. Axial Section of Reactor Leibstadt

5. SECOND-DEGREE ITERATIVE METHODS

This section begins with a brief introduction to the second-degree methods applied to a 2D neutron diffusion equation case in [6] (A and B methods). Also, we propose two variations to these methods in order to decrease their computational work (named C and D methods).

5.1 Method A

Consider the coefficient matrix T of the linear system (8) and the Jacobi splitting, $T = M - N$, with matrices M and N given by

$$M = \begin{bmatrix} T_{11} & 0 \\ 0 & T_{22} \end{bmatrix}, N = \begin{bmatrix} 0 & -T_{12} \\ -T_{21} & 0 \end{bmatrix},$$

where iteration matrix B_J is represented by

$$B_J = M^{-1}N = \begin{bmatrix} 0 & -T_{11}^{-1}T_{12} \\ -T_{22}^{-1}T_{21} & 0 \end{bmatrix}.$$

Now, considering the matrices $G_1 = \omega B_J$, $G_0 = (1-\omega)B_J$, where ω is an extrapolation factor, and the vector $k = M^{-1}e$, we can write the following second-degree method based on the Jacobi Over-relaxation (JOR) splitting

$$\psi^{(n+1)} = G_1\psi^{(n)} + G_0\psi^{(n-1)} + k = B_J(\omega\psi^{(n)} + (1-\omega)\psi^{(n-1)}) + k,$$

which corresponds to the following operations

$$\begin{aligned} T_{11}\psi_1^{l+1} &= e_1 - T_{12}(\omega\psi_2^l + (1-\omega)\psi_2^{l-1}), \\ T_{22}\psi_2^{l+1} &= e_2 - T_{21}(\omega\psi_1^l + (1-\omega)\psi_1^{l-1}). \end{aligned} \tag{9}$$

Let us identify these operations as Method A.

5.2 Method B

In the same manner we can construct another method based on the accelerated Gauss-Seidel splitting, whose iteration matrix B_{GS} is given by

$$B_{GS} = M^{-1}N = \begin{bmatrix} T_{11} & 0 \\ T_{21} & T_{22} \end{bmatrix}^{-1} \begin{bmatrix} 0 & -T_{12} \\ 0 & 0 \end{bmatrix}.$$

The operations that correspond to this method, method B, are represented by

$$\begin{aligned} T_{11}\psi_1^{l+1} &= e_1 - T_{12}(\omega\psi_2^l + (1-\omega)\psi_2^{l-1}), \\ T_{22}\psi_2^{l+1} &= e_2 - T_{21}(\omega\psi_1^{l+1} + (1-\omega)\psi_1^l). \end{aligned} \tag{10}$$

Methods A and B, can be described by the following algorithm:

Algorithm 1

(1) Set ψ_2^0*;* $\quad \{\psi_2^0 := \psi_2^*\}$
(2) Solve $T_{11}\psi_1^1 = e_1 - T_{12}\psi_2^0$
(3) Solve $T_{22}\psi_2^1 = e_2 - T_{21}\psi_1^1$
(4) Do $l = 1, 2, 3, \ldots$
(4a) Solve for ψ_1^{l+1} *with* T_{11} *in accordance with A or B method*
(4b) Solve for ψ_2^{l+1} *with* T_{22} *in accordance with A or B method*
until $\|\psi_1^{l+1} - \psi_1^l\| < tol$ *and* $\|\psi_2^{l+1} - \psi_2^l\| < tol$

where ψ_2^* represents the solution of a previous time step.

As we can see, the main difference between method A (9) and method B (10), is that in (10), the new solution for ψ_1 is used as soon as it is available to compute ψ_2. Therefore, a fast convergence rate may be

expected. In both methods, we distinguish between *outer* and *inner* iterations. Outer iterations are identified by Step (4), and inner iterations are represented by Steps (4a) and (4b), which correspond to the process solution of the linear systems T_{11} and T_{22} respectively. The solution of the linear systems is done with a Krylov Subspace Iterative Method combined with a preconditioning technique.

5.3 Method C

As we can see, the solution process with T_{11} and T_{22} blocks can be carried out independently each one of other. This fact has motivated experiments with the following operations

$$\begin{aligned} T_{11}\psi_1^{l+1} &= e_1 - T_{12}(\omega_1\psi_2^l + (1-\omega_1)\psi_2^{l-1}) \\ T_{22}\psi_2^{l+1} &= e_2 - T_{21}(\omega_2\psi_1^{l+1} + (1-\omega_2)\psi_1^l). \end{aligned} \tag{11}$$

In this new scheme, method C, we have added two different parameters ω_1 and ω_2 for each system to be solved, in order to accelerate its convergence.

5.4 Method D

Besides to method C, we have carried out experiments with an *'adaptable'* technique, achieving some improvements in the process efficiency. This technique computes the solution of the systems $T_{11}\psi_1^{l+1}$ and $T_{22}\psi_2^{l+1}$ with a cheap precision ϵ_{ρ_i} at initial stages of the method. Then, this precision is 'adapted' or 'improved' towards a more demanding one in successive iterations. The application of this technique to method C gives rise to the algorithm 2 (method D), where r_l means 'precision' r achieved in stage l.

General theorems about the convergence of second-degree methods can be found in [7].

Algorithm 2 (Adaptable version)

(1) Set ψ_2^0*;* $\quad \{\psi_2^0 := \psi_2^*\}$
(2) Set $\epsilon_\rho = \{\epsilon_{\rho_1}, \epsilon_{\rho_2}, \ldots, \epsilon_{\rho_n}\}$ *where* $\epsilon_{\rho_i} > \epsilon_{\rho_{i+1}}$*;*
(3) Solve $T_{11}\psi_1^1 = e_1 - T_{12}\psi_2^0$
(4) Solve $T_{22}\psi_2^1 = e_2 - T_{21}\psi_1^1$
(5) Do $l = 1, 2, 3 \ldots$
(5a) Solve for ψ_1^{l+1} *with tolerance* ϵ_{ρ_i}
(5b) Solve for ψ_2^{l+1} *with tolerance* ϵ_{ρ_i}
(5c) if $r_{l+1} \leq r_l$
$i := i + 1$
end if
until $\|\psi_1^{l+1} - \psi_1^l\| < tol$ *and* $\|\psi_2^{l+1} - \psi_2^l\| < tol$

6 NUMERICAL EXPERIMENTS

We have used a random vector as initial solution (ψ^*) for all methods and we have chosen the Conjugate Gradient method because the blocks M_{11} and M_{22} are symmetric positive-definite matrices [16]. In addition, we have required that the relative error be less than 10^{-7} as the criterion for convergence.

In order to identify the optimum ω and the precision achieved for each one of the second-degree iterative methods in our test case, we have carried out a heuristic study, which is described in [17] (see Table I).

Table I. Optimum ω for A-D Methods.

METHOD	ω **values**
A	1.0
B	1.3
C,D	$\omega_1 = 1.0, \omega_1 = 1.9$

6.1. Sequential Execution Times

Thus, when we execute the methods with and without preconditioning described before, we obtain the sequential execution times showed in Table II. In this table we can see that *without* preconditioning, the performance of C and D methods it is more efficient than performance registered by A and B methods. Also, making a comparison between C and D methods, the performance of method D is slightly more efficient than method C. Now, considering the performance between A and D methods, we can observe that method D is 90% faster than method A.

From Table II we can observe that the execution times *with* preconditioning present the same behaviour in performance with respect to the previous case. However, the execution time of method D with preconditioning obtains a benefit in performance of 37% with respect to the case without preconditioning. This fact shows that use of preconditioning techniques it has improved the performance substantially for our test case.

Table II. Sequential Execution Times (Relative Units) for Method A using different Preconditioners.

METHOD	**PRECONDITIONER**			
	NONE	SOR	JACOBI	BJACOBI
A	2797	2246	2002	**1698**
B	864	668	616	**525**
C	401	298	288	**235**
D	292	244	216	**185**

6.2. Parallel Execution Times

For the parallel case, we present a summary of the execution times for each method in Table III using different number of processors (see Section 3) and preconditioning techniques.

The analysis of Table III shows that the use of parallel computing (using more than one processor) help us to reduce the sequential execution times for all methods. For example, in the case of method A, its sequential time has been reduced to 11% when we use $p = 12$ processors, meanwhile the sequential time of method D has been reduced to 14% when we use the same number of processors.

Also, it is necessary to emphasize, as in the sequential case, the influence of preconditioning in the methods considered. For example, the execution time for method A with $p = 12$ processors without preconditioning has been reduced to 61% when we use Block-Jacobi preconditioning. In method D, with equal number of processors this reduction is 62% of the execution time obtained without preconditioning.

Table III. Parallel Execution Times (Relative Units) for Method A using different Preconditioners.

METHOD	**PC**	$p = 1$	$p = 2$	$p = 4$	$p = 6$	$p = 8$	$p = 10$	$p = 12$
A	NONE	2797	1450	753	413	410	346	**311**
	JACOBI	2002	985	525	297	294	243	**217**
	BJACOBI	1698	**764**	**459**	**326**	**264**	**219**	**191**
B	NONE	864	456	250	188	156	138	**124**
	JACOBI	616	332	268	135	111	98	**88**
	BJACOBI	525	**301**	**172**	**123**	**99**	**85**	**75**
C	NONE	401	210	163	86	71	64	**58**
	JACOBI	288	151	124	62	52	45	**41**
	BJACOBI	235	**140**	**108**	**56**	**45**	**39**	**34**
D	NONE	292	154	118	63	52	46	**42**
	JACOBI	216	111	63	46	38	33	**30**
	BJACOBI	185	**105**	**59**	**43**	**34**	**30**	**26**

In general, from Table III we can observe that method D combined with a Block-Jacobi preconditioner presents the best execution times for any number of processors (see Figure 2).

6.3. Speedup and Efficiency

The parallel performance metrics such as *speedup* and *efficiency* [18] of method D are represented in Table IV, where we can see that speedup and efficiency values are good enough for our test case.

For example, we have reduced the sequential time from 185 to 26 relative units of time when we use $p = 12$ processors, which represents a speedup of 7.12 and an efficiency of 69% (see Figure 3). Also, we can see that the efficiency is always greater than 60% for any number of processors.

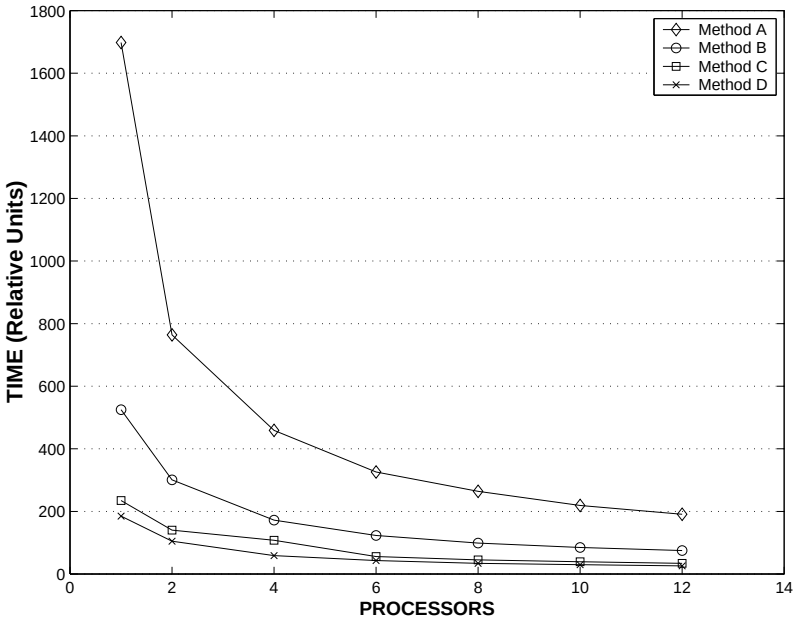

Figure 2. Parallel Execution Time using Block-Jacobi Preconditioner

Table IV. Parallel performance of method D combined with Block-Jacobi PC.

p	T_p	S_p	E_p
1	185	1.00	100%
2	105	1.76	88%
4	59	3.14	79%
6	43	4.30	72%
8	34	5.44	68%
10	30	6.17	62%
12	26	7.12	69%

7. CONCLUSIONS

We have presented the application of two second-degree iterative methods (A and B methods) to a sequence of solutions in time of a real nuclear core transient using the PETSc library. In addition, we have modified A and B methods in order to reduce their computational work. For this, we have implemented two versions: the first, named method C, based on the use of two different relaxation parameters (ω_1 and ω_2) for each energy group obtaining a great performance with respect to A and B methods. The second, that we have named method D, is based on method C but combined with an *adaptable* technique that improves even more the performance with regard to the others methods presented.

We have used parallel computing to decrease the sequential time for the methods. For this, we have coded the methods using the numerical parallel library of PETSc and we have achieved acceptable parallel performance for our test case. Also, these methods have been combined with well-known preconditioning techniques such Jacobi and Block Jacobi, which have helped to accelerate the rate of convergence of the

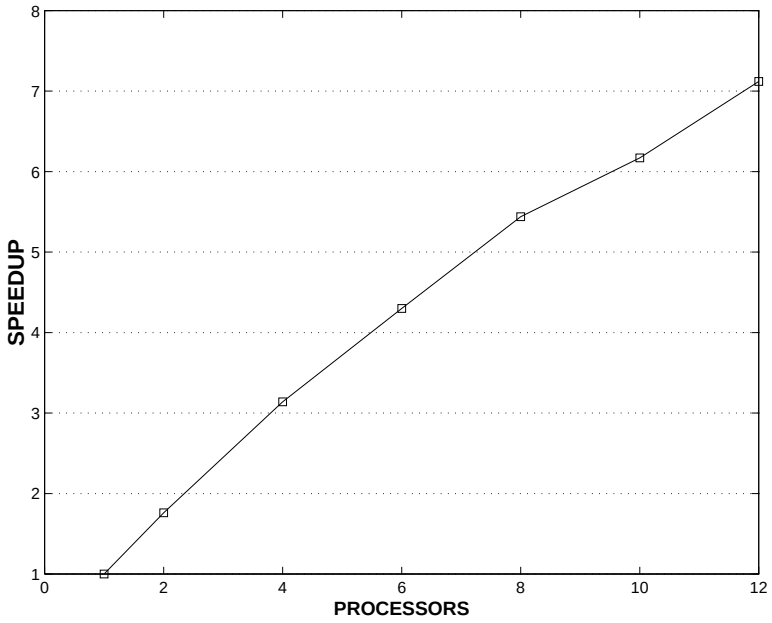

Figure 3. Speedup of method D

methods.

The main advantage of the second-degree methods presented, is that matrix T does not need be formed explicitly, because we require only the solution with T_{11} and T_{22}. Thus, simulation with more than 2 groups can be feasible in future and the use of these methods could represent a great saving in space requirements for reactor core computer simulation.

ACKNOWLEDGEMENTS

This work has been supported by Spanish MEC and FEDER under Grants ENE2005-09219-C02-02, ENE2005-09219-C02-01 and SEIT-SUPERA-ANUIES (México).

REFERENCES

[1] I.S. Duff et al., *Direct Methods for Sparse Matrices*, Clarendon Press, Oxford (1986)

[2] J.W. Demmel et al. *SuperLU Users Guide*, Technical Report ANL-44289, Lawrence Berkeley National Laboratory,(2003).

[3] Y. Saad, *Iterative Solution of Large Linear Systems*, Academic Press Inc., New York (1971).

[4] W. M. Stacey, *Space-Time Nuclear Reactor Kinetics*, Academic Press, New York (1970).

[5] A. F. Henry, *Nuclear Reactor Analysis*, The MIT Press, (1975).

[6] R. Bru, D. Ginestar, J. Marín, G. Verdú, J. Mas, T. Manteuffel, "Iterative Schemes for the Neutron Diffusion Equation," *Computers and Mathematics with Applications*, **Vol.44**, pp. 1307−1323 (2002)

[7] D.M. Young, *Iterative Solution of Large Linear Systems*, Academic Press Inc., New York (1971).

[8] G. Verdú, D. Ginestar, V. Vidal, and J.L. Muñoz-Cobo,"A consistent multidimensional nodal method for transient calculations," *Annals of Nuclear Energy*, **Vol. 22**, pp. 395410 (1995).

[9] D. Ginestar, G. Verdú, V. Vidal, R. Bru, J. Marín, and J.L. Muñoz-Cobo, "High order backward discretization of the neutron diffusion equation," *Annals of Nuclear Energy*, **Vol. 25**, pp. 4764 (1998).

[10] V.M. García, V.Vidal, G. Verdú, R. Miró,"Sequential and Parallel Resolution of the 3D Transient Neutron Diffusion Equation," *Mathematics, and Computation, Supercomputing, Reactor Physics and Nuclear and Biological Applications*, Francia, (2005).

[11] G. H. Golub, R.S. Varga, "Chebyshev semi-iterative methods, successive over-relaxation iterative methods, and second-order Richardson iterative methods," *Numerische Mathematik*, **Vol. 3**, pp. 147−168 (1961).

[12] G. H. Golub, C.F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, (1988).

[13] S. Balay, W. D. Gropp, L. C. McInnes, B.F. Smith,"PETSc Users Manual," *Report ANL-95/11 - Revision 2.1.5*, Argonne National Laboratory, (1997).

[14] S. Balay, W. D. Gropp, L. C. McInnes, B. F. Smith, "Efficient Management of Parallelism in Object Oriented Numerical Software Libraries, Modern Software Tools in Scientific Computing", E. Arge and A.M. Bruaset and H.P. Langtangen, 163-202, Nirkhauser Press, (1997)

[15] J. Blomstrand, "The KKL Core Stability Test. Conducted in September 1990," *ABB Report BR91-245* (1992)

[16] O. Flores, V. Vidal, "Free Distribution Parallel Sparse Solvers. Application to Lambda Modes Equation," *Proceedings of the IADIS International Conference Applied Computing 2006*, pp. 115−122 (2006)

[17] O. Flores, V. Vidal, V. García, P. Flores "Sequential and Parallel Resolution of the Two-Group Transient Neutron Diffusion Equation using second-degree Iterative Methods," *Proceedings of the VECPAR'06 7TH International Meeting on High-Performance Computing for Computational Science*, Vol. 4395, pp. 426−438 (2006)

[18] V. Kumar and A. Grama and A. Gupta and G. Karypis, *Introduction to parallel computing:design and analysis of parallel algorithms*, The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA (1994)