Iterative method for large linear systems

POPÎRLAN, Cristina
University of Craiova, Department of Computer Science
cpopirlan@gmail.com

Abstract

Iterative methods are used to solve large linear systems (analyze situations where quantitative data is unknown and qualitative data is known). Projection algorithms are particular cases of iterative methods, so they can be more efficient used to solve systems that require a large amount of calculations. The paper presents a parallel implementation of the modified (CQ) algorithm used to solve large linear systems. Some numerical tests are used to analyze the results improvement obtained with the modified (CQ) algorithm.

Keywords: *iterative methods, projection algorithms, parallel immplementation.*

ACM/AMS Classification: 65Y05, 68W10, 68W25.

1. Introduction

Consider a nonlinear maping $T : \mathcal{C} \to \mathcal{C}$, where $\mathcal{C}$ is a closed convex subset of a real Hilbert space $\mathcal{H}$. I denote by $Fix(T)$ the set of fixed points of T in $\mathcal{C}$, $Fix(T) \neq \emptyset$.

A general iterative process, which convergence was intensively studied ([2], [7], [8], [10]), is given by the following Mann iteration

$$x_{k+1} = (1 - t_k)x_k + t_k T(x_k)$$

where $\{t_k\}$ is the control sequence, $0 < t_k < 1$.

The projection algorithms are particular cases of the Mann iteration process, so the convergence properties of projection methods are obtained from the general convergence properties of the Mann iteration.

In order to obtain strong convergence, many algorithms were created [12], some of them impose restrictions on the system, others on the mapping T on on the set C. A complete study on projection algorithms was given by Bauschke and Borwein [4].

Projection algorithms derives from iterative methods; the first one was introduced in 1954 by Agmon, Motzkin and Schoenberg [1] which used a relaxation algorithm in order to solve a system of linear inequalities.

If $P_M(x)$ denote the projection of x onto M, then the classical projection method is

$$x_{k+1} = (1 - t_k)x_k + t_k P_M(x_k),$$

where t_k is the weight factor, $0 < t_k < 2$.

In 2003, Nakajo and Takahashi [13] introduced the (CQ) algorithm in order to obtain strong convergence of the constructed sequence:

$$\begin{cases} x_0 = x \in \mathcal{R}^n, \\ y_k = (1 - t_k)x_k + t_k T(x_k), \\ C_k = \{z \in \mathcal{R}^n : \ \|y_k - z\| \leq \|x_k - z\|\}, \\ Q_k = \{z \in \mathcal{R}^n : \ \langle x_k - z, x_0 - x_k\rangle \geq 0\}, \\ x_{k+1} = P_{C_k \cap Q_k}(x_0) \end{cases}$$

where $\{t_k\} \subset (a, 1]$ for some $a \in (0, 1)$.

The convergence (strong convergence) of this algorithm was studied by different authors [11] and was obtained for: T demicontractive mapping; I-T demiclosed at 0.

Numerous studies were made in order to simplify the method but to keep the strong convergence. The (CQ) algorithm was investigated in numerous papers and a generous number of generalizations were given (*modified methods*).

In a recent paper, Takahashi, Takevchi and Kubota [16] introduced a modified method for a family of nonexpansive operators, similar with the (CQ) algorithm. Starting with the initial iteration $x_0 \in C$, taking $C_1 = C$ and $x_1 = P_{C_1} x_0$, the algorithm is defined by:

$$\begin{cases} y_n = t_n x_n + (1 - t_n)Tx_n, \\ C_{n+1} = \{z \in C_n : ||y_n - z|| \leq ||x_n - z||\}, \\ x_{n+1} = P_{C_{n+1}} x_0, \end{cases}$$

were the control sequence $\{t_n\}_{n \geq 0}$ satisfy $0 \leq t_n \leq t < 1$.

In a previous paper [14], I implemented a parallel (CQ) algorithm used to solve linear large system. The results obtained from the numerical tests were not so good; a big number of iterations were calculated until the solution of the system was found.

In this paper, I present a parallel implementation of the modified (CQ) algorithm in order to obtain better results.

2. **Modified (CQ) algorithm parallelization**

In order to solve large linear systems, I efficiently partition the system into smaller systems that can be solved separately, distributed to a number of parallel processors [9]. The goal is obtain a balance so that the waiting time of the processors to be very small. Communication between processors is generally expensive, the processors may have different communication needs, thus achieving a balance in the parallel processing of data can be very difficult [15].

The ideal parallel system would be that each variable is calculated by a different processor, so the number of variables is equal to the number of processors.

In real parallel systems this is impossible. It should be taken into account that, most likely, solving large linear system is part of a much larger problem that also requires making other calculations to be solved.

In the considered implementation, the large linear system (the problem that need to be solved) can be partitioned and distributed to a number of parallel processing elements. An important discussion on the parallelization technics can be found in papers like ([5], [6]).

Running the application for different systems, we observe that the execution time of the program increases significantly if at a point one processor must wait another processor to complete the work, this happening at every step. If I run the application for a system without making a prior pre-conditioning operation, I observed that the running time is bigger than the situation when I used a pre-conditioned system [3].

In the parallel system implemented, all nodes execute the same operation, but working with different data. Finally, a concatenation processor handles all data from processors who worked at step k and evaluate the solution obtained by checking if the current step is a good solution to the linear system.

The modified (CQ) algorithm allows efficient parallelization because each step uses the original data and the values calculated in the previous step. The set of data stored in the shared memory at a time is not very high (increases as the system size increases). The modified (CQ) process that need to be solved at every step by each processor is given by the algorithm:

do{

$y_{p_k} = t_{p_k} x_{p_k} + (1 - t_{p_k} T(x_{p_k})$

In parallel, different processor calculate the set $C_{p_{k+1}}$

$C_{p_{k+1}} = \{z \in C_n : \ \|y_{p_k} - z\| \leq \|x_{p_k} - z\|\}$

$x_{p_{k+1}} = P_{C_{p_{k+1}}}(x_{p_0})$

Calculate the residual vector $\zeta_p = b_p - A_p x_{p_k}$

}while $(||\zeta_p|| \geq \epsilon)$

where we consider the following large linear system:

$$Ax = b$$

with A is a large real matrix ($A \in \mathcal{R}^{n \times n}$).

Parallelization of the modified (CQ) algorithm used to solve large linear systems involves decomposing of the large matrix A and The vector solution x_n into blocks so that partitions can be resolved individually, or wait a minimum number of data from other processors. Different processors calculate parts of the modified (CQ) algorithm in order to achieve a good parallelization of the method.

I used a parallel system with shared memory in order to achieve a better time, so the matrix A and vector are stored in the common memory because they must be easily accessible.

Table 1: Experimental results with (CQ) algorithm for a parallel system with 11 processors

Size	Iterations	Time/iteration	Iterations	Time/iteration
10	170	0.01	162	0.01
20	250	0.23	233	0.03
30	423	0.87	397	0.14
40	501	1.16	488	0.47
50	582	1.74	524	0.98
75	887	2.34	803	1.45
100	1131	2.85	876	2.15

The application requires two iterations vectors of the same size as the system stored in the memory. At each step the algorithm calculates the set C_k that is stored for a short time (until the next iteration is computed).

For the parallel implementation of the modified (CQ) algorithm I have developed an application that uses threads (each thread serves as a processor). A main thread check if the solution is calculated at the current step k is the desired solution for user application; other threads are used to calculate intermediate values of the algorithm for each block separately. After each secondary thread finishes its task, it writes in shared memory the calculated values so that the main thread uses them to check if the solution was obtained.

The application was run for variuos examples. Were considered various large linear systems and it was observed that the results are very good, in the sense that time is quite small if we consider the large number of unknowns of the system.

In the application that implements the (CQ) algorithm I maked some numerical tests [14]. The same test, the same large linear system, will be made for the modified (CQ) algorithm and the results will be listed in tabels in order to observe the improvement obtain with this application.

The experimental results obtained by the parallelized modified (CQ) algorithm are described in table 1 (the parallel system has 11 processors) and tabel 2 (the parallel system has 21 processors): first column represents number of unknowns (system size), the second column represents the number of iterations calculated with the (CQ) algorithm, the third column represents the execution time for one iteration of the algorithm for the (CQ) algorithm, the fourth column represents the number of iterations calculated with the modified (CQ) algorithm and the last column represents the time for one iteration of the modified (CQ) algorithm.

If the number of processors is increased to 21, table 2 shows that the execution time decreases significantly and the number of iterations calculated is reduced because each block will contain a smaller number of variables.

Evaluation of the numerical results obtained using the modified (CQ) algorithm implementation can be influenced by: system pre-conditioning; processors type; processors speed; communication network; system memery type

Table 2: Experimental results for a parallel system with 21 processors

Size	Iterations	Time/iteration	Iterations	Time/iteration
10	142	0.01	140	0.01
20	232	0.03	229	0.01
30	376	0.13	364	0.08
40	455	0.32	436	0.14
50	542	0.52	532	0.21
75	721	0.94	682	0.42
100	875	1.72	771	0.74

and accesibility.

Studying the results obtained we observe that the number of iterations performed to determine the solution of the system is influenced by the number of processors used because it uses a different system preconditioning. Number of processors used directly influences the total execution time, as can be seen in tables 1 and 2. In the considered examples the communication costs are considered to be 0.

3. **Conclusion**

This paper presents the results obtained with a parallel modified (CQ) algorithm that is used solve large linear systems. I analyzed the implementation of the parallel algorithm. The experiments were performed in order to study the algorithm application.

The implementations allow us to analize the differences between the modified (CQ) algorithm and (CQ) algorithm. A smaller amount of data are calculated at every iteration and so the total time is much smaller. We observe that there is also a great improvement in the total number of iterations calculated until a solution of the large linear system is achieved.

Numerical experiments were performed in order to study the algorithm implementation and applicability. The results obtained with the application were better then the ones obtained with the (CQ) algorithm.

As a future work, I intend to improve the performance of the algorithm and to apply the parallel approach to other projection algorithms in order to study the differences between them.

Acknowledgement

This paper has been financially supported within the project entitled Horizon 2020 - Doctoral and Postdoctoral Studies: Promoting the National Interest through Excellence, Competitiveness and Responsibility in the Field of Romanian Fundamental and Applied Economic Research, contract number POSDRU/159/1.5/S/140106. This project is co-financed by European Social Fund through Sectoral Operational Programme for Human Resources Development 2007-2013. Investing in people!

References

1. S. Agmon, *The relaxation method for linear inequalities*, "Canad. J. Math.", Vol. 6, 382-392, 1954.
2. O. Axelsson, *Iterative Solution Methods*, "Cambridge University Press, 1996.
3. R. Barrett, M. Berry, T.F. Chan, J. Demmel, J.M. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. Van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, "Philadalphia: Society for Industrial and Applied Mathematics", 1994.
4. H. Bauschke, J. Borwein, *On projection algorithms for solving convex feasibility problems*, "SIAM REVIEW", Vol. 38(3), 367-426, 1996.
5. G. Ciardo, *Distributed and structured analysis approaches to study large and complex systems*, "Lectures on formal methods and performance analysis", Vol. 2090, 344-374, 2001.
6. I.S. Duff, H.A. Van der Vorst, *Developments and trends in the parallel solution of linear systems*, "Parallel Computing", Vol. 25(1314), 1931-1970, 1999.
7. C.T. Kelley, *Iterative Methods for Optimization*, "Society for Industrial and Applied Mathematics", Philadelphia, 1999.
8. T.H. Kim, H.K. Xu, *Convergence of the modified manns iteration method for asymptotically strict pseudo-contractions*, "Nonlinear Analysis", Vol. 329, 336-346, 2007.
9. M. Kwiatkowska, R. Mehmood, *Out-of-Core Solution of Large Linear Systems of Equations arising from Stochastic Modelling*, "Proceedings PAPM-PROBMIV02", 135-151, 2002.
10. P.E. Mainge, *Regularized and inertial algorithms for common fixed points of nonlinear operators*, "Journal of Mathematical Analysis and Applications", Vol. 344, 876-887, 2008.
11. C. Martinez-Yanes, H.K. Xu, *Strong convergenceof the CQ method for fixed point iteration processes*, "Nonlinear Analysis", Vol. 64, 2400-2411, 2006.
12. H.S. Najafi, S.A. Edalatpanah, A.H. Refahi Sheikhani, *Convergence Analysis of Modified Iterative Methods to Solve Linear Systems*, "Mediterranean Journal of Mathematics", 2014.
13. K. Nakajo, W. Takahashi, *Strong convergence theorems for nonexpansive mappings and non-expansive semigroups*, "Journal of Mathematical Analysis and Applications", Vol. 279, 372-379, 2003.
14. C. Popîrlan, *Algoritm de proiecţie paralel pentru rezolvarea sistemelor liniare de dimensiuni mari*, "Impactul transformărilor socio-economice şi tehnologice la nivel naţional, european şi mondial", 2014.
15. C. Popîrlan, *Parallel projection algorithm*, "Proceedings of the 12th Conference on Artficial Intelligence and Digital Communications (AIDC 2012)", 62-68, 2012.
16. W. Takahashi, Y. Takeuchi, R. Kubota, *Strong convergence theorems by hybrid methods for families of nonexpansive mappings in hilbert spaces*, "Journal of Mathematical Analysis and Applications", Vol. 341, 276-286, 2008.