Comparison of steepest descent method and conjugate gradient method

Oleksandra Osadcha
Faculty of Applied Mathematics
Silesian University of Technology
Gliwice, Poland
Email:oleosa338@student.polsl.pl

Zbigniew Marszaek
Institute of Mathematics
Silesian University of Technology
Gliwice, Poland
Email: Zbigniew.Marszalek@polsl.pl

***Abstract*—This paper illustrates comparison of two methods, which are used for solving systems of linear equations . The aim of the research is to analyze, which method is faster solve this equations and how many iteration required each method for solving. This paper presents some ways to deal with this problem. Also there are analyzed two methods. The first of them minimizes the standard deviation of the next iteration of the solution, the other constructs orthogonal vectors and is theoretically finite method.**

***Index Terms*—conjugate gradient method, steepest descent method, comparison, analysis**

I. Introduction

Computer algorithms are important methods for numerical processing. In all implementations it is important to make them more efficient and decrease complexity but without loss of efficiency. Cryptographic algorithms are one of most important methods for computer science therefore efficiency and complexity of these is crucial for optimal processing [14]. Knowledge systems demand fast and precise methods for information processing [4], [6]. Similarly decision support models need devoted techniques for lower complexity [12].

This publication present comparison of steepest descent method and conjugate gradient method. These methods are used for solving systems of linear equations. In our publication, we analyze, which method is faster and how many iteration required each method. First, we describe these methods, than we compare them and make conclusions.

II. The steepest descent method

The steepest descent method formulated by Stiefel. We will present the mathematical description of the method of steepest descent and we will make implementation in the form of code. Consider the system linear equations in the general form

$$Ax = B \tag{1}$$

We suppose that matrix A is symmetric and positive definite. With x^* we denote solution of the system equations 1, i.e.: $Ax^* = b$. The methods of gradient constructed a sequence of successive approximations to determine the solutions x^* with using formula:

$$x^{(k+1)} = x^{(k)} + c_k r^{(k)} \tag{2}$$

Occurring in the equation 2 vector $r^{(k)}$ is called residuum of system and takes the following form:

$$r^{(k)} = b - Ax^{(k)}, \tag{3}$$

or

$$r^{(k+1)} = r^{(k)} - c_k A r^{(k)} = (I - c_k A) r^{(k)}. \tag{4}$$

In iterative process this vector can be determined in each iteration of the algorithm as a suitable combination of the previous values, as described is in next formula:

$$r^{(k+1)} = r^{(k)} - c_k A r^{(k)} \tag{5}$$

The coefficients c_k iterations occurring in the equation 3 are determined on the basis of the following equality:

$$\|x^* - x^{(k+1)}\|_B = \inf_{c_k} \|x^* - x^{(k)} - c_k r^{(k)}\|_B \tag{6}$$

In contrast, norm $\|\cdot\|_B$ is defined as

$$\|x\|_B \stackrel{def}{=} \sqrt{x^T B x} \tag{7}$$

Matrix B occurring in formula 7 is symmetric and positive definite. At the same time it fulfills the condition the alternation of the matrix A, i.e.: $A \cdot B = b \cdot A$. It is not difficult to show that searched coefficients c_k are described by the equation:

$$c_k = \frac{(r^{(k)}, B(x^* - x^{(k)}))}{(r^{(k)}, Br^{(k)})}. \tag{8}$$

Just for some matrix B, you can determine the value of $B(x^* - x^{(k)})$. For example, if $B = A^{(p)}$, $p \in \mathbb{N}$ then there is following equality:

$$A^{(p)}(x^* - x^{(k)}) = A^{(p-1)}(b - Ax^{(k)}) = A^{(p-1)} r^{(k)}. \tag{9}$$

Also it can be described in specific cases. For example, for $p = 1$, method is called steepest descent, while $p = 2$ method minimizes the Euclidean norm, and we call it the method of least residuum, where

$$c_k = \frac{(r^{(k)}, Ar^{(k)}))}{(Ar^{(k)}, Ar^{(k)})}. \tag{10}$$

III. Conjugate gradient method

We use conjugate gradient method to solve the system of linear equations given in the form of

$$Ax = b, \tag{11}$$

where A is a positive definite matrix with $n \times n$ sizes.As a result of operation of this method we obtain a sequence of vectors starting from a vector initial x_0

$$x_0 \to x_1 \to \cdots \to x_n,$$

which stops after accurate calculations at most n steps, so we get the required solution in the form of $x_{n+1} = x$. Since the calculations performed occurs rounding error value, so the value of x_{n+1} is generally not very accurate result. Therefore, in this method, as in all other iterative methods we conduct further steps $x_k \to x_{k+1}$, respectively, until we get the exact solution. Since the amount of work at each step of the workload of the matrix multiplication A by the vector, and therefore this method is not suitable for use in the matrix band and solid. In contrast, as much as possible, this method is suitable for a medium-sized array diluted. The general idea of our method is based on the fact that the functional $F : \mathbf{R}^n \to \mathbf{R}$ which has a form:

$$F(z) = \frac{1}{2}(b-Az)^T A^{-1}(b-Az) = \frac{1}{2}z^T b^T z - Az + \frac{1}{2}b^T A^{-1} b,$$

is minimized with accurate solution x, thus

$$0 = F(x) = \min_{z \in \mathbf{R}^n} F(z).$$

This equality follows from the fact that the matrix A is positive definite, and thus the matrix A^{-1} also has the same property. While corresponding with the vector z residualny vector r : $Az = -b$, disappears only $z := x$. It reminds us that the method of steepest descent in which a string of $x_1 \to x_2 \to \ldots$ is found with 1-dimensional minimize the functional F in the direction of the gradient:

$$x_{k+1} : F(x_{k+1}) = \min_u F(x_k + ur_k),$$

where

$$r_k := DF(x_k)^T = b - Ax_k.$$

However, in the case of conjugate gradient method, we must to carry out the minimization of the k dimensions:

$$x_{k+1} : F(x_{k+1}) = \min_{u_1,\ldots,u_k} F(x_k + u_1 r_1 + \cdots + u_k r_k),$$
$$r_i := b - Ax_i \; for \; i \le k. \tag{12}$$

It's very easy to compute x_{k+1}. In addition, vectors r_i are ortogonal, and hence lineary independent provided that $r_k \neq 0$. Becouse, in space $\mathbf{R}$, there are no more than n in depended vectors, so in exact calculations, there is the smallest $k \le n+1$ such that r_k is equal zero. Vector x_k corresponds to number r_k. Now we well show step k $x_k \to r_{k+1}$ of our method. The matrix dimensions $n \times k$ we denote R_k,

$$R_k := (r_1, r_2, \ldots, r_k).$$

With the k-dimensional minimilizing point in formula 11 are allowed all vectors from z being in the form of

$$z = x_k + R_k u, \quad u \in \mathbf{R}^k,$$

where $r = r_k + AR_k u$ is residual vector. Hence,

$$F(z) = \frac{1}{2}r^T A^{-1} r = \frac{1}{2}r_k^T A^{-1} r_k + u^T R_k^T r_k + \frac{1}{2}u^T R_k^T A R_k u.$$

Differentiating our functional by u, functional adopts a minimum for

$$x_{k+1} = x_k + R_k \widetilde{u},$$

where $\widetilde{u} \in \mathbf{R}^k$ is solution to the equation

$$R_k^T A R_k \widetilde{u} = -R_k^T r_k. \tag{13}$$

We introduce the following designations

$$R := (r_1, \ldots, r_k, \ldots, r_n), \quad v_k := \left.\begin{bmatrix} \widetilde{u} \\ 0 \\ \vdots \\ 0 \end{bmatrix}\right\}_{n-k} \in \mathbf{R}^n$$

Then we get $Rv_k = R_k \widetilde{u}$, and

$$x_{k+1} = x_k + R_k \widetilde{u} = x_k + Rv_k, \tag{14}$$

$$r_{k+1} = r_k + AR_k \widetilde{u} = r_k + ARv_k. \tag{15}$$

From the above fact follows that: 1. $R_k^T r_{k+1} = 0$ by (12), after that we have ortogonal vector r_{k+1} to vectors $r_1, \ldots, r_k$. Thus the right-hand side of equation (12) is as follows:

$$-R_k^T r_k = \left.\begin{bmatrix} 0 \\ \vdots \\ 0 \\ \\ * \end{bmatrix}\right\}^{k-1}, \quad so \quad R^T A R v_k = \left.\begin{bmatrix} 0 \\ \vdots \\ 0 \\ \\ * \\ \vdots \\ * \end{bmatrix}\right\}^{k-1}. \tag{16}$$

2. If vector $r_k \neq 0$, then vectors $r_1, \ldots, r_k$ are linearly independent, but matrix $R_k^T A R_k$ is positively determined. Thereafter, $R_k^T r_k \ge 0$ hence to the equation (11) and (14)

$$0 < \widetilde{u}^T R_k^T A R_k \widetilde{u} = -\widetilde{u}^T R_k^T r_k,$$

where last component in vector $\widetilde{u}$, and therefore k-th component of the vector v_k is negative.

3. In view of equations (13) and (15) for $i < k$ we have next equality:

$$(x_{i+1}-x_i)^T A(x_{k+1}-x_k) = v_i^T[R^T A R v_k] = \\ = (\underbrace{*\cdots *}_{i},0,\ldots,0)\begin{bmatrix}0\\ \vdots\\ 0\\ \\ *\\ \vdots\\ *\end{bmatrix}\Bigg\}^{k-1} = 0, \tag{17}$$

it is mean, that directions $p_i := x_{i+1} - x_i$ are A-coupled: $p_i^T A p_k = 0$ for $i < k$.

4. In view of equations (12), (1.1.13) and (1.1.14) we have next equality

$$F(x_{k+1}) = \frac{1}{2} r_{k+1}^T A^{-1} r_{k+1} = \frac{1}{2} r_k^T A^{-1} r_k + \widetilde{u}^T R_k^T r_k + \\ + \frac{1}{2}\widetilde{u}^T R_k^T A R_k \widetilde{u} = F(x_k) - \frac{1}{2}\widetilde{u}^T R_k^T A R_k \widetilde{u} = \\ = F(x_k) - \frac{1}{2}(x_{k+1}-x_k)^T A^{-1}(x_{k+1}-x_k).$$

This formula explains the extend of decrease value of functional F in k-th step $x_k \to x_{k+1}$

5. If vector $v_k = (v_1,\ldots,v_k,0,\ldots,0) r_k \neq 0$ has k components, where $r_k \neq 0$ is negative. From formula (14) follows next equality

$$r_{k+1} - r_k = A R v_k = \sum_{j=1}^{k} v_j A r_j,$$

then

$$A r_k = ((r_{k+1} - r_k) - \sum_{j=1}^{k-1} v_j A r_j)/v_k.$$

Applying induction we get description for Ar_k in following form

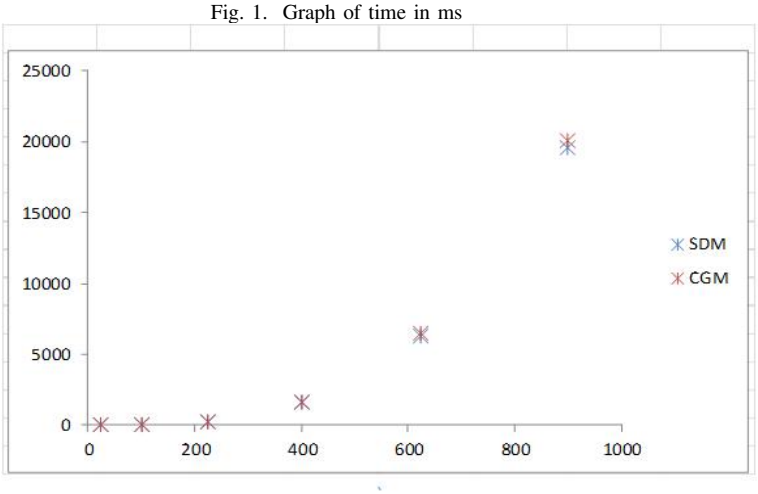
$$A r_k = q_{k+1}(r_k - r_{k+1}) - e_k(r_{k-1} - r_k) - \\ - \sum_{j=1}^{k-2} c_j (r_j - r_{j+1}), \quad (e_1 := 0) \tag{18}$$

where q_{k+1}, e_k are constans and additionally $q_{k+1} = -1/v_k > 0$. By solving the above equation due tor_{k+1} we get the equality:

$$r_{k+1} = r_k + [-Ar_k + e_k(r_k - r_{k-1}) + \sum_{j=1}^{k-2} c_j(r_{j+1} - r_j)]/q_{k+1}. \tag{19}$$

If we take into account orthogonality of vectors $r_1,\ldots,r_k,r_{k+1}$, then we can easily demonstrate that all

Fig. 1. Graph of time in ms



the constants c_j disappear. Thus, we obtain formulas for q_{k+1} i e_k. Then, for $k - 2 \geq 1$ we get

$$r_1^T r_{k+1} = 0 \Rightarrow 0 = r_1^T A r_k + c_1 r_1^T r_1 = r_k^T (A r_1) + c_1 r_1^T r_1. \tag{20}$$

By using the formula (17), we can conclude that vector Ar_1 also is a linear combination of vectors r_1 and r_2, then $r_k^T A r_1 = 0$ providing that $k > 2$. Thus, from equality (18) implies that $c_1 = 0$.

We assume, that for some $j \geq 1$, which satisfying the following inequality $1 \leq j \leq k-2$, there is equality $c_1 = c_2 = \cdots = c_{j-1} = 0$. Therefore, by using formula(18) as before, we obtain:

$r_j^T r_{k+1} = 0 \Rightarrow 0 = r_k^T A r_j + c_j r_j^T r_j =$

$= r_k^T$(linear combination of vectors $r_{j+1}, r_j, \ldots, r_1$) + $c_j r_j^T r_j =$

$c_j r_j^T r_j \Rightarrow c_j = 0$, vectors $r_1,\ldots,r_k$ are linearly independent, as is apparent from the fact that $r_0 \neq 0$. In this connection, in particular, we have $r_j \neq 0$. Therefore, all solid c_j, where $j = 1,\ldots,k-2$ disappear in equations (17) and (18). Due to the fact, we get

$$r_{k-1}^T r_{k+1} = 0 \Rightarrow e_k = -\frac{r_k^T A r_{k-1}}{r_{k-1}^T r_{k-1}} = q_k \frac{r_k^T r_k}{r_{k-1}^T r_{k-1}}$$

$$r_k^T r_{k+1} = 0 \Rightarrow q_{k+1} = \frac{r_k^T A r_k}{r_k^T r_k} - e_k$$

This gives us an algorithm conjugate gradient method.

Initial data: x_1 is intended, $r_1 := b - Ax_1, e_1 := 0$.

For $k = 1,2,\ldots,n$:]1) If $r_k = 0$, stop; x_k is sought for solution. 2) Otherwise, we must to calculate:

$$q_{k+1} := \frac{r_k^T A r_k}{r_k^T r_k} - e_k,$$

Tab. I
COMPARISON OF METHODS

		Conjugate gradient method			Steepest descent method		
N	n	Iteration	Time in ms	Time in ti	Iteration	Time in ms	Time in ti
5	25	13	1	1983	128	0	1780
10	100	33	36	68281	405	34	64634
15	225	49	286	530934	837	272	505238
20	400	65	1680	3110493	1413	1621	3002271
25	625	79	6499	12032438	2093	6329	11716167
30	900	94	20053	37122809	2909	19606	36295497

$$r_{k+1} := r_k + (-Ar_k + e_k(r_k - r_{k-1}))/q_{k+1},$$

$$x_{k+1} := x_k + (-r_k + e_k(x_k - x_{k-1}))/q_{k+1},$$

$$e_{k+1} := q_{k+1}\frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}.$$

IV. THE COMPARISON OF METHODS

A. Analysis of efficiency

There is a big difference between presented methods. The steepest descent method is several times faster than conjugate gradient method. But, if we look on iteration of this methods in Table.1, we see, that conjugate method converge after less number of iteration. For example, for $n = 20$, number of iterations of conjugate gradient method equals 65, and achieve the desired accuracy - 1413.

B. Analysis of time

On graph of time, we can see that, for example, for $n = 15$ time in ms of conjugate gradient method equals 286 and time in ti of steepst descent method equals 271. After that, we could see, that when number of n increases there is substantial difference between methods. For example, when $n = 20$, time in ms of conjugate gradient method is equals 1680, time in ti of achieve the desired accuracy - 1621 iterations.

V. CONCLUSIONS

We have compared two method implemented to solve systems of linear equations. The steepest descent method is faster method, because it solve equations in less amount of time. Conjugate gradient method is slower, but more productive, because, it converges after less iterations. So, we can see, that one method can be used, when we want to find solution very fast and another can be converge to maximum in less iteration.

REFERENCES

[1] O. Axelsson, G. Lindskog *On the Rate of Convergence of the Preconditioned Conjugate Gradient Method*, „Numerische Mathematik", 1986, nr 48, s. 499-523

[2] Mordecai Avriel *Nonlinear Programming: Analysis and Methods*, 2003, Dover Publishing, ISBN 0-486-43227-0

[3] D. Połap, M. Woźniak, , C. Napoli, and E. Tramontana, "Real-Time Cloud-based Game Management System via Cuckoo Search Algorithm" *International Journal of Electronics and Telecommunications*, vol. 61, No. 4, pp. 333–338, 2015, DOI: 10.1515/eletel-2015-0043.

[4] P. Artiemjew, Stability of Optimal Parameters for Classifier Based on Simple Granules of Knowledge, *Technical Sciences*, vol. 14, no. 1, pp. 57-69. UWM Publisher, Olsztyn 2011.

[5] Knyazev, Andrew V.; Lashuk, Ilya *Steepest Descent and Conjugate Gradient Methods with Variable Preconditioning*. SIAM Journal on Matrix Analysis and Applications. 29 (4): 1267. doi:10.1137/060675290

[6] Z. Marszałek, "Novel Recursive Fast Sort Algorithm," in *Communications in Computer and Information Science*, vol. 639, pp. 344–355, 2016, DOI: 10.1007/978-3-319-46254-7_27.

[7] G. Capizzi, F. Bonanno, C. Napoli, "Hybrid neural networks architectures for SOC and voltage prediction of new generation batteries storage", *International Conference on Clean Electrical Power (ICCEP)*, pp. 341-344, IEEE, 2015, DOI: 10.1109/ICCEP.2011.6036301.

[8] C. Napoli, F. Bonanno, G. Capizzi, "An hybrid neuro-wavelet approach for long-term prediction of solar wind", *Proceedings of the International Astronomical Union*, vol. 6, no. 274, pp. 153-155, IAU, 2015, DOI: 10.1017/S174392131100679X.

[9] Yvan Notay *Flexible Conjugate Gradients*, 2000, SIAM Journal on Scientific Computing. 22 (4): 1444. doi:10.1137/S1064827599362314

[10] D. Połap, M. Woźniak, , C. Napoli, and E. Tramontana, "Is swarm intelligence able to create mazes?" *International Journal of Electronics and Telecommunications*, vol. 61, No. 4, pp. 305–310, 2015, DOI: 10.1515/eletel-2015-0039.

[11] C. Napoli, G. Pappalardo, E. Tramontana, "An agent-driven semantical identifier using radial basis neural networks and reinforcement learning", *XV Workshop Dagli Oggetti agli Agenti (WOA)*, CEUR-WS, vol. 1260, 2015, DOI: 10.13140/2.1.1446.7843.

[12] D. Połap, M. Woźniak, "Flexible Neural Network Architecture for Handwritten Signatures Recognition" *International Journal of Electronics and Telecommunications*, vol. 62, no. 2, pp. 197–202, 2016, DOI: 10.1515/eletel-2016-0027.

[13] Jan A. Snymam *Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*, 2005, Springer Publishing, ISBN 0-387-24348-8

[14] J. Toldinas, R. Damasevicius, A. Venckauskas, T. Blazauskas, and J. Ceponis, "Energy Consumption of Cryptographic Algorithms in Mobile Devices" *ELEKTRONIKA IR ELEKTROTECHNIKA*, vol. 20, pp. 158–161, 2014.

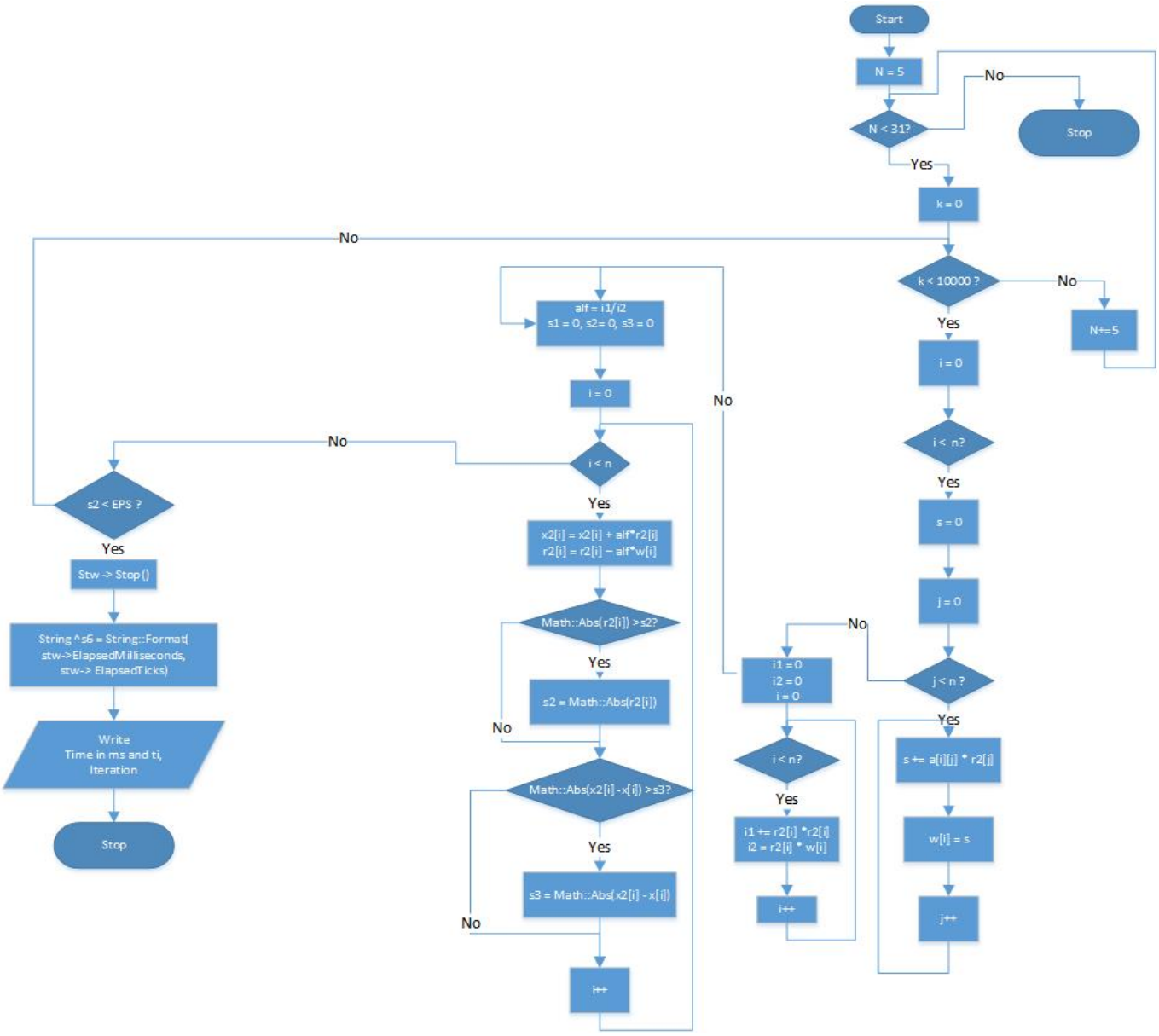


Fig. 2. Block Diagram of the Steepest Descent Method

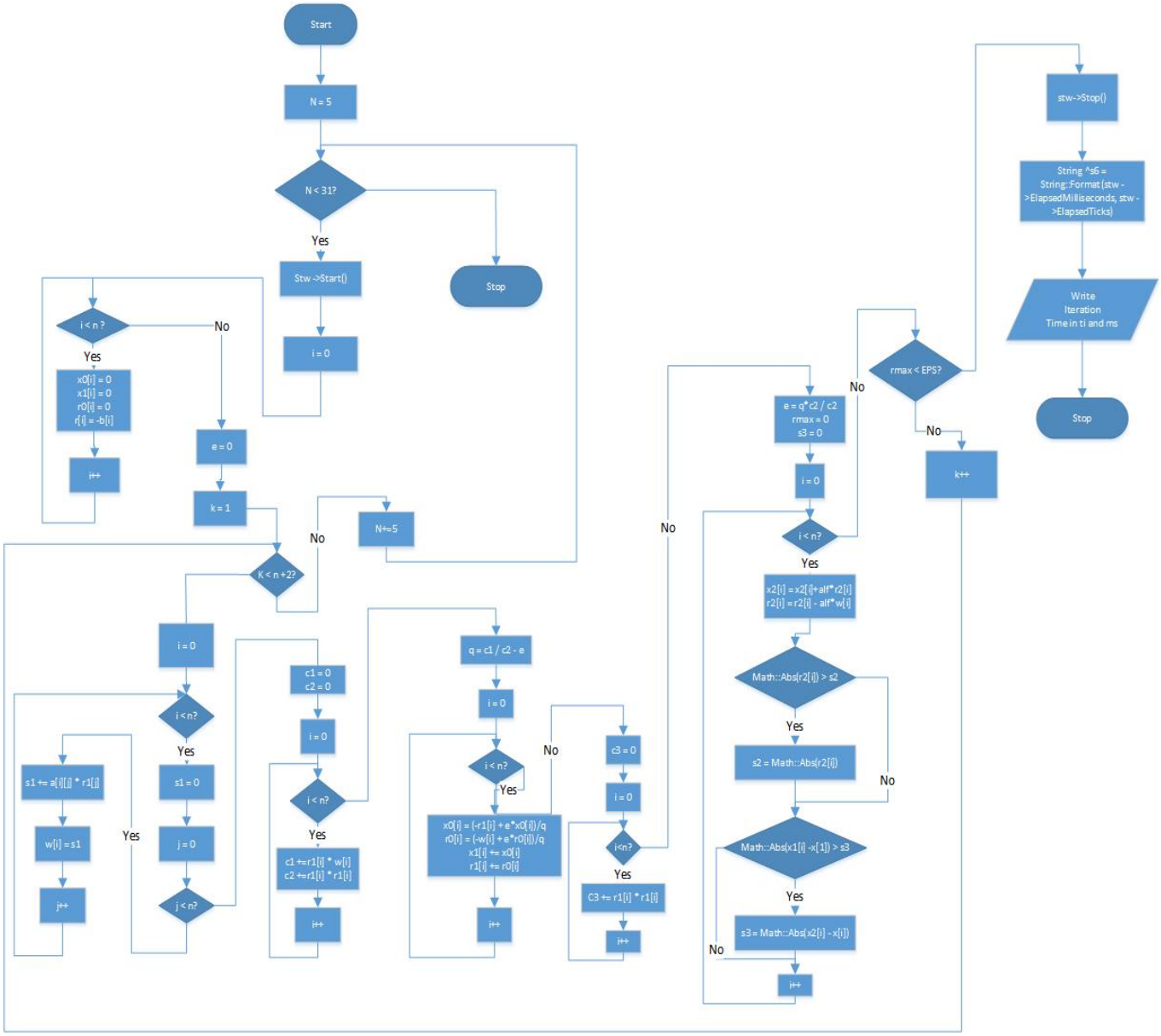


Fig. 3. Block Diagram of the Conjugate Gradient Method