archives
of thermodynamics
Vol. **31**(2010), No. 4, 37–50
DOI: 10.2478/v10173-010-0026-5

The analysis of gradient algorithm effectiveness – two dimensional heat transfer problem

STANISŁAW ŁOPATA*
PAWEŁ OCŁOŃ

Cracow University of Technology, Division of Power Engineering,
31-864 Kraków, al. Jana Pawła II 37, Poland

Abstract The analysis of effectiveness of the gradient algorithm for the two-dimension steady state heat transfer problems is being performed. The three gradient algorithms – the BCG (biconjugate gradient algorithm), the BICGSTAB (biconjugate gradient stabilized algorithm), and the CGS (conjugate gradient squared algorithm) are implemented in a computer code. Because the first type boundary conditions are imposed, it is possible to compare the results with the analytical solution. Computations are carried out for different numerical grid densities. Therefore it is possible to investigate how the grid density influences the efficiency of the gradient algorithms. The total computational time, residual drop and the iteration time for the gradient algorithms are additionally compared with the performance of the SOR (successive over-relaxation) method.

Keywords: Heat transfer; Control volume method; Gradient algorithms

Nomenclature

a	–	width of the square edge, m
A	–	coefficient matrix of the system
A^*	–	coefficient matrix of the dual system
b	–	given constants of the system

*Corresponding author. E-mail address: lopata@mech.pk.edu.pl

b^*	–	given constants of the dual system
dx	–	width of control volume, m
dy	–	height of control volume, m
d	–	conjugate search direction vector
i_{max}	–	maximal number of iterations
l	–	length of the rod, m
n	–	number of Fourier series terms
N	–	number of nodes on the edge
N_{total}	–	total number of the nodes in the computational domain
q	–	auxiliary vector necessary to compute the conjugate search direction vector d in the CGS algorithm
$Q_{i,j}$	–	heat flow rate transferred between the control volumes i and j, W
r	–	residual vector
r^*	–	residual vector of the dual system
r_{max}	–	maximal element of the residual vector
s	–	residual vector without stabilization in the BICGSTAB algorithm
T_i	–	temperature in the control volume i, K
T_{SB}	–	temperature at the side and bottom edges, K
T_U	–	temperature at the uppermost edge, K
u	–	auxiliary vector necessary to compute the conjugate search direction vector d in the CGS algorithm
x	–	vector of the unknowns of the system
x^*	–	vector of the unknowns of the dual system
x_i	–	the x vector after i iterations
x_i^k	–	value of the i-th element of the vector x, the k-th iteration

Greek symbols

α	–	line search parameter
β	–	Gram-Schmidt constant
δ	–	convergence criterion value, K
λ	–	thermal conductivity, W/(m K)
ω	–	stabilization parameter of the BICGSTAB algorithm

1 Introduction

Nowadays, with the rapid increase in the computational power it is possible to use numerical methods for large scale engineering problems. In the design offices, the commercial software using the Finite Element Method [1–3] and the Control Volume Method [4] is applied to optimize the heat performance of devices. For these two methods the computational domain is divided into small elements. The heat balance equation is established for each element and finally the large system of equations is solved for the whole structure in order to examine e.g. the temperature or the heat flux distribution inside the domain.

The iterative methods for solving the large systems of equations are applied in the majority of the numerical codes. They guarantee a quicker way to achieve a convergence level than the direct methods like e.g. the Gauss elimination one. The most effective iterative techniques are probably the gradient methods [5–6]. For heat transfer problems, especially in the cases when the mixed second and third type boundary conditions occur, the matrix of the system of equations is unsymmetrical. Therefore in the work three, utilized for both symmetrical and non-symmetrical matrices, algorithms are tested: the BCG (biconjugate gradients algorithm), the BICGSTAB (biconjugate gradients stabilized algorithm) and the CGS (conjugate gradients squared algorithm).

In order to ensure that the computations are correct, the solution is compared with the analytical one. The two dimensional steady conduction in the infinitely long rod with square cross section is analyzed. The boundary conditions are of the first type. The computations are carried out using the Control Volume Method.

The three computational cases are carried out for 50.1, 100 and 360 thousands of nodes. The total computational time, the iteration number and the average computational time per iteration are investigated to find which algorithm is the most effective one. Moreover, the residual drop in a function of subsequent iterations is followed to examine the algorithm behavior. The results are compared to the widely used SOR algorithm.

2 Control volume method

The boundary conditions for the computational case are presented in Fig. 1. The temperature of the sides and the bottom T_{SB} equals 800 K and on the upper edge of the rod T_U equals 60 K.

The temperature inside the control volumes is unknown. In order to obtain the temperatures in the interior nodes, it is necessary to solve the system of heat balance equation constructed for the inside nodes. The discretization scheme is shown in Fig. 2. For the interior node i the heat balance equation can be written as:

$$Q_{i-N,i} + Q_{i+N,i} + Q_{i+1,i} + Q_{i-1,i} = 0 \,. \tag{1}$$

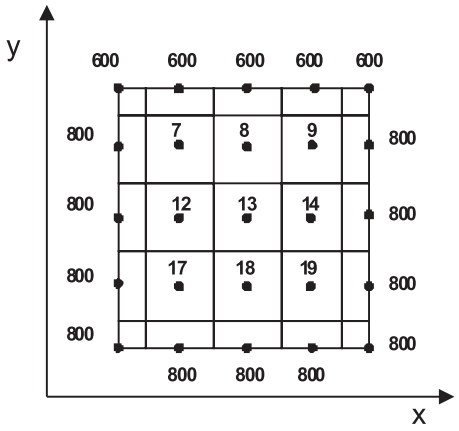

Figure 1. Boundary conditions.

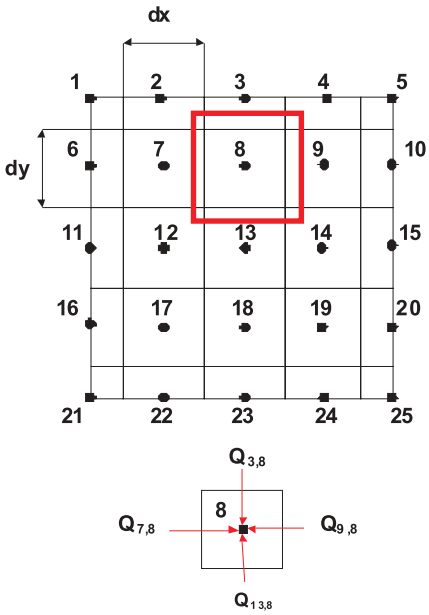

Figure 2. Discretization scheme.

Using the Fourier's law, Eq. (1) is presented in the form:

$$\frac{\lambda}{dy}dx\,l\,(T_{i-N}-T_i)+\frac{\lambda}{dy}dx\,l\,(T_{i+N}-T_i)+ \\ +\frac{\lambda}{dx}dy\,l\,(T_{i+1}-T_i)+\frac{\lambda}{dx}dy\,l\,(T_{i-1}-T_i)=0\,. \tag{2}$$

Denoting the terms:

$$R_x=\frac{\lambda}{dx}dy\,,\quad R_y=\frac{\lambda}{dy}dx\,, \tag{3}$$

Eq. (2) is simplified to the form:

$$R_y\,(T_{i-N}-T)+R_y\,(T_{i+N}-T_i)+R_x\,(T_{i+1}-T_i)+R_x\,(T_{i-1}-T_i)=0 \tag{4}$$

Denoting $diag=-2(R_x+R_y)$ for the all interior nodes shown in Fig. 2 the

system of equations to solve $A \cdot x = b$ is composed as below:

$$A = \begin{pmatrix} diag & R_x & & R_y & & & & & \\ R_x & diag & R_x & & R_y & & & & \\ & R_x & diag & & & R_y & & & \\ R_y & & & diag & R_x & & R_y & & \\ & R_y & & R_x & diag & R_x & & R_y & \\ & & R_y & & R_x & diag & & & R_y \\ & & & R_y & & & diag & R_x & \\ & & & & R_y & & R_x & diag & R_x \\ & & & & & R_y & & R_x & diag \end{pmatrix},$$

$$x = \begin{pmatrix} T_7 \\ T_8 \\ T_9 \\ T_{12} \\ T_{13} \\ T_{14} \\ T_{17} \\ T_{18} \\ T_{19} \end{pmatrix}, \quad b = \begin{pmatrix} -T_2R_y - T_6R_x \\ -T_3R_y \\ -T_4R_y - T_{10}R_x \\ -T_{11}R_x \\ 0 \\ -T_{15}R_x \\ -T_{16}R_y - T_{22}R_x \\ -T_{23}R_y \\ -T_{20}R_x - T_{24}R_y \end{pmatrix}.$$

The empty fields in the matrix A denote 0. The analogical system of equations for the dense mesh is built up using the code written in the C^{++} language. The dimension of the rod square cross section and the number of nodes used for computations are presented in Tab. 1.

Table 1. Computational cases.

Computational case	a [m]	N	N_{total}
1	1	224	50176
2	1	317	100489
3	1	600	360000

The systems of equations for the computational cases presented in Tab. 1 are solved using the gradient methods. The background of the used algorithms is presented in the following section.

3 Iterative gradient algorithms

3.1 Conjugate gradient algorithm

The basic idea of conjugate gradient algorithms (CG) is to minimize the quadratic form (see Fig. 3) given by the equation:

$$f(x) = \frac{1}{2} \cdot x^T \cdot A \cdot x - b^T \cdot x + c \,. \tag{5}$$

The gradient $f'(x)$is defined as:

$$f'(x) = \Big[\frac{\partial}{\partial x_1} f(x), \frac{\partial}{\partial x_2} f(x), ..., \frac{\partial}{\partial x_n} f(x)\Big]^T \,. \tag{6}$$

If the matrix A is symmetric then:

$$f'(x) = A \cdot x - b \,. \tag{7}$$

Setting $f'(x) = 0$, the extreme of the quadratic form is reached. If the matrix is positive definite and symmetric, then the extreme is a minimum. Therefore $A \cdot x - b = 0$ can be solved by finding the x that minimizes $f(x)$. In the case of unsymmetrical matrix, the solution of the system $0.5(A^T + A) \cdot x = b$ is found. The matrix $0.5(A^T + A)$ is symmetric.

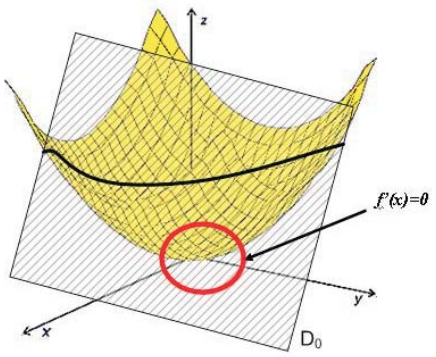

Figure 3. The quadratic form $f(x)$, and line search procedure.

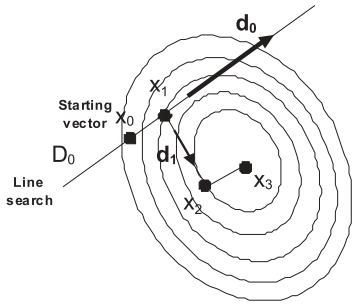

Figure 4. The procedure of searching the minimum of quadratic form.

The gradient algorithm presented in Fig. 4 starts with the arbitrary chosen vector x_0. Then in the steps 1 and 2, the x approaches the bottom of the quadratic form $f(x)$, reaching finally the minimum in step 3. The residual r_i found from the formula $r_i = b - A \cdot x_i$ informs about the direction of the steepest descent.

The i-th step can be expressed as:

$$x_{i+1} = x_i + \alpha_i \cdot d_i \ . \tag{8}$$

Premultiplying (8) by A and adding b leads to:

$$r_{i+1} = r_i - \alpha_i \cdot A \cdot d_i \ , \tag{9}$$

where α_i is the line search parameter given by:

$$\alpha_i = \frac{r_i^T \cdot r_i}{d_i^T \cdot A \cdot d_i} \ , \tag{10}$$

and d denotes the conjugate search direction. The parameter α_i is chosen in order to minimalize the function $f(x)$ along the search direction d_0. This procedure is called *the line search*.

It can be observed in Fig. 4, that the search directions are orthogonal one to another. It is a quick way to find the minimum of $f(x)$. On the basis of the knowledge about r_i the A-orthogonal search directions can be generated by the Gram-Schmidt procedure. Two vectors d_i and d_j are A-orthogonal (conjugate) when:

$$d_i^T \cdot A \cdot d_j^T = 0 \ . \tag{11}$$

The conjugation of d_0 and d_1 vectors (see Fig. 5) implicates, that they are the orthogonal in the stretched space. It can be observed in Fig. 6. The sections of the paraboloid in the stretched space (the quadratic form) are circular, therefore for orthogonal directions d_i as in Fig. 6, the x moves always in the right way – to the center of the section, reaching the minimum finally. The A – conjugation is a mathematical trick to introduce the residuals r_i calculated from Eq. (9) to the Gram-Schmidt procedure.

The final formula for the Gram Schmidt conjugation parameter β, necessary to find the A-orthogonal search directions, can be written in the form:

$$\beta_i = \frac{r_{i+1}^T \cdot r_{i+1}}{r_i^T \cdot r_i} \ . \tag{12}$$

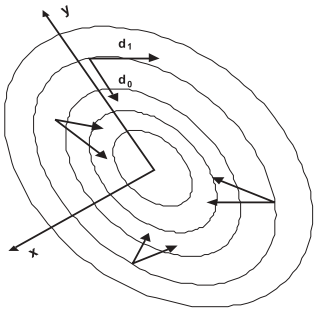

Figure 5. The A-orthogonal (conjugate) direction vectors.

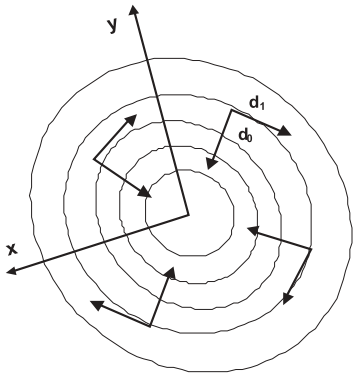

Figure 6. The orthogonal direction vectors.

The orthogonal search directions are obtained from the Gram-Schmidt dependence:

$$d_{i+1} = r_{i+1} + \beta_i \cdot d_i \ . \tag{13}$$

More detailed information about the derivation of the formula (8)–(13) may be found in [5].

3.2 Biconjugate gradient algorithm — BCG and its variations

The algorithm is the version of the general CG algorithm [5], described above. The advantage of the BCG is that it works fine with the nonsymmetrical matrices. Implicitly, the algorithm solves not only the original system $A \cdot x - b$ but also the dual linear system$A^T \cdot x^* - b^*$. This dual system is often ignored in the later parts of the algorithm.

The BCG algorithm has two variations: the CGS and the BICGSTAB which e.g. in the heat transfer problems gain faster convergence. The conjugate gradients algorithm, biconjugate gradients algorithm, conjugate gradients squared algorithm and biconjugate gradients stabilized algorithm are presented in Tab. 2 [6]. These algorithms are applied in the computer code.

Table 2. Gradient algorithms.

Conjugate gradient algorithm (CG)	Biconjugate gradient algorithm (BCG)	Conjugate gradient squared algorithm (CGS)	Biconjugate gradient stabilized algorithm (BICGSTAB)
$r_0 = b - A \cdot x_0$ $d_0 = r_0$ $i = 0$ **Do until** convergence **and** i< i_{max} $\alpha_i = \frac{r_i^T \cdot r_i}{d_i^T \cdot A \cdot d_i}$ $x_{i+1} = x_i + \alpha_i \cdot d_i$ $r_{i+1} = r_i - \alpha_i \cdot A \cdot d_i$ $\beta_i = \frac{r_{i+1}^T \cdot r_{i+1}}{r_i^T \cdot r_i}$ $d_{i+1} = r_{i+1} + \beta_i \cdot d_i$ $i = i + 1$ **Loop**	$r_0 = b - A \cdot x_0$ $r_0^T \cdot r_0^* \neq 0$ $d_0^* = r_0^*, \quad d_0 = r_0$ $i = 0$ **Do until** convergence **and** i< i_{max} $\alpha_i = \frac{r_i^T \cdot r_i}{d_i^T \cdot A \cdot d_i}$ $x_{i+1} = x_i + \alpha_i \cdot d_i$ $r_{i+1} = r_i - \alpha_i \cdot A \cdot d_i$ $r_{i+1}^* = r_i^* - \alpha_i \cdot A^T \cdot d_i^*$ $\beta_i = \frac{r_{i+1}^T \cdot r_{i+1}}{r_i^T \cdot r_i}$ $d_{i+1} = r_{i+1} + \beta_i \cdot d_i$ $d_{i+1}^* = r_{i+1}^* + \beta_i \cdot d_i^*$ $i = i + 1$ **Loop**	$d_0 = r_0 = b - A \cdot x_0$ $r_0^* = r_0$ $u_0 = r_0, \quad d_0 = r_0$ $i = 0$ **Do until** convergence **and** i< i_{max} $\alpha_i = \frac{r_i^T \cdot r_i}{d_i^T \cdot A \cdot d_i}$ $q_i = u_i - \alpha_i \cdot A \cdot d_i$ $x_{i+1} = x_i + \alpha_i \cdot (u_i + +q_i)$ $r_{i+1} = r_i - \alpha_i \cdot A \cdot (u_i + +q_i)$ $\beta_i = \frac{r_{i+1}^T \cdot r_0^*}{r_i^T \cdot r_0^*}$ $u_{i+1} = r_{i+1} + \beta_i \cdot q_i$ $d_{i+1} = u_{i+1} + \beta_i \cdot (q_j + +\beta_i \cdot d_i)$ $i = i + 1$ **Loop**	$r_0 = b - A \cdot x_0$ $p_0 = r_0$ $i = 0$ **Do until** convergence **and** i< i_{max} $\alpha_i = \frac{r_i^T \cdot r_i}{d_i^T \cdot A \cdot d_i}$ $s_j = r_j - \alpha_j \cdot A \cdot d_j$ $\omega_j = \frac{s_j^T \cdot A \cdot s_j}{(s_j^T \cdot A)^2}$ $x_{j+1} = x_j + \alpha_j \cdot d_j + +S\omega_j \cdot s_j$ $r_{j+1} = s_j - \omega_j \cdot A \cdot s_j$ $\beta_i = \frac{r_{i+1}^T \cdot r_0^*}{r_i^T \cdot r_0^*} \cdot (\alpha_j / s_j)$ $d_{j+1} = r_{j+1} + \beta_j \cdot (d_j - -\omega_j \cdot A \cdot d_j)$ $i = i + 1$ **Loop**

4 Comparison of the gradient algorithms

4.1 Comparison between numerical and analytical results

As mentioned before the results of numerical computations are compared with the theoretical solution. Investigated case is the one of the few 2D heat conduction problems where the analytical solution is possible. According to Incropera *et al.* [7], it is possible to compute the temperature distribution $T(x, y)$ in the cross section of the rod, using the dependence:

$$T(x,y) = (T_U - T_{SB}) \frac{2}{\pi} \sum_{n=1}^{\infty} \frac{(-1)^{n+1} + 1}{n} \sin\left(\frac{n\pi x}{a}\right) \frac{\sin\left(\frac{n\pi y}{a}\right)}{\sin(n\pi)} + T_{SB} \,. \tag{14}$$

The comparison of the isotherms obtained using the analytical and numerical solution is presented in Fig 7. The temperature contour plot is

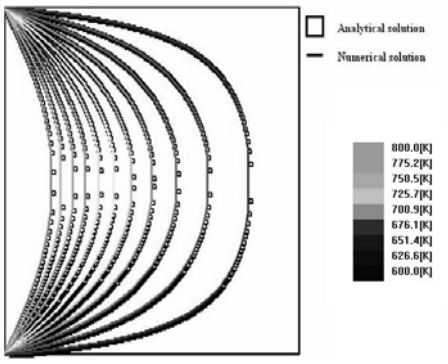

Figure 7. The isotherms – numerical and analytical solution.

Figure 8. The contour plot of temperature distribution.

presented in Fig. 8. The computations are carried out for $N = 224$. In the first step the components of vector x values are assumed as 0 K for each method.

The numerical computations are carried out with the convergence criterion:

$$\delta = r_{\max} = 0.01 \ . \tag{15}$$

The δ is equal to the maximum residual. The analytical solution is computed for $n = 80$. In Fig. 7 it is possible to observe that the numerical solution is consistent with the analytical one.

5 The comparison between the gradient algorithms and the SOR (successive over — relaxation)

The well known SOR algorithm [8] is presented below:

$$x_i^{k+1} = (1-\omega)\, x_i^k + \frac{\omega}{A_{ii}} \left(b_i - \sum_{j>i} A_{ij} x_j^k - \sum_{j<i} A_{ij} x_j^k) \right) \ . \tag{16}$$

The computations are stopped when the maximal difference between x_l^{k+1} and x_l^k is less than 0.01 K. After some computational experiments the value of $\omega = 1.2$ is chosen. This value decreases the number of iterations in

comparison to the value of $\omega = 1.0$ (Gauss-Seidel) by about 30%. The computations are carried out for this value. The single – thread computations are carried out using the Intel Core2 Duo T7250 2.0 GHz processor.

In Fig. 9 the total computational time is shown for the three computational cases, presented in Tab. 1. It is possible to observe that the CGS algorithm is the fastest one in all the cases. In the first computational case, the difference between the total computation time for the CGS and the BICGSTAB algorithms is the smallest and equals 20%. However for the case no. 3, that is the most practical, because of the largest system of equations, the CGS algorithm is 2.5 times faster than the BICGSTAB and 4 times faster than BCG. In each computational case, the gradient algorithms are more efficient than the SOR method, that is indicated in Fig. 9.

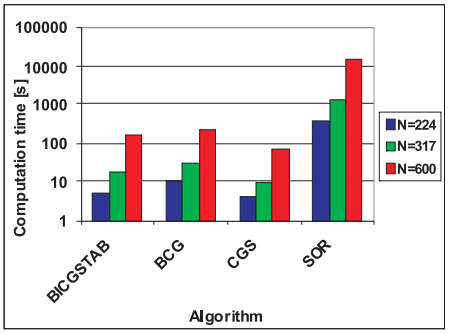

Figure 9. The number of iterations.

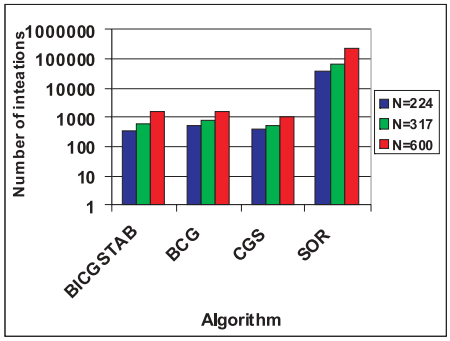

Figure 10. The total computation time.

The number of iterations is presented in Fig. 10. In the computational case no. 1, the number of iterations for the BICGSTAB and the CGS algorithm is nearly the same. However with the increasing size of the equations set (case 2 and case 3), the number of iterations for the BICGSTAB is higher than for the CGS algorithm. In the case 3, the CGS algorithm needs nearly two times less iterations to achieve convergence than the BICGSTAB. The BCG needs the highest number of iterations to converge. In the Fig. 10 it is possible to observe that the number of iterations for the SOR algorithm is over 100 times higher than for the gradient algorithms. It happens in each computational case.

The averaged computational time per iteration is presented in the Fig. 11. The values for the SOR method and the CGS algorithm are the highest. The BCG algorithm needs the longest computation time iteration.

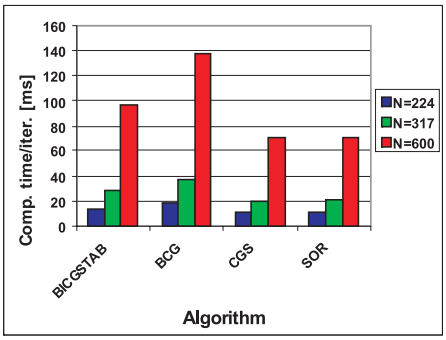

Figure 11. The average computation time for iteration.

The BICGSTAB algorithm is 1.5 times faster than the BCG. The computation time per iteration for the CGS algorithm and for the SOR method is two times shorter than for the BCG. It results from the smallest amount of multiplications of the matrix vector for these algorithms. However, the residuals for the SOR algorithm drop the slowest. This issue is discussed in the next subsection.

5.1 The residual drop

The effectiveness of the gradient algorithms is very high because the residual drop per iteration is much faster than for the SOR method. It can be observed for all computational cases. The case no. 2 (see Tab. 1) is selected for the analysis of the residual drop for each method.

In Fig. 12, the residual drop is presented as a function of iterations for the CGS algorithm. It is possible to observe the relatively slight drop in the first phase of computations. In the second phase after the single rapid increase, the residual starts to drop faster. In the last phase of the computation the residual drop is the highest. For the SOR method (see Fig. 13) the residual drop at the beginning of the computation is very high; however during the whole computation procedure it is nearly constant. In comparison to the CGS algorithm, the average drop per iteration is over one hundred times smaller.

For the BICGSTAB algorithm (see Fig. 14) it is possible to observe that the convergence path is not as smooth as for the CGS method. The sharp increases and decreases of residuals are observed. The algorithm finds the solution after relatively small number of iterations – 612.

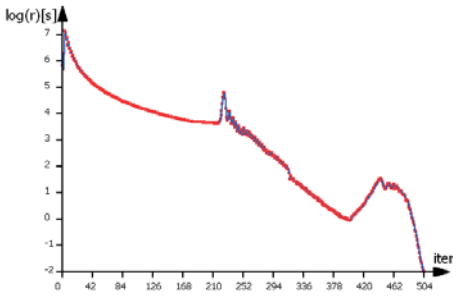

Figure 12. The residuals drop for CGS algorithm ($N = 317$).

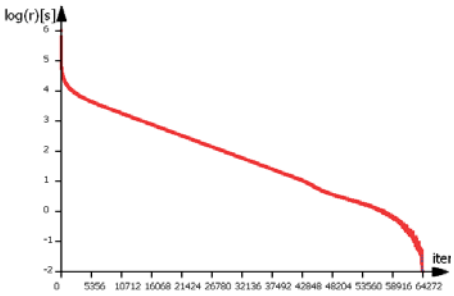

Figure 13. The residuals drop for CGS algorithm ($N = 317$).

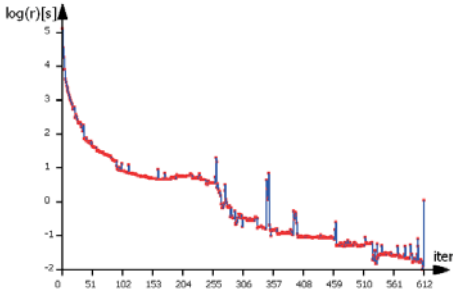

Figure 14. The residuals drop for BICGSTAB algorithm ($N = 317$).

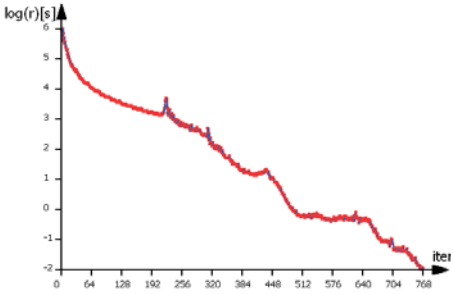

Figure 15. The residuals drop for BCG algorithm ($N = 317$).

The convergence path for the BCG algorithm, presented in Fig. 15, is the smoothest from the all presented gradient algorithms. However, the algorithm needs the highest number of iterations to achieve the desired convergence level.

6 Conclusions and outlook

The paper demonstrates the advantages of using the gradient algorithms during the numerical heat transfer computations. These iterative algorithms are very efficient for heat transfer problems especially for the large systems of equations. The performance of the SOR (successive over relaxation) is exceeded over 100 times. The behavior of the three gradient algorithms – the BCG (biconjugate gradients algorithm), the BICGSTAB (biconjugate

gradients stabilized algorithm) and the CGS (conjugate gradients squared algorithm) was investigated. The CGS algorithm is the most efficient one for the two-dimensional heat transfer conduction problems with first type boundary conditions.

Received 28 August 2010

References

[1] Zienkiewicz O. C., Tayler R. L.: *The Finite Element Method, Vol. 2. Solid Mechanics*. McGraw-Hill, Oxford 2000.

[2] Chung T. J.: *Computational Fluid Dynamics*, Cambridge Univ. Press., Cambridge 2002.

[3] Hoffman K. A, Chiang S.: *Computational Fluid Dynamics*, Vol. 1–3, EES 2000.

[4] Taler J., Duda P.: *Solving Direct and Inverse Heat Conduction Problems*. WNT, Warszawa 2003 (in Polish).

[5] Shewchuk J. R.: *An Introduction to The Conjugate Gradient Metchod Without The Agonizing Pain*, School of Computer Science Carnegie Melon, University Pittsburgh, 1994.

[6] Saad Y.: *Iterative Methods for Sparse Linear Systems*, 2nd edn., 2000.

[7] Incopera F. P., DeWitt D. P., Bergman T., Lavine A.: *Fundamentals of Heat and Mass Transfer*, Willey, Los Angeles 2007.

[8] Chapra S.: *Applied Numerical Methods with MATLAB*, 2nd edn., McGraw-Hill, 2004.