www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

archives
of thermodynamics
Vol. **34**(2013), No. 3, 15–44
DOI: 10.2478/aoter-2013-0013

Comparative study of conjugate gradient algorithms performance on the example of steady-state axisymmetric heat transfer problem

PAWEŁ OCŁOŃ[*]
STANISŁAW ŁOPATA
MARZENA NOWAK

Cracow University of Technology, Department of Thermal Power Engineering, Jana Pawła II 37, 31-864 Kraków, Poland

Abstract The finite element method (FEM) is one of the most frequently used numerical methods for finding the approximate discrete point solution of partial differential equations (PDE). In this method, linear or nonlinear systems of equations, comprised after numerical discretization, are solved to obtain the numerical solution of PDE. The conjugate gradient algorithms are efficient iterative solvers for the large sparse linear systems. In this paper the performance of different conjugate gradient algorithms: conjugate gradient algorithm (CG), biconjugate gradient algorithm (BICG), biconjugate gradient stabilized algorithm (BICGSTAB), conjugate gradient squared algorithm (CGS) and biconjugate gradient stabilized algorithm with l GMRES restarts (BICGSTAB(l)) is compared when solving the steady-state axisymmetric heat conduction problem. Different values of l parameter are studied. The engineering problem for which this comparison is made is the two-dimensional, axisymmetric heat conduction in a finned circular tube.

Keywords: Conjugate gradient method; Finite Element Method; Finned circular tube

[*]Corresponding Author. E-mail: poclon@mech.pk.edu.pl

www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

Nomenclature

h	–	heat transfer coefficient, W/(m^{2}K)
k	–	thermal conductivity in radial, circumferential and axial directions, W/(m K)
l	–	number of GMRES restarts
n	–	number of nodes for a single finite element
n_r, n_z	–	radial and axial components of the unit normal vector
P_i	–	ith degree polynomial (used in CGS and BICGSTAB algorithm)
Q_i	–	ith degree polynomial (used in BICGSTAB algorithm)
q	–	heat flux density, W/m^2
q_v	–	rate of volumetric heat source, W/m^3
R	–	thermal resistance, (m^{2}K)/W
r, z	–	radial and axial coordinates, m
T	–	temperature, K
tol	–	stoping criterion

Matrices and vectors

$[\mathbf{A}]$	–	coefficient matrix of linear system
$[\mathbf{B}]$	–	temperature gradient matrix, K/m
$\{\mathbf{b}\}$	–	right hand side vector of linear system
$[\mathbf{D}]$	–	thermal conductivity matrix, W/(m K)
$\{\mathbf{d}\}$	–	conjugate directions vector
$\{\mathbf{f}\}$	–	vector of nodal loads
$[\mathbf{J}]$	–	Jacobian matrix
$[\mathbf{K}]$	–	stiffness matrix
$[\mathbf{L}]$	–	lower triangular matrix
$[\mathbf{n}]$	–	unit outward normal
$[\mathbf{P}]$	–	preconditioner matrix
$[\mathbf{q}]$	–	recurrence vector
$\{\mathbf{r}\}$	–	residual vector
$\{\mathbf{T}\}$	–	vector of nodal temperatures
$[\mathbf{u}]$	–	auxiliary vector
$\{\mathbf{x}\}$	–	vector of unknowns of linear system
$\{\mathbf{z}\}$	–	vector of unknowns for preconditioned linear system

Greek symbols

α	–	line search parameter
β	–	Gram-Schmidt conjugation parameter
Γ	–	boundary, m
δ	–	thickness of thermal resistance layer, m
ε	–	relative residual
η, ξ	–	local element coordinate
θ	–	circumferential coordinate, rad
κ	–	condition number of $[\mathbf{A}]$ matrix

www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

λ	–	scalar used in CGS and BiCGSTAB algorithms
ρ	–	density, kg/m^3
ϕ	–	shape function
φ	–	circumferential coordinate,m
ψ	–	polynominal
Ω	–	area, m^2
$[\Phi^e]$	–	matrix of element shape functions

Subscripts

b	–	boundary
cg	–	combustion gas
f	–	fin
i	–	iteration number
j	–	node number
r	–	thermal resistance layer
s	–	superheated steam
t	–	tube
0	–	gravity center
∞	–	free steam

Superscripts

e	–	element (the distribution of variable inside element domain)
$*$	–	dual system of linear quations

1 Introduction

Nowadays, as computational power increases rapidly, the numerical methods are widely used for solving the large scale engineering problems. The numerical techniques like the finite element method (FEM) [1–6] and the finite volume method (FVM) [7,8] allow obtaining the approximate solution of partial differential equations (PDE) at discrete points. For the FEM method, the computational domain is divided into small elements. Next, the energy balance equation is formulated for each element. In the subsequent steps, the elements are assembled to estimate the behaviour of the whole structure. Finally, the nodal unknowns are obtained after solving the system of ordinary differential equations (ODE) which results from the assembly procedure. The number of nodes determines the size of the global system of equations. If the numerical grid is dense, then large systems of ODE (time dependent problem) or linear algebraic equations (steady-state problem) are obtained. Hence, sophisticated numerical techniques are necessary to solve the large scale problems with hundreds or even millions of algebraic or differential equations. The iterative methods are an intuitive choice to deal with this task. These numerical techniques perform well in

www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

the large sparse linear systems and often converge faster than the direct methods like, e.g., the Gaussian elimination or the LU factorization [24].

Moreover, the numerical errors are controlled when specifying the relative tolerance, at which the solution of the numerical procedure must converge. Now, the most widely used iterative techniques to solve large linear systems are the conjugate gradient (CG) algorithm [9–13] and the biconjugate gradient (BICG) algorithm with its variants: the biconjugate gradient stabilized (BICGSTAB), conjugate gradient squared (CGS) and the biconjugate gradient stabilized l-version (BICGSTAB(l)) algorithms. Those variants have been developed to extend the ability of the conjugate gradient (CG) algorithm to solve the linear systems with nonsymmetric coefficient matrices.

One of the major drawbacks of the iterative methods is the lack of robustness. Therefore the preconditioning techniques have been developed to improve convergence and efficiency of the iterative solvers. The preconditioned conjugate gradient (PCG) algorithm is applied in many commercial FEM packages. Some of them (e.g., ANSYS [32]) allow modifying a single iteration algorithm to improve the global performance of the PCG.

Preconditioning is a transformation of the original linear system to the equivalent one. This new linear system can be solved more efficiently and with higher accuracy, using the iterative solver.

Test computations which compare all the aforementioned iterative algorithms (their preconditioned and unpreconditioned versions) are performed for the axisymmetric heat transfer problem: heat conduction in a finned circular tube with a thermal resistant layer between the fin and the tube (see Fig. 1). Importance of thermal contact resistance is widely discussed in the literature [14–18, 25–29]. The major problem with this type of thermal resistance is improper heat transfer across the finned tube between the media flowing the inside and the outside of the tube. Therefore the temperature field inside the heat transfer domain is disturbed strongly, because the finned tubes do not operate properly.

In the reference cylindrical coordinate frame, the axisymmetric heat transfer problems are often solved when it is possible to simplify the geometry and the boundary conditions in such manner that they do not depend on the circumferential coordinate. The axisymmetric model of heat transfer can be applied to the simplified convection-conduction systems. Basing on the analytical correlations [14], it is often possible to estimate the heat transfer coefficients of the liquids flowing inside and outside of the tube,

www.czasopisma.pan.pl www.journals.pan.pl

as well as the free-stream temperatures of media flowing inside and outside of the finned tube. Next, these heat transfer coefficients and free-stream temperatures are applied as the boundary conditions to the axisymmetric heat transfer problems, and finally the temperature distribution inside the heat transfer domain can be determined.

The obtained temperature field from the axisymmetric analysis can be used – as the starting values for the solid domain, when solving the large conjugate heat transfer problems [19,20]. It can significantly reduce the computational time, that can be an important gain if the large computational fluid dynamics (CFD) or FEM models are being solved many times, e.g., to find the optimal process parameters or the optimal geometrical parameters of the analysed system.

For the heat transfer domain presented in Fig. 1, the convective heat transfer occurs on the outer and the inner surface of the finned tube. The hot combustion gas flows around the finned tube and the superheated steam flows inside the tube. The steam temperature is significantly lower than the temperature of the combustion gas. Heat transfer occurs across the wall of the tube and its finned surface. The heat transfer coefficients of the gas and superheated steam sides are denoted as h_{cg} and h_s respectively. The symbols T_{cg} and T_s relate to the free stream temperature of the combustion gas and the superheated steam respectively.

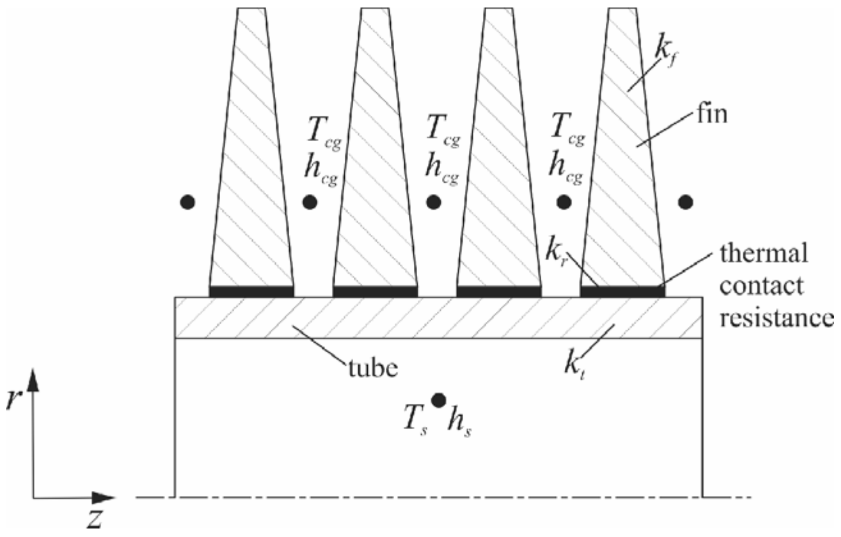

Figure 1. The finned circular tube with the thermal resistant layer.

The thermal properties of the fin and tube materials are thermal conductivity of the fin material, k_f, and of the tube material, k_t, respectively. The thermal resistant layer exists between the fin and the tube. Thickness of the resistance layer is denoted as δ. Thermal conductivity of the thermal resistant layer is denoted as k_r.

The heat transfer domain is axisymmetric, thus the energy equation can be solved, using the axisymmetric isoparimetric quadrilateral finite elements. For the system presented here, the energy equation is formulated using the Galerkin FEM. Then, it is being solved for the nodal temperatures using the different iterative algorithms based on the conjugate gradient method. The similar heat transfer problem was studied in [30], however the performance of only four conjugate gradient algorithms were presented: the BiCG algorithm, the CGS algorithm, the BiCGSTAB algorithm and the BICGSTAB(2) algorithm. In the presented paper the influence of GMRES (l = 2–5) restarts on the computational efficiency of BiCGSTAB (l) algorithm is analysed. Additionally the performance of CG and PCG algorithms is studied.

2 Galerkin finite element method for axisymmetric heat transfer problems

For isotropic material, the steady-state heat transfer equation for solids, in a cylindrical coordinate frame, can be written as follows [2,3,6,21]:

$$\frac{1}{r}\frac{\partial}{\partial r}\left(rk\frac{\partial T}{\partial r}\right)+\frac{1}{r^2}\frac{\partial}{\partial \varphi}\left(k\frac{\partial T}{\partial \varphi}\right)+\frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right)+q_v=0\ , \tag{1}$$

where r, φ and z are the radial, circumferential and axial coordinates. The thermal conductivity is denoted as k and the volumetric rate of heat source is q_v. Assuming the temperature independency on the circumferential coordinate the axisymmetric loadings and the boundary conditions, Eq. (1) can be simplified to

$$k\left(\frac{\partial^2 T}{\partial r^2}+\frac{1}{r}\frac{\partial T}{\partial r}+\frac{\partial^2 T}{\partial z^2}\right)+q_v=0\ . \tag{2}$$

If n denotes the number of nodes in the element and j is the local node number in the element, then temperature distribution inside the element can be approximated using the nodal temperatures and the element shape

www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

functions $\phi_j^e = \phi_j^e(r,z)$ in the following way:

$$T^e = T^e(r,z) = \sum_{j=1}^{n} \phi_j^e(r,z)\, T_j^e = \left[\phi_1^e\ \phi_2^e\ \phi_3^e\ \phi_4^e\right] \left\{ \begin{array}{c} T_1^e \\ T_2^e \\ T_3^e \\ T_4^e \end{array} \right\} = [\mathbf{\Phi}^e]\{\mathbf{T}^e\}\ . \tag{3}$$

In Eq. (3), the element shape functions matrix is denoted as $[\mathbf{\Phi}^e]$ and the nodal temperature vector is $\{\mathbf{T}^e\}$. Applying the weighted residuals method, the Galerkin formulation [2,5,6,21] to Eq. (2), one can obtain

$$\int\limits_{\Omega^e} \left[k\left(\frac{\partial^2 T^e}{\partial r^2} + \frac{1}{r}\frac{\partial T^e}{\partial r} + \frac{\partial^2 T^e}{\partial z^2} \right) + q_v \right] \phi_i^e dr d\varphi dz = 0\ . \tag{4}$$

The volume integral of the term in parentheses can be expressed as

$$\int\limits_{\Omega^e} k\left(\frac{\partial^2 T^e}{\partial r^2} + \frac{1}{r}\frac{\partial T^e}{\partial r} + \frac{\partial^2 T^e}{\partial z^2} \right) \phi_i^e dr d\varphi dz = $$
$$= \int\limits_{\Omega^e} k\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial T^e}{\partial r} \right) + \frac{\partial^2 T^e}{\partial z^2} \right) \phi_i^e dr d\varphi dz\ . \tag{5}$$

If the integral is independent on the circumferential coordinate, then

$$\int\limits_{\Omega^e} k\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial T^e}{\partial r} \right) + \frac{\partial^2 T^e}{\partial z^2} \right) \phi_i^e dr d\varphi dz = $$
$$= 2\pi \int\limits_{\Omega^e} k\left(\frac{\partial}{\partial r}\left(r\frac{\partial T^e}{\partial r} \right) + r\frac{\partial^2 T^e}{\partial z^2} \right) \phi_i^e dr dz\ . \tag{6}$$

Applying a divergence theorem to Eq. (6), it yields

$$2\pi \int\limits_{\Omega^e} k\left(\frac{\partial}{\partial r}\left(r\frac{\partial T^e}{\partial r} \right) + r\frac{\partial^2 T^e}{\partial z^2} \right) \phi_i^e dr dz = $$
$$= -2\pi \int\limits_{\Omega^e} kr\left(\frac{\partial \phi_i^e}{\partial r}\frac{\partial T^e}{\partial r} + \frac{\partial \phi_i^e}{\partial z}\frac{\partial T^e}{\partial z} \right) dr dz + \int\limits_{\Gamma^e} kr\frac{\partial T^e}{\partial \mathbf{n}}\phi_i^e d\Gamma\ , \tag{7}$$

where the boundary integration is performed in the r-z plane and the $\mathbf{n}$ is the unit outward normal vector to the element boundary Γ^e. The boundary

www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

integral can be written as follows:

$$\int\limits_{\Gamma^e} kr\frac{\partial T^e}{\partial \mathbf{n}}\phi_i^e d\Gamma = \int\limits_{\Gamma^e} kr\left(-\frac{\partial T^e}{\partial r}n_r - \frac{\partial T^e}{\partial z}n_z\right)\phi_i^e d\Gamma \ , \tag{8}$$

where n_r and n_z are the radial and the axial components of the unit normal vector. Consider the bilinear quadrilateral element (see Fig. 2), for which the following boundary conditions and initial conditions are applied:

- the Neumann type boundary condition

$$-kr\left(\frac{\partial T^e}{\partial r}n_r + \frac{\partial T^e}{\partial z}n_z\right) = rq_b \ , \quad r,z \subset \Gamma_1 \ , \tag{9}$$

  where $q_b = q_b(r,z)$ denotes heat flux specified on the element boundary (Fig. 2, edge 1-2)

- the mixed type (convective) boundary condition

$$-kr\left(\frac{\partial T^e}{\partial r}n_r + \frac{\partial T^e}{\partial z}n_z\right) = -rh(T^e - T_\infty) \ , \quad r,z \subset \Gamma_2 \ , \tag{10}$$

  where the free stream temperature $T_\infty = T_\infty(r,z)$ and the heat transfer coefficient $h = h(r,z)$ must be given (Fig. 2, edge 2-3);

- the insulation surface boundary condition (Fig. 2, edge 3-4)

$$-kr\left(\frac{\partial T^e}{\partial r}n_r + \frac{\partial T^e}{\partial z}n_z\right) = 0 \ , \quad r,z \subset \Gamma_3 \ , \tag{11}$$

- the Dirichlet boundary condition (Fig. 2, edge 4-1)

$$T^e = T_b = T_b(r,z) \ , \quad r,z \subset \Gamma_4 \ , \tag{12}$$

  where the boundary temperature T_b must be given. The Dirichlet boundary condition sets the fixed value of the nodal temperature and must be satisfied by Eq. (4).

Substituting Eqs. (9)–(11) into Eq. (8), the following is obtained:

$$-\int\limits_{\Gamma^e} kr\left(\frac{\partial T^e}{\partial r}n_r + \frac{\partial T^e}{\partial z}n_z\right)\phi_i^e d\Gamma = \int\limits_{\Gamma_1^e} rq_b\phi_i^e d\Gamma - \int\limits_{\Gamma_2^e} rh(T^e - T_\infty)\phi_i^e d\Gamma \ . \tag{13}$$

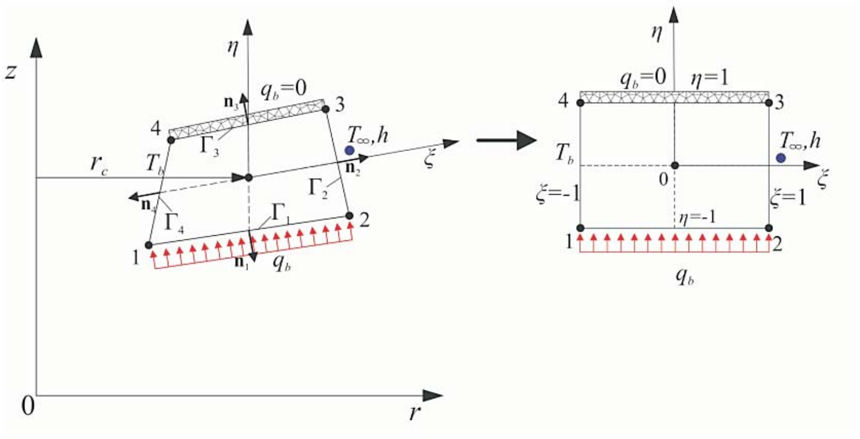

Figure 2. Bilinear quadrilateral element.

Including Eqs. (13) and (7) into Eq. (6), it is obtained

$$\int\limits_{\Omega^e} k\left(\frac{\partial^2 T^e}{\partial r^2}+\frac{1}{r}\frac{\partial T^e}{\partial r}+\frac{\partial^2 T^e}{\partial z^2}\right)\phi_i^e dr d\varphi dz = -2\pi k\int\limits_{\Omega^e} r\left(\frac{\partial\phi_i^e}{\partial r}\frac{\partial T^e}{\partial r}+\frac{\partial\phi_i^e}{\partial z}\frac{\partial T^e}{\partial z}\right)drdz+$$
$$+\int\limits_{\Gamma_1^e} r q_b \phi_i^e d\Gamma - \int\limits_{\Gamma_2^e} rh(T^e-T_\infty)\phi_i^e d\Gamma\ . \tag{14}$$

The remaining volume integrals in Eq. (4) can be expressed as

$$\int\limits_{\Omega^e} q_v\phi_i^e dr d\varphi dz = 2\pi\int\limits_{\Omega^e} r q_v \phi_i^e drdz\ . \tag{15}$$

Substituting Eq. (3) into Eqs. (13)–(14), it is obtained

$$\int\limits_{\Omega^e} k\left(\frac{\partial^2 T^e}{\partial r^2}+\frac{1}{r}\frac{\partial T^e}{\partial r}+\frac{\partial^2 T^e}{\partial z^2}\right)\phi_i^e drdz=$$
$$= -2\pi k\int\limits_{\Omega^e} r\left(\frac{\partial\phi_i^e}{\partial r}\sum_{j=1}^{n}\frac{\partial\phi_j^e}{\partial r}+\frac{\partial\phi_i^e}{\partial z}\sum_{j=1}^{n}T_j^e\frac{\partial\phi_j^e}{\partial z}\right)drdz+$$
$$+\int\limits_{\Gamma_1^e} r\phi_i^e q_b d\Gamma - \int\limits_{\Gamma_2^e} rh\phi_i^e\left(\sum_{j=1}^{n}\phi_j^e T_j^e - T_\infty\right)d\Gamma\ . \tag{16}$$

www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

Then after substituting Eqs. (15)–(16) into Eq. (4), the following is obtained:

$$
\begin{aligned}
&2\pi k \int\limits_{\Omega^e} r \left(\frac{\partial \phi_i^e}{\partial r} \sum_{j=1}^{n} \frac{\partial \phi_j^e}{\partial r} + \frac{\partial \phi_i^e}{\partial z} \sum_{j=1}^{n} T_j^e \frac{\partial \phi_j^e}{\partial z} \right) drdz - \\
&\quad - \int\limits_{\Gamma_1^e} r\phi_i^e q_b d\Gamma - \int\limits_{\Gamma_2^e} rh\phi_i^e \left(\sum_{j=1}^{n} \phi_j^e T_j^e - T_\infty \right) d\Gamma + \\
&\quad + 2\pi \int\limits_{\Omega^e} r q_v \phi_i^e drdz = 0 \ . \qquad (17)
\end{aligned}
$$

Introducing matrix notation, the element stiffness matrix is a sum of a conductive $[\mathbf{K}^e_{cond}]$ and a convective $[\mathbf{K}^e_{conv}]$ matrices and can be expressed as

$$
[\mathbf{K}^e] = [\mathbf{K}^e_{cond}] + [\mathbf{K}^e_{cov}] \ , \qquad (18)
$$

or

$$
\begin{aligned}
[\mathbf{K}^e] &= \int\limits_{\Omega^e} r[\mathbf{B}^e]^T[\mathbf{D}^e][\mathbf{B}^e]d\Omega^e + \int\limits_{\Gamma_2^e} rh[\mathbf{\Phi}^e]^T[\mathbf{\Phi}^e]d\Gamma^e = \\
&= 2\pi r_c \int\limits_{\Omega^e} [\mathbf{B}^e]^T[\mathbf{D}^e][\mathbf{B}^e]d\Omega^e + \int\limits_{\Gamma_2^e} rh[\mathbf{\Phi}^e]^T[\mathbf{\Phi}^e]d\Gamma^e \ , \qquad (19)
\end{aligned}
$$

where the superscript T denotes the transpose. The element load vector includes the heat flux and convective boundary condition, as well as the heat source term in a following way:

$$
\begin{aligned}
&\{\mathbf{f}^e\} = \{\mathbf{f}^e_{q_v}\} + \{\mathbf{f}^e_{q}\} + \{\mathbf{f}^e_{f}\} = \\
&\quad = 2\pi \int\limits_{\Omega^e} r q_v [\mathbf{\Phi}^e]^T d\Omega^e + \int\limits_{\Gamma_1^e} rq[\mathbf{\Phi}^e]^T d\Gamma^e + \int\limits_{\Gamma_2^e} rhT_\infty [\mathbf{\Phi}^e]^T d\Gamma^e \ . \qquad (20)
\end{aligned}
$$

In Eq. (22), r_c denotes the gravity center of the element and the matrices of temperature derivatives $[\mathbf{B}^e]$ and material properties $[\mathbf{D}^e]$ are defined as

$$
[\mathbf{B}^e] = \left\{ \begin{array}{c} \frac{\partial T^e}{dr} \\ \frac{\partial T^e}{dz} \end{array} \right\} = \left\{ \begin{array}{cccc} \frac{\partial \phi_1^e}{dr} & \frac{\partial \phi_2^e}{dr} & \cdots & \frac{\partial \phi_n^e}{dr} \\ \frac{\partial \phi_1^e}{dz} & \frac{\partial \phi_2^e}{dz} & \cdots & \frac{\partial \phi_n^e}{dz} \end{array} \right\}, [\mathbf{D}^e] = \left[\begin{array}{cc} k & 0 \\ 0 & k \end{array} \right] . \qquad (21)
$$

www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

Substituting Eqs. (22) and (23) into Eq. (19), the following can be obtained:

$$[\mathbf{K}^e]\{\mathbf{T}^e\} = \{\mathbf{f}^e\} \, . \tag{22}$$

The components of the shape function matrix $[\mathbf{\Phi}^e]$ for the bilinear axisymmetric quadrilateral element can be expressed in the natural (local) element coordinates ξ, η as for the plane bilinear quadrilateral elements [1,2]

$$\begin{aligned} \phi_1^e &= \tfrac{1}{4}(1-\xi)1-\eta \, , \quad & \phi_2^e &= \tfrac{1}{4}(1+\xi)1-\eta \, , \\ \phi_3^e &= \tfrac{1}{4}(1+\xi)1+\eta \, , \quad & \phi_4^e &= \tfrac{1}{4}(1-\xi)1+\eta \, . \end{aligned} \tag{23}$$

Using these element shape functions, the element coordinates in a global cylindrical coordinate system are interpolated as follows:

$$r^e = \sum_{j=1}^{4} \varphi_j^e(\xi,\eta)\, r_j^e = [\varphi_1^e \; \varphi_2^e \; \varphi_3^e \; \varphi_4^e] \begin{Bmatrix} r_1^e \\ r_2^e \\ r_3^e \\ r_4^e \end{Bmatrix} = [\mathbf{\Phi}^e]\,\{\mathbf{r}^e\} \; , \tag{24}$$

$$z^e = \sum_{j=1}^{4} \varphi_j^e(\xi,\eta)\, z_j^e = [\varphi_1^e \; \varphi_2^e \; \varphi_3^e \; \varphi_4^e] \begin{Bmatrix} z_1^e \\ z_2^e \\ z_3^e \\ z_4^e \end{Bmatrix} = [\mathbf{\Phi}^e]\,\{\mathbf{z}^e\} \; , \tag{25}$$

where $\{\mathbf{r}^e\}$, $\{\mathbf{z}^e\}$ are the vectors of nodal coordinates in a global cylindrical coordinate frame. The derivatives of nodal coordinates with respect to local coordinates are calculated using the chain rule

$$\frac{\partial \phi_i^e}{\partial \xi} = \frac{\partial \phi_i^e}{\partial r}\frac{\partial r}{\partial \xi} + \frac{\partial \phi_i^e}{\partial z}\frac{\partial z}{\partial \xi} \; , \tag{26}$$

and

$$\frac{\partial \phi_i^e}{\partial \eta} = \frac{\partial \phi_i^e}{\partial r}\frac{\partial r}{\partial \eta} + \frac{\partial \phi_i^e}{\partial z}\frac{\partial z}{\partial \eta} \; . \tag{27}$$

Equations (29) and (30) can be rewritten in matrix form

$$\begin{Bmatrix} \frac{\partial \phi_i^e}{\partial \xi} \\ \frac{\partial \phi_i^e}{\partial \eta} \end{Bmatrix} = \begin{bmatrix} \frac{\partial r}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial r}{\partial \eta} & \frac{\partial z}{\partial \eta} \end{bmatrix} \begin{Bmatrix} \frac{\partial \phi_i^e}{\partial r} \\ \frac{\partial \phi_i^e}{\partial z} \end{Bmatrix} = [\mathbf{J}] \begin{Bmatrix} \frac{\partial \phi_i^e}{\partial r} \\ \frac{\partial \phi_i^e}{\partial z} \end{Bmatrix} \; , \tag{28}$$

www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

where $[\mathbf{J}]$ is the Jacobian matrix of coordinate mapping. Therefore, the partial derivatives of the shape function with respect to the global coordinates, necessary to calculate the matrix $[\mathbf{B}^e]$ (see Eq. 21), are

$$\left\{ \begin{array}{c} \frac{\partial \phi_i^e}{\partial r} \\ \\ \frac{\partial \phi_i^e}{\partial z} \end{array} \right\} = [\mathbf{J}]^{-1} = \left\{ \begin{array}{c} \frac{\partial \phi_i^e}{\partial \xi} \\ \\ \frac{\partial \phi_i^e}{\partial \eta} \end{array} \right\} . \tag{29}$$

The coordinate mapping implies, that the differential area element is given by

$$d\Omega = drdz = |\mathbf{J}|d\xi d\eta \ , \tag{30}$$

where $|\mathbf{J}|$ is a determinant of the Jacobian matrix. Therefore, Eqs. (22) and (23) can be written as

$$\begin{aligned}[] [\mathbf{K}^e] &= 2\pi r_c \int\limits_{\Omega^e} [\mathbf{B}^e]^T [\mathbf{D}^e][\mathbf{B}^e] d\Omega^e + \int\limits_{\Gamma^e} rh[\mathbf{\Phi}^e]^T[\mathbf{\Phi}^e]d\Gamma^e = \\ &= 2\pi r_c \int\limits_{-1}^{1}\int\limits_{-1}^{1} [\mathbf{B}^e]^T [\mathbf{D}^e][\mathbf{B}^e] d\xi d\eta + \int\limits_{\Gamma^e} rh[\mathbf{\Phi}^e]^T[\mathbf{\Phi}^e]d\Gamma^e \ , \end{aligned} \tag{31}$$

$$\{\mathbf{f}^e\} = 2\pi \int\limits_{-1}^{1}\int\limits_{-1}^{1} rq_v[\mathbf{\Phi}^e]^T|\mathbf{J}|d\xi d\eta + \int\limits_{\Gamma^e} rq[\mathbf{\Phi}^e]^T d\Gamma^e \int\limits_{\Gamma^e} rhT_\infty[\mathbf{\Phi}^e]^T d\Gamma^e \ . \tag{32}$$

At the heat flux boundary conditions Eqs. (9) and (10) are incorporated in Eq. (35) as

$$\begin{aligned} \{\mathbf{f}^e\} = \{\mathbf{f}^e_{qv}\} + \{\mathbf{f}^e_q\} + \{\mathbf{f}^e_h\} = 2\pi q_v [\mathbf{\Phi}^e]^T[\mathbf{\Phi}^e] \left\{ \begin{array}{c} r_1 \\ r_2 \\ r_3 \\ r_4 \end{array} \right\} d\Omega^e + \\ + \frac{q\pi}{3} l_{12} \left\{ \begin{array}{c} 2r_1 + r_2 \\ r_2 + 2r_2 \\ 0 \\ 0 \end{array} \right\} + \frac{\pi h T_\infty}{3} l_{23} \left\{ \begin{array}{c} 0 \\ 2r_2 + r_3 \\ r_2 + 2r_3 \\ 0 \end{array} \right\} \end{aligned} \tag{33}$$

and the convective part of the matrix $[\mathbf{K}^e]$ is

$$[\mathbf{K}^e_{conv}] = \frac{2\pi h l_{23}}{12} \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 3r_2 & r_3 & 0 \\ 0 & r_2 & 3r_3 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right] . \tag{34}$$

www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

The area integral in Eqs. (34) is calculated using the Gauss quadrature rule for two dimensions [2,5] in order to obtain $[\mathbf{K}^e]$. If the matrix $[\mathbf{K}^e]$ and the vector of the element loads $\{\mathbf{f}^e\}$ are evaluated for each element, then the assembly procedure [1,2,5,8] is performed in order to obtain the global form of Eq. (26)

$$[\mathbf{K}]\{\mathbf{T}\} = \{\mathbf{f}\} \,, \tag{35}$$

which is solved using the conjugate gradient based algorithms.

3 Iterative solvers

The FEM discretization of the parabolic PDE given by Eq. (1) results in forming the linear systems, which size is determined by the number of nodes in the discretized heat transfer domain. In the practical engineering problems, the number of nodes can even exceed 10^7 for the large three dimensional structures or CFD problems. However, from the authors' experience, the number of nodes rarely exceeds 10^6 for the axisymmetric problems, which are often solved to simplify the three-dimensional problem or to find the estimated initial values for solving the large three-dimensional problems. One of the most common choices for solving the moderate size and the large linear systems is the conjugate gradient method. This method works well for the sparse symmetric and positive definite matrices which comprise from the FEM discretization of the parabolic PDE.

3.1 Conjugate gradient algorithm

The conjugate gradient (CG) algorithm [9,10] is the efficient method for solving the large sparse linear systems with the positive definite and symmetric matrix $[\mathbf{A}]$. The basic idea of the conjugate gradient algorithms is to find the minimum of the quadratic form given by the formula

$$f(\mathbf{x}) = \frac{1}{2}\{\mathbf{x}\}^T[\mathbf{A}]\{\mathbf{x}\} - \{\mathbf{b}\}^T\{\mathbf{x}\} + c \,, \tag{36}$$

where $[\mathbf{A}]$ is the coefficient matrix of linear system,$\{\mathbf{x}\}$ is the vector of unknowns and $\{\mathbf{b}\}$ is the right side vector, and c is the arbitrary chosen constant. The gradient $f'(\mathbf{x})$ can be defined as

$$f'(\mathbf{x}) = \left[\frac{\partial}{\partial x_1}f(\mathbf{x}), \frac{\partial}{\partial x_2}f(\mathbf{x}) \,, \ldots \,, \frac{\partial}{\partial x_n}f(\mathbf{x})\right]^T \,. \tag{37}$$

If the matrix $[\mathbf{A}]$ is symmetric then Eq. (37) simplifies to

$$f'(\mathbf{x}) = [\mathbf{A}]\{\mathbf{x}\} - \{\mathbf{b}\} \ . \tag{38}$$

If $f'(\mathbf{x})$=0 and the matrix $[\mathbf{A}]$ is positive definite and symmetric, then the minimum of $f(\mathbf{x})$ is found. Hence, the system of linear equations $[\mathbf{A}]\{\mathbf{x}\}-\{\mathbf{b}\}=\{\mathbf{0}\}$ can be solved by finding the vector of unknowns $\{\mathbf{x}\}$ that minimizes $f(\mathbf{x})$. The conjugate gradient algorithm starts with the arbitrary chosen vector $\{\mathbf{x}\}=\{\mathbf{x}_0\}$. Then in the successive iteration, the vector of unknowns$\{\mathbf{x}\}$ approaches the minimum of the quadratic form $f(\mathbf{x})$. The residual vector $\{\mathbf{r}_i\}$ found from the formula $\{\mathbf{r}_i\}=\{\mathbf{b}\}-\{\mathbf{A}\}\{\mathbf{x}_i\}$ indicates the direction of the steepest descent. Here, the subscript i denotes the iteration number. The $i^{th}+1$ step can be expressed as [9,13]

$$\{\mathbf{x}_{i+1}\} = \{\mathbf{x}_i\} + \alpha_i\{\mathbf{d}_i\} \ . \tag{39}$$

Multiplying Eq. (39) by $[\mathbf{A}]$ and adding $\{\mathbf{b}\}$ to both sides of Eq. (39) one obtains the residual vector for the $i^{th}+1$ step

$$\{\mathbf{r}_{i+1}\} = \{\mathbf{r}_i\} - \alpha_i[\mathbf{A}]\{\mathbf{d}_i\} \ , \tag{40}$$

where α_i is the line search parameter given by

$$\alpha_i = \left(\{\mathbf{r}_i^T\}\{\mathbf{r}_i\}\right) / \left(\{\mathbf{d}_i^T\}[\mathbf{A}]\{\mathbf{d}_i\}\right) \ . \tag{41}$$

In Eqs. (39)–(41) symbol $\{\mathbf{d}_i\}$ denotes the vector of conjugate search direction. The parameter α_i is chosen in order to minimize the function $f(\mathbf{x})$ along vector $\{\mathbf{d}_i\}$. This procedure is called *the line search.* For the CG algorithm, the search directions are orthogonal to each other. Moving along these directions is the fastest method to find the minimum of $f(\mathbf{x})$. If the residual vector $\{\mathbf{r}_i\}$ is known, then the $\mathbf{A}$ – orthogonal search directions can be generated by the Gram-Schmidt conjugation procedure [19], which states, that two vectors $\{\mathbf{d}_i\}$ and $\{\mathbf{d}_j\}$ are $\mathbf{A}$ – orthogonal (conjugate), if

$$\{\mathbf{d}_i\}^T[\mathbf{A}]\{\mathbf{d}_j\} = 0 \ . \tag{42}$$

Proceeding towards the orthogonal search direction, vector $\{\mathbf{x}\}$ always approaches to the minimum of $f(\mathbf{x})$. The $\mathbf{A}$-conjugation allows introduction of the residual vector $\{\mathbf{r}_i\}$ to the Gram-Schmidt procedure. The Gram-Schmidt conjugation parameter β at step i, necessary to find the $\mathbf{A}$-orthogonal search directions, is given by

$$\beta_i = \left(\{\mathbf{r}_{i+1}\}^T\{\mathbf{r}_{i+1}\}\right) / \left(\{\mathbf{r}_i\}^T\{\mathbf{r}_i\}\right) \ . \tag{43}$$

www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

Then, the orthogonal search directions are obtained from the following relationship

$$\{\mathbf{d}_{i+1}\} = \{\mathbf{r}_{i+1}\} + \beta_i\{\mathbf{d}_i\}\ . \tag{44}$$

The procedure given by Eqs. (39)–(44) is repeated in successive steps until satisfying the stopping criterion given in

$$\varepsilon = (\|\{\mathbf{b}\} - |\mathbf{A}|\,\{\mathbf{x}_i\}\|\,/\,\|\{\mathbf{b}\}\|) < tol\ . \tag{45}$$

The symbol ε denotes the relative residual (the scaled root mean square residual), which is the ratio of the Euclidean norm of the residual vector $\{\mathbf{r}_{i+1}\} = \{\mathbf{b}\} - \{\mathbf{A}\}\{\mathbf{x}_{i+1}\}$ and the Euclidean norm of the right hand side vector $\{\mathbf{b}\}$. The value of ε must be lower than the specified tolerance *tol* in order to stop the algorithmic operation. For the computational cases presented in this paper, the tolerance parameter *tol* is assumed to be equal to 10^{-5}.

3.2 Preconditioned conjugate gradient method

One of the largest drawbacks of iterative solvers is lack of robustness. In order to improve the efficiency of the conjugate gradient algorithm, the preconditioning conjugate gradient (PCG) technique has been developed. This allows the transformation of the original linear system into the equivalent one, with the same solution, which can be solved easier with the iterative solver [10,13]. For the symmetric matrices comprising from the FEM discretization, one of the most common option is the Cholesky factorization of sparse matrices. The preconditioner matrix $[\mathbf{P}]$ is given in the following form

$$[\mathbf{P}] = [\mathbf{L}][\mathbf{L}]^T\ , \tag{46}$$

where $[\mathbf{L}]$ is the lower triangular matrix obtained from the decomposition of the coefficient matrix $[\mathbf{A}]$ (see Subsection 3.1). Using the preconditioner matrix, the conjugate gradient algorithm can be modified in the following way [9,13]:

Pseudocode for preconditioned conjugate gradient algorithm (PCG)

```
Assume arbitrary $\{\mathbf{x}_0\}$, e.g., $\{\mathbf{x}_0\} = \{\mathbf{0}\}$, set $tol = 10^{-5}$
Compute $\{\mathbf{r}_0\} = \{\mathbf{b}\} - [\mathbf{A}]\{\mathbf{x}_0\}$, $\{\mathbf{z}_0\} = [\mathbf{P}]^{-1}\{\mathbf{r}_0\}$
Assume $\{\mathbf{d}_0\} = \{\mathbf{z}_0\}$

Compute $\varepsilon = ||\{\mathbf{r}_0\}||/||\{\mathbf{b}\}||$

For $i = 0, 1, \ldots$ until $\varepsilon > tol_a$

$\alpha_i = (\{\mathbf{z}_i\}^T\{\mathbf{r}_i\})/(\{\mathbf{d}_i\}^T[\mathbf{P}]\{\mathbf{d}_i\})$
$\{\mathbf{x}_{i+1}\} = \{\mathbf{x}_i\} + \alpha_i \{\mathbf{d}_i\}$
$\{\mathbf{r}_{i+1}\} = \{\mathbf{r}_i\} - \alpha_i [\mathbf{A}]\{\mathbf{d}_i\}$

Compute $\varepsilon = || \{\mathbf{r}_{i+1}\}||/||\{\mathbf{b}\}||$

$\{\mathbf{z}_{i+1}\} = [\mathbf{P}]^{-1} \{\mathbf{r}_{i+1}\}$
$\beta_i = (\{\mathbf{z}_{i+1}\}^T\{\mathbf{r}_{i+1}\})/(\{\mathbf{z}_i\}^T\{\mathbf{r}_i\})$
$\{\mathbf{d}_{i+1}\} = \{\mathbf{z}_{i+1}\} + \beta_i \{\mathbf{d}_i\}$

End
```

The residual from the preconditioned system

$$[\mathbf{P}]^{-1} [\mathbf{A}]\{\mathbf{x}\} = [\mathbf{P}]^{-1} \{\mathbf{b}\} \tag{47}$$

is written as

$$\{\mathbf{z}_i\} = [\mathbf{P}]^{-1} \{\mathbf{r}_i\} \ . \tag{48}$$

It is possible to note in the pseudocode of the PCG method, that this residual is used to calculate scalars α and β in the CG algorithm. The linear system given in (48) is solved at each iterative step, thus the preconditioner must be chosen in the way, that solving the linear system

$$[\mathbf{P}] \{\mathbf{z}_i\} = \{\mathbf{r}_i\} \ , \tag{49}$$

has a low computational cost. The incomplete LU factorization for sparse matrices ILU(0) with no fill-in [9,13], one of the most often applied preconditioners, is used for the computations presented in this paper. All the other algorithms presented in this paper can be preconditioned in the similar way.

3.3 Biconjugate conjugate gradient method

The conjugate gradient method is not suitable for nonsymmetric linear systems, because the residual vectors cannot be made orthogonal in short recur-

www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

rences. The CG algorithm is prone to round-off errors, another drawback, that may cause the search direction vectors $\mathbf{d}$ to lose their $\mathbf{A}$-orthogonality and the algorithm can encounter difficulties in converging. It may happen, when the condition number of the symmetric matrix is high. Sometimes, slower alternative algorithms can be used to ensure that the converged solution can be obtained. One of the most widely used algorithm to handle the nonsymmetric systems is the biconjugate gradient (BICG) algorithm [13], which replaces the orthogonal sequence of residuals by two mutually orthogonal sequences, at the cost of losing a minimization of $\mathrm{f}(\mathbf{x})$. Implicitly, the algorithm solves not only the original system $[\mathbf{A}]\{\mathbf{x}\} - \{\mathbf{b}\}$ but also the dual linear system $[\mathbf{A}]^T\{\mathbf{x}^*\} - \{\mathbf{b}^*\}$. This dual system is often ignored in the later parts of the algorithm. Two sets of residuals are updated in the iterative procedure in the following way:

$$\{\mathbf{r}_{i+1}\} = \{\mathbf{r}_i\} - \alpha_i[\mathbf{A}]\{\mathbf{d}_i\}\ , \tag{50}$$

$$\{\mathbf{r}^*_{i+1}\} = \{\mathbf{r}^*_i\} - \alpha_i[\mathbf{A}]^T\{\mathbf{d}^*_i\}\ . \tag{51}$$

The residuals of the dual system are denoted as $\{\mathbf{r}^*\}$. The two sets of search directions are updated as given below

$$\{\mathbf{d}_{i+1}\} = \{\mathbf{r}_i\} + \beta_i[\mathbf{A}]\{\mathbf{d}_i\}\ , \tag{52}$$

$$\{\mathbf{d}^*_{i+1}\} = \{\mathbf{r}^*_i\} + \beta_i[\mathbf{A}]^T\{\mathbf{d}^*_i\}\ . \tag{53}$$

The parameters α and β at ith step are obtained from

$$\alpha_i = \left(\{\mathbf{r}^*_i\}^T\{\mathbf{r}_i\}\right) / \left(\{\mathbf{d}^*_i\}^T[\mathbf{A}]\{\mathbf{d}_i\}\right), \tag{54}$$

$$\beta_i = \left(\{\mathbf{r}^*_{i+1}\}^T\{\mathbf{r}_{i+1}\}\right) / \left(\{\mathbf{r}^*_i\}^T\{\mathbf{r}_i\}\right). \tag{55}$$

3.4 Conjugated gradient squared (CGS)

The conjugate gradient squared (CGS) method [9,15] is the modification of the BICG developed to improve its performance. In the BICG method a residual vector $\{\mathbf{r}_i\}$ can be regarded as a product of $\{\mathbf{r}_0\}$ and an ith degree polynomial in $[\mathbf{A}]$, that is

$$\{\mathbf{r}_i\} = Q_i\left([\mathbf{A}]\right)\{\mathbf{r}_0\}\,. \tag{56}$$

This same polynomial satisfies

$$\{\mathbf{r}^*_i\} = Q_i\left([\mathbf{A}]^T\right)\{\mathbf{r}^*_0\}\,. \tag{57}$$

www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

The idea of this algorithm is to avoid computation of $\{\mathbf{r}^*\}$ and $\{\mathbf{d}^*\}$ given by Eqs. (51) and (53), which are used in calculation of scalars α and β in the BICG algorithm, by calculating these scalars on the basis of the recurrences (56) and (57).

The scalar λ, which is the nominator in Eqs. (54) and (55), can be expressed on the basis of Eqs. (56) and (57) as follows:

$$\lambda_i = (\{\mathbf{r}_i^*\}, \{\mathbf{r}_i\}) = \left(Q_i\left([\mathbf{A}]^T\right)\{\mathbf{r}_0^*\}, Q_i([\mathbf{A}])\{\mathbf{r}_0\}\right) = \left(\{\mathbf{r}_0^*\}, Q_i^2([\mathbf{A}])\{\mathbf{r}_0\}\right) . \tag{58}$$

Therefore only the $\{\mathbf{r}^*\}$ at the initial step is necessary for calculating α and β. Moreover, if $Q_i([\mathbf{A}])$ reduces the $\{\mathbf{r}^*\}$ to a smaller number, then it is expected that the squared $Q_i^2([\mathbf{A}])$ may improve this effect. The detailed explanation how to incorporate the recurrence given by Eq. (58) in the BiCG algorithm is given in [13]. The recurrence vector $\{\mathbf{q}\}$ is defined as

$$\{\mathbf{q}_i\} = \{\mathbf{u}_i\} + \alpha_i[\mathbf{A}]\{\mathbf{d}_i\} , \tag{59}$$

where $\{\mathbf{u}\}$ is an auxiliary vector given by

$$\{\mathbf{u}_i\} = \{\mathbf{r}_i\} + \beta_{i-1}\{\mathbf{q}_{i-1}\} . \tag{60}$$

These vectors $\{\mathbf{q}\}$ and $\{\mathbf{u}\}$ are incorporated in the CGS algorithm as below [13]:

Pseudocode for conjugate gradient squared algorithm (CGS)

```
  Assume arbitrary $\{\mathbf{x}_0\}$, e.g., $\{\mathbf{x}_0\} = \{\mathbf{0}\}$, set $tol = 10^{-5}$
  Compute $\{\mathbf{r}_0\} = \{\mathbf{b}\} - [\mathbf{A}]\{\mathbf{x}_0\}$, $\{\mathbf{r}_0^*\} = \{\mathbf{r}\}$, $\{\mathbf{u}_0\} = \{\mathbf{r}_0\}$
  Assume $\{\mathbf{d}_0\} = \{\mathbf{u}_0\}$
  Compute $\varepsilon = ||\{\mathbf{r}_0\}||/||\{\mathbf{b}\}||$

  For $i = 0, 1, \ldots$ until $\varepsilon > tol$

  $\alpha_i = (\{\mathbf{r}_0^*\}^T\{\mathbf{r}_i\})/(\{\mathbf{r}_0^*\}^T [\mathbf{A}]\{\mathbf{d}_i\})$
  $\{\mathbf{q}_i\} = \{\mathbf{u}_i\} - \alpha_i[\mathbf{A}]\{\mathbf{d}_i\}$
  $\{\mathbf{x}_{i+1}\} = \{\mathbf{x}_i\} + \alpha_i (\{\mathbf{u}_i\} + \{\mathbf{q}_i\})$
  $\{\mathbf{r}_{i+1}\} = \{\mathbf{r}_i\} - \alpha_i [\mathbf{A}]\{\mathbf{d}_i\}(\{\mathbf{u}_i\} + \{\mathbf{q}_i\})$

  Compute $\varepsilon = ||\{\mathbf{r}_{i+1}\}||/||\{\mathbf{b}\}||$

  $\beta_i = (\{\mathbf{r}_0^*\}^T \{\mathbf{r}_{i+1}\})/(\{\mathbf{r}_0^*\}^T \{\mathbf{r}_i\})$
  $\{\mathbf{u}_{i+1}\} = \{\mathbf{r}_{i+1}\} + \beta_i\{\mathbf{q}_i\}$
  $\{\mathbf{d}_{i+1}\} = \{\mathbf{u}_{i+1}\} + \beta_i (\mathbf{q}_i + \beta_i \{\mathbf{d}_i\})$

  End
```

www.czasopisma.pan.pl www.journals.pan.pl

In this algorithm the matrix by vector products with $[\mathbf{A}]^T$ is no longer used. Furthermore, the calculations of vectors $\{\mathbf{r}^*\}$ and $\{\mathbf{d}^*\}$ are no longer necessary. Therefore, the performance of the CGS is expected to be better than of the BICG algorithm. The drawback of the CGS algorithm is that the polynomials Q are squared, therefore an irregular convergence pattern occurs and the round off errors can be more damaging than for the BICG algorithm.

3.5 Biconjugated gradient stabilized method

The biconjugate gradient stabilized (BICGSTAB) method [22,23] has been developed to solve nonsymmetric linear systems while avoiding the often irregular convergence patterns of the CGS algorithm. In a single iteration algorithm the sequence $Q_i^2\left([\mathbf{A}]\right)\{\mathbf{r}_0\}$ is replaced by $\psi([\mathbf{A}])Q_i([\mathbf{A}])\{\mathbf{r}_0\}$, where $\psi\left([\mathbf{A}]\right)$ is the ith degree polynomial describing the steepest descent update. Thus, the scalar λ is given by

$$\lambda_i = \left(\{\mathbf{r}_0^*\}, \psi_i\left([\mathbf{A}]\right) Q_i\left([\mathbf{A}]\right)\{\mathbf{r}_0\}\right). \tag{61}$$

The BICGSTAB is the combination of the BICG and the generalized minimal residuals method (GMRES) [9,13] where each BICG step is followed by a GMRES(1) (i.e., one GMRES restarted at each step) step in order to repair the irregular convergence behaviour of the CGS, as an improvement the bicgstab has been developed for. Nevertheless, if the matrix $[\mathbf{A}]$ has large complex eigenpairs [9], such a repair may be ineffective due to the use of first degree minimum residual polynomials. In such cases, the BICGSTAB is likely to stagnate as confirmed by numerical experiments. The higher-degree minimum residual polynomials could handle this situation better [23]. Therefore the BICGSTAB(2) and the more general BICGSTAB(l) have been developed. In the BICGSTAB(l), a GMRES(l) step follows every one BICG step. The BICGSTAB(l) is equivalent to the BICGSTAB(l) with $l = 2$. The BICGSTAB(l) algorithm improves the convergence behaviour of the BICGSTAB algorithm, but the computation cost of one iteration is significantly larger and depends on the number of the GMRES restarts.

4 Numerical example

In this section the results of numerical computations of the temperature field inside a finned circular tube with a thermal resistant layer between the fin and the tube are presented. The single thread computations were performed on the personal computer with the following parameters: processor Intel Core i7, 2.8 GHz, memory 8 GB. The numerical codes were written in MATLAB (release R2011b) [31]. The goal of these computations was not to achieve the solution in the shortest possible time, that can be done using parallel computing and coding the FEM procedures in FORTRAN or C instead of MATLAB, but to determine which conjugate gradient algorithms (see Section 3) are suitable for solving efficiently the different types of axisymmetric heat transfer problems.

In order to analyse the effectiveness of the conjugate gradient algorithms, introduced in Section 3, the numerical code which allows determination of the steady-state and transient temperature distribution inside the finned circular tube (see Fig. 3) has been developed.

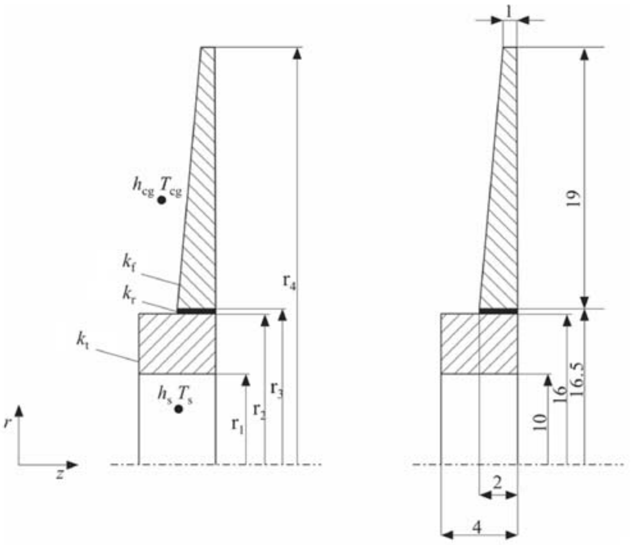

Figure 3. Geometrical model of the finned circular tube.

The code which incorporates the isoparametric quadrilateral elements (see Fig. 2), allows mesh refinement, therefore it is possible to increase the mesh density and consequently to investigate the algorithm performance for different size of linear systems.

The conjugate gradient algorithms are put into the MATLAB routines [9] for solving the linear systems comprising from the FEM discretization. The tube and fin are made of carbon steel AISI 1010, which properties evaluated at temperature $T = 748.15$ K are given in Tab. 1. For the presented computational example this temperature value is assumed to be equal to the expected mean value of temperature in heat transfer domain. Thermal resistance, $R = \delta/k_r$, and thermal conductivity of the thermal resistance layer, k_r, and the tube and fin material, $k_t = k_f$, as well as the heat transfer coefficients for combustion gas-side and steam-side are listed in Tab. 1. Free steam temperatures of the combustion gas and the superheated steam are also given in Tab. 1. The thermal resistance layer is assumed to consist of the mixed combustion gas and the corrosion residues with thermal conductivity significantly lower than thermal conductivity of the tube material. In all the presented computational examples the volumetric rate of the heat source $q_v = 0$ (see Eq. (2)).

Table 1. Material properties and thermal boundary conditions.

Parameter	Value	Unit
$k_f = k_t$	44	W/(m K)
k_r	0.5	W/(m K)
R	1×10^{-3}	(m^2 K)/W
$\delta = \mathrm{r}_3 - \mathrm{r}_2$	5×10^{-4}	M
h_{cg}, h_s	60, 2000	W/(m^{2}K)
T_{cg}, T_s	873.15, 673.15	K

If the material properties and the boundary condition are given, then the linear system given by Eq. (35) is formed using the Galerkin FEM formulation, as described in Section 2. Next, this linear system is being solved using different iterative algorithms described in Section 3.

The numerical results for a coarse mesh are illustrated in Fig. 4. The values of nodal temperatures do not change up to 3 significant digits, when the mesh is refined. The grid shown in Fig. 4 is refined in order to compare the effectiveness of the iterative algorithms, as described in Section 3, for solving the linear systems of different sizes.

Four computational cases (see Tab. 2) are studied. For these computations, the starting value vector $\{\mathbf{T}_0\} = \{\mathbf{x}_0\}$ is equal to 273.15 K for all the iterative algorithms based on the conjugate gradient method.

www.czasopisma.pan.pl www.journals.pan.pl

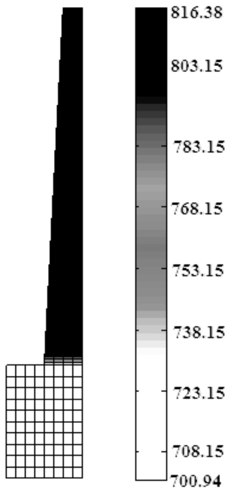

Figure 4. The steady-state solution for material properties and boundary conditions given in Tab. 1.

Table 2. The number of nodes, the number of finite elements and the condition number of stiffness matrix for performed computational cases.

Case no.	Number of nodes	Number of finite elements	Condition number κ ($[\mathbf{K}]$)
1	10012	9650	1.84×10^{6}
2	52650	51840	1.06×10^{7}
3	100422	99260	1.87×10^{7}
4	261931	260000	5.31×10^{7}

In Tab. 2, there are the number of finite element and the number of nodes, which determine the size of global stiffness matrix $[\mathbf{K}]$. Furthermore the condition number of the global stiffness matrices $\kappa([\mathbf{K}])$ defined as [24]

$$\kappa(\mathbf{K}) = \left\|\mathbf{K}^{-1}\right\| \|\mathbf{K}\| \tag{62}$$

is given for the presented computational cases. If this number is large, e.g., $\kappa([\mathbf{K}]) > 10^{9}$, then the problem is ill-conditioned, and the iterative solvers may encounter difficulties in converging.

The total computational time and the number of iterations performed, until the desired convergence criterion (45) is satisfied, are shown in Tab. 3

(algorithms without preconditioning) and in Tab. 4 (algorithms with preconditioning). For the low number of nodes (case no. 1) the robustness of the analysed iterative algorithms is comparable; the algorithms work fast and satisfy the desired convergence criterion in a short period of time.

Table 3. Computational times and number of iterations performed for the BICG, CGS, BICGSTAB, BICGSTAB(l) algorithms, without preconditioning.

Case		BICG STAB	CGS	BICG	BICG STAB(2)	BICG STAB(3)	BICG STAB(4)	BICG STAB(5)	CG
1	CPU time, s	1.3	1.3	1.2	1.6	1.9	1.9	1.9	1.0
2		21.8	17.1	16.1	30.1	31.6	33.1	32.8	11.62
3		68.8	62.4	41.2	90.4	74.9	75.2	94.1	29.1
4		545.3	283.3	237.9	602.1	652.3	640.3	614.1	164.2
1	Iterations	1027	1299	1377	505	365	260	197	1048
2		3385	3281	3385	1622	979	698	520	3641
3		5532	6301	4310	2478	1217	844	765	4587
4		14392	9776	8480	5203	3435	2318	1645	9025

Table 4. Computational time and number of iterations performed for the BICG, CGS, BiCGSTAB, BICGSTAB(l) algorithms, with ILU preconditioner.

Case		BICG STAB	CGS	BICG	BICG STAB(2)	BICG STAB(3)	BICG STAB(4)	BICG STAB(5)	CG
1	CPU time, s	0.5	0.5	0.8	0.63	0.59	0.63	0.67	0.38
2		8.8	6.5	10.8	7.9	8.6	8.8	9.1	4.0
3		33.8	20.1	33.0	26.6	25.9	27.5	27.3	11.13
4		107.1	93.6	149.4	117.3	127.6	140.4	138.2	56.5
1	Iterations	222	226	237	108.25	68	53	41	256
2		656	571	578	244	167	124	98	604
3		1294	858	839	400	254	192	144	869
4		1322	1278	1330	568	396	279	233	1377

As expected, the most efficient one is the CG algorithm, which is about 30% faster for small size of linear systems (case no. 1, Tab. 2) and about 50% faster for the large linear systems (case no. 4, Tab. 2) than the BICG algorithm.

However, if the large systems of equations are solved (cases no. 2–4), the CG, BICG and CGS algorithms achieve the valid solution in the shortest time. It was expected so, because the CG is adapted to solving large symmetric linear systems resulting from the FEM discretization of the heat transfer equation. Moreover, the CG computational cost for the single iteration algorithm is the lowest from all the presented algorithms. Another recommended option for solving large linear systems (case no. 4, Tab. 2) is the preconditioned CGS algorithm (see Tab. 4), which converged 1.65 times slower than the CG algorithm.

The computational time and the number of iterations listed in Tab. 4 confirms that the preconditioned algorithms converge significantly faster than their unpreconditioned versions. The most pronounced benefits of using the ILU(0) preconditions can be observed when the large linear systems are being solved (Tab. 4, case 4). For this computational case the ratio of the computational time for the preconditioned and the unpreconditioned versions of the analysed algorithm varies from 1.6 to 5.5. The highest values are obtained for the BICGSTAB and the BICGSTAB(2) algorithms. The lowest ratio is obtained for the BICG algorithm for all the presented computational cases.

As mentioned previously, the vector of the starting values for all the tested algorithms is assumed as $\{\mathbf{x}_0\} = \{\mathbf{273.15}\}$ K. There is no significant improvement in the performance of the BICGSTAB(l) algorithm for these initial values of vector $\{\mathbf{x}\}$, when the number of the GMRES restarts l is greater than 2. This can be observed for both the preconditioned and unpreconditioned versions of the BICGSTAB(3)–(5) algorithms. The BICGSTAB(2) is the most robust from all the BICGSTAB(2)–(5) algorithms in majority of the analysed computational cases. With increase of the l number, the count of the performed iteration is lower. However, because of the inner GMRES iterations, the computational cost of a single iteration of the BICGSTAB(l) grows with the increasing l number. Therefore, the total computational time does not drop significantly with decreasing number of iterations, and it can even be greater than for the BICGSTAB (see Tab. 4).

The convergence path for computational case no. 1 (Tab. 2) is shown in Fig. 5 for the algorithms without preconditioning and in Fig. 6 for the algorithms with preconditioning. As mentioned in Section 3.4, a strongly irregular convergence pattern exists when the linear systems are solved using the CGS algorithm. It happens because the polynomials $Q([\mathbf{A}])$ are squared in subsequent iterations (see Section 3.4).

www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

When the solution is obtained using the unpreconditioned algorithms, then the values of relative residuals, ε, are the highest for the CGS algorithm. Furthermore, the CGS algorithm requires higher number of iterations, than the other modifications of the BICG algorithm – the BICGSTAB and the BICGSTAB(2)–(5) in order to converge. The smoothest relative residual drop in succeeding iterations is observed for the BICG and the CG algorithms, but the number of performed iterations is slightly greater than for other algorithms.

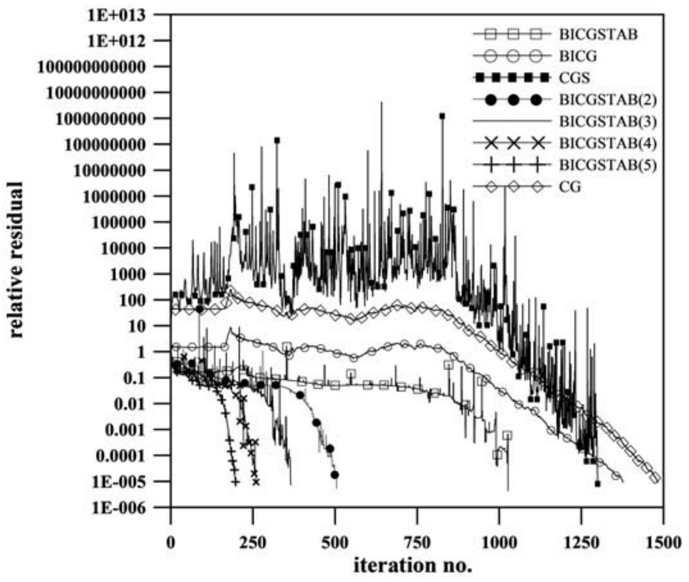

Figure 5. Convergence path for different conjugate gradient algorithms without preconditioning, case no. 1 (see Tab. 3).

If the ILU(0) preconditioner is applied (Fig. 6), then the relative residual path is smoother than when the algorithms works without preconditioning. Moreover, the number of performed iterations to achieve the desired convergence level is considerably lower for all applied algorithms. The convergence path for the largest linear system obtained for computational case no. 4 is presented in Figs. 7 and 8.

Comparing Figs. 5–6 and Figs. 7–8 one can note, that when the size of the linear system increases, number of iterations performed to achieve the desired convergence level also increases. It can even happen, that the convergence level is not achieved. In Fig. 7 one can observe that the CGS algorithm failed to converge to the desired value of $tol = 10^{-5}$ and stopped

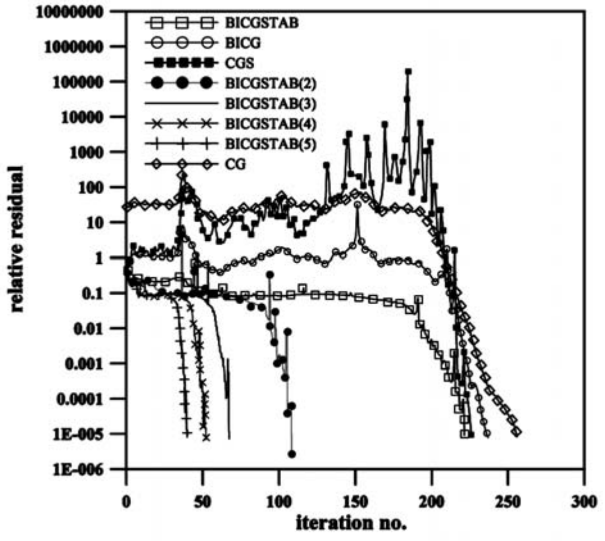

Figure 6. Convergence path for different conjugate gradient algorithms with preconditioning, case no. 1 (see Tab. 4).

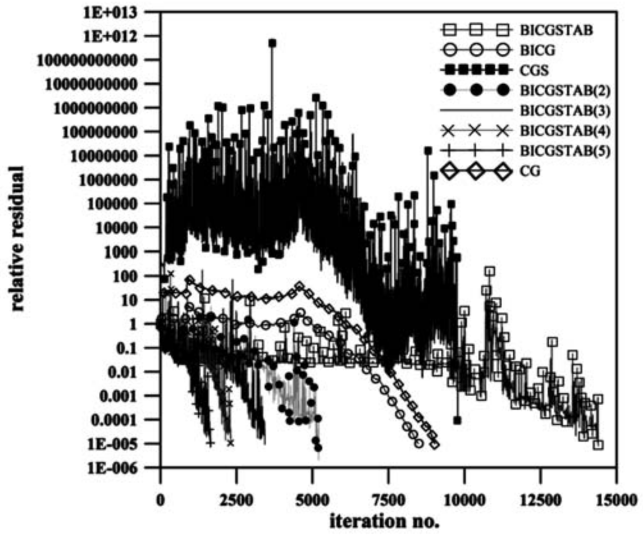

Figure 7. Convergence path for different conjugate gradient algorithms with preconditioning, case no. 4 (see Tab. 4).

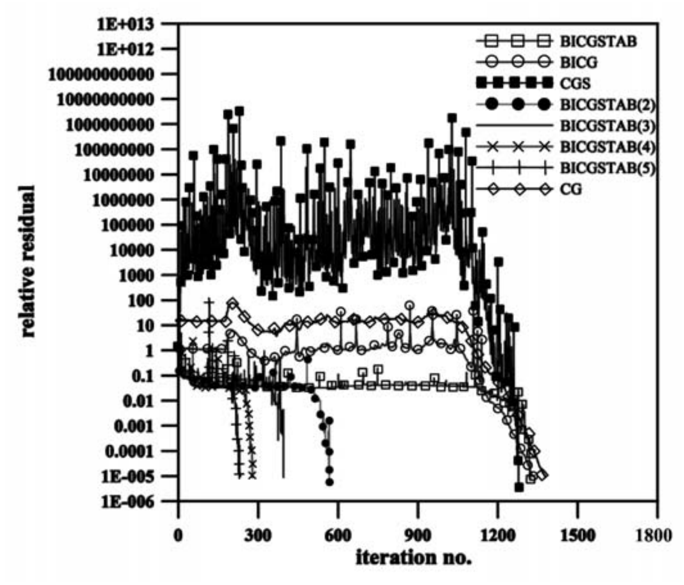

Figure 8. Convergence path for different gradient algorithms without preconditioning, case no. 4 (see Tab. 3).

at $\varepsilon = 0.00035$. For the other algorithms, the convergence level can be achieved.

For the computational case no. 4 (Tab. 3) the smoothest convergence path is one for the unpreconditioned CG and BICG algorithms (see Fig. 7). Surprisingly, when the preconditioning for the BICG is being applied, then the residual path is getting sharper and the number of performed iterations is greater when comparing to the BICGSTAB, the BICGSTAB(2)–(5) and the CGS (see Fig. 8). Probably, the other type of the preconditioner, different than the ILU(0) may improve the convergence pattern for the BICG algorithm.

One can observe, comparing Figs. 7 and 8, that the number of iterations performed for the preconditioned algorithms is up to 10 times lower than for the unpreconditioned versions. Despite the greater computational cost of a single iteration, due to solving Eq. (49) at each iterative step, the preconditioning reduces significantly the computational time of the algorithm (compare Tabs. 3 and 4). Therefore, for the large linear systems it is strongly recommended to use the preconditioned gradient algorithms. The preconditioning can also increase the accuracy of the algorithm [9]. The preconditioned CGS algorithm satisfies the convergence criterion (see Fig. 8) but the unpreconditioned CGS (see Fig. 7) does not (relative residuals are larger than *tol*).

5 Summary

This paper demonstrates the application of the finite element method for solving steady-state heat conduction in the finned circular tube. In order to solve the moderate-size or the large-size linear systems, the efficient solvers must be applied. Therefore, the performance of these iterative algorithms were tested: the conjugate gradient, the biconjugate gradient, the conjugate gradient squared, the biconjugate gradient stabilized and the biconjugate gradient stabilized with l generalized minimal residual method restarts biconjugate gradient stabilized and the biconjugate gradient stabilized (l).

For the steady-state heat conduction and temperature independent thermal conductivity, the CG algorithm is the most robust one among the analyzed algorithms. It converged the fastest for all analysed computational cases. In general, for the coefficient matrix $[\mathbf{A}]$, which is symmetric and positive definite this algorithm performs the fastest. The other Conjugate Gradient algorithms perform slower: the CGS algorithm is the fastest from CGS, BiCG, BiCGSTAB and BiCGSTAB(l). However when the coefficient matrix $[\mathbf{A}]$ is large and the preconditioning is not applied, the CGS algorithm encounters difficulties in converging to the desired tolerance value. For the presented computational example the increase in the value of l parameter of BICGSTAB(l) algorithm decreases the number of iterations, but does not speed up the computations. This occurs because the computational cost of one GMRES restart is large and for the presented axisymmetric heat transfer problem the $l = 2$ value should be used.

The ILU(0) factorization preconditioner of the coefficient matrix $[\mathbf{A}]$ has been applied to the iterative solvers in order to improve their performance. For steady-state heat conduction, when the large linear systems are to be solved and the vector of the starting values $\{\mathbf{x}_0\}$ is far from the solution, the preconditioned CG and CGS algorithms are the most robust ones. However, without applying the preconditioning technique, the CGS algorithm encounters difficulties in converging when large sparse linear systems are solved.

www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

Acknowledgments The results of presented paper are used in the development of project ”2D Heat Conduction Solver” is a part of EU Grant Project – TEWI [33], which supports the development of computer infrastructure and applications.

Received 8 July 2013

References

[1] Zienkiewicz O.C., Taylor R.L: *Finite Element Method.* Elsevier, 2006.

[2] Kwon Y.W., Bang H.: *The Finite Element Method using MATLAB.* CRC Press, 2000.

[3] Lewis R.W., Nithiarasu P., Seetharamu K..N.: *Fundamentals of the Finite Element Method for Heat and Fluid Flow.* Wiley, West Sussex 2004.

[4] Ocłoń P., Taler J.: *Mixed finite element and finite volume formulation – linear quadrilateral elements.* In: Encyclopedia of Thermal Stresses (R. Hetnarski, Ed.) Springer (accepted for print).

[5] Taler J., Duda P.: *Solving Direct and Inverse Heat Conduction Problems.* Springer, Berlin 2006.

[6] Taler J., Ocłoń P.: *Finite element method in steady state and transient heat conduction.* In: Encyclopedia of Thermal Stresses (R. Hetnarski, Ed.). Springer (accepted for print).

[7] Anderson J.D.: *Computational Fluid Dynamics.* McGraw-Hill Science, Boca Raton 1995.

[8] Chung T.J.: *Computational Fluid Dynamics.* Cambridge University Press, 2010.

[9] Barrett R., Berry M., Chan T.F. *et al.*: *Templates for the solution of linear systems: building blocks for iterative methods.* SIAM, Philadelphia 1999.

[10] Krizek M., Neittaanmäki P., Glowinski P.R., Korotov S.: *Conjugate Gradient Algorithms and Finite Element Methods.* Springer 2004.

[11] Łopata S., Ocłoń P.: *The analysis of gradient algorithm effectiveness — two dimensional heat transfer problem.* Arch. Thermodyn. **31**(2010), 4, 37–50.

[12] Meurant G.: *The Lanczos and Conjugate Gradient Alghoritms.* SIAM, Philadelphia 2006.

[13] Saad Y.: *Iterative Methods for Sparse Linear Systems.* SIAM, Philadelphia 1999.

[14] Incopera F.P., DeWitt D.P.: *Introduction to Heat Transfer.* Wiley, Hoboken 2006.

[15] Sonneveld P.: *CGS: A fast Lanczos-type solver for nonsymmetric linear systems.* SIAM J. Sci. Comput. **10**(1989), 1, 36–52.

[16] Parihar S.K., Wright N.T.: *Thermal contact resistance at elastomer to metal interfaces.* Int. Commun. Heat Mass Transf. **24**(1997), 8, 1083–1092.

www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

[17] Wolff E.G., Schneider D.A.: *Prediction of thermal contact resistance between polished surfaces.* Int. Commun. Heat Mass Transf. **41**(1988), 22, 3469–3482.

[18] Massé H., Arquis E.E., Delaunay D., Quilliet S., Le Bot P.H.: *Heat transfer with mechanically driven thermal contact resistance at the polymer-mold interface in injection molding of polymers.* Int. J. Heat and Mass Transf. **47**(2004), 8–9, 2015–2027.

[19] Łopata S., Ocłoń P.: *Modelling and optimizing operating conditions of heat exchanger with finned elliptical tubes.* In: Computational Modeling and Applications Fluid Dynamics (L.H. Juarez, Ed.), InTech ISBN: 978-953-51-0052-2, 327–356.

[20] Łopata S., Ocłoń P.: *Analysis of operating conditions for high performance heat exchanger with the finned elliptical tube.* Rynek Energii 5 (102)(2012), 112–124.

[21] Xiangqiao Y.: *Finite element formulation of a heat transfer problem for an axisymmetric composite structure.* Comput. Mech. **36**(2005), 76–82.

[22] Sleijpen G.L., Van der Vorst H.A., Fokkema D.R.: *BICGstab(l) and other hybrid Bi-CG methods.* Numer. Algorithms (1994)7: 75–109.

[23] Van der Vorst H.A.: Bi-CGSTAB *A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems.* SIAM J. Sci. Stat. Comput. **13**(1992), 2, 631–644.

[24] Gerub W.: *Linear Algebra.* Springer, Berlin, 1975.

[25] Taler D.: *Effect of thermal contact resistance on the heat transfer in plate finned tube heat exchangers.* In: Proc. of the Conf. Heat Exchanger Fouling and Cleaning, 1-6 Jul., Tomar, Eng. Conf. Int., New York 2007, 255–364.

[26] Taler D: *Determining thermal resistance between tube and fins in tube and plate fin heat exchangers.* In: Współczesne technologie i urządzenia energetyczne. Cracow University of Technology, Cracow 2007, 649–668.

[27] Cebula A., Taler D.: *Determining thermal contact resistance of the fin-to-tube attachment in plate fin-and tube heat exchangers.* EngOpt 2010, The 2nd Int. Conf. on Engineering Optimization, Lisbon, 6-9 Sep., 2010, Book of Abstracts, 310–320, CD-ROM proc. 1–10.

[28] Taler D., Cebula A.: *Analysis of thermal contact resistance on the heat transfer in plate fin-and-tube heat exchangers.* HTRSE Heat Transfer and Renewable Sources of Energy, Szczecin 2010, 331–340.

[29] Taler D., Cebula A.: *A new method for determination of thermal contact resistance of a fin-to-tube attachment in plate fin-and-tube heat exchangers.* Chem. Process Eng. J. **31**(2010), 839–855.

[30] Ocłoń P.: *Gradient algorithms in solving steady state and transient heat conduction problems.* In: Proc, of the ECCOMAS Special lnterest Conference, Numerical Heat Transfer 2012, CD-ROM, Wroclaw, 2012, 583–595.

[31] *MATLAB online documentation.* http://www.mathworks.com/help/matlab

[32] *ANSYS v.121, Theory reference for Mechanical ADPL and Mechanical Applications*, www1.ansys.com/customer/content/documentation/121/ans_ thry.pdf

[33] TEWI Project, http://www.tewi.ics.p.lodz.pl