The Fast Parametric Integral Equations System for Polygonal 2D Potential Problems

Andrzej Kużelewski and Eugeniusz Zieniuk

***Abstract*—Application of techniques for modelling of boundary value problems implies three conflicting requirements: obtaining high accuracy of the results, high speed of the solution and low occupancy of computers memory (RAM). It is very difficult to satisfy such requirements particularly in process of solving large-scale engineering problems. Numerical solution of these problems require computations on large matrices. Accurate results can be obtained only by using appropriate models and algorithms. In the previous papers the authors applied the parametric integral equations system (PIES) in modelling and solving boundary value problems. The first requirement was satisfied - the results were obtained with very high accuracy. This paper fulfils other requirements by novel approach to accelerate the PIES. For this purpose the fast multipole method (FMM) is included into conventional PIES, therefore the fast PIES method is obtained.**

***Index Terms*—parametric integral equations system, boundary value problems, fast multipole method.**

I. Introduction

FOR many years the authors of this paper have worked on development and application of the parametric integral equations system (PIES) to solve boundary value problems. The PIES has already been used to solve problems modelled by 2D and 3D partial differential equations, such as: Laplace [1], [2], Helmholtz [3] or Navier-Lamé [4], [5], [6]. The remarkable advantage of the PIES, compared to traditional boundary integral equation (BIE), is direct inclusion in its mathematical formalism a shape of the boundary of the considered problem [7]. The shape of the boundary is generally defined using particular functions. For this purpose, the curves (eg. Bézier, B-spline, etc.) or surface patches (such as Coons, Bézier and others) widely used in computer graphics, were applied in the PIES. The PIES is an analytical modification of traditional BIE. The above mentioned curves and surface patches are applied in modelling the shape of the boundary, instead of the contour integral as in the case of BIE. Therefore in practice, a small number of control points is required to define any shape of the boundary. It is definitely much easier than in case of element methods (such as boundary element method BEM [8] or finite element method FEM [9]). Moreover, the accuracy of solutions can be efficiently improved without interference in modelling the shape of the boundary.

The former studies focused on accuracy and efficiency of the results obtained using the PIES in comparison with the well-known algorithms such as the FEM or the BEM, as well as analytical methods. These studies confirmed the effectiveness and accuracy of the PIES in solving 2D [1], [4] and 3D [2], [5], [7] boundary value problems. The authors also proposed some extensions of the PIES method, i.e. to solve uncertainly defined problems (interval PIES [10], [11]) or to solve transient problems [12]. The former studies show, that modelling of the boundary in the PIES is definitely more simple and efficient compared to the FEM or the BEM. However efficient solving of large-scale engineering problems by the PIES is limited (similarly to conventional BEM) due to the fact, that the PIES in general produces dense and non-symmetrical matrices. That matrices are definitely smaller in sizes than in the BEM, although to compute their coefficients we need $O(N^2)$ operations and another $O(N^3)$ operations to solve obtained system using direct solvers (where N - the number of equations of the algebraic system of linear equations).

In the paper [13] the authors proposed parallel version of the PIES obtained using OpenMP, while in [14], [15], [16] they accelerated numerical computing of coefficients and solving of the system of linear equations using CUDA [17]. These approaches reduce quite significantly the time of computations, however the problem of limited resources of RAM in PC still exists. Therefore, it not allows for convenient and efficient solving of large-scale engineering problems.

In the mid of 1980s of XX century Rokhlin and Greengard proposed the fast multipole method (FMM) [18], [19], [20], which allows to reduce the CPU time in the FMM accelerated methods to $O(N)$, as well as definitely reduce occupancy of RAM. However, application of the FMM has increased the complexity of implementation of the PIES. It requires a new approach for computing coefficients and solving the system of linear equations.

The main goal of this paper is to present possibility of acceleration of the PIES for numerical solving of boundary value problems using the FMM. The main concept of the FMM is adopted from the FMM-BEM method [20]. However we think, that inclusion of the FMM into the PIES should be more effective than into the BEM. It is strictly connected with the different way of defining of BRC in the PIES and the BEM. To verify this concept, we need to include the FMM into conventional PIES. Additionally, we must modify the way of solving of the linear system of equations and apply iterative solver. Therefore, we obtain the new fast PIES method. In our preliminary studies we try to confirm high efficiency of the fast PIES on the example of polygonal 2D potential boundary problem.

II. The PIES for two-dimensional potential boundary problem

The PIES for two-dimensional Laplaces equation was obtained as the result of analytical modification of boundary integral equations (BIE), where the boundary is defined

Manuscript received March 14, 2017; revised April 14, 2017.
A. Kużelewski and E. Zieniuk are with the Faculty of Mathematics and Informatics, University of Bialystok, 15-245 Białystok, Ciołkowskiego 1M, Poland, e-mails: akuzel@ii.uwb.edu.pl, ezieniuk@ii.uwb.edu.pl.

ISBN: 978-988-14048-4-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

using boundary integrals. The main goal of modification was inclusion of the shape of boundary in mathematical formalism of BIE. The shape of boundary is defined by parametrical linear (or non-linear) functions. The PIES for polygonal domains is presented by the following formula [7]:

$$\frac{1}{2}u_l(s_k) = \sum_{j=1}^{n}\left\{\int_{s_{j-1}}^{s_j} \overline{U}^*_{lj}(s_k,s)p_j(s)ds - \right. \\ \left. - \int_{s_{j-1}}^{s_j} \overline{P}^*_{lj}(s_k,s)u_j(s)\right\}J_j, \tag{1}$$

where: $l = 1, 2, ..., n,\ s_{l-1} \le s_k \le s_l,\ s_{j-1} \le s \le s_j, J_j$ is the Jacobian, n is the number of parametric segments that creates polygonal boundary of domain in 2D. In PIES defined in the parametric reference system, s_{l-1} and s_{j-1} correspond to the beginning of l-th and j-th segment, while s_l and s_j to their ends.

Integrands $\overline{U}^*_{lj}(s_k,s)$ and $\overline{P}^*_{lj}(s_k,s)$ in (1) are presented in the following form:

$$\overline{U}^*_{lj}(s_k,s) = \frac{1}{2\pi}\ln\frac{1}{\sqrt{S_1^2+S_2^2}}, \\ \overline{P}^*_{lj}(s_k,s) = \frac{1}{2\pi}\frac{S_1 n_1^{(j)} + S_2 n_2^{(j)}}{S_1^2+S_2^2}, \tag{2}$$

where: $S_1 = S_l^{(1)}(s_k) - S_j^{(1)}(s)$ and $S_2 = S_l^{(2)}(s_k) - S_j^{(2)}(s)$. Expressions $S_n^{(i)}(s)$ $(n = j, l,\ i = 1, 2)$ are parametric linear functions

$$S_n^{(i)}(s) = a_n^{(i)}s + b_n^{(i)}, \tag{3}$$

which describe particular segments of polygonal domain. Coefficients $a_n^{(i)}$ and $b_n^{(i)}$ are obtained in an easy way by simple algorithm utilizing corner points of polygon.

Similarly to the previous researches [1], [3], [6], the pseudospectral method [22] was applied to numerical solving of the PIES (1). Boundary functions are approximated by the following series:

$$u_j(s) = \sum_{k=0}^{N} u_j^{(k)} L_j^{(k)}(s), \\ p_j(s) = \sum_{k=0}^{N} p_j^{(k)} L_j^{(k)}(s), \tag{4}$$

where $u_j^{(k)}$ and $p_j^{(k)}$ are unknown coefficients on segment j, N - is the number of terms in approximating series (4), $L_j^{(k)}(s)$ - the base functions (Lagrange polynomials) on segment j.

Finally, an algebraic version of PIES (1) is transformed into system of algebraic equations $\mathbf{Ax} = \mathbf{b}$. To solve the system, Gaussian elimination with partial pivoting and iterative refinement is used. Solutions on the boundary are obtained after solving the system.

III. The fast PIES formulation

The FMM is applied to accelerate the solving of equation (1). The main idea of the FMM is to transform calculating interactions between segments into interactions between the cells that create the hierarchical structure (tree) with the smallest cells (called leaves) containing a number of segments. Because the PIES for 2D problems is defined on parametric line s (Fig. 1), the tree structure is created on the basis of that line, unlike in the FMM-BEM [21], where whole plane is used. Example of the tree structure for the fast PIES is presented in Fig. 2.

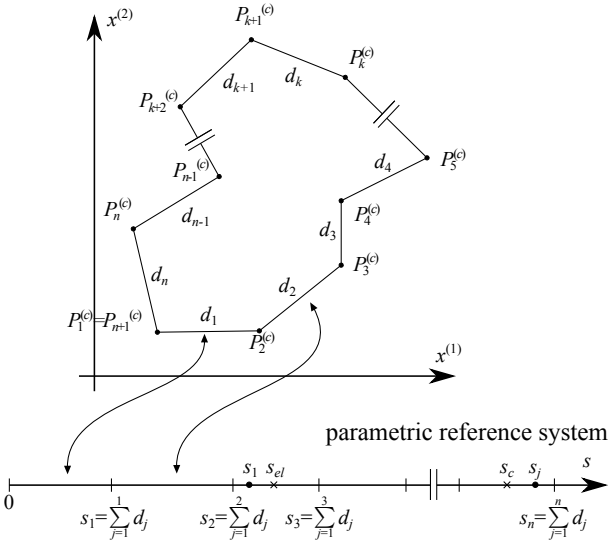

Fig. 1. Mapping of the shape of the boundary into parametric reference system

On the basis of the tree the following steps of the FMM procedure are performed: multipole expansion (calculation of moments), moment-to-moment translation, moment-to-local translation and local-to-local translation. During the computations, the complex notation is introduced, due to convienient describing of points on the plane. They are reduced to the form of $P_1^{(c)} = P_1^{(1)} + iP_1^{(2)}$ (Fig. 1), where (c) - complex, (1) - coordinate in direction $x^{(1)}$, (2) - coordinate in direction $x^{(2)}$, $i = \sqrt{-1}$.

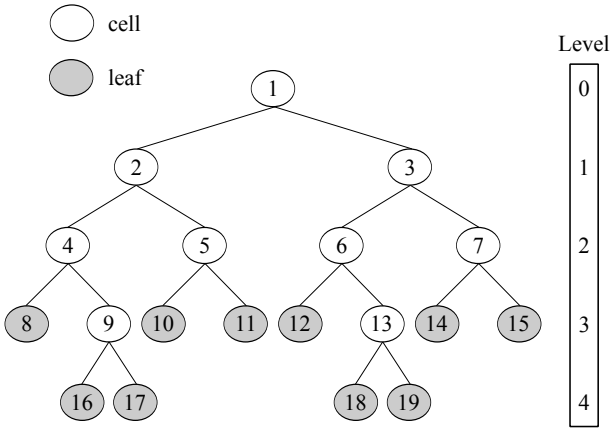

Fig. 2. Example of the tree structure for the fast PIES

1) Multipole expansion: At first, we consider the integral connected with the kernel $\overline{U}^*_{lj}(s_k,s)$. Transformations for the kernel $\overline{P}^*_{lj}(s_k,s)$ are presented in the last subsection. Considering the complex notation in the kernel (2), it can be

noted, that:

$$\overline{U}^{*}_{lj}(s_k,s) = -\frac{1}{2\pi}\ln\sqrt{S_1^2+S_2^2} = \\ = -\frac{1}{2\pi}\Re\left\{\ln\left|S_l^{(c)}-S_j^{(c)}\right|\right\} = \Re\left\{\overline{U}^{*(c)}_{lj}(s_k,s)\right\}, \tag{5}$$

where $S_l^{(c)} = S_l^{(1)} + iS_l^{(2)}$, $S_j^{(c)} = S_j^{(1)} + iS_j^{(2)}$ and $\overline{U}^{*(c)}_{lj}(s_k,s) = -\frac{1}{2\pi}\ln\left|S_l^{(c)}-S_j^{(c)}\right|$, $\Re$ - real part of complex number. The collocation point s_k is located in the segment $S_l^{(c)}$, and observation point s_j in the segment $S_j^{(c)}$.

Assuming that introduced point s_c (located in the segment $S_c^{(c)}$), which is the key element of the FMM, is close to the point s_j (Fig. 1), the kernel can be expanded about $S_c^{(c)}$ using the Taylor series expansion:

$$\overline{U}^{*(c)}_{lj}(s_k,s) = \frac{1}{2\pi}\left\{-\ln\left|S_l^{(c)}-S_c^{(c)}\right| + \right. \\ \left. + \sum_{k=1}^{\infty}\frac{(k-1)!}{\left[S_l^{(c)}-S_c^{(c)}\right]^k}\frac{\left[S-S_c^{(c)}\right]^k}{k!}\right\}. \tag{6}$$

In order to simplify the calculations we should change the base of integration s into S similarly to (3): $S = a_j s + b_j$, where $a_j = \frac{P_j^{(c)}-P_{j-1}^{(c)}}{d_{j-1}}$ and $b_j = P_{j-1}^{(c)} - \frac{(P_j^{(c)}-P_{j-1}^{(c)})s_{j-1}}{d_{j-1}}$, $j = 1,2,...,n$. Therefore, new limits of integration are $P_{j-1}^{(c)}$ (lower) and $P_j^{(c)}$ (upper). We also need to change ds by dS:

$$\frac{dS}{ds} = a_j \quad => \quad ds = \frac{dS}{a_j}.$$

Substituting the kernel $\overline{U}^{*(c)}_{lj}(s_k,s)$ into integral (1) and $W_j = a_j$, we obtain the following expression:

$$\int_{s_{j-1}}^{s_j}\overline{U}^{*(c)}_{lj}(s_k,s)p_j(s)ds = \\ = \int_{P_{j-1}^{(c)}}^{P_j^{(c)}}\frac{1}{2\pi}p_j(S)\left\{-\ln\left|S_l^{(c)}-S_c^{(c)}\right|\frac{1}{W_j} + \right. \\ \left. + \sum_{k=1}^{\infty}\frac{(k-1)!}{\left[S_l^{(c)}-S_c^{(c)}\right]^k}\frac{\left[S-S_c^{(c)}\right]^k}{k!\cdot W_j}\right\}dS = \\ = \frac{1}{2\pi}\sum_{k=0}^{\infty}U_k(S_l^{(c)},S_c^{(c)})M_k(S_c^{(c)}), \tag{7}$$

where $M_k(S_c^{(c)})$ are called moments about $S_c^{(c)}$:

$$M_k(S_c^{(c)}) = \int_{P_{j-1}^{(c)}}^{P_j^{(c)}}\frac{\left[S-S_c^{(c)}\right]^k}{k!}\frac{p_j(S)}{W_j}dS.$$

They are independent of the collocation point s_k and should be computed once only. Expressions $U_k(S_l^{(c)},S_c^{(c)})$ have the following form:

$$U_k(S_l^{(c)},S_c^{(c)}) = \begin{cases} -\ln\left|S_l^{(c)}-S_c^{(c)}\right| & \text{for } k=0 \\ \frac{(k-1)!}{\left[S_l^{(c)}-S_c^{(c)}\right]^k} & \text{for } k\geq 1 \end{cases}$$

where s_c are mid-point of leaves.

2) Moment-to-moment translation: If we want to move the point s_c to a new location s'_c (when we change the level of the tree during computations), we can use the following expression to find new moments about s'_c:

$$M_k(S_c^{'(c)}) = \int_{P_{j-1}^{(c)}}^{P_j^{(c)}}\frac{\left[S-S_c^{'(c)}\right]^k}{k!}\cdot\frac{p_j(S)}{W_j}dS. \tag{8}$$

Taking into account, that:

$$\frac{\left[S-S_c^{'(c)}\right]^k}{k!} = \frac{\left[(S-S_c^{(c)})+(S_c^{(c)}-S_c^{'(c)})\right]^k}{k!}$$

and applying the binomial formula:

$$(a+b)^n = \sum_{m=0}^{n}\left(\begin{array}{c} n \\ m \end{array}\right)a^m b^{n-m} \tag{9}$$

at last we obtain moments in the point s'_c:

$$M_k(S_c^{'(c)}) = \sum_{m=0}^{k}\frac{\left[S_c^{(c)}-S_c^{'(c)}\right]^{(k-m)}}{(k-m)!}M_m(S_c^{(c)}) \tag{10}$$

using a finite number of term in the translation.

3) Moment-to-local translation: Assuming, that the point s_{el} is close to the collocation point s_k (see Fig. 1), the equation (7) can be expanded about $S_{el}^{(c)}$ (the segment, where the point s_{el} is located) using the Taylor series expansion:

$$\int_{s_{j-1}}^{s_j}\overline{U}^{*(c)}_{lj}(s_k,s)p_j(s)ds = \\ = \frac{1}{2\pi}\sum_{l=0}^{\infty}L_l(S_{el}^{(c)},S_c^{(c)})\frac{\left[S_l^{(c)}-S_{el}^{(c)}\right]^l}{l!} \tag{11}$$

where:

$$L_0(S_l^{(c)},S_c^{(c)}) = -\ln\left|S_{el}^{(c)}-S_c^{(c)}\right|M_0(S_c^{(c)}) + \\ + \sum_{k=1}^{\infty}\frac{(k-1)!\cdot M_k(S_c^{(c)})}{\left[S_{el}^{(c)}-S_c^{(c)}\right]^k}$$

and

$$L_l(S_l^{(c)},S_c^{(c)}) = (-1)^l\sum_{k=0}^{\infty}\frac{(k+l-1)!\cdot M_k(S_c^{(c)})}{\left[S_{el}^{(c)}-S_c^{(c)}\right]^{k+l}} \text{ for } l\geq 1.$$

Points s_{el}, similarly to s_c, are mid-points of leaves. Described procedure is also called local expansion.

4) Local-to-local translation: Similarly to moment-to-moment translation, the point s_{el} can be moved to new location s'_{el} (when we change the level of the tree during computations). It is performed using the binomial formula (9) and the following transformation:

$$\sum_{l=0}^{\infty}\sum_{m=0}^{l} = \sum_{m=0}^{\infty}\sum_{l=m}^{\infty}.$$

ISBN: 978-988-14048-4-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

At last we obtain the following local-to-local translation:

$$\int_{s_{j-1}}^{s_j} \overline{U}_{lj}^{*(c)}(s_k,s)p_j(s)ds = \frac{1}{2\pi}\sum_{l=0}^{\infty}(-1)^l \cdot \cdot \left\{ \sum_{k=0}^{\infty}\sum_{m=l}^{\infty} \frac{(k+m-1)!\cdot M_k(S_c^{(c)})}{\left[S_{el}^{(c)} - S_c^{(c)}\right]^{k+m}} \cdot \cdot \frac{\left[S_{el}'^{(c)} - S_{el}^{(c)}\right]^{m-l}}{(m-l)!} \right\} \cdot \frac{\left[S_l^{(c)} - S_{el}'^{(c)}\right]^l}{l!}. \quad (12)$$

5) Translations for the kernel $\overline{P}_{lj}^*(s_k,s)$: The kernel $\overline{P}_{lj}^*(s_k,s)$ in the complex notation can be computed on the basis of the following expression:

$$\overline{P}_{lj}^{*(c)}(s_k,s) = \frac{\partial \overline{U}_{lj}^{*(c)}(s_k,s)}{\partial n^{(c)}} = n^{(c)}\frac{\partial \overline{U}_{lj}^{*(c)}(s_k,s)}{\partial S}, \quad (13)$$

where $n^{(c)} = n_1 + in_2$ - normal vector to segment j in the complex notation.

Hence

$$\overline{P}_{lj}^*(s_k,s) = \Re\left\{\overline{P}_{lj}^{*(c)}(s_k,s)\right\} = n_1\Re\left\{\frac{\partial \overline{U}_{lj}^{*(c)}(s_k,s)}{\partial S}\right\} - n_2\Im\left\{\frac{\partial \overline{U}_{lj}^{*(c)}(s_k,s)}{\partial S}\right\}, \quad (14)$$

where $\Re, \Im$ - real and imagine part of complex number.

Finally, after calculating the derivative (13) we obtain:

$$\int_{s_{j-1}}^{s_j} \overline{P}_{lj}^{*(c)}(s_k,s)u_j(s)ds = = \frac{1}{2\pi}\sum_{k=1}^{\infty} P_k(S_l^{(c)}, S_c^{(c)})N_k(S_c^{(c)}), \quad (15)$$

where

$$N_k(S_c^{(c)}) = \int_{P_{j-1}^{(c)}}^{P_j^{(c)}} \frac{\left[S - S_c^{(c)}\right]^{k-1}}{(k-1)!}\frac{u_j(S)n^{(c)}}{W_j}dS.$$

Expressions $N_k(S_c^{(c)})$, similarly to $M_k(S_c^{(c)})$ (7), are called moments about $S_c^{(c)}$ and they are independent of the collocation point s_k and should be computed once only. Expressions $P_k(S_l^{(c)}, S_c^{(c)})$ have the following form:

$$P_k(S_l^{(c)}, S_c^{(c)}) = \frac{(k-1)!}{\left[S_l^{(c)} - S_c^{(c)}\right]^k} \quad \text{for} \quad k \geq 1,$$

where s_c are mid-point of leaves.

It can be noted, that all translations described in previous subsections remain exactly the same for the kernel $\overline{P}_{lj}^{*(c)}(s_k,s)$, except for the fact that $N_0(S_c^{(c)})$. Therefore, we can directly applied them for the moments $N_k(S_c^{(c)})$.

6) The fast PIES algorithm: The fast PIES algorithm runs in several steps. The first step is to determine the structure of the tree. On the basis of 2D problems mapped into parametric reference system (Fig. 1) the structure of the tree is created. Level 0 cell covers the whole problem. It is the straight line where the PIES is defined. The length of this line is equal to the sum of all segments, which created the boundary of the problem. Level 0 parent cell is divided into two equal child cells of level 1. Then, dividing is continued in the same way for each level parent cells. The division is carried out until the number of segments in a cell is less than or equal to the predefined maximum number of segments in a cell. Each cell, which has no children, is called a leaf. The division is completed if at the highest level we obtained leaves only or predefined maximum number of levels is reached.

The next step is called upward pass. Starting from the highest number level, moments in all leaves are computed (up to k terms in the Taylor series expansion). Then, tracing the tree structure upward and using moment-to-moment translation, all moments are computed in each parent cell, up to the level 2 (Fig. 3).

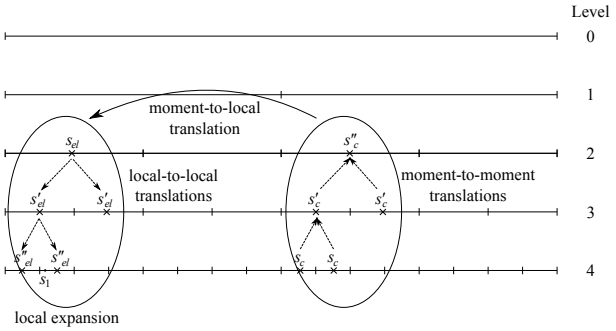

Fig. 3. Translations in the fast PIES

The next step is called downward pass. To clearly described this pass, we should define a few terms connected with cells neighbourhood. Two cells are adjacent at level i if they have common end. Two cells are well separated at level i if they are not adjacent at level i, but their parent cells are adjacent (at level $i-1$). Then the interaction list of cell K is created using the list of well-separated cells from a level i cell K. Two cells are far cells if their parent cells are not adjacent.

Coefficients of local expansion are computed starting from the level 2 and tracing the tree structure downward to all leaves (Fig. 3). Coefficients of local expansion at cell K are the sum of the contributions from the cells in the interaction list of cell K (computed using moment-to-local translation) and from all the far cells (computed using local-to-local translation). For a cell K at level 2, only moment-to-local translation is used to compute coefficients of the local expansion. At the highest number level, contributions from leaf K and its adjacent cells are computed directly, as in the conventional PIES. Finally, we obtain a vector, which is the result of multiplication matrix **A** by a vector **x** and there is no need to store the entire matrix **A** in the computer's memory. This approach requires the use of iterative solvers for system of equations (eg. GMRES).

IV. Testing example

The following example shows the accuracy and efficiency of the fast PIES in solving 2D potential problems. Intel Core i5-4590S (4 cores, 4 threads, 3.0 GHz, 6MB cache memory) with 8 GB RAM was used during tests. Both the fast and conventional PIES was compiled using g++ 5.4.0

ISBN: 978-988-14048-4-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

with -O2 optimization. Numerical tests were carried out on 64-bit Ubuntu Linux operation system (kernel 4.4.0).

The testing example concerns the problem described by Laplaces equation. The shape of the boundary is shown in Fig. 4. It is simple boundary problem, however in order to increase its complexity, we assumed that the edge looks like a gear with 256 teeth. We used 512 linear segments to model the problem using both version of the PIES. Boundary conditions are identical on each tooth and they are presented in Fig. 4 (where p Neumanns and u Dirichlets boundary conditions). On each segment we have defined the same number of collocation points (from 2 to 8) and finally have solved the system of 1024 to 4096 algebraic equations. We assumed value of GMRES convergence criterion (GMRES tolerance) equal to 1e-8 and the number of terms in Taylor series is 15.

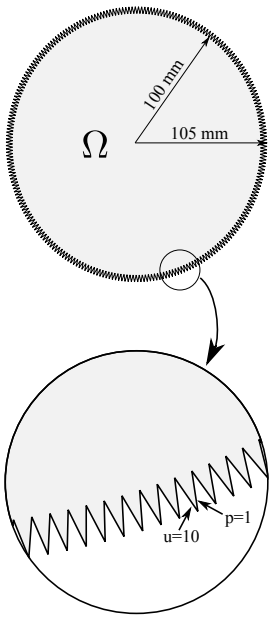

Fig. 4. The shape of modelled problem

TABLE I
THE FAST PIES VS CONVENTIONAL PIES: CPU TIME AND RAM OCCUPANCY

No. of equations	CPU time [s]		Speed-up	RAM occupancy [MB]	
	fast PIES	PIES		fast PIES	PIES
1024	0.31	3.06	9.87	1	37
2048	1.18	15.16	12.85	3	112
3072	2.11	43.49	20.61	7	228
4096	3.58	94.28	26.34	12	396

Table I presents obtained results of accelerating calculations in the fast PIES compared to conventional PIES. Growing number of equations results in small increase of computation time in the fast PIES, contrary to conventional PIES. The fast PIES also needs more than 30 times less RAM during computations.

Relative error norm L_2 between solutions obtained by the fast and conventional PIES is presented in Table II. The value of L_2 for GMRES tolerance equal 1e-8 do not exceed 0.001% in all cases. Decreasing tolerance to 1e-6 results in grow of solutions errors (in the worst case L_2 norm is less than 0.05%), however the number of iterations is reduced, as well as computation time.

TABLE II
ACCURACY OF THE FAST PIES SOLUTIONS VS VALUE OF GMRES CONVERGENCE CRITERION

No. of eq.	No. of iterations for GMRES tol.		CPU time [s] for GMRES tol.		Rel. error norm [%] for GMRES tol.	
	1e-8	1e-6	1e-8	1e-6	1e-8	1e-6
1024	38	9	0.31	0.25	8.36e-5	8.42e-4
2048	150	114	1.18	0.95	9.14e-4	9.26e-4
3072	184	144	2.11	1.72	6.82e-5	0.031
4096	291	176	3.58	2.61	2.41e-4	0.048

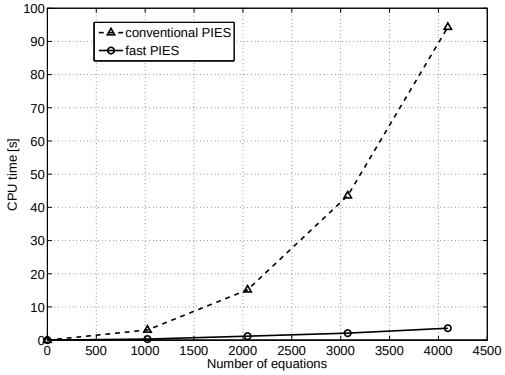

Fig. 5. Comparison of the CPU time used by the fast PIES and conventional PIES

Fig.5 presents comparison of used CPU time between conventional PIES and fast PIES. It should be noted, that the shape of plotted lines is consistent with theoretical considerations. Conventional PIES needs $O(N^2)$ operations to compute their coefficients and another $O(N^3)$ operations to solve the system by the direct solver. Application of the fast PIES is definitely more efficient.

Additionally, we solve the example using a bit old application of the FMM-BEM, which has been written in fortran by the authors of [21]. We want to find solutions comparable with the fast PIES, therefore the mesh in the FMM-BEM is composed of 1024, 2048, 3072 or 4096 linear elements (each teeth is describe by 4, 8, 12 or 16 elements). The example of discretization (for 1024 elements mesh) of a few teeth of the gear is presented in Fig. 6. Tolerance of GMRES is 1e-8 and the number of terms in Taylor series is 15.

TABLE III
THE FAST PIES VS THE FMM-BEM: CPU TIME AND RAM OCCUPANCY

No. of equations	CPU time [s]		RAM occupancy [MB]	
	fast PIES	FMM-BEM	fast PIES	FMM-BEM
1024	0.31	0.78	1	5
2048	1.18	5.01	3	11
3072	2.11	24.17	7	18
4096	3.58	52.12	12	25

Table III presents obtained results of accelerating calculations in the fast PIES compared to the FMM-BEM. Growing number of equations results in small increase of computation time in the fast PIES contrary to the FMM-BEM. However,

ISBN: 978-988-14048-4-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

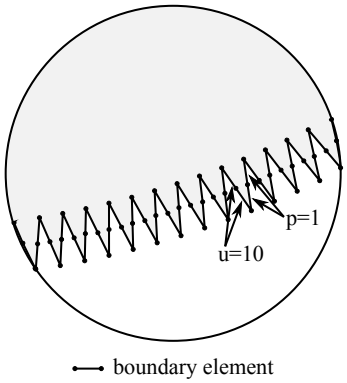

Fig. 6. Example of the FMM-BEM discretization for 1024 elements mesh

the problem of the FMM-BEM is rather connected with too big size of allocated RAM compared to really used. Therefore, we can use maximum 70 elements in a cell.

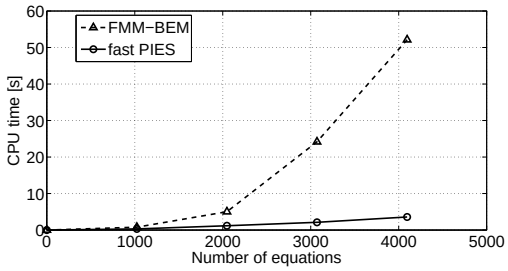

Fig. 7. Comparison of the CPU time used by the fast PIES and the FMM-BEM

Fig.7 presents comparison of used CPU time between the fast PIES and the FMM-BEM. Application of the fast PIES is more efficient than the FMM-BEM. However, we should highlight the fact, that the FMM-BEM program is a bit old. Additionally, relative error norm L_2 computed between the FMM-BEM and the fast PIES is smaller than 0.1%.

V. Conclusion

The paper presents possibility of acceleration of computation and reduction RAM occupancy for numerical solving of boundary value problems using the PIES. Verification of this concept required inclusion of the FMM into conventional PIES. The numerical example shows significant reduction of computation time of the fast PIES. The speed-up in comparison to conventional PIES increases with the size of the considered problem. We noted almost no difference between accuracy of solutions obtained by the fast PIES and conventional one. The fast PIES is also slightly faster than the FMM-BEM, while accuracy of obtained results is almost the same.

This paper is our first attempt to use the fast PIES for solving 2D potential boundary value problems. The presented technique of acceleration of computations should be extended to the problems modelled by non-linear segments. Obtained results suggest that chosen direction of studies should be continued.

Acknowledgment

The authors would like to thank prof. Naoshi Nishimura from Department of Applied Analysis and Complex Dynamical Systems Graduate School of Informatics, Kyoto University, for sharing source code of the FMM-BEM application.

References

[1] E. Zieniuk, “Modelling and effective modification of smooth boundary geometry in boundary problems using B-spline curves,” *Engineering with Computers*, Vol. 23, No. 1, pp. 39-48, 2007.
[2] E. Zieniuk and K. Szerszeń, “Triangular Bézier surface patches in modelling shape of boundary geometry for potential problems in 3D,” *Engineering with Computers*, Vol. 29, No. 4, pp. 517-527, 2013.
[3] E. Zieniuk and K. Szerszeń, “Triangular Bézier patches in modelling smooth boundary surface in exterior Helmholtz problems solved by PIES,” *Archives of Acoustics*, Vol. 34, No. 1, pp. 51-61, 2009.
[4] E. Zieniuk and A. Bołtuć, “Non-element method of solving 2D boundary problems defined on polygonal domains modeled by Navier equation,” *International Journal of Solids and Structures*, Vol. 43, No. 25-26, pp. 7939-7958, 2006.
[5] E. Zieniuk, A. Bołtuć and K. Szerszeń, “Modeling complex homogeneous regions using surface patches and reliability verification for Navier-Lamé boundary problems,” *Proceedings of The 2012 International Conference on Scientific Computing WORLDCOMP 2012*, pp. 166-172, 2012.
[6] E. Zieniuk, A. Bołtuć and A. Kużelewski, “Algorithms of identification of multi-connected boundary geometry and material parameters in problems described by Navier-Lamé equation using the PIES,” in *Advances in Information Processing and Protection: International Advanced Computer Systems Conference 2006*, pp. 409-418, 2006.
[7] E. Zieniuk, “Computational method PIES for solving boundary value problems (in polish),” *PWN*, Warszawa, 2013.
[8] C. A. Brebbia, J. C. F. Telles and L. C. Wrobel, “Boundary element techniques, theory and applications in engineering,” *Springer-Verlag*, New York, 1984.
[9] O. C. Zienkiewicz “The Finite Element Methods,” *McGraw-Hill*, London, 1977.
[10] E. Zieniuk and M. Kapturczak and A. Kużelewski, “Concept of modeling uncertainly defined shape of the boundary in two-dimensional boundary value problems and verification of its reliability,” *Applied Mathematical Modelling*, Vol. 40, No. 23-24, pp. 10274-10285, 2016.
[11] E. Zieniuk, A. Kużelewski and M. Kapturczak, “The influence of interval arithmetic on the shape of uncertainly defined domains modelled by closed curves,” *Computational & Applied Mathematics*, doi:10.1007/s40314-016-0382-0, to be published.
[12] E. Zieniuk, D. Sawicki and A. Bołtuć, “Parametric integral equations systems in 2D transient heat conduction analysis,” *International Journal of Heat and Mass Transfer*, Vol. 78, pp. 571-587, 2014.
[13] A. Kużelewski and E. Zieniuk, “OpenMP for 3D potential boundary value problems solved by PIES,” *AIP Conference Proceedings 2016: 13th International Conference of Numerical Analysis and Applied Mathematics ICNAAM 2015*, No. 480098, 2015.
[14] A. Kużelewski and E. Zieniuk, “GPU-based acceleration of computations in elasticity problems solving by parametric integral equations system,” *Advances in Engineering Software*, Vol. 79, pp. 27-35, 2015.
[15] A. Kużelewski, E. Zieniuk and M. Kapturczak, “Acceleration of integration in parametric integral equations system using CUDA,” *Computers & Structures*, Vol. 152, pp. 113-124, 2015.
[16] A. Kużelewski, E. Zieniuk and A. Bołtuć, “Application of CUDA for Acceleration of Calculations in Boundary Value Problems Solving Using PIES,”, *Lecture Notes in Computer Science: Parallel Processing and Applied Mathematics PPAM 2013, PT II*, pp. 322-331, 2014.
[17] “CUDA C Programming Guide,” *http://docs.nvidia.com/cuda/cuda-c-programming-guide/* [access date: 03 March 2017].
[18] V. Rokhlin, “Rapid solution of integral equations of classical potential theory,” *Journal of Computational Physics*, Vol. 60, No. 2, pp. 187-207, 1985.
[19] L. F. Greengard and V. Rokhlin, “A fast algorithm for particle simulations,” *Journal of Computational Physics*, Vol. 73, No. 2, pp. 325-348, 1987.
[20] L. F. Greengard, “The rapid evaluation of potential fields in particle systems,” *The MIT Press*, Cambridge, 1988.
[21] Y. J. Liu and N. Nishimura, “The fast multipole boundary element method for potential problems: A tutorial,” *Engineering Analysis with Boundary Elements*, Vol. 30, No. 5, pp. 371-381, 2006.
[22] D. Gottlieb and S. A. Orszag, “Numerical Analysis of Spectral Methods: Theory and Applications,” *SIAM*, Philadelphia, 1977.