Numerical Algorithms (2021) 88:1025–1050
https://doi.org/10.1007/s11075-020-01066-6

ORIGINAL PAPER

The FMM accelerated PIES with the modified binary tree in solving potential problems for the domains with curvilinear boundaries

Andrzej Kużelewski[1] · **Eugeniusz Zieniuk**[1]

Received: 21 July 2020 / Accepted: 29 December 2020 / Published online: 31 March 2021

Abstract
The paper presents an accelerating of solving potential boundary value problems (BVPs) with curvilinear boundaries by modified parametric integral equations system (PIES). The fast multipole method (FMM) known from the literature was included into modified PIES. To consider complex curvilinear shapes of a boundary, the modification of a binary tree used by the FMM is proposed. The FMM combined with the PIES, called the fast PIES, also allows a significant reduction of random access memory (RAM) utilization. Therefore, it is possible to solve complex engineering problems on a standard personal computer (PC). The proposed algorithm is based on the modified PIES and allows for obtaining accurate solutions of complex BVPs described by the curvilinear boundary at a reasonable time on the PC.

Keywords Fast parametric integral equations system · Fast multipole method · Boundary value problems

1 Introduction

It is a known fact that to obtain accurate results of modelling and solving boundary value problems (BVPs), the appropriate method should be applied. Nowadays, some well-established numerical methods, sometimes called classical or conventional, have professional implementations and they are well known in the industry.

✉ Andrzej Kużelewski
akuzel@ii.uwb.edu.pl

Eugeniusz Zieniuk
ezieniuk@ii.uwb.edu.pl

[1] Institute of Computer Science, University of Bialystok, Ciolkowskiego 1M, 15-245, Bialystok, Poland

For example, ANSYS and COMSOL Multiphysics use the finite element method (FEM) [1–3], while BEASY, the boundary element method (BEM) [4–6].

However, classical methods have some disadvantages, e.g. necessity of the boundary (or domain) discretization and pyroper selection of the shape functions. The accuracy of solutions obtained by the conventional FEM is strictly connected with the size of a mesh obtained in process of the domain discretization, thus with a number and a type of finite elements. A growing number of finite elements requires longer computations time and greater random access memory (RAM) utilization. It is noticeable, particularly in large-scale problems. The FEM usually produces large sparse matrices and uses iterative solvers. Conventional BEM (which uses collocation method) discretizes only the boundary of the problem. However, it produces dense and non-symmetric matrices and usually uses direct solvers. Thus, the process of computing coefficients of the BEM matrices performs $O(N^2)$ operations and direct solver uses another $O(N^3)$ operations (N—the number of equations of the system of algebraic equations). Therefore, many researchers work on new approaches which are free from the weakness of classical methods. A wide group of such methods includes, among others, FEM-BEM hybrids [7, 8], meshless methods [9, 10], isogeometric analysis (IgA) [11, 12], the virtual element method (VEM) [13, 14] or the parametric integral equations systems (PIES) [15].

One of the new approaches, isogeometric analysis, uses the FEM or the BEM extended by the set of polynomial basis functions, called Non-Uniform Rational B-spline (NURBS). It allows the exact representation of a boundary using computer-aided design (CAD) tools. Also, it gives higher efficiency of automated meshing process [11]. Another approach, the VEM, uses trial functions with an unknown degree of freedom within the interior of the polygonal domain. The method does not require an extension of interpolation functions to the interior of the element [13]. It is efficient in solving problems with polygonal meshes. However, the computational effort of the method is high due to complex procedures used by the VEM.

The method called PIES proposed and applied in solving different kinds of BVPs by the authors of this paper is still developed. The PIES has already been used to solve 2D and 3D problems described by Laplace's [15], Helmholtz [16] or Navier-Lamé equations [17]. The remarkable advantage of the method is direct inclusion of the shape of the boundary into the mathematical formalism of the PIES. It implies lack of discretization either boundary or domain contrary to the BEM and the FEM. Some parametric functions known from computer graphics are applied to describe the shape of the boundary in the PIES (among others, Bézier, B-spline curves and Coons, Bézier surface patches). Another advantage is that these functions require only a small number of control points to define any shape of the boundary, which is more efficient than the discretization process. At last, to improve the accuracy of solutions, remodelling of the shape of the boundary is unnecessary [15] contrary to the BEM and the FEM where the process of discretization should be repeated.

The former studies confirmed the accuracy of the PIES in solving 2D and 3D engineering problems in comparison to the analytical and classical numerical methods, e.g. [15–17]. The authors of this paper also have worked on the extension of the PIES method to solve uncertainly defined problems [18, 19]. However, similarly to the classic BEM, conventional PIES produces dense and non-symmetric matrices,

therefore solving of a complex or large-scale engineering problems needs a huge amount of the RAM and time-consuming computations.

Acceleration of numerical computations is usually performed using parallelization methods on the computational clusters [20] or multiprocessor machines. The most popular methods are the message passing interface (MPI) and open multi-processing (OpenMP) (e.g. applied in [21, 22]). Another very common method is the application of graphics processing unit (GPU) for numerical calculations (by CUDA or OpenCL) [23, 24]. In previous studies, the authors of this paper applied above-mentioned techniques to accelerate the PIES. OpenMP technology is used in the research described in [25], while the acceleration of numerical computations of integrals in the PIES by CUDA is presented in [26, 27]. These approaches significantly reduce the time of computations. However, the problem of huge utilization of the RAM in PC is still not solved. Complex or large-scale engineering problems cannot be solved in standard PC.

In the mid of 1980s, Rokhlin and Greengard proposed the fast multipole method (FMM) [28–30]. The method significantly reduces the computation time of the numerical problem to order $O(NlogN)$. However, the most important is huge reducing the RAM utilization. There are efficient implementations of the FMM for solving potential problems [31, 32], also by the BEM [33, 34]. Therefore, the authors of this paper have modified the PIES to include the FMM, and the fast PIES was obtained [35]. Application of the FMM increased the difficulty of implementation of the method. However, the proposed approach gives accurate solutions in a short time for problems with a quite simple shape of the boundary. Unfortunately, the application of a classic binary tree in the FMM may result in obtaining incorrect solutions for complex shapes of a boundary (e.g. heat sink) in the fast PIES. It is connected with the way of defining the boundary in the PIES—the 2D problem is mapped into 1D parametric reference system. The binary tree is built based on the 1D system, while neighbourhood in the FMM is based on the physical coordinates. Therefore, the modification of the binary tree and some changes in the FMM algorithm is required to consider the complex shapes of a boundary.

The main goal of this paper is to present the FMM accelerated PIES (called the fast PIES) applied for numerical solving of complex curvilinear 2D potential BVPs. The FMM algorithm and the binary tree are modified to allow solving the problems with the complex shape of a boundary. The fast PIES reduces memory utilization of PC and accelerates numerical solving of BVPs. The efficiency and accuracy of the fast PIES are tested on curvilinear 2D potential BVPs.

2 Conventional PIES for curvilinear 2D potential BVPs

Conventional PIES for 2D potential problems was obtained as the result of analytical modification of boundary integral equation (BIE) [15]. The modification includes the shape of boundary into mathematical formalism of BIE. The shape of boundary is defined on 1D parametric line s by parametric linear or curvilinear functions

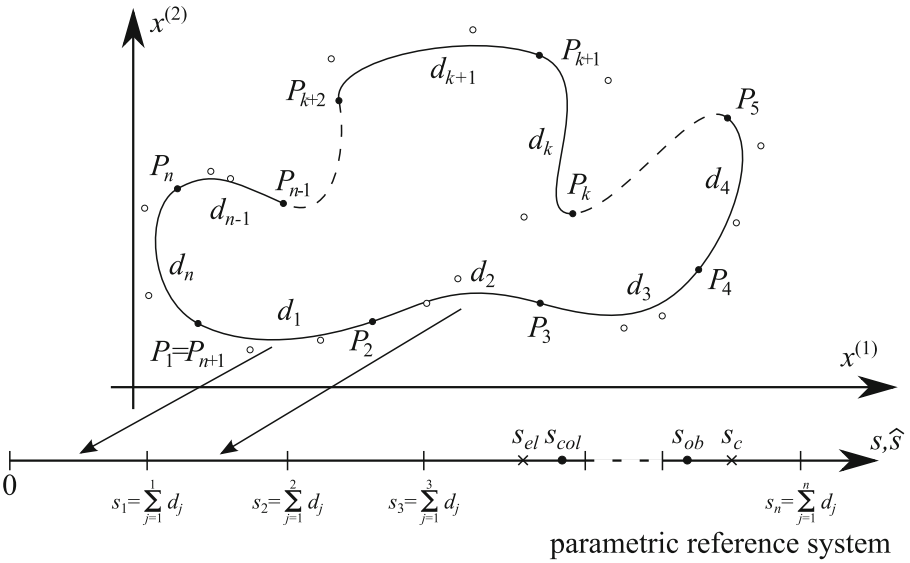

Fig. 1 Defining the shape of the boundary in the PIES on a straight line in parametric reference system $s, \widehat{s}$

(presented in Fig. 1). The PIES for the potential problem with smooth boundary is presented by the following formula [15]:

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

where n is the number of parametric segments that creates boundary of domain in 2D, s_{l-1} and s_{j-1} correspond to the beginning of lth and jth segments, while s_l and s_j to their ends and $J_j(s)$ is the Jacobian.

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

$$\widehat{U}^*_{lj}(\widehat{s}, s) = \frac{1}{2\pi} \ln \frac{1}{\sqrt{\left(S^{(1)}\right)^2 + \left(S^{(2)}\right)^2}},$$
$$\widehat{P}^*_{lj}(\widehat{s}, s) = \frac{1}{2\pi} \frac{S^{(1)} \cdot n^{(1)}_j(s) + S^{(2)} \cdot n^{(2)}_j(s)}{\left(S^{(1)}\right)^2 + \left(S^{(2)}\right)^2}, \tag{2}$$

where $S^{(1)} = S^{(1)}_l(\widehat{s}) - S^{(1)}_j(s)$ and $S^{(2)} = S^{(2)}_l(\widehat{s}) - S^{(2)}_j(s)$ and $n^{(j)}_k(s)$, $(k = 1, 2)$ are the components of normal vector to segment j.

Expressions $S^{(i)}_k(s_n)$ ($i = \{1, 2\}$, $k = \{j, l\}$ $s_n = \{\widehat{s}, s\}$) are parametric curves, which define particular segments of curvilinear boundary of the problem. In this paper the following Bézier curves of the third degree are applied:

$$S^{(i)}_k(s_n) = a^{(i)}_k s^3_n + b^{(i)}_k s^2_n + c^{(i)}_k n + d^{(i)}_k, \quad 0 \le s_n \le 1, \tag{3}$$

where i is the direction of coordinates in 2D Cartesian reference system (see in Fig. 1), coefficients $a_k^{(i)}, b_k^{(i)}, c_k^{(i)}, d_k^{(i)}$ describing particular segments of the boundary are computed using the points which define the curves (see in Fig. 2):

$$a_k^{(i)} = P_{k+1}^{(i)} - 3P_{kb}^{(i)} + 3P_{ka}^{(i)} - P_k^{(i)},\ b_k^{(i)} = 3(P_{kb}^{(i)} - 2P_{ka}^{(i)} + P_k^{(i)}),$$
$$c_k^{(i)} = 3(P_{ka}^{(i)} - P_k^{(i)}),\ d_k^{(i)} = P_k^{(i)}.$$

Boundary functions $u_j(s)$ and $p_j(s)$ in (1) are approximated by the following series:

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

where $u_j^{(k)}$ and $p_j^{(k)}$ are unknown or given values of boundary functions in defined points of the segment j, N is the number of terms in approximating series (4), which approximated boundary functions on the segment j, and $L_j^{(k)}(s)$ is the base functions (Lagrange polynomials) on segment j.

In previous researches, the collocation method [36] was applied to solve the PIES (1); thus, the system of algebraic equations **Ax=b** was obtained. Solutions on the boundary are obtained directly after solving the system. We try to use different methods to solve the system. Own implementation of Gaussian elimination method with partial pivoting was too slow; therefore, we applied LU decomposition with partial pivoting and row interchanges available in LAPACK library [37] (presented in [35, 38]). The CPU time of solving the system of 4096 equations decreases from 60 to 4 s. However, it did not affect the huge RAM utilization of the method.

In the presented research, we also applied the generalized minimal residual iterative solver (GMRES) [39] to the PIES. It is well known that iterative solvers require substantially less memory and computational effort compared to direct solvers for

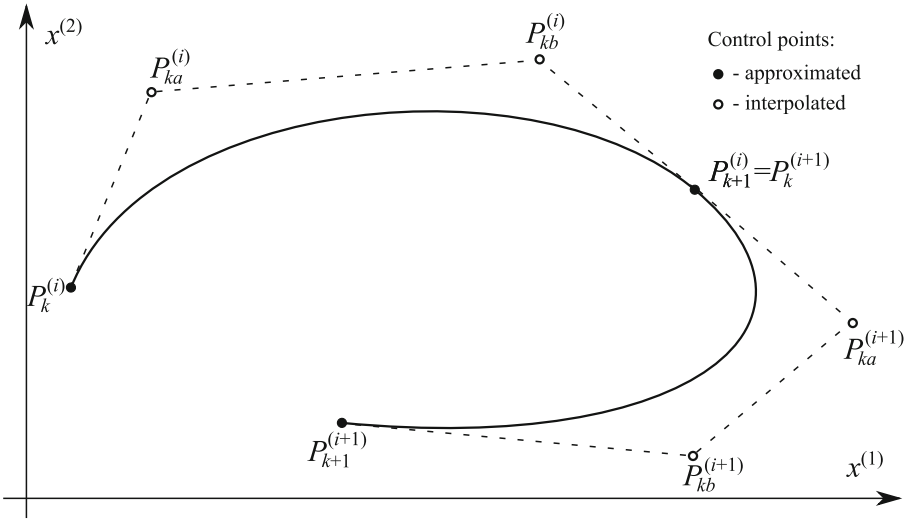

Fig. 2 Bézier curve of the third degree used to define segment of the boundary in the PIES

large linear systems. Described below the fast PIES uses GMRES and application of the same solver should be better to compare the conventional PIES and the fast PIES to quantify the obtained improvements from the use of a modified binary tree in the FMM.

3 The PIES kernels modification

First of all, to accelerate the process of solving equation (1) the authors of this paper modified the PIES kernels [35]. The modification of the PIES was necessary due to the need of calculation of subsequent derivatives of kernels (2) for the Taylor series approximation used by the FMM. It should be noted that the function $\sqrt{\left(S^{(1)}\right)^2+\left(S^{(2)}\right)^2}$ under the logarithm in the kernel $\widehat{U}^*_{lj}(\widehat{s},s)$ has the form similar to magnitude (absolute value) of a complex function. Therefore,

$$\begin{aligned}
\widehat{U}^*_{lj}(\widehat{s},s) &= \tfrac{1}{2\pi}\ln\frac{1}{\sqrt{\left(S^{(1)}\right)^2+\left(S^{(2)}\right)^2}} = -\tfrac{1}{2\pi}\ln\sqrt{\left(S^{(1)}\right)^2+\left(S^{(2)}\right)^2} = \\
&= -\tfrac{1}{2\pi}\ln\sqrt{\left(S^{(1)}_l(\widehat{s})-S^{(1)}_j(s)\right)^2+\left(S^{(2)}_l(\widehat{s})-S^{(2)}_j(s)\right)^2} = \\
&= \Re\left\{-\tfrac{1}{2\pi}\ln\left[\left(S^{(1)}_l(\widehat{s})-S^{(1)}_j(s)\right)+i\left(S^{(2)}_l(\widehat{s})-S^{(2)}_j(s)\right)\right]\right\} = \\
&= \Re\left\{-\tfrac{1}{2\pi}\ln\left[\left(S^{(1)}_l(\widehat{s})+iS^{(2)}_l(\widehat{s})\right)-\left(S^{(1)}_j(s)+iS^{(2)}_j(s)\right)\right]\right\} = \\
&= \Re\left\{-\tfrac{1}{2\pi}\ln\left[S^{(c)}_l(\widehat{s})-iS^{(c)}_j(s)\right]\right\} = \Re\left\{-\tfrac{1}{2\pi}\ln(\widehat{\tau}-\tau)\right\},
\end{aligned}$$

where (c) means complex variable, the imaginary unit (an indeterminate) $i=\sqrt{-1}$ and $\Re$ is the real part of complex number. For simplification, the parametric functions (3) in modified PIES are described as $\widehat{\tau}$ and τ:

$$\widehat{\tau}=S^{(c)}_l(\widehat{s})=S^{(1)}_l(\widehat{s})+iS^{(2)}_l(\widehat{s}),\quad \tau=S^{(c)}_j(s)=S^{(1)}_j(s)+iS^{(2)}_j(s).$$

The collocation point $\widehat{s}$ is on the segment $\widehat{\tau}$ while the observation point s on τ.

Similarly to the kernel $\widehat{U}^*_{lj}(\widehat{s},s)$, the kernel $\widehat{P}^*_{lj}(\widehat{s},s)$ can be described using complex notation:

$$\begin{aligned}
\widehat{P}^*_{lj}(\widehat{s},s) &= \frac{\partial\widehat{U}^*_{lj}(\widehat{s},s)}{\partial n} = \Re\left\{\frac{\partial\widehat{U}^{*(c)}_{lj}(\widehat{\tau},\tau)}{\partial n^{(c)}}\right\} = \\
&= \Re\left\{n^{(c)}\frac{\partial\widehat{U}^{*(c)}_{lj}(\widehat{\tau},\tau)}{\partial\tau}\right\} = \Re\left\{\tfrac{1}{2\pi}\tfrac{n^{(c)}}{\widehat{\tau}-\tau}\right\}
\end{aligned}$$

where $n^{(c)}=n^{(1)}+in^{(2)}$ is the complex notation of normal vector to the curve, which creates segment j, while $\frac{\partial\widehat{U}^{*(c)}_{lj}(\widehat{\tau},\tau)}{\partial\tau}=\frac{1}{2\pi}\frac{1}{\widehat{\tau}-\tau}$.

Therefore, the complex form of kernels (2) (called as modified kernels) has the following form:

$$\begin{aligned} \widehat{U}_{lj}^{*(c)}(\widehat{\tau},\tau) &= -\frac{1}{2\pi}\ln(\widehat{\tau}-\tau), \\ \widehat{P}_{lj}^{*(c)}(\widehat{\tau},\tau) &= \frac{1}{2\pi}\frac{n^{(c)}}{\widehat{\tau}-\tau}. \end{aligned} \tag{5}$$

The FMM can be include into these kernels in easy and efficient way.

Finally, the PIES with modified kernels have the following form:

$$\begin{aligned} \frac{1}{2}u_l(\widehat{s}) = &\sum_{j=1}^{n}\Re\left\{\int_{s_{j-1}}^{s_j}\widehat{U}_{lj}^{*(c)}(\widehat{\tau},\tau)\,p_j(s)\,J_j(s)\,ds\right\} - \\ &-\sum_{j=1}^{n}\Re\left\{\int_{s_{j-1}}^{s_j}\widehat{P}_{lj}^{*(c)}(\widehat{\tau},\tau)\,u_j(s)\,J_j(s)ds\right\}. \end{aligned} \tag{6}$$

4 Implementation of the FMM into modified PIES

The main idea of the FMM is to transform interactions between some elements into interactions between the cells that create the hierarchical structure (tree) with the smallest cells (called leaves) containing some elements. In classic FMM, the binary tree for 1D, quad-tree for 2D and octa-tree for 3D problems are applied. In this paper, 2D problems are discussed; hence, the way of constructing quad-tree for classic FMM implementation is presented in Fig. 3.

The level 0 cell is a square surrounding the entire domain of the problem (Fig. 3a). This cell (called a parent) is divided into four identical squares—cells of the level 1. Only the cell crossing the boundary of the problem is considered. This cell is called a child. Generally, the parent cell of level k ($k \geq 0$) is divided into children cells of a level $k+1$. The division is performed until the assumed maximum level of k has been reached or the predetermined number of elements are inside a cell. A childless cell is called a leaf. Figure 3b presents a quad-tree obtained in the described way. This kind of tree is used, among others, by the FMM accelerated BEM (FMBEM) [33].

4.1 Modification of binary tree in the PIES

The PIES for 2D potential problems is defined in 1D parametric reference system $s, \widehat{s}$ (presented in Fig. 1). Therefore, the binary tree structure for the PIES created in the parametric space can be used. It is more simple than quad-tree in the FMBEM. In the classic FMM, the binary tree for 1D problems has the form presented in Fig. 4b.

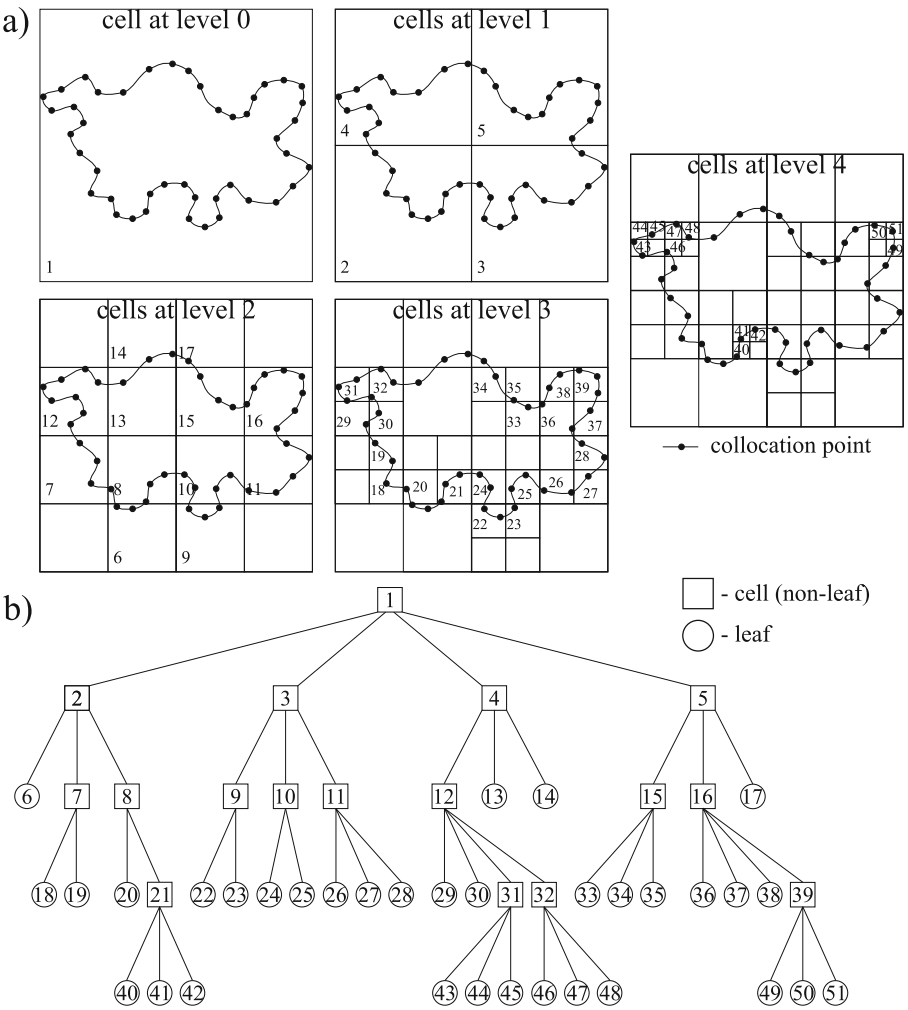

Fig. 3 **a** Constructing the FMM tree in the FMBEM, **b** structure of obtained quad-tree for 2D problem

However, the FMM definition based on the physical coordinates and some cells which are far away in the parametric space might be close in physical coordinates (e.g. the first and the last parametric segment which are neighbours in the closed curve). Therefore, the modified binary tree in the form presented in Fig. 4c is proposed [40]. In each level, the first and the last cells are treated as neighbours.

Similarly to mentioned quad-tree, a level 0 cell covers the entire boundary of the problem (presented in Fig. 4a). It is the parent of two identical children cells of level 1 obtained as a result of dividing level 0 cell. Generally, the parent cell of level k ($l \geq 0$) is divided into two level $k+1$ children cells. The division is performed until a predetermined number of segments are inside a cell or the assumed maximum level of k has been reached.

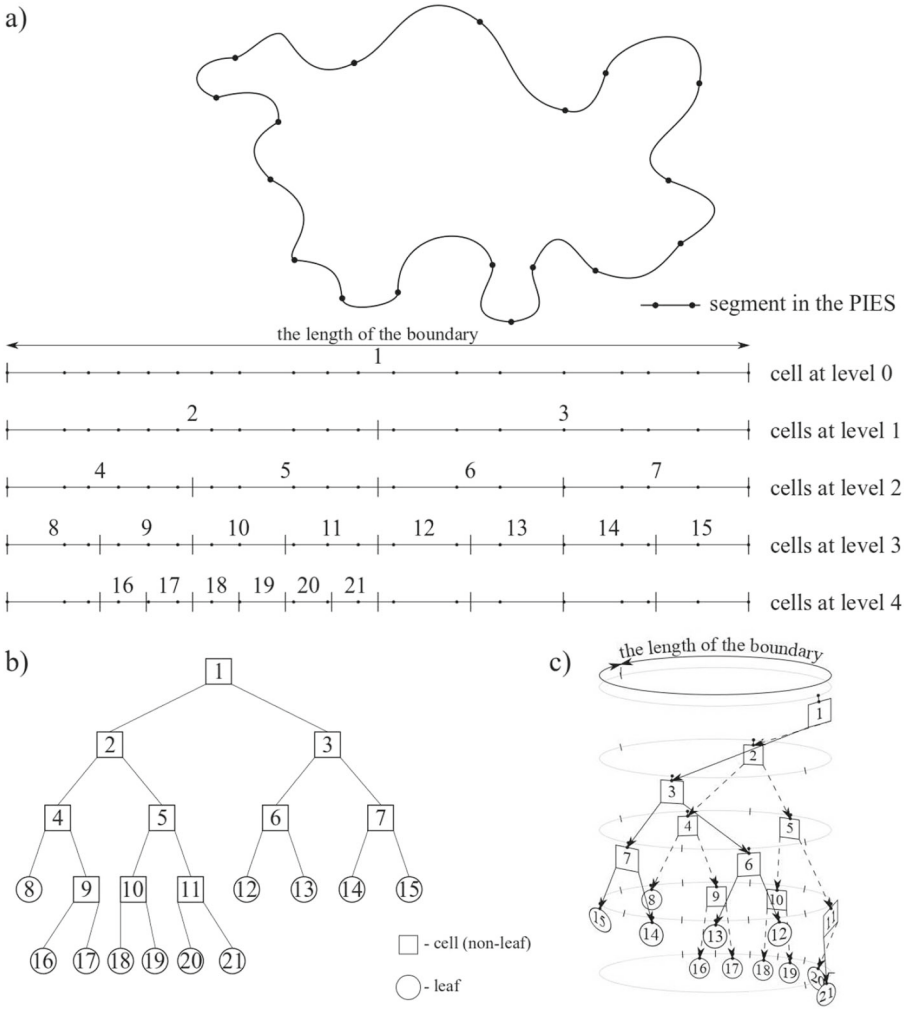

Fig. 4 **a** Constructing the FMM tree in the PIES, **b** classic binary tree in the FMM for the 1D problem, **c** modified binary tree in the FMM accelerated PIES for the 2D problem

4.2 The fast multipole procedure for modified PIES

The fast multipole procedure is composed of two main steps: making calculations of multipole moments during tracing tree structure upward (called upward pass) and making calculations of local expansions to the smallest cells tracing the tree structure downward (called downward pass).

The first is upward pass. First of all, the point s_c called mid-point of a leaf (see in Fig. 1) is introduced. This point is the key element of the FMM. The point s_c corresponds to complex point τ_c. Assuming that s_c is close to the observation point

s_{ob} (presented in Fig. 1), kernels $\widehat{U}_{lj}^{*(c)}(\widehat{\tau},\tau)$ and $\widehat{P}_{lj}^{*(c)}(\widehat{\tau},\tau)$ can be expanded about $\widehat{\tau}$ using the Taylor series expansion:

$$\begin{aligned}\widehat{U}_{lj}^{*(c)}(\widehat{\tau},\tau) &= \frac{1}{2\pi}\left\{-\ln(\widehat{\tau}-\tau_c)+\sum_{k=1}^{\infty}\frac{(k-1)!}{(\widehat{\tau}-\tau_c)^k}\frac{(\tau-\tau_c)^k}{k!}\right\},\\ \widehat{P}_{lj}^{*(c)}(\widehat{\tau},\tau) &= \frac{1}{2\pi}\left\{\sum_{k=1}^{\infty}\frac{(k-1)!}{(\widehat{\tau}-\tau_c)^k}\frac{(\tau-\tau_c)^{(k-1)}}{(k-1)!}\right\}.\end{aligned} \tag{7}$$

Substituting kernels (7) into integrals (6), the following expressions are obtained:

$$\begin{aligned}\int_{s_{j-1}}^{s_j}\widehat{U}_{lj}^{*(c)}(\widehat{\tau},\tau)\,p_j(s)\,J_j(s)ds &= \frac{1}{2\pi}\sum_{k=0}^{\infty}U_k(\widehat{\tau},\tau_c)\,M_k(\tau_c),\\ \int_{s_{j-1}}^{s_j}\widehat{P}_{lj}^{*(c)}(\widehat{\tau},\tau)\,u_j(s)\,J_j(s)ds &= \frac{1}{2\pi}\sum_{k=1}^{\infty}P_k(\widehat{\tau},\tau_c)\,N_k(\tau_c),\end{aligned} \tag{8}$$

where

$$\begin{aligned}M_k(\tau_c) &= \int_{s_{j-1}}^{s_j}\frac{(\tau-\tau_c)^k}{k!}p_j(s)\,J_j(s)ds,\\ N_k(\tau_c) &= \int_{s_{j-1}}^{s_j}\frac{(\tau-\tau_c)^{k-1}}{(k-1)!}n^{(c)}u_j(s)\,J_j(s)\,ds,\end{aligned} \tag{9}$$

and

$$\begin{aligned}U_k(\widehat{\tau},\tau_c) &= \begin{cases}-\ln(\widehat{\tau}-\tau_c) & \text{for } k=0\\ \frac{(k-1)!}{(\widehat{\tau}-\tau)^k} & \text{for} k\geq 1\end{cases}.\\ P_k(\widehat{\tau},\tau_c) &= \frac{(k-1)!}{(\widehat{\tau}-\tau_c)^k}\ \text{ for }\ k\geq 1.\end{aligned}$$

$M_k(\tau_c)$ and $N_k(\tau_c)$ are called moments about τ_c [33] and they should be computed only once. Moments are independent from $\widehat{\tau}$ as well as from $\widehat{s}$. Computation of moments is often called *moment expansion*.

The moment expansion is used to calculate moments in the leaves only. However, the FMM procedure assumes, that the point s_c can be moved to a new location s_c' in the bigger cell (presented in Fig. 5). The change of the tree level does not require recalculation of moments in points τ_c'.

Substituting τ_c' directly to formulas (9), the following expressions are obtained:

$$M_k\left(\tau_c'\right) = \int_{s_{j-1}}^{s_j} \frac{\left(\tau - \tau_c'\right)^k}{k!} p_j(s) J_j(s) ds,$$
$$N_k\left(\tau_c'\right) = \int_{s_{j-1}}^{s_j} \frac{\left(\tau - \tau_c'\right)^{k-1}}{(k-1)!} n^{(c)} u_j(s) J_j(s) ds. \tag{10}$$

Considering that:

$$\frac{\left(\tau - \tau_c'\right)^k}{k!} = \frac{\left[(\tau - \tau_c) + \left(\tau_c - \tau_c'\right)\right]^k}{k!}$$

and using the following binomial formula:

$$(a+b)^n = \sum_{m=0}^{n} \binom{n}{m} a^m b^{n-m} \tag{11}$$

finally we obtain moments about τ_c':

$$M_k(\tau_c') = \sum_{m=0}^{k} \frac{\left(\tau_c - \tau_c'\right)^{(k-m)}}{(k-m)!} M_m(\tau_c),$$
$$N_k(\tau_c') = \sum_{m=0}^{k} \frac{\left(\tau_c - \tau_c'\right)^{(k-m)}}{(k-m)!} N_m(\tau_c), \tag{12}$$

using a finite number of terms in the translation. The procedure of calculation of the moments in points τ_c' is called *moment-to-moment translation*. The process described above is called *the upward pass*, due to the direction of levels change in the tree structure during computations (see in Fig. 5).

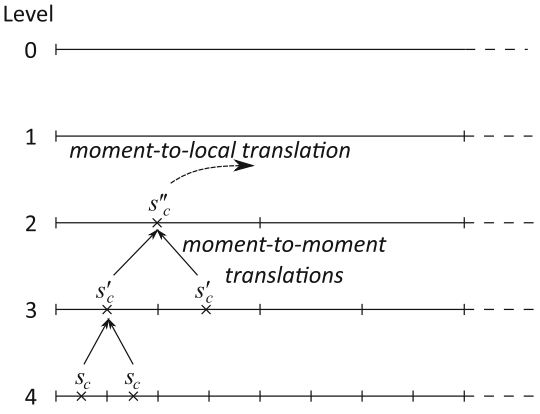

Fig. 5 The upward pass translations

Similarly to the upward pass, a new point s_{el} (corresponding to complex point τ_{el}) close to the collocation point s_{col} is introduced (see in Fig. 1). Therefore, (8) can be expanded about τ_{el} using the Taylor series expansion:

$$\begin{aligned}
\int_{s_{j-1}}^{s_j} \widehat{U}_{lj}^{*(c)}(\widehat{\tau},\tau)\, p_j(s)\, J_j(s) ds &= \frac{1}{2\pi} \sum_{l=0}^{\infty} L_l^U(\tau_{el},\tau_c) \frac{(\widehat{\tau}-\tau_{el})^l}{l!} \\
\int_{s_{j-1}}^{s_j} \widehat{P}_{lj}^{*(c)}(\widehat{\tau},\tau)\, u_j(s)\, J_j(s) ds &= \frac{1}{2\pi} \sum_{l=0}^{\infty} L_l^P(\tau_{el},\tau_c) \frac{(\widehat{\tau}-\tau_{el})^l}{l!}
\end{aligned} \tag{13}$$

where

$$\begin{aligned}
L_0^U(\tau_{el},\tau_c) &= -\ln(\tau_{el}-\tau_c) M_0(\tau_c) + \sum_{k=1}^{\infty} \frac{(k-1)!\cdot M_k(\tau_c)}{(\tau_{el}-\tau_c)^k}, \\
L_l^U(\tau_{el},\tau_c) &= (-1)^l \sum_{k=0}^{\infty} \frac{(k+l-1)!\cdot M_k(\tau_c)}{(\tau_{el}-\tau_c)^{k+l}} \quad \text{for} \quad l \geq 1
\end{aligned}$$

and

$$L_l^P(\tau_{el},\tau_c) = (-1)^l \sum_{k=1}^{\infty} \frac{(k+l-1)!\cdot N_k(\tau_c)}{(\tau_{el}-\tau_c)^{k+l}} \quad \text{for} \quad l \geq 0.$$

Similarly to s_c, points s_{el} are mid-points of leaves and they correspond to complex points τ_{el}. This procedure is called *moment-to-local translation*.

Next part of computations is connected with the downward pass. Tracing the tree structure downward we need to change the tree levels again. Therefore, the point s_{el} can be moved to a new location s'_{el} (see in Fig. 6). The procedure is called *local-to-local translation*.

Applying the binomial formula (11) and the following transformation:

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

the following forms of integrals (1) are finally obtained:

$$\begin{aligned}
\int_{s_{j-1}}^{s_j} \widehat{U}_{lj}^{*}(\widehat{s},s)\, p_j(s)\, J_j(s) ds &= \Re\left\{ \frac{1}{2\pi} \sum_{l=0}^{\infty} (-1)^l \cdot \right. \\
&\left. \cdot \left\{ \sum_{k=0}^{\infty} \sum_{m=l}^{\infty} \frac{(k+m-1)!\cdot M_k(\tau_c)}{(\tau_{el}-\tau_c)^{k+m}} \cdot \frac{(\tau'_{el}-\tau_{el})^{m-l}}{(m-l)!} \right\} \frac{(\widehat{\tau}-\tau'_{el})^l}{l!} \right\}, \\
\int_{s_{j-1}}^{s_j} \widehat{P}_{lj}^{*}(\widehat{s},s)\, u_j(s)\, J_j(s) ds &= \Re\left\{ \frac{1}{2\pi} \sum_{l=0}^{\infty} (-1)^l \cdot \right. \\
&\left. \cdot \left\{ \sum_{k=1}^{\infty} \sum_{m=l}^{\infty} \frac{(k+m-1)!\cdot N_k(\tau_c)}{(\tau_{el}-\tau_c)^{k+m}} \cdot \frac{(\tau'_{el}-\tau_{el})^{m-l}}{(m-l)!} \right\} \frac{(\widehat{\tau}-\tau'_{el})^l}{l!} \right\}.
\end{aligned} \tag{14}$$

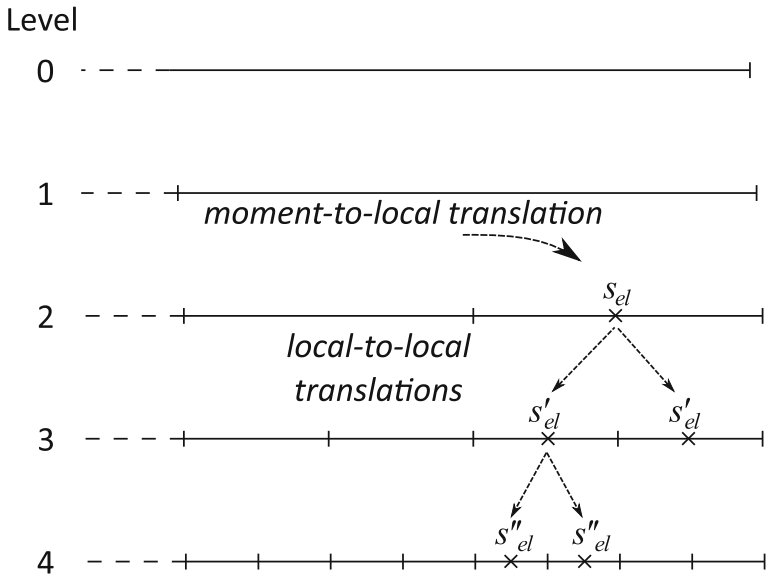

Fig. 6 The downward pass translations

5 The algorithm of the FMM accelerated PIES

The algorithm of FMM accelerated PIES proceeds in the following steps (presented in Fig. 7): determination of the modified tree structure, calculation of the right-hand sided vector **b** and solution of the system of algebraic equations using GMRES solver.

The modified binary tree structure of the FMM combined with the PIES is created based on 2D problems mapped into a parametric reference system (presented in Fig. 1 and described in Section 4.1).

The next step is the calculation of the right-hand sided vector **b** using the first run of the fast multipole procedure. The FMM is composed of two steps: the upward pass and the downward pass. In the upward pass (presented in Algorithm 1) at the lowest level of the tree, all moments in leaves are calculated using the moment expansion procedure (line 4).

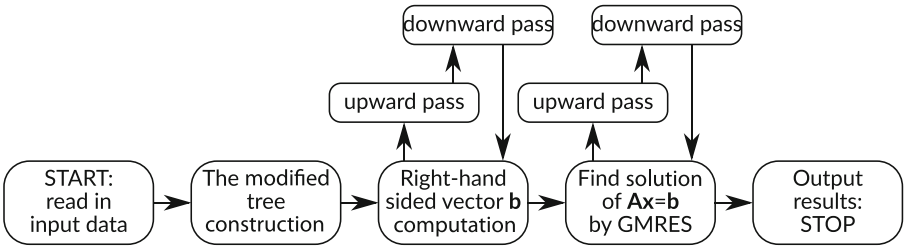

Fig. 7 The fast PIES flow chart

Algorithm 1 Fast multipole procedure - the upward pass.

Require:
lev - the number of tree levels
$\min c_i$ - the lowest number of a cell on the level i
$\max c_i$ - the highest number of a cell on the level i

```
1: for i ← lev to 2 do
2:     for icell ← min c_i to max c_i do
3:         if i == lev then
4:             moments[icell] = moment_expansion(icell)
5:         else
6:             moments[icell] = moment_to_moment(icell)
7:         end if
8:     end for
9: end for
```

Next, tracing the tree structure upward up to level 2, moments in all parent cells are calculated using moment-to-moment translation (line 6), as is presented in Fig. 4.

These moments are used during calculations performed in the downward pass. To present this step, it should be reminded some information about cells neighbourhood [35]. Three relations between any two cells are described in classic FMM—cells might be $adjacent$, $well\text{-}separated$ or far (presented in Fig. 8). Cells are $adjacent$ at level i, if they have a common end at level i. They are $well\text{-}separated$ at level i, if they are not adjacent at level i, but their parent cells are adjacent at level $i-1$. The list of cells $well\text{-}separated$ from cell K at level i is called $interaction\ list$ of cell K. The last relation occurs when parents are not adjacent—cells are far at level i.

The PIES for 2D potential problems is defined in 1D parametric reference system, unlike the FMM which is based on the physical coordinates of 2D problem. Hence, for complex problems some cells which are far away in the parametric space might be close in physical coordinates. Therefore, in the FMM accelerated PIES relations between cells are modified. Two cells K and L are ***not adjacent*** at level i if the distance between K and L mid-points is much bigger than the distance between

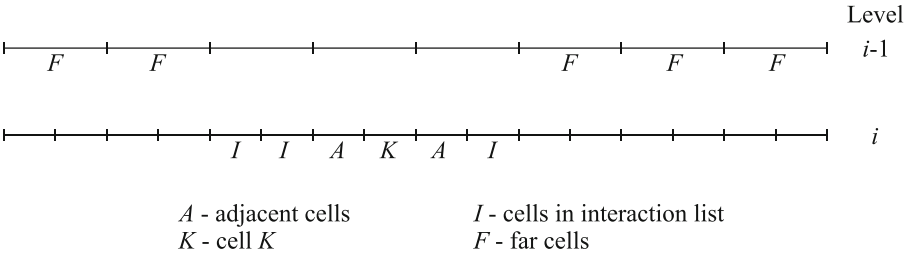

Fig. 8 The neighbourhood of cell K at level i

mid-point of the cell K (or L) and the farthest collocation point on any segment in the cell K (or observation point on any segment in L):

$$|\tau_K - \tau_L| \gg (max\{|\tau_K - \boldsymbol{\tau_{colK}}|\} \,||\, max\{|\tau_L - \boldsymbol{\tau_{colL}}|\})$$

where $\boldsymbol{\tau_{colK}}, \boldsymbol{\tau_{colL}}$ are vectors of all collocation (observation) points coordinates in complex notation in the cells K and L. *Well-separated* cells at level i are *not adjacent* on level i, while their parents are *adjacent* at level $i-1$. Other cells are *far* at level i. Therefore, the FMM algorithm described in [35] should be modified to take into consideration new description of relations between cells.

Tracing the tree structure downward from level 2, coefficients of local expansion for all leaves (the line 15 in Algorithm 2) are computed.

Algorithm 2 Fast multipole procedure - the downward pass.

Require:

lev - the number of tree levels
$\min c_i$ - the lowest number of a cell on the level i
$\max c_i$ - the highest number of a cell on the level i
$moments[n]$ - moments computed during upward pass, where n - the number of cells

```
1:  for i ← 2 to lev do
2:      for icell ← min c_i to max c_i do
3:          if i ≠ 2 then
4:              local_to_local(icell)
5:          end if
6:          for jcell ← min c_i to max c_i do
7:              if parent(icell) & parent(jcell) are neighbours then
8:                  if (cell(icell) & cell(jcell) are neighbours) || !(|τ_col − τ_jcell| ≪
    |τ_icell − τ_jcell|) then
9:                      direct(icell, jcell)
10:                 else
11:                     moment_to_local(icell, jcell, moments[jcell])
12:                 end if
13:             end if
14:         end for
15:         local_expansion(icell)
16:     end for
17: end for
```

Coefficients of the cell K at level i are the sum of two contributions: from all far cells (computed using the local-to-local procedure in line 4) and from well-separated cells (computed using the moment-to-local procedure in line 11). At level 2, far cells to the cell K are absent; therefore, the moment-to-local procedure is used to compute coefficients. At the lowest level, contributions from adjacent cells of the leaf K are computed using directly (direct procedure presented in line 9), i.e. in the same way as in conventional PIES. At last, a right-hand sided vector **b** is produced by the FMM with modified relations between cells.

The FMM accelerated PIES produces the system of algebraic equations implicitly, unlike conventional PIES. Application of iterative GMRES solver requires the multiplication of the matrix **A** by the vector of unknowns **x** procedure, which is the result of the FMM. The solver is directly integrated with the FMM and replaces Gaussian elimination applied in conventional PIES.

Algorithm 3 The FMM accelerated PIES algorithm

Require:
$\mathbf{x_1}, \mathbf{x_2}$ - vectors of coordinates of control points
vbc - vector of boundary conditions values on all segments
kbc - vector of boundary conditions kinds on all segments
lev - the number of tree levels
$\min c_i$ - the lowest number of a cell on the level i
$\max c_i$ - the highest number of a cell on the level i
$moments[n]$ - moments computed during upward pass, where n - the number of cells

1: $[lev, \min c_i, \max c_i] = construct\_tree(\mathbf{x_1}, \mathbf{x_2}, \mathbf{vbc}, \mathbf{kbc})$
2: $\mathbf{b} = right\_hand\_vector(lev, \min c_i, \max c_i)\{$
3: $\quad upward\_pass(lev, \min c_i, \max c_i)$
4: $\quad downward\_pass(lev, \min c_i, \max c_i, moments[n])\}$
5: $\mathbf{x} = call\_GMRES(lev, \min c_i, \max c_i, \mathbf{b})\{$ *//used in proper places of GMRES:*
6: $\quad upward\_pass(lev, \min c_i, \max c_i)$
7: $\quad downward\_pass(lev, \min c_i, \max c_i, moments[n])\}$
8: $output\_results(\mathbf{x})$

The FMM algorithm in GMRES is performed in the same way as for vector **b**. Algorithm 3 presents the FMM accelerated PIES algorithm.

The modified binary tree is created based on the parameters of the problem (see line 1). The next step is calculating the right-sided vector **b** by the FMM procedure with modified relations between cells (in lines 2–4). Then, the GMRES solver integrated with the FMM procedure is used (in lines 5–7). At last, results of the fast PIES, i.e. the vector of solutions **x**, are written to a file (see line 8).

6 Tests of proposed algorithm

Tests are performed on standard PC based on Intel Core i5-4590S with 8 GB RAM and g++ 7.4.0 compiler with -O2 optimization on 64-bit Linux operation system (Ubuntu, kernel 5.3.0) is used. OpenBLAS 0.3.7 (uses BLAS and LAPACK 3.10.3) is used in the implementation of the fast PIES and GMRES procedure in conventional PIES.

6.1 The study of the FMM parameters

First of all, we want to find proper parameters connected with the FMM, such as the number of tree levels, the number of terms in the Taylor series approximated PIES kernels and the value of GMRES tolerance. These parameters have a significant impact on CPU time and RAM utilization of the fast PIES. Two potential boundary problems presented in Fig. 9 are considered.

The first example is the gear-shaped problem described by Laplace's equation (presented in the upper part of Fig. 9). We used 512 curvilinear segments (Bézier

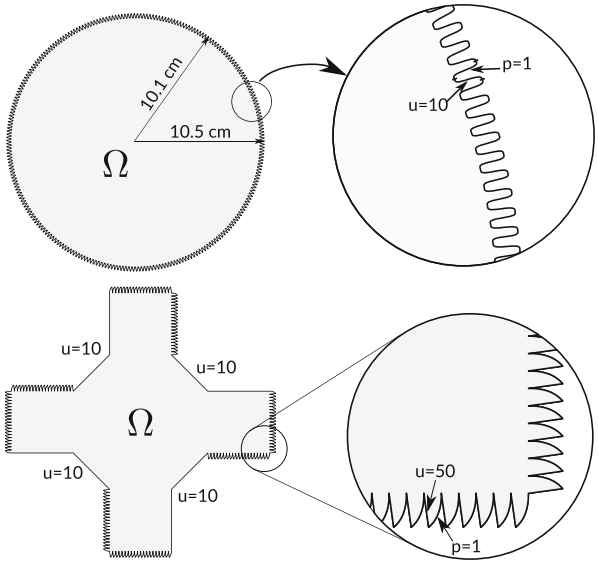

Fig. 9 Shapes of boundaries of considered problems

curves of the third degree) to model the gear composed of 256 teeth. Boundary conditions are the same in each tooth and they are presented in Fig. 9 (where u is Dirichlet and p is Neumann boundary conditions). We used the same number of collocation points on each segment (from 2 to 8) and finally the system of 1024 to 4096 algebraic equations is solved.

The second example is the cross-shaped plate (also described by Laplace's equation), presented in the bottom part of Fig. 9. The boundary is composed of 376 segments (160 curvilinear and 216 linear). Boundary conditions are presented in Fig. 9. Similarly to the previous example, the same number of collocation points on each segment (from 2 to 8) are used, and the system of 752 to 3008 algebraic equations is solved.

6.1.1 The number of tree levels

The first research focused on the influence of the number of tree levels (hence the maximum number of segments in a cell) on the speed of computations, RAM utilization and accuracy of the fast PIES. Approximation of the modified PIES kernels uses 25 terms in the Taylor series, and the GMRES tolerance is equal to 10^{-8}.

As can be seen from Figs. 10 and 11, both CPU time and RAM utilization decrease with the growing number of tree levels reaching the minimum value for about 6–7 levels regardless of the number of collocation points. Hence, the next tests are carried out for 7 tree levels in the FMM accelerated PIES.

We should note a higher RAM utilization in the second example contrary to the first. It is connected with a large number of GMRES iterations. To decrease RAM utilization we can use restarted GMRES. However, it increases CPU time, e.g. for 7

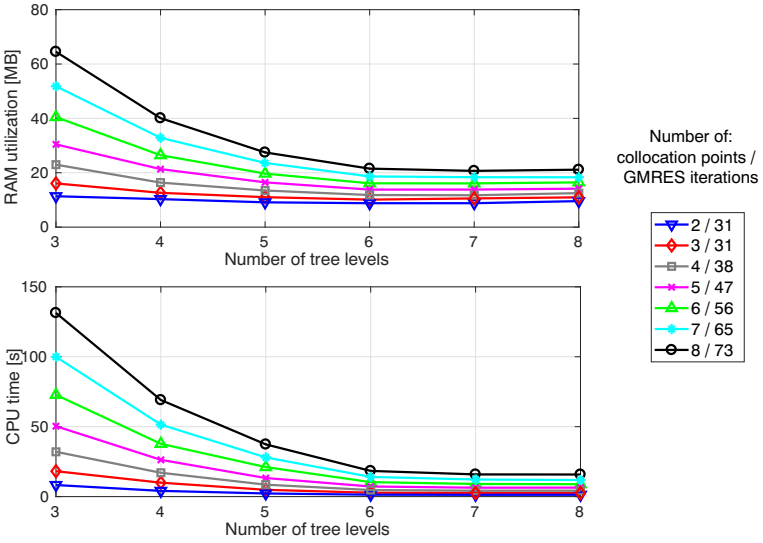

Fig. 10 Comparison of RAM utilization and computation time in the fast PIES with different levels of the tree for the gear-shaped problem

collocation points and 7 levels of the tree RAM utilization decreases from 36.22 to 15.1 MB with a simultaneous CPU time increase from 11.39 to 26.05 s (the GMRES is restarted after every 100 iterations).

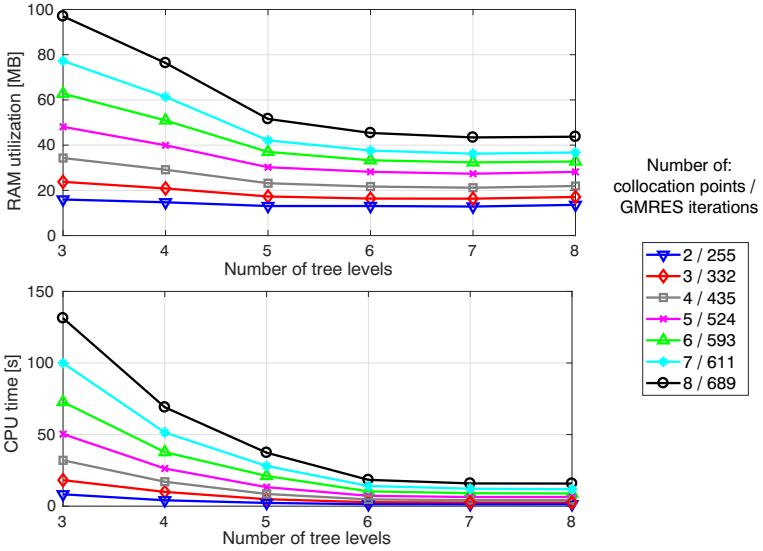

Fig. 11 Comparison of RAM utilization and computation time in the fast PIES with different levels of the tree for the cross-shaped problem

Table 1 The MSE between the fast PIES and two versions of the conventional PIES

Number of	MSE for the gear-shaped problem		MSE for the cross-shaped problem	
collocation points	$PIES_d$	$PIES_i$	$PIES_d$	$PIES_i$
2	$3.045 \cdot 10^{-10}$	$2.842 \cdot 10^{-10}$	$4.166 \cdot 10^{-09}$	$1.487 \cdot 10^{-09}$
3	$5.364 \cdot 10^{-10}$	$7.947 \cdot 10^{-11}$	$7.378 \cdot 10^{-09}$	$4.760 \cdot 10^{-09}$
4	$4.906 \cdot 10^{-10}$	$9.934 \cdot 10^{-11}$	$1.448 \cdot 10^{-08}$	$1.682 \cdot 10^{-08}$
5	$2.197 \cdot 10^{-10}$	$9.542 \cdot 10^{-11}$	$2.694 \cdot 10^{-08}$	$3.383 \cdot 10^{-08}$
6	$5.004 \cdot 10^{-10}$	$7.643 \cdot 10^{-11}$	$1.761 \cdot 10^{-08}$	$2.358 \cdot 10^{-08}$
7	$1.217 \cdot 10^{-09}$	$1.343 \cdot 10^{-10}$	$2.214 \cdot 10^{-08}$	$2.700 \cdot 10^{-08}$
8	$2.396 \cdot 10^{-09}$	$3.977 \cdot 10^{-10}$	$5.379 \cdot 10^{-08}$	$7.255 \cdot 10^{-08}$

To find the accuracy of solutions of the fast PIES, mean square error (MSE) between the conventional and the fast PIES is computed. Two versions of conventional PIES are considered: the PIES with direct solver $PIES_d$ (LAPACK routine DGESV—LU decomposition with partial pivoting) and the PIES with iterative solver $PIES_i$ (GMRES). Therefore, we can also find how strongly iterative solver affects solutions accuracy. The number of terms in the Taylor series is set to 25, the GMRES tolerance is equal to 10^{-8} and 7 tree levels in the fast PIES are used.

As can be seen from Table 1, the mean square error (MSE) between the fast PIES and conventional versions of the PIES is on almost the same very low level in both examples and not exceed 10^{-8}. Hence, the fast PIES is as accurate as both versions of conventional PIES.

6.1.2 The GMRES tolerance

In the next step, we focused on the comparison of the accuracy of solutions obtained by the FMM accelerated PIES with the conventional PIES for different GMRES tolerance values. The PIES with direct solver $PIES_d$ is considered due to higher MSE obtained in the previous example. The number of terms in Taylor series, which approximate the PIES kernels, is again set to 25. The number of FMM tree levels is equal to 7.

As can be seen from Fig. 12, the MSE calculated between the solutions obtained by the fast and conventional PIES for the GMRES tolerance value less or equal 10^{-8}, regardless of the number of equations, is on a very low level and not exceed 10^{-9}. An increase in the MSE of solutions is noticeable for the tolerance values 10^{-6} and higher. Hence, the next tests are carried out for the GMRES tolerance value equal to 10^{-8}.

6.1.3 The number of terms in the Taylor series

Next research concerns on the influence of the number of terms in the Taylor series on the accuracy of the fast PIES solutions for a different number of collocation points.

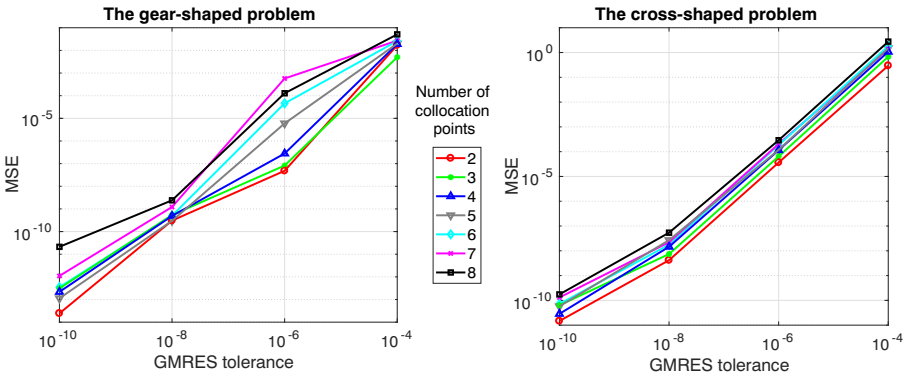

Fig. 12 Comparison of the accuracy of proposed algorithm solutions for different values of the GMRES convergence criterion

Similarly to the previous example, the PIES with direct solver $PIES_d$ is considered. The number of the FMM tree levels is set to 7 and the GMRES tolerance value is equal 10^{-8}.

As can be seen from Fig. 13, a growing number of terms in the Taylor series weakly affects the accuracy of solutions. It means, that even 15 terms in the Taylor series are sufficient to accurately approximate PIES kernels. The MSE, regardless of the defined number of collocation points, has a very low value. Additionally, a growing number of terms in the Taylor series has a slight impact on the time of computation.

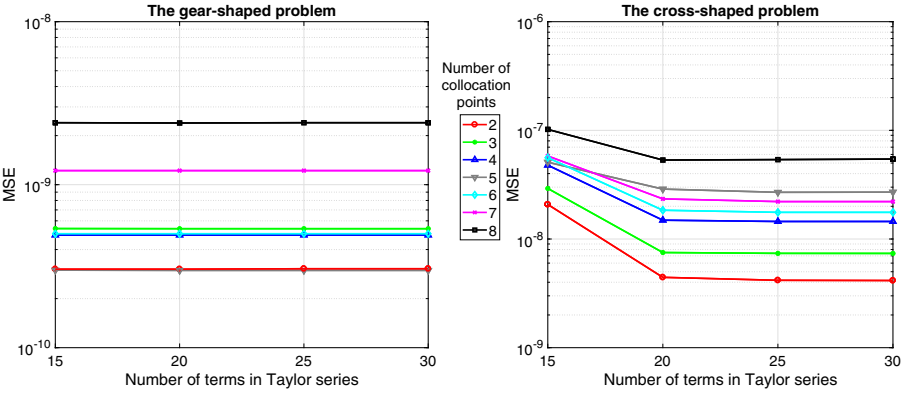

Fig. 13 The influence of the number of terms k in the Taylor series on the accuracy of the fast PIES solutions

6.2 Comparison of the speed and RAM utilization of the fast and conventional PIES

The tests involved a comparison of the speed and RAM utilization between the FMM accelerated PIES ($fPIES$) and two conventional PIES versions ($PIES_d$ and $PIES_i$) for the gear-shaped and the cross-shaped problems. The convergence criterion of the GMRES is equal to 10^{-8}. We assumed 25 terms in the Taylor series, which approximate the modified PIES kernels and 7 levels of the FMM tree. The number of collocation points was the same on each segment (from 2 to 8).

The CPU time and RAM utilization of the FMM accelerated PIES compared to both conventional PIES versions are presented in Table 2.

The fast PIES is significantly faster than any conventional PIES version—the computation time of the FMM accelerated PIES grows linearly contrary to almost square increase in conventional PIES (see in Fig. 14). The fast PIES also needs up to 20 times less RAM during computations—memory utilization of both methods is presented in Fig. 14.

Presented tests involve a rather small number of equations, therefore require less than 400 MB of RAM with the conventional versions of PIES. Thus, we prepared another example of the gear-shaped problem to show the advantage of the FMM accelerated PIES. The main difference from the first problem is a greater number of

Table 2 Comparison of CPU time and RAM utilization between the fast and two versions of conventional PIES

Number of	CPU (s)			RAM (MB)			Number of
col. pts. (eqs)	$fPIES$	$PIES_d$	$PIES_i$	$fPIES$	$PIES_d$	$PIES_i$	GMRES it.
First problem							
2 (1024)	1.28	8.95	8.38	8.80	35.09	23.52	31
3 (1536)	2.56	20.38	19.95	10.56	68.04	44.02	31
4 (2048)	4.21	39.72	38.77	11.72	109	73.04	38
5 (2560)	6.37	58.26	57.53	13.82	165	107	47
6 (3072)	9.03	84.32	83.78	16.09	231	152	56
7 (3584)	12.22	114.19	113.84	18.36	311	205	65
8 (4096)	15.87	142.09	141.70	20.69	401	265	73
Second problem							
2 (752)	2.44	7.42	7.46	12.84	22.76	18.17	255
3 (1128)	3.66	16.42	16.48	16.32	41.28	30.21	332
4 (1504)	5.40	29.29	29.40	21.16	64.98	46.91	435
5 (1880)	7.36	45.97	46.20	27.34	96.72	67.74	524
6 (2256)	9.52	66.30	66.84	32.37	131	93.07	593
7 (2632)	11.39	90.81	91.82	36.22	173	119	611
8 (3008)	14.35	119.41	120.92	43.41	222	153	689

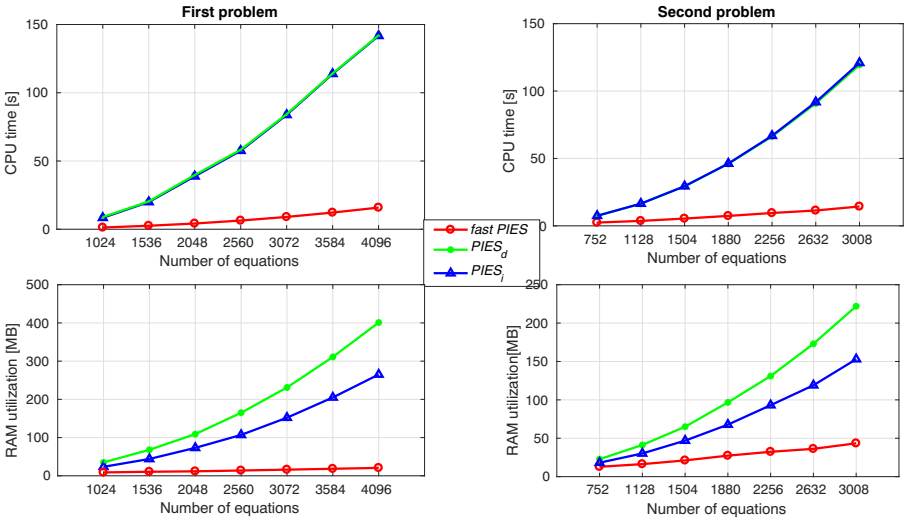

Fig. 14 Comparison of computation time and RAM utilization between the fast and two versions of conventional PIES

teeth (1024), therefore we should solve up to over 16,000 equations (for 8 collocation points on the segment).

Comparison of the speed and RAM utilization of the fast and conventional PIES is presented in Fig. 15.

As can be seen, the efficiency of fast PIES grows significantly with an increasing number of equations. It is most noticeable for RAM utilization. High accuracy of solutions is also preserved—MSE between the fast and conventional PIES not exceed $4.32 \cdot 10^{-10}$.

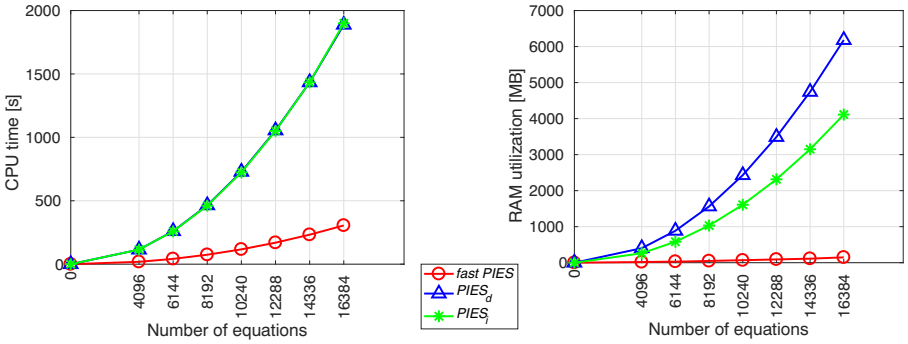

Fig. 15 Comparison of computation time and RAM utilization between the fast and two versions of conventional PIES for bigger size gear-shaped problem

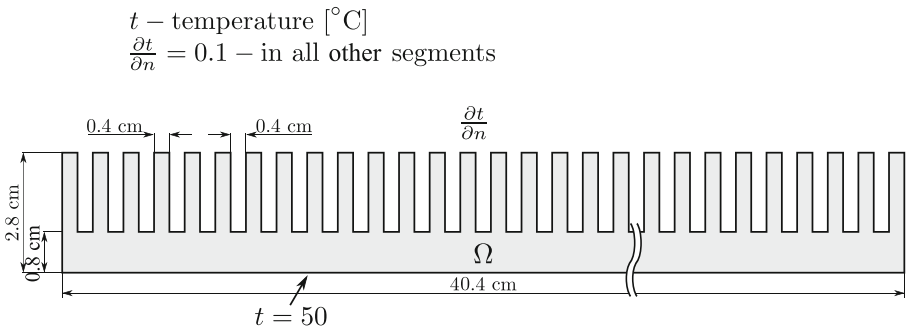

Fig. 16 The shape of considered heat sink

6.3 Comparison of the speed and RAM utilization of the fast PIES and the FMBEM

The last test is connected with a comparison between the fast PIES and the fast multipole BEM (FMBEM) [33]. We considered the problem of temperature distribution in heat sink (the shape of the boundary and boundary conditions are shown in Fig. 16). The boundary in the fast PIES is composed of 716 linear segments. The same number of collocation points (4 and 8) is defined on each segment, and finally, we should solve the system of algebraic equations composed of 2864 and 5728 equations. The boundary in the FMBEM is also discretized by 2864 and 5728 elements to obtain the same size of algebraic equations systems as in the fast PIES.

The value of tolerance (convergence criterion) of the GMRES and the number of terms in the Taylor series is the same as previous, i.e. 10^{-8} and 25 respectively. The accuracy of the solutions is calculated as the mean square error (MSE) between the results of the fast PIES and the fast multipole BEM.

As can be seen from Table 3, the fast PIES is about 2 times faster than the FMBEM. The RAM utilization is on the same level. The MSE between the fast PIES and the fast multipole BEM is not as small as in previous examples. Our previous studies on the accuracy of the conventional PIES (e.g. [15]) proved that the PIES is more accurate than the BEM.

However, we also computed MSE between the fast PIES (2864 equations) and the FMBEM with two other meshes (composed of 11456 and 22,908 boundary elements). Therefore, we obtained two values of MSE $5.836 \cdot 10^{-5}$ and $4.789 \cdot 10^{-6}$ respectively. It proved that application of the FMBEM requires meshes with a large

Table 3 Comparison of computational time and RAM utilization between the fast multipole BEM and the fast PIES with modified binary tree

No. of	CPU time (s)		RAM utilization (MB)		No. of GMRES it.		MSE
equations	fPIES	FMBEM	fPIES	FMBEM	fPIES	FMBEM	
2864	4.63	9.56	40.96	105.9	189	290	0.051
5728	26.79	46.18	124	107.5	267	386	0.063

number of elements to obtain accuracy similar to the fast PIES. The CPU time for these meshes is greater, as well, and reached 69.25 s. and 91.69 s. respectively. It should be noted that RAM utilization is on the same level (110.7 MB and 117.2 MB respectively).

7 Conclusions

The paper presents a new way of accelerating computations and reducing RAM utilization of the PIES applied to solve complex curvilinear potential 2D BVPs. To verify the proposed concept, the FMM is included into the PIES with modified kernels and the fast PIES is obtained. To consider the complex shapes of a boundary, the modification of the binary tree used by the FMM is presented. Numerical tests show a significant reduction of the computation time of the fast PIES compared to the conventional one, while the accuracy of solutions obtained by both methods is almost the same. The fast PIES significantly reduces utilization of RAM, therefore complex curvilinear engineering problems can be solved using a standard PC in a reasonable time.

Obtained results strongly suggest that the chosen direction of research should be continued. Presented algorithm of accelerating computations based on the FMM with modified tree and the PIES should be extended to the problems modelled by other differential equations.

References

1. Zienkiewicz, O.C.: The Finite Element Method. McGraw-Hill, London (1977)
2. Zienkiewicz, O.C., Taylor, R.L., Zhu, J.Z. The Finite Element Method: Its Basis and Fundamentals, 7th edn. Butterworth-Heinemann, Oxford (2013)
3. Babuska, I., Banerjee, U., Osborn, J.E.: Generalized finite element methods: main ideas, results, and perspective. Int. J. Comput. Methods **1**(1), 67–103 (2004)
4. Brebbia, C.A., Telles, J.C.F., Wrobel, L.C.: Boundary Element Techniques, Theory and Applications in Engineering. Springer, New York (1984)
5. Banerjee, P.K., Butterfield, R.: Boundary Element Methods in Engineering Science. McGraw-Hill, London (1981)
6. Katsikadelis, J.T.: Boundary Elements Theory and Applications. Elsevier, Amsterdam (2002)
7. Geng, H., Xu, Z.: Coupling of boundary integral equation and finite element methods for transmission problems in acoustics. Numer. Algo. **82**(2), 479–501 (2019)
8. Rodopoulos, D.C., Gortsas, T.V., Polyzos, K., Tsinopoulos, S.V.: New BEM/BEM and BEM/FEM scalar potential formulations for magnetostatic problems. Eng. Anal. Bound. Elem. **106**, 160–169 (2019)

9. Singh, R., Singh, K.M.: Interpolating meshless local Petrov-Galerkin method for steady state heat conduction problem. Eng. Anal. Bound. Elem. **101**, 56–66 (2019)
10. Milewski, S.: Selected computational aspects of the meshless finite difference method. Numer. Algo. **63**(1), 107–126 (2013)
11. Hughes, T.J.R., Cottrell, J.A., Bazilevs, Y.: Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Comput. Methods Appl. Mech. Eng. **194**(39-41), 4135–4195 (2005)
12. Bazilevs, Y., Beirao Da Veiga, L., Cottrell, J.A., Hughes, T.J.R., Sangalli, G.: Isogeometric analysis: Approximation, stability and error estimates for h-refined meshes. Math. Models Methods Appl. Sci. **16**(7), 1031–1090 (2006)
13. Beirao Da Veiga, L., Brezzi, F., Cangiani, A., Manzini, G., Marini, L.D., Russo, A.: Basic principles of virtual element methods. Math. Models Methods Appl. Sci. **23**(1), 199–214 (2013)
14. Beirao Da Veiga, L., Russo, A., Vacca, G.: The virtual element method with curved edges. ESAIM Math. Model. Numer. Anal. **53**(2), 375–404 (2019)
15. Zieniuk, E.: Hermite curves in the modification of integral equations for potential boundary-value problems. Eng. Comput. **20**(1-2), 112–128 (2003)
16. Zieniuk, E., Szerszeń, K.: Triangular Bézier patches in modelling smooth boundary surface in exterior Helmholtz problems solved by PIES. Arch. Acoust. **34**(1), 51–61 (2009)
17. Zieniuk, E., Boł tuć, A.: Non-element method of solving 2D boundary problems defined on polygonal domains modeled by Navier equation. Int. J. Solids Struct. **43**(25-26), 7939–7958 (2006)
18. Zieniuk, E., Kapturczak, M., Kużelewski, A.: Concept of modeling uncertainly defined shape of the boundary in two-dimensional boundary value problems and verification of its reliability. Appl. Math. Model. **40**(23-24), 10274–10285 (2016)
19. Zieniuk, E., Kużelewski, A., Kapturczak, M.: The influence of interval arithmetic on the shape of uncertainly defined domains modelled by closed curves. Comput. Appl. Math. **37**(2), 1027–1046 (2018)
20. Martínez-Frutos, J., Herrero-Pérez, D., Kessler, M., Periago, F.: Risk-averse structural topology optimization under random fields using stochastic expansion methods. Comput. Meth. Appl. Mech. Eng. **330**, 180–206 (2018)
21. Zapletal, J., Of, G., Merta, M.: Parallel and vectorized implementation of analytic evaluation of boundary integral operators. Eng. Anal. Bound. Elem. **96**, 194–208 (2018)
22. Fu, J., Liang, J., Ba, Z.: Non-singular boundary element method on impedances of three-dimensional rectangular foundations. Eng. Anal. Bound. Elem. **99**, 100–110 (2019)
23. Torky, A.A., Rashed, Y.F.: GPU acceleration of the boundary element method for shear-deformable bending of plates. Eng. Anal. Bound. Elem. **74**, 34–48 (2017)
24. Belinassi, G., Goldman, A., Gubitoso, M.D., Carrion, R.: Vibration soil isolation analysis based on a 3-D frequency domain direct boundary element implementation: GPGPU acceleration. Eng. Anal. Bound. Elem. **105**, 178–187 (2019)
25. Kużelewski, A., Zieniuk, E.: OpenMP for 3D potential boundary value problems solved by PIES. In: 13th International Conference of Numerical Analysis and Applied Mathematics ICNAAM 2015, AIP Conf. Proc., vol. 1738, p. 480098 (2016)
26. Kużelewski, A., Zieniuk, E., Bołtuć, A.: Application of CUDA for acceleration of calculations in boundary value problems solving using PIES. In: Parallel Processing and Applied Mathematics PPAM 2013, LNCS 8385, Part II, pp. 322–331. Springer, Berlin (2014)
27. Kużelewski, A., Zieniuk, E., Kapturczak, M.: Acceleration of integration in parametric integral equations system using CUDA. Comput. Struct. **152**, 113–124 (2015)
28. Rokhlin, V.: Rapid solution of integral equations of classical potential theory. J. Comput. Phys. **60**(2), 187–207 (1985)
29. Greengard, L.F., Rokhlin, V.: A fast algorithm for particle simulations. J. Comput. Phys. **73**(2), 325–348 (1987)
30. Greengard, L.F.: The Rapid Evaluation of Potential Fields in Particle Systems. MIT Press, Cambridge (1988)
31. Kropinski, M.C.A., Nigam, N.: Fast integral equation methods for the Laplace-Beltrami equation on the sphere. Adv. Comput. Math. **40**(2), 577–596 (2014)
32. O'Neil, M.: Second-kind integral equations for the Laplace-Beltrami problem on surfaces in three dimensions. Adv. Comput. Math. **44**(5), 1385–1409 (2018)
33. Liu, Y.J., Nishimura, N.: The fast multipole boundary element method for potential problems: A tutorial. Eng. Anal. Bound. Elem. **30**(5), 371–381 (2006)

34. Dansou, A., Mouhoubi, S., Chazallon, C.: Optimizations of a fast multipole symmetric Galerkin boundary element method code. Numer. Algo. **84**(3), 825–846 (2020)
35. Kużelewski, A., Zieniuk, E.: The fast parametric integral equations system in an acceleration of solving polygonal potential boundary value problems. Adv. Eng. Softw. **141**, 102770 (2020)
36. Gottlieb, D., Orszag, S.A.: Numerical Analysis of Spectral Methods: Theory and Applications. SIAM, Philadelphia (1977)
37. Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D.: LAPACK Users' Guide, Society for Industrial and Applied Mathematics Philadelphia (1999)
38. Kużelewski, A., Zieniuk, E.: OpenMP, multi-threaded libraries for numerical linear algebra and the FMM in an acceleration of numerical solving of the PIES. In: Nketsa, A. et al. (eds.) The 34th Annual European Simulation and Modelling Conference ESM2020 (2020). Modelling and Simulation 2020, pp. 21–26. EUROSIS-ETI Publication, Ostend
39. Saad, Y., Schultz, M.H.: GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. **7**(3), 856–869 (1986)
40. Kużelewski, A., Zieniuk, E., Bołtuć, A., Szerszeń, K.: Modified binary tree in the fast PIES for 2D problems with complex shapes. In: International Conference on Computational Science ICCS 2020, LNCS 12138, Part II, pp. 1–14. Springer, Berlin (2020)