Domain solutions obtained by the FPIES for potential 2D BVPs

Andrzej Kużelewski[0000−0003−2247−2714] and Eugeniusz Zieniuk[0000−0002−6395−5096]

Faculty of Computer Science, University of Bialystok
Ciolkowskiego 1M, 15-245 Bialystok, Poland
{a.kuzelewski,e.zieniuk}@uwb.edu.pl

Abstract. The paper presents the fast parametric integral equations system (FPIES) for domain solutions of potential 2D boundary value problems (BVPs). The FPIES has been successfully applied in modelling 2D potential BVPs and finding solutions on the boundary. The combination of the modified fast multipole technique with the PIES reduced the numerical computation time and RAM usage in the fast PIES. Similar techniques are used to find solutions in the domain. The method is demonstrated with the solution in the domain of two BVPs.

Keywords: Fast PIES · BVPs · Fast Multipole Method.

1 Introduction

The fast parametric integral equations system (FPIES) [1, 2] is a robust numerical tool for solving boundary value problems (BVPs). The method is an extension of the conventional PIES [3], allowing the solution of large-scale BVPs on a standard PC. The PIES does not require a mesh. Therefore, it can be considered as a mesh-free method. However, unlike other mesh-free methods (such as particle methods [4], Galerkin methods [5] or cloud methods [6]), which are mainly used in simulations related to, for example, plastic materials, fluid dynamics or crack simulations, the application of the PIES is comparable to widely used numerical element methods such as the FEM [7–9] and the BEM [10–12] or the still being developed the FEM-BEM hybrids [13], isogeometric analysis (IgA) [14] or the Virtual Element Method (VEM) [15].

The FPIES, as the successor of the PIES, offers the same remarkable advantages compared to the FEM and the BEM. At first, the lack of discretization of the problem's boundary and domain is due to the direct inclusion of parametric functions describing the boundary of the problem into the mathematical formalism of the FPIES. These functions are widely used in computer graphics and include, among others, Bézier and B-spline curves, Coons and Bézier surface patches. Therefore, a small number of control points is required to define the shape of the boundary in the FPIES. In addition, the use of parametric functions reduces the dimensionality of a solved problem by one. At last, despite the high

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_9

accuracy of the solutions obtained by the FPIES, it is possible to improve the accuracy with only a slight modification of the input data (precisely, the number of collocation points), unlike the BEM and the FEM, where the discretization process has to be repeated.

Previous studies on the conventional PIES have focused on various aspects of the method, including the distribution of collocation points [16], the use of NURBS curves [17], or application to problems described by other PDEs, such as Navier-Lame equations [18]. PIES use for nonlinear problems is also described in [19]. All these studies have confirmed its higher accuracy compared to BEM or FEM. However, time-consuming calculations and RAM utilization increase with the problem size's square. Therefore, similarly to the BEM, to compute the PIES matrices, we need $O(N^2)$ operations and another $O(N^3)$ operations to solve the obtained system using direct solvers (where N - the number of equations of algebraic equations system).

On the other hand, the FPIES uses the fast multipole method (FMM) [20–22], which eliminates the main disadvantage of the PIES, namely the generation of dense non-symmetric coefficient matrices and the Gaussian elimination used to solve the final system of algebraic equations. Thanks to the FMM and modified binary tree [23], the system of algebraic equations $\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}$ in the IFPIES is generated implicitly and solved by iterative GMRES solver [24]. It means that only the result of the multiplication of the matrix $\boldsymbol{A}$ by the vector of unknowns $\boldsymbol{x}$ is stored in memory, in contrast to the conventional PIES, which has to store the dense matrix $\boldsymbol{A}$ and the vector $\boldsymbol{b}$ in RAM. The application of the FMM allows for a significant reduction in computational time to order $O(NlogN)$ and a decrease in utilization of RAM to $O(N)$. Previous studies on the FPIES have focused on efficient and accurate solutions to the problems described by single- and multi-connected domains [2, 25], as well as some analysis on key parameters [26, 27]. Also, a comparison to a competing method such as fast multipole BEM (FMBEM) is described in [28].

So far, the authors of this paper have used the FPIES to find solutions of BVPs on the boundary. Solutions in the domain can be obtained using the technique applied in the conventional PIES [3]. The main goal of this paper is to present the FPIES applied to find numerical solutions for 2D potential BVPs in the domain using the fast multipole technique. The efficiency and accuracy of the FPIES are tested on the potential problems described by linear and curvilinear boundary segments.

2 The fast parametric integral equations system

The FPIES for 2D potential problems [2] was obtained as the result of modification of the conventional PIES [3]. The way of obtaining the FPIES included the modification of the PIES kernels (to allow for the Taylor series approximation used by the FMM) and the modification of the tree used by the FMM (to include the way of defining the boundary in the PIES) is clearly presented in, among

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_9

others, [23] and [25]. The following formula presents the final form of the FPIES for solving BVPs on the boundary [2]:

$$\frac{1}{2}u_l(\widehat{s}) = \sum_{j=1}^{n}\mathbb{R}\left\{\frac{1}{2\pi}\sum_{l=0}^{N_T}(-1)^l\cdot\left[\sum_{k=0}^{N_T}\sum_{m=l}^{N_T}\frac{(k+m-1)!\cdot M_k(\tau_c')}{(\tau_{el}-\tau_c)^{k+m}}\right.\right.$$
$$\left.\left.\cdot\frac{(\tau_{el}'-\tau_{el})^{m-l}}{(m-l)!}\right]\cdot\frac{(\widehat{\tau}-\tau_{el}')^l}{l!}\right\} - \sum_{j=1}^{n}\mathbb{R}\left\{\frac{1}{2\pi}\sum_{l=0}^{N_T}(-1)^l\cdot\right. \tag{1}$$
$$\left.\left[\sum_{k=1}^{N_T}\sum_{m=l}^{N_T}\frac{(k+m-1)!\cdot N_k(\tau_c')}{(\tau_{el}-\tau_c)^{k+m}}\cdot\frac{(\tau_{el}'-\tau_{el})^{m-l}}{(m-l)!}\right]\cdot\frac{(\widehat{\tau}-\tau_{el}')^l}{l!}\right\},$$

where moments $M_k(\tau_c')$ and $N_k(\tau_c')$ are computed twice only and have the form:

$$M_k(\tau_c') = \sum_{l=0}^{k}\frac{(\tau_c-\tau_c')^{k-l}}{(k-l)!}M_l(\tau_c), \quad M_l(\tau_c) = \int_{s_{j-1}}^{s_j}\frac{(\tau-\tau_c)^l}{l!}p_j(s)\,J_j(s)ds,$$
$$N_k(\tau_c') = \sum_{l=1}^{k}\frac{(\tau_c-\tau_c')^{k-l}}{(k-l)!}N_l(\tau_c),\ N_l(\tau_c) = \int_{s_{j-1}}^{s_j}\frac{(\tau-\tau_c)^{l-1}}{(l-1)!}n^{(c)}u_j(s)\,J_j(s)\,ds, \tag{2}$$

N_T is the number of terms in Taylor series, $J_j(s)$ is the Jacobian, s_{j-1} correspond to the beginning of j-th segment, while s_j to its end, $u_j(s)$ and $p_j(s)$ are boundary functions (3), $n^{(c)}$ is the complex notation of normal vector to the curve, which creates segment j, $\mathbb{R}$ is the real part of complex number and superscript (c) means the complex variable.

Expressions $\widehat{\tau}, \tau$ are the complex version of parametric functions describing the boundary:

$$\widehat{\tau} = S_l^{(c)}(\widehat{s}) = S_l^{(1)}(\widehat{s}) + iS_l^{(2)}(\widehat{s}), \quad \tau = S_j^{(c)}(s) = S_j^{(1)}(s) + iS_j^{(2)}(s)$$

where $S_k^{(i)}(s_n)$ ($i = \{1,2\}$, $k = \{j,l\}$ $s_n = \{\widehat{s}, s\}$) are parametric curves or lines, which define particular segments of the boundary of the problem (l or j). The points τ_c' and τ_{el}' are obtained during tracing the tree structure.

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

$$u_j(s) = \sum_{k=0}^{N}u_j^{(k)}L_j^{(k)}(s), \quad p_j(s) = \sum_{k=0}^{N}p_j^{(k)}L_j^{(k)}(s), \tag{3}$$

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 (3), which approximated boundary functions on the segment j and $L_j^{(k)}(s)$ - the base functions (Lagrange polynomials) on segment j.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_9

3 Numerical solutions in the domain

Solving the FPIES (1) results in solutions on the boundary only. To find solutions in the domain, a modification of the integral identity from the conventional PIES is required. The original identity uses the solutions of the PIES, and it has the following form [3]:

$$u(x) = \sum_{j=1}^{n} \int_{s_{j-1}}^{s_j} \left\{ \widehat{U}_j^*(x,s) p_j(s) - \widehat{P}_j^*(x,s) u_j(s) \right\} J_j(s) ds. \tag{4}$$

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

$$\begin{aligned} \widehat{U}_j^*(x,s) &= \frac{1}{2\pi} \ln \frac{1}{\sqrt{{r_1}^2 + {r_2}^2}} \\ \widehat{P}_j^*(x,s) &= \frac{1}{2\pi} \frac{r_1 n^{(1)}(s) + r_2 n^{(2)}(s)}{r_1^2 + r_2^2} \end{aligned} \tag{5}$$

where $r_1 = x^{(1)} - S_j^{(1)}(s)$ and $r_2 = x^{(2)} - S_j^{(2)}(s)$, $x^{(1)}$ and $x^{(2)}$ are the coordinates of the solution point in the domain, $n^{(1)}$ and $n^{(2)}$ are components of vector normal to the curve. The shape of the boundary is included in (5) by r_1 and r_2 which contain parametric curves.

At first, we should write data in complex number system:

$$\tau = S_j^{(1)}(s) + i S_j^{(2)}(s), \quad x^{(c)} = x^{(1)} + i x^{(2)}, \quad n^{(c)}(s) = n^{(1)}(s) + i n^{(2)}(s).$$

Then

$$x^{(c)} - \tau = x^{(1)} - S_j^{(1)}(s) + i\left(x^{(2)} - S_j^{(2)}(s)\right) = \left|x^{(c)} - \tau\right| e^{i \operatorname{atan} \frac{x^{(2)} - S_j^{(2)}(s)}{x^{(1)} - S_j^{(1)}(s)}},$$

where $\left|x^{(c)} - \tau\right| = \sqrt{\left(x^{(1)} - S_j^{(1)}(s)\right)^2 + \left(x^{(2)} - S_j^{(2)}(s)\right)^2} = \sqrt{r_1^2 + r_2^2}$.

Therefore, the complex integrands $\widehat{U}_j^{*(c)}(x^{(c)}, \tau)$ and $\widehat{P}_j^{*(c)}(x^{(c)}, \tau)$ have the following form:

$$\widehat{U}_j^{*(c)}(x^{(c)}, \tau) = -\frac{1}{2\pi} \ln\left(x^{(c)} - \tau\right), \quad \widehat{P}_j^{*(c)}(x^{(c)}, \tau) = \frac{1}{2\pi} \frac{n^{(c)}(s)}{x^{(c)} - \tau}, \tag{6}$$

while

$$\widehat{U}_j^*(x,s) = \mathbb{R}\{\widehat{U}_j^{*(c)}(x^{(c)}, \tau)\} = \mathbb{R}\left\{ -\frac{1}{2\pi} \left[\ln\left|x^{(c)} - \tau\right| + i \operatorname{atan} \frac{x^{(2)} - S_j^{(2)}(s)}{x^{(1)} - S_j^{(1)}(s)} \right] \right\} =$$

$$= -\frac{1}{2\pi} \ln\left|x^{(c)} - \tau\right| = \frac{1}{2\pi} \ln \frac{1}{\sqrt{{r_1}^2 + {r_2}^2}},$$

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_9

$$\widehat{P}_j^*(x,s) = \mathbb{R}\{\widehat{P}_j^{*(c)}(x^{(c)},\tau)\} = \mathbb{R}\left\{\frac{1}{2\pi}\frac{n^{(c)}(s)\cdot(x^{(c)}-\tau)^*}{(x^{(c)}-\tau)\cdot(x^{(c)}-\tau)^*}\right\} =$$

$$= \frac{1}{2\pi}\frac{n^{(1)}(s)\big(x^{(1)}-S_j^{(1)}(s)\big)+n^{(2)}(s)\big(x^{(2)}-S_j^{(2)}(s)\big)}{\big(x^{(1)}-S_j^{(1)}(s)\big)^2+\big(x^{(2)}-S_j^{(2)}(s)\big)^2} = \frac{1}{2\pi}\frac{r_1 n^{(1)}(s)+r_2 n^{(2)}(s)}{r_1^2+r_2^2}$$

where $(x^{(c)}-\tau)^*$ is a complex conjugate to $(x^{(c)}-\tau)$.

Then, similarly to the FPIES, we can use the same fast multipole tree and Taylor series. Therefore, moments $M_k(\tau_c)$ and $N_k(\tau_c)$ are calculated previously. Substituting kernels $\widehat{U}_j^{*(c)}\left(x^{(c)},\tau\right)$ and $\widehat{P}_j^{*(c)}\left(x^{(c)},\tau\right)$ into (4), we obtain the following expression:

$$u(x) = \frac{1}{2\pi}\sum_{j=1}^{n}\mathbb{R}\Big\{\sum_{k=0}^{N_T}U_k(x^{(c)},\tau_c)M_k(\tau_c) - \sum_{k=1}^{N_T}P_k(x^{(c)},\tau_c)N_k(\tau_c)\Big\}, \qquad (7)$$

where:

$$U_k(x^{(c)},\tau_c) = \begin{cases} -\ln\left(x^{(c)}-\tau_c\right) & \text{for } k=0 \\ \frac{(k-1)!}{\left(x^{(c)}-\tau_c\right)^k} & \text{for } k\geq 1 \end{cases},$$

$$P_k(x^{(c)},\tau_c) = \frac{(k-1)!}{\left(x^{(c)}-\tau_c\right)^k} \quad \text{for} \quad k\geq 1.$$

To find solutions in the domain, only moments in the leaves are used.

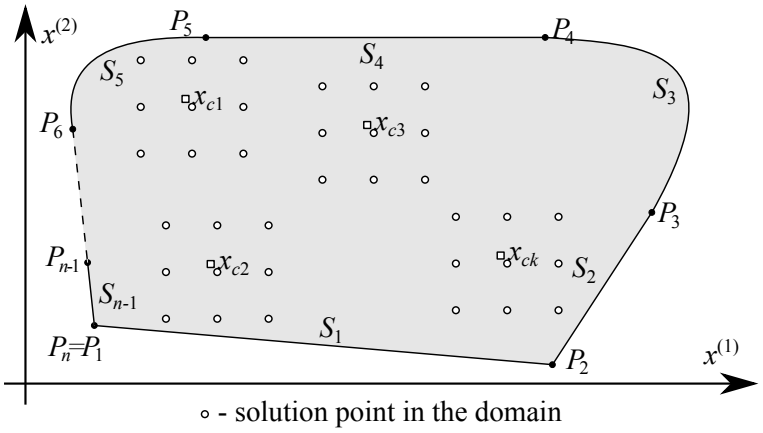

Fig. 1. Solutions in the domain

To reduce the number of computations, we also expanded equation (7) using Taylor series about any point $x_c \in \{x_{c1}, x_{c2}, ..., x_{ck}\}$ close to the points of solutions (presented in Fig.1). Therefore, the integral identity in the FPIES has the

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_9

following form:

$$u(x) = \sum_{j=1}^{n} \mathbb{R} \sum_{l=0}^{N_T} \frac{(-1)^l}{2\pi} \left(\sum_{k=0}^{N_T} \frac{(k-1)! \cdot \left(M_k(\tau_c) - N_k(\tau_c)\right)}{(x_c - \tau_c)^k} \right) \frac{\left(x^{(c)} - x_c\right)^l}{l!}, \quad (8)$$

where $N_0(\tau_c) = 0$.

4 Numerical results

All tests of the presented algorithm are performed on PC based on Intel Core i5-4590S with 32 GB RAM. The program is compiled by g++ 7.5.0 (with -O2 optimization) on 64-bit Ubuntu Linux operation system (kernel 6.8.0). conventional PIES was a bit modified and uses iterative solver GMRES to find the solution of the system of algebraic equations.

4.1 Current flow through the square plate

The first example is the current flow through the square plate presented in Fig. 2. Boundary conditions presented in the figure mean potential u (red electrodes) and flux p on the rest of the boundary.

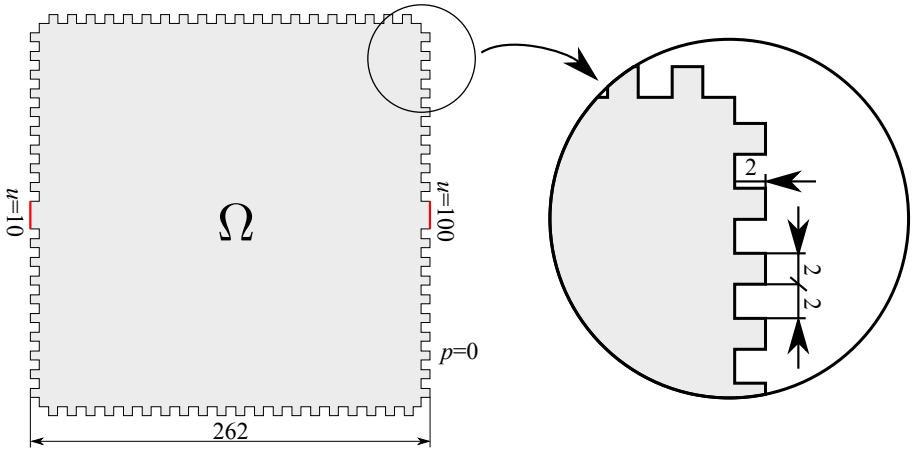

Fig. 2. The example of current flow through the square plate

The research focused on computation speed of solutions in the domain. The FPIES approximation of the modified PIES kernels uses 20 terms in the Taylor series, and the GMRES tolerance is 10^{-8}. The number of collocation points is the same for all segments (from 2 to 8). Therefore, we should solve systems from 2 040 to 8 160 algebraic equations. The same number of terms in Taylor series

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_9

was also used to approximate the integral identity. The number of groups of solution points (points x_c) was equal to 289, while the number of all solution points in the domain was equal to 65 025 (uniformly distributed over the entire domain).

Tab. 1 presents the obtained time of solving the problem (CPU time - solving), the computation of solutions in the domain (CPU time - domain) and the RAM usage of the applications.

Table 1. CPU time and RAM utilization between the fast and conventional PIES

Number of	CPU time - solving [s]		CPU time - domain [s]		RAM utilization [MB]	
col. pts.	*FPIES*	*PIES*	*FPIES*	*PIES*	*FPIES*	*PIES*
2	0.86	7.79	9.82	270.89	6.73	70.14
3	1.61	16.42	13.26	369.18	8.47	148
4	2.88	30.92	19.43	559.84	10.85	260
5	4.42	51.12	24.44	733.08	13.61	403
6	6.94	78.12	31.65	926.5	16.89	578
7	8.80	112.66	35.59	1 174.21	20.68	786
8	12.40	154.06	44.14	1 462.57	24.73	1 024

As can be seen from Tab. 1, the conventional PIES is much slower than the FPIES. For the highest number of equations (8 collocation points), the FPIES required less than 1 minute and 24.73 MB of RAM to solve the problem and compute solutions in the domain, while the conventional PIES required almost 27 minutes and 1024 MB of RAM. Fig. 3 presents the overview of the CPU time and RAM utilization between both applications.

We also calculated the mean square error (MSE) of the domain solutions between the FPIES and the conventional PIES to show the accuracy of the method.

Table 2. MSE of domain solutions between the fast and conventional PIES

Number of collocation points						
2	3	4	5	6	7	8
$1.97 \cdot 10^{-11}$	$2.18 \cdot 10^{-11}$	$2.18 \cdot 10^{-11}$	$2.12 \cdot 10^{-11}$	$1.78 \cdot 10^{-11}$	$2.17 \cdot 10^{-11}$	$1.95 \cdot 10^{-11}$

As can be seen from Tab. 2, solutions in the domain in the FPIES are as accurate as in the conventional PIES. The MSE for over 65 000 solution points does not exceed $3 \cdot 10^{-11}$. A graphical representation of solutions in the domain in the form of potential distribution on the plate is presented in Fig. 4.

4.2 The perforated plate

The second example is the perforated plate (square plate with many holes) shown in Fig. 5. The boundary conditions are given in this figure.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_9

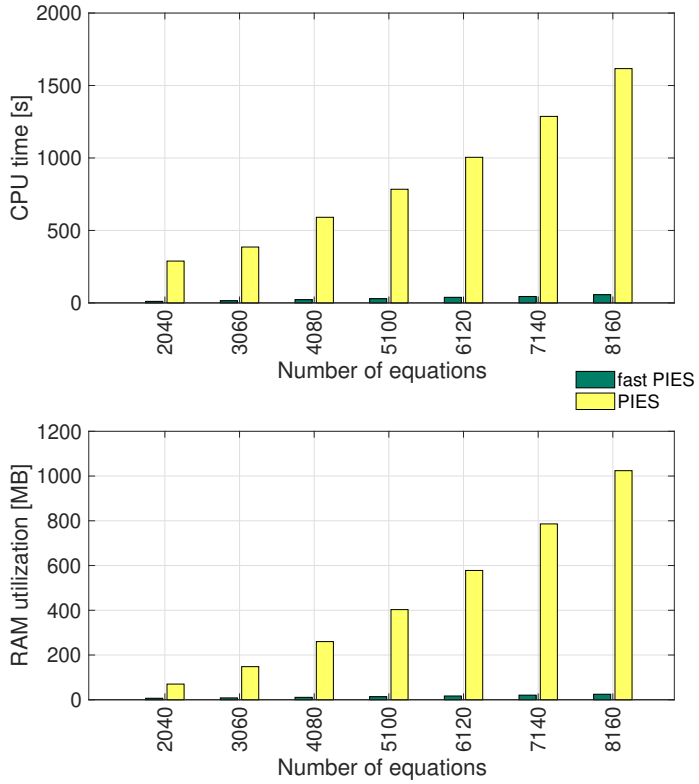

Fig. 3. CPU time and RAM utilization between the PIES and conventional PIES

As in the first example, the research focused on the computation speed of solutions in the domain. The same number of terms in the Taylor series ($N_T = 20$) is used in the FPIES kernels, and the GMRES tolerance is also equal to 10^{-8}. The number of collocation points is the same on all segments (from 2 to 8). We should solve systems from 4 160 to 16 640 algebraic equations in this example. The same number of terms in Taylor series was also used to approximate the integral identity. The number of groups of solution points (points x_c) was equal to 1 600, while the number of all solution points in the domain was close to 300 000 (uniformly distributed over the whole domain).

This research also confirms that the conventional PIES is much slower than the FPIES. As can be seen from Tab. 3, for the highest number of equations (8 collocation points), the FPIES required less than $4\frac{3}{4}$ minutes and 165 MB of RAM to solve the problem and compute solutions in the domain, while the conventional PIES used over $4\frac{3}{4}$ hours and 4 247 MB of RAM. Fig. 6 gives an overview of the CPU time and RAM usage between the two applications.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_9

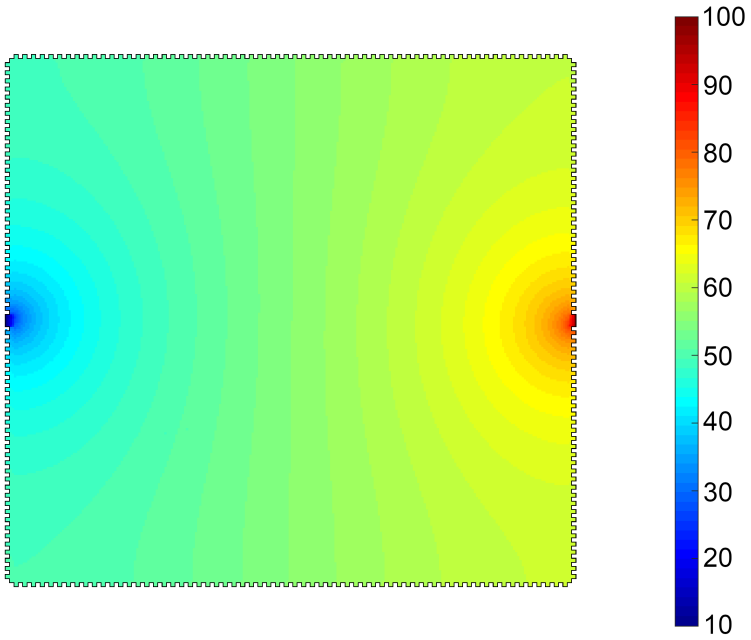

Fig. 4. Potential distribution on the plate

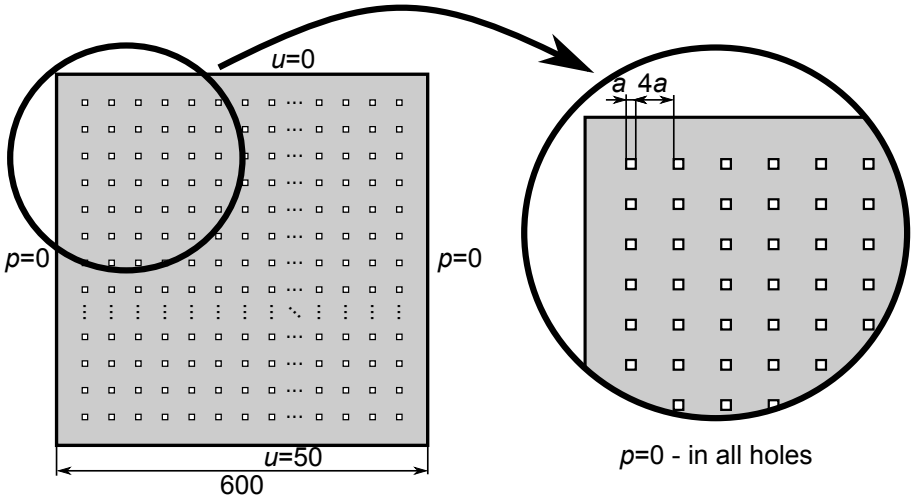

Fig. 5. The example of perforated plate

The highest MSE value of domain solutions between the FPIES and conventional PIES is less than $5.0 \cdot 10^{-8}$. Therefore, the accuracy of the FPIES method is as accurate as the conventional PIES.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_9

Table 3. CPU time and RAM utilization between the fast and conventional PIES

Number of col. pts.	CPU time - solving [s]		CPU time - domain [s]		RAM utilization [MB]	
	FPIES	*PIES*	*FPIES*	*PIES*	*FPIES*	*PIES*
2	10.01	31.76	35.18	2 855.86	19.01	272
3	18.59	75.34	52.42	4 167.13	32.77	604
4	30.79	142.77	73.78	6 380.91	50.76	1 068
5	47.28	226.65	94.62	8 295.64	73.72	1 665
6	68.11	343.78	116.39	10 481.7	98	2 393
7	92.80	498.10	138.73	13 025.4	129	3 254
8	121.85	679.37	161.31	16 566.5	165	4 247

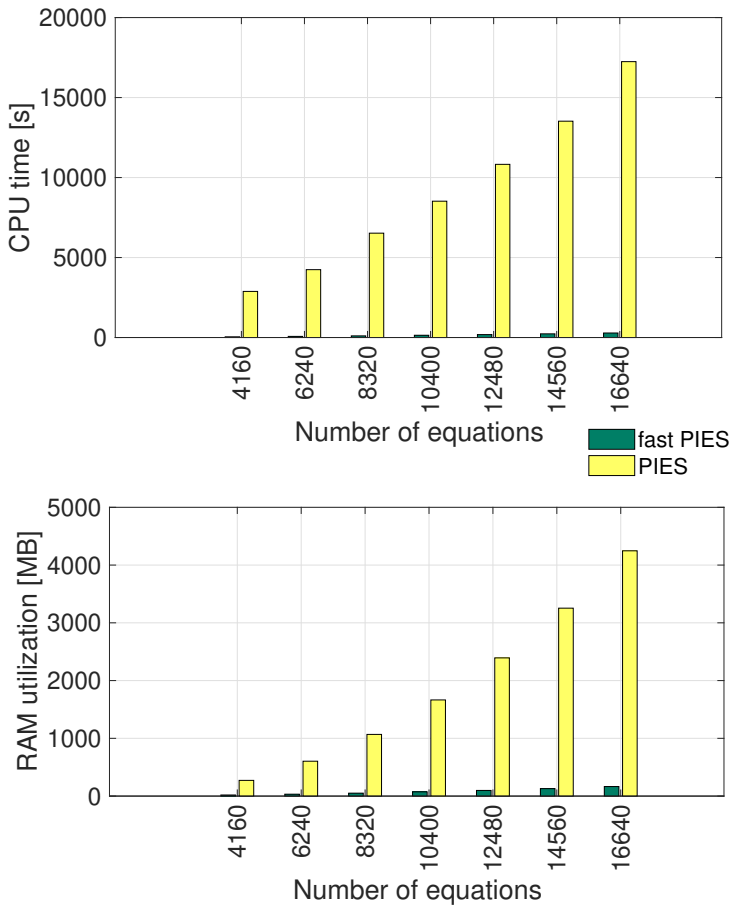

Fig. 6. CPU time and RAM utilization between the PIES and conventional PIES

We also compared the FPIES with the FMBEM [28]. It has been proved previously (among others in [3]) that the application of the BEM requires meshes

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_9

with a large number of elements to obtain accuracy similar to the PIES. Therefore, the boundary in the FMBEM is composed of 10 400 linear elements.

Solutions are compared with ones obtained from 4 collocation points FPIES. The GMRES tolerance (convergence criterion) equals 10^{-8}, and the number of terms in the Taylor series is set to 20, similar to previous examples. The accuracy of the solutions is calculated as the mean square error (MSE) between the results of the FPIES and the fast multipole BEM.

Table 4. Comparison of CPU time and RAM utilization between the FMBEM and the fast PIES

CPU time [s]		RAM utilization [MB]		No. of GMRES it.		MSE
FPIES	FMBEM	FPIES	FMBEM	FPIES	FMBEM	
104.57	217.38	50.76	121	17	77	$2.3904 \cdot 10^{-5}$

As can be seen from Table 4, both CPU time and RAM utilization are smaller in the FPIES. The MSE between the FPIES and the FMBEM is not as small as in the previous examples.

5 Conclusions

The paper presents the FPIES for domain solutions of potential 2D BVPs. The FPIES has previously been successfully applied to modelling 2D potential BVPs and finding solutions on the boundary. The fast multipole technique applied to the integral identity significantly reduces the CPU time for computing domain solutions. Presented examples confirm the high efficiency of the FPIES in solving complex engineering problems on a standard PC in a reasonable time. However, the real power of the FPIES is related to the size of the problems to be solved. The conventional PIES to solve the problems with a system of 16 640 equations (perforated plate, 8 collocation points) uses almost 4.25 GB RAM in $4\frac{3}{4}$ h, while the FPIES requires only 165 MB RAM in $4\frac{3}{4}$ min.

Obtained results strongly suggest that this line of research should be continued. The authors intend to extend the FPIES algorithm to problems modelled by other differential equations.

References

1. A. Kużelewski, E. Zieniuk, The FMM accelerated PIES with the modified binary tree in solving potential problems for the domains with curvilinear boundaries. Numerical Algorithms 88(3), 1025-1050 (2021).
2. A. Kużelewski, E. Zieniuk, Solving of multi-connected curvilinear boundary value problems by the fast PIES. Computer Methods in Applied Mechanics and Engineering 391, 114618 (2022).
3. E. Zieniuk, Hermite curves in the modification of integral equations for potential boundary-value problems. Engineering Computations 20(1-2), 112-128 (2003).

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_9

4. H.-D. Seo, H.-J. Park, J.-I. Kim, P. S. Lee, The particle-attached element interpolation for density correction in smoothed particle hydrodynamics. Advances in Engineering Software 154, 102972 (2021).
5. J. Jaśkowiec, P. Pluciński, Discontinuous Galerkin method in numerical simulation of two-dimensional thermoelasticity problem with single stabilization parameter. Advances in Engineering Software 122, 62-80 (2018).
6. C. A. Duarte, J. T. Oden, H-p clouds - an h-p meshless method. Numerical Methods for Partial Differential Equations 12, 673-705 (1996).
7. O.C. Zienkiewicz, The Finite Element Method, McGraw-Hill, London (1977).
8. O. C. Zienkiewicz, R. L. Taylor, J. Z. Zhu, The Finite Element Method: Its Basis and Fundamentals (7 ed.), Butterworth-Heinemann, Oxford (2013).
9. I. Babuska, U. Banerjee, J. E. Osborn, Generalized Finite Element Methods: Main Ideas, Results, and Perspective. International Journal of Computational Methods 1(1), 67–103 (2004).
10. C.A. Brebbia, J.C.F. Telles, L.C. Wrobel, Boundary element techniques, theory and applications in engineering, Springer-Verlag, New York (1984).
11. P. K. Banerjee, R. Butterfield, Boundary Element Methods in Engineering Science, McGraw-Hill, London (1981).
12. J. T. Katsikadelis, Boundary Elements Theory and Applications, Elsevier, Amsterdam (2002).
13. B. Nedjar, A coupled BEM-FEM method for finite strain magneto-elastic boundary-value problems. Computational Mechanics 59(5), 795-807 (2017).
14. T. J. R. Hughes, J. A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Computer Methods in Applied Mechanics and Engineering 194(39-41), 4135-4195 (2005).
15. L. Beirao Da Veiga, A. Russo, G. Vacca, The Virtual Element Method with curved edges. ESAIM: Mathematical Modelling and Numerical Analysis 53(2), 375-404 (2019).
16. E. Zieniuk, K. Szerszeń, A. Bołtuć, Genetic algorithms applied to optimal arrangement of collocation points in 3D potential boundary-value problems, in: Saeed, K., Pejaś, J. (eds) Information Processing and Security Systems, Springer, Boston, 113-122 (2005).
17. E. Zieniuk, M. Kapturczak, D. Sawicki, The NURBS curves in modelling the shape of the boundary in the parametric integral equations systems for solving the Laplace equation, in: International Conference of Numerical Analysis and Applied Mathematics 2015, ICNAAM 2015, AIP Conference Proceedings 1738, 480100 (2016).
18. E. Zieniuk, A. Bołtuć, Non-element method of solving 2D boundary problems defined on polygonal domains modeled by Navier equation, International Journal of Solids and Structures 43(25-26), 7939-7958 (2006).
19. E. Zieniuk, A. Bołtuć, K. Szerszeń, Shape identification in nonlinear boundary problems solved by PIES method, Acta Mechanica et Automatica 8(1), 16–21 (2014).
20. V. Rokhlin, Rapid solution of integral equations of classical potential theory, Journal of Computational Physics 60(2), 187-207 (1985).
21. L.F. Greengard, V. Rokhlin, A fast algorithm for particle simulations, Journal of Computational Physics 73(2), 325-348 (1987).
22. L.F. Greengard, The rapid evaluation of potential fields in particle systems, The MIT Press, Cambridge (1988).
23. A. Kużelewski, E. Zieniuk, A. Bołtuć, K. Szerszeń, Modified binary tree in the fast PIES for 2D problems with complex shapes, in: International Conference on

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_9

Computational Science ICCS 2020, LNCS 12138, Part II, Springer-Verlag, Berlin, 1-14 (2020).
24. Y. Saad, M. H. Schultz, A generalized minimal residual algorithm for solving non-symmetric linear systems. SIAM Journal on Scientific and Statistical Computing 7, 856-869 (1986).
25. A. Kużelewski, E. Zieniuk, The fast parametric integral equations system in an acceleration of solving polygonal potential boundary value problems. Advances in Engineering Software 141, 102770 (2020).
26. A. Kużelewski A., E. Zieniuk, Searching for the Best Tree Parameters in the IFPIES, in: The 37th Annual European Simulation and Modelling Conference ESM2023, Modelling and Simulation 2023, EUROSIS-ETI Publications, Ghent, Belgium, 48-52 (2023).
27. A. Kużelewski A., E. Zieniuk, Influence of selected IFPIES parameters on CPU time and RAM utilization, in: The 38th Annual European Simulation and Modelling Conference ESM2024, Modelling and Simulation 2024, EUROSIS-ETI Publications, Ghent, Belgium, 288-293 (2024).
28. Y.J. Liu, N. Nishimura, The fast multipole boundary element method for potential problems: A tutorial, Engineering Analysis with Boundary Elements 30(5), 371-381 (2006).

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_9