Engineering Analysis with Boundary Elements 37 (2013) 683–690

Contents lists available at SciVerse ScienceDirect

Engineering Analysis with Boundary Elements

journal homepage: www.elsevier.com/locate/enganabound

A gradient free integral equation for diffusion–convection equation with variable coefficient and velocity

J. Ravnik*, L. Škerget

Faculty of Mechanical Engineering, University of Maribor, Smetanova 17, SI-2000 Maribor, Slovenia

ARTICLE INFO

Article history:
Received 4 June 2012
Accepted 14 January 2013

Keywords:
Boundary element method
Diffusion–convection equation
Diffusion–advection equation
Variable coefficient
Domain decomposition

ABSTRACT

In this paper a boundary-domain integral diffusion–convection equation has been developed for problems of spatially variable velocity field and spatially variable coefficient. The developed equation does not require a calculation of the gradient of the unknown field function, which gives it an advantage over the other known approaches, where the gradient of the unknown field function is needed and needs to be calculated by means of numerical differentiation. The proposed equation has been discretized by two approaches—a standard boundary element method, which features fully populated system matrix and matrices of integrals and a domain decomposition approach, which yields sparse matrices. Both approaches have been tested on several numerical examples, proving the validity of the proposed integral equation and showing good grid convergence properties. Comparison of both approaches shows similar solution accuracy. Due to nature of sparse matrices, CPU time and storage requirements of the domain decomposition are smaller than those of the standard BEM approach.

1. Introduction

Many natural phenomena are governed by the diffusion and convection transport processes. Heat transfer, mass transfer and flow of fluids are some of the examples. In nature and for most engineering purposes, the transport phenomena occur in environments, where the velocity of the fluid changes within the domain in question. In some cases, for example, in turbulence modelling with turbulent viscosity hypothesis, the coefficient also changes within the domain.

Solution of the diffusion–convection partial differential equation is a challenging task. Many numerical algorithms have been proposed. In terms of the boundary element method by using the diffusion–convection fundamental solution, the problem can (at least for constant velocity field and constant coefficient) be described by pure boundary integral equations. This approach has been extensively studied in the past, where methods of solution have been proposed handling the problem up to very high Péclet numbers (Škerget et al. [22], Qiu et al. [16], Wrobel and DeFigueiredo [25], Žagar et al. [23]).

To solve the problem of variable velocity, a decomposition of the velocity field into a constant and variable part has been proposed. The decomposition leads to a domain integral involving the variable part of the velocity field and the unknown field function. DeSilva [4] used this approach to solve the transient conduction convection in 2-D. These approaches are now well known and described in textbooks of Wrobel [24] and Divo and Kassab [5].

More recently, several authors solved the diffusion–convection equations with variable coefficients (Rap et al. [17], Škerget and Ravnik [21]). Decomposition to constant and variable part has been used here as well. It leads to a domain integral involving variable part of the coefficient and the gradient of the unknown field function. Since a gradient of the unknown function is needed, it must be calculated by means of numerical differentiation from a solution of a previous nonlinear iteration step. Having the gradient of the unknown function on the right hand side strengthens the non-linear properties of the problem and requires more nonlinear iterations to reach the solution.

In this work, we present an alternative approach, where the gradient of the coefficient is needed and gradient of the field function is not needed. Thus, the final integral equation (Eq. (21)) includes only the unknown function on the boundary and in the domain and its flux on the boundary. The proposed equation is linear and after discretization requires only a single solution of a system of linear equations to obtain the solution.

The derivation employed is similar to the solution of the diffusion and Helmholtz equations in non-homogenous media, as presented in [10,11,3,2,1,26].

2. Governing equation

Let us consider a domain Ω in $\mathbb{R}^3$ with a boundary Γ. The domain is filled with a fluid. Let $\vec{r}$ be a vector pointing to

* Corresponding author. Tel.: +386 2 220 7745.
E-mail addresses: jure.ravnik@um.si (J. Ravnik), leopold.skerget@um.si (L. Škerget).

0955-7997/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.enganabound.2013.01.012

a point in the domain and let $\vec{v}$ be the fluid velocity. A field function, u, which is subjected to convective and diffusive processes in the domain, is governed by the following PDE:

$$\vec{v}(\vec{r})\cdot\vec{\nabla}u=\vec{\nabla}\cdot(\alpha(\vec{r})\vec{\nabla}u),\quad \vec{r}\in\Omega, \tag{1}$$

with the following Dirichlet and/or Neumann type boundary conditions

$$u(\vec{r})=\overline{u}(\vec{r}),\quad \vec{r}\in\Gamma_D,$$
$$\vec{n}\cdot\vec{\nabla}u(\vec{r})=q(\vec{r})=\overline{q}(\vec{r}),\quad \vec{r}\in\Gamma_N, \tag{2}$$

where Γ_D and Γ_N are the Dirichlet and Neumman parts of the boundary with $\Gamma=\Gamma_D\cup\Gamma_N$.

The fluid is considered incompressible, thus $\vec{\nabla}\cdot\vec{v}=0$. The fluid velocity varies in space. The coefficient, α, in the domain is isotropic and non-homogenous, thus $\alpha(\vec{r})$ (the thermal conductivity in the case of heat transfer or the diffusivity in the case of mass transfer or turbulent viscosity in case of turbulence modelling) is a function of the location.

3. Integral representation

Let us consider a source point $\vec{\xi}$ where the coefficient has a value of $\alpha(\vec{\xi})=\alpha_0$. Furthermore, let $\vec{v}_0$ be the constant part of the velocity. It can be the average velocity in the domain or the velocity at the source point. Taking these values of the coefficient and velocity, we can find the fundamental solution of diffusion–convection equation by keeping the coefficient and velocity constant, i.e. $\alpha(\vec{r})=\alpha(\vec{\xi})=\alpha_0$ and $\vec{v}(\vec{r})=\vec{v}_0$, yielding (Driessen [6])

$$\alpha_0\nabla^2u^\star+\vec{v}_0\cdot\vec{\nabla}u^\star=-\delta(\vec{r},\vec{\xi}), \tag{3}$$

with

$$u^\star(\vec{r},\vec{\xi})=\frac{1}{4\pi|\vec{r}-\vec{\xi}|\alpha_0}\exp\left(\frac{\vec{v}_0\cdot(\vec{r}-\vec{\xi})-v_0|\vec{r}-\vec{\xi}|}{2\alpha_0}\right), \tag{4}$$

where $v_0=|\vec{v}_0|$. The gradient of the fundamental solution is

$$\vec{\nabla}u^\star(\vec{r},\vec{\xi})=\left(\left(\frac{1}{|\vec{r}-\vec{\xi}|}+\frac{v_0}{2\alpha_0}\right)\frac{\vec{r}-\vec{\xi}}{|\vec{r}-\vec{\xi}|}-\frac{\vec{v}_0}{2\alpha_0}\right)u^\star(\vec{r},\vec{\xi}). \tag{5}$$

In this work, we derive and test the boundary-domain diffusion–convection integral equation in 3D. However, all results are valid in 2D cases as well, provided that the appropriate 2D fundamental solution is used.

The variable coefficient and the velocity field are decomposed into constant and variable parts as follows:

$$\alpha(\vec{r})=\alpha_0+\alpha',\quad \vec{v}(\vec{r})=\vec{v}_0+\vec{v}', \tag{6}$$

where α' and $\vec{v}'$ are the variable parts. Using this decomposition, we may rewrite Eq. (1) as

$$\alpha_0\nabla^2u=\vec{v}_0\cdot\vec{\nabla}u-\vec{\nabla}\cdot(\alpha'\vec{\nabla}u)+\vec{v}'\cdot\vec{\nabla}u. \tag{7}$$

Considering for a moment the second and third terms on the right hand side of Eq. (7) as source terms, the standard BEM derivation (Wrobel [24]) yields the following integral representation for a source point $\vec{\xi}\in\Gamma$ located at the boundary:

$$c(\vec{\xi})u(\vec{\xi})+\int_\Gamma\alpha_0u\vec{\nabla}u^\star\cdot d\vec{\Gamma}=\int_\Gamma u^\star\alpha_0\vec{\nabla}u\cdot d\vec{\Gamma}-\int_\Gamma u^\star u\vec{v}_0\cdot d\vec{\Gamma}$$
$$+\underbrace{\int_\Omega u^\star\vec{\nabla}\cdot(\alpha'\vec{\nabla}u)\,d\Omega}_{\text{due to variable coef.}}-\underbrace{\int_\Omega u^\star\vec{v}'\cdot\vec{\nabla}u\,d\Omega}_{\text{due to var. velocity}}, \tag{8}$$

where $c(\vec{\xi})$ is the free coefficient given by the solid surface angle at $\vec{\xi}$. The two domain integrals in Eq. (8) are due to the source terms of Eq. (7) and include the variable parts of the coefficient α' and the variable part of velocity $\vec{v}'$. In their kernels they both include gradient of the solution, $\vec{\nabla}u$. Such a formulation can be used (Škerget and Ravnik [21]), however, the fact that the gradient of the solution is needed to construct the source terms, means that numerical differentiation must be employed. Furthermore, an iterative scheme must be employed, where the solution of (8) and the numerical calculation of $\vec{\nabla}u$ are alternatively calculated until convergence is achieved.

In the following, we are presenting a derivation using algebraic relations and integral clauses to avoid the calculation of the gradient of the solution that leads to a novel integral formulation. Let us first focus on the domain integral of Eq. (8), which is due to variable coefficient. We use the following algebraic relation to transform the integral kernel:

$$\vec{\nabla}\cdot(u^\star\alpha'\vec{\nabla}u)=u^\star\vec{\nabla}\cdot(\alpha'\vec{\nabla}u)+\alpha'\vec{\nabla}u\cdot\vec{\nabla}u^\star. \tag{9}$$

Furthermore, we make use of the rule chain derivation, which states that

$$\vec{\nabla}(\alpha'u)=\alpha'\vec{\nabla}u+u\vec{\nabla}\alpha'. \tag{10}$$

Using (9) and (10), the kernel can be rewritten as

$$u^\star\vec{\nabla}\cdot(\alpha'\vec{\nabla}u)=\vec{\nabla}\cdot(u^\star\alpha'\vec{\nabla}u)+u\vec{\nabla}\alpha'\cdot\vec{\nabla}u^\star-\vec{\nabla}(\alpha'u)\cdot\vec{\nabla}u^\star. \tag{11}$$

The last expression of Eq. (11) can be rewritten using

$$\vec{\nabla}\cdot(\alpha'u\vec{\nabla}u^\star)=\vec{\nabla}(\alpha'u)\cdot\vec{\nabla}u^\star+\alpha'u\nabla^2u^\star. \tag{12}$$

Using (12) in Eq. (11) we obtain the final version for the integral kernel

$$u^\star\vec{\nabla}\cdot(\alpha'\vec{\nabla}u)=\vec{\nabla}\cdot(u^\star\alpha'\vec{\nabla}u)+u\vec{\nabla}\alpha'\cdot\vec{\nabla}u^\star-\vec{\nabla}\cdot(\alpha'u\vec{\nabla}u^\star)+\alpha'u\nabla^2u^\star. \tag{13}$$

The domain integral on the right hand side of Eq. (8) can thus be written as

$$\int_\Omega u^\star\vec{\nabla}\cdot(\alpha'\vec{\nabla}u)\,d\Omega=\int_\Omega\vec{\nabla}\cdot(u^\star\alpha'\vec{\nabla}u)\,d\Omega+\int_\Omega u\vec{\nabla}\alpha'\cdot\vec{\nabla}u^\star\,d\Omega$$
$$-\int_\Omega\vec{\nabla}\cdot(\alpha'u\vec{\nabla}u^\star)\,d\Omega+\int_\Omega\alpha'u\nabla^2u^\star\,d\Omega. \tag{14}$$

The two integrals that feature a divergence of the kernel can be written as boundary integrals using Gauss clause, yielding

$$\int_\Omega u^\star\vec{\nabla}\cdot(\alpha'\vec{\nabla}u)\,d\Omega=\int_\Gamma u^\star\alpha'\vec{\nabla}u\cdot d\vec{\Gamma}+\int_\Omega u\vec{\nabla}\alpha'\cdot\vec{\nabla}u^\star\,d\Omega$$
$$-\int_\Gamma\alpha'u\vec{\nabla}u^\star\cdot d\vec{\Gamma}+\int_\Omega\alpha'u\nabla^2u^\star\,d\Omega. \tag{15}$$

The kernel of the last domain integral in Eq. (15) includes a Laplacian of the fundamental solution. This can be rewritten by using the definition in Eq. (3) as

$$\int_\Omega\alpha'u\nabla^2u^\star\,d\Omega=-\frac{1}{\alpha_0}\int_\Omega\alpha'u\delta(\vec{r},\vec{\xi})\,d\Omega-\frac{1}{\alpha_0}\int_\Omega\alpha'u\vec{v}_0\cdot\vec{\nabla}u^\star\,d\Omega. \tag{16}$$

At his point we choose the constant part of the coefficient to be $\alpha_0=\alpha(\vec{\xi})$. Thus α' is equal to zero at the source point, and since the Kronecker delta is zero everywhere else, the first integral on the right hand side of Eq. (16) vanishes. Using this, Eq. (15) simplifies to

$$\int_\Omega u^\star\vec{\nabla}\cdot(\alpha'\vec{\nabla}u)\,d\Omega=\int_\Gamma u^\star\alpha'\vec{\nabla}u\cdot d\vec{\Gamma}-\int_\Gamma\alpha'u\vec{\nabla}u^\star\cdot d\vec{\Gamma}$$
$$+\int_\Omega u\left(\vec{\nabla}\alpha'-\frac{\alpha'}{\alpha_0}\vec{v}_0\right)\cdot\vec{\nabla}u^\star\,d\Omega. \tag{17}$$

Noticing that $\vec{\nabla}\alpha = \vec{\nabla}\alpha'$, $\alpha'/\alpha_0 = \alpha/\alpha_0 - 1$, using (6) and (17) we rewrite Eq. (8) as

$$c(\vec{\xi})u(\vec{\xi}) + \int_\Gamma \alpha u \vec{\nabla} u^\star \cdot d\vec{\Gamma} = \int_\Gamma u^\star \alpha \vec{\nabla} u \cdot d\vec{\Gamma} - \int_\Gamma u^\star u \vec{v}_0 \cdot d\vec{\Gamma} + \int_\Omega u\left(\vec{\nabla}\alpha + \vec{v}_0 - \frac{\alpha}{\alpha_0}\vec{v}_0\right)\cdot \vec{\nabla} u^\star \, d\Omega - \int_\Omega u^\star \vec{v}' \cdot \vec{\nabla} u \, d\Omega. \tag{18}$$

Next, we turn our attention to the last domain integral of Eq. (18). This integral is needed because of the spatially varying velocity field and it still features the gradient of the function u, which we want to avoid. We start by transforming the kernel using the definition of divergence:

$$\vec{\nabla}\cdot(u^\star u \vec{v}') = u^\star u \vec{\nabla}\cdot\vec{v}' + \vec{v}'\cdot\vec{\nabla}(u^\star u) = \vec{v}'\cdot\vec{\nabla}(u^\star u) = u^\star \vec{v}'\cdot\vec{\nabla} u + u\vec{v}'\cdot\vec{\nabla} u^\star, \tag{19}$$

where we have taken into account that the velocity field is solenoidal, i.e. $\vec{\nabla}\cdot\vec{v}' = 0$. Using (19), the last domain integral of Eq. (18) can be transformed as

$$\int_\Omega u^\star \vec{v}'\cdot\vec{\nabla} u \, d\Omega = \int_\Omega \vec{\nabla}\cdot(u^\star u \vec{v}')\, d\Omega - \int_\Omega u\vec{v}'\cdot\vec{\nabla} u^\star \, d\Omega = \int_\Gamma u^\star u \vec{v}'\cdot d\vec{\Gamma} - \int_\Omega u\vec{v}'\cdot\vec{\nabla} u^\star \, d\Omega, \tag{20}$$

where the Gauss clause has been used to transform the domain integral into a boundary integral. Finally, using (20) in (18) we are able to write the following integral equation:

$$c(\vec{\xi})u(\vec{\xi}) + \int_\Gamma \alpha u \vec{\nabla} u^\star \cdot d\vec{\Gamma} = \int_\Gamma u^\star \alpha \vec{\nabla} u \cdot d\vec{\Gamma} - \int_\Gamma u^\star u \vec{v}\cdot d\vec{\Gamma} + \int_\Omega u\left(\vec{\nabla}\alpha + \vec{v} - \frac{\alpha}{\alpha_0}\vec{v}_0\right)\cdot\vec{\nabla} u^\star \, d\Omega. \tag{21}$$

Here α is the coefficient and $\vec{v}$ is the fluid velocity, which both vary in space. α_0 and $\vec{v}_0$ are the constant parts, which are used to calculate the fundamental solution and its gradient. α_0 is the coefficient at the location of the source point. The constant part of the velocity can be chosen arbitrarily, for example, it can be the average of the velocity field in the domain or the velocity at the source point. The equation is valid for incompressible flows.

For completeness, Eq. (21) written in Cartesian tensor notation is

$$c(\vec{\xi})u(\vec{\xi}) + \int_\Gamma \alpha u \frac{\partial u^\star}{\partial x_j} n_j \, d\Gamma = \int_\Gamma u^\star \alpha q d\Gamma - \int_\Gamma u^\star u v_j n_j \, d\Gamma + \int_\Omega u\left(\frac{\partial \alpha}{\partial x_j} + v_j - \frac{\alpha}{\alpha_0} v_{0,j}\right)\frac{\partial u^\star}{\partial x_j}\, d\Omega, \tag{22}$$

where the flux is defined by $q = \vec{n}\cdot\vec{\nabla} u = n_j \partial u/\partial x_j$.

4. Discretization

Eq. (21) includes boundary as well as domain integrals. In the literature, many approaches have been proposed to handle the domain integral. One option is to make a mesh inside of the domain and calculate the domain integral. This requires a high computational effort and large storage requirements, since the number of integrals that need to be calculated and stored scales with a square of the number of nodes in the domain. The calculation of the domain integral can be avoided by approximate techniques. These use approximations to transform the domain integral onto the boundary, such as Dual reciprocity method (Partridge [13]) and Radial integration method (Yang and Gao [26], Fata [7]). Another option is to approximate the resulting matrix of domain integrals with techniques based on kernel expansion. (Fast multipole method (Popov et al. [15]), panel clustering method (Hackbusch and Nowak [8]) as well as algebraic approximation techniques such as wavelet compression (Ravnik et al. [18,20]) or adaptive cross-approximation (Maerten [9]).) Finally, domain decomposition of the domain and calculation of domain integrals also reduces the cost and storage requirements significantly, since the resulting matrix of domain integrals is sparse (Popov et al. [14]).

Although all of the mentioned techniques can be applied to handle the domain integral of Eq. (21), we choose only two: first, we calculate the domain integral in full by making a domain mesh and storing the fully populated matrix of domain integrals and secondly, we employ the domain decomposition approach. We chose the first because it does not rely on any approximation and can serve as a benchmark of other approximate approaches. We designate the first approach as BDIE and the second as SDBDIE. Both are explained in detail in the following subsections.

4.1. Standard approach—BDIE

In order to obtain a discrete version of (21) we use shape functions to interpolate field functions and flux across the boundary elements and inside of each domain element. In this work we use hexahedral mesh elements with 27 nodes to discretize the volume elements, which enable continuous quadratic interpolation of field functions. The boundary elements are the sides of these hexahedrons. Nine nodes on each boundary element enable continuous quadratic interpolation. On each boundary element we interpolate the flux using discontinuous linear interpolation scheme with 4 nodes. The elements are shown in Fig. 1. By using discontinuous interpolation we avoid flux definition problems in corners and edges. Details of the interpolation scheme are given in Ravnik et al. [19]. A function, u, is interpolated over a boundary elements as $u = \sum \varphi_i u_i$, inside each domain element as $u = \sum \Phi_i u_i$, while flux is interpolated over boundary elements as $q = \sum \phi_i q_i$. According to (21) the following integrals must be calculated:

$$[H] = \int_\Gamma \varphi_i \vec{\nabla} u^\star \cdot \vec{n}\, d\Gamma, \quad [G] = \int_\Gamma \phi_i u^\star \, d\Gamma, \tag{23}$$

$$[\vec{A}] = \int_\Gamma \varphi_i \vec{n} u^\star \, d\Gamma, \quad [\vec{D}] = \int_\Omega \Phi_i \vec{\nabla} u^\star \, d\Omega. \tag{24}$$

The square brackets denote integral matrices. In order to calculate the integrals, a Gaussian quadrature algorithm is used. The integrals are calculated in local coordinate system via weighted summation of 8 integration points per coordinate axis. Eight integration points were found sufficient by Ravnik et al. [19]. A polar coordinate system transformation is used to handle weakly singular integrals. Calculation of the free coefficient $c(\vec{\xi})$ is

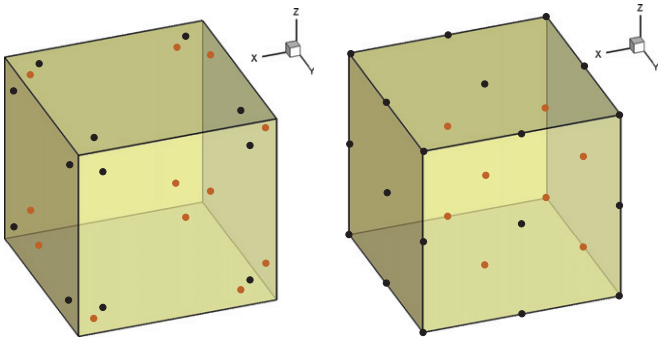

Fig. 1. The $1 \times 1 \times 1$ mesh. The figure on the right shows the location of 27 nodes for the interpolation of the function, while the figure on the left shows the 6×4 nodes for interpolation of flux across the six boundary elements.

performed indirectly via the solution of the rigid body movement problem.

Each source point location yields one row in these matrices. The source point is set in accordance with boundary conditions. For each boundary element, the source point is set into all nine function nodes in case of Neumann boundary conditions and into all four flux nodes in case of Dirichlet boundary conditions. Furthermore, since domain values of the function u are also unknown, the source point has to be set into all domain nodes as well. Discrete version of Eq. (21) is thus

$$[H]\{\alpha u\} = [G]\{\alpha q\} - [\vec{A}] \cdot \{\vec{v} u\} + [\vec{D}] \cdot \left\{ u \left(\vec{\nabla}\alpha + \vec{v} - \frac{\alpha}{\alpha_0} \vec{v}_0 \right) \right\}, \quad (25)$$

where $q = \vec{n} \cdot \vec{\nabla} u$ and curly brackets denote vectors of nodal values. All matrices are fully populated. The number of rows in matrices is equal to the number of all source points, i.e. the number of function nodes plus the number of flux nodes in the whole domain $n+n_q$. The number of columns in boundary matrices is equal to the number of boundary function nodes N_u or the number of boundary flux nodes N_q. The number of columns of the domain matrices is n. Thus, the total memory requirements are $(n+n_q)(4N_u+N_q+3n)$.

After application of boundary conditions, the resulting system of linear equations is solved by LU decomposition. The problem is linear, thus solution of (25) yields the u and q without the need of an iterative procedure. In contrast to the previous integral formulation of such problems, which featured the gradient of u in the domain integral, an iterative procedure was needed, alternatively calculating the solution and the gradient of u until convergence was achieved.

In order to calculate the fundamental solution and its gradient, the constant parts of the coefficient α_0 and velocity $\vec{v}_0$ must be chosen. In BDIE we used values at the source point location, i.e. $\alpha_0 = \alpha(\vec{\xi})$ and $\vec{v}_0 = \vec{v}(\vec{\xi})$.

4.2. Domain decomposition approach—SDBDIE

In this approach we keep the domain mesh and consider each mesh element as a subdomain. Within each subdomain standard approach (BDIE) is used. Between neighbouring subdomains, which share some of the nodes, compatibility conditions are prescribed, i.e. the function value at the node, which is shared by several subdomains, is equal for all subdomains. For two subdomains, which share a face, the flux through this face has the same value but opposite sign. Compatibility conditions lead to an over-determined system of equations, since the number of unknowns is smaller than the number of equations. The over-determined system is solved in a least squares manner, using the Paige and Saunders [12] LSQR solver with diagonal preconditioning.

The source point is set into all function and flux nodes, thus the total number of rows is equal to the number of subdomains (mesh elements) times 51 (=27+24). The number of columns is 26 for $[H]$ and $[\vec{A}]$ matrices, since there are 26 function nodes on the surface of each subdomain. The matrix $[G]$ has 24 columns, since there are 24 flux nodes on the surface of each subdomain. Matrices $[\vec{D}]$ have 27 columns since there are 27 nodes in each subdomain. Further details of the SDBDIE including the properties of the over-determined system are given in Ravnik et al. [19].

Now, we estimate the storage requirements of BDIE and SDBDIE. Consider a cube, meshed by x^3 mesh elements. In the case of BDIE, the number of elements of domain matrices scales approximately with the square of the number of function nodes in the mesh, that is with $(2x+1)^6$. In the case of SDBDIE, the number of elements is equal to $51 \cdot x^3 \cdot 27$. This makes the ratio of storage requirements equal to

$$\frac{\text{SDBDIE}}{\text{BDIE}} \approx \frac{51 \cdot x^3 \cdot 27}{(2x+1)^6} \approx \frac{21}{x^3}. \quad (26)$$

For $x=8$ the SDBDIE storage requirements are only 4% of BDIE, while at $x=16$ this is already at 0.5%.

In order to calculate the fundamental solution and its gradient, we used coefficient value at the source point location, i.e. $\alpha_0 = \alpha(\vec{\xi})$. Constant part of velocity, $\vec{v}_0$, was estimated by calculating the average velocity in the subdomain.

5. Numerical examples

In order to prove the validity of the integral formulation proposed in Eq. (21) and to compare BDIE and SDBDIE, we performed several numerical examples.

5.1. Setup

Table 1 lists 14 examples. Case number, coefficient, velocity and the analytical solution are presented. 1D, 2D and 3D cases are considered having different coefficients, velocities and analytical solutions.

Dirichlet boundary conditions were prescribed on all walls in 3D cases. In 2D cases, Neumann zero flux boundary condition was used in the direction perpendicular to the solution plane in order to simulate a 2D problem in a 3D solution domain. In the same way, in 1D cases, four walls had Neumann zero flux boundary conditions, which enable simulation of a 1D problem in a 3D domain.

The domain was a unit cube located at $[(1,1,1) \times (2,2,2)]$. The cube was meshed with 1, 2^3, 4^3, 8^3, 16^3 and 32^3 domain elements having 3^3, 5^3, 9^3, 17^3, 33^3, 65^3 nodes. All elements were identical cubes, no concentration in areas of high gradients was used. In BDIE only meshes up to 8^3 elements were used, since memory storage requirements for 16^3 and 32^3 meshes were too large.

In order to estimate the simulation error, the root mean square norm was used. It is based on the difference between the simulation result u and the analytical solution u_a as

$$\|u-u_a\|_{RMS} = \left(\frac{\sum_{i=1}^{n} (u_i - u_{a,i})^2}{\sum_{i=1}^{n} u_{a,i}^2} \right)^{1/2}, \quad (27)$$

where i denotes a nodal value and n is the number of all nodes in the domain.

5.2. Results

The results are presented in terms of RMS error versus number of nodes in the mesh. Figs. 2–6 display the appropriate graphs. The 1D test cases are shown in Fig. 2. The simulation was performed at Péclet number $Pe=10$. Both methods show good convergence properties for all three test cases. Since the first mesh has only one element and only one internal node, the RMS calculation is biased by the analytically prescribed boundary conditions, and the resulting RMS difference seems very accurate.

Test 2 was solved for different Péclet number values. Fig. 3 displays the simulation error for $Pe=1$, $Pe=2$ and $Pe=10$. In all cases for both methods we see an increase of solution accuracy when increasing the mesh density. The accuracy decreases when increasing the Péclet number. This was expected, since higher Péclet number means higher velocity and higher gradients within the solution. Since our meshes were not concentrated in regions where high gradients are expected, it is reasonable, that the solution accuracy decreases when increasing the Péclet number.

Table 1
Numerical examples solved by BDIE and SDBDIE in order to prove the validity of the integral formulation (21) and compare the methods. 3D cubical domain located at $[(1,1,1)\times(2,2,2)]$ is used in all cases. Dirichlet boundary conditions are employed. Péclet number for test 1,2 and 3 was set to $Pe=1$, $Pe=2$ and $Pe=10$.

Case	Coefficient $\alpha(\vec{r})$	Velocity $\vec{v}(\vec{r})$	Solution $u_a(\vec{r})$
1D cases			
1	x	$(Pe,0,0)$	$1-x^{Pe}$
2	x^2	$(Pe,0,0)$	$\exp(-\frac{Pe}{x})$
3	$\sin(x)$	$(Pe,0,0)$	$\left(\tan\frac{x}{2}\right)^{Pe}$
2D cases			
4	$x+y$	$(1,1,0)$	xy
5	$\sqrt{x+y}(2\sqrt{x+y}-x+y)$	$(1,1,0)$	$\sqrt{x+y}$
6	$\frac{x^2+y^2}{x+y}$	$(1,1,0)$	$(x+y)^2$
7	$\frac{2x^3+6xy^2+3(-x+y)}{3(x+y)^2}$	$(1,1,0)$	$(x+y)^3$
3D cases			
8	$x+y+z$	$(1,1,1)$	xyz
9	$\sqrt{x+y+z}(2\sqrt{x+y+z}-x+y)$	$(1,1,1)$	$\sqrt{x+y+z}$
10	$\frac{-3x^2+6xy+6xz+2(z-x)}{2(x+y+z)}$	$(1,1,1)$	$(x+y+z)^2$
11	$\frac{1}{3}(x+y+z)$	$(1,1,1)$	$(x+y+z)^3$
12	$x(5+x-2y+z)$	$(x,5-2y,z)$	$x+y+z$
13	$\frac{-5x^2-x^3+10xy+4x^2y-4xy^2+10xz+x^2z-2xyz+2xz^2}{2(x+y+z)}$	$(x,5-2y,z)$	$(x+y+z)^2$
14	$\frac{1}{9}(5+x-2y+z)(x+y+z)$	$(x,5-2y,z)$	$(x+y+z)^3$

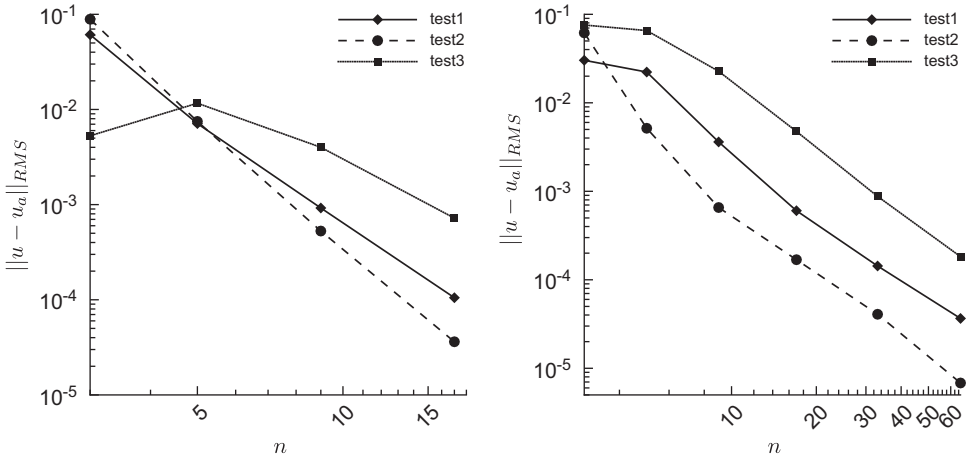

Fig. 2. RMS norms versus number of nodes for 1D test cases simulated with Péclet number of $Pe=10$. BDIE (left), SDBDIE (right).

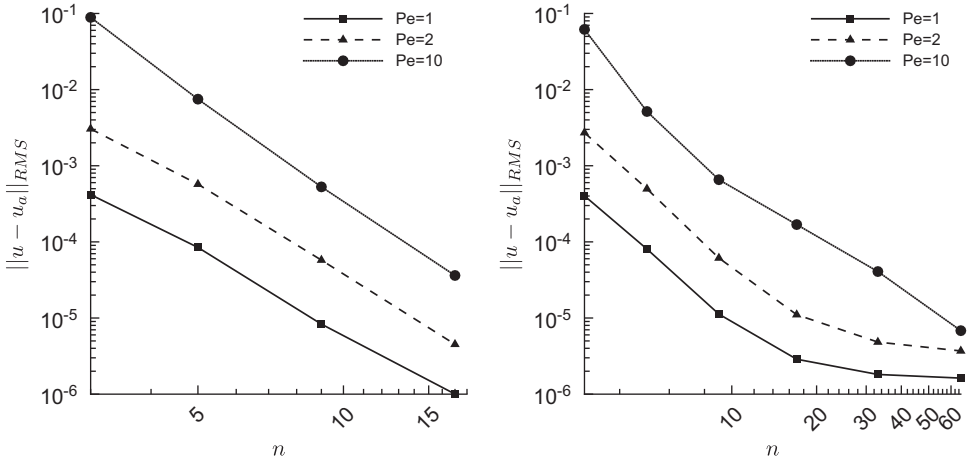

Fig. 3. Test 2 solved for three values of Péclet number. BDIE (left), SDBDIE (right).

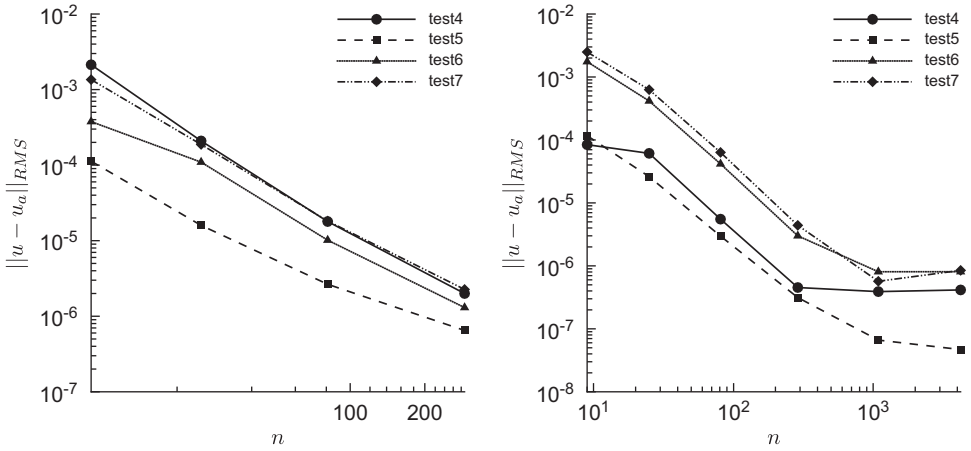

Fig. 4. RMS norms versus number of nodes for 2D test cases. BDIE (left), SDBDIE (right).

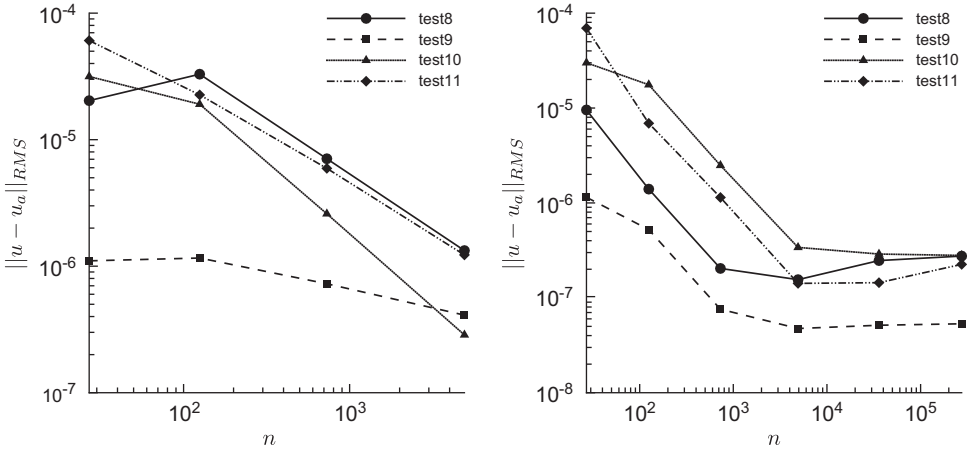

Fig. 5. RMS norms versus number of nodes for 3D test cases with constant velocity field. BDIE (left), SDBDIE (right).

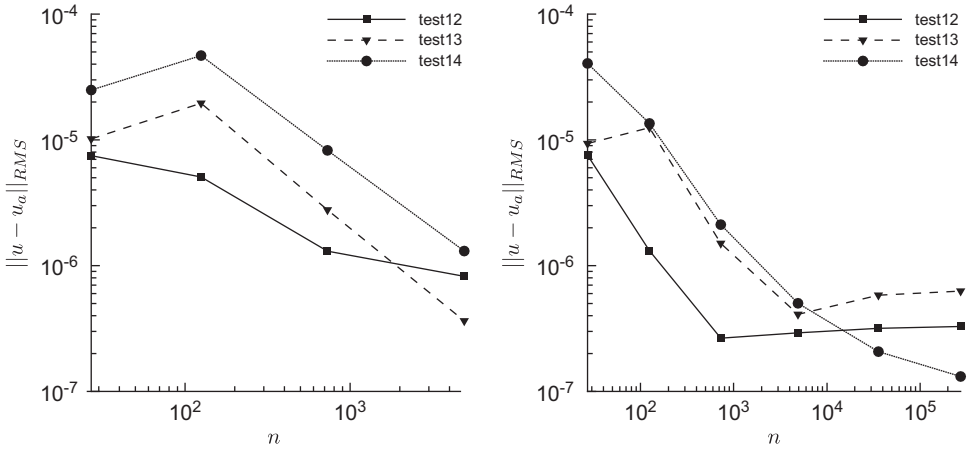

Fig. 6. RMS norms versus number of nodes for 3D test cases with spatially varying velocity field. BDIE (left), SDBDIE (right).

Fig. 4 displays results of the 2D test cases. We observe good convergence for both methods. The SDBDIE reaches 10^{-6} accuracy at 16^3 mesh. No improvement of solution is observed at 32^3 mesh. The reason for this is the fact, that the solver precision and the precision of calculation of integrals is reached, and thus increase of mesh density does not improve the solution.

Solution accuracy of 3D test cases with constant velocity field are shown in Fig. 5. High order of accuracy is reached by both methods. The most accurate results are obtained with test 9, while tests 8, 10 and 11 have lower accuracy. This is due to the fact that the solution of test 9 is simpler, while the solutions of other tests include higher order terms, which can not be accurately described by our interpolation scheme.

Finally, Fig. 6 presents the accuracy of 3D test cases with variable velocity field. Similar conclusions can be drawn as for other cases. Good convergence is observed, all cases are solved with RMS norm $< 10^{-6}$, when sufficient number of nodes are used.

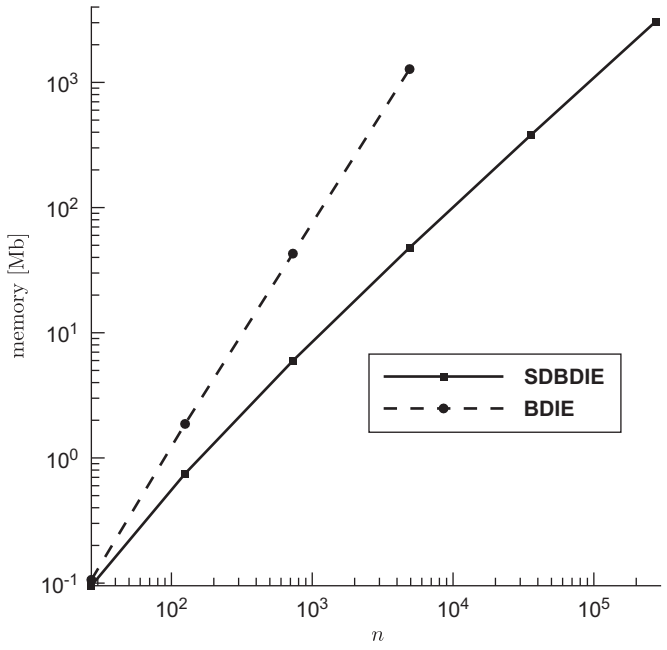

Fig. 7. Computer memory required by both approaches.

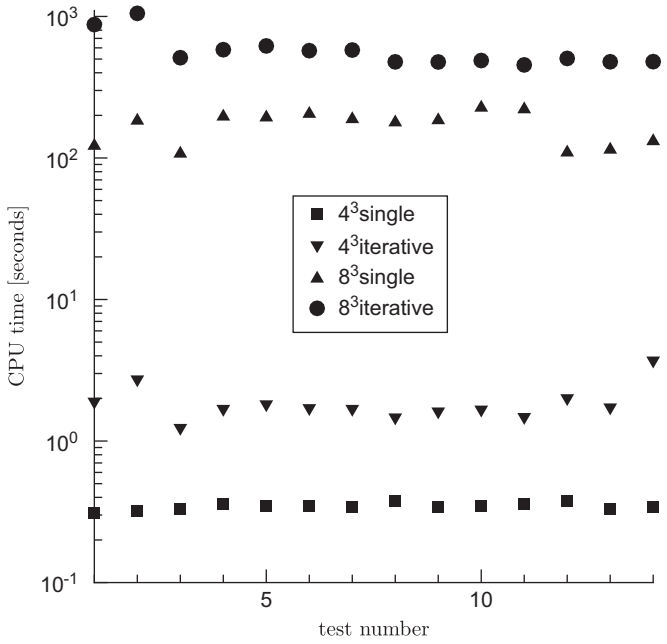

Fig. 8. CPU time for solution of individual tests solved by BDIE. Time for integration of integrals is not included. Single refers to solution proposed in present work, iterative refers to iterative solution, which was required by the previous formulation of the integral equation. The meshes of 4^3 and 8^3 elements are considered.

Since the sizes in our meshes decrease by a factor of 2, we were able to use the Richardson extrapolation to estimate the order of our methods. The order of methods is calculated as the log of the RMS norm at $4\times4\times4$ mesh over the RMS norm at $8\times8\times8$ mesh divided by the log of 2, i.e.

$$\mathcal{O}=\frac{1}{\log 2}\log\left(\frac{\|u-u_a\|_{RMS,4\times4\times4}}{\|u-u_a\|_{RMS,8\times8\times8}}\right). \tag{28}$$

Calculating the average value for all test cases, we obtain $\mathcal{O}=2.7$ for BDIE and $\mathcal{O}=2.4$ for SDBDIE, establishing more than second order accuracy for the proposed methods.

Fig. 7 displays computer memory requirements for BDIE and SDBDIE. In terms of computer memory requirements, SDBDIE uses only a fraction of the memory required by the BDIE. This is due to the sparse versus full integral matrix and has been foreseen by Eq. (26).

The integral equation (21) enables the solution of the proposed problems in a single iteration. Previous formulations featured a gradient of the function in the domain integral and thus required the domain integral to be put on the right hand side of the system of equations. In this case an iterative procedure must be set up in which the function and its gradient are alternatively calculated until convergence is achieved. For CPU time comparison, we also set up the solution of (21) in an iterative way, with the domain integral on the right hand side of the system. No under-relaxation was needed to get converged solutions of our tests. In terms of accuracy, the single iteration and iterative approaches produce identical results. In Fig. 8 we compare the CPU time needed for the solution of individual tests by BDIE. The single iteration solution based on (21) is compared with the iterative solution, in which the domain integral is on the right hand side. The iterative solution needed on average 14 iterations to converge. Fig. 8 shows that the newly proposed integral formulation (21) enables almost an order of magnitude faster computation as compared to the iterative approach, which is required by the old formulation. This is a direct result of the fact that the iterative approach needs on average 14 solutions of a system of linear equations, while the integral equation (21) can be solved by only a single solution of a system of linear equations.

6. Summary

In this work we derived a boundary-domain integral diffusion–convection equation with variable coefficient and variable velocity field. The equation does not include the gradient of the unknown function, which is the main advantage of the present formulation against the formulations developed by other authors. The formulation enables the solution of the diffusion–convection equation with variable coefficient and variable velocity field by a single solution of a system of linear equations.

Two discretization approaches have been developed to prove the validity of the formulation. The standard approach, in which integrals are calculated and their values stored in fully populated matrices, and a sub-domain approach, where a domain decomposition technique is used, which yields sparse matrices of integrals. Using numerical tests, both approaches were found to yield similar accuracy. The memory requirements were smaller in the case of the domain decomposition.

References

[1] AL-Jawary MA, Ravnik J, Wrobel LC, Škerget L. Boundary element formulations for numerical solution of two-dimensional diffusion problems with variable coefficients. Comput Math Appl 2012;64:2695–711.
[2] AL-Jawary MA, Wrobel LC. Numerical solution of two-dimensional mixed problems with variable coefficients by the boundary-domain integral and integro-differential equation methods. Eng Anal Boundary Elem 2011;35:1279–87.
[3] AL-Jawary MA, Wrobel LC. Radial integration boundary integral and integro-differential equation methods for two-dimensional heat conduction problems with variable coefficients. Eng Anal Boundary Elem 2012;36:685–95.
[4] DeSilva SJ, Chan CL, Chandra A, Lim J. Boundary element method analysis for the transient conduction-convection in 2-D with spatially variable convective velocity. Appl Math Model 1998;22(1–2):81–112.
[5] Divo EA, Kassab AJ. Boundary element methods for heat conduction: with application in non-homogenous media. WIT Press; 2003.
[6] Driessen BJ, Dohner JL. A finite element boundary element method for advection diffusion problems with variable advective fields and infinite domains. Int J Heat Mass Transfer 2001;44:2183–91.

[7] Fata SN. Treatment of domain integrals in boundary element methods. Appl Numer Math 2012;62:720–35.
[8] Hackbusch W, Nowak ZP. On the fast multiplication in the boundary element method by panel clustering. Numer Math 1989;54:463–91.
[9] Maerten F. Adaptive cross-approximation applied to the solution of system of equations and post-processing for 3D elastostatic problems using the boundary element method. Eng Anal Boundary Elem 2010;34:483–91.
[10] Mikhailov SE. Localized boundary-domain integral formulations for problems with variable coefficients. Eng Anal Boundary Elem 2002;26:681–90.
[11] Mikhailov SE, Nakhova IS. Mesh-based numerical implementation of the localized boundary domain integral equation method to a variable-coefficient Neumann problem. J Eng Math 2005;51:251–9.
[12] Paige CC, Saunders MA. LSQR: an algorithm for sparse linear equations and sparse least squares. ACM Trans Math Software 1982;8:43–71.
[13] Partridge PW, Brebbia CA, Wrobel LC. The dual reciprocity boundary element method. Computational Mechanics Publications Southampton, UK, Boston: Computational Mechanics Publications; London; New York; 1992.
[14] Popov V, Power H, Škerget L. Domain decomposition techniques for boundary elements application to fluid flow. Southampton, Boston: WIT Press; 2007.
[15] Popov V, Power H, Walker SP. Numerical comparison between two possible multipole alternatives for the BEM solution of 3D elasticity problems based upon Taylor series expansions. Eng Anal Boundary Elem 2003;27:521–31.
[16] Qiu ZH, Wrobel L, Power H. Numerical solution of convection–diffusion problems at high Peclet number using boundary elements. Int J Numer Methods Eng 1998;41:899–914.
[17] Rap A, Elliott L, Ingham DB, Lesnic D, Wen X. DRBEM for Cauchy convection–diffusion problems with variable coefficients. Eng Anal Boundary Elem 2004;28(11):1321–33.
[18] Ravnik J, Škerget L, Hriberšek M. The wavelet transform for BEM computational fluid dynamics. Eng Anal Boundary Elem 2004;28:1303–14.
[19] Ravnik J, Škerget L, Žunič Z. Combined single domain and subdomain BEM for 3D laminar viscous flow. Eng Anal Boundary Elem 2009;33:420–4.
[20] Ravnik J, Škerget L, Žunič Z. Comparison between wavelet and fast multipole data sparse approximations for Poisson and kinematics boundary—domain integral equations. Comput Methods Appl Mech Eng 2009;198:1473–85.
[21] Škerget L, Ravnik J. BEM simulation of compressible fluid flow in an enclosure induced by thermoacoustic waves. Eng Anal Boundary Elem 2009;33:561–71.
[22] Škerget L, Žagar I, Alujevič A. Three-dimensional steady-state diffusion–convection. In: Boundary elements IX, vol. 3. Berlin: Springer; 1987.
[23] Žagar I, Škerget L, Alujevič A. Diffusion–convection problems using boundary-domain integral formulation for non-uniform flows. In: Boundary element method in fluid dynamics II. Southampton: Computational Mechanics Publications; 1994.
[24] Wrobel LC. The boundary element method. John Willey & Sons, LTD; 2002.
[25] Wrobel LC, DeFigueiredo DB. Numerical analysis of convection–diffusion problems using the boundary element method. Int J Numer Methods Heat Fluid Flow 1991;1:3–18.
[26] Yang K, Gao X-W. Radial integration BEM for transient heat conduction problems. Eng Anal Boundary Elem 2010;34:557–63.