Numer Algor (2014) 65:275–291
DOI 10.1007/s11075-013-9711-1

ORIGINAL PAPER

Solving heat conduction problems by the Direct Meshless Local Petrov-Galerkin (DMLPG) method

Davoud Mirzaei · Robert Schaback

Received: 8 August 2012 / Accepted: 3 April 2013 / Published online: 28 April 2013

Abstract As an improvement of the *Meshless Local Petrov–Galerkin (MLPG)*, the *Direct Meshless Local Petrov–Galerkin (DMLPG)* method is applied here to the numerical solution of transient heat conduction problem. The new technique is based on *direct* recoveries of test functionals (local weak forms) from values at nodes without any detour via classical moving least squares (MLS) shape functions. This leads to an absolutely cheaper scheme where the numerical integrations will be done over low–degree polynomials rather than complicated MLS shape functions. This eliminates the main disadvantage of MLS based methods in comparison with finite element methods (FEM), namely the costs of numerical integration.

Keywords Generalized moving least squares (GMLS) approximation · Meshless methods · MLPG methods · DMLPG methods · Heat conduction problem

1 Introduction

Meshless methods have received much attention in recent decades as new tools to overcome the difficulties of mesh generation and mesh refinement in classical mesh-based methods such as the finite element method (FEM) and the finite volume method (FVM).

D. Mirzaei (✉)
Department of Mathematics, University of Isfahan, 81745-163 Isfahan, Iran
e-mail: d.mirzaei@sci.ui.ac.ir

R. Schaback
Institut für Numerische und Angewandte Mathematik, Universität Göttingen, Lotzestraße 16-18, 37073 Göttingen, Germany
e-mail: schaback@math.uni-goettingen.de

The classification of numerical methods for solving PDEs should always start from the classification of PDE problems themselves into *strong*, *weak*, or *local weak* forms. The first is the standard pointwise formulation of differential equations and boundary conditions, the second is the usual weak form dominating all FEM techniques, while the third form splits the integrals of the usual global weak form into local integrals over many small subdomains, performing the integration by parts on each local integral. Local weak forms are the basis of all variations of the *Meshless Local Petrov–Galerkin* technique (MLPG) of S.N. Atluri and collaborators [1]. This classification is dependent on the PDE problem itself, and independent of numerical methods and the trial spaces used. Note that these three formulations of the "same" PDE and boundary conditions lead to three essentially different mathematical problems that cannot be identified and need a different mathematical analysis with respect to existence, uniqueness, and stability of solutions.

Meshless *trial spaces* mainly come via *Moving Least Squares* or *kernels* like *Radial Basis Functions*. They can consist of *global* or *local* functions, but they should always parametrize their trial functions "*entirely in terms of nodes*" [3, 15] and require no triangulation or meshing.

A third classification of PDE methods addresses where the discretization lives. *Domain type* techniques work in the full global domain, while *boundary type* methods work with exact solutions of the PDE and just have to care for boundary conditions. This is independent of the other two classifications.

Consequently, the literature should confine the term "meshless" to be a feature of *trial spaces*, not of PDE problems and their various formulations. But many authors reserve the term *truly meshless* for meshless methods that either do not require any discretization with a background mesh for calculating integrals or do not require integration at all. These techniques have a great advantage in computational efficiency, because numerical integration is the most time–consuming part in all numerical methods based on local or global weak forms. This paper focuses on a truly meshless method in this sense.

Most of the methods for solving PDEs in global weak form, such as the Element-Free Galerkin (EFG) method [4], are not *truly meshless* because a triangulation is still required for numerical integration. The *Meshless Local Petrov-Galerkin* (MLPG) method solves PDEs in local weak form and uses no global background mesh to evaluate integrals because everything breaks down to some regular, well-shaped and independent sub-domains. Thus the MLPG is known as a truly meshless method.

We now focus on meshless methods using Moving Least Squares as trial functions. If they solve PDEs in global or local weak form, they still suffer from the cost of numerical integration. In these methods, numerical integrations are traditionally done over MLS shape functions and their derivatives. Such shape functions are complicated and have no closed form. To get accurate results, numerical quadratures with many integration points are required. Thus the MLS subroutines must be called very often, leading to high computational costs. In contrast to this, the stiffness matrix in finite element methods (FEMs) is constructed by integrating over polynomial basis functions which are much cheaper to evaluate. This relaxes the cost of numerical integrations. For an account of the importance of numerical integration within meshless methods, we refer the reader to [2].

To overcome this shortage within the MLPG based on MLS, Mirzaei and Schaback [8] proposed a new technique, *Direct Meshless Local Petrov-Galerkin (DMLPG) method*, which avoids integration over MLS shape functions in MLPG and replaces it by the much cheaper integration over polynomials. It ignores shape functions completely. Altogether, the method is simpler, faster and often more accurate than the original MLPG method. DMLPG uses a generalized MLS (GMLS) method of [9] which directly approximates boundary conditions and local weak forms as some *functionals*, shifting the numerical integration into the MLS itself, rather than into an outside loop over calls to MLS routines. Thus the concept of GMLS must be outlined first in Section 2 before we can go over to the DMLPG in Section 4 and numerical results for heat conduction problems in Section 6.

The analysis of heat conduction problems is important in engineering and applied mathematics. Analytical solutions of heat equations are restricted to some special cases, simple geometries and specific boundary conditions. Hence, numerical methods are unavoidable. Finite element methods, finite volume methods, and finite difference methods have been well applied to transient heat analysis over the past few decades [5]. MLPG methods were also developed for heat transfer problems in many cases. For instance, J. Sladek et al. [12] proposed MLPG4 for transient heat conduction analysis in functionally graded materials (FGMs) using Laplace transform techniques. V. Sladek et al. [13] developed a local boundary integral method for transient heat conduction in anisotropic and functionally graded media. Both authors and their collaborators employed MLPG5 to analyze the heat conduction in FGMs [10, 11].

The aim of this paper is the development of DMLPG methods for heat conduction problems. This is the first time where DMLPG is applied to a time–dependent problem. Moreover, compared to [8], we will discuss all DMLPG methods, go into more details and provide explicit formulae for the numerical implementation. DMLPG1/2/4/5 will be proposed, and the reason of ignoring DMLPG3/6 will be discussed. The new methods will be compared with the original MLPG methods in a test problem, and then a problem in FGMs will be treated by DMLPG1.

In all application cases, the DMLPG method turned out to be superior to the standard MLPG technique, and it provides excellent accuracy at low cost.

2 Meshless methods and GMLS approximation

Whatever the given PDE problem is and how it is discretized, we have to find a function u such that M linear equations

$$\lambda_k(u) = \beta_k, \ 1 \leqslant k \leqslant M, \tag{1}$$

defined by M linear *functionals* $\lambda_1, \ldots, \lambda_M$ and M prescribed real values $\beta_1, \ldots, \beta_M$ are to be satisfied. Note that weak formulations will involve functionals that integrate u or a derivative against some test function. The functionals can discretize either the differential equation or some boundary condition.

Now *meshless methods* construct solutions from a *trial space* whose functions are parametrized "*entirely in terms of nodes*" [3]. We let these nodes form a set $X := \{x_1, \ldots, x_N\}$. Theoretically, meshless trial functions can then be written as

linear combinations of *shape functions* $u_1, \ldots, u_N$ with or without the Lagrange conditions $u_j(x_k) = \delta_{jk},\ 1 \leqslant j, k \leq N$ as

$$u(x) = \sum_{j=1}^{N} u_j(x) u(x_j)$$

in terms of values at nodes, and this leads to solving the system (1) in the form

$$\lambda_k(u) = \sum_{j=1}^{N} \lambda_k(u_j) u(x_j) = \beta_k,\ 1 \leqslant k \leqslant M$$

approximately for the nodal values. Setting up the coefficient matrix requires the evaluation of all functionals on all shape functions, and this is a tedious procedure if the shape functions are not cheap to evaluate, and it is even more tedious if the functionals consist of integrations of derivatives against test functions.

But it is by no means mandatory to use shape functions at this stage at all. If each functional λ_k can be well approximated by a formula

$$\lambda_k(u) \approx \sum_{j=1}^{N} \alpha_{jk} u(x_j) \tag{2}$$

in terms of nodal values for smooth functions u, the system to be solved is

$$\sum_{j=1}^{N} \alpha_{jk} u(x_j) = \beta_k,\ 1 \leqslant k \leqslant M \tag{3}$$

without any use of shape functions. There is no trial space, but everything is still written in terms of values at nodes. Once the approximate values $u(x_j)$ at nodes are obtained, any multivariate interpolation or approximation method can be used to generate approximate values at other locations. This is a postprocessing step, independent of PDE solving.

This calls for efficient ways to handle the approximations (2) to functionals in terms of nodal values. We employ a generalized version of Moving Least Squares (MLS), adapted from [9], and without using shape functions.

The techniques of [9] and [8] allow to calculate coefficients α_{jk} for (2) very effectively as follows. We fix k and consider just $\lambda := \lambda_k$. Furthermore, the set X will be formally replaced by a much smaller subset that consists only of the nodes that are locally necessary to calculate a good approximation of λ_k, but we shall keep X and N in the notation. This reduction of the node set for the approximation of λ_k will ensure sparsity of the final coefficient matrix in (3).

Now we have to calculate a coefficient vector $a(\lambda_k) = (\alpha_{1k}, \ldots, \alpha_{Nk})^T \in \mathbb{R}^N$ for (2) in case of $\lambda = \lambda_k$. We choose a space $\mathcal{P}$ of polynomials which is large enough to let zero be the only polynomial p in $\mathcal{P}$ that vanishes on X. Consequently, the dimension Q of $\mathcal{P}$ satisfies $Q \leqslant N$, and the $Q \times N$ matrix P of values $p_i(x_j)$ of a basis $p_1, \ldots, p_Q$ of $\mathcal{P}$ has rank Q. Then for any vector $w = (w_1, \ldots, w_N)^T$ of positive weights, the generalized MLS solution $a(\lambda)$ to (2) can be written as

$$a(\lambda_k) = W P^T (P\, W\, P^T)^{-1} \lambda_k(\mathcal{P}) \tag{4}$$

where W is the diagonal matrix with diagonal w and $\lambda_k(\mathcal{P}) \in \mathbb{R}^Q$ is the vector with values $\lambda_k(p_1), \ldots, \lambda_k(p_Q)$.

Thus it suffices to evaluate λ_k on low–order polynomials, and since the coefficient matrix in (4) is independent of k, one can use the same matrix for different λ_k as long as X does not change locally. This will significantly speed up numerical calculations, if the functional λ_k is complicated, e.g. a numerical integration against a test function. Note that the MLS is just behind the scene, no shape functions occur. But the weights will be defined locally in the same way as in the usual MLS, e.g. we choose a continuous function $\phi : [0, \infty) \to [0, \infty)$ with

- $\phi(r) > 0,\ 0 \leqslant r < 1,$
- $\phi(r) = 0,\ r \geqslant 1,$

and define

$$w_j(x) = \phi\left(\frac{\|x - x_j\|_2}{\delta}\right)$$

for $\delta > 0$ as a weight function, if we work locally near a point x.

3 MLPG formulation of heat conduction

In the Cartesian coordinate system, the transient temperature field in a heterogeneous isotropic medium is governed by the diffusion equation

$$\rho(x)c(x)\frac{\partial u}{\partial t}(x,t) = \nabla \cdot (\kappa \nabla u) + f(x,t), \tag{5}$$

where $x \in \Omega$ and $0 \leqslant t \leqslant t_F$ denote the space and time variables, respectively, and t_F is the final time. The initial and boundary conditions are

$$u(x,0) = u_0(x), \quad x \in \Omega, \tag{6}$$

$$u(x,t) = u_D(x,t), \quad x \in \Gamma_D, \quad 0 \leqslant t \leqslant t_F, \tag{7}$$

$$\kappa(x)\frac{\partial u}{\partial n}(x,t) = u_N(x,t), \quad x \in \Gamma_N, \quad 0 \leqslant t \leqslant t_F. \tag{8}$$

In (5)–(8), $u(x,t)$ is the temperature field, $\kappa(x)$ is the thermal conductivity dependent on the spatial variable x, $\rho(x)$ is the mass density and $c(x)$ is the specific heat, and $f(x,t)$ stands for the internal heat source generated per unit volume. Moreover, n is the unit outward normal to the boundary Γ, u_D and u_N are specified values on the Dirichlet boundary Γ_D and Neumann boundary Γ_N where $\Gamma = \Gamma_D \cup \Gamma_N$.

Meshless methods write everything entirely in terms of scattered nodes forming a set $X = \{x_1, x_2, \ldots, x_N\}$ located in the spatial domain Ω and its boundary Γ. In the standard MLPG, around each x_k a small subdomain $\Omega_s^k \subset \overline{\Omega} = \Omega \cup \Gamma$ is chosen such that integrations over Ω_s^k are comparatively cheap. For instance, Ω_s^k is conveniently taken to be the intersection of $\overline{\Omega}$ with a ball $B(x_k, r_0)$ of radius r_0 or a cube (or a square in 2D) $S(x_k, r_0)$ centered at x_k with side-length r_0. On these subdomains, the PDE including boundary conditions is stated in a localized weak form

$$\frac{\partial}{\partial t}\int_{\Omega_s^k} \rho c u v \, d\Omega = \int_{\Omega_s^k} \nabla \cdot (\kappa \nabla u) v \, d\Omega + \int_{\Omega_s^k} f v \, d\Omega, \tag{9}$$

for an appropriate *test function* v. Applying integration by parts, this weak equation can be partially symmetrized to become the *first* local weak form

$$\frac{\partial}{\partial t}\int_{\Omega_s^k}\rho c u v\,d\Omega = \int_{\partial\Omega_s^k}\kappa\frac{\partial u}{\partial n}\,v\,d\Gamma - \int_{\Omega_s^k}\kappa\nabla u\cdot\nabla v\,d\Omega + \int_{\Omega_s^k} f v\,d\Omega. \tag{10}$$

The *second* local weak form, after rearrangement of (5) and integration by parts twice, can be obtained as

$$\begin{aligned}\frac{\partial}{\partial t}\int_{\Omega_s^k}\frac{1}{\kappa}\rho c u v\,d\Omega = \int_{\Omega_s^k} u\Delta v\,d\Omega - \int_{\partial\Omega_s^k} u\frac{\partial v}{\partial n}\,d\Gamma + \int_{\partial\Omega_s^k} v\frac{\partial u}{\partial n}\,d\Gamma \\ + \int_{\Omega_s^k}\frac{1}{\kappa}\nabla\kappa\cdot\nabla u\,v\,d\Omega + \int_{\Omega_s^k}\frac{1}{\kappa} f v\,d\Omega.\end{aligned} \tag{11}$$

If the boundary of the local domain Ω_s^k hits the boundary of Ω, the MLPG inserts boundary data at the appropriate places in order to care for boundary conditions. Since these local weak equations are all affine–linear in u even after insertion of boundary data, the equations of MLPG are all of the form (1) after some rearrangement, employing certain linear functionals λ_k. In all cases, the MLPG evaluates these functionals on shape functions, while our DMLPG method will use the GMLS approximation of Section 2 without any shape function.

However, different choices of test functions v lead to the six different well–known types of MLPG. The variants MLPG1/5/6 are based on the weak formulation (10). If v is chosen such that the first integral in the right hand side of (10) vanishes, we have MLPG1. In this case v should vanish on $\partial\Omega_s^k$. If the Heaviside step function v on local domains is used as test function, the second integral disappears and we have a pure local boundary integral form in the right hand side. This is MLPG5. In MLPG6, the trial and test functions come from the same space. MLPG2/3 are based on the local unsymmetric weak formulation (9). MLPG2 employs Dirac's delta function as the test function in each Ω_s^k, which leads to a pure collocation method. MLPG3 employs the error function as the test function in each Ω_s^k. In this method, the test functions can be the same as for the discrete least squares method. The test functions and the trial functions come from the same space in MLPG3. Finally, MLPG4 (or LBIE) is based on the weak form (11), and a modified fundamental solution of the corresponding elliptic spatial equation is employed as a test function in each subdomain.

We describe these types in more detail later, along with the way we modify them when going from MLPG to DMLPG.

4 DMLPG formulations

Independent of which variation of MLPG we go for, the DMLPG has its special ways to handle boundary conditions, and we describe these first.

Neither Lagrange multipliers nor penalty parameters are introduced into the local weak forms, because the Dirichlet boundary conditions are imposed directly. For nodes $x_k \in \Gamma_D$, the values $u(x_k,t) = u_D(x_k,t)$ are known from the Dirichlet

boundary conditions. To connect them properly to nodal values $u(x_j, t)$ in neighboring points x_j inside the domain or on the Neumann boundary, we turn the GMLS philosophy upside down and ask for coefficients $a_j(x_k)$ that allow to reconstruct nodal values at x_k from nodal values at the x_j. This amounts to setting $\lambda_k = \delta_{x_k}$ in Section 2 , and we get localized equations for Dirichlet boundary points x_k as

$$\sum_{j=1}^{N} a_j(x_k)u(x_j, t) = u_D(x_k, t), \quad x_k \in \Gamma_D, \quad t \in [0, t_F]. \tag{12}$$

Note that the coefficients are time–independent. In matrix form, (12) can be written as

$$B\boldsymbol{u}(t) = \boldsymbol{u}_D(t), \tag{13}$$

where $\boldsymbol{u}(t) \in \mathbb{R}^N$ is the time–dependent vector of nodal values at $x_1, x_2, ..., x_N$. These equations are added into the full matrix setup at the appropriate places, and they are in truly meshless form, since they involve only values at nodes and are without numerical integration. Note that (10) has no integrals over the Dirichlet boundary, and thus we can impose Dirichlet conditions always in the above strong form. For (11) there are two possibilities. We can impose the Dirichlet boundary conditions either in the local weak form or in the collocation form (12). Of course the latter is the cheaper one.

We now turn to Neumann boundary conditions. They can be imposed in the same way as Dirichlet boundary conditions by assuming $\lambda_k(u) = \frac{\partial u}{\partial n}(x_k)$ in the GMLS approximation

$$\sum_{j=1}^{N} a_j(x_k)u(x_j, t) = \frac{\partial u}{\partial n}(x_k, t), \quad x_k \in \Gamma_N, \quad t \in [0, t_F]. \tag{14}$$

Note that the coefficients again are time–independent, and we get a linear system like (13), but with a vector $\boldsymbol{u}_N(t)$ of nodal values of normal derivatives in the right–hand side. This is collocation as in subsection 4.2. But it is often more accurate to impose Neumann conditions directly into the local weak forms (10) and (11). We will describe this in more detail in the following subsections. We now turn the different variations of the MLPG method into variations of the DLMPG.

4.1 DMLPG1/5

These methods are based on the local weak form (10). This form recasts to

$$\begin{aligned} \frac{\partial}{\partial t}\int_{\Omega_s^k} \rho c u v \, d\Omega + \int_{\Omega_s^k} \kappa \nabla u \cdot \nabla v \, d\Omega - \int_{\partial\Omega_s^k \setminus \Gamma_N} \kappa \frac{\partial u}{\partial n} v \, d\Gamma \\ = \int_{\Gamma_N \cap \partial\Omega_s^k} u_N v \, d\Gamma + \int_{\Omega_s^k} f v \, d\Omega \end{aligned} \tag{15}$$

after inserting the Neumann boundary data from (8), when the domain Ω_s^k of (10) hits the Neumann boundary Γ_N. All integrals in the top part of (15) can be

efficiently approximated by GMLS approximation of Section 2 as purely spatial formulas

$$\begin{aligned}
\lambda_{1,k}(u) &:= \int_{\Omega_s^k} \rho c u v \, d\Omega && \approx \widehat{\lambda_{1,k}(u)} = \sum_{j=1}^{N} a_{1,j}(x_k) u(x_j), \\
\lambda_{2,k}(u) &:= -\int_{\Omega_s^k} \kappa \nabla u \cdot \nabla v \, d\Omega && \approx \widehat{\lambda_{2,k}(u)} = \sum_{j=1}^{N} a_{2,j}(x_k) u(x_j), \\
\lambda_{3,k}(u) &:= -\int_{\partial\Omega_s^k \setminus \Gamma_N} \kappa \frac{\partial u}{\partial n} v \, d\Gamma && \approx \widehat{\lambda_{3,k}(u)} = \sum_{j=1}^{N} a_{3,j}(x_k) u(x_j).
\end{aligned} \tag{16}$$

While the two others can always be summed up, the first formula, if applied to time–varying functions, has to be modified into

$$\frac{\partial}{\partial t} \int_{\Omega_s^k} \rho c u v \, d\Omega \approx \sum_{j=1}^{N} a_{1,j}(x_k) \frac{\partial}{\partial t} u(x_j, t)$$

and expresses the main PDE term not in terms of values at nodes, but rather in terms of time derivatives of values at nodes.

Again, everything is expressed in terms of values at nodes, and the coefficients are time–independent. Furthermore, Section 2 shows that the u part of the integration runs over low–order polynomials, not over any shape functions.

The third functional can be omitted if the test function v vanishes on $\partial\Omega_s^k \setminus \Gamma_N$. This is DMLPG1. An example of such a test function is

$$v = v(x; x_k) = \phi\left(\frac{\|x - x_k\|_2}{r_0}\right),$$

where ϕ is the weight function in the MLS approximation with the radius δ of the support of the weight function being replaced by the radius r_0 of the local domain Ω_s^k.

In DMLPG5, the local test function is the constant $v = 1$. Thus the functionals $\lambda_{2,k}$ of (16) are not needed, and the integrals for $\lambda_{1,k}$ take a simple form, if c and ρ are simple. DMLPG5 is slightly cheaper than DMLPG1, because the domain integrals of $\lambda_{2,k}$ are replaced by the boundary integrals of $\lambda_{3,k}$.

Depending on which parts of the functionals are present or not, we finally get a time–dependent system of the form

$$A^{(1)} \frac{\partial}{\partial t} \boldsymbol{u}(t) + A^{(\ell)} \boldsymbol{u}(t) = \boldsymbol{b}(t), \quad \ell = 2 \text{ or } 3 \tag{17}$$

where $\boldsymbol{u}(t)$ is the time–dependent vector

$$\boldsymbol{u}(t) = (u(x_1, t), \ldots, u(x_N, t))^T \in \mathbb{R}^N$$

of nodal values, $\boldsymbol{b}(t) \in \mathbb{R}^M$ collects the time–dependent right–hand sides with components

$$b_k = \int_{\Omega_s^k} f(x, t) v(x; x_k) \, d\Omega + \int_{\Gamma_N \cap \partial\Omega_s^k} u_N(x, t) v(x; x_k) \, d\Gamma,$$

and $A^{(\ell)}_{kj} = a_{\ell,j}(x_k)$, $\ell = 1, 2, 3$. The k-th row of $A^{(\ell)}$ is

$$\boldsymbol{a}^{(\ell)}_k = WP(PWP^T)^{-1}\lambda_{\ell,k}(\mathcal{P}), \quad \ell = 1, 2, 3,$$

where

$$\lambda_{1,k}(\mathcal{P}) = \left[\int_{\Omega^k_s} \rho c p_1 v \, d\Omega, \int_{\Omega^k_s} \rho c p_2 v \, d\Omega, \ldots, \int_{\Omega^k_s} \rho c p_Q v \, d\Omega\right]^T,$$

$$\lambda_{2,k}(\mathcal{P}) = -\left[\int_{\Omega^k_s} \kappa \nabla p_1 \cdot \nabla v \, d\Omega, \int_{\Omega^k_s} \kappa \nabla p_2 \cdot \nabla v \, d\Omega, \ldots, \int_{\Omega^k_s} \kappa \nabla p_Q \cdot \nabla v \, d\Omega\right]^T,$$

$$\lambda_{3,k}(\mathcal{P}) = \left[\int_{\partial\Omega^k_s \setminus \Gamma_N} \kappa \frac{\partial p_1}{\partial n} v \, d\Gamma, \int_{\partial\Omega^k_s \setminus \Gamma_N} \kappa \frac{\partial p_2}{\partial n} v \, d\Gamma, \ldots, \int_{\partial\Omega^k_s \setminus \Gamma_N} \kappa \frac{\partial p_Q}{\partial n} v \, d\Gamma\right]^T.$$

As we can immediately see, *numerical integrations are done over low-degree polynomials p_1, p_2, ..., p_Q only, and no shape function is needed at all. This reduces the cost of numerical integration in MLPG methods significantly.*

4.2 DMLPG2

In this method, the test function v on the local domain Ω^k_s in (9) is replaced by the test functional δ_{x_k}, i.e. we have strong collocation of the PDE and all boundary conditions. Depending on where x_k lies, one can have the functionals

$$\begin{aligned} \mu_{1,k}(u) &:= u(x_k), \\ \mu_{2,k}(u) &:= \frac{\partial u}{\partial n}(\rho c u)(x_k), \\ \mu_{3,k}(u) &:= \nabla \cdot (\kappa \nabla u)(x_k) \end{aligned} \tag{18}$$

connecting u to Dirichlet, Neumann, or PDE data. The first form is used on the Dirichlet boundary, and leads to (12) and (13). The second applies to points on the Neumann boundary and is handled by (14), while the third can occur anywhere in $\overline{\Omega}$ independent of the other possibilities. In all cases, the GMLS method of Section 2 leads to approximations of the form

$$\mu_{i,k}(u) \approx \sum_{j=1}^{N} a_{i,j}(x_k) u(x_j), \ i = 1, 2, 3$$

entirely in terms of nodes, where values on nodes on the Dirichlet boundary can be replaced by given data.

This DMLPG2 technique is a pure collocation method and requires no numerical integration at all. Hence it is truly meshless and the cheapest among all versions of DMLPG and MLPG. But it needs higher order derivatives, and thus the order of convergence is reduced by the order of the derivative taken. Sometimes DMLPG2 is called *Direct MLS Collocation (DMLSC) method* [8].

It is worthy to note that the recovery of a functional such as $\mu_{2,k}(u)$ or $\mu_{3,k}(u)$ in (18) using GMLS approximation gives *GMLS derivative approximation*. These kinds of derivatives have been comprehensively investigated in [9] and a rigorous

error bound was derived for them. Sometimes they are called *diffuse* or *uncertain* derivatives, because they are not derivatives of shape functions, but [9] proves there is nothing diffuse or uncertain about them and they are direct and usually very good numerical approximation of corresponding function derivatives.

4.3 DMLPG4

This method is based on the local weak form (11) and uses the *fundamental solution* of the corresponding elliptic spatial equation as test function. Here we describe it for a two–dimensional problem. To reduce the unknown quantities in local weak forms, the concept of *companion solutions* was introduced in [16]. The companion solution of a 2D Laplace operator is

$$v(x;y)=\frac{1}{2\pi}\ln\frac{r_0}{r},\quad r=\|x-y\|_2,$$

which corresponds to the Poisson equation $\Delta v(x;y)+\delta(r)=0$ and thus is a *fundamental* solution vanishing for $r=r_0$. Dirichlet boundary conditions for DMLPG4 are imposed as in (12). The resulting local integral equation corresponding to a node x_k located inside the domain or on the Neumann part of the boundary is

$$\begin{aligned}\frac{\partial}{\partial t}\int_{\Omega_s^k}\frac{1}{\kappa}\rho c u v\,d\Omega-\alpha_k u(x_k)+\oint_{\partial\Omega_s^k}\frac{\partial v}{\partial n}u\,d\Gamma-\int_{\Omega_s^k}\frac{1}{\kappa}\nabla\kappa\cdot\nabla u\,v\,d\Omega\\ =\int_{\partial\Omega_s^k\cap\Gamma_N}u_N v\,d\Gamma+\int_{\Omega_s^k}\frac{1}{\kappa}f v\,d\Omega,\end{aligned}\tag{19}$$

where α_k is a coefficient that depends on where the source point x_k lies. It is $1/2$ on the smooth boundary, and $\theta_k/(2\pi)$ at a corner where the interior angle at the point x_k is θ_k. The symbol $\oint$ represents the Cauchy principal value (CPV). For interior points x_k we have $\alpha_k=1$ and CPV integrals are replaced by regular integrals.

In this case

$$\lambda_{1,k}(\mathcal{P})=\left[\int_{\Omega_s^k}\frac{1}{\kappa}\rho c p_1 v\,d\Omega,\int_{\Omega_s^k}\frac{1}{\kappa}\rho c p_2 v\,d\Omega,\ldots,\int_{\Omega_s^k}\frac{1}{\kappa}\rho c p_Q v\,d\Omega\right]^T,$$

and $\lambda_{2,k}(\mathcal{P})=\alpha_k\lambda_{2,k}^{(1)}(\mathcal{P})+\lambda_{2,k}^{(2)}(\mathcal{P})+\lambda_{2,k}^{(3)}(\mathcal{P})$, where

$$\lambda_{2,k}^{(1)}(\mathcal{P})=\left[p_1(x_k),p_2(x_k),\ldots,p_Q(x_k)\right]^T,$$

$$\lambda_{2,k}^{(2)}(\mathcal{P})=-\left[\oint_{\Gamma_s^k}\frac{\partial v}{\partial n}p_1\,d\Gamma,\oint_{\Gamma_s^k}\frac{\partial v}{\partial n}p_2\,d\Gamma,\ldots,\oint_{\Gamma_s^k}\frac{\partial v}{\partial n}p_Q\,d\Gamma\right]^T,$$

$$\lambda_{2,k}^{(3)}(\mathcal{P})=\left[\int_{\Omega_s^k}\frac{1}{\kappa}\nabla\kappa\cdot\nabla p_1\,v\,d\Omega,\int_{\Omega_s^k}\frac{1}{\kappa}\nabla\kappa\cdot\nabla p_2\,v\,d\Omega,\ldots,\int_{\Omega_s^k}\frac{1}{\kappa}\nabla\kappa\cdot\nabla p_Q\,v\,d\Omega\right]^T.$$

Finally, we have the time-dependent linear system of equations

$$A^{(1)}\frac{\partial}{\partial t}\boldsymbol{u}(t)+A^{(2)}\boldsymbol{u}(t)=\boldsymbol{b}(t),\tag{20}$$

where the k-th row of $A^{(\ell)}$ is

$$\boldsymbol{a}_k^{(\ell)} = WP(PWP^T)^{-1}\lambda_{\ell,k}(\mathcal{P}), \quad \ell = 1, 2.$$

The components of the right-hand side are

$$b_k(t) = \int_{\Omega_s^k} \frac{1}{\kappa(x)} f(x,t)v(x;x_k)\,d\Omega + \int_{\partial\Omega_s^k \cap \Gamma_N} u_N(x,t)v(x;x_k)\,d\Gamma.$$

This technique leads to weakly singular integrals which must be evaluated by special numerical quadratures.

4.4 DMLPG3/6

In both MLPG3 and MLPG6, the trial and test functions come from the same space. Therefore they are Galerkin type techniques and should better be called MLG3 and MLG6. But they annihilate the advantages of DMLPG methods with respect to numerical integration, because the integrands include shape functions. Thus we ignore DMLPG3/6 in favour of keeping all benefits of DMLPG methods. Note that MLPG3/6 are also rarely used in comparison to the other MLPG methods.

5 Time stepping

To deal with the time variable in meshless methods, some standard methods were proposed in the literature. The Laplace transform method [10, 12], conventional finite difference methods such as forward, central and backward difference schemes are such techniques. A method which employs the MLS approximation in both time and space domains, is another different scheme [6, 7].

In our case the linear system (3) turns into the time–dependent version (17) coupled with (13) that could, for instance, be solved like any other linear first–order implicit Differential Algebraic Equations (DAE) system. Invoking an ODE solver on it would be an instance of the Method of Lines. If a conventional time–difference scheme such as a Crank-Nicolson method is employed, if the time step Δt remains unchanged, and if $M = N$, then a single LU decomposition of the final stiffness matrix and corresponding backward and forward substitutions can be calculated once and for all, and then the final solution vector at the nodes is obtained by a simple matrix–vector iteration.

The classical MLS approximation can be used as a postprocessing step to obtain the solution at any other point $x \in \Omega$.

6 Numerical results

Implementation is done using the basis polynomials

$$\left\{ \frac{(x-z)^\beta}{h^{|\beta|}} \right\}_{0 \leqslant |\beta| \leqslant m}$$

where h is an average mesh-size, and z is a fixed evaluation point such as a test point or a Gaussian point for integration in weak–form based techniques. Here $\beta = (\beta_1, \ldots, \beta_d) \in \mathbb{N}_0^d$ is a multi-index and $|\beta| = \beta_1 + \ldots + \beta_d$. If $x = (\chi_1, \ldots, \chi_d)$ then $x^\beta = \chi_1^{\beta_1} \ldots \chi_d^{\beta_d}$. This choice of basis function, instead of $\{x^\beta\}_{0 \leqslant |\beta| \leqslant m}$, leads to a well-conditioned matrix PWP^T in the (G)MLS approximation. The effect of this variation on the conditioning has been analytically investigated in [9].

A test problem is first considered to compare the results of MLPG and DMLPG methods. Then a heat conduction problem in functionally graded materials (FGM) for a finite strip with an exponential spatial variation of material parameters is investigated. In numerical results, we use the quadratic shifted scaled basis polynomial functions ($m = 2$) in (G)MLS approximation for both MLPG and DMLPG methods. Moreover, the Gaussian weight function

$$w_j(x) = \begin{cases} \frac{\exp(-(\|x-x_j\|_2/c)^2) - \exp(-(\delta/c)^2)}{1-\exp(-(\delta/c)^2)}, & 0 \leqslant \|x - x_j\|_2 \leq \delta, \\ 0, & \|x - x_j\|_2 > \delta \end{cases}$$

where $\delta = \delta_0 h$ and $c = c_0 h$ is used. The parameter δ_0 should be large enough to ensure the regularity of the moment matrix PWP^T in (G)MLS approximation. It depends on the degree of polynomials in use. Here we put $\delta_0 = 2m$. The constant c_0 controls the shape of the weight function and has influence on the stability and accuracy of (G)MLS approximation. There is no optimal value for this parameter at hand. Experiments show that $0.4 < c_0 < 1$ lead to more accurate results.

All routines were written using MATLAB© and run on a Pentium 4 PC with 2.50 GB of Memory and a twin–core 2.00 GHz CPU.

6.1 Test problem

Let $\Omega = [0, 1]^2 \subset \mathbb{R}^2$ and consider (5)–(8) with $\rho c = 2\pi^2$, $\kappa = 1$ and $f(x, t) = 0$. Boundary conditions using $x = (\chi_1, \chi_2) \in \mathbb{R}^2$ are

$$\begin{aligned} \frac{\partial u}{\partial n} &= 0, \ (\chi_1, \chi_2 = 0) \cup (\chi_1, \chi_2 = 1), \ \chi_1 \in [0, 1], \\ u &= e^{-t} \cos(\pi \chi_2), \ (\chi_1 = 0, \chi_2), \ \chi_2 \in [0, 1], \\ u &= -e^{-t} \cos(\pi \chi_2), \ (\chi_1 = 1, \chi_2), \ \chi_2 \in [0, 1]. \end{aligned}$$

The initial condition is $u(x, 0) = \cos(\pi \chi_1) \cos(\pi \chi_2)$, and $u(x, t) = e^{-t} \cos(\pi \chi_1) \cos(\pi \chi_2)$ is the exact solution. Let $t_F = 1$ and $\Delta t = 0.01$ in the Crank-Nicolson scheme. A regular node distribution with distance h in both directions is used. In Table 1 the CPU times used by MLPG1/2/4/5 and DMLPG1/2/4/5 are compared. As we can immediately see, DMLPG methods are absolutely faster than MLPG methods. There is no significant difference between MLPG2 and DMLPG2, because they are both collocation techniques and no numerical integration is required.

The maximum absolute errors are drawn in Fig. 1 and compared. MLPG2 and DMLPG2 coincide, but DMLPG1/4/5 are more accurate than MLPG1/4/5. DMLPG1 is the most accurate method among all. Justification needs a rigorous error and stability analysis which is not presented here. But, according to [8, 9] and all numerical

Table 1 Comparison of MLPG and DMLPG methods in terms of CPU times used (Sec.)

h	Type 1		Type 2		Type 4		Type 5	
	MLPG	DMLPG	MLPG	DMLPG	MLPG	DMLPG	MLPG	DMLPG
0.2	4.3	0.2	0.2	0.2	1.9	0.2	1.4	0.2
0.1	22.6	0.3	0.3	0.3	9.8	0.3	6.8	0.3
0.05	116.4	1.4	0.8	0.6	52.9	1.1	35.6	1.2
0.025	855.8	9.6	8.3	7.0	446.5	8.3	302.2	8.5

results, we can expect an error behavior like $\mathcal{O}\left(h^{m+1-k}\right)$, where k is the maximal order of derivatives of u involved in the functional, and if numerical integration and time discretization have even smaller errors.

For more details see the elliptic problems in [8] where the ratios of errors of both method types are compared for $m = 2, 3, 4$.

6.2 A problem in FGMs

Consider a finite strip with a unidirectional variation of the thermal conductivity. The exponential spatial variation is taken

$$\kappa(x) = \kappa_0 \exp(\gamma \chi_1), \tag{21}$$

with $\kappa_0 = 17\ \mathrm{Wm}^{-1}\,^{\circ}\mathrm{C}^{-1}$ and $\rho c = 10^6$. This problem has been considered in [12] using the meshless LBIE method (MLPG4) with Laplace transform in time, and in [6, 7] using MLPG4/5 with MLS approximation for both time and space domains,

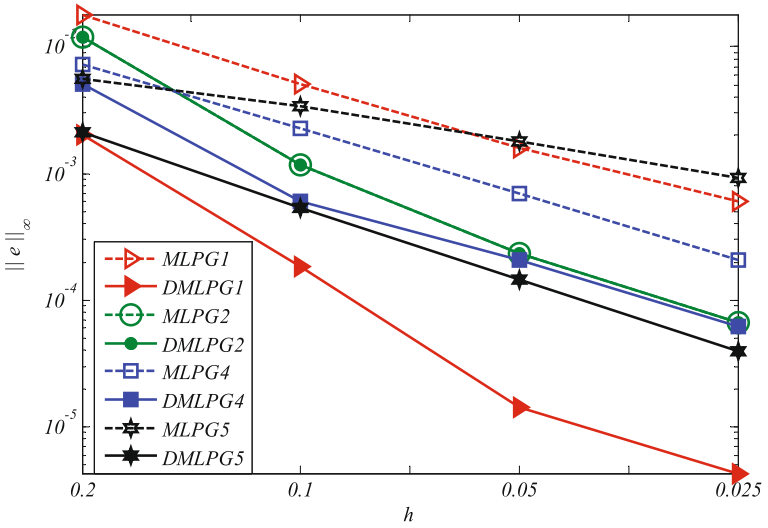

Fig. 1 Comparison of *MLPG* and *DMLPG* methods in terms of maximum errors

and in [14] using a RBF based meshless collocation method with time difference approximation.

In numerical calculations, a square with a side-length $a = 4$ cm and a 11×11 regular node distribution is used.

Boundary conditions are imposed as bellow: the left side is kept to zero temperature and the right side has the Heaviside step time variation i.e., $u = TH(t)$ with $T = 1°C$. On the top and bottom sides the heat flux vanishes.

We employed the ODE solver `ode15s` from MATLAB for the final DAE system, and we used the relative and absolute tolerances `1e-5` and `1e-6`, respectively. With these, we solved on a time interval of `[0 60]` with initial condition vector u_0 at time 0. The Jacobian matrix can be defined in advance because it is constant in our linear DAE. The integrator will detect stiffness of the system automatically and adjust its local stepsize.

In special case with an exponential parameter $\gamma = 0$ which corresponds to a homogeneous material the analytical solution

$$u(x,t) = \frac{T\chi_1}{a} + \frac{2}{\pi}\sum_{n=1}^{\infty}\frac{T\cos n\pi}{n}\sin\frac{n\pi\chi_1}{a} \times \exp\left(-\frac{\alpha_0 n^2\pi^2 t}{a^2}\right),$$

is available. It can be used to check the accuracy of the present numerical method.

Numerical results are computed at three locations along the χ_1-axis with $\chi_1/a = 0.25$, 0.5 and 0.75. Results are depicted in Fig. 2. An excellent agreement between numerical and analytical solutions is obtained.

It is known that the numerical results are rather inaccurate at very early time instants and at points close to the application of thermal shocks. Therefore in Fig. 3 we have compared the numerical and analytical solutions at very early time instants ($t \in [0, 0.4]$). Besides, in Fig. 4 the numerical and analytical solutions at points very close to the application of thermal shocks are given and compared for sample time $t(70) \approx 10.5$ sec.

The discussion above concerns heat conduction in homogeneous materials in a case where analytical solutions can be used for verification. Consider now the cases

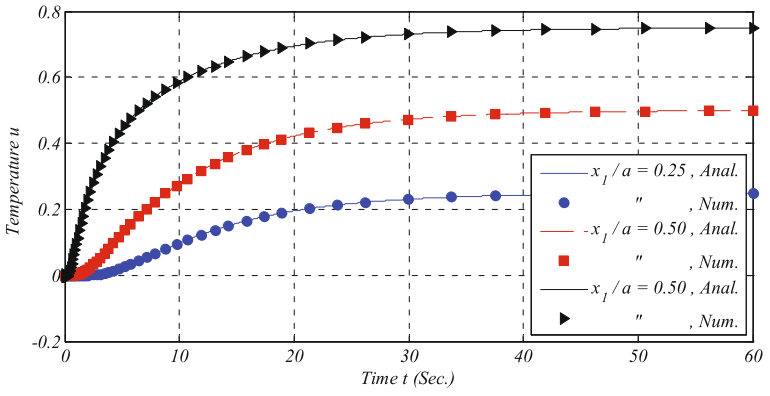

Fig. 2 Time variation of the temperature at three positions with $\gamma = 0$

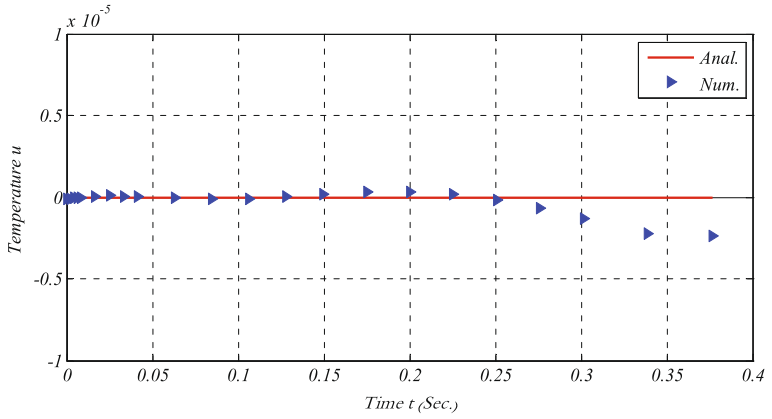

Fig. 3 Accuracy of method for early time instants at position $x_1/a = 0.5$

$\gamma = 0$, 20, 50, and 100 m^{-1}, respectively. The variation of temperature with time for the three first γ-values at position $\chi_1/a = 0.5$ are presented in Fig. 5. The results are in good agreement with Figure 11 presented in [14], Figure 6 presented in [7] and Figure 4 presented in [6].

In addition, in Fig. 6 numerical results are depicted for $\gamma = 100$ m^{-1}. For high values of γ, the steady state solution is achieved rapidly.

It is found from Figs. 5 and 6 that the temperature increases with an increase in γ-values.

For the final steady state, an analytical solution can be obtained as

$$u(x, t \to \infty) = T\frac{\exp(-\gamma\chi_1) - 1}{\exp(-\gamma a) - 1}, \quad \left(u \to T\frac{\chi_1}{a}, \text{ as } \gamma \to 0\right).$$

Analytical and numerical results computed at time $t = 60$ sec. are presented in Fig. 7. Numerical results are in good agreement with analytical solutions for the steady state temperatures.

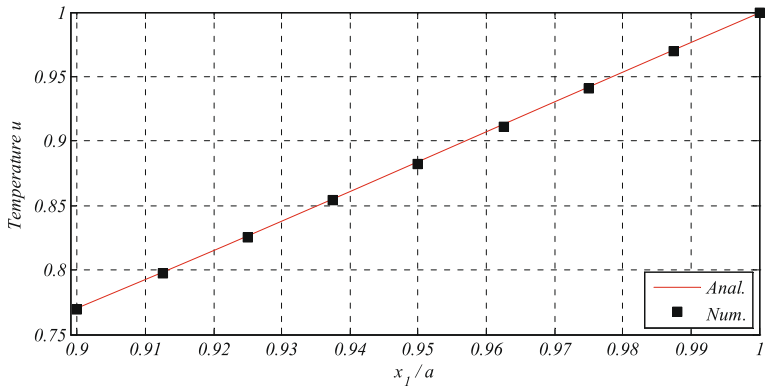

Fig. 4 Accuracy of method for points close to the thermal shock at time t (70) sec

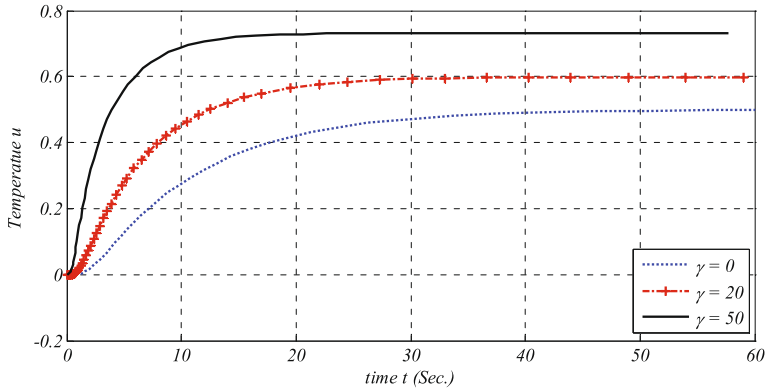

Fig. 5 Time variation of the temperature at position $x_1/a = 0.5$ for $\gamma = 0, 20, 50$ m^{-1}

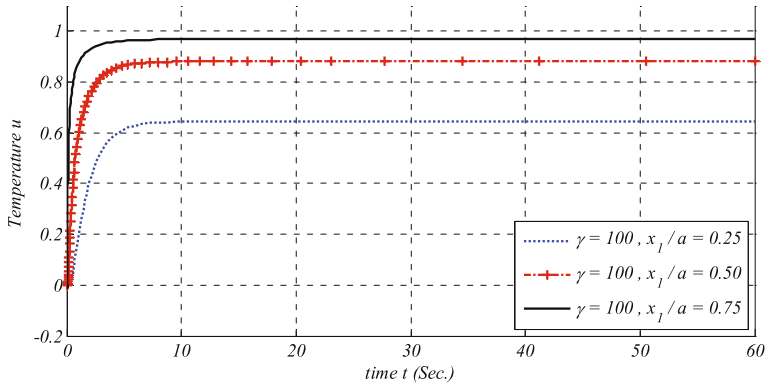

Fig. 6 Time variation of the temperature at positions $x_1/a = 0.25, 0.5, 0.75$ for $\gamma = 100$ m^{-1}

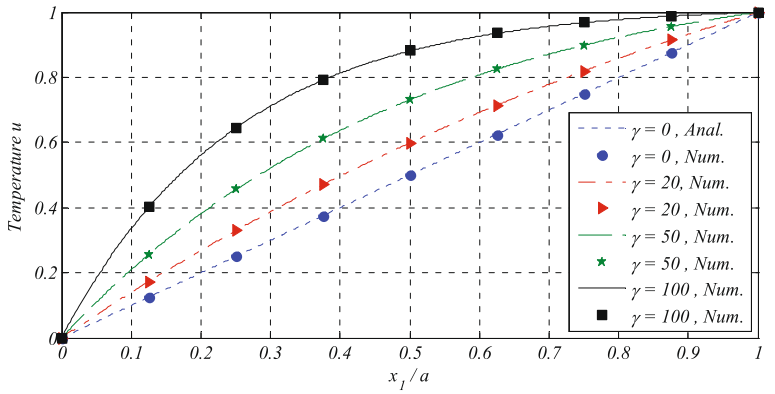

Fig. 7 Distribution of temperature along x_1-axis under steady-state loading conditions

Acknowledgments The first author was financially supported by the Center of Excellence for Mathematics, University of Isfahan.

References

1. Atluri, S.N.: The meshless method (MLPG) for domain and BIE discretizations. Tech Science Press, Encino (2005)
2. Babuska, I., Banerjee, U., Osborn, J., Zhang, Q.: Effect of numerical integration on meshless methods. Comput. Methods Appl. Mech Engrg **198**, 27–40 (2009)
3. Belytschko, T., Krongauz, Y., Organ, D., Fleming, M., Krysl, P.: Meshless methods: an overview and recent developments. Comput. Methods Appl. Mech. Eng., special issue **139**, 3–47 (1996)
4. Belytschko, T., Lu, Y., Gu, L.: Element-Free Galerkin methods. Int. J. Numer. Methods Eng. **37**, 229–256 (1994)
5. Minkowycz, W., Sparrow, E., Schneider, G., Pletcher, R.: Handbook of numerical heat transfer. Wiley, New York (1988)
6. Mirzaei, D., Dehghan, M.: MLPG method for transient heat conduction problem with MLS as trial approximation in both time and space domains. CMES–Comput. Model. Eng. Sci. **72**, 185–210 (2011)
7. Mirzaei, D., Dehghan, M.: New implementation of MLBIE method for heat conduction analysis in functionally graded materials. Eng. Anal. Bound. Elem. **36**, 511–519 (2012)
8. Mirzaei, D., Schaback, R.: Direct Meshless Local Petrov-Galerkin (DMLPG) method: a generalized MLS approximation. Appl. Numer. Math. **68**, 73–82 (2013). doi:10.1016/j.apnum.2013.01.002
9. Mirzaei, D., Schaback, R., Dehghan, M.: On generalized moving least squares and diffuse derivatives. IMA J. Numer. Anal. **32**, 983–1000 (2012)
10. Sladek, J., Sladek, V., Atluri, S.: Meshless local Petrov-Galerkin method for heat conduction problem in an anisotropic medium. CMES–Comput. Model. Eng. Sci. **6**, 309–318 (2004)
11. Sladek, J., Sladek, V., Hellmich, C., Eberhardsteiner, J.: Heat conduction analysis of 3-D axisymmetric and anisotropic FGM bodies by meshless local Petrov-Galerkin method. Comput. Mech. **39**, 223–233 (2007)
12. Sladek, J., Sladek, V., Zhang, C.: Transient heat conduction analysis in functionally graded materials by the meshless local boundary integral equation method. Comput. Mater. Sci. **28**, 494–504 (2003)
13. Sladek, V., Sladek, J., Tanaka, M., Zhang, C.: Transient heat conduction in anisotropic and functionally graded media by local integral equations. Eng. Anal. Bound. Elem. **29**, 1047–1065 (2005)
14. Wang, H., Qin, Q.H., Kang, Y.L.: A meshless model for transient heat conduction in functionally graded materials. Comput. Mech. **38**, 51–60 (2006)
15. Wendland, H.: Scattered data approximation. Cambridge University Press, Cambridge (2005)
16. Zhu, T., Zhang, J., Atluri, S.: A local boundary integral equation (LBIE) method in computational mechanics, and a meshless discretization approach. Comput. Mech. **21**, 223–235 (1998)