 CMES, vol.88, no.3, pp.183-209, 2012

Numerical Investigation on Direct MLPG for 2D and 3D Potential Problems

Annamaria Mazzia[1], **Giorgio Pini**[1] **and Flavio Sartoretto**[2]

Abstract: Pure meshless techniques are promising methods for solving Partial Differential Equations (PDE). They alleviate difficulties both in designing discretization meshes, and in refining/coarsening, a task which is demanded e.g. in adaptive strategies. Meshless Local Petrov Galerkin (MLPG) methods are pure meshless techniques that receive increasing attention. Very recently, new methods, called Direct MLPG (DMLPG), have been proposed. They rely upon approximating PDE via the Generalized Moving Least Square method. DMLPG methods alleviate some difficulties of MLPG, e.g. numerical integration of tricky, non–polynomial factors, in weak forms. DMLPG techniques require lower computational costs respect to their MLPG counterparts. In this paper we numerically analyze the solution of test 2D problems via DMLPG. We report about our expansion of meshless techniques to 3D problems. Finally, we perform comparisons between DMLPG and two MLPG techniques, when solving 3D problems.

Keywords: Meshless Methods; Poisson Problem; Generalized Moving Least Squares; Radial Basis Functions; Tensor Product Functions.

1 Introduction

Overcoming Finite Element (FE) Methods is one of the keywords for improving both accuracy and efficiency in the numerical solution of Partial Differential Equations (PDE). Non–standard numerical methods for the solution of differential equations are often based on heuristic ideas, and verified by numerical experiments (Babuska, Banerjee, and Osborn, 2005). We focus our attention on *true* Meshless methods, which do not exploit any mesh neither for discretizing the problem domain, nor for numerical integrations. As a matter of fact, some methods were

[1] Dipartimento ICEA Università di Padova Via Trieste 63, 35121 Padova, Italy {annamaria.mazzia,giorgio.pini}@unipd.it

[2] DAIS, Università Ca' Foscari Venezia Via Torino 155, 10173 Venezia, Italy flavio.sartoretto@unive.it

proposed which are labelled "meshless" because they do not exploit any mesh for discretizing the variational form of a PDE, but they enroll a mesh in order to perform numerical quadratures, see e.g. (Fries and Matthies, 2004; Belytschko, Krongauz, Organ, Fleming, and Krysl, 1996). True meshless methods are apt to reduce the high computational cost spent in order both to design discretization meshes for complex geometries, and to implement adaptive techniques.

Meshless Petrov Galerkin (MLPG) methods (see e.g. (Atluri and Zhu, 2002)) are true meshless methods which receive increasing attention. In the literature one finds many implementations relying upon Radial Basis Functions (RBF) (Buhmann, 2003), see e.g. (Atluri, 2004), and the references herein. A smaller number of papers exploiting Tensor Product Functions (TPF) is also available (Ni, Ho, Yang, and Ni, 2004; Duflot and Nguyen-Dang, 2002; Sun, Wang, and Miao, 2008; Liu, 2009; Sterk and Trobec, 2008; Mazzia and Sartoretto, 2010).

The advantages of meshless methods are peculiarly dominant when 3D problems are considered, see (Mazzia, Pini, and Sartoretto, 2008) and the References herein.

Very recently, new MLPG methods were proposed, called Direct MLPG (DMLPG) (Mirzaei and Schaback, 2011), which exploit the Generalized Moving Least Squares method (GMLS) (Mirzaei, Schaback, and Dehghan, 2011). While Moving Least Squares (MLS), proposed in (Lancaster and Salkauskas, 1981) aim at approximating a multivariate function, GMLS approximates linear functionals. The solution of a linear PDE, formulated either in strong or weak form, can be computed by approximating via GMLS the functionals involved. While MLPG approximates the PDE solution by projection on a finite dimensional space, DMLPG approximates the operators involved in the formulation of the problem. In other words, MLPG generalizes Finite Element Methods, while DMLPG generalizes Finite Difference techniques.

The performance of DMLPG methods have been started to be studied (Mirzaei and Schaback, 2011, 2012). Their great potential needs validation from an experimental point of view. In this paper we numerically analyze the efficiency and accuracy of DMLPG when solving test 2D potential problems. We extended meshless procedures to 3D problems. Such operation is not straightforward. Handling of sparse data structures is required, and appropriate techniques for solving the final linear system must be identified. When attacking test 3D problems, we compare DMLPG with two MLPG schemes. Convergence analysis of DMLPG and MLPG methods is not so mature as for FE, hence we analyze advantages and disadvantages of some scenarios by numerical experiments.

This paper is organized as follows. Section 2 describes GMLS approximation technique, which is the starting point for DMLPG. Section 3 sketches the potential

problem to be solved and the solution technique. Section 4 describes the finite dimensional spaces which identify meshless approximation techniques. Section 5 analyzes the efficiency of DMLPG on 2D problems. Section 6 compares the performance of DMLPG with two MLPG techniques, when solving 3D problems. Conclusions are drawn in Section 7.

2 Approximation schemes

The Moving Least Squares (MLS) method (Lancaster and Salkauskas, 1981) was proposed for approximating a function, $u(\underline{x})$, inside a region $\Omega \subset \mathbb{R}^d$ after a number, N, of its values, $u(\underline{x}_i)$, $\underline{x}_i \in \Omega$ is given. In the context of MLPG methods, in order to approximate a given d-dimensional problem, MLS is exploited for generating a set of d-dimensional trial functions on the ground of a suitable "weight" function (Mazzia, Pini, and Sartoretto, 2008; Mazzia and Sartoretto, 2010). The ensuing trial functions are called the "shape" functions of MLS.

The Generalized Moving Least Squares (GMLS) method (Mirzaei, Schaback, and Dehghan, 2011), is a technique developed after MLS formulation in (Levin, 1998), for approximating continuous linear functionals in the dual of $\mathscr{C}^k(\Omega)$, for any $k \geq 0$. GMLS aims to approximate a linear functional, $\lambda(u)$, based upon a given set of N linear functionals $\lambda_i(u)$. One obtains an approximation

$$\widehat{\lambda(u)} = \sum_{i=1}^{N} \phi_i(\lambda)\, \lambda_i(u). \tag{1}$$

Each coefficient $\phi_i(\lambda)$, called a GMLS shape function, must be linear in λ. As an example, $\lambda(u)$ can be a partial derivative of u, while $\lambda_i(u) = u(\underline{x}_i)$ can be a given set of u values on given nodes $\underline{x}_i \in \Omega$; $\widehat{\lambda(u)}$ in this case can be interpreted as a generalization of Finite Difference methods, based upon all $u(\underline{x}_i)$ values, in place of a given stencil.

Let $\mathbb{P}_q^d$ be the linear space of d–variate polynomials with degree up to q, whose dimension is

$$m = \binom{q+d}{d}.$$

For a given continuous linear functional $\lambda(u)$, GMLS based upon $\mathbb{P}_q^d$ computes an approximation $\widehat{\lambda(u)} = \lambda(p^*)$, where p^* minimizes in $\mathbb{P}_q^d$ the least–squares error functional

$$J = \sum_{i=1}^{N} w(\lambda_i, \lambda)(\lambda_i(u) - \lambda_i(p))^2,$$

where $w(\lambda_i,\lambda)$ are given non–negative weights.

Assume $\underline{p}(\underline{x}) = (p_1(\underline{x}),\dots,p_m(\underline{x}))^T$ is a polynomial basis of $\mathbb{P}_q^d$. As an example, when $d=2$ and $q=1$, one can set $\underline{p}(\underline{x}) = (1,x,y)^T$, when $d=2$ and $q=2$, $\underline{p}(\underline{x}) = (1,x,y,x^2,xy,y^2)^T$, etc. Any polynomial $p \in \mathbb{P}_q^d$ can be written as $p(\underline{x}) = \underline{p}(\underline{x})^T \underline{a} = \sum_j p_j(\underline{x})\, a_j$, where $\underline{a} = (a_1,\dots,a_m)^T$ is a coefficient vector. The functional J can be rewritten into

$$J = \sum_{i=1}^{N} w(\lambda_i,\lambda)(\lambda_i(u) - \lambda_i(\underline{p}(\underline{x})^T\,\underline{a}))^2,$$

Since the λ_i are linear, one has

$$\lambda_i(\underline{p}(\underline{x})^T\,\underline{a}) = \lambda_i(\sum_j p_j(\underline{x})a_j) = \sum_j \lambda_i(p_j(\underline{x}))a_j = \underline{p}_{\lambda_i}(\underline{x})^T\,\underline{a},$$

being $\underline{p}_{\lambda_i}(\underline{x}) = (\lambda_i(p_1(\underline{x})),\dots,\lambda_i(p_m(\underline{x})))^T$.

Define the $N\times m$ matrix $E = (\underline{p}_{\lambda_1}|\dots|\underline{p}_{\lambda_N})^T$, the N vector $\tilde{\underline{u}} = (\lambda_1(u),\dots,\lambda_N(u))^T$, and the diagonal matrix $W = \mathrm{diag}(w_1(\lambda),\dots,w_N(\lambda))$, being $w_i(\lambda) = w(\lambda_i,\lambda)$.

Now we can write

$$J = (E\underline{a} - \tilde{\underline{u}})^T W (E\underline{a} - \tilde{\underline{u}}).$$

Minimizing over $\underline{a}$ gives the relation

$$E^T W E \underline{a} = E^T W \tilde{\underline{u}}.$$

Hence the minimizing coefficient vector is

$$\underline{a}^* = (E^T W E)^{-1} E^T W \tilde{\underline{u}}.$$

and the polynomial minimizing J is

$$p^* = \underline{p}(\underline{x})^T \underline{a}^* = \underline{p}(\underline{x})^T (E^T W E)^{-1} E^T W \tilde{\underline{u}}.$$

By defining $\underline{\phi}(\lambda) = (\phi_1(\lambda),\dots,\phi_N(\lambda))^T$, one has

$$\lambda(\underline{p}(\underline{x})^T\,\underline{a}^*) = \underline{\phi}^T(\lambda)\tilde{\underline{u}}.$$

Due to linearity of λ,

$$\lambda(\underline{p}^*) = \lambda(\underline{p}(\underline{x})^T\,\underline{a}^*) = \lambda(\underline{p}(\underline{x})^T)\,\underline{a}^* = \lambda(\underline{p}(\underline{x})^T)\,(E^T W E)^{-1} E^T W \tilde{\underline{u}},$$

hence recalling that $W = W^T$, since W is diagonal, we define the vector of GMLS shape functions

$$\underline{\phi}(\lambda) = WE(E^T WE)^{-1}\lambda(\underline{p}(\underline{x})).$$

One has

$$\widehat{\lambda(u)} = \underline{\phi}^T(\lambda)\underline{\tilde{u}} = \sum_{i=1}^{N} \phi_i(\lambda)\lambda_i(u).$$

This result shows that $\lambda(p^*) = \widehat{\lambda(u)}$ can be recast into form (1). For more details on the relations sketched above, see e.g. (Levin, 1998; Mirzaei and Schaback, 2011).

3 Meshless techniques

Let us consider the linear Poisson equation on the domain Ω

$$-\nabla^2 u(\underline{x}) = f(\underline{x}), \tag{2}$$

where f is a given source function, $\underline{x}$ being any point in Ω. Dirichlet and Neumann boundary conditions are imposed on the domain boundary $\partial\Omega$

$$u = \bar{u} \quad \text{on } \Gamma_u, \quad \frac{\partial u}{\partial \underline{n}} \equiv q = \bar{q} \quad \text{on } \Gamma_q \tag{3}$$

where $\bar{u}$ and $\bar{q}$ are the prescribed potential and normal flux, respectively, on the Dirichlet boundary, Γ_u, and on the Neumann boundary, Γ_q, being $\partial\Omega = \Gamma = \Gamma_u \cup \Gamma_q$, $\Gamma_u \cap \Gamma_q = \emptyset$. The outward normal direction to Γ is denoted by $\underline{n}$.

Assume that the residual of eq. (2) is multiplied by a suitable test function τ. The divergence theorem is applied, hence obtaining the weak formulation for (2)

$$\int_\Omega \nabla u \cdot \nabla \tau d\Omega - \int_\Gamma (\nabla u \cdot \underline{n})\, \tau d\Gamma = \int_\Omega f\, \tau\, d\Omega, \quad \forall \tau \in \mathscr{S}, \tag{4}$$

once a suitable functional space $\mathscr{S}$ is identified. There are many Meshless Petrov–Galerkin (MLPG) methods (Atluri, 2004; Fries and Matthies, 2004), each one can be identified by an appropriate choice of trial and test functions.

In order to approximate the solution of our weak formulation, a set of discretization nodes must be given. Let N be the total number of nodes.

A set of trial functions, ξ_i, is enrolled, each one being "centered" on node $\underline{x}_i$. The support of ξ_i, S_{ξ_i} is a ball centered at $\underline{x}_i$, whose radius is r_i. Actually, for each i we set $\xi_i = 0$ outside Ω, which is equivalent to considering $S_{\xi_i} \cap \Omega$, in place of

S_{ξ_i}. In order to compute an approximation $\tilde{u} = \sum_i \tilde{u}_i \xi_i$ of the solution, a finite set of test functions τ_i, $i = 1,\ldots,N$ is elected. The support of τ_i, $S_{\tau_i} = \Omega_i$, is a ball (or a cuboid) centered at $\underline{x}_i$, whose radius (or half sidelength) is ρ_i. We assume $\tau_i = 0$ outside Ω, hence considering $S_{\tau_i} \cap \Omega$. We assume $\tau_i = 0$ on $\partial\Omega_i$, a typical setting in many MLPG schemes (Mirzaei and Schaback, 2011; Mazzia and Sartoretto, 2010). In principle the test functions can be centered on nodes which do not coincide with the $\underline{x}_i$, but for simplicity we exploit the same set of nodes, both for the trial and the test functions.

A set of Local Weak Forms (LWF) is obtained by writing eq. (4) for each test function

$$\int_{\Omega_i} \nabla u \cdot \nabla \tau_i d\Omega - \int_{\Gamma_i^{(u)}} (\nabla u \cdot \underline{n})\, \tau_i d\Gamma = \int_{\Omega_i} f\, \tau_i\, d\Omega + \int_{\Gamma_i^{(q)}} (\nabla u \cdot \underline{n})\, \tau_i d\Gamma, \tag{5}$$

where $\Gamma_i^{(u)} = \partial\Omega_i \cap \partial\Gamma_u$ is the intersection of our local integration domain boundary with *Dirichlet* boundary. Analogously, $\Gamma_i^{(q)} = \partial\Omega_i \cap \partial\Gamma_q$ is the intersection of our local integration domain boundary with *Neumann* boundary. Integrals on $\Gamma_i = \partial\Omega_i \backslash (\Gamma_i^{(u)} \cup \Gamma_i^{(q)})$, the portion of $\partial\Omega_i$ lying inside Γ, give null contribution, being $\tau_i = 0$ on $\partial\Omega_i$. Boundary conditions are treated by exploiting suitable techniques (Atluri, 2004; Mazzia, Pini, and Sartoretto, 2008).

In the sequel, following (Mazzia and Sartoretto, 2010), we consider MLPG methods where the trial functions are the MLS shape functions generated by suitable Radial Basis Functions (RBF), while the test functions are either RBF, or Tensor Product Functions (TPF).

Concerning DMLPG, it is obtained by applying GMLS to the weak problem (5). We identify the following linear functionals, to be approximated via GMLS, by using the point functionals $\lambda_i(u) = u(\underline{x}_i)$.

For convenience we divide the discretization nodes into three subsets, i.e. the nodes $\underline{x}_j$, falling inside Ω, the set of nodes $\underline{x}_k \in \Gamma_u$, the set of nodes $\underline{x}_l \in \Gamma_q$. For each internal node $\underline{x}_j$, the left-hand side of the weak form (5)

$$\mu_j(u) \equiv \int_{\Omega_j} \nabla u \cdot \nabla \tau_j d\Omega - \int_{\Gamma_j^{(u)}} (\nabla u \cdot \underline{n})\, \tau_j d\Gamma,$$

is the functional to be approximated, while the known term is

$$\beta_j \equiv \int_{\Omega_j} f\, \tau_j\, d\Omega + \int_{\Gamma_j^{(q)}} (\nabla u \cdot \underline{n})\, \tau_j d\Gamma,$$

Following GMLS steps, the ensuing approximation is

$$\widehat{\mu_j(u)} = \sum_i \phi_i(\mu_j) u(\underline{x}_i).$$

The j-th DMLPG approximated equation is

$$\widehat{\mu_j(u)} = \beta_j.$$

Now being $(\underline{p}_{\lambda_i})_k = \lambda_i(p_k) = p_k(\underline{x}_i)$, $k = 1,\ldots,m$, the matrix E in GMLS coincides with MLS one (Mirzaei, Schaback, and Dehghan, 2011). The GMLS shape functions are

$$\underline{\phi}(\mu_j) = WE(E^TWE)^{-1}\mu_j(\underline{p}) = Q\mu_j(\underline{p}),$$

being $Q = WE(E^TWE)^{-1}$. Now the weight functions are the same compact supported RBF used in MLS, hence

$$w(\lambda_i,\mu_j) = w_j(x_i) = w(\left\|\underline{x}_i - \underline{x}_j\right\|),$$

is non-zero only when $\left\|\underline{x}_i - \underline{x}_j\right\| < r_i$, for a given r_i value. So the smaller r_i values are, the smaller size $N_i < N$ the matrix Q has.

Note that $\mu_j(\underline{p})$ involves evaluating the weak form where the integrating functions are products whose factors are test functions (or their partial derivatives), and polynomials, whereas in MLPG the factors corresponding to the latter ones are more complex MLS shape functions, and their partial derivatives.

Dirichlet boundary conditions can be directly imposed as $\lambda_k(u) = u(\underline{x}_k) = \bar{u}(\underline{x}_k)$, $\underline{x}_k \in \Gamma_k$. Such approach is a simplification over using MLPG techniques, where the unknowns are the so called "fictitious" values, which do not coincide with $u(\underline{x}_i)$ (Atluri, 2004), leading to many techniques developed in order to fulfill Dirichlet BC (see e.g. (Mazzia, Pini, and Sartoretto, 2008; Wu and Plesha, 2002)). Moreover, if one needs to approximate $u(\underline{x}_i)$, a "reconstruction" step must be performed (Mazzia, Pini, and Sartoretto, 2008).

Neumann BC are imposed by approximating the normal derivative operator $\eta_l(u) = \partial u(\underline{x}_l)/\partial \underline{n}$ and setting the approximate relations $\widehat{\eta_l(u)} = \bar{q}(\underline{x}_l)$ for each $\underline{x}_l \in \Gamma_q$.

We emphasize that the MLPG methods described above require computing for each discretization node the MLS shape functions and their partial derivatives on its associated numerical quadrature points. In place of MLS shape functions, DMLPG involves polynomials. This is a key difference: MLS shape functions are computationally more demanding. Moreover, DMLPG requires evaluating one single time for each node the matrix Q, while each evaluation of a MLS shape function in MLPG needs evaluating a Q-like matrix. Such differences are the main responsible of the higher CPU time one well guesses it is spent by MLPG over DMLPG, for a given approximation tolerance.

4 Finite dimensional spaces

On the ground of our previous 2D and 3D results (Mazzia, Pini, and Sartoretto, 2008; Mazzia and Sartoretto, 2010), we exploit DMLPG trial functions based upon GMLS. In order to identify a trial space, both for MLPG and DMLPG we exploit a set of suitable *weights*, hence obtaining the so called *shape* functions (Atluri and Zhu, 2002; Belytschko, Krongauz, Organ, Fleming, and Krysl, 1996; Lancaster and Salkauskas, 1981; Mazzia, Pini, and Sartoretto, 2008).

In order to fully specify our meshless procedure, we need to identify suitable *test* functions. On the ground of our previous results on 2D problems (Mazzia and Sartoretto, 2010), as the test function space generators, we exploited TPF generated by suitable 1D functions (see the sequel).

4.1 Generating functions

Assume $w(t)$ is a given, differentiable, compact supported, generator function; when $t > 1$, $w(t) = 0$ holds.

A RBF associated to node $\underline{x}_i$ is identified by suitably setting a support radius r_i and considering

$$w\left(\frac{\|\underline{x}-\underline{x}_i\|_2}{r_i}\right).$$

Our weights and test spaces were obtained by using three types of "generating" functions.

We considered cubic and quartic spline functions like in (Belytschko, Krongauz, Organ, Fleming, and Krysl, 1996). The cubic spline is

$$w(t) = \begin{cases} \frac{2}{3} - 4t^2 + 4t^3 & 0 \le t \le 1/2, \\ \frac{4}{3} - 4t + 4t^2 - \frac{4}{3}t^3 & 1/2 \le t \le 1, \\ 0 & t \ge 1. \end{cases} \tag{6}$$

The quartic spline is

$$w(t) = \begin{cases} 1 - 6t^2 + 8t^3 - 3t^4 & 0 \le t \le 1, \\ 0 & t \ge 1. \end{cases} \tag{7}$$

Gaussian generator after (Lu, Belytschko, and Gu, 1994) is

$$w(t) = \begin{cases} \dfrac{\exp(-(\sigma t)^2) - \exp(-\sigma^2)}{1 - \exp(-\sigma^2)} & 0 \le t \le 1 \\ 0 & t \ge 1, \end{cases} \tag{8}$$

where σ is a parameter controlling the function shape.

Concerning TPF, we provide definitions for 3D problems. 2D problems are treated by disregarding the z values. To each node, $\underline{x}_i$, we associate the TPF

$$\tau_i(x,y,z) = f(|x-x_i|/\eta_i^{(x)}) \cdot f(|y-y_i|/\eta_i^{(y)}) \cdot f(|z-z_i|/\eta_i^{(z)}), \tag{9}$$

by suitably choosing the $\eta_i^{(*)}$ factors. For simplicity, we assume $\eta_i^{(x)} = \eta_i^{(y)} = \eta_i^{(z)} = \bar{\eta}_i$, hence the support of $\tau_i(\underline{x})$ is a cube centered at $\underline{x}_i$. We call $\bar{\eta}_i$ the "radius" of the cube. We exploited polynomial generators after (Liu, 2009)

$$f(t) = \begin{cases} 1-t^2, & 0 \le t \le 1, \\ 0, & t \ge 1. \end{cases} \tag{10}$$

4.2 Hardware and software

We implemented our algorithms into FORTRAN 77 codes, compiled via XLF v9.1. They were run on an IBM Power 5, 2 dual–core 1.9 GHz CPUs. The machine has a 16 GB RAM, two–level cache, 64 KB first level, 2 MB second level. Its operating system is AIX version 5.3.

5 2D problems

5.1 Uniform and irregular discretizations

Many parameters must be tuned in order to identify effective DMLPG methods. Recall that, despite DMLPG was introduced in order to deal with irregular "clouds" of points, tuning is not like to be safely performed on completely random sets of discretization points. In order to analyze the main features of our meshless procedures, we start focusing on *uniform* discretizations, U_j, on $[0,1]^2$, i.e. sets of nodes which are corners of uniform grids. Node distances are $h_j = 1/2^{j+1}$, $j = 1,\ldots,5$. The number of discretization nodes is N = 25, 81, 289, 1089, 4225, respectively.

We also performed extensive numerical experiments on irregular discretizations. They showed that completely random refinements produce high error oscillations. In order to provide "reasonably irregular" discretizations, we had to produce nested clouds, i.e. each finer cloud encompasses all nodes in the coarser one. Following (Mirzaei, Schaback, and Dehghan, 2011), we exploited Halton points (Halton, 1960). We built an irregular discretization, H_j, $j = 1,\ldots,5$, by taking the uniformly distributed boundary nodes in U_j, and generating a number of internal Halton points so that the total number of nodes equals that one of U_j. Halton points are computed

j	N	$h_{H_j,[0,1]^2}$	q_{H_j}
1	25	2.4420×10^{-1}	3.6325×10^{-2}
2	81	1.4555×10^{-1}	1.6800×10^{-2}
3	289	6.7585×10^{-2}	4.0964×10^{-3}
4	1089	3.2606×10^{-2}	1.5982×10^{-3}
5	4225	1.6876×10^{-2}	1.7474×10^{-4}

Table 1: Fill distance and separation distance of the irregular discretizations H_j.

using the code documented in (Fasshauer, 2007), ensuring that when $j < k$, $H_j \subset H_k$ holds true.

When dealing with an irregular distribution of points, H, discretizing the domain Ω, the following two measures are worth considering (Fasshauer, 2007). The fill distance is

$$h_{H,\Omega} = \sup_{\underline{x} \in \Omega} \min_{1 \le i \le N} \|\underline{x} - \underline{x}_i\|.$$

The separation distance is

$$q_H = \frac{1}{2} \min_{j \neq i} \|\underline{x}_j - \underline{x}_i\|.$$

Table 1 shows fill and separation distances of our irregular discretizations H_j. Note that the fill distance approximately halves when considering a given discretization and the immediately finer one. Moreover each H_j is quasi–uniform e.g. respect to the constant $c = 100$, i.e.

$$q_{H_j} < h_{H_j,[0,1]^2} < 100\, q_{H_j}.$$

The fill distance is also connected to shifted–scaled polynomial basis. Assume h is either the edge length, when an uniform discretization is exploited, or the fill distance, when an irregular discretization is considered. When computing GMLS on point (x_i, y_i), one can use $(x - x_i)/h, (y - y_i)/h$ in place of (x, y) for building the polynomial basis. Such a shifted–scaled basis is expected to be more numerically stable than the standard one (Mirzaei, Schaback, and Dehghan, 2011).

5.2 Test problems

In the sequel we report our numerical results concerning the solution of two test problems.

- Let us consider Poisson problem on $[0,1]^2$ with Dirichlet boundary conditions, whose solution is (Wagner and Liu, 2001)

$$u(x,y) = \left(\cosh(\pi y) - \frac{\sinh(\pi y)}{\tanh(\pi)}\right)\sin(\pi x).$$

  This problem is labeled $\mathscr{P}_D$ in the sequel.

- Now consider Poisson problem $\mathscr{P}_M$, where mixed BC are set. More precisely, Dirichlet conditions on the left and right boundaries of the domain were set; Neumann conditions on the top and bottom boundaries were set; the associated forcing function was analytically identified, so that the exact solution of Poisson problem is

$$u(x,y) = \sin x + \sin y + \sin(3x) + \sin(3y).$$

5.3 DMLPG parameter tuning

In order to evaluate the accuracy of our DMLPG method, we compute the relative error

$$\varepsilon = \sqrt{\frac{\sum_{i=1}^{N}(\tilde{u}_i - u_i)^2}{\sum_{i=1}^{N}(u_i)^2}}, \tag{11}$$

where $\tilde{u}_i$ is our approximate solution on $\underline{x}_i$, while $u_i = u(\underline{x}_i)$ is the exact solution. One can see that using e.g. the maximum norm, one is lead to the same conclusions drawn in the sequel.

Identifying effective meshless methods, requires tuning the trial support radius, r_i, on each node $\underline{x}_i$, and the test support radius, ρ_i. In order to fix ideas, for computational convenience, and according to our numerical experience, see (Mazzia, Pini, and Sartoretto, 2008; Mazzia and Sartoretto, 2010), we assume that $r_i > \rho_i$, $i = 1,\ldots,N$, hold true.

When an uniform discretization U_j is exploited, we set

$$\rho_i = \rho = \alpha h_i, \quad r_i = r = \beta h_i,$$

the α and β parameters must be tuned. When an irregular discretization H_j is given, for each node $\underline{x}_i$ we compute the distances, $\Delta x_i^{(k)}$, $k = 1,\ldots,6$ of the six closest nodes to $\underline{x}_i$. The "average" distance

$$\tilde{h}_i = \frac{1}{6}\sum_{k=1}^{6}\Delta x_i^{(k)},$$

is then computed. We tune the parameters α and β so that

$$\rho_i = \alpha \tilde{h}_i, \quad r_i = \beta \tilde{h}_i,$$

are effective support radii for the i-th test and trial functions, respectively. Assume $\underline{x}_i$ is a node internal to $\Omega = [0,1]^2$, its distance from the boundary of Ω being b_i. In order to avoid that the integration domain in (5) go outside Ω, we set $\rho_i := b_i$, when $b_i < \rho_i$.

Let us solve Poisson problem $\mathscr{P}_D$ by DMLPG. On the ground of our numerical experience with MLPG (Mazzia, Pini, and Sartoretto, 2008; Mazzia and Sartoretto, 2010), we set $\alpha = 1$. Figure 1 shows the errors raised by DMLPG vs β. Either the Gaussian, or the cubic spline, quartic spline, is exploited as the GMLS weight function. Either a quadratic basis ($q = 2$, top frame) or a cubic one ($q = 3$, bottom frame) is exploited in GMLS. The maximum number of non–zero elements per row in the final linear system matrix is also shown. By inspecting Figure 1 one can see that when using a GMLS quadratic basis ($q = 2$), and Gaussian weights, the error does not appreciably change with β, while using a cubic basis ($q = 3$) slight error variations occur, when $2 < \beta < 3.5$. On the ground of these results, in the sequel we assume $\beta = 2.5$, if not otherwise stated. By numerical experiments we found that setting $\beta = 2.5$ the error does not appreciably change when $\alpha \in [10^{-3}, 1.5]$. This result suggests a "collocation–like" behavior of DMLPG.

In order to perform surface integrals in (5), we exploited 2D Gauss–Legendre formulas, enrolling $m \times m$ quadrature nodes, $2 \leq m \leq 64$. By numerical experiments we found that the solution accuracy does not appreciably depend on the accuracy of quadrature formulas, unlike in MLPG methods (Mazzia and Pini, 2010). The results shown in the sequel were obtained by exploiting computationally cheap 3×3 Gauss–Legendre quadrature.

5.4 Numerical results

Inspecting Figure 2 one can see that setting $\beta = 2.5$, $\alpha = 1$, $\sigma = 4$, provides effective convergence with both quadratic and cubic polynomial basis.

Assume h is the fill distance of a discretization. Recall that in both our uniform discretizations, and Halton–based ones (see Table 1) a refining step halves the fill distance. Hence the convergence rate can be numerically approximated by

$$\gamma = \log_2(\frac{\varepsilon_j}{\varepsilon_{j+1}}).$$

Table 2 reports errors and convergence rates obtained when solving problem $\mathscr{P}_D$ by exploiting uniform node discretizations. Quadratic GMLS basis is enrolled

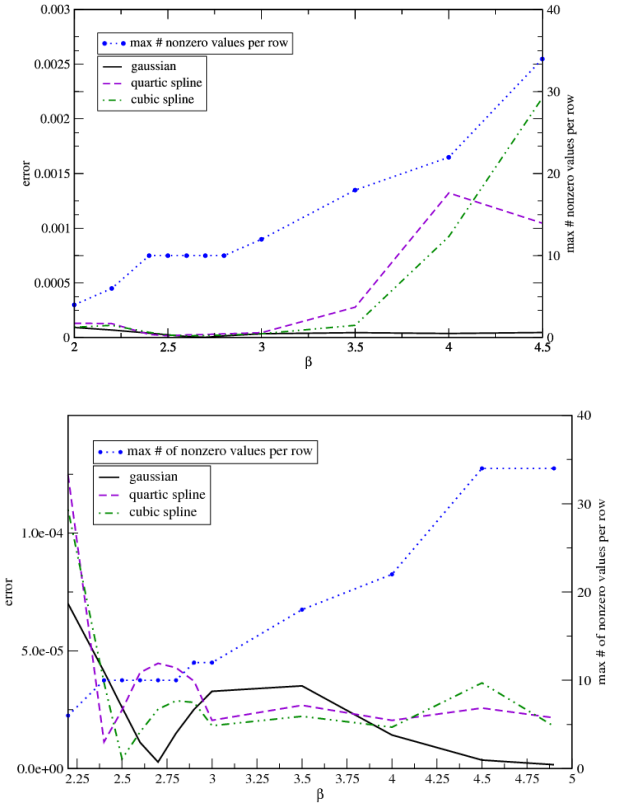

Figure 1: Errors raised solving problem $\mathscr{P}_D$, when β parameter changes; $\alpha = 1$ was set. The finest uniform grid, U_5, was exploited. Either Gaussian or spline weights are enrolled. A quadratic GMLS basis ($q = 2$) is exploited in the top frame; A cubic one ($q = 3$) is exploited in the bottom frame. The maximum number of non–zero values per row in the final coefficient matrix is also shown.

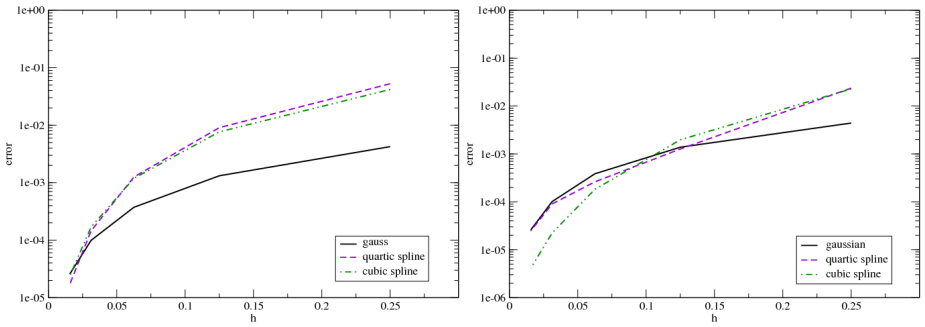

Figure 2: Problem $\mathscr{P}_D$, errors raised when exploiting uniform discretizations. We set $\beta = 2.5$, $q = 2$ (left frame), $q = 3$ (right frame).

N	gaussian		quartic spline		cubic spline	
	ε	rate	ε	rate	ε	rate
25	4.2380e-03		5.2641e-02		4.1836e-02	
81	1.3182e-03	1.7	9.0659e-03	2.5	7.6598e-03	2.4
289	3.7276e-04	1.8	1.2521e-03	2.9	1.1866e-03	2.7
1089	9.9492e-05	1.9	1.4561e-04	3.1	1.7001e-04	2.8
4225	2.5725e-05	2.0	1.7095e-05	3.1	2.3886e-05	2.8

Table 2: Errors and convergence rates recorded when solving problem $\mathscr{P}_D$ by exploiting uniform distributions of nodes, quadratic GMLS polynomial basis function. Either Gaussian, or cubic and quartic spline weights were enrolled for generating trial functions. The value $\beta = 2.5$ was set.

N	gaussian		quartic spline		cubic spline	
	ε	rate	ε	rate	ε	rate
25	4.4121e-03		2.3604e-02		2.2797e-02	
81	1.3834e-03	1.7	1.2673e-03	4.2	1.9729e-03	3.5
289	3.8457e-04	1.8	2.6149e-04	2.3	1.8224e-04	3.4
1089	1.0124e-04	1.9	9.0376e-05	1.5	2.2378e-05	3.0
4225	2.5962e-05	2.0	2.4830e-05	1.9	4.1240e-06	2.4

Table 3: Analogous to Table 2, when a cubic polynomial GMLS basis is exploited.

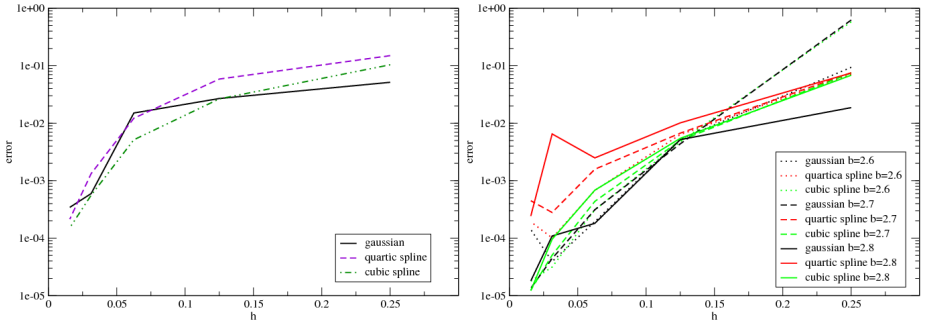

Figure 3: Solving problem $\mathscr{P}_D$ using Halton nodes. $q = 2$, $\beta = 2.5$ (left frame); $q = 3$, $\beta = 2.6, 2.7, 2.8$ (right frame).

($q = 2$). The weights for generating the trial functions are either Gaussian, or quartic splines, or cubic splines. For each given discretization, errors are quite comparable, and they decrease when the discretization is refined. Being $q = 2$, $k = 1$, the optimal convergence rate should be (Mirzaei, Schaback, and Dehghan, 2011) $\gamma \simeq 2 = q + 1 - k$. When Gaussian weights are exploited, the estimated convergence rates approach the optimal value. When spline weights are exploited, estimated over–optimal values $\gamma > 3$ are shown. On the other hand, let us inspect Table 3, which is analogous to the previous one, reporting results when cubic polynomial basis is enrolled ($q = 3$). Now optimal convergence rate should be $\gamma \simeq 3 = q + 1 - k$, being now $q = 3$, $k = 1$. Gaussian and quartic spline weights provide suboptimal rates, while cubic spline weights seems to give over–optimal convergence.

Figure 3 reports errors when Halton nodes are exploited for solving problem $\mathscr{P}_D$. One can see that in order to avoid that the error increases for small h values, slightly larger β values must be set, when $q = 3$. The value $\beta = 2.8$ works for all our three weights.

Let us now consider problem $\mathscr{P}_M$ where mixed BC are set. Figure 4 reports the errors computed when exploiting the uniform discretizations U_j, $j = 1, \ldots, 5$. Using the quadratic basis, the error behavior resembles that one reported on problem $\mathscr{P}_D$. When a cubic polynomial basis is enrolled, we had to slightly enlarge β, setting $\beta = 3.5$ produced an acceptable error behavior. Figure 5 records the errors computed when exploiting Halton nodes. Respect to exploiting an uniform discretization, the error behavior depends slightly more from the weights. Again when $q = 3$ β value must be slightly enlarged, $\beta = 2.8$ being a suitable setting.

Table 4 shows errors and convergence rates recorded when solving problem $\mathscr{P}_M$, using quadratic polynomial basis, and an uniform node distribution. One can see

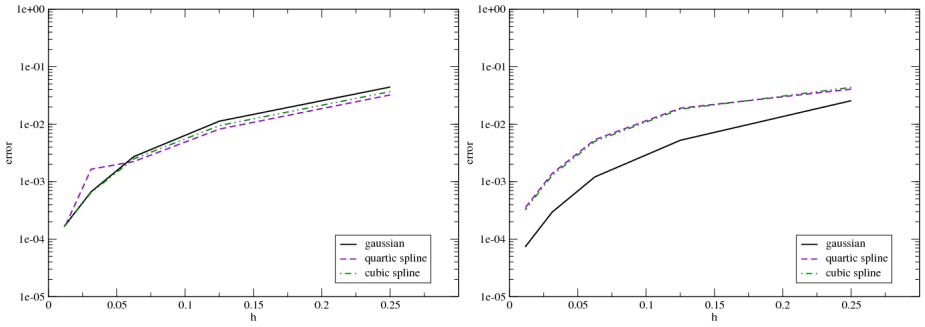

Figure 4: Errors raised when solving problem $\mathscr{P}_M$, using uniform discretizations. Left frame: $\beta = 2.5$, $q = 2$. Right frame: $\beta = 3.5$, $q = 3$.

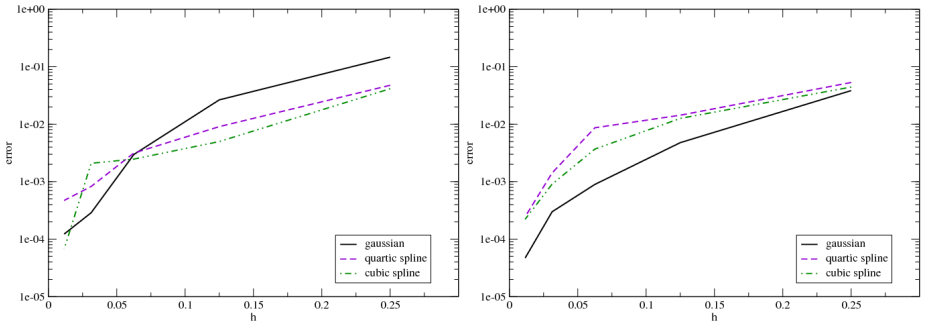

Figure 5: Errors raised when solving problem $\mathscr{P}_M$, using Halton nodes. Left frame: $\beta = 2.5$, $q = 2$. Right frame: $\beta = 2.8$, $q = 3$.

N	gaussian		quartic spline		cubic spline	
	ε	rate	ε	rate	ε	rate
25	4.4280e-02		3.2234e-02		3.6830e-02	
81	1.1270e-02	2.0	8.2056e-03	2.0	9.4441e-03	2.0
289	2.7300e-03	2.0	2.2435e-03	1.9	2.4957e-03	1.9
1089	6.6857e-04	2.0	1.6499e-03	0.4	6.5004e-04	1.9
4225	1.6535e-04	2.0	1.5529e-04	3.4	1.6636e-04	2.0

Table 4: Problem $\mathscr{P}_M$. Errors and convergence rates. Uniform distribution of nodes, quadratic polynomial basis functions, $\beta = 2.5$.

N	gaussian		quartic spline		cubic spline	
	ε	rate	ε	rate	ε	rate
25	2.5620e-02		4.0671e-02		4.3711e-02	
81	5.2458e-03	2.3	1.8984e-02	1.1	1.8208e-02	1.3
289	1.2110e-03	2.1	5.3767e-03	1.8	5.0197e-03	1.9
1089	2.9526e-04	2.0	1.3901e-03	2.0	1.2845e-03	2.0
4225	7.3291e-05	2.0	3.5002e-04	2.0	3.2331e-04	2.0

Table 5: Problem $\mathscr{P}_M$. Errors and convergence rates. Uniform distribution of nodes, cubic polynomial basis functions, $\beta = 3.5$.

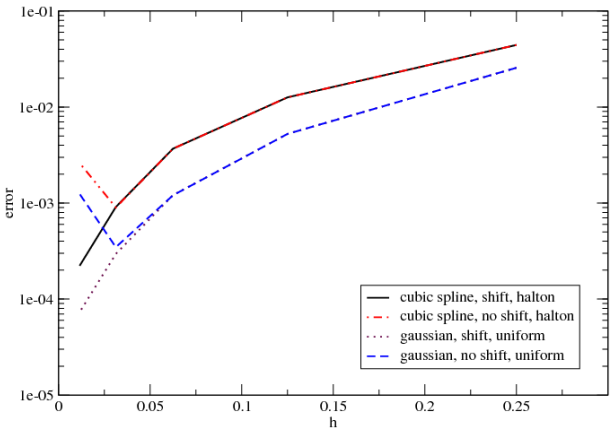

Figure 6: Errors raised solving problem $\mathscr{P}_M$, using either shifted (dashed lines) or non shifted (solid lines) polynomial basis functions.

that rates with gaussian and cubic spline weights are acceptable. When quartic splines are enrolled, oscillating rates are recorded.

Table 5 refers to cubic polynomial basis. Note that in order to achieve convergence, $\beta = 3.5$ was set.

Figure 6 show convergence profiles when $\mathscr{P}_M$ problem is solved by using either shifted or non–shifted basis. Two settings are considered: uniform discretization, $q = 2$, $\beta = 2.5$, cubic spline weight, and Halton discretization $q = 3$, $\beta = 2.8$, gaussian weights. One can see that when the finest discretization is enrolled, a shifted basis must be exploited in order to avoid a large increment in the error. These error behaviors suggest that shifted bases are likely to be enrolled when fine discretizations are exploited.

6 3D problems

We extended DMLPG technique to 3D problems, a non trivial task which needs identifying suitable data structures in order to deal with larger sets of nodes. Moreover, solving the ensuing large linear systems demands for efficient iterative solvers (see the sequel).

Now we aim to compare DMLPG with MLPG techniques. Quadratic ($q = 2$) polynomial bases are enrolled for MLS and GMLS. On the ground of our previous results on 2D and 3D problems (Mazzia, Pini, and Sartoretto, 2008; Mazzia and Sartoretto, 2010), we consider two effective MLPG techniques. One technique, labelled "RR" in the sequel, uses Gaussian RBF weights to provide MLS shape functions as the trial functions. The test functions are again Gaussian RBF weights, hence local integration domains are spheres. Our alternative MLPG technique exploits the same MLS shape functions as the trial functions, but the TPF arising from generators of type (10) as the test functions. This latter technique will be called "RT", a shorthand for RBF-TPF, as it uses trial RBF, and test TPF.

We numerically compare RR and RT when solving 3D problems, with our DMLPG technique, for shortness also labelled "D" in the sequel. Gaussian generators are exploited in order to provide GMLS weights. The test functions are the same TPF as in RT. Cuboidal local integration domains are thus considered.

Numerical cubatures on spheres were performed using Stroud's 512 node, degree 15 cubature rule (subroutine `SPH15`, after (Stroud, 1971, pag. 352)). In our previous works, we found that such rule provide accurate integral values at affordable computational costs (Mazzia, Pini, and Sartoretto, 2008). Numerical integrations on cuboids were accomplished using Gauss-Legendre product formula with 5^3 nodes.

In order to analyze the main features of our Meshless procedures, we focus on *uniform* discretizations on $[0,1]^3$, i.e. sets of nodes which are corners of uniform grids. Their edge lengths are $h_j = 1/2^{j+1}$, $j = 1,2,3,4$. The number of discretization nodes is 125, 729, 4913, 35937.

Like done above dealing with 2D problems, for any node inside each one of our uniform discretizations, we must identify suitable α and β values. Now $\alpha = 1$ is set on each discretization, while β values are reported below.

6.1 Test solutions

In the sequel we report our numerical results concerning the solution of problem (2) on the $[0,1]^3$ domain. Both the forcing function, f, and the BC, were computed after each solution in a given set of test ones. The latter solutions are drawn after our previous work on 3D potential problems (Mazzia, Pini, and Sartoretto, 2008). One polynomial sample was added, in order to enlarge our set.

After (Zhang, Tanaka, and Endo, 2004), we consider the harmonic polynomial

$$u(x,y,z) = x^3 + y^3 + z^3 - 3yx^2 - 3xz^2 - 3zy^2. \tag{12}$$

In the sequel, this test solution is labeled "uP3".
Moreover, we added the slightly different, incomplete polynomial (labeled "uP3i")

$$u(x,y,z) = x^3 + y^3 + z^3 + xyz. \tag{13}$$

After (Wang, Zhong, and Zhang, 2006), two trigonometric functions are considered.

- The function "uT1" which evenly changes along the three coordinate directions

$$u(x,y,z) = \sin x + \sin y + \sin z + \sin(3x) + \sin(3y) + \sin(3z). \tag{14}$$

- The function "uT2" which changes more rapidly in the z- and y-directions.

$$u(x,y,z) = \sin x + \sin y + \sin z + \sin(5y) + \sin(10z). \tag{15}$$

Moreover, we consider the *composed cosine–polynomial* solution "uCP" after (Mazzia, Pini, Putti, and Sartoretto, 2003)

$$u = \cos\left(3\pi\left(\frac{x^3+y^3+z^3}{3} - \frac{x^2+y^2+z^2}{2}\right)\right). \tag{16}$$

6.2 Solving linear systems

When considering 2D problems, MLPG linear systems are usually solved via direct methods, like (Li, Demmel, Gilbert, and Grigori, 2012; MUMPS, 2012; PARDISO, 2012). These methods are efficient and accurate for 2D problems, but too storage demanding when 3D problems are attacked by using fine discretizations. Preconditioned iterative methods are the best choice for large 3D problems. We efficiently solved our non symmetric, sparse, 3D linear systems via preconditioned Bi-CGSTAB (van der Vorst, 1992). Preconditioning was performed via incomplete Crout factorization (Kershaw, 1978; Pini and Zilli, 1989). Parallel implementations of this solver for non symmetric systems were exploited inside meshless techniques (Bergamaschi, Martinez, and Pini, 2009; Ferronato, Janna, and Pini, 2012). The iterations were stopped when the i-th relative residual is smaller than 10^{-15}. Such tolerance ensures achieved double precision (64-bit) numerical accuracy attained. Each linear system was solved to maximum accuracy, in order to avoid extra numerical errors in the solution procedure.

		$e^{(max)}$			convergence rates		
Test	nx	RR	RT	D	RR	RT	D
uP3	4	7.37E-04	1.13E-03	1.21E-13	-	-	-
	8	1.04E-04	1.78E-04	2.25E-12	2.8	2.7	-
	16	1.39E-05	2.26E-05	1.67E-11	2.9	3.0	-
	32	2.26E-06	3.78E-06	5.29E-10	2.6	2.6	-
uP3i	4	8.73E-04	1.05E-03	9.30E-14	-	-	-
	8	1.18E-04	1.63E-04	1.69E-12	2.9	2.7	-
	16	1.50E-05	2.15E-05	2.01E-11	3.0	2.9	-
	32	2.10E-06	3.40E-06	6.24E-10	2.8	2.7	-
uT1	4	3.98E-03	5.07E-03	1.58E-03	-	-	-
	8	5.36E-04	7.34E-04	4.18E-04	2.9	2.8	1.9
	16	6.87E-05	3.50E-05	1.06E-04	3.0	4.4	2.0
	32	1.69E-05	1.20E-05	2.66E-05	2.0	1.5	2.0
uT2	4	1.49E-01	1.81E-01	3.24E-03	-	-	-
	8	1.51E-02	1.70E-02	3.89E-03	3.3	3.4	-
	16	1.67E-03	2.00E-03	1.17E-03	3.2	3.1	1.7
	32	3.09E-04	2.69E-04	3.05E-04	2.4	2.9	1.9
uCP	4	2.14E-03	3.85E-03	2.68E-03	-	-	-
	8	3.80E-04	5.14E-04	5.08E-04	2.5	2.9	2.4
	16	9.70E-05	6.88E-05	1.26E-04	2.0	2.9	2.0
	32	2.95E-05	2.11E-05	3.16E-05	1.7	1.7	2.0

Table 6: Dirichlet BC. Errors and convergence rates recorded when approximating the proposed test solutions. Dashes identify non computable and meaningless (negative) rates.

6.3 Numerical comparison

In the sequel, in order to focus on local errors on an N–node discretization, we exploit the maximum norm error

$$e^{(max)} = \max_{i=1}^{N} |u_i - \tilde{u}_i|. \tag{17}$$

We observed that in our 3D problems the values of this norm are larger than the relative error (11).

Table 6 reports errors and convergence rates for RR, RT, and D methods, when pure Dirichlet BC are set. Values $\beta = 4$, $\sigma = 4$ were set for RR and RT, while $\beta = 1.9$, $\sigma = 3$ for D technique. The second column reports the number of nodes on the

		CPU seconds			CPU ratios		Iterations		
Test	nx	RR	RT	D	RR/D	RT/D	RR	RT	D
uP3	4	9.9	2.8	3.4	2.9	0.8	3	3	4
	8	129.3	33.5	3.6	36.2	9.4	5	5	6
	16	1291.2	329.6	11.0	118.1	30.2	8	7	10
	32	11583.5	2881.4	461.7	25.1	6.2	13	12	18
uP3i	4	9.7	2.9	3.2	3.0	0.9	3	3	2
	8	134.6	33.6	3.6	37.7	9.4	5	5	5
	16	1310.7	323.6	11.1	118.4	29.2	8	7	9
	32	11657.4	2786.5	462.3	25.2	6.0	14	12	19
uT1	4	9.8	2.9	2.3	4.3	1.3	3	3	2
	8	129.4	33.9	3.3	39.8	10.4	5	5	5
	16	1292.1	320.5	8.1	160.3	39.8	8	7	10
	32	11755.1	2859.2	469.6	25.0	6.1	14	12	19
uT2	4	9.9	2.9	3.4	2.9	0.9	3	3	2
	8	129.1	34.5	3.6	36.4	9.7	5	5	6
	16	1289.4	323.2	10.9	118.7	29.8	8	7	10
	32	11333.5	2898.7	460.6	24.6	6.3	14	12	18
uCP	4	9.8	2.9	2.5	3.9	1.2	3	3	2
	8	129.6	33.8	2.8	46.0	12.0	5	5	5
	16	1302.6	328.5	10.9	119.7	30.2	8	7	9
	32	11578.7	2890.7	412.0	28.1	7.0	14	12	16

Table 7: Dirichlet BC. CPU seconds, their ratios, and iterations spent for solving the final linear systems.

x-axis, which equals those on the y- and z- ones. Note that when approximating cubic polynomial solutions (i.e. uP3, uP3i) DMLPG displays much smaller errors (in the range 10^{-14} to 10^{-10}) than the other methods, errors which slightly increase when the discretization is refined. This counterintuitive behavior is to be ascribed to high precision in the approximate solution, which slightly decreases when a larger number of nodes, and hence of floating point operations, is introduced. This behavior produces meaningless convergence ratios, due to floating point errors. Note that a quadratic polynomial basis is exploited, hence DMLPG is not guaranteed to exactly recover third order polynomials. By inspecting Table 6, one can see that the errors raised by DMLPG when approximating our non–polynomial solutions are well comparable with those recorded by RR and RT techniques. On the other hand, DMLPG convergence rates oscillates less respect to RR and RT, when the number of nodes changes. While RR and RT rates are far from constant, DMLPG ones quite well approach the theoretically optimal $\gamma = 2$ value.

Table 7 reports CPU seconds spent for solving our test problems, together with the number of iterations spent to solve the final linear system. Note that inspecting column 6, which reports the ratio between CPU seconds spent by RR divided by D running seconds, DMLPG time can be up to approximately 160 times smaller. Column 7 reports RT/D CPU ratio, which is at most 39.8, since RT is faster than RR. On the other hand, let us see the number of iterations spent. Both RR, RT and D spent quite the same number of iterations in each test case.

Note that when the number of nodes on each axis is doubled, RR and RT CPU times are roughly multiplied by 9. On the other hand, DMLPG time increases slower, except an abrupt increase when going from 16 to 32 nodes per axis. Such increase is likely to be ascribed to higher DMLPG storage requirements, which triggers storage access problems when the number of discretization nodes increases much.

Concerning 3D problems with Mixed BC, assume we set Neumann boundary conditions on the portion $0.5 \leq x,y \leq 0.75$ of the $z = 0$ face on the $[0,1]^3$ domain. Dirichlet conditions are set elsewhere. For each given test solution, we compute the corresponding BC for Poisson problem on the unit cube.

Table 8 shows errors and convergence ratios for tests uT2 and uCP. The value $\alpha = 1$, as above, was set in both tests. Parameter tuning for RR, RT, and D techniques resulted to be a bit harder for Mixed BC than for pure Dirichlet BC. For RR and RT, the scaling parameter in Gauss generator was $\sigma = 3$, while $\beta = 4$ was set.

When $q = 2$ CPU times are quite the same as in Table 6, hence they are not reported in the sequel. It is not worth running RR and RT with $q = 3$, since CPU times are too large. On the contrary, switching from $q = 2$ to $q = 3$, DMLPG CPU times do not increase appreciably. DMLPG results are reported for both quadratic ($q = 2$)

		$e^{(max)}$				convergence rates			
		$q=2$			$q=3$	$q=2$			$q=3$
Test	nx	RR	RT	D	D	RR	RT	D	D
uT2	4	1.47E-01	2.17E-01	8.76E-01	9.13E-01	-	-	-	-
	8	1.81E-02	7.83E-02	4.69E-01	5.80E-01	3.0	1.5	0.9	0.7
	16	1.52E-02	1.42E-02	1.11E-01	4.33E-02	0.3	2.5	2.1	3.7
	32	1.37E-02	2.90E-03	2.94E-02	2.06E-03	0.1	2.3	1.9	4.4
uCP	4	2.12E-03	7.23E-03	1.86E-01	9.30E-02	-	-	-	-
	8	3.08E-04	1.06E-03	2.40E-02	2.60E-01	2.8	2.8	3.0	-
	16	9.75E-05	2.14E-04	3.78E-03	1.36E-03	1.7	2.3	2.7	7.6
	32	2.95E-05	4.37E-05	6.80E-04	8.10E-05	1.7	2.3	2.5	4.1

Table 8: Mixed BC. Analogous to Table 6.

and cubic ($q=3$) bases. The value $\sigma=5$ was set, while $\beta=2.75$ was set when $q=2$. Recall that a higher degree basis requires a bit larger β, i.e. larger supports for trial functions, in order to attain a non–singular Q matrix in GMLS. When $q=3$ we set $\beta=3.5$.

By inspecting Table 6 one can see that DMLPG with $q=3$ is preferable over $q=2$: good accuracy and higher convergence ratios, even better than the assumed optimal $\gamma=3$ value, are attained. Recalling that DMLPG CPU times do not appreciably increase when $q=3$, we infer that this technique is expected to be more effective than RR and RT.

7 Conclusions

The accuracy and efficiency of a DMLPG technique when solving 2D problems were numerically analyzed. Afterwards, 3D problems were considered. DMLPG performance was compared with those ones of two MLPG procedures.

The following points are worth emphasizing.

- Concerning the solution of 2D problems by DMLPG.
  - Parameter tuning must be carefully assessed. Setting the radiuses of trial functions is not a difficult task to perform. The number of non-zero elements per row in the final linear system matrix rapidly increases with the radius, hence in order to keep the computational cost low, small radiuses are wellcome. Using a cubic basis can ask to slightly enlarge the radiuses respect to exploiting a quadratic basis.

  - The accuracy of numerical quadrature formulas required to compute the weak forms, is not so critical as in MLPG methods. DMLPG involves "simplified" integrating functions (they do not involve shape functions as MLPG methods do) hence simpler and faster quadrature rules can be exploited.
  - Solving either pure Dirichlet or Mixed Poisson problems provides effective DMLPG approximations. Their convergence rates can depend upon the weights enrolled in order to provide the trial functions.
  - Shifted polynomial bases are likely to be enrolled when fine discretizations are exploited, in order to avoid large errors.

- Comparing DMLPG with RR and RT procedures, in solving 3D problems.

  - The CPU time spent by DMLPG can be up to one hundred time lower than RR running time, when achieving comparable or even higher DMLPG accuracy respect to RR and RT.
  - Due to storage requirements, DMLPG running time can increase more rapidly when the number of discretization nodes becomes large.
  - Respect to exploiting a quadratic basis, when considering a mixed problem, a cubic basis improves the accuracy and convergence rates of DMLPG, without appreciably increasing the CPU time spent. On the other hand RR and RT with a cubic basis are too time consuming to be worth exploiting.

Future work is planned in order to solve unsteady diffusion problems.

Acknowledgement: This work was partially funded by Fondi ex 60%, "Metodi ed Algoritmi Numerici per la Simulazione di Mezzi Porosi". The third Author received support from Università Ca' Foscari Venezia.

References

Atluri, S. N. (2004): *The Meshless method (MLPG) for domain & BIE discretizations*. Tech Science, 2004, Forsyth GA.

Atluri, S. N.; Zhu, T. (2002): The meshless local Petrov-Galerkin (MLPG) method: A simple & less–costly alternative to the finite element methods. *Computer Modeling in Engineering and Sciences*, vol. 3, no. 1, pp. 11–51.

Babuska, I.; Banerjee, U.; Osborn, J. E. (2005): Survey of meshless and generalized finite element methods: A unified approach. Technical report, Office of

Naval Research, One Liberty Center, 875 North Randolph Street Suite 1425, Arlington, VA, 22203-1995, 2005.

Belytschko, T.; Krongauz, Y.; Organ, D.; Fleming, M.; Krysl, P. (1996): Meshless methods: an overview and recent developments. *Comp. Methods App. Mech. Eng.*, vol. 139, pp. 3–47.

Bergamaschi, L.; Martinez, A.; Pini, G. (2009): An efficient parallel mlpg method for poroelastic models. *Computer Modeling in Engineering and Science*, vol. 49, no. 3, pp. 191–216.

Buhmann, M. D. (2003): *Radial Basis Functions: Theory and Implementation*, volume 12 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge.

Duflot, M.; Nguyen-Dang, H. (2002): A truly meshless method based on a moving least squares quadrature. *Commun. Numer. Methods Eng.*, vol. 18, pp. 441–449.

Fasshauer, G. E. (2007): *Meshfree approximation methods with MATLAB*, volume 6 of *Interdisciplinary Mathematical Sciences*. World Scientific Publishing Co., pub-WORLD-SCI:adr. With 1 CD-ROM (Windows, Macintosh and UNIX).

Ferronato, M.; Janna, C.; Pini, G. (2012): Shifted FSAI preconditioner for the efficient parallel solution of non-linear groundwater flow models. *International Journal for Numerical Methods in Engineering*, vol. 89, pp. 1707–1719.

Fries, T.-P.; Matthies, H.-G. (2004): Classification and overview of meshfree methods. Technical Report 2003-3, Technical University Braunschweig, Brunswick, Germany, 2004.

Halton, J. H. (1960): On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numer. Math.*, vol. 2, pp. 84–90.

Kershaw, D. S. (1978): The incomplete Cholesky-conjugate gradient method for the iterative solution of systems of linear equations. *J. Comp. Phys.*, vol. 26, pp. 43–65.

Lancaster, P.; Salkauskas, K. (1981): Surfaces generated by moving least squares methods. *Math. Comp.*, vol. 37, no. 155, pp. 141–158.

Levin, D. (1998): The approximation power of moving least–squares. *Math. Comp.*, vol. 67, no. 234, pp. 1517–1531.

Li, S.; Demmel, J.; Gilbert, J.; Grigori, L. (2012): SuperLU. `http://crd-legacy.lbl.gov/~xiaoye/SuperLU/`, 2012. Last accessed: September 7, 2012.

Liu, G. R. (2009): *Meshfree Methods: Moving Beyond the Finite Element Method*. CRC Press, second edition.

Lu, Y. Y.; Belytschko, T.; Gu, L. (1994): A new implementation of the element free Galerkin method. *Comp. Methods App. Mech. Eng.*, vol. 113, pp. 397–414.

Mazzia, A.; Pini, G. (2010): Product Gauss quadrature rules vs cubature rules in the meshless local Petrov-Galerkin method. *Journal of Complexity*, vol. 26, pp. 82–101.

Mazzia, A.; Pini, G.; Putti, M.; Sartoretto, F. (2003): Comparison of 3D flow fields arising in mixed and standard unstructured finite elements. In et al., P. S.(Ed): *Computational Science – ICCS 2003*, volume 2657 of *Lecture Notes in Computer Sciences*, pp. 560–567, Berlin. Springer-Verlag.

Mazzia, A.; Pini, G.; Sartoretto, F. (2008): Accurate MLPG solution for 3D potential problems. *Computer Modeling in Engineering & Sciences*, vol. 36, no. 1, pp. 43–63.

Mazzia, A.; Sartoretto, F. (2010): Meshless solution of potential problems by combining radial basis functions and tensor product ones. *Computer Modeling in Engineering & Sciences*, vol. 68, no. 1, pp. 95–112.

Mirzaei, D.; Schaback, R. (2011): Direct meshless local Petrov–Galerkin (DMLPG) method: a generalized MLS approximation. Preprint, 2011.

Mirzaei, D.; Schaback, R. (2012): Solving heat conduction problems by the direct meshless local Petrov–Galerkin (DMLPG) method. preprint, 2012.

Mirzaei, D.; Schaback, R.; Dehghan, M. (2011): On generalized moving least squares and diffuse derivatives. *IMA Journal of Numerical Analysis*, vol. 32, no. 3, pp. 983–1000.

MUMPS: a MUltifrontal Massively Parallel sparse direct Solver. `http://graal.ens-lyon.fr/MUMPS`, 2012. Last accessed: September 7, 2012.

Ni, G.; Ho, S. L.; Yang, S.; Ni, P. (2004): Meshless local Petrov-Galerkin method and its application to electromagnetic field computations. *International Journal of Applied Electromagnetics and Mechanics*, vol. 19, no. 1, pp. 111–117.

PARDISO solver project. `http://www.pardiso-project.org/`, 2012. Last accessed: September 7, 2012.

Pini, G.; Zilli, G. (1989): Preconditioned iterative algorithms for large sparse unsymmetric problems. *Numerical Methods for Partial Differential Equations*, vol. 5, pp. 107–120.

Sterk, M.; Trobec, R. (2008): Meshless solution of a diffusion equation with parameter optimization and error analysis. *Engineering Analysis with Boundary Elements*, vol. 32, pp. 567–577.

Stroud, A. H. (1971): *Approximate Calculation of Multiple Integrals*. Prentice-Hall, Englewood Cliffs, NJ.

Sun, H.; Wang, Y.; Miao, Y. (2008): Meshless numerical method based on tensor product. *Front. Archit. Civ. Eng. China*, vol. 2, no. 2, pp. 166–171.

van der Vorst, H. A. (1992): Bi-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, vol. 13, pp. 631–644.

Wagner, G. J.; Liu, W. K. (2001): Hierarchical enrichment for bridging scales and mesh-free boundary conditions. *Int. J. Numer. Methods Eng.*, vol. 50, no. 3, pp. 507–524.

Wang, J.; Zhong, W.; Zhang, J. (2006): A general meshsize fourth-order compact difference discretization scheme for 3D Poisson equation. *Applied Mathematics and Computation*, vol. 183, no. 2, pp. 804–812.

Wu, C.-K. C.; Plesha, M. E. (2002): Essential boundary condition enforcement in meshless methods: boundary flux collocation method. *Int. J. Numer. Meth. Engng.*, vol. 53, pp. 499–514.

Zhang, J.; Tanaka, M.; Endo, M. (2004): Meshless analysis of potential problems in three dimensions with the hybrid boundary node method. *International Journal for Numerical Methods in Engineering.*, vol. 59, pp. 1147–1166.