Applied Mathematical Sciences, Vol. 11, 2017, no. 38, 1845 - 1870
HIKARI Ltd, www.m-hikari.com
https://doi.org/10.12988/ams.2017.76189

Discretization Coarsening for the Accurate Meshless Solution of Poisson Problems

Annamaria Mazzia

Dipartimento di Ingegneria Civile, Edile e Ambientale - ICEA
Università di Padova, Via F. Marzolo 9, 35131 Padova, Italy

Flavio Sartoretto

DAIS, Università Ca' Foscari Venezia
Via Torino 155, 30172 Mestre VE, Italy

Abstract

Meshless methods are nowadays emerging, alternative or subsidiary techniques to classical Finite Element (FE) and Finite Difference (FD) methods, for the numerical solution of Partial Differential Equations (PDE). Among the huge number of proposed meshless methods, the Meshless Petrov–Galerkin (MLPG) class of methods is one of the most promising. Recently, the Direct MLPG (DMLPG) methods were added to the MLPG class. MLPG is a Generalized FE method, while DMLPG is a Generalized FD method. Notwithstanding elegant theoretical analysis of meshless methods have been performed, effective, practical applications rely upon numerical experiments. That is why our paper is focused on performing neat numerical experiments on simple test problems. Adaptive methods are the most efficient for solving many problems, and MLPG techniques are well apt for adaptivity. The adaptive methods for MLPG techniques that one can find in the literature are based upon intricate norm estimations. This paper aims at proposing a simple yet effective technique for coarsening a discretization cloud, by deleting only "useless" nodes, hence allowing for reducing the computational cost without loosing accuracy. We analyze the effectiveness and efficiency of MLPG and DMLPG methods when coupled with our coarsening procedure. We point out some differences in the performances of these two methods.

Mathematics Subject Classification: 65N99

Keywords: Meshless methods, MLPG, DMLPG, Coarsening, Poisson problem

1 Introduction

Meshless methods are nowadays an emerging alternative to classical Finite Element (FE) and Finite Difference (FD) methods, for the numerical solution of Partial Differential Equations (PDE).

Among the huge number of proposed meshless methods, the Meshless Local Petrov–Galerkin (MLPG) class of methods, introduced by Atluri et. al [2], is one of the most promising. An extensive review of recent MLPG applications can be found in [19].

MLPG methods are truly meshless methods, i.e. they do not require any hidden mesh for performing auxiliary tasks, e.g. performing numerical integrations, like EFG methods [9]. As such, these methods are credited to be the best choice for developing adaptive approaches, relying upon changing the number of discretization nodes. Actually, FE and FD methods require remeshing, which is a time consuming and difficult to manage task [4]. MLPG methods do not require any mesh, just a cloud of nodes, hence in principle adding and/or deleting nodes is an easy task to perform.

A number of adaptive methods for MLPG techniques were proposed in the literature. See [6] for an updated list of adaptive methods. They are based upon intricate norm estimations. This paper aims at proposing a simple yet effective technique for coarsening a discretization cloud by deleting nodes without loosing accuracy.

Recently, the Direct MLPG (DMLPG) methods have been introduced in [16].

Though DMLPG can be viewed as a byproduct of MLPG, the former being obtained by applying Generalized Moving Least Squares (GMLS) in place of Moving Least Squares (MLS), the latter is deeply different form the former. While MLPG is a Generalized FE method, DMLPG is a Generalized FD method. Actually, MLPG aims at approximating the PDE solution, while DMLPG approximates linear functionals, e.g. the derivatives involved in the PDE problem.

Such a deep difference entails different behaviours when solving a PDE. In this paper we compare some characteristics of DMLPG vs MLPG, and we show the advantages and disadvantages of one method over the other.

While elegant theoretical analysis of meshless methods have been performed [18], effective, practical applications of meshless methods rely upon numerical experiments. They are necessary in order to identify those many ingredients (trial and test functions) which profoundly affect the efficiency and accuracy of these methods. That is why our present paper is focused on performing neat numerical experiments on simple test problems.

In our previous papers [13, 14], we proposed and analyzed a technique for refining a coarse discretization cloud.

In this paper, we devise a dual procedure allowing to reduce the number of nodes in a fine cloud, by deleting only "useless" nodes, hence allowing for reducing the computational cost without loosing accuracy.

This paper is organized as follows: Section 2 briefly recalls basic facts about the MLPG and DMLPG methods. Section 3 describes our coarsening procedure, and attached error estimation. Section 4 introduces our test problems. Section 5 analyzes our numerical results. Section 6 points out noteworthy results and draws our conclusions.

2 MLPG and DMLPG techniques

Let us consider the linear Poisson equation on the domain Ω

$$-\nabla^2 u(\underline{x}) = f(\underline{x}), \tag{1}$$

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 $\partial\Omega$

$$u = \bar{u} \quad \text{on } \Gamma_u, \quad \frac{\partial u}{\partial \underline{n}} \equiv q = \bar{q} \quad \text{on } \Gamma_q \tag{2}$$

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 $\underline{n}$.

Let us assume that the residual of eq. (1) is multiplied by a suitable test function τ. The divergence theorem is applied, thus obtaining the weak formulation for (1)

$$\int_\Omega \nabla u \cdot \nabla \tau d\Omega - \int_\Gamma (\nabla u \cdot \underline{n})\, \tau d\Gamma = \int_\Omega f\, \tau\, d\Omega, \quad \forall \tau \in \mathcal{S}, \tag{3}$$

for any τ in a suitable functional space $\mathcal{S}$.

In order to compute an approximation $\tilde{u} = \sum_i \tilde{u}_i \xi_i$ of the solution, a finite set of trial functions ξ_i is chosen. A plethora of MLPG methods have been proposed [1, 5], each of which can be identified by an appropriate choice of trial and test functions. For each method, many settings have been proposed (see e.g. [19]). Our implementation follows that one in [15].

In order to approximate the solution of our weak formulation, a set of N discretization nodes must be identified.

The set of trial functions, ξ_i, and test functions τ_i must be given, each of which is "centered" on node $\underline{x}_i$. A set of Local Weak Forms (LWF) is obtained by writing eq. (3) for each test function

$$\int_{\Omega_i} \nabla\tilde{u}\cdot\nabla\tau_i d\Omega - \int_{\Gamma_i^{(u)}} (\nabla\tilde{u}\cdot\underline{n})\,\tau_i d\Gamma = \int_{\Omega_i} f\,\tau_i\, d\Omega + \int_{\Gamma_i^{(q)}} (\nabla\tilde{u}\cdot\underline{n})\,\tau_i d\Gamma, \quad (4)$$

where $\Gamma_i^{(u)} = \partial\Omega_i \cap \partial\Gamma_u$ is the intersection of our local integration domain boundary with the *Dirichlet* boundary. Similarly, $\Gamma_i^{(q)} = \partial\Omega_i \cap \partial\Gamma_q$ is the intersection of our local integration domain boundary with the *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, contribute nothing, since $\tau_i = 0$ is imposed on $\partial\Omega_i$. The boundary conditions are managed using suitable techniques [10].

Our trial functions are the MLS shape functions obtained by suitable exponential RBF. The support of each trial function ξ_i, is a disc centered at $\underline{x}_i$, whose radius, called "trial radius" in the sequel, is r_i. Our test functions τ_i are tensor products of splines. The support of each test function τ_i, is a square centered at $\underline{x}_i$, whose half side length is the "test radius" ρ_i.

One crucial step in devising effective trial and test functions is identifying a pair (r_i, ρ_i) for each discretization node. These values depend upon the distance of that node from a number of its neighbours (see [13] for the details).

The Direct MLPG (DMLPG) technique is obtained by applying the Generalized Moving Least Squares (GMLS) method [8, 17] to the weak problem (4). The linear functionals in our weak formulation (4) are directly approximated by using a polynomial space. All the details of our implementation of the DMLPG technique for diffusion problems are given in [11].

3 Discretizations

Let us consider a cloud C which discretizes a domain Ω by enrolling N nodes $\underline{x}_1, \ldots, \underline{x}_N$.

The following measures are worth considering: The fill distance $h_{C,\Omega}$, and the separation distance, q_C, defined by [3]

$$h_{C,\Omega} = \sup_{\underline{x}\in\Omega} \min_{1\le i\le N} \|\underline{x} - \underline{x}_i\|, \quad q_C = \frac{1}{2}\min_{k\neq i} \|\underline{x}_k - \underline{x}_i\|, \quad 1 \le k, i \le N. \quad (5)$$

The accuracy of a given numerical method for solving a differential problem on Ω is like to essentially depend on $h_{C,\Omega}$ and q_C.

In order to perform a neat analysis, let us assume that our domain is the $[0,1]^2$ square.

Let us start with a discretization cloud which is the uniform grid on $[0,1]^2$ identified by intersecting the sides of the square and those three x–parallel and three y–parallel lines that divide the x- and y-side evenly into $n_x = n_y = 65$ parts. Let us call this uniform discretization U_0, the "level" $\ell = 0$ uniform discretization. The interval spacing is $h_0 = 1/65$, $N = 65 \times 65 = 4225$ nodes are identified.

3.1 Coarsening strategy

Let us introduce our coarsening strategy, which relies upon deleting nodes from a given, fine cloud C_0, hence producing a coarsened C_1 cloud, then the process can be iterated.

The first discretization cloud is set to $C_0 = U_0$, the $N = 4225$ node uniform grid. We call it the level $\ell = 0$ cloud.

The approximate solution $\tilde{u}$ is computed by using this cloud.

To obtain a coarser discretization we consider the Total Variation (TV) of the solution u

$$\|u\|_{TV} = \|\nabla u\|_1 = \int\int_{\Omega_i} (|u_x| + |u_y|)\, d\Omega,$$

where u_x and u_y are the partial derivatives of the solution.

The advantages of this "variation measure" are clearly explained in [20]: The variational methods which use the TV norm "allow for discontinuities but disfavor oscillations".

To approximate the TV on node $\underline{x}_i$ by taking our numerical solution $\tilde{u}$ into account, we define a "Local" Total Variation (LTV)

$$\|\tilde{u}\|_{TV,i} \simeq (|\tilde{u}_x(\underline{x}_i)| + |\tilde{u}_y(\underline{x}_i)|)\, |\Omega_i|,$$

$|\Omega_i|$ being the area of the i-th integration sub-domain in our weak formulation (4).

Concerning the partial derivatives $\tilde{u}_x(\underline{x}_i)$, $\tilde{u}_y(\underline{x}_i)$, when the DMLPG method is exploited, they are directly approximated by using the the GMLS approach.

When using the MLPG approach, the partial derivatives are estimated by differentiating the MLS solution [12].

Note that using DMLPG enables the derivatives to be approximated directly via the GMLS approach, while by exploiting MLPG the approximation is more involved. It relies upon estimating the derivatives of the approximated solution, hence of the MLS trial functions. From this point of view, DMLPG is superior to MLPG.

Let us assume that a coarse uniform "critical" cloud discretizing the domain is identified, whose nodes are called "critical nodes". Any node in the critical cloud is neither processed nor deleted, in order to avoid remaining with a riddled cloud, which does not properly discretize the domain. Usually, a uniform, sufficiently coarse discretization is used as the critical cloud, which is independent of the discretization level.

Let

$$\mu = \max_{i=1,\dots,N} \|\tilde{u}\|_{TV,i}.$$

Let us assume that a threshold parameter $0 < \gamma < 1$ is given. For each node $\underline{x}_i$ in the input cloud, which is not a critical node, the node is deleted if

$$\|\tilde{u}\|_{TV,i} < \gamma\mu,$$

i.e. the LTV being "low", the solution does not undergo large variations around $\underline{x}_i$, hence it can be deleted without loosing accuracy.

Once all the nodes in a given discretization C_ℓ have been processed, we say that a new "discretization cloud level" $C_{\ell+1}$ is ready.

A fresh approximated solution is computed by using the new cloud, and the approximation error is estimated.

The coarsening process is iterated, unless

- No more than a given, "small" number ℓ_{max} of discretization levels have been computed.
- The iterations are stopped when no node is deleted, i.e. the input cloud remains unchanged.

3.2 Error estimation

Concerning error estimations on our coarsened grids, let us assume that we start with the most refined cloud C_0 counting N_0 nodes.

Our coarsening procedure deletes nodes form C_0, hence producing less fine clouds C_ℓ, $\ell = 1, 2, \dots$, counting $n_0 > n_1 > n_2 > \dots$ nodes.

Let $\tilde{u}^{(\ell)}$ be the approximate solution computed via the nodes in C_ℓ. Let us assume that we estimate the (relative) error on the ℓ-th cloud, by

$$e_\ell^{(\ell)} = \max_{\underline{x} \in C_\ell} \frac{\left|u(\underline{x}) - \tilde{u}^{(\ell)}(\underline{x})\right|}{|u(\underline{x})|}. \tag{6}$$

Such an error estimation does not allow for a safe comparison between different clouds, since it takes into account only the nodes in the ℓ-th cloud. For appropriately comparing the errors on fine and coarse clouds, we must consider the "reconstruction" error with respect to a "reference" grid, which we assume to be the finest grid C_0. Our error measure on the ℓ-th cloud is

$$e_\ell = \max_{\underline{x} \in C_0} \frac{\left| u(\underline{x}) - \tilde{u}^{(\ell)}(\underline{x}) \right|}{|u(\underline{x})|}. \tag{7}$$

Recall that the MLPG technique aims at computing a set of coefficients $\tilde{u}_i^{(MLPG,\ell)}$, also called "fictitious" node values, each one assigned to node $\underline{x}_i \in C_\ell$. The ensuing approximate solution is

$$\tilde{u}^{(MLPG,\ell)}(\underline{x}) = \sum_{i=1}^{n_\ell} \tilde{u}_i^{(MLPG,\ell)} \xi_i^{(MLPG,\ell)}(\underline{x}), \tag{8}$$

where $\xi_i^{(MLPG,\ell)}$ is the i–th MLPG trial function.

Computing (7) requires the evaluation of $\tilde{u}^{(MLPG,\ell)}(\underline{x})$ on all the nodes in C_0, which is a super-set of C_ℓ (when $\ell > 0$).

Hence, when dealing with MLPG, like in FE methods, we can evaluate via eq. (8) the approximate solution on each node $\underline{x}_i \in C_0$, by using the same trial functions exploited for computing $\tilde{u}^{(MLPG,\ell)}$ via the cloud C_ℓ.

On the other hand, like FD methods, when the discretization C_ℓ is exploited, the DMLPG technique merely computes approximate solution values on the nodes in C_ℓ. Indeed, our DMLPG technique requires computing effective MLS shape functions, $w_i^{(DMLPG,\ell)}$, $i = 1, \ldots, N_\ell$. One could argue that by setting $\xi_i^{(DMLPG,\ell)} := w_i^{(DMLPG,\ell)}$, formula (8) should provide a sound reconstruction for the DMLPG, too, i.e. for any point in Ω. By numerical experiments we found that this is not true. In order to obtain an accurate reconstruction, we had to double the radius of the support for $w_i^{(DMLPG,\ell)}$, hence obtaining effective $\xi_i^{(DMLPG,\ell)}$ trial functions for DMLPG. This fact underlines another noteworthy problem of DMLPG: the "reconstruction" of the solution on arbitrary points in Ω is an open question (see page 3 in [16]).

4 Test problems

To check our adaptive strategy, we assign the forcing function f and compute the boundary conditions in eq. (1), so that its "test" solution is a function u undergoing large variations on a small portion of the domain.

First, we consider the classical Gaussian function, centered on a given point $P_0 = (x_0, y_0)$, i.e.

$$u(x, y) = \exp(-c \left((x - x_0)^2 + (y - y_0)^2 \right)) \,. \tag{9}$$

The parameter c is a large positive value that generates a high "hump" around P_0. In the sequel, we set $c = 200$ unless stated otherwise.

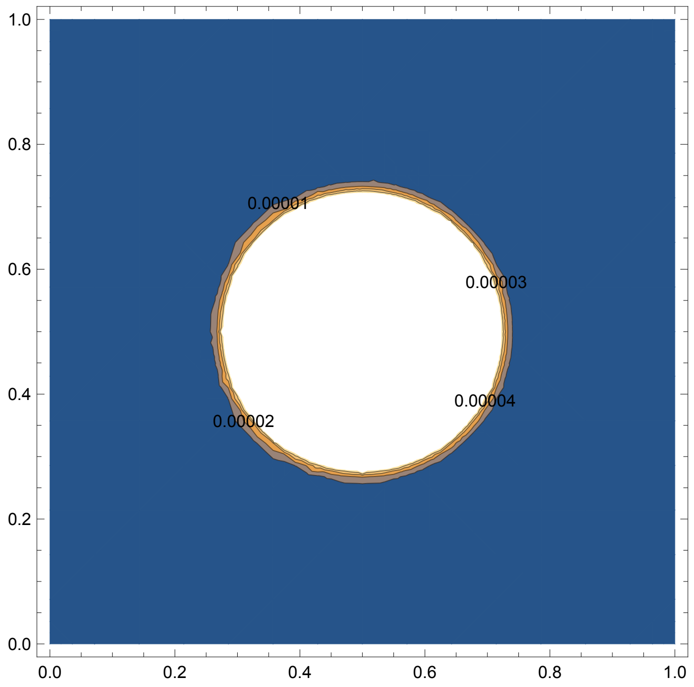


Figure 1: Contour regions for the solution of the test problem $P(GC)$.

Let us assume that we numerically solve the Poisson problem (1) in $\Omega = [0,1]^2$, having set the Dirichlet boundary conditions such that its solution is the function (9). The setting $P_0 = (1/2, 1/2)$, the centroid of our domain, corresponds to the 2D problem called $P(GC)$ in the sequel, where "GC" stands for "Gaussian–centroid".

This function displays a "bump" on the center of the domain. Figure 1 shows some contour levels of the surface.

Any coarsening procedure is likely to be effective when a large number of discretization nodes are left near the point P_0, where a large variation in u occurs. On the other hand, "far" away from P_0 the u values are small, and u does not display large variations, so the nodes can be safely deleted.

As a further test problem we consider, after [7]

$$u(x,y) = \tan^{-1}(1000\,x^2\,y^2 - 1). \tag{10}$$

This function displays a "hill" rising from the bottom left of $[0,1]^2$. Figure 2 shows some contour levels of the surface. In this case, an effective coarsening procedure is likely to leave the nodes around the bottom and left sides of the domain, yet the nodes can be deleted elsewhere.

Let us assume that we numerically solve the Poisson problem (1) in $\Omega = [0,1]^2$, having set the Dirichlet boundary conditions such that its solution is the function (10).

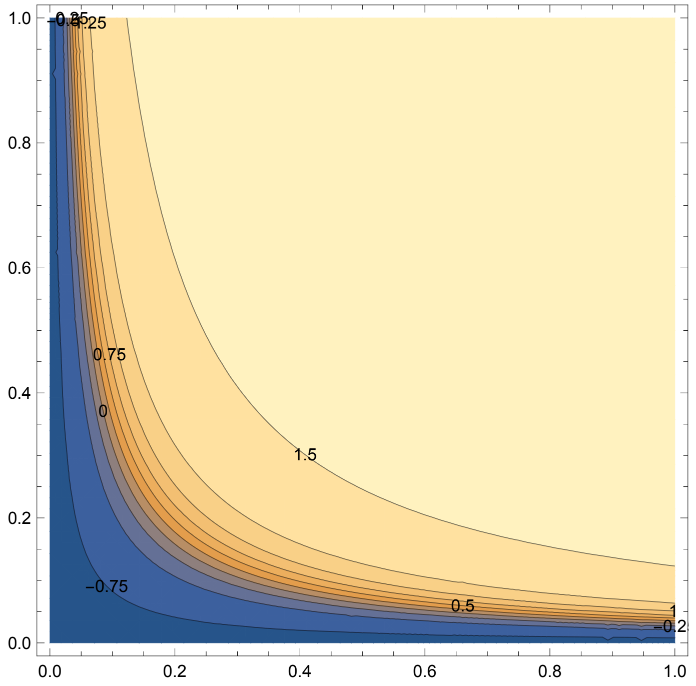


Figure 2: Contour regions for the solution of the test problem $P(T)$.

The ensuing differential problem is labeled test problem $P(T)$, where "T" stands for the "arcTan–based" solution.

5 Numerical results

In the sequel, we analyze our numerical results.

The trial and test radiuses in the MLPG and DMLPG methods were set to their respective optimal values, identified by numerical experiments.

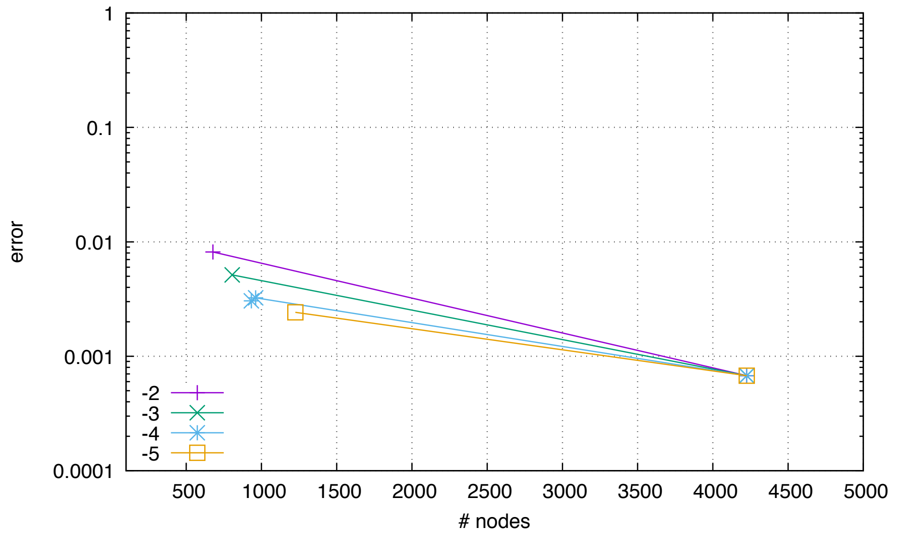
Our coarsening procedure is repeated until at most $\ell_{max} = 7$ coarsening steps have been performed. Moreover, we elect as the "critical cloud", the 17×17 uniform grid on $[0,1]^2$, counting "only" $N = 17^2 = 289$ nodes.

5.1 Gaussian–based test solution

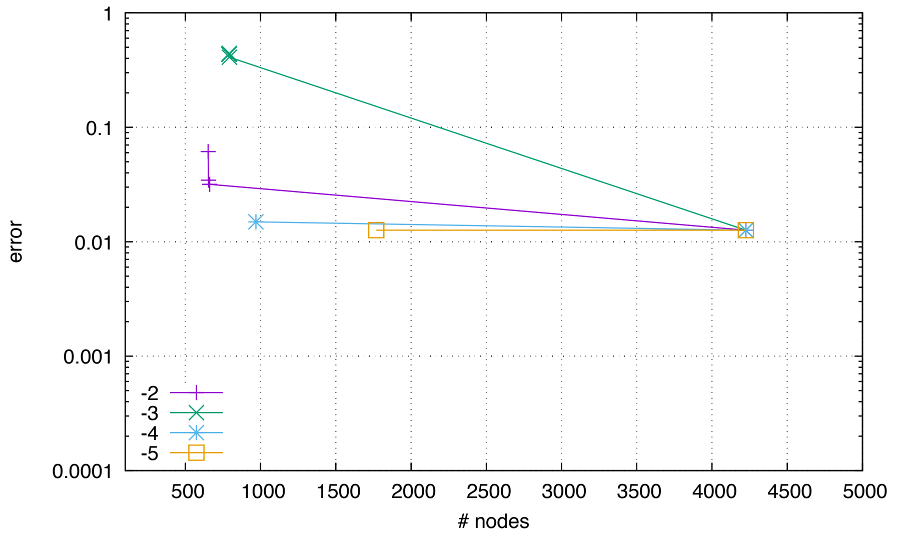
Concerning the test problem $P(GC)$, Table 1 reports our coarsening results. The first column shows the $\log_{10}$ value of the threshold coarsening parameter γ. The following columns show the cloud level number ℓ, the corresponding fill–distance $h_{C_\ell,\Omega}$, the number of cloud nodes N_ℓ, the error e_l raised in the approximation. The last–but–one column reports the ratio N_ℓ/N_0 of the number of coarsened nodes to that in the initial, finest cloud. The last column shows

MLPG						
$\log_{10}(\gamma)$	ℓ	$h_{C_\ell,\Omega}$	N_ℓ	e_ℓ	N_ℓ/N_0	e_ℓ/e_0
any	0	1.10E-02	4225	6.77E-04	1.00	1.00
-2	1	4.42E-02	677	8.16E-03	0.16	12.05
-3	1	4.42E-02	805	5.14E-03	0.19	7.59
-4	1	4.42E-02	961	3.24E-03	0.23	4.79
-4	2	4.42E-02	933	3.05E-03	0.22	4.51
-5	1	4.42E-02	1225	2.42E-03	0.29	3.57
DMLPG						
$\log_{10}(\gamma)$	ℓ	$h_{C_\ell,\Omega}$	N_ℓ	e_ℓ	N_ℓ/N_0	e_ℓ/e_0
any	0	1.10E-02	4225	1.26E-02	1.00	1.00
-2	1	4.42E-02	661	3.17E-02	0.16	2.52
-2	2	4.42E-02	654	3.45E-02	0.15	2.74
-2	3	4.42E-02	651	6.13E-02	0.15	4.87
-3	1	4.42E-02	793	4.10E-01	0.19	32.54
-3	2	4.42E-02	792	4.42E-01	0.19	35.08
-3	3	4.42E-02	790	4.34E-01	0.19	34.44
-3	4	4.42E-02	789	4.34E-01	0.19	34.44
-4	1	4.42E-02	969	1.49E-02	0.23	1.18
-5	1	4.42E-02	1769	1.26E-02	0.42	1.00

Table 1: Result summary for our test problem $P(GC)$. The fill distance $h_{C_\ell,\Omega}$, number of nodes N_ℓ, estimated error e_ℓ are shown for some threshold γ values, and coarsening levels ℓ. Note that at level $\ell = 0$, when either MLPG or DMLPG is enrolled, the given values does not depend on γ, hence they are shown only one time for each algorithm, corresponding to the pseudo-value "any" on the $\log_{10}\gamma$ column. The ratios N_ℓ/N_0 and e_ℓ/e_0 between node numbers and errors are also shown for providing an easy comparison of each coarsening level with the reference, initial cloud at level $\ell = 0$.



(a)



(b)

Figure 3: Test problem $P(GC)$. Errors obtained by exploiting the thresholds $\gamma = 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}$, vs the number of cloud nodes. Each error curve is labeled by the corresponding $\log_{10}\gamma$ value. Frame (a) reports MLPG errors, Frame (b) reports DMLPG ones.

the ratio e_ℓ/e_0 between the error raised when exploiting the coarsened cloud C_ℓ, and the error obtained by using the initial, finest cloud C_0.

Note that at level $\ell = 0$, the initial grid being the same, all values are the same, irrespective of γ value. This is the reason why the first row in the "MLPG" portion of the Table, pertaining to level $\ell = 0$, shows the value "any" for γ. In order to save space, this row is not repeated in the MLPG portion of our Table, when γ changes. The same holds true for the rows pertaining to the DMLPG method.

The values $\gamma = 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}$, are considered, corresponding to the $\log_{10}\gamma = -2, -3, -4, -5$ values, shown in the Table. Note that when $\gamma \leq 10^{-2}$ is set, our coarsening procedure drives to a so large error, that these threshold values cannot be considered as feasible ones. Unfeasible are also $\gamma \geq 10^{-5}$ values: no node is deleted, hence our procedure is useless.

Note that the separation distance is not reported, nor it is recorded in any of the Tables in the sequel. Actually, performing all our coarsening experiments we did not experience any change from the initial value q_{C_0}=1/128. This is the separation distance of the initial, uniform cloud, consisting of 65×65 uniformly distributed nodes on the unit square, hence q_{C_0}=(1/2) (1/64) = 1/128 (recall definition (5)). Deleting nodes inside portions of the domain leaves somewhere unchanged pairs of neighbouring nodes, hence the separation distance, which is driven by the distance of the closest nodes, is unchanged, too. Assume that a pair of neighbour nodes in the initial cloud is left. Then the separation distance does not change. In order to be more confident about this statement, the reader can also have a look at Figures 4, 5, 7 and 8 in the sequel, and the attached discussions.

Concerning the fill-distance, one can see that a change is reported at any first coarsening. The initial $h_{C_0,\Omega}$ =1.10E-2, grows $h_{C_1,\Omega}$ =4.42E-2, irrespective of γ values. This result confirms that the first coarsening indeed produce a change in the overall distribution, as tested by the N_ℓ/N_0 column values. By inspecting Table 1 one can see that successive coarsening does not affect the fill-distance value.

Figure 3 summarizes the error behaviors vs the number of cloud nodes. By inspecting this Figure, one can see that our procedure can allow for significant reductions in the number of cloud nodes, at the expense of a moderately to large increase in the approximation error.

Let us focus on the errors raised by the MLPG method (Frame (a)). One can see that when the coarsening threshold value is $\gamma = 10^{-2}$, the coarsening produces the largest reduction in the node number. On the other hand, the error becomes larger than for any other γ value. Setting $\gamma = 10^{-5}$ the error on the coarsened cloud is smaller than for any other γ value. By inspecting the upper part of Table 1, one can see that when $\gamma = 10^{-2}$ the coarsening step allows reducing the initial $N_0 = 4225$ nodes to N_1 =0.16N_0=677. The initial

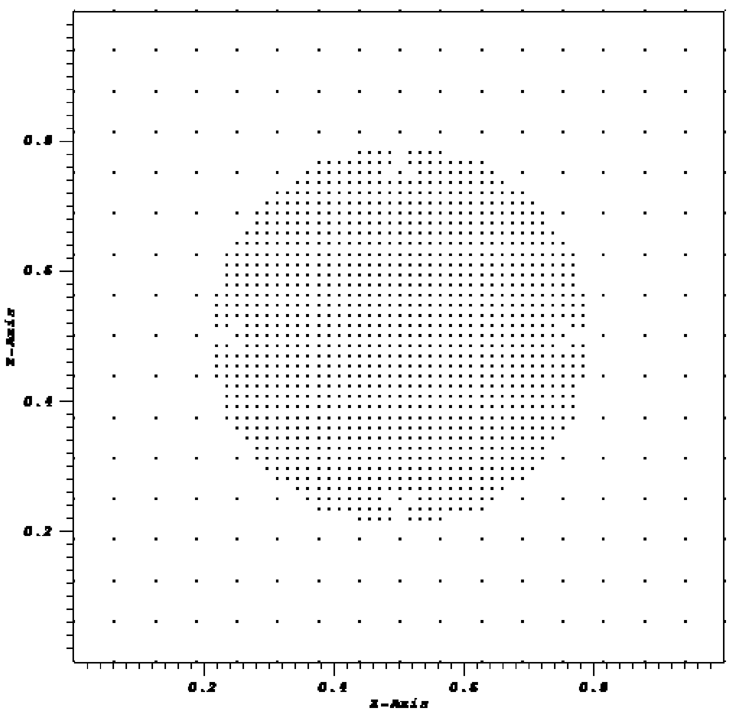


Figure 4: Test problem $P(GC)$. The nodes in the MLPG coarsened discretization level $\ell = 1$ are shown. The value $\gamma = 10^{-5}$ is considered.

error, e_0 =6.77E-04, grows to e_1 =8.16E-03, i.e. a large $e_1/e_0 = 12.5$ ratio is reported. We can say that an effective (16%) reduction in the number of cloud nodes is obtained, but the error of the approximated solution computed by exploiting the new, coarse cloud, is more than than 12 times higher than using the initial, finer grid.

By analyzing Table 1, one can argue that the best performance is likely to be obtained by setting $\gamma = 10^{-5}$: the first coarsening level cloud counts as few as N_1=0.29$\cdot N_0$=1225 nodes, while the error e_1 is "only" 3.57 times e_0.

Figure 4 shows the first level $\ell = 1$ discretization cloud obtained by MLPG. Recall that the exact solution is of "Gaussian" type, displaying large variation close to the domain center, while mild variation is displayed elsewhere. Accordingly, one can see that the nodes in our coarsened cloud "cluster" around the domain center. Here, the finest discretization is left unchanged, hence the separation distance for the coarsened cloud is the same as in C_0.

Let us now consider the errors obtained using the DMLPG method (Frame (b) in Figure 3). By inspecting the lower part of Table 1, one can see that unlike

MLPG, DMLPG usually drives many coarsening levels, up to $\ell = 4$. This is a first noteworthy difference one experiences when coarsening with DMLPG, respect to using MLPG. Recall that MLPG evaluates the LTV on the ground of a completely different approach respect to DMLPG. Hence, one can explain the difference in iterations by guessing that DMLPG produces a less smooth LTV than MLPG does. After one coarsening step has been performed, unlike in MLPG, the DMLPG LTV pertaining to a handful of nodes remains large, hence those nodes are deleted.

The first DMLPG coarsening level counts not many nodes more than the following $\ell = 2, 3, 4$ levels, hence performing one only coarsening step is the best procedure to elect.

By inspecting Frame (b) in Figure 3, one can guess that the best DMLPG setting is electable to be $\gamma = 10^{-4}$. An appreciable node reduction is observed (far better than for $\gamma = 10^{-5}$), and the error does not grow too much. The values reported in Table 1 confirm that the setting $\gamma = 10^{-4}$ provides a "good" 23% reduction in the number of nodes, while driving to a "mild" 1.18 times higher error.

By comparing Figure 3 and Table 1, one can see that when using DMLPG, by setting $\gamma = 10^{-3}$ the error at level $\ell = 1$ is as large as 32.54 times the error on the initial, finest cloud. Further coarsenings allow for deleting a mere node or a pair of nodes, and the error remains approximately 35 times larger than for the C_0 cloud. Note that such a large error inflation is not experienced neither when performing one coarsening step with $\log_{10}\gamma$=-2 (e_ℓ/e_0 =2.52), nor when $\log_{10}\gamma$=-4 (e_ℓ/e_0 =1.18). By plotting the attached coarsened clouds, one could see (not shown here, for brevity) that the nodes in the cloud obtained by one coarsening step for $\log_{10}\gamma$=-3 are a subset of the nodes obtained when $\log_{10}\gamma$=-4 is exploited. Such an overlap strongly suggests that a numerical instability is experienced when $\gamma = 10^{-3}$.

Figure 5 shows the discretization level $\ell = 1$ when the DMLPG coarsening is performed. The threshold value $\gamma = 10^{-5}$ is considered, like in Figure 4. By comparing the latter Figure with the former, one can see that the MLPG cloud counts less nodes "far" from the domain center than the DMLPG–coarsened cloud. Inside both plots, we can identify a disc, centered in $(1/2, 1/2)$ where the nodes are "close", and a "rarefied" area at the border of this disc. Recall that the "dense" area corresponds, as expected, to the "high variation" area of the exact solution, testified by the contour plot in Figure 1. The MLPG disc radius is approximately 0.14 large, while DMLPG one has a larger 0.17 radius. Note that the initial cloud is not rarefied around the domain center, hence the separation distance remains unchanged.

Table 2 reports the CPU seconds spent for solving our test problem and the mesh coarsenings.

By inspecting this Table one can see that the time for performing the

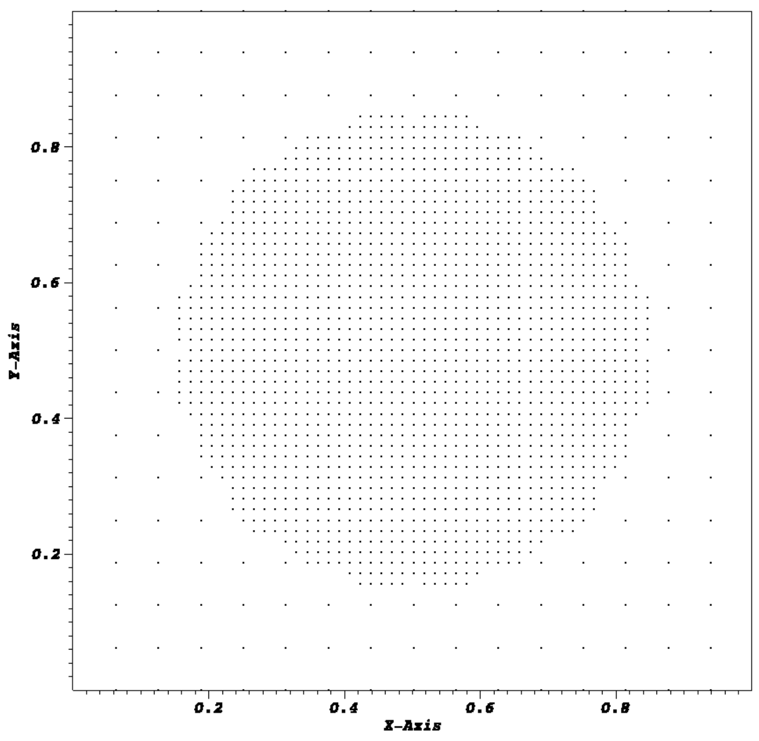


Figure 5: Test problem $P(GC)$. DMLPG discretization cloud at level $\ell = 1$, $\gamma = 10^{-5}$.

MLPG						
$\log_{10}(\gamma)$	ℓ	N_ℓ	$T_\ell^{(s)}$	$T_\ell^{(r)}$	$T_\ell^{(t)}$	$T_\ell^{(s)}/N_\ell$
-2	0	4225	17.53	1.41	18.97	4.15E-03
-2	1	677	3.14	0.30	22.41	4.63E-03
-3	0	4225	17.67	1.40	19.09	4.18E-03
-3	1	805	3.76	0.34	23.20	4.67E-03
-4	0	4225	17.94	1.39	19.36	4.25E-03
-4	1	961	4.53	0.42	24.30	4.71E-03
-4	2	933	4.51	0.43	29.24	4.84E-03
-5	0	4225	17.61	1.40	19.04	4.17E-03
-5	1	1225	5.84	0.59	25.47	4.77E-03
DMLPG						
$\log_{10}(\gamma)$	ℓ	N_ℓ	$T_\ell^{(s)}$	$T_\ell^{(r)}$	$T_\ell^{(t)}$	$T_\ell^{(s)}/N_\ell$
-2	0	4225	5.19	0.36	5.56	1.23E-03
-2	1	661	0.76	0.03	6.35	1.15E-03
-2	2	654	0.76	0.02	7.13	1.16E-03
-2	3	651	0.73	0.02	7.88	1.12E-03
-3	0	4225	5.30	0.36	5.68	1.25E-03
-3	1	793	0.86	0.03	6.57	1.08E-03
-3	2	792	0.84	0.03	7.45	1.07E-03
-3	3	790	0.83	0.03	8.31	1.05E-03
-3	4	789	0.83	0.04	9.17	1.05E-03
-4	0	4225	5.25	0.37	5.63	1.24E-03
-4	1	969	0.99	0.04	6.66	1.02E-03
-5	0	4225	5.21	0.36	5.60	1.23E-03
-5	1	1769	1.67	0.09	7.36	9.43E-04

Table 2: CPU seconds spent for solving our test problem $P(GC)$. The value $T_\ell^{(s)}$ reports the CPU seconds spent for performing the solution scheme, $T_\ell^{(r)}$ shows the seconds for the refinement procedure, while $T_\ell^{(t)}$ displays the total CPU time spent. The ratio $T_\ell^{(s)}/N_\ell$ is also shown.

refinements is very small, less than 1/10 of the time spent for computing the approximation.

In order to compare the efficiency of MLPG vs DMLPG, let us consider the ratios $T_\ell^{(s)}/N_\ell$, which measure the average computational "core" cost for performing the computations on each node. Only the time for computing the solution, refining, and evaluating errors, are considered. Let us assume that the best γ value for MLPG, i.e. $\gamma = 10^{-5}$, is set. At level $\ell = 1$ one can see that the MLPG core cost per node is 4.77E-3. At level $\ell = 1$ one can see that in the DMLPG best setting, $\gamma = 10^{-4}$, the DMLPG core cost per node is 1.02E-3. Hence the DMLPG cost per node is $4.77/1.02 \simeq 4.7$ times smaller then MLPG.

By analyzing Table 2 one can see that in all our cases DMLPG core cost per node is more than 4 times faster than MLPG. This result is a consequence of the use of GMLS in place of MLS. The direct approximation of the functionals proves to be appreciably faster than the approximation of the solution.

5.2 ArcTan–based test solution

Let us now consider the test problem $P(T)$, whose solution depicts a "hill" close to the bottom and left sides of the domain (cf. Figure 2).

Figure 6 shows the errors vs the number of cloud nodes. Let us focus on the errors raised by the MLPG method, shown in Frame (a). One can see that the best node reduction is obtained by setting $\gamma = 10^{-2}$, at the expense of a large error inflation. A smaller node reduction is experienced when $\gamma = 10^{-3}$, at the expense of a smaller error increase. By inspecting Table 3, we argue that the best choice for MLPG should be $\gamma = 10^{-2}$ which drives a substantial 40% reduction in the node number, yet enlarging the error only 1.38 times. The other γ values does not allow for a satisfactory deletion percentage.

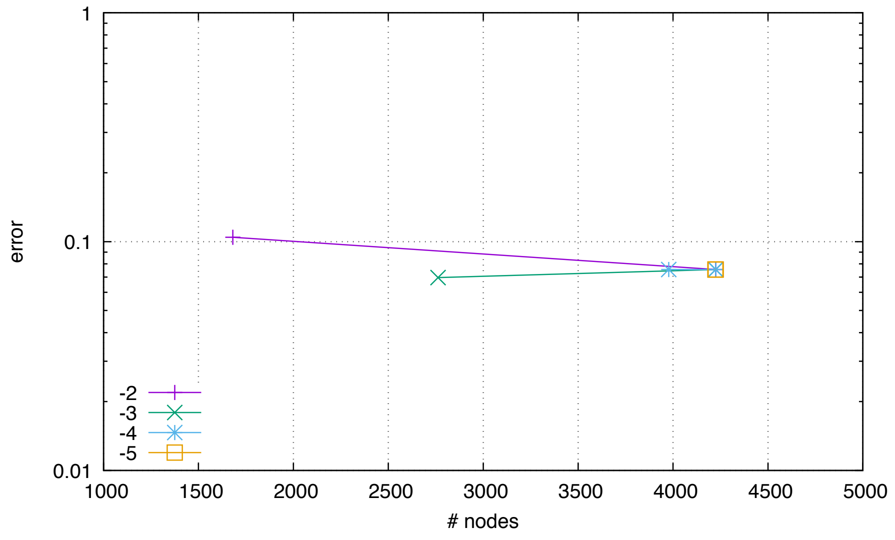
Recall that the test solution now displays a "hill" close to the bottom and left sides of the domain (cf. Figure 2). Figure 7 shows the MLPG coarsened cloud obtained by setting $\gamma = 10^{-2}$. One can see that many nodes remain in the bottom and left sides of the domain, i.e. where the test solution displays a large variation. Note that there is a portion of the domain where the initial grid is left unchanged, hence again the separation distance does not change after the coarsening.

Let us consider now the behavior of the DMLPG coarsening. By inspecting Frame (b) in Figure 6 and Table 3, one can see that, like in test problem $P(GC)$, DMLPG drives to more than one coarsening level, unlike MLPG.

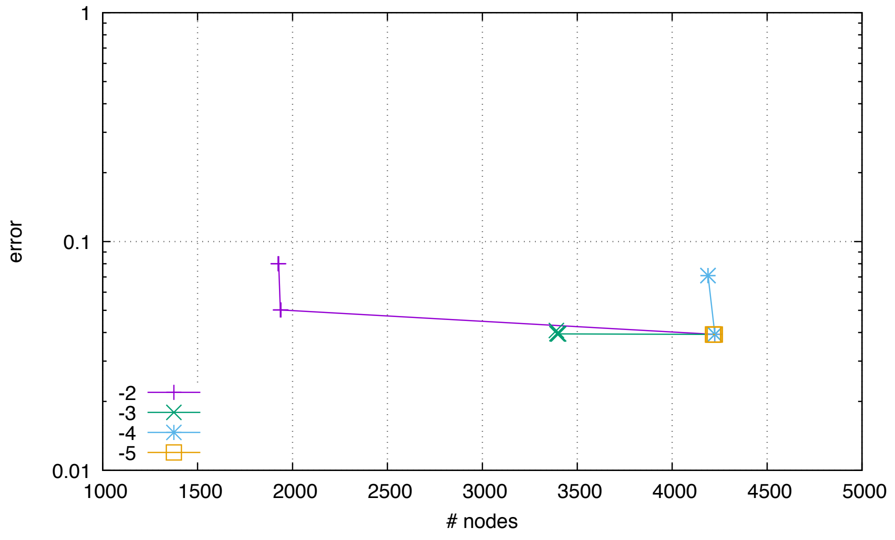
On the other hand, like in test problem $P(GC)$, the first coarsening level is always the most effective, the successive iterations allow for deleting an handful of nodes. Hence computing one only level proves better than iterating the coarsening.

MLPG						
$\log_{10}(\gamma)$	ℓ	$h_{C_\ell,\Omega}$	N_ℓ	e_ℓ	N_ℓ/N_0	e_ℓ/e_0
any	0	1.10E-02	4225	7.55E-02	1.00	1.00
-2	1	4.42E-02	1681	1.04E-01	0.40	1.38
-3	1	4.42E-02	2762	6.96E-02	0.65	0.92
-4	1	4.42E-02	3977	7.55E-02	0.94	1.00
-5	1	1.56E-02	4223	7.55E-02	1.00	1.00
DMLPG						
$\log_{10}(\gamma)$	ℓ	$h_{C_\ell,\Omega}$	N_ℓ	e_ℓ	N_ℓ/N_0	e_ℓ/e_0
any	0	1.10E-02	4225	3.93E-02	1.00	1.00
-2	1	4.42E-02	1939	5.02E-02	0.46	1.28
-2	2	4.42E-02	1936	5.02E-02	0.46	1.28
-2	3	4.42E-02	1927	8.00E-02	0.46	2.04
-2	4	4.42E-02	1924	8.00E-02	0.46	2.04
-3	1	4.42E-02	3402	3.94E-02	0.81	1.00
-3	2	4.42E-02	3400	3.95E-02	0.80	1.01
-3	3	4.42E-02	3396	3.95E-02	0.80	1.01
-3	4	4.42E-02	3394	3.95E-02	0.80	1.01
-3	5	4.42E-02	3393	3.96E-02	0.80	1.01
-3	6	4.42E-02	3390	4.08E-02	0.80	1.04
-4	1	1.95E-02	4189	7.09E-02	0.99	1.80
-4	2	1.95E-02	4188	7.09E-02	0.99	1.80
-5	1	1.75E-02	4218	3.92E-02	1.00	1.00
-5	2	1.75E-02	4216	3.92E-02	1.00	1.00
-5	3	1.75E-02	4215	3.92E-02	1.00	1.00

Table 3: Result summary for our test problem $P(T)$. Concerning the meaning of the symbols, see Table 1



(a)



(b)

Figure 6: Analogous to Figure 3, concerning test problem $P(T)$.

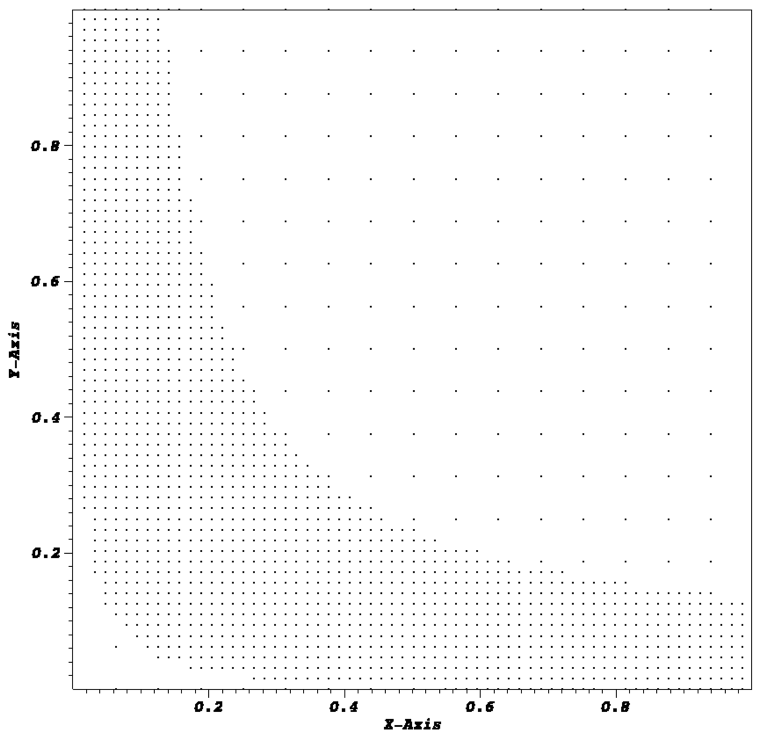


Figure 7: Test problem $P(T)$. MLPG coarsened cloud, $\ell = 1$, $\gamma = 10^{-2}$.

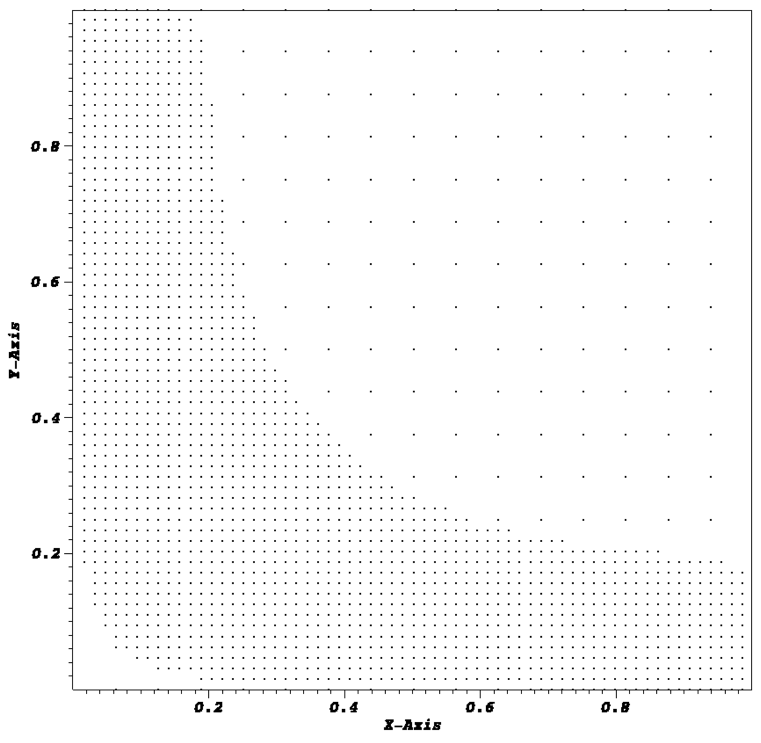


Figure 8: Test problem $P(T)$. DMLPG coarsened cloud, $\ell = 1$, $\gamma = 10^{-2}$.

By considering Frame (b) in Figure 6, one can see that best node reduction is obtained when $\gamma = 10^{-2}$. Smaller γ values, i.e. $\gamma = 10^{-3}, 10^{-4}, 10^{-5}$ does not produce an appreciable reduction in the number of cloud nodes. By analyzing Table 3, one can say that the best choice for DMLPG computations is performing one only coarsening when $\gamma = 10^{-2}$. This setting allows for an appreciable 46% reduction in the node number, producing a "mild" 1.28 times larger error.

Figure 8 shows the DMLPG coarsened cloud obtained by setting $\gamma = 10^{-2}$. One can see that, like previously for MLPG, the nodes remaining after the coarsening are in the domain part where the exact solution undergoes large variations (see Figure 2).

By comparing Figure 8 with Figure 7, one can see that the coarsened clouds are similar. Again note that there are portions of the domain where the initial cloud node inter-distances are not affected. Hence, the coarsening does not change the separation distance.

Table 4 reports the CPU seconds spent for solving our test problem and the

MLPG						
$\log_{10}(\gamma)$	ℓ	N_ℓ	$T_\ell^{(s)}$	$T_\ell^{(r)}$	$T_\ell^{(t)}$	$T_\ell^{(s)}/N_\ell$
-2	0	4225	17.52	1.40	18.94	4.15E-03
-2	1	1681	7.00	0.63	26.57	4.16E-03
-3	0	4225	17.52	1.36	18.90	4.15E-03
-3	1	2762	11.28	0.94	31.12	4.08E-03
-4	0	4225	17.60	1.37	18.98	4.16E-03
-4	1	3977	16.50	1.28	36.77	4.15E-03
-5	0	4225	17.56	1.37	18.95	4.16E-03
-5	1	4223	17.36	1.36	37.66	4.11E-03
DMLPG						
$\log_{10}(\gamma)$	ℓ	N_ℓ	$T_\ell^{(s)}$	$T_\ell^{(r)}$	$T_\ell^{(t)}$	$T_\ell^{(s)}/N_\ell$
-2	0	4225	5.17	0.35	5.53	1.22E-03
-2	1	1939	1.81	0.10	7.44	9.32E-04
-2	2	1936	1.86	0.10	9.40	9.63E-04
-2	3	1927	1.85	0.10	11.35	9.61E-04
-2	4	1924	1.80	0.10	13.26	9.36E-04
-3	0	4225	5.20	0.35	5.57	1.23E-03
-3	1	3402	3.76	0.24	9.57	1.11E-03
-3	2	3400	3.75	0.24	13.56	1.10E-03
-3	3	3396	3.77	0.24	17.57	1.11E-03
-3	4	3394	3.75	0.24	21.56	1.10E-03
-3	5	3393	3.75	0.24	25.56	1.11E-03
-3	6	3390	3.77	0.24	29.56	1.11E-03
-4	0	4225	5.16	0.37	5.56	1.22E-03
-4	1	4189	5.08	0.36	11.00	1.21E-03
-4	2	4188	5.10	0.34	16.44	1.22E-03
-5	0	4225	5.17	0.37	5.57	1.22E-03
-5	1	4218	5.20	0.36	11.13	1.23E-03
-5	2	4216	5.18	0.35	16.66	1.23E-03
-5	3	4215	5.19	0.34	22.19	1.23E-03

Table 4: Analogous to Table 2. CPU times spent when solving our test problem $P(T)$.

mesh coarsening. By analyzing this Table one can see that the best DMLPG setting $\gamma = 10^{-2}$, the coarsening level $\ell = 1$ needs a 4.16E-3 core cost per node to be computed. On the other hand in the best MLPG setting $\gamma = 10^{-2}$, the coarsening level $\ell = 1$ needs a 9.32E-4 core cost per node to be computed. Hence DMLPG core time is $4.16/0.932 \simeq 4.5$ times smaller than MLPG one.

Like in our test example $P(GC)$, DMLPG core cost is more than four times smaller than MLPG one, confirming the higher efficiency of GMLS approach over MLS one.

6 Conclusions

A coarsening procedure for the accurate MLPG/DMLPG solution of the Poisson problem has been introduced and numerically analyzed.

The following points are worth considering.

- Our coarsening procedure allows for effectively reducing the number of discretization nodes, without enlarging the error too much. Both these two competing tasks can be achieved by identifying a suitable value for the threshold parameter γ acting on the local TV, in order to effectively drive the deletion strategy. The value is problem–dependent.

- When our coarsening procedure is combined with the MLPG method, one coarsening level only is computed, while when the DMLPG procedure is exploited, more than one coarsening level is computed. However, the coarsening levels $\ell = 2, 3, \ldots$ perform a so small reduction in the overall node number, that they are useless for reducing the DMLPG computational cost. This difference suggests that the DMLPG LTV computation gives an handful of large values, i.e. it is “more unstable” than when using the MLPG-driven approach. Summarizing, achieving one only coarsening level is the best choice in any case.

- While the MLPG methods is numerically stable vs the coarsening threshold γ value, DMLPG can raise abrupt changes in the error. Some γ values trigger numerical instabilities in the DMLPG method, hence its exploitation in our coarsening procedure seems questionable in practical applications.

- The DMLPG “core cost”, i.e. the cost for computing the solution, refining, and evaluating the errors is more than four times smaller than MLPG. Such a better performance is to be ascribed to the use of GMLS procedure in DMLPG, in place of MLS in MLPG.

- In order to perform an accurate error estimation for comparing our results on different discretization clouds, the reconstruction (i.e. evaluation) of the approximate solution on the reference, finest cloud must be performed. Such a reconstruction is amenable when the MLPG method is exploited, while it is more cumbersome when DMLPG is exploited. This is a noteworthy advantage of MLPG over DMLPG.

Future work. We aim at applying our coarsening procedure to the solution of time–dependent problems. By splitting the operator into one spatial and one temporal part, at each time step we can coarsen the spatial discretization cloud according to the evolution of the problem solution, in order to improve efficiency, without loosing accuracy.

Acknowledgements. Some computations were performed on SCSCF, a multiprocessor cluster system owned by Universitá Ca' Foscari Venezia, Italy.

References

[1] S. N. Atluri, *The Meshless Method (MLPG) for Domain & BIE Discretizations*, Tech Science, Forsyth, Georgia, 2004.

[2] S. N. Atluri and Shengping Shen, The Meshless Local Petrov-Galerkin (MLPG) method: A simple & less-costly alternative to the finite element and boundary element methods, *Computer Modeling in Engineering and Sciences*, **3** (2002), no. 1, 11-52.
https://doi.org/10.3970/cmes.2002.003.011

[3] G. E. Fasshauer, *Meshfree Approximation Methods with MATLAB*, volume 6 of Interdisciplinary Mathematical Sciences, World Scientific Publishing Co., Singapore, 2007. https://doi.org/10.1142/6437

[4] P. J. Frey and P. L. George, *Mesh Generation*, Wiley, 2010.
https://doi.org/10.1002/9780470611166

[5] T.-P. Fries and H.-G. Matthies, *Classification and Overview of Meshfree Methods*, Technical Report 2003-3, Technical University Braunschweig, Brunswick, Germany, 2004.

[6] M. Kamranian, M. Dehghan and M. Tatari, An adaptive meshless local Petrov-Galerkin method based on a posteriori error estimation for the boundary layer problems, *Applied Numerical Mathematics*, **111** (2017), 181-196. https://doi.org/10.1016/j.apnum.2016.09.007

[7] B. B. T. Kee, G. R. Liu, G. Y. Zhang and C. Lu, A residual based error estimator using radial basis functions, *Finite Elem. Anal. Des.*, **44** (2008), no. 9-10, 631-645. https://doi.org/10.1016/j.finel.2008.02.002

[8] D. Levin, The approximation power of moving least-squares, *Math. Comp.*, **67** (1998), no. 224, 1517-1531. https://doi.org/10.1090/s0025-5718-98-00974-0

[9] Y. Y. Lu, T. Belytschko and L. Gu, A new implementation of the element free Galerkin method, *Comp. Methods App. Mech. Eng.*, **113** (1994), 397-414. https://doi.org/10.1016/0045-7825(94)90056-6

[10] A. Mazzia, G. Pini and F. Sartoretto, Accurate MLPG solution for 3D potential problems, *Computer Modeling in Engineering & Sciences*, **36** (2008), no. 1, 43-63. https://doi.org/10.3970/cmes.2008.036.043

[11] A. Mazzia, G. Pini and F. Sartoretto, Numerical investigation on direct MLPG for 2d and 3d potential problems, *Computer Modeling in Engineering & Sciences*, **88** (2012), 183-210.

[12] A. Mazzia, G. Pini and F. Sartoretto, Meshless techniques for anisotropic diffusion, *Appl. Math. Comp.*, **236** (2014), 54-66. https://doi.org/10.1016/j.amc.2014.03.032

[13] A. Mazzia, G. Pini and F. Sartoretto, MLPG Refinement Techniques for 2D and 3D Diffusion Problems, *Computer Modeling in Engineering and Sciences*, **102** (2014), no. 6, 475-497.

[14] A. Mazzia, G. Pini and F. Sartoretto, A DMLPG Refinement Technique for 2D and 3D Potential Problems, *Computer Modeling in Engineering and Sciences*, **108** (2015), no. 4, 239-262.

[15] A. Mazzia and F. Sartoretto, Meshless solution of potential problems by combining radial basis functions and tensor product ones, *Computer Modeling in Engineering & Sciences*, **68** (2010), no. 1, 95-112. https://doi.org/10.3970/cmes.2010.068.095

[16] D. Mirzaei and R. Schaback, Direct meshless local Petrov-Galerkin (DMLPG) method: A generalized MLS approximation, *Applied Numerical Mathematics*, **68** (2013), 73-82. https://doi.org/10.1016/j.apnum.2013.01.002

[17] D. Mirzaei, R. Schaback and M. Dehghan, On generalized moving least squares and diffuse derivatives, *IMA Journal of Numerical Analysis*, **32** (2012), no. 3, 983-1000. https://doi.org/10.1093/imanum/drr030

[18] R. Schaback, A computational tool for comparing all linear PDE solvers, *Adv. Comput. Math.*, **41** (2015), no. 2, 333-355.
https://doi.org/10.1007/s10444-014-9360-5

[19] J. Sladek, P. Stanak, Z.-d. Han, V. Sladek and S. N. Atluri, Applications of the MLPG method in engineering & sciences: A review, *Computer Modeling in Engineering & Sciences*, **92** (2013), no. 5, 423-475.

[20] G. Strang, *Computational Science and Engineering*, Wellesley-Cambridge Press, Wellesley, MA, 2007.

Received: June 20, 2017; Published: July 12, 2017