Applied Numerical Mathematics 115 (2017) 68–81

Contents lists available at ScienceDirect

Applied Numerical Mathematics

www.elsevier.com/locate/apnum

Piece-wise moving least squares approximation ☆

Wen Li [a,b], Guohui Song [b], Guangming Yao [b,*]

[a] School of Mathematics, Taiyuan University of Technology, 030024, China
[b] Department of Mathematics, Clarkson University, Potsdam, NY 13699, USA

ARTICLE INFO

Article history:
Received 13 September 2016
Received in revised form 31 December 2016
Accepted 4 January 2017
Available online 10 January 2017

Keywords:
Moving least square
RBF
PDE
Interpolation

ABSTRACT

The standard moving least squares (MLS) method might have an expensive computational cost when the number of test points and the dimension of the approximation space are large. To reduce the computational cost, this paper proposes a piece-wise moving least squares approximation method (PMLS) for scattered data approximation. We further apply the PMLS method to solve time-dependent partial differential equations (PDE) numerically. It is proven that the PMLS method is an optimal design with certain localized information. Numerical experiments are presented to demonstrate the efficiency and accuracy of the PMLS method in comparison with the standard MLS method in terms of accuracy and efficiency.

Published by Elsevier B.V. on behalf of IMACS.

1. Introduction

The moving least squares (MLS) [21] is a popular method of approximating a function from a set of its values at some scattered data points. It is a flexible meshless method that does not require the construction of a mesh on the domain. It has been widely used in curve and surface fitting [7,21,27,32], and many meshless weak-form methods for solving PDEs, such as the diffuse element method (DEM) [29], the element-free Galerkin methods (EFG) [5] and the meshless local Petrov–Galerkin (MLPG) approach [3], etc.

For each test point, the approximation function of the MLS method is assumed to lie in a finite-dimensional approximation space with certain basis and its coefficients are calculated through a weighted least-squares problem with the weights concentrating at the region around the point. We point out that the weight is depending (moving) on the test point. That is, for different test points, we need to solve different least square problems with the size given by the dimension of the approximation space. The MLS might have a very expensive computational cost when the number of the test points and the dimension of the approximation space are large.

To reduce the computational cost of the standard MLS, many techniques have been introduced in the literature [23]. We propose in this paper a piece-wise moving least squares (PMLS) method. The standard MLS considers a "point-wise" weight that is different for each test point. We will use a "piece-wise" weight instead in the PMLS method. Specifically, we would decompose the domain into some small and disjoint regions and define the weight function for each region rather than for each test point. It is in particular useful in reducing the computational cost when there are many test points lying in the same region.

Moreover, we will consider the PMLS in the view of optimal recovery. In particular, we shall show that it is an optimal design with certain localized information. For the approximation spaces in our numerical experiments, we will focus on two

☆ This work is supported in part by grant NSF-DMS 1521661.
* Corresponding author.
E-mail addresses: liw2@clarkson.edu (W. Li), gsong@clarkson.edu (G. Song), gyao@clarkson.edu (G. Yao).

http://dx.doi.org/10.1016/j.apnum.2017.01.001
0168-9274/Published by Elsevier B.V. on behalf of IMACS.

different basis functions: polynomials and radial basis functions (RBF). The use of RBFs in engineering and sciences leads to many advantages in terms of simplification in high-dimensional problems [8,10,12]. We will test the performance of the proposed PMLS method in scattered data approximation of some benchmark test functions.

We will further apply the PMLS method in numerical solutions of some time-dependent PDEs. There are generally two kinds of approaches for solving time-dependent PDEs numerically [30]. One way is to convert the PDE to a system of ordinary differential equations (ODE) [1,2]. Many traditional techniques of solving system of ODEs could be employed afterwards. The other approach is to apply time discretization and spatial discretization respectively [4,6,16]. Two approaches are basically the same except that the first approach transfers a PDE into a system of ODEs by discretization of the spatial domain first. This is in contrary to the second approach that usually discretizes the time domain first and then discretizes the spatial domain afterwards. A series of PDEs usually needs to be solved in the second approach. We will take the latter approach in this paper. When an explicit time stepping method is used, the time dependent PDE becomes a fitting problem of the solution to the PDEs at the previous time step and approximation of its derivatives. Specifically, we will use the traditional forward Euler formula to discretize the time domain and then apply the PMLS method in spatial discretization of each time step.

The rest of this paper is organized as follows. In section 2, we introduce the piece-wise moving least squares (PMLS) method for scattered data approximation. We prove that the PMLS method is an optimal design with certain localized information in Section 3. We implement in Section 4 the PMLS method in both scattered data approximation and numerical solution of time-dependent PDEs. In particular, we compare the accuracy and efficiency of the proposed PMLS method with the standard MLS method there.

2. Piece-wise moving least squares

We will introduce the PMLS method for scattered data approximation in this section. To this end, we first review the standard MLS method and some notations.

We first give a brief description of the standard MLS as described in [13]. Suppose $\Omega \subseteq \mathbb{R}^d$ is the domain of an unknown function f and $\boldsymbol{x}_j \in \Omega$, $1 \le j \le N$, are some scattered data points in the domain. We are given function values $\boldsymbol{f} = (f(\boldsymbol{x}_j) : 1 \le j \le N)$ on such data points. Assume the approximation space $\mathcal{U}$ is a finite-dimensional space with basis $\{u_1, u_2, \ldots, u_m\}$. For any $\boldsymbol{x} \in \Omega$ and a weight function $w : \mathbb{R} \to \mathbb{R}$, we define a weighted ℓ^2 inner product for functions g_1, g_2 on Ω

$$\langle g_1,\ g_2\rangle_{w_{\boldsymbol{x}}} = \sum_{i=1}^{N} g_1(\boldsymbol{x}_i) g_2(\boldsymbol{x}_i) w(\|\boldsymbol{x}_i - \boldsymbol{x}\|),$$

where $\|\cdot\|$ is the Euclidean norm. For a test point $\boldsymbol{y} \in \Omega$, we will try to find the best approximation function $T_{\boldsymbol{y}}$ in the approximation space $\mathcal{U} = \operatorname{span}\{u_1, \ldots, u_m\}$ such that it is "close" to f with respect to the norm $\|\cdot\|_{w_{\boldsymbol{y}}}$ induced by the above weighted inner product $\langle\cdot,\cdot\rangle_{w_{\boldsymbol{y}}}$. That is, we define

$$T_{\boldsymbol{y}} = \operatorname{argmin}_{g\in\mathcal{U}} \|g - f\|_{w_{\boldsymbol{y}}}. \tag{2.1}$$

We next give a reformulation of $T_{\boldsymbol{y}}$ in the convenience of computation. We also assume that for any $\boldsymbol{y} \in \Omega$, $\mathcal{U} = \operatorname{span}\{u_1(\cdot - \boldsymbol{y}), \ldots, u_m(\cdot - \boldsymbol{y})\}$. For example, the space of polynomials with degree no more than m would satisfy this assumption. We could then write the best approximation function $T_{\boldsymbol{y}}$ in the following form

$$T_{\boldsymbol{y}}(\boldsymbol{x}) = \sum_{j=1}^{m} c_j(\boldsymbol{y}) u_j(\boldsymbol{x} - \boldsymbol{y}), \quad \boldsymbol{x} \in \Omega,$$

where the coefficients $\boldsymbol{c}(\boldsymbol{y}) = (c_1(\boldsymbol{y}), c_2(\boldsymbol{y}), \ldots, c_m(\boldsymbol{y}))$ are given by

$$\boldsymbol{c}(\boldsymbol{y}) := \operatorname{argmin}_{\boldsymbol{a}\in\mathbb{R}^m} \sum_{i=1}^{N} \left[f_i - \sum_{j=1}^{m} a_j u_j(\boldsymbol{x}_i - \boldsymbol{y}) \right]^2 w(\|\boldsymbol{x}_i - \boldsymbol{y}\|). \tag{2.2}$$

We remark that it is a quadratic form and we are able to find the closed form of $\boldsymbol{c}(\boldsymbol{y})$. Let

$$\mathsf{G}(\boldsymbol{y}) := \left[\langle u_i(\cdot - \boldsymbol{y}), u_j(\cdot - \boldsymbol{y})\rangle_{w_{\boldsymbol{y}}}\right]_{i,j=1}^{m},$$

and

$$\boldsymbol{L}_{\boldsymbol{y}}(f) = \left[\langle f, u_j(\cdot - \boldsymbol{y})\rangle_{w_{\boldsymbol{y}}}\right]_{j=1}^{m}. \tag{2.3}$$

It follows from a direct calculation (also presented in [11,13]) that

$$\boldsymbol{c}(\boldsymbol{y}) = [\mathsf{G}(\boldsymbol{y})]^{-1} \boldsymbol{L}_{\boldsymbol{y}}(f). \tag{2.4}$$

We are assuming that the Gram matrix $\mathsf{G}(\boldsymbol{y})$ is invertible in the above equality. Otherwise, we will use its Moore–Penrose pseudo-inverse instead.

For each test point $\boldsymbol{y}$, we shall use $T_{\boldsymbol{y}}(\boldsymbol{y})$ to approximate the unknown function value $f(\boldsymbol{y})$. That is, we approximate the unknown function f by $\tilde{f}$ defined by $\tilde{f}(\boldsymbol{y}) = T_{\boldsymbol{y}}(\boldsymbol{y})$ for $\boldsymbol{y} \in \Omega$.

It is direct to observe that the coefficients $\boldsymbol{c}$ depend on the test point $\boldsymbol{y}$. For example, when $\mathcal{U}$ consists of polynomials, $T_{\boldsymbol{y}}$ is a polynomial with coefficients depending on $\boldsymbol{y}$ and the approximation function $\tilde{f}$ could be viewed as a "point-wise" polynomial in this sense. We also note that the approximation function $\tilde{f}$ may not be a polynomial. When the dimension m of the approximation space is small, it is possible to solve the above linear system analytically. For example, Shepard's method considers $m = 1$ and $u_1 = 1$. The coefficients $\boldsymbol{c}$ in Shepard's method [28] are given by

$$\boldsymbol{c} = \frac{1}{\sum_{j=1}^{N} w(\|\boldsymbol{x}_i - \boldsymbol{y}\|)} \boldsymbol{L}_{\boldsymbol{y}}(f) = \frac{\sum_{j=1}^{N} f(\boldsymbol{x}_i) w(\|\boldsymbol{x}_i - \boldsymbol{y}\|)}{\sum_{j=1}^{N} w(\|\boldsymbol{x}_i - \boldsymbol{y}\|)}.$$

However, the computational cost of solving the linear system will increase as m becomes large. In particular, when the number of test points is also large, the computational cost might be very expensive since we need to solve a different linear system for each test point.

To reduce the computational cost, we shall next introduce the PMLS method that considers a piece-wise weight function instead. We first partition the domain Ω into some small subsets Ω_j, $1 \le j \le n$ and select a point $\boldsymbol{t}_j$ in each subset Ω_j. For any $\boldsymbol{y} \in \Omega$, we have $\boldsymbol{y} \in \Omega_k$ for some $1 \le k \le n$ and we shall consider the following best approximation S_k of f in $\mathcal{U}$ with respect to the norm induced by $\langle \cdot, \cdot \rangle_{w_{\boldsymbol{t}_k}}$. More specifically, for any $1 \le k \le n$, we define

$$S_k(\boldsymbol{y}) = \sum_{j=1}^{m} \lambda_{k,j} u_j(\boldsymbol{y} - \boldsymbol{t}_k), \tag{2.5}$$

where $\boldsymbol{\lambda}_k = (\lambda_{k,j} : 1 \le j \le m)$ is given by

$$\boldsymbol{\lambda}_k := \operatorname{argmin}_{\boldsymbol{a} \in \mathbb{R}^m} \sum_{i=1}^{N} \left[f(\boldsymbol{x}_i) - \sum_{j=1}^{m} a_j u_j(\boldsymbol{x}_i - \boldsymbol{t}_k) \right]^2 w(\|\boldsymbol{x}_i - \boldsymbol{t}_k\|). \tag{2.6}$$

The overall approximation function is defined as

$$S(\boldsymbol{y}) = \sum_{k=1}^{n} S_k(\boldsymbol{y}) \chi_{\Omega_k}(\boldsymbol{y}), \quad \boldsymbol{y} \in \Omega,$$

where χ_A denotes the indicator function, that is $\chi_A(x) = 1$ if $x \in A$ and 0 otherwise. Similarly, we have a closed form solution of the coefficients $\boldsymbol{\lambda}_k$. It follows from a direct computation that

$$\boldsymbol{\lambda}_k = [\mathsf{G}(\boldsymbol{t}_k)]^{-1} \boldsymbol{L}_{\boldsymbol{t}_k}(f). \tag{2.7}$$

We observe from the above formulation that we only need to solve a linear system for each subset Ω_k of the domain Ω rather than for each test point $\boldsymbol{y}$. It will save the computational cost when the number of test points is large.

We shall introduce a localized PMLS method to further reduce the computational cost. In measure the "closeness" to the given sampling data, we only consider the sampling points centered around the test point with certain distance. Specifically, for a specified threshold q, we define

$$S_k^q(\boldsymbol{y}) = \sum_{j=1}^{m} \lambda_{k,j}^q u_j(\boldsymbol{y} - \boldsymbol{t}_k),$$

where $\boldsymbol{\lambda}_k^q = (\lambda_{k,j}^q : 1 \le j \le m)$ is given by

$$\boldsymbol{\lambda}_k^q := \operatorname{argmin}_{\boldsymbol{a} \in \mathbb{R}^m} \sum_{\|\boldsymbol{x}_i - \boldsymbol{t}_k\| \le q} \left[f(\boldsymbol{x}_i) - \sum_{j=1}^{m} a_j u_j(\boldsymbol{x}_i - \boldsymbol{t}_k) \right]^2 w(\|\boldsymbol{x}_i - \boldsymbol{t}_k\|).$$

The approximation function is defined as

$$S^q(\boldsymbol{y}) = \sum_{k=1}^{n} S_k^q(\boldsymbol{y}) \chi_{\Omega_k}(\boldsymbol{y}), \quad \boldsymbol{y} \in \Omega. \tag{2.8}$$

3. Optimal design

We will show in this section that the PMLS method is an optimal design with certain localized information. To this end, we shall next provide another point of view of the standard MLS method as a minimal norm interpolation problem. We point out that we are not necessarily interpolating the function values but the vector $\boldsymbol{L}_{\boldsymbol{y}}(f)$ defined on (4.8).

We remark that the vector $\boldsymbol{L}_{\boldsymbol{y}}(f)$ could be viewed as a *localized information* of the unknown function f. It is a weighted inner product with more weight on the function values of the sampling points closer to $\boldsymbol{y}$. In other words, when approximating the function value at a given point $\boldsymbol{y}$, we will use the function values of the sampling points around $\boldsymbol{y}$ and impose more weight on the sampling points that are closer to $\boldsymbol{y}$.

For the MLS approximation function $T_{\boldsymbol{y}}$, it is direct to observe that $\boldsymbol{L}_{\boldsymbol{y}}(T_{\boldsymbol{y}}) = \boldsymbol{L}_{\boldsymbol{y}}(f)$. We next show that it is the mini-norm "interpolation" of $\boldsymbol{L}_{\boldsymbol{y}}(f)$ with respect to the norm $\|\cdot\|_{w_{\boldsymbol{y}}}$.

Proposition 3.1. *For any $\boldsymbol{y} \in \Omega$, $v \in \mathcal{U}$, and any function g on Ω with $\boldsymbol{L}_{\boldsymbol{y}}(g) = \boldsymbol{L}_{\boldsymbol{y}}(f)$, there holds*

$$\langle T_{\boldsymbol{y}} - g, v\rangle_{w_{\boldsymbol{y}}} = 0 \quad \text{and} \quad \|T_{\boldsymbol{y}}\|_{w_{\boldsymbol{y}}} \le \|g\|_{w_{\boldsymbol{y}}}.$$

Proof. Assume g is a function on Ω and $\boldsymbol{L}_{\boldsymbol{y}}(g) = \boldsymbol{L}_{\boldsymbol{y}}(f)$. It is direct to see that $\boldsymbol{L}_{\boldsymbol{y}}(T_{\boldsymbol{y}}) = \boldsymbol{L}_{\boldsymbol{y}}(f)$. It implies that $\boldsymbol{L}_{\boldsymbol{y}}(T_{\boldsymbol{y}} - g) = 0$. That is, $\langle T_{\boldsymbol{y}} - g, u_j(\cdot - \boldsymbol{y})\rangle_{w_{\boldsymbol{y}}} = 0$ for all $j = 1, \dots, m$, which implies that $\langle T_{\boldsymbol{y}} - g, v\rangle = 0$ for any $v \in \mathcal{U}$. In particular, $\langle T_{\boldsymbol{y}} - g, T_{\boldsymbol{y}}\rangle_{w_{\boldsymbol{y}}} = 0$ since $T_{\boldsymbol{y}} \in \mathcal{U}$. It follows that $\langle T_{\boldsymbol{y}}, T_{\boldsymbol{y}}\rangle_{w_{\boldsymbol{y}}} = \langle g, T_{\boldsymbol{y}}\rangle_{w_{\boldsymbol{y}}}$. By the Cauchy–Schwarz inequality, we have $\langle g, T_{\boldsymbol{y}}\rangle_{w_{\boldsymbol{y}}} \le \|g\|_{w_{\boldsymbol{y}}}\|T_{\boldsymbol{y}}\|_{w_{\boldsymbol{y}}}$, which implies the desired result. □

We then define the localized weighted inner product for any $\boldsymbol{x} \in \Omega$ and functions g_1, g_2 on Ω

$$\langle g_1,\, g_2\rangle_{w^q_{\boldsymbol{x}}} = \sum_{\|\boldsymbol{t}_j - \boldsymbol{x}\| \le q} g_1(\boldsymbol{t}_j) g_2(\boldsymbol{t}_j) w(\|\boldsymbol{t}_j - \boldsymbol{x}\|),$$

where q is a constant to be specified later. For any $\boldsymbol{y} \in \Omega$, we define the corresponding localized information

$$\boldsymbol{L}^q_{\boldsymbol{y}}(f) = \left[\langle f, u_j(\cdot - \boldsymbol{y})\rangle_{w^q_{\boldsymbol{y}}}\right]_{j=1}^m.$$

We have a similar result for the PMLS method.

Corollary 3.2. *For any $1 \le k \le n$, $v \in \mathcal{U}$, and any function g on Ω with $\boldsymbol{L}^q_{\boldsymbol{t}_k}(g) = \boldsymbol{L}^q_{\boldsymbol{t}_k}(f)$, there holds*

$$\langle S^q_k - g, v\rangle_{w^q_{\boldsymbol{t}_k}} = 0 \quad \text{and} \quad \|S^q_k\|_{w^q_{\boldsymbol{t}_k}} \le \|g\|_{w^q_{\boldsymbol{t}_k}}.$$

Proof. It follows in a similar way as the proof of Proposition 3.1. □

We shall show that the PWLS method defined in (2.8) is an optimal design with the given localized sampling information $\boldsymbol{L}^q_{\boldsymbol{t}_k}(f)$. To this end, we will define the norm of measuring the approximation error. Let $\tau_{\mathbf{x}} f := f(\cdot - \mathbf{x})$, $\mathbf{x} \in \mathbb{R}^d$. Suppose that $\mathcal{U} \subseteq \mathcal{G}$, where $\mathcal{G}$ is a normed linear space of functions on $\mathbb{R}^d$ such that for any $f \in \mathcal{G}$ and $\boldsymbol{x} \in \mathbb{R}^d$, we have $\tau_{\boldsymbol{x}} f \in \mathcal{G}$ and

$$\|\tau_{\mathbf{x}} f\|_{\mathcal{G}} = \|f\|_{\mathcal{G}}. \tag{3.1}$$

For example, $\mathcal{G} = L^p(\mathbb{R})$ satisfies this assumption for any $p \in [1, \infty]$. Moreover, any translation-invariant space would also satisfy this assumption. We next define the best approximation error with respect to the norm $\|\cdot\|_{\mathcal{G}}$. Assume $A : \mathbb{R}^m \to \mathcal{U}$ is a linear algorithm of approximating the unknown function f from its localized information $\boldsymbol{L}^q_{\boldsymbol{t}_k}(f)$. We define its approximation error as

$$E_q(A) := \max_{1 \le k \le n} E_{q,k}(A),$$

where

$$E_{q,k}(A) = \sup\left\{\left\|\left[f - A(\boldsymbol{L}^q_{\boldsymbol{t}_k}(f))\right]\chi_{\Omega_k}\right\|_{\mathcal{G}} : \|f\|_{w^q_{\boldsymbol{t}_k}} \le 1\right\}. \tag{3.2}$$

The best approximation error is defined to be the smallest approximation error for all possible linear approximation algorithms.

$$\sigma_q := \inf\{E_q(A) : A \in L(\mathbb{R}^m, \mathcal{U})\}, \tag{3.3}$$

where $L(\mathbb{R}^m, \mathcal{U})$ denotes the set of operators from $\mathbb{R}^m$ to $\mathcal{U}$. We will show that the PMLS approximation S_q defined in (2.8) is an optimal design by showing that $E_q(P) = \sigma_q$, where P denotes the mapping from the localized information $\boldsymbol{l}^q_{\boldsymbol{t}_k}(f)$ to the PMLS approximation function S^q. To this end, we introduce the following constant that plays a central role in the proof:

$$\rho_q := \max_{1 \le k \le n} \rho_{q,k}, \tag{3.4}$$

where

$$\rho_{q,k} = \sup\{\|f\chi_{\Omega_k}\|_{\mathcal{G}} : \|f\|_{w^q_{\boldsymbol{t}_k}} \le 1, \boldsymbol{L}^q_{\boldsymbol{t}_k}(f) = 0\}. \tag{3.5}$$

Theorem 3.1. *Suppose q is a positive constant and P is the PWLS operator. It holds that*

$$E_q(P) = \sigma_q = \rho_q.$$

Proof. We will prove the desired equalities by showing that $E_q(P) \ge \sigma_q$, $\sigma_q \ge \rho_q$, and $\rho_q \ge E_q(S^q)$.

We first show that $E_q(P) \ge \sigma_q$. It follows immediately from the definition of σ_q in (3.3) and the fact that $P \in L(\mathbb{R}^m, \mathcal{U})$.

We next show that $\sigma_q \ge \rho_q$. By the definition of σ_q in (3.3), it is equivalent to show $E_q(A) \ge \rho_q$ for any $A \in L(\mathbb{R}^m, \mathcal{U})$. Suppose $A \in L(\mathbb{R}^m, \mathcal{U})$. We will show it by proving $E_{q,k}(A) \ge \rho_{q,k}$ for any $1 \le k \le n$. Assume $g \in \mathcal{U}$ satisfies $\|g\|_{w^q_{\boldsymbol{t}_k}} \le 1$ and $\boldsymbol{L}^q_{\boldsymbol{t}_k}(g) = 0$. It follows from the definition of $E_{q,k}(A)$ in (3.2) that

$$E_{q,k}(A) \ge \left\| \left[g - A(\boldsymbol{L}^q_{\boldsymbol{t}_k}(g)) \right] \chi_{\Omega_k} \right\|_{\mathcal{G}} = \left\| [g - A(0)]\, \chi_{\Omega_k} \right\|_{\mathcal{G}}$$

Note that $-g$ also satisfies $\|-g\|_{w^q_{\boldsymbol{t}_k}} \le 1$ and $\boldsymbol{L}^q_{\boldsymbol{t}_k}(-g) = 0$, which implies

$$E_{q,k}(A) \ge \left\| [-g - A(0)]\, \chi_{\Omega_k} \right\|_{\mathcal{G}} = \left\| [g + A(0)]\, \chi_{\Omega_k} \right\|_{\mathcal{G}}$$

It follows from applying triangle inequality with the above two inequalities that

$$E_{q,k}(A) \ge \left\| g\chi_{\Omega_k} \right\|_{\mathcal{G}}.$$

Since it holds for arbitrary g satisfying $\|g\|_{w^q_{\boldsymbol{t}_k}} \le 1$ and $\boldsymbol{L}^q_{\boldsymbol{t}_k}(g) = 0$, it follows from the definition of $\rho_{q,k}$ in (3.5) that

$$E_{q,k}(A) \ge \rho_{q,k}.$$

It remains to prove $\rho_q \ge E_q(P)$. It is enough to show $\rho_{q,k} \ge E_{q,k}(P)$ for all $1 \le k \le n$. Assume $1 \le k \le n$ is arbitrary but given and $\|f\|_{w^q_{\boldsymbol{t}_k}} \le 1$. Since P is the PMLS approximation, it follows from Corollary 3.2 that

$$\langle P(\boldsymbol{L}^q_{\boldsymbol{t}_k}(f)) - f, P(\boldsymbol{L}^q_{\boldsymbol{t}_k}(f)) \rangle_{w^q_{\boldsymbol{t}_k}} = \langle S^q_k - f, S^q_k \rangle_{w^q_{\boldsymbol{t}_k}} = 0,$$

which implies

$$\|f - P(\boldsymbol{L}^q_{\boldsymbol{t}_k}(f))\|^2_{w^q_{\boldsymbol{t}_k}} = \|f\|^2_{w^q_{\boldsymbol{t}_k}} - \|P(\boldsymbol{L}^q_{\boldsymbol{t}_k}(f))\|^2_{w^q_{\boldsymbol{t}_k}} \le \|f\|_{w^q_{\boldsymbol{t}_k}} \le 1.$$

Note that $\boldsymbol{L}^q_{\boldsymbol{t}_k}(f) = \boldsymbol{L}^q_{\boldsymbol{t}_k}(P(\boldsymbol{L}^q_{\boldsymbol{t}_k}(f)))$. It follows from the definition of $\rho_{q,k}$ in (3.5) that

$$\rho_{q,k} \ge \|(f - P(\boldsymbol{L}^q_{\boldsymbol{t}_k}(f)))\chi_{\Omega_k}\|^2_{\mathcal{G}}.$$

Since it holds for arbitrary f satisfying $\|f\chi_{\Omega_k}\|_{w^q_{\boldsymbol{t}_k}} \le 1$, we have $\rho_{q,k} \ge E_{q,k}(P)$, which finishes the proof. □

4. Numerical experiments

We will test the proposed PMLS method with some numerical experiments in this section. In particularly, we will test it in two kinds of problems: scattered data approximation of seven benchmark test functions and numerical solutions of time-dependent PDEs and system of PDEs.

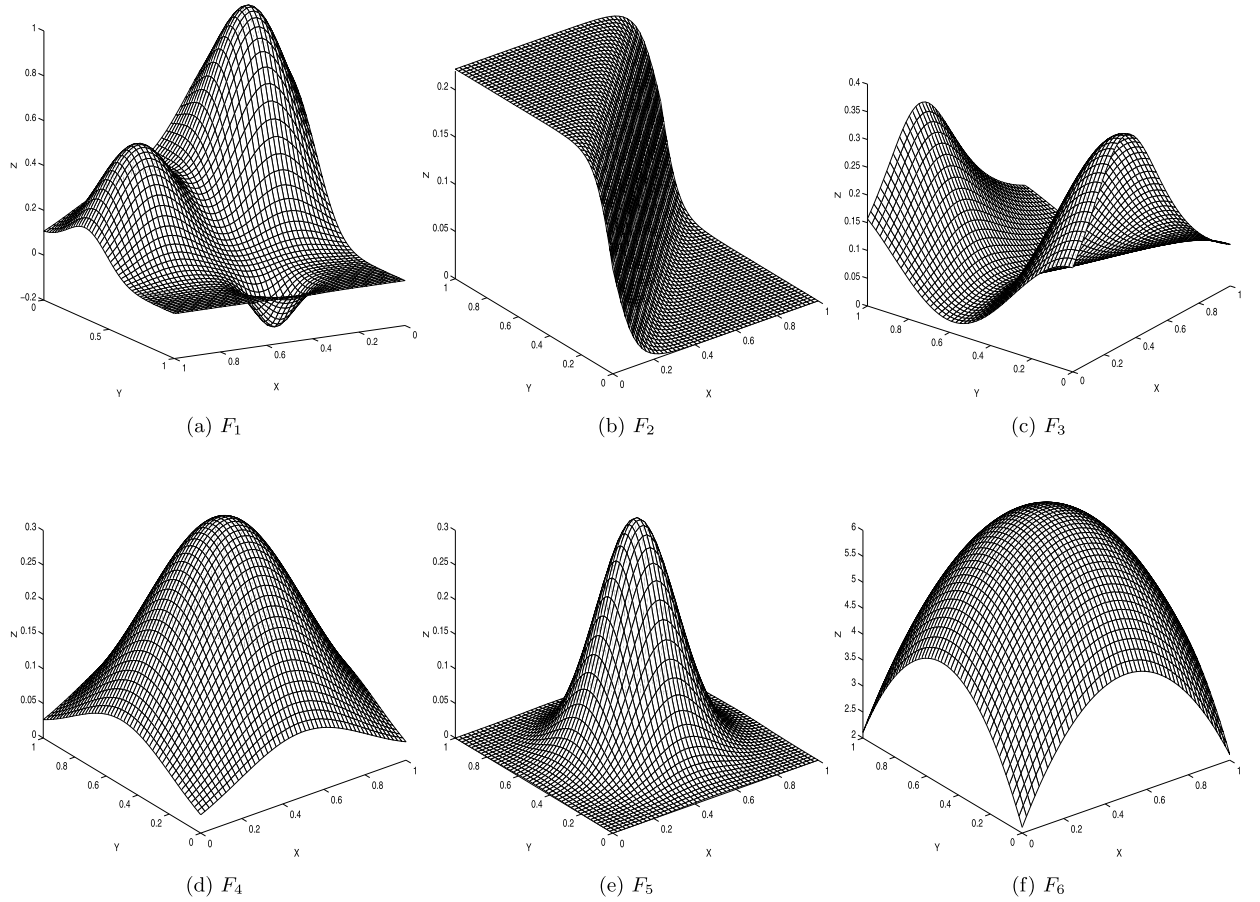
4.1. Scattered data approximation

We shall present numerical experiments for some scattered data approximation problems in this subsection. In particular, we shall compare the proposed PMLS method with the standard MLS method in terms of accuracy and efficiency.

Example 1. We will consider the following Franke's six benchmark test functions [15] on the unit square $[0,1] \times [0,1]$:

(1) $F_1(x,y) = \frac{3}{4}\exp\left(-\frac{1}{4}\left((9x-2)^2 + (9y-2)^2\right)\right) + \frac{3}{4}\exp\left(-\frac{1}{49}(9x+1)^2 - \frac{1}{10}(9y+1)^2\right)$
$+ \frac{1}{2}\exp\left(-\frac{1}{4}(9x-7)^2 - \frac{1}{4}(9y-3)^2\right) - \frac{1}{5}\exp\left(-(9x-4)^2 - (9y-7)^2\right).$

(2) $F_2(x,y) = \frac{1}{9}(\tanh(9y-9x)+1).$



(a) F_1 (b) F_2 (c) F_3

(d) F_4 (e) F_5 (f) F_6

Fig. 1. Franke's six benchmark test functions [15].

(3) $F_3(x, y) = \frac{1.25+\cos(5.4y)}{6[1+(3x-1)^2]}$.

(4) $F_4(x, y) = \frac{1}{3}\exp\left[-\frac{81}{16}\left(\left(x-\frac{1}{2}\right)^2+\left(y-\frac{1}{2}\right)^2\right)\right]$.

(5) $F_5(x, y) = \frac{1}{3}\exp\left[-\frac{81}{4}\left(\left(x-\frac{1}{2}\right)^2+\left(y-\frac{1}{2}\right)^2\right)\right]$.

(6) $F_6(x, y) = \frac{1}{9}\left[64-81\left(\left(x-\frac{1}{2}\right)^2+\left(y-\frac{1}{2}\right)^2\right)\right]-\frac{1}{2}$.

We also display these functions in Fig. 1.

We will decompose the domain $\Omega = [0, 1]^2$ into n_s^2 squares $\{\Omega_k : 1 \le k \le n = n_s^2\}$ of the same size with n_s ranging from $11, 21, \ldots, 101$. The anchor points $\{\boldsymbol{t}_k\}$ are chosen to be the centers of Ω_k correspondingly. Fig. 2 shows an example of the evenly distributed sampling points and the anchor points in the square domain.

For both the MLS and PMLS methods, we aim to approximate the function values of the test function $f(\boldsymbol{x})$ at test points $\boldsymbol{x}_j^e$, $j = 1, 2, \cdots, 101^2$ that are evenly distributed in the unit square. We will compare both methods in terms of the following three measures of accuracy:

$$\epsilon_{rms} = \frac{1}{\sqrt{n_t}}\|f(\boldsymbol{x}^e) - \hat{f}(\boldsymbol{x}^e)\|_2, \qquad \epsilon_r = \frac{\|f(\boldsymbol{x}^e) - \hat{f}(\boldsymbol{x}^e)\|_2}{\|f(\boldsymbol{x}^e)\|_2}, \qquad \epsilon_\infty = \|f(\boldsymbol{x}^e) - \hat{f}(\boldsymbol{x}^e)\|_\infty, \tag{4.1}$$

where $\hat{f}(\boldsymbol{x}^e)$ are the approximated values at $\boldsymbol{x}^e$, n_t is the number of test points need to be evaluated by MLS method and PMLS method.

We will use the same RBF weight function $w(x) = (1-\frac{x}{r_w})_+^2$ for both the MLS and the PMLS, where r_w is a constant to be specified later.

The efficiency of PMLS method can be illustrated by comparing its asymptotic computational complexity with MLS method. We need to calculate $\boldsymbol{c}$ in Eq. (2.4) for n_t times using MLS method. For each time, we use k–d tree technique to

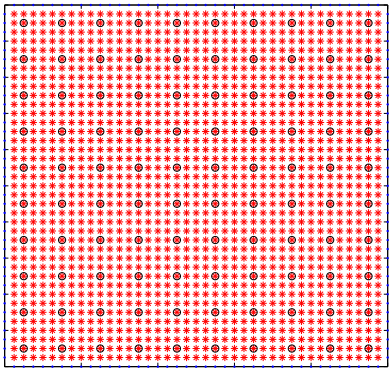


Fig. 2. Nodes distribution, where blue dots represent the boundary points, red stars represent sampling points, and black circles represent the anchor points.

Table 1
Comparison of MLS and PMLS with polynomials basis, $n_s = 11$, $r_w = 0.1$.

Function	ϵ_{rms}		ϵ_r		ϵ_∞		CPU(s)	
	PMLS	MLS	PMLS	MLS	PMLS	MLS	PMLS	MLS
F1	6.61E−5	1.46E−5	1.62E−4	3.59E−5	6.62E−4	1.79E−4	0.17	3.30
F2	9.59E−5	2.34E−5	6.44E−4	1.57E−4	5.77E−4	9.42E−5	0.19	3.24
F3	1.73E−6	2.54E−7	1.18E−5	1.73E−6	1.18E−5	2.67E−6	0.16	3.23
F4	3.95E−7	4.30E−8	2.20E−6	2.39E−7	1.29E−6	5.55E−7	0.15	3.26
F5	7.18E−6	8.74E−7	7.81E−5	9.50E−6	3.74E−5	5.75E−6	0.17	3.25
F6	4.24E−8	1.31E−8	8.20E−9	2.54E−9	4.65E−7	2.60E−7	0.14	3.24

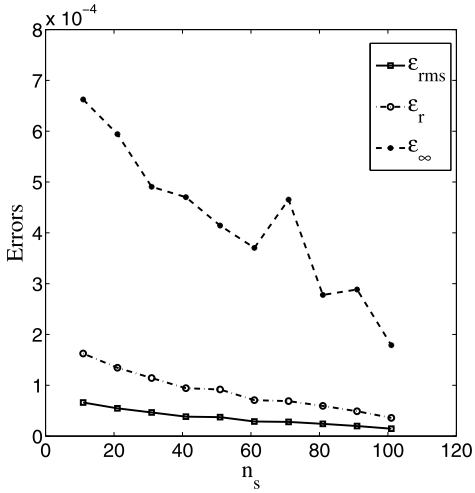


Fig. 3. Errors of the PMLS method for F_1 using polynomial basis function with n_s ranging from $11, 21, \ldots, 101$.

search the support nodes in the support domain with radius of r_w. Suppose the number of support nodes in a support domain is n_w, the asymptotic computational complexity for calculating coefficient $\boldsymbol{c}$ in (2.4) is $\mathcal{O}(\log N + n_w m^2)$ [33,34]. Then the computational complexity for all coefficients is $\mathcal{O}(n_t(\log N + n_w m^2))$ using MLS method. Whereas, the computational complexity for computing all the coefficients $\boldsymbol{\lambda}_k$ in (2.7) using PMLS is $\mathcal{O}(n_s^2(\log N + n_w m^2))$. Since the other operations are almost the same for both methods except decomposing subdomain needed by PMLS, the difference of the running time between two methods are mainly due to the computational complexity of the coefficients calculation. Note that $n_s^2 \ll n_t$. Thus, the PMLS method is more efficient than the MLS method.

We first consider the space $\mathcal{U}$ to be the polynomials of degree no more than 4. For $n_s = 11$, $r_w = 0.1$ and $N = 51^2$ evenly distributed sampling points, we present the comparison of the MLS and the PMLS in terms of the three measures of accuracy defined above and the running time in Table 1. Furthermore, for the test function F_1, we increase the number of anchor points (subdomains) n_s from 11 to $21, 31, \ldots 101$ using the same polynomial basis and display the convergence of the approximation errors of the PMLS method in Fig. 3.

Table 2
Comparison of MLS and PMLS with polyharmonic splines basis, $n_s = 11$, $r_w = 0.15$, and $\delta = 0.2$.

Function	ϵ_{rms}		ϵ_r		ϵ_∞		CPU(s)	
	PMLS	MLS	PMLS	MLS	PMLS	MLS	PMLS	MLS
F1	7.44E−5	4.03E−5	1.83E−4	9.92E−5	8.55E−4	4.54E−4	0.67	13.95
F2	4.96E−5	2.41E−5	3.33E−4	1.62E−4	6.23E−4	9.30E−4	0.74	14.20
F3	2.25E−6	1.36E−6	1.53E−5	9.21E−6	1.98E−6	5.77E−5	0.70	13.54
F4	4.46E−7	4.69E−7	2.48E−6	2.61E−6	3.32E−6	3.18E−5	0.76	13.57
F5	8.05E−6	4.71E−6	8.75E−5	5.12E−5	4.84E−5	1.09E−4	0.76	14.37
F6	5.29E−8	5.72E−7	1.02E−8	1.11E−7	6.08E−7	4.32E−5	0.70	13.59

Table 3
Comparison of MLS and PMLS with polyharmonic splines basis, $n_s = 11$, $r_w = 0.15$, and $\delta = 0.8$.

Function	ϵ_{rms}		ϵ_r		ϵ_∞		CPU(s)	
	PMLS	MLS	PMLS	MLS	PMLS	MLS	PMLS	MLS
F1	8.11E−5	5.97E−5	2.00E−4	1.47E−4	8.45E−4	8.07E−4	0.72	14.34
F2	9.45E−5	4.75E−5	6.34E−4	3.19E−4	8.99E−4	5.93E−4	0.75	13.71
F3	2.34E−6	1.73E−6	1.59E−5	1.18E−5	1.58E−5	3.29E−5	0.73	13.68
F4	5.49E−7	4.80E−7	3.06E−6	2.67E−6	7.01E−6	1.85E−5	0.73	13.71
F5	9.50E−6	7.74E−6	1.03E−4	8.42E−5	9.98E−5	6.70E−5	0.73	13.94
F6	4.40E−8	2.02E−7	8.51E−9	3.91E−8	7.07E−7	1.90E−5	0.67	13.95

We next consider the polyharmonic splines [10,26,36] of degree no more than 4. That is, $\mathcal{U}$ is spanned by the following basis with $n_r = 9$ and $\alpha = 4$:

$$r_1^{2\alpha}\ln(r_1), r_2^{2\alpha}ln(r_2), \cdots, r_{n_r}^{2\alpha}\ln(r_{n_r}), 1, x, y, x^2, xy, y^2, \cdots, x^\alpha, x^{\alpha-1}y, \cdots, y^\alpha,$$

where $r_i = \|\boldsymbol{x} - \boldsymbol{x}_i^{\boldsymbol{t}_k}\|$, and $\boldsymbol{x}_i^{\boldsymbol{t}_k}$, $i = 1, 2, \cdots, n_r$ are the nearest n_r sampling points of the anchor point $\boldsymbol{t}_k$ in the subset Ω_k that the point $\boldsymbol{x}$ belongs to.

To verify the stability of PMLS, the sampling data points $\{\boldsymbol{x}_j = (\tilde{x}_j, \tilde{y}_j) : 1 \le j \le N\}$ are chosen to be $N = 51^2$ randomly distributed points in the following way:

$$\tilde{x}_j = x_j + x_j^{rand} D_x \delta, \quad \tilde{y}_j = y_j + y_j^{rand} D_y \delta, \tag{4.2}$$

where (x_j, y_j), $i = 1, 2, \cdots, 51^2$ are the evenly distributed points in the unit square $[0, 1]^2$, $D_x = D_y = 1/50$ are the distance between two adjacent points in x and y directions, x_i^{rand} and y_i^{rand} are random numbers in $[0, 1]$, and δ is a constant that represents the degree of randomness. We will present the numerical experiments for a few different δ's later.

We set $n_s = 11$, $r_w = 0.15$, and $\delta = 0.2$. The numerical comparison of the MLS and the PMLS method is presented in Table 2. For the same polyharmonic splines basis, we next increase the degree of randomness $\delta = 0.8$ and show the numeric results in Table 3.

From Tables 1, 2 and 3, we observe that the PMLS method has a much less computational time while maintaining a similar level of accuracy in all of such cases. To illustrate, we compare the CPU times of MLS and PMLS with polyharmonic splines basis in Tables 2 and 3. The CPU time for calculating coefficients $\boldsymbol{c}$ in (2.4) is around 13.30 seconds which accounts for most of the running time listed in the Tables 2 and 3, and for $\boldsymbol{\lambda}_k$ in (2.7) is about 0.16 seconds. The ratio of CUP time of MLS and PMLS for calculating the coefficients, which is 83, is nearly equal to the ratio of their asymptotic computational complexity, which is $n_t/n_s^2 \approx 84$.

We also observe from Fig. 3 that the approximation error of the PMLS method will converge as the number of anchor points increases. Fig. 4 shows the ϵ_{rms} ratio of PMLS method and MLS method related to the ratio of the number of anchor points and test points. We can clearly see from this figure that the error of PMLS method converges to that of the MLS method as the number of anchor points approaches to the number of test points.

In many practical applications, the mesh plays a important role in determining the solution and many solvers loose their accuracy if the mesh is poorly constructed. An advantage of the least squares method is that it does not require grids to approximate the function. There are a number of important factors that effect the performance of PMLS, including the number of the scattered data, size of the support domain, the degree of the polynomial basis, choice of replacing polynomial basis by radial basis functions (RBFs) and such [14,19,22]. In this section, we introduced PMLS for solving fitting problems in terms of accuracy and efficiency.

4.2. Solving time-dependent PDEs using piecewise MLS

In this section, we will further validate the proposed PMLS method on parabolic PDEs or system of parabolic PDEs. To solve the heat conduction problem

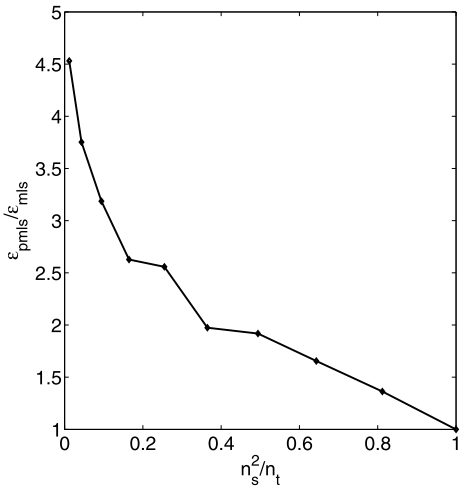


Fig. 4. The error ratio of PMLS method and MLS method for F_1 versus the ratio of the number of anchor points and test points using polynomial basis function with n_s ranging from $11, 21, \ldots, 101$.

$$\frac{\partial f}{\partial t}(\boldsymbol{x}, t) = \nabla^2 f(\boldsymbol{x}, t), \quad \boldsymbol{x} \in \Omega, t > 0, \tag{4.3}$$

with initial condition

$$f(\boldsymbol{x}, 0) = f^0(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega \cup \partial\Omega \tag{4.4}$$

and boundary condition

$$Bf(\boldsymbol{x}, t) = g(\boldsymbol{x}, t), \quad \boldsymbol{x} \in \partial\Omega, t > 0,$$

where B is a boundary operator.

MLS and PMLS are essentially methods for data fitting. In meshless methods based on MLS approximation for solving PDEs [3,5], MLS method is used to construct shape functions. For the meshless weak-form methods for solving PDEs, imposing the essential boundary conditions is a kind of tough problem. Some techniques are proposed for imposition of the essential boundary conditions [5,17,18,25,39]. For some strong-form meshless methods based on radial basis functions [9, 38], the essential boundary conditions are easy to be enforced. If the shape functions are constructed by MLS method, which do not have the Kronecker delta function property, we can use discretization method to enforce the essential boundary conditions while assembling global matrices [24]. In this paper, we use strong-form method combined with forward difference scheme for approximating $\partial f/\partial t$, which is more straightforward, so that we can evaluate and compare the PMLS and the MLS methods for solving time-dependent PDEs more directly. Using forward difference to approximate $\partial f/\partial t$ at every time step t^h, $h = 1, 2, \cdots$, we have:

$$\frac{f^h - f^{h-1}}{\Delta t} = \nabla^2 f^{h-1}, \quad h = 1, 2, \cdots$$

where Δt is the size of the time step and $f^h = f(\boldsymbol{x}, t^h)$. Thus,

$$f^h = f^{h-1} + \Delta t \nabla^2 f^{h-1}, \quad h = 1, 2, \cdots \tag{4.5}$$

The idea of solving this heat conduction problem using PMLS is to approximate $\nabla^2 f^{h-1}$ by

$$\nabla^2 f^{h-1} = \nabla^2 \boldsymbol{u}^T(\boldsymbol{x} - \boldsymbol{t}_k)\boldsymbol{\lambda}_k^{q,h-1}, \quad \boldsymbol{x} \in \Omega_k, k = 1, 2, \cdots, n_s^2, \tag{4.6}$$

where $\boldsymbol{u} = (u_j : 1 \le j \le m)$ and

$$\boldsymbol{\lambda}_k^{q,h-1} := \operatorname{argmin}_{\boldsymbol{a} \in \mathbb{R}^m} \sum_{\|\boldsymbol{x}_i - \boldsymbol{t}_k\| \le q} \left[f^{h-1}(\boldsymbol{x}_i) - \sum_{j=1}^{m} a_j u_j(\boldsymbol{x}_i - \boldsymbol{t}_k) \right]^2 w(\|\boldsymbol{x}_i - \boldsymbol{t}_k\|). \tag{4.7}$$

Since it follows from a direct computation that

$$\boldsymbol{\lambda}_k^{q,h-1} = [\mathsf{G}(\boldsymbol{t}_k)]^{-1} \boldsymbol{L}_{\boldsymbol{t}_k}^q (f^{h-1}),$$

Table 4
Errors of numerical methods at different time in Example 2 using polynomial basis function, where $\Delta t = 0.0001$, $N = 1681$, $n_i = 1521$, $n_s = 20$, $\alpha = 5$, $r_w = 0.15$.

t	ϵ_{rms}		ϵ_r		ϵ_∞		CPU(s)	
	PMLS	MLS	PMLS	MLS	PMLS	MLS	PMLS	MLS
10^{-3}	5.54E−3	5.52E−3	6.01E−3	5.99E−3	3.33E−2	2.26E−2	163.70	603.81
10^{-2}	1.52E−3	1.29E−3	2.18E−3	1.85E−3	1.34E−2	2.65E−3	166.43	606.47
10^{-1}	9.21E−4	2.30E−4	7.97E−3	2.00E−3	4.59E−3	4.47E−4	170.11	635.97

Table 5
Errors of numerical methods at different time in Example 2 using polyharmonic splines basis function, where $\Delta t = 0.0001$, $N = 2601$, $n_i = 2401$, $n_s = 10$, $n_r = 9$, $\alpha = 4$, $r_w = 0.1$.

t	ϵ_{rms}		ϵ_r		ϵ_∞		CPU(s)	
	PMLS	MLS	PMLS	MLS	PMLS	MLS	PMLS	MLS
10^{-3}	6.50E−3	4.84E−3	7.09E−3	5.27E−3	5.38E−2	2.41E−2	36.04	628.10
10^{-2}	1.38E−3	9.74E−4	1.99E−3	1.40E−3	8.45E−3	2.26E−3	40.89	635.65
10^{-1}	1.97E−4	2.23E−4	1.71E−3	1.94E−3	5.03E−4	4.78E−4	48.31	640.21

where

$$\boldsymbol{L}^q_{\boldsymbol{t}_k}(f^{h-1}) = \left[\langle f^{h-1}(\boldsymbol{x}_i), u_j(\cdot - \boldsymbol{t}_k) \rangle_{w^q_{\boldsymbol{t}_k}} \right]_{j=1}^{m}, \quad \|\boldsymbol{x}_i - \boldsymbol{t}_k\| \le q, \tag{4.8}$$

equation (4.6) can be written as

$$\nabla^2 f^{h-1} = \nabla^2 \boldsymbol{\Phi}^T(\boldsymbol{x} - \boldsymbol{t}_k) \boldsymbol{f}^{h-1}(\boldsymbol{x}_i), \quad \boldsymbol{x} \in \Omega_k, k = 1, 2, \cdots, n_s^2, \quad \|\boldsymbol{x}_i - \boldsymbol{t}_k\| \le q,$$

where $\nabla^2 \boldsymbol{\Phi}^T(\boldsymbol{x} - \boldsymbol{t}_k)$ are the Laplace shape functions. Since $f^0(\boldsymbol{x}_i)$, $i = 1, 2, \cdots, N$ are known, we can solve the problem iteratively. Here, we just need to compute the Laplace shape functions $\nabla^2 \boldsymbol{\Phi}^T(\boldsymbol{x} - \boldsymbol{t}_k)$ once, and store them for being used in each time step for calculating f^h in Eq. (4.5), which is efficient [20]. If we use backward difference scheme to approximate $\partial f / \partial t$, the Eq. (4.5) will become a modified Helmholtz equation. We need to assemble and solve the global matrix to get the f^h. By using above methods, even though smaller time interval is requested, we do not need to assemble and solve the global system. Furthermore, the Dirichlet boundary conditions can be naturally used in the iterations for solving the values at interior points.

Example 2. In this example, we consider the following heat conduction problem in the bounded domain $\Omega \cup \partial\Omega = [-0.5, 0.5]^2$.

$$\frac{\partial f}{\partial t}(x, y, t) = \nabla^2 f(x, y, t), \quad (x, y) \in \Omega, t > 0, \tag{4.9}$$

with initial condition $f(x, y, 0) = 1$, for $(x, y) \in \Omega \cup \partial\Omega$ and Dirichlet boundary condition $f(x, y, t) = 0$, for $(x, y) \in \partial\Omega$, $t > 0$. The analytical solution is given by

$$f(x, y, t) = \frac{16}{\pi^2} f_s(x, t) f_s(y, t)$$

where

$$f_s(\eta, t) = \sum_{i=0}^{\infty} \frac{(-1)^i \exp[-(2i+1)^2 \pi^2 t] \cos[(2i+1)\pi\eta]}{(2i+1)},$$

with $\eta = x$ or y.

The process for solving this kind of heat conduction equations using PMLS has been given in Eq. (4.3)–(4.7). Table 4 shows the errors of numerical results when $t = 10^{-3}$, 10^{-2} and 10^{-1} by using PMLS and MLS. Polynomial of degree $\alpha = 5$ is the basis function. n_i is the number of interior points. The computational complexity of MLS method and PMLS method for calculating Laplace shape functions are $\mathcal{O}(n_i(\log N + n_w m^2))$ and $\mathcal{O}(n_s^2(\log N + n_w m^2))$. Their CUP time are 602 seconds and 106 seconds respectively, which account for majority of the total running time in Table 4. We can see the ratio of CUP time $602/106 \approx 3.7$ for Laplace shape functions computation is consistent with the ratio of asymptotic computational complexity, which is $n_i/n_s^2 \approx 3.8$. The accuracy of the two methods are in the same order of magnitudes.

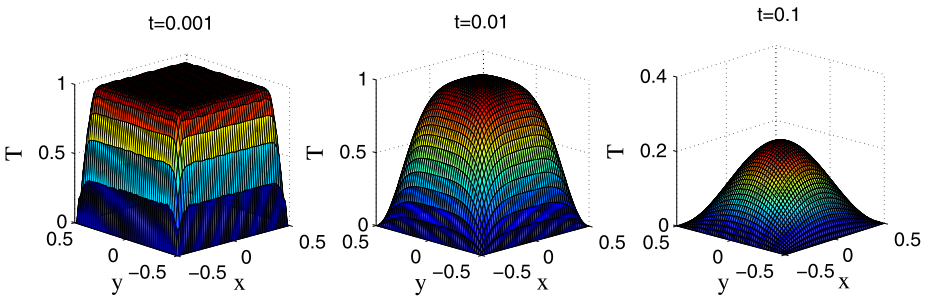


Fig. 5. The numerical solutions of heat conduction problem in Example 2 using PMLS with polyharmonic splines basis function.

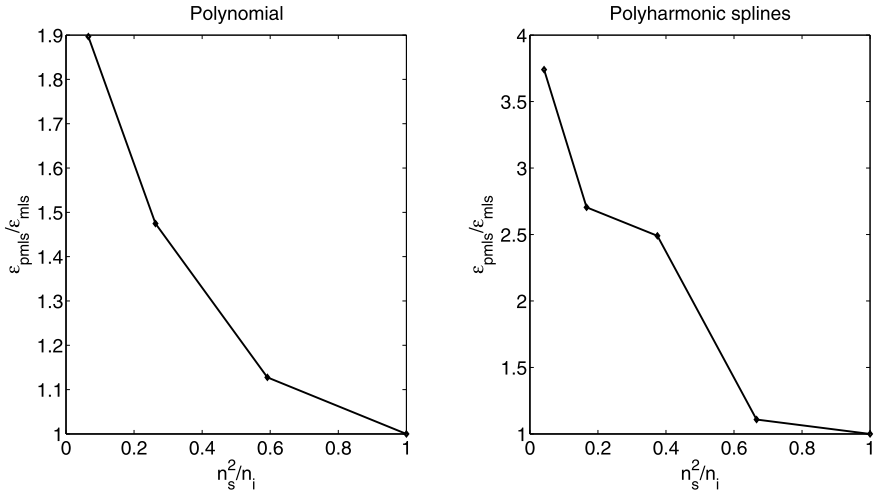


Fig. 6. The error ratio of PMLS method and MLS method versus the ratio of the number of anchor points and interior points based on polynomial basis with $n_s = 10, 20, 30, 39$ (left) and polyharmonic splines basis with $n_s = 10, 20, 30, 40, 49$ (right).

Next, we replace the polynomial basis function by polyharmonic splines. We increase the number of sampling points from 1681 to 2601, decrease n_s from 20 to 10 and r_w from 0.15 to 0.1. As shown in Table 5, although the CPU time of PMLS is much less than MLS, former method is almost as accuracy as the latter one. Fig. 5 shows the profile of the numerical solutions. Therefore, for solving this kind of time-dependent PDEs iteratively, which is time consuming, the efficient and effective PMLS is a better choice than traditional MLS. The convergence of the PMLS method based on polynomial and polyharmonic splines basis functions for solving this heat conduction problem is show in Fig. 6.

The accuracy from PMLS is even better than the results with smaller time interval reported in [37]. The computational costs of PMLS for Laplace shape functions is usually more expensive compared to that of the explicit method in [37], which is $\mathcal{O}(n_i(logN + m^3))$, due to the number of basis m we chose in PMLS is usually higher than $m = 5$ basis in [37] and $n_w >> 5$.

Example 3. We next consider a nonlinear system of reaction-diffusion equations: the Brusselator system in two dimensional space $\Omega = [0, 1]^2$:

$$v_t = \beta(v_{xx} + v_{yy}) + b_2 - (b_1 + 1)v + v^2\omega \tag{4.10}$$

$$\omega_t = \beta(\omega_{xx} + \omega_{yy}) + b_1 v - v^2\omega \tag{4.11}$$

where $v(x, y, t)$ and $w(x, y, t)$ represent concentrations of two chemicals at time t and position (x, y). The diffusion coefficient is $\beta = 0.002$. $b_1 = 0.5$, $b_2 = 1$. The initial conditions are given by

$$v(x, y, 0) = \frac{1}{2}x^2 - \frac{1}{3}x^3, \tag{4.12}$$

$$\omega(x, y, 0) = \frac{1}{2}y^2 - \frac{1}{3}y^3, \tag{4.13}$$

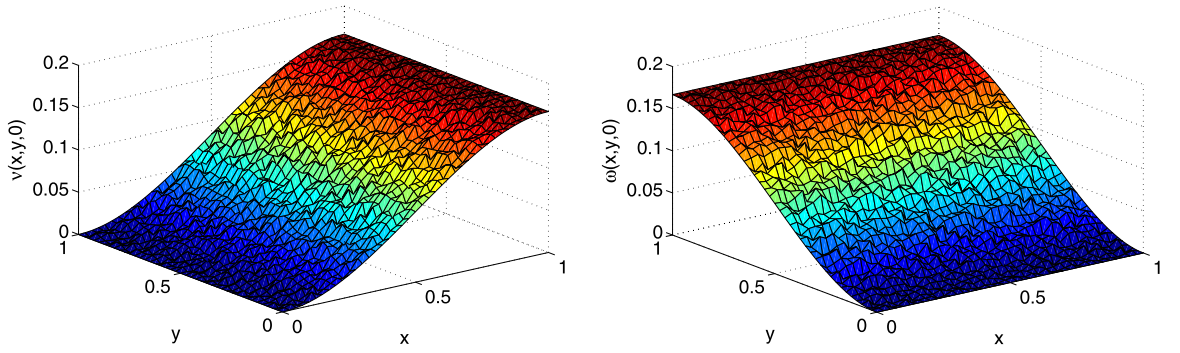


Fig. 7. The initial concentration of ν and ω.

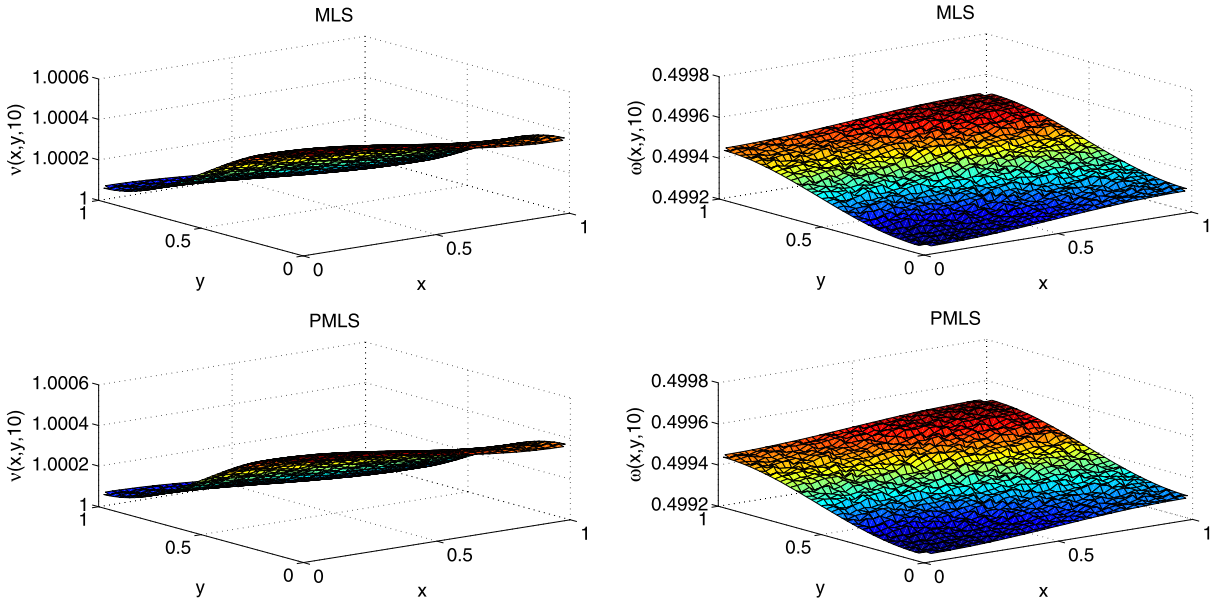


Fig. 8. The solutions of ν and ω using MLS (on the top) and PMLS (on the bottom) respectively at $t = 10$.

and the boundary conditions are zero flux Neumann's boundary conditions shown below

$$\frac{\partial \nu}{\partial \mathbf{n}}(x, y, t) = 0, \tag{4.14}$$

$$\frac{\partial \omega}{\partial \mathbf{n}}(x, y, t) = 0, \tag{4.15}$$

where $(x, y) \in \partial\Omega$, $t > 0$. As we know, for small β, if $1 - b_1 + b_2^2 > 0$, the solution of Brusselator system converges to the equilibrium point $(b_2, b_1/b_2)$ [31,35].

The initial conditions are shown in Fig. 7. The approximation of the ν and ω at $t = 10$ using MLS and PMLS with polynomial basis of order 4 are shown in Fig. 8. 1681 randomly distributed nodes are used with $\delta = 0.8$, $r_w = 0.15$. $n_s = 10$ anchor points are arranged in each axis direction. From Fig. 8, we can see that no matter which method we use, the solution at $t = 10$ at every node is very close to the equilibrium solution. The solutions at $(0.5, 0.5)$ as a function of time are shown in Fig. 9. From Fig. 10, we can see the results ν and ω of PMLS method converge to those of MLS method as n_s^2 approaches to N, and PMLS method has very similar accuracy with MLS method regardless the number of anchor points n_s^2.

Furthermore, the PMLS is more efficient than the MLS. It takes 94 seconds for a simulation to $t = 10$ with time step size 0.01 using PMLS. The computational time using MLS is 516 seconds.

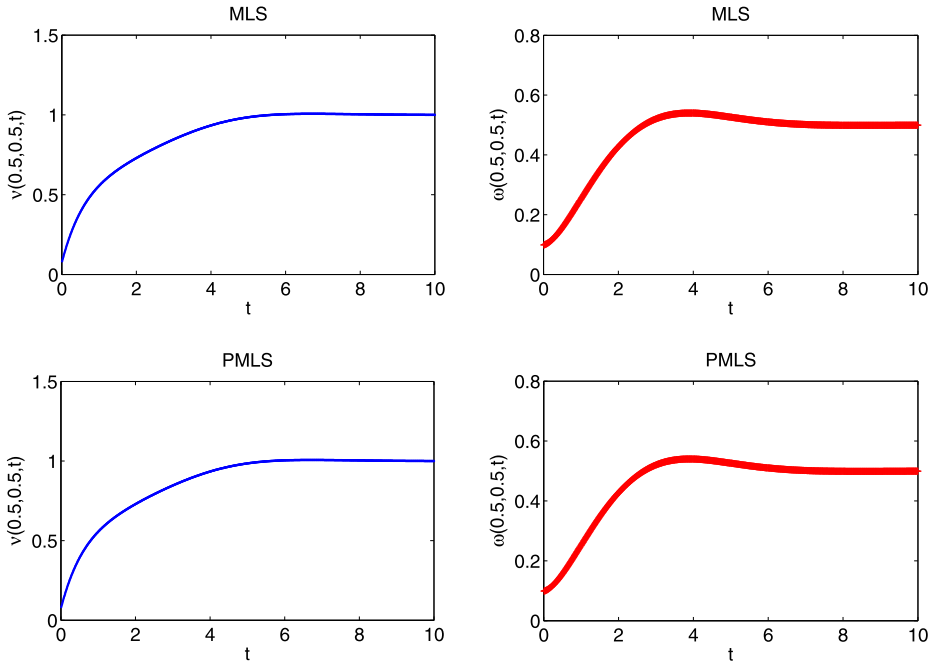


Fig. 9. The solution of v and ω at point $(0.5, 0.5)$ versus t using MLS (on the top) and PMLS (on the bottom) respectively.

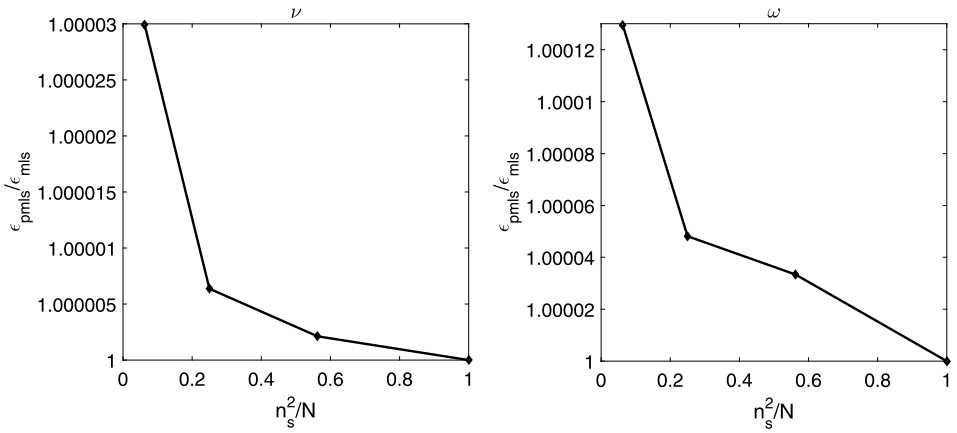


Fig. 10. The error ratio of PMLS method and MLS method versus the ratio of the number of anchor points and test points with $n_s = 10, 20, 30, 41$.

5. Conclusion

We propose a piece-wise MLS method in this paper to reduce the computational cost of the standard MLS method. We prove the proposed PMLS method is an optimal design with certain localized information. We further implement it in some scattered data approximation problems and some time-dependent PDEs. Numerical comparisons with the standard MLS method show that the proposed PMLS method is more efficient while maintaining a similar level of accuracy.

Acknowledgement

The authors wish to thank Dr. Gregor Kosec for useful discussions and suggestions which results in an improved manuscript.

References

[1] William F. Ames, Numerical Methods for Partial Differential Equations, Academic Press, 2014.
[2] Uri M. Ascher, Steven J. Ruuth, Raymond J. Spiteri, Implicit–explicit Runge–Kutta methods for time-dependent partial differential equations, Appl. Numer. Math. 25 (2) (1997) 151–167.

[3] Satya N. Atluri, Tulong Zhu, A new meshless local Petrov–Galerkin (MLPG) approach in computational mechanics, Comput. Mech. 22 (2) (1998) 117–127.
[4] R.K. Beatson, W.A. Light, S. Billings, Fast solution of the radial basis function interpolation equations: domain decomposition methods, SIAM J. Sci. Comput. 22 (5) (2001) 1717–1740.
[5] Ted Belytschko, Yun Yun Lu, Lei Gu, Element-free Galerkin methods, Int. J. Numer. Methods Eng. 37 (2) (1994) 229–256.
[6] Gregory Beylkin, James M. Keiser, Lev Vozovoi, A new class of time discretization schemes for the solution of nonlinear PDEs, J. Comput. Phys. 147 (2) (1998) 362–387.
[7] Piotr Breitkopf, Hakim Naceur, Alain Rassineux, Pierre Villon, Moving least squares response surface approximation: formulation and metal forming applications, Comput. Struct. 83 (17) (2005) 1411–1428.
[8] Martin D. Buhmann, Radial basis functions: theory and implementations, in: Cambridge Monographs on Applied and Computational Mathematics, vol. 12, 2003, pp. 147–165.
[9] C.S. Chen, C.M. Fan, P.H. Wen, The method of approximate particular solutions for solving certain partial differential equations, Numer. Methods Partial Differ. Equ. 28 (2) (2012) 506–522.
[10] C.S. Chen, Y.C. Hon, R.A. Schaback, Scientific Computing with Radial Basis Functions, vol. 39406, Department of Mathematics, University of Southern Mississippi, Hattiesburg, MS, 2005.
[11] G.E. Fasshauer, Meshfree Methods, Handbook of Theoretical and Computational Nanotechnology, American Scientific Publishers, 2005.
[12] Gregory E. Fasshauer, Meshfree Approximation Methods with Matlab, vol. 6, World Scientific, 2007.
[13] Gregory E. Fasshauer, Meshfree Approximation Methods with Matlab, World Scientific Publishing Co., Singapore, 2007.
[14] Gregory E. Fasshauer, Michael J. McCourt, Stable evaluation of gaussian radial basis function interpolants, SIAM J. Sci. Comput. 34 (2) (2012) A737–A762.
[15] Richard Franke, Scattered data interpolation: tests of some methods, Math. Comput. 38 (1982) 181–200.
[16] M.A. Golberg, C.S. Chen, A.S. Muleshkov, The method of fundamental solutions for time-dependent problems, WIT Trans. Model. Simul. 23 (1970).
[17] Norbert Heuer, Thanh Tran, Radial basis functions for the solution of hypersingular operators on open surfaces, Comput. Math. Appl. 63 (11) (2012) 1504–1518.
[18] Norbert Heuer, Thanh Tran, A mixed method for Dirichlet problems with radial basis functions, Comput. Math. Appl. 66 (10) (2013) 2045–2055.
[19] Arta A. Jamshidi, Michael J. Kirby, A radial basis function algorithm with automatic model order determination, SIAM J. Sci. Comput. 37 (3) (2015) A1319–A1341.
[20] G. Kosec, A local numerical solution of a fluid-flow problem on an irregular domain, Adv. Eng. Softw. (2016), http://dx.doi.org/10.1016/j.advengsoft.2016.05.010.
[21] Peter Lancaster, Kes Salkauskas, Surfaces generated by moving least squares methods, Math. Comput. 37 (155) (1981) 141–158.
[22] Leevan Ling, E.J. Kansa, Preconditioning for radial basis functions with domain decomposition methods, Math. Comput. Model. 40 (13) (2004) 1413–1427.
[23] Leevan Ling, Robert Schaback, An improved subspace selection algorithm for meshless collocation methods, Int. J. Numer. Methods Eng. 80 (13) (2009) 1623–1639.
[24] Gui-Rong Liu, Yuan-Tong Gu, An Introduction to Meshfree Methods and Their Programming, Springer Science & Business Media, 2005.
[25] Y.Y. Lu, T. Belytschko, Lu Gu, A new implementation of the element free Galerkin method, Comput. Methods Appl. Mech. Eng. 113 (3–4) (1994) 397–414.
[26] Wally R. Madych, S.A. Nelson, Polyharmonic cardinal splines, J. Approx. Theory 60 (2) (1990) 141–156.
[27] Gia G. Maisuradze, Donald L. Thompson, Albert F. Wagner, Michael Minkoff, Interpolating moving least-squares methods for fitting potential energy surfaces: detailed analysis of one-dimensional applications, J. Chem. Phys. 119 (19) (2003) 10002–10014.
[28] Joseph J. Monaghan, An introduction to SPH, Comput. Phys. Commun. 48 (1) (1988) 89–96.
[29] B. Nayroles, G. Touzot, P. Villon, Generalizing the finite element method: diffuse approximation and diffuse elements, Comput. Mech. 10 (5) (1992) 307–318.
[30] E. Rothe, Zweidimensionale parabolische Randwertaufgaben als Grenzfall eindimensionaler Randwertaufgaben, Math. Ann. 102 (1930) 650–670.
[31] Ahmad Shirzadi, Vladimir Sladek, Jan Sladek, A local integral equation formulation to solve coupled nonlinear reaction–diffusion equations by using moving least square approximation, Eng. Anal. Bound. Elem. 37 (1) (2013) 8–14.
[32] N. Sukumar, R.W. Wright, Overview and construction of meshfree basis functions: from moving least squares to entropy approximants, Int. J. Numer. Methods Eng. 70 (2) (2007) 181–205.
[33] R. Trobec, G. Kosec, M. Šterk, B. Šarler, Comparison of local weak and strong form meshless methods for 2-d diffusion equation, Eng. Anal. Bound. Elem. 36 (3) (2012) 310–321.
[34] Roman Trobec, Marjan Šterk, Borut Robič, Computational complexity and parallelization of the meshless local Petrov–Galerkin method, Comput. Struct. 87 (1) (2009) 81–90.
[35] E.H. Twizell, A.B. Gumel, Q. Cao, A second-order scheme for the “brusselator” reaction–diffusion system, J. Math. Chem. 26 (4) (1999) 297–316.
[36] G. Yao, An improved localized method of approximate particular solutions for solving elliptic PDEs, Comput. Math. Appl. 71 (1) (2016) 171–184.
[37] G. Yao, C.S. Chen, Marina Jelen, B. Sarler, Meshless solutions of temperature fields for use in dendritic growth simulations, in: International Conference on Optimization Using Exergy-Based Methods and Computational Fluid Dynamics, 2009.
[38] Guangming Yao, Joseph Kolibal, C.S. Chen, A localized approach for the method of approximate particular solutions, Comput. Math. Appl. 61 (9) (2011) 2376–2387.
[39] T. Zhu, S.N. Atluri, A modified collocation method and a penalty formulation for enforcing the essential boundary conditions in the element free Galerkin method, Comput. Mech. 21 (3) (1998) 211–222.