Contents lists available at ScienceDirect

Computers and Structures

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

Efficient implementation of Galerkin meshfree methods for large-scale problems with an emphasis on maximum entropy approximants

Christian Peco, Daniel Millán, Adrian Rosolen, Marino Arroyo *

LaCàN, Universitat Politècnica de Catalunya (UPC), Barcelona 08034, Spain

ARTICLE INFO

Article history:
Received 16 July 2014
Accepted 3 December 2014
Available online 10 January 2015

Keywords:
Meshfree methods
Local maximum entropy
Sparse matrix efficient assembly
Matrix structure creation
Optimal memory storage
Code optimization

ABSTRACT

In Galerkin meshfree methods, because of a denser and unstructured connectivity, the creation and assembly of sparse matrices is expensive. Additionally, the cost of computing basis functions can be significant in problems requiring repetitive evaluations. We show that it is possible to overcome these two bottlenecks resorting to simple and effective algorithms. First, we create and fill the matrix by coarse-graining the connectivity between quadrature points and nodes. Second, we store only partial information about the basis functions, striking a balance between storage and computation. We show the performance of these strategies in relevant problems.

© 2014 Elsevier Ltd. All rights reserved.

1. Introduction

Meshfree methods have emerged in recent years as a viable alternative to finite elements in a number of applications, see [1–5] for a detailed review. These methods are based on basis functions that do not rely on a mesh. As a consequence, many of the requirements associated with the quality of the elements in traditional finite element method (FEM) are relaxed or disappear, but this extra flexibility raises new challenges in the numerical implementation [6]. Meshfree methods also present several advantages such as basis functions with high-order continuity, robustness in dramatic grid deformations [7–9], and easier local adaptivity [10,11]. Galerkin meshfree methods require a quadrature mesh to perform numerical integration, commonly requiring a higher number of quadrature points to accurately integrate the weak form due to their nonpolynomial nature and nonelement-wise support [12,13]. Additionally, most of the meshfree methods present an awkward treatment of essential boundary conditions due to non-satisfaction of the Kronecker delta property [3,14].

Since smoothed particle hydrodynamics [15], a variety of techniques have emerged, such as reproducing kernel particle method [16], partition of unity finite element method [17], element free Galerkin [18] and the method of finite spheres [19,20], to mention a few. We resort in this work to the local maximum entropy (LME) approximation schemes, a meshfree method inspired on information theory that generates nonnegative and smooth basis functions (see [21–25] for a detailed description, properties and extensions). Because the LME basis functions do not satisfy the Kronecker-delta property at nodes, these schemes are referred to as approximants instead of interpolants. The capabilities of LME approximants have been examined in a variety of computational mechanics applications, such as linear and nonlinear elasticity [25,26], plate [27] and thin-shell analysis [28,29], convection–diffusion problems [30,31], and phase-field models of biomembranes [32,33] and fracture mechanics [34–36].

Like other meshfree methods, LME approximants involve a dilation or locality parameter that modulates their behavior and support. LME approximants show an exponential decay controlled by the locality parameter, and far from the boundaries they look like Gaussian weighted functions [37,24]. Their effective support is controlled by setting a cut off or threshold value ($\mathtt{Tol}_0$) below which the basis functions are taken numerically to be zero (see Fig. 1 and Appendix A). The proper choice of the locality parameter is problem dependent and not easy in general, which has motivated a systematic studies for general meshfree methods [38] and for LME approximants [25] in particular. In LME approximations, the locality parameter is an aspect ratio parameter γ, which allows us to smoothly move from linear finite elements shape functions ($\gamma > 4.0$) to more spread out approximation schemes (e.g., $\gamma = 0.6$), as illustrated in Fig. 2. In general, broader functions lead to more accurate results for problems with smooth solutions at the expense of higher computational cost and worse matrix conditioning [22,28].

* Corresponding author.
E-mail address: marino.arroyo@upc.edu (M. Arroyo).

http://dx.doi.org/10.1016/j.compstruc.2014.12.005
0045-7949/© 2014 Elsevier Ltd. All rights reserved.

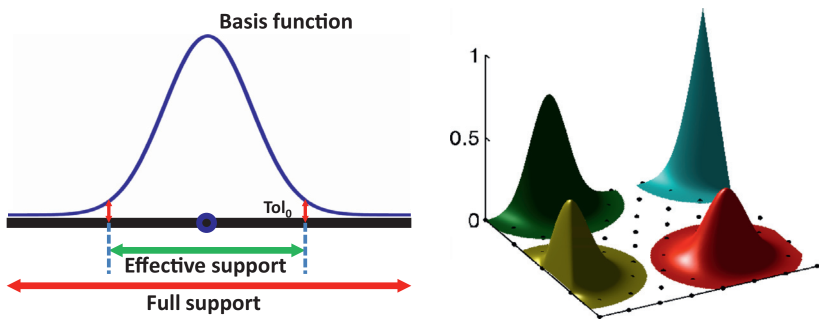

Fig. 1. Full support of some meshfree basis functions, such as local maximum entropy approximants, covers the convex hull of the computational domain. The effective numerical support radius r_a is determined by a cutoff basis function value $\mathtt{Tol}_0$ (left). Representation of two-dimensional LME approximants basis functions (right). Notice the noninterpolant character and the smoothness of the basis functions, and the fulfillment of a weak Kronecker-delta at the boundary of the convex hull.

In contrast to conventional FEM, where the structure of the matrix is inherited from the mesh graph, stencils of meshfree schemes depend strongly on the aspect ratio parameter γ. In our experience, a noticeable run-time computational cost of meshfree methods is due to the creation of the sparse matrix structure and the assembly process, which can be specially harmful for iterative processes. These stages can be as expensive as the solver stage in two-dimensional problems and exceed it in three-dimensional ones. In a typical implementation of the assembly process in meshfree methods, the code loops over the quadrature points. The denser sparsity pattern and the large number of Gauss points required for accurate integration can make these methods unpractical for large-scale calculations. To overcome this issue, we propose here a set of algorithms based on a loop over cells/elements, as commonly done in FEM. We illustrate in this work how this simple approach reduces significantly the computational cost associated with the matrix structure creation and the assembly process.

Additionally, a widespread practice (both in FEM and in meshfree methods) is to store in memory the basis functions and their derivatives for repetitive calculations required in nonlinear iterative solvers, incremental loading, or evolution in time. In FEM, this storage is insignificant because the basis functions of the parent element are mapped to each physical element. Since this is not the case in meshfree methods, the amount of memory and its access can become a bottleneck and substantially reduce the code efficiency, especially in large-scale problems. If meshfree basis functions are not stored in memory but recomputed every time, the computational cost can also increase significantly. To alleviate this issue, we propose here a strategy that is a trade-off solution between memory storage and computational time. The technique, based on a data structure that stores only partial information about the basis functions and an algorithm to reconstruct them when needed, reduces considerably the memory usage at the expense of a minimum increment in the overall computational cost. We illustrate and exploit this concept on LME approximants.

The paper is organized as follows. In Section 2 we review the basic technicalities for a meshfree method particularized to LME approximants and the classical implementation to approximate partial differential equations (PDEs). We then propose an algorithm to speed-up the matrix assembly and an algorithm for the compressed memory storage of LME approximants in Section 3. We extensively test our proposals with numerical examples in Section 4 and finish with some concluding remarks in Section 5.

2. A standard meshfree scheme

Let $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\} \subset \mathbb{R}^d$, for $d = 1, 2, 3$, be an unstructured set of nodes used to describe a domain Ω, and $p_a(\boldsymbol{x})$ the meshfree basis function associated to the a-th node, for $a = 1, \ldots, N$. A continuous field Φ can be approximated as

$$\Phi(\boldsymbol{x}) = \sum_{a=1}^{N} p_a(\boldsymbol{x})\Phi_a,$$

where Φ_a stand for the nodal coefficients. Here we adopt the LME approximants as meshfree basis functions in a Galerkin method to approximate a general PDE. They are nonnegative, smooth, satisfy at least up to the first order consistency conditions and present a weak Kronecker-delta property. We rely on a background mesh to define the quadrature points, typically through a Gauss–Legendre quadrature rule. Since this mesh is just required to place the Gauss points, its requirements are less strict than in a mesh-based method. We use here meshes made of triangles/tetrahedra in 2D/3D, which we easily obtain via the library QHULL [39]. The procedure needed to compute the system matrix in a Galerkin meshfree approach requires mainly four steps: (i) neighborhood search, (ii) computation of the basis functions, (iii) creation of the sparse matrix structure and (iv) Gauss point-wise matrix filling. The pseudocode shown in Algorithm 1 summarizes these four steps. In the present work, we do not deal with solver performance. In the following we briefly extend on the computational implications of every step.

Algorithm 1. Pseudo-code for scheme based on a loop over quadrature points (see Section 2).

```
(i) Determine the neighborhood nodal index set $\mathcal{N}_{\boldsymbol{y}}^{X}$ for
    each Gauss point.
(ii) Compute shape functions (array $p_a$).
(iii) Construct sparse matrix structure (arrays $ia$ and $ja$).
(iv) Fill sparse matrix (array $an$) with the quadrature point
    loop based algorithm.
```

The objective of step (i) is to compute the so-called neighbor lists, which can be interpreted as the counterpart of the mesh connectivity in FEM where the neighbor lists are given by the mesh itself. In a meshfree scheme this is made by specialized algorithms, i.e. neighbor searchers which identify the relationship between the quadrature points and the nodes. We will refer to the neighbor lists as *primal* and *dual lists* [28], which are complementary. In particular, a dual list identifies the quadrature points that are influenced by a particular node i.e. the quadrature points falling within the effective support of a nodal basis function. Conversely, the primal list contains the nodes that influence a particular quadrature point.

Formally, let $Y = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_L\} \subset \Omega$ be a set of quadrature points. The dual list containing the nearest points from Y associated with a node $\boldsymbol{x}_a \in X$ can be defined as follows

$$\widetilde{\mathcal{N}}_{\boldsymbol{x}_a}^{Y} = \{k \in \{1, 2, .., L\} \mid |\boldsymbol{y}_k - \boldsymbol{x}_a| < r_a\},$$

Download English Version:

https://daneshyari.com/en/article/510012

Download Persian Version:

https://daneshyari.com/article/510012

Daneshyari.com