Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 305 (2016) 1–36
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Stable Generalized Finite Element Method and associated iterative schemes; application to interface problems

Kenan Kergrene[a], Ivo Babuška[b], Uday Banerjee[c,*]

[a] *Department of Mathematics and Industrial Engineering, École Polytechnique de Montréal, Canada*
[b] *ICES, University of Texas at Austin, Austin, TX, United States*
[c] *Department of Mathematics, 215 Carnegie, Syracuse University, Syracuse, NY 13244, United States*

Received 4 November 2015; received in revised form 22 February 2016; accepted 23 February 2016
Available online 8 March 2016

Abstract

The Generalized Finite Element Method (GFEM) is an extension of the Finite Element Method (FEM), where the standard finite element space is augmented with a space of non-polynomial functions, called the enrichment space. The functions in the enrichment space mimic the local behavior of the unknown solution of the underlying variational problem. GFEM has been successfully applied to a wide range of problems. However, it often suffers from bad conditioning, i.e., its conditioning may not be robust with respect to the mesh and in fact, the conditioning could be much worse than that of the standard FEM. In this paper, we present a numerical study that shows that if the "angle" between the finite element space and the enrichment space is bounded away from 0, uniformly with respect to the mesh, then the GFEM is stable, i.e., the conditioning of GFEM is not worse than that of the standard FEM. A GFEM with this property is called a Stable GFEM (SGFEM). The last part of the paper is devoted to the derivation of a robust iterative solver exploiting this angle condition. It is shown that the required "wall-clock" time is greatly reduced compared to popular GFEMs used in the literature.

Keywords: Generalized Finite Element Method (GFEM); Partition of Unity Method (PUM); Stable GFEM (SGFEM); Condition number; Angle condition

1. Introduction

The Generalized Finite Element Method (GFEM) has sparked a lot of interest in the last 20 years and has been successfully applied to a wide range of engineering problems, e.g., crack-propagation, material modeling, and solid–fluid interactions. We refer to the review articles [1–4] and the citations therein for various applications of GFEM. The method has been incorporated into commercial codes, e.g., Abaqus and LS-DYNA [5,6]. It is also known in the literature as the Extended Finite Element Method (XFEM). We will simply refer to the method as GFEM and we will address special instances of this method such as SGFEM and M-GFEM.

* Corresponding author.
E-mail address: banerjee@syr.edu (U. Banerjee).

http://dx.doi.org/10.1016/j.cma.2016.02.030

As the name GFEM/XFEM suggests, the GFEM is a generalization/extension of the standard Finite Element Method (FEM). Specific non-polynomial local basis functions that mimic special features (e.g., singularity) of the solution of the underlying PDE model of interest, are used in this method in addition to the standard “hat-functions”. These additional local basis functions are called the local *enrichment* functions. In fact, the GFEM is a particular instance of the Partition of Unity Method (PUM). The PUM, developed in [7–9], allows the use of any Partition of Unity (PU) together with local enrichment functions. The GFEM is a PUM, where the finite element “hat-functions” serve as the PU. Various methods for solving multi-scale problems are also based on PUM; see for example [10,11,3]. The PUM based on a “flat-top” PU was developed in [12,13]. A similar idea, referred to as $h-p$ Cloud method, was developed in [14,15]. The original idea of GFEM, i.e., the use of hat-functions as the PU, was introduced in [16]. Since then, the GFEM has been developed, refined, and used in various applications in two and three dimensions, e.g., in [17–25]. The GFEM is often referred to as the XFEM in the literature. We mention that the use of local non-polynomial approximation, not in the framework of PUM, was suggested earlier in [26].

The XFEM/GFEM was initially developed as a computational method with essentially intuitive understanding of the necessity of appropriate enrichments for convergence. The method was primarily tested numerically to ensure convergence. Appropriate enrichment functions for various applications were identified in the literature and the optimal convergence of the approximate solution was shown through computations. A rigorous mathematical proof of optimal convergence was derived in [27], in the context of a crack problem.

Though approximability and optimal convergence are very important features of a numerical method such as GFEM, it is equally important that the underlying linear system could be solved accurately and efficiently. Solving such linear systems accurately and efficiently depends on the stability of the GFEM, i.e., on the conditioning of the underlying linear system. It was reported early in [7,4] that the GFEM could be unstable and that its conditioning may not be robust with respect to the mesh. However, there are very few papers that addressed these issues by carefully studying the conditioning of the GFEM and by examining the performance of associated iterative solvers to solve the linear system. Various ad-hoc stabilization procedures were used to address these issues in [28–30,22,24]. Stabilization based on a local orthogonalization idea was used in [29]. We mention however that local orthogonalization was also addressed in the context of PUM with flat-top PU in [31].

Extensive literature is available on the loss of accuracy in the computed solution of a linear system; we refer to the monographs [32,33]. In [34], it was shown that the Scaled Condition Number (SCN) of the matrix, related to FEM and GFEM, is a good indicator of the stability and loss of accuracy in the solution obtained from elimination methods.

The conditioning of GFEM was addressed in [34] where the idea of a stable GFEM (SGFEM) was introduced. In general, a GFEM is called stable (SGFEM) if

(i) it yields the optimal order of convergence, and
(ii) the SCN of the linear system associated with the GFEM is of the same order $O(h^{-2})$ (h being the discretization parameter) as that of a standard FEM in a robust manner with respect to the mesh.

It was shown in [34] that the SCN of a GFEM could be much higher than that of the FEM, e.g., $O(h^{-4})$. It was mathematically established in that paper that if the enrichments satisfy two specific conditions, then the SCN of the underlying GFEM is of the same order as that of a standard FEM. For various problems in the 1-D setting, a simple modification based on subtracting the piecewise linear interpolant of the standard enrichment was suggested in [34] and it was shown that the modified GFEM was indeed an SGFEM for these problems. However, the modification suggested in [34] may lead to loss of accuracy in some problems in higher dimensions as shown in [35–37]. It was shown in [35,36] that a further modification of “Heaviside enrichment”, in the context of a problem with a crack, is required for a GFEM to be an SGFEM, i.e., the further modification restores the accuracy of the computed solution while retaining the well-conditioning of the linear system. Thus a GFEM with the simple modification of enrichments as suggested in [34] may not yield an SGFEM for every problem; further modifications of the enrichments may be required for a GFEM to be an SGFEM.

In this paper, we consider an “interface problem” modeled by a scalar second order elliptic PDE in 2-D with piecewise smooth coefficients. We will numerically investigate the accuracy, conditioning, and the robustness of the GFEM associated with various forms of enrichments used in the literature, when applied to this problem. We will especially investigate the performance of an iterative procedure to solve the underlying linear system of the GFEM, where the stopping criterion is based on computed *discretization error* and *truncation error*.