Computational and Applied Mathematics (2020) 39:78
https://doi.org/10.1007/s40314-020-1081-4

An adaptive strategy for solving convection dominated diffusion equation

Zahra Jannesari[1] · **Mehdi Tatari**[1,2]

Received: 12 August 2018 / Revised: 17 December 2019 / Accepted: 7 January 2020 /
Published online: 15 February 2020

Abstract
In this work, an adaptive element free Galerkin (EFG) technique is proposed for solving convection diffusion equation. A post-processed gradient is used as a posteriori error estimation to find locations with large contribution of the error. Also, to avoid instabilities, the EFG method is applied on a modified equation instead of the original equation. In the modified equation diffusion coefficient (known as artificial diffusion) depends to the distance between nodal points. The numerical results reveal efficiency of the adaptive technique.

Keywords Element free Galerkin (EFG) method · Moving least squares (MLS) approximation · Error estimation · Adaptive technique · Local refinement

Mathematics Subject Classification 65N50 · 65N30 · 65N99

1 Introduction

The convection diffusion equation is given by

$$\begin{aligned} &-\epsilon\Delta u + \mathbf{b}.\nabla u + cu = f \quad \text{in } \Omega,\\ &u = g_D \quad \text{on } \Gamma_D,\\ &\epsilon\frac{\partial u}{\partial n} = g_N \quad \text{on } \Gamma_N, \end{aligned} \tag{1.1}$$

where ϵ is a small positive parameter and Ω is an open bounded domain enclosed with $\partial\Omega = \Gamma_D \cup \Gamma_N$. Moreover, n is the outward unit normal to the boundary. This type of equations plays

Communicated by Corina Giurgea.

✉ Mehdi Tatari
mtatari@cc.iut.ac.ir

Zahra Jannesari
z.jannesari@shahreza.ac.ir

1 Department of Mathematical Sciences, Isfahan University of Technology, Isfahan 84156-83111, Iran

2 School of Mathematics, Institute for Research in Fundamental Sciences (IPM), P.O. Box: 19395-5746, Tehran, Iran

an important role in many physical phenomena. For example, scalar convection diffusion equations describe the transport of a scalar quantity, e.g. temperature or concentration. In these equations the operators $-\epsilon\Delta$ and $\mathbf{b}.\nabla$ determine what the solution of (1.1) looks like. The first one relates u proportionally to ϵ and the second one transports u in the direction of vector $\mathbf{b}$ (Larson and Bengzon 2013). Hence, these operators model the physical processes of diffusion and convection, respectively. Generally, solutions of the convection diffusion equations have layers, due to this trait there are some small parts of the domain where derivative of the solution is very large (John 2000). In Tang and Trummer (1996) boundary layer resolving pseudospectral methods are presented to overcome this problem. Adaptive techniques are very suitable to this kind of problems. In adaptive techniques a posteriori error estimation is needed to find information about locations for local mesh refinement as well as for estimating the global error. There are several works in implementation and analysis of a posteriori error estimation for solving many classes of partial differential equations in finite element methods (Ainsworth and Oden 1993; Bank and Weiser 1985; Eriksson et al. 1995). A good review of some a posteriori error estimation for convection diffusion equations can be found in John (2000).

The main challenge in convection diffusion problems depends to how large is "*Péclet number*" which provides a measure of how much the convective term prevails over the diusive one. A problem featuring $Pe \gg 1$ will be named convection dominated diffusion problem (Quarteroni et al. 2014). In this case behaviour of the numerical solutions is oscillatory. To avoid these solutions, discretization parameters should be chosen sufficiently small which makes the numerical method inconvenient. An approximate solution of the problem does not exhibit oscillations if $Pe < 1$. A stabilization technique is based on adding an artificial diffusion to make Péclet number as small as possible. Therefore, for smaller values of ϵ, this stabilization technique is more efficient. This more diffusion should be as little as possible not to sacrifice accuracy, but as much as need to obtain stability (Larson and Bengzon 2013).

The current work presents an adaptive element free Galerkin method for solving convection diffusion equations. The EFG method was introduced by Belytschko et al. (1994), Dolbow and Belytschko (1998). This method is based on the moving least squares (MLS) Yaw (2009) approximation and a background mesh for the integration purpose. The MLS shape functions do not have Kronecker delta property, therefore, the essential boundary condition should be enforced. In Dehghan and Abbaszadeh (2018a, b) the authors have combined the EFG method with the moving Kriging interpolation and radial point interpolation which have Kronecker delta property for solving transport problems, incompressible Navier–Stokes equation and some PDEs with discontinuous solutions. The element free Galerkin method is used for solving real world problems in Jannesari and Tatari (2016, 2017) and Dehghan and Narimani (2018). More about meshfree methods are presented in Liu (2003).

Rest of the paper is organized as follows: Sect. 2 is devoted to explain EFG method and MLS approximation. Section 3 explains the proposed adaptive technique and in Sect. 4, numerical results for some problems are presented to confirm validity of the approach. The last section is conclusion.

2 The EFG method

The element free Galerkin method is based on the weak formulation of the considered problem and MLS approximation. In MLS approximation nodal points are used and does not need any mesh generation. This approximation is not necessarily interpolant, hence enforcement of Dirichlet boundary conditions is not straightforward. There are several ways to impose these boundary conditions such as Lagrange multipliers and penalty method. In this work, the penalty method is used to impose Dirichlet boundary conditions.

2.1 MLS approximation

The MLS approximation which was introduced by Lancaster and Salkauskas (1981) is well-known technique with acceptable accuracy. This approximation is local and at any arbitrary evaluation point $\mathbf{x}$, only the neighboring nodes in influence domain of point $\mathbf{x}$ are consequential. The influence of a node $\mathbf{x}_i$ is governed with a weight function $w(\mathbf{x}-\mathbf{x}_i)$, which vanishes outside of the influence domain of node $\mathbf{x}_i$. Let the true solution u be known at some selected points $\mathbf{x}_i$. To approximate solution $u_h(\mathbf{x})$ in the problem domain $\bar{\Omega}$, by least squares sense, the goal is to find minimum of the expression $(u_h(\mathbf{x}_i)-u(\mathbf{x}_i))^2$ for each i. Let the approximation $u_h(\mathbf{x})$, be posed as a polynomial of order m with variant coefficients in the following matrix form

$$u_h(\mathbf{x}) = \mathbf{p}^{\mathrm{T}}(\mathbf{x})\mathbf{a}(\mathbf{x}), \qquad \forall \mathbf{x} \in \bar{\Omega}, \tag{2.1}$$

where $\mathbf{p}^{\mathrm{T}}(\mathbf{x}) = [p_1(\mathbf{x})\ p_2(\mathbf{x})\ \cdots\ p_m(\mathbf{x})]$ consists of complete monomial of order m. In two dimensional cases, linear and quadratic basis are defined as:

$$\begin{aligned}\mathbf{p}^{\mathrm{T}}(\mathbf{x}) &= [1\ x\ y] \quad \text{linear basis}, \quad m=3,\\ \mathbf{p}^{\mathrm{T}}(\mathbf{x}) &= [1\ x\ y\ x^2\ xy\ y^2] \quad \text{quadratic basis}, \quad m=6.\end{aligned}$$

The vector $a(\mathbf{x})$ is given by

$$\mathbf{a}(\mathbf{x}) = [a_1(\mathbf{x})\ \cdots\ a_m(\mathbf{x})]^{\mathrm{T}}.$$

The unknown parameters $a_j(\mathbf{x})$, $j=1,\ldots,m$, vary with space coordinates $\mathbf{x}$. Therefore, $\mathbf{a}(\mathbf{x})$ should be determined at any given point $\mathbf{x}$. In doing so, the least squares functional is written as following:

$$\mathcal{J}(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{N_t} w(\mathbf{x}-\mathbf{x}_i)\Big(\mathbf{p}^{\mathrm{T}}(\mathbf{x}_i)\mathbf{a}(\mathbf{x}) - u(\mathbf{x}_i)\Big)^2. \tag{2.2}$$

where N_t is the total number of points. The coefficient $\frac{1}{2}$ is added for mathematical convenience. Also, each summation term, is weighted by a weight function $w(\mathbf{x}-\mathbf{x}_i)$ thus the local solution is influenced by the local nodes that are not far away. To minimize $\mathcal{J}$ with respect to each $a_i(\mathbf{x})$, for the sake of simplicity, first write functional $\mathcal{J}$ in the following matrix form

$$\mathcal{J}(\mathbf{x}) = \frac{1}{2}[\mathbf{Pa}(\mathbf{x})-\mathbf{u}]^{\mathrm{T}}\mathbf{W}[\mathbf{Pa}(\mathbf{x})-\mathbf{u}], \tag{2.3}$$

where

$$\mathbf{P} = \begin{bmatrix}\mathbf{p}^{\mathrm{T}}(\mathbf{x}_1)\\ \mathbf{p}^{\mathrm{T}}(\mathbf{x}_2)\\ \vdots\\ \mathbf{p}^{\mathrm{T}}(\mathbf{x}_{N_t})\end{bmatrix}_{N_t\times m}, \mathbf{W} = \begin{bmatrix} w(\mathbf{x}-\mathbf{x}_1) & \cdots & 0\\ \cdots & \ddots & \cdots\\ \mathbf{0} & \cdots & w(\mathbf{x}-\mathbf{x}_{N_t})\end{bmatrix}_{N_t\times N_t},$$

and

$$\mathbf{u} = [u_1, u_2, \ldots, u_{N_t}]^{\mathrm{T}}.$$

Setting $\frac{\partial \mathcal{J}}{\partial \mathbf{a}} = 0$ yields the following:

$$(\mathbf{Pa}(\mathbf{x}) - \mathbf{u})^{\mathrm{T}}\mathbf{WP} = 0. \tag{2.4}$$

Transpose the whole Eq. (2.4) gives:

$$(\mathbf{WP})^{\mathrm{T}}(\mathbf{Pa}(\mathbf{x}) - \mathbf{u}) = 0. \tag{2.5}$$

This equation can be rewritten as follows:

$$\mathbf{P}^{\mathrm{T}}\mathbf{WPa}(\mathbf{x}) = \mathbf{P}^{\mathrm{T}}\mathbf{Wu}. \tag{2.6}$$

Define moment matrices $\mathbf{A}(\mathbf{x})$ and $\mathbf{B}(\mathbf{x})$ as follows:

$$\mathbf{A}(\mathbf{x}) = \mathbf{P}^{\mathrm{T}}\mathbf{WP} = \sum_{i=1}^{N_t} w(\mathbf{x} - \mathbf{x}_i)\mathbf{p}(\mathbf{x}_i)\mathbf{p}^{\mathrm{T}}(\mathbf{x}_i), \tag{2.7}$$

$$\mathbf{B}(\mathbf{x}) = \mathbf{P}^{\mathrm{T}}\mathbf{W} = [w(\mathbf{x} - \mathbf{x}_1)\mathbf{p}(\mathbf{x}_1), w(\mathbf{x} - \mathbf{x}_2)\mathbf{p}(\mathbf{x}_2), \ldots, w(\mathbf{x} - \mathbf{x}_{N_t})\mathbf{p}(\mathbf{x}_{N_t})]. \tag{2.8}$$

Using these definitions, Eq. (2.6) becomes:

$$\mathbf{A}(\mathbf{x})\mathbf{a}(\mathbf{x}) = \mathbf{B}(\mathbf{x})\mathbf{u}, \tag{2.9}$$

solve for unknown coeffiecints $\mathbf{a}(\mathbf{x})$ and substituting it into Eq. (2.1):

$$u_h(\mathbf{x}) = \sum_{i=1}^{N_t} \phi_i(\mathbf{x})u_i = \mathbf{\Phi}^{\mathrm{T}}(\mathbf{x})\mathbf{u}, \qquad \mathbf{x} \in \bar{\Omega}, \tag{2.10}$$

where

$$\mathbf{\Phi}^{\mathrm{T}}(\mathbf{x}) = \mathbf{p}^{\mathrm{T}}(\mathbf{x})\mathbf{A}^{-1}(\mathbf{x})\mathbf{B}(\mathbf{x}), \tag{2.11}$$

is the vector of shape functions. Moreover, derivatives of shape function can be obtained using the product rule on Eq. (2.11)

$$\mathbf{\Phi}^{\mathrm{T}}_{,k}(\mathbf{x}) = \mathbf{p}^{\mathrm{T}}_{,k}\mathbf{A}^{-1}\mathbf{B} + \mathbf{p}^{\mathrm{T}}\mathbf{A}^{-1}_{,k}\mathbf{B} + \mathbf{p}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{B}_{,k}, \tag{2.12}$$

with

$$\mathbf{A}^{-1}_{,k} = -\mathbf{A}^{-1}\mathbf{A}_{,k}\mathbf{A}^{-1}, \tag{2.13}$$

where

$$\mathbf{A}_{,k}(\mathbf{x}) = \sum_{i=1}^{N_t} w_{,k}(\mathbf{x} - \mathbf{x}_i)\mathbf{p}(\mathbf{x}_i)\mathbf{p}^{\mathrm{T}}(\mathbf{x}_i). \tag{2.14}$$

Further details about MLS shape functions can be found in Yaw (2009). In all numerical examples we have used shifted and scaled MLS shape functions that computationally are more efficient. Details about how to construct these shape functions are presented in Belytschko et al. (1996).

2.2 The governing system

Consider Eq. (1.1) as follows:
find $u \in H^1(\Omega)$ from

$$a(u, v) = l(v), \quad \forall v \in H^1(\Omega), \tag{2.15}$$

where

$$a(u, v) = \int_\Omega (-\epsilon \Delta u v + \mathbf{b}.\nabla u v + cuv) \mathrm{d}\Omega,$$

and

$$l(v) = (f, v) = \int_\Omega f v d\Omega,$$

where v is the test function that is chosen as MLS shape function. Applying the integration by parts formula to the above equation and adding the penalty terms to enforce Dirichlet boundary conditions lead to the following system (Dolbow and Belytschko 1999):

$$(\mathbf{S} + \mathbf{E} + \mathbf{M} + \mathbf{M_p})\mathbf{u} = \mathbf{F} + \mathbf{F_N} + \mathbf{F_p},$$

where

$$S_{ij} = \epsilon \int_\Omega \nabla\phi_i.\nabla\phi_j \mathrm{d}\Omega, \tag{2.16}$$

$$E_{ij} = \int_\Omega \mathbf{b}.\nabla\phi_i\phi_j \mathrm{d}\Omega, \tag{2.17}$$

$$M_{ij} = \int_\Omega c\phi_i\phi_j \mathrm{d}\Omega, \tag{2.18}$$

$$M_{p_{ij}} = \int_{\Gamma_D} \phi_i\gamma\phi_j \mathrm{d}s, \tag{2.19}$$

and

$$F_i = \int_\Omega f_i\phi_i \mathrm{d}\Omega, \tag{2.20}$$

$$F_{N_i} = \int_{\Gamma_N} \epsilon \frac{\partial u_i}{\partial n}\phi_i \mathrm{d}s, \tag{2.21}$$

$$F p_i = \int_{\Gamma_D} g_{D_i}\gamma\phi_i \mathrm{d}s. \tag{2.22}$$

The parameter γ in Eqs. (2.19) and (2.22) is used to penalize difference between Dirichlet boundary conditions and the obtained solution by EFG approximation. Moreover, it should be noted that the EFG method requires the partitioning of the domain into cells, to evaluate all integrals appeared in Eqs. (2.16)–(2.22).

In the point of geometry, these cells should be as simple as possible and usually are rectangle or triangle. In this work, we use a triangular mesh as a partition of the domain. Based on Reference (Belytschko et al. 1994) a proper ratio of Gaussian points to total number of nodes is 4–7. To get this ratio, 3-point Gaussian rule is used, however, the results are acceptable using one point rule. An excellent detail about integration on triangle can be found in Gockenbach (2006).

2.2.1 Stabilization

Based on what is discussed in Larson and Bengzon (2013), when ϵ in Eq. (1.1) decreases we lose control of gradients of u, i.e. ∇u can grow sharply. In other words, small perturbations of f can lead to a large local values of ∇u. Indeed, it is common for u to has thin regions called layers where it changes rapidly. Due to large local values of ∇u there are great difficulties in handling layers and thus need modification of the numerical method. There are several techniques to do this such as adding an artificial diffusion, least squares stabilization (John 2000) and edge stabilization that is based on least square stabilization of the gradient jumps across the element boundaries (Burman and Hansbo 2004). In this paper $\frac{h}{2}\mathbf{b}$ is added to the diffusion term as an artificial diffusion, where h is mesh size. Therefore, Eqs. (2.16) and (2.21) are replaced by the following equations

$$S_{ij} = \int_{\Omega} \left[\epsilon + \frac{h}{2} b_1 \phi_{i_x}, \epsilon + \frac{h}{2} b_2 \phi_{i_y}\right] . \nabla \phi_j \mathrm{d}\Omega,$$

and

$$F_{N_i} = \int_{\Gamma_N} \left[\epsilon + \frac{h}{2} b_1 u_{i_x}, \epsilon + \frac{h}{2} b_2 u_{i_y}\right] . n \phi_i \mathrm{d}s.$$

Although, this work is presented with the aim of implementation of an adaptive EFG technique for convection diffusion problems, stabilization strategy can help us to get better results with less computational effort.

3 The adaptive algorithm

Let $\mathcal{T} = \{T\}$ be a partition of domain Ω and the mesh size h_T is defined by $h_T = \mathrm{diam}(T)$. The following adaptive algorithm generates a sequence of meshes, $\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_2, \ldots$.

- Start with a coarse mesh $\mathcal{T}_0$.
- Solve the problem to get the discrete solution u_h on the current mesh.
- Compute the error on each element using a suitable a posteriori error estimation.
- Mark a fixed (θ) percentage of those element with largest error contribution.
- Refine the marked element with a local mesh refinement technique such as red-green refinement and longest edge bisection, to generate new triangulation $\mathcal{T}_{k+1}$. Details of these local refinement techniques can be found in most finite element books such as Gockenbach (2006). If in the previous step we chose $\theta = 1$, then the refinement is global.
- Continue this process until get acceptable tolerance, or when the maximum number of elements in the mesh exceeds from a user predefined number.

3.1 A posteriori error control

Let $e_h = u - u_h$ be the numerical error relates to the exact solution u and numerical solution u_h. Instead of measuring the error of the solution in some applications, it maybe useful to consider the gradient of error, i.e. $\nabla e_h = \nabla u - \nabla u_h$. In most problems the exact value of the gradient is not known. Here, the main idea is post processing the gradient and to find an estimate for the true error by comparing the post-processed gradient and non post-processed gradient of the numerical solution u_h (Gratsch and Bathe 2005). Let Gu_h denote an improved (post-processed) approximation to the gradient. It can be approximated by

$$Gu_h(\mathbf{x}) = \sum_{i=1}^{n_i} \phi_i(\mathbf{x}) gu_{h_i}, \tag{3.1}$$

where n_i is number of points in the influence domain of $\mathbf{x}$. The coefficients gu_{h_i} can be determined by solving the following equation:

$$(Gu_h, \phi) = (\nabla u_h, \phi), \tag{3.2}$$

or, in the other words, solving the following linear system:

$$\int_\Omega (Gu_h - \nabla u_h)\phi_j \mathrm{d}\Omega = 0, \quad j = 1, 2, \ldots n_i. \tag{3.3}$$

Substituting (3.1) into Eq. (3.3) leads to:

$$\sum_{i=1}^{n_i} \int_\Omega \phi_i \phi_j gu_{h_i} \mathrm{d}\Omega = \int_\Omega \phi_j \nabla u_{h_j} \mathrm{d}\Omega. \tag{3.4}$$

Then using gu_{h_i} to find Gu_h and applying it instead of the true gradient, yield the following a posteriori error estimation (Gratsch and Bathe 2005)

$$\|e_h\|_E^2 \approx (E_h)^2 = \sum_{T \in \mathcal{T}} \eta_T^2, \tag{3.5}$$

where $\|.\|_E$ shows the energy norm and

$$\eta_T^2 = \|Gu_h - \nabla u_h\|_{L^2(T)}^2.$$

This kind of error estimation is known as recovery-based error estimation was introduced by Zienkiewicz and Zhu (1992).

4 Numerical investigation

In this section, the EFG method is applied for some examples to demonstrate efficiency and accuracy of the proposed method. In all of the examples linear basis ($m = 3$) and following cubic spline weight function with rectangular support are used

$$w(r) = \begin{cases} 4r^3 - 4r^2 + \frac{2}{3} & r \le \frac{1}{2}, \\ -\frac{4}{3}r^3 + 4r^2 - 4r + \frac{4}{3} & \frac{1}{2} < r \le 1, \\ 0 & r > 1. \end{cases} \tag{4.1}$$

where $r = \frac{||x - x_i||_2}{d_i}$, and d_i is about $1.5h$ where h is the maximum diameter of the triangles that one of their vertices is x_i. Moreover, all examples are solved in unit square $[0, 1] \times [0, 1]$ and with $\gamma = 1000$ as the penalty parameter. To show efficiency of the present method, the initial mesh is chosen very coarse. The unit square is only divided into two triangles as the first mesh. Also, we set $\theta = 0.3$, and use the following definition as effectivity index

$$\text{Effectivity index} := \frac{(\sum_{T \in \mathcal{T}} \eta_T^2)^{1/2}}{\|e_h\|_E}.$$

Besides, in the following tables L^2 error is reported on some level of refinements where adaptive.S and uniform.S are adaptive and uniform methods using stabilization and N_t is total number of nodes.

Table 1 Error of computed solution by uniform refinement, Example 1

Method	Level 5		Level 9		Level 13	
	N_t	$\|u-u_h\|_{L^2}$	N_t	$\|u-u_h\|_{L^2}$	N_t	$\|u-u_h\|_{L^2}$
Uniform	41	1.38e+0	545	4.92e−1	8321	6.10e−2
Uniform.S	41	1.46e−1	545	7.68e−2	8321	3.75e−2

Table 2 Error of computed solution on adaptive refined mesh, Example 1

Method	Level 6		Level 12		Level 18	
	N_t	$\|u-u_h\|_{L^2}$	N_t	$\|u-u_h\|_{L^2}$	N_t	$\|u-u_h\|_{L^2}$
Adaptive	26	2.89e−1	393	8.81e−1	4773	7.24e−2
Adaptive.S	25	1.87e−1	242	6.55e−2	2036	3.76e−2

Table 3 Relative error and CPU time of computed solution, Example 1

Adaptive.S				Uniform.S			
Level	N_t	$Re(u)$	Time	Level	N_t	$Re(u)$	Time
6	25	8.76e−2	2	7	145	3.30e−2	3
12	242	8.38e−3	10	10	1089	8.57e−3	19
18	2036	1.13e−3	106	13	8321	1.43e−3	472

4.1 Example 1

As the first numerical experiment consider following example with $\epsilon = 10^{-3}$, $\mathbf{b} = (2, 3)^{\mathrm{T}}$ and $c = 1$ and $\partial\Omega = \Gamma_D$. The right hand side f and Dirichlet boundary conditions are chosen such that

$$u = xy^2 - y^2 \exp\left(\frac{2(x-1)}{\epsilon}\right) - x \exp\left(\frac{3(y-1)}{\epsilon}\right) + \exp\left(\frac{2(x-1)+3(y-1)}{\epsilon}\right).$$

In this case, the solution has typical regular boundary layers at $x = 1$ and $y = 1$ (John 2000). In Tables 1 and 2, L^2 error of solutions computed by uniform and adaptive refinements are reported respectively. In each of these tables stabilized and unstabilized cases are investigated. According to these tables stabilization technique is more efficient for adaptive refinement. In Table 3 relative error and CPU time of stabilized adaptive and uniform refinements are compared which shows efficiency of the presented stabilized adaptive method.

The numerical solution and the last mesh of this test are plotted in Fig. 1. Also, Fig. 2 shows the effectivity indices and obtained L^2 error for this example. In this example, the effectivity indices are lower than one. Indeed, the error is underestimated by around a factor between 0.1 and 0.5. We emphasize that an a posteriori error estimator is called efficient, if the effectivity index (EI) and inverse of it $(\frac{1}{EI})$ are bounded for all meshes. Besides, if it does not vary too much with respect to a given mesh. Also, Fig. 3 shows convergence rate for this example.

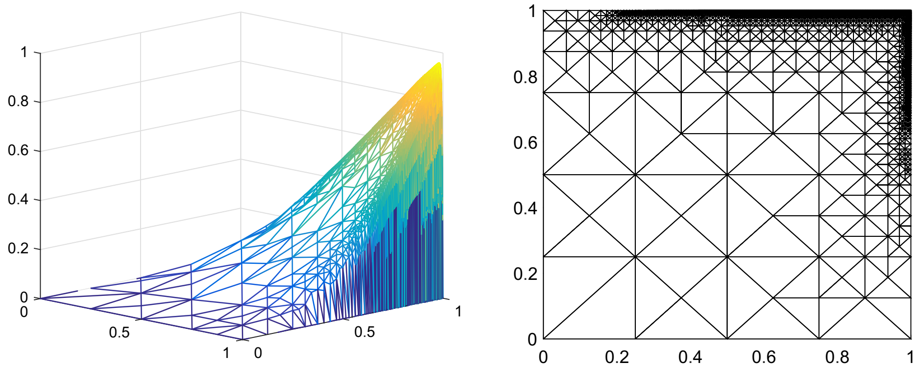

Fig. 1 EFG solution by adaptive technique (left) and corresponding background mesh on level 20 (right), Example 1

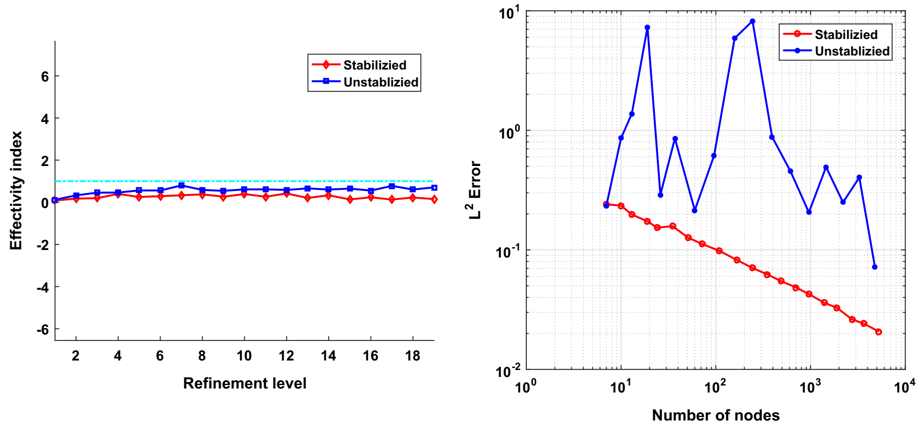

Fig. 2 The effectivity index as a function of refinement (left) and L^2 error (right), Example 1

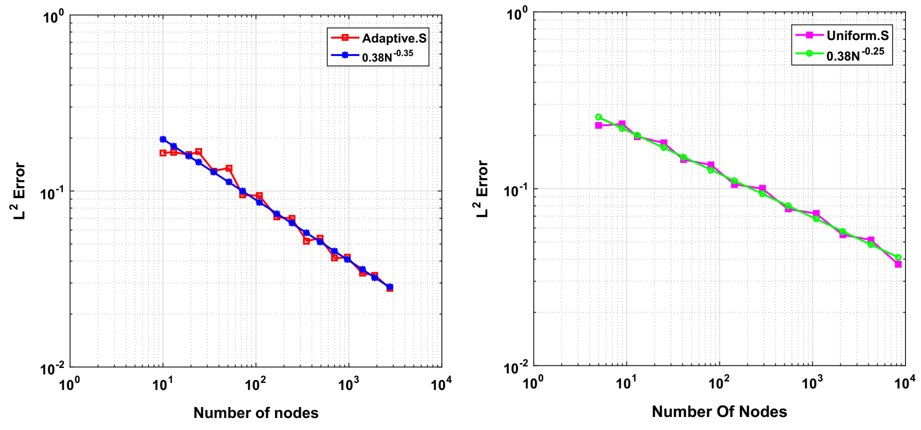

Fig. 3 Convergence rate by adaptive and uniform refined mesh with stabilization, Example 1

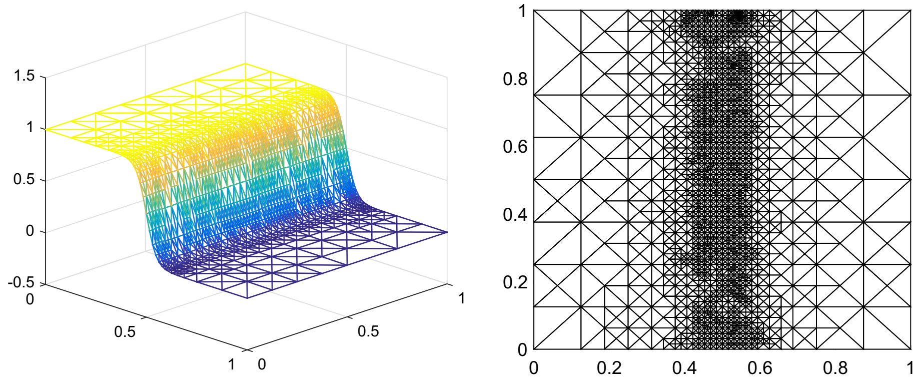

Fig. 4 EFG solution by adaptive technique (left) and the last adapted mesh (right), Example 2

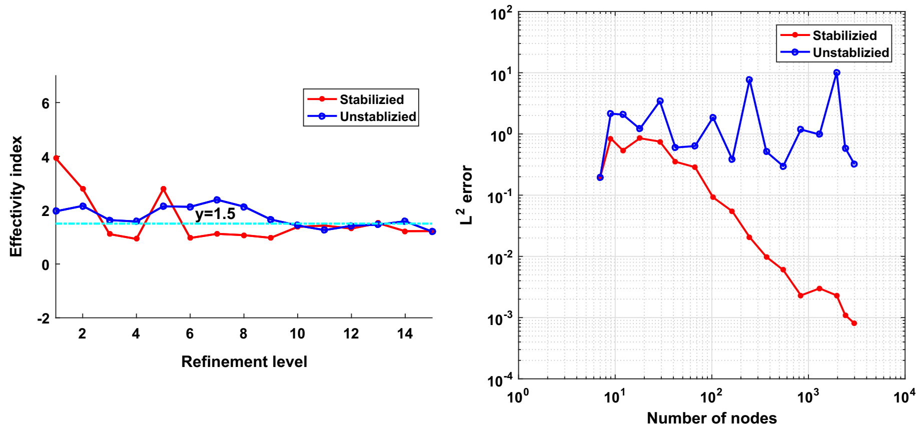

Fig. 5 The effectivity index as a function of refinement (left) and L^2 error (right), Example 2

4.2 Example 2

This example is taken from Burman and Hansbo (2004). Consider (1.1) with $\epsilon = 10^{-5}$, $\mathbf{b} = (1, 0)^{\mathrm{T}}$ and $c = 1$. The exact solution that has an internal boundary layer, is given by

$$u = \frac{1}{2}\left(1 - \tanh\left(\frac{x - 0.5}{0.05}\right)\right).$$

The corresponding f and Dirichlet boundary conditions are obtained by inserting the exact solution into Eq. (1.1). The numerical solution and the final mesh of this test at level 15 of adaptive refinements are shown in Fig. 4. Moreover, the effectivity indices and the obtained L^2 error at this refinement level have shown in Fig. 5. In this case, the effectivity indices for both stabilized and unstabilized are convergent to 1.5. However, L^2 errors are too different in with and without stabilization. In Fig. 6, convergence rates of the solutions obtained by adaptive and uniform refined mesh with stabilization are compared. As in the previous example, Tables 4, 5 and 6 are devoted to the comparison of adaptive and uniform refinements and effect of stabilization technique. Also, in Table 7 results of the current method and finite element method using edge stabilization (FEM-ES) are compared. Results of FEM-ES have

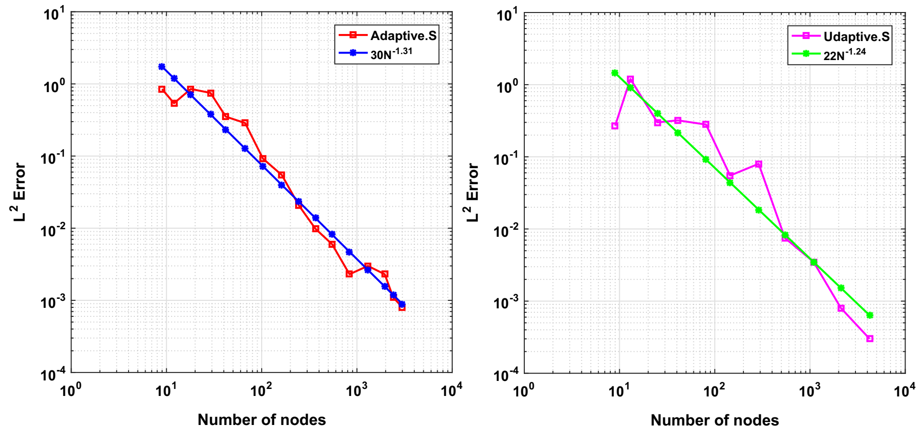

Fig. 6 Convergence rate by adaptive and uniform refined mesh with stabilization, Example 2

Table 4 Error of computed solution by uniform refinement, Example 2

Method	Level 4		Level 8		Level 12	
	N_t	$\|u - u_h\|_{L^2}$	N_t	$\|u - u_h\|_{L^2}$	N_t	$\|u - u_h\|_{L^2}$
Uniform	25	6.84e−1	289	9.02e−2	4225	4.09e−4
Uniform.S	25	2.56e−1	289	6.78e−2	4225	2.84e−4

Table 5 Error of computed solution on adaptive refined mesh, Example 2

Method	Level 6		Level 11		Level 16	
	N_t	$\|u - u_h\|_{L^2}$	N_t	$\|u - u_h\|_{L^2}$	N_t	$\|u - u_h\|_{L^2}$
Adaptive	32	5.08e−1	214	4.63e−1	1626	5.52e−1
Adaptive.S	36	3.04e−1	250	9.80e−3	1677	1.09e−3

Table 6 Relative error and CPU time of computed solution, Example 2

Adaptive.S				Uniform.S			
Level	N_t	$Re(u)$	Time	Level	N_t	$Re(u)$	Time
6	36	8.69e−2	2	4	25	7.64e−2	1
11	250	1.37e−3	6	8	289	5.77e−3	4
16	1677	5.73e−5	64	12	4225	6.33e−6	140

extracted from Burman and Hansbo (2004). It should be noted that, the reported CPU time is the total elapsed time, from level 1 to the last level, and it is not the CPU time of last level.

Table 7 Comparison of Adaptive.S and Uniform.S and FEM-ES, Example 2

Method	Adaptive.S		Uniform.S		FEM-ES	
	N_t	$\|u-u_h\|_{L^2}$	N_t	$\|u-u_h\|_{L^2}$	N_t	$\|u-u_h\|_{L^2}$
	1677	1.09e−3	2113	3.78e−3	1681	2.5e−3

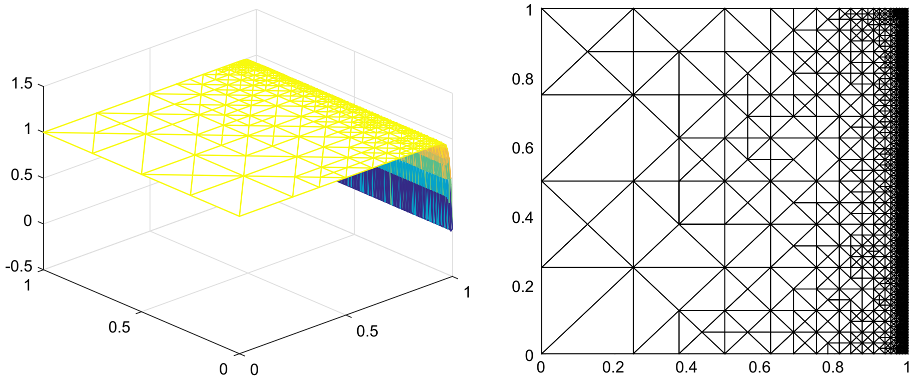

Fig. 7 EFG solution by adaptive technique (left) and corresponding adapted background mesh (right) on level 15, Example 3

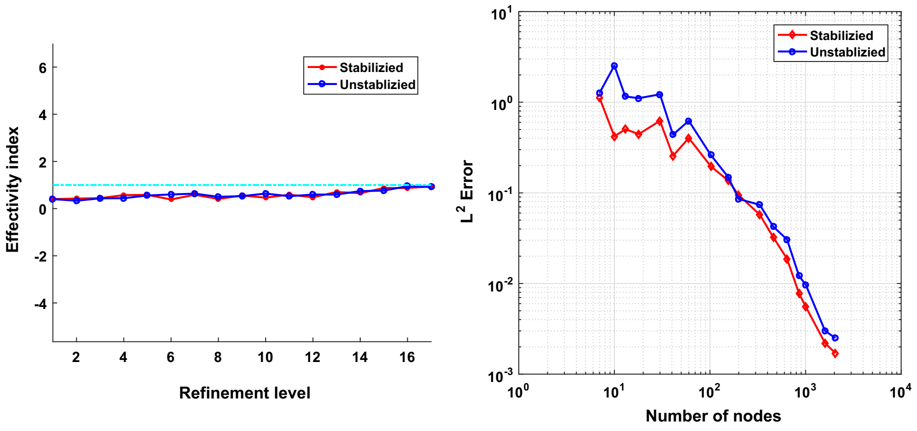

Fig. 8 The effectivity index as a function of refinement (left) and L^2 error (right), Example 3

4.3 Example 3

In this case $\epsilon = 10^{-2}$, $\mathbf{b} = (0, 1)^{\mathrm{T}}$ and $c = 0$. The exact solution is as follows:

$$u = \frac{\exp(\frac{1}{\epsilon}) - \exp(\frac{x}{\epsilon})}{\exp(\frac{1}{\epsilon}) - 1}.$$

The right-hand side f and Dirichlet boundary conditions can be computed from the exact solution. The results of this test are shown in Fig. 7. The effectivity indices and L^2 error are gathered in Fig. 8. Also, as Example 2, we have reported the effectivity indices for both

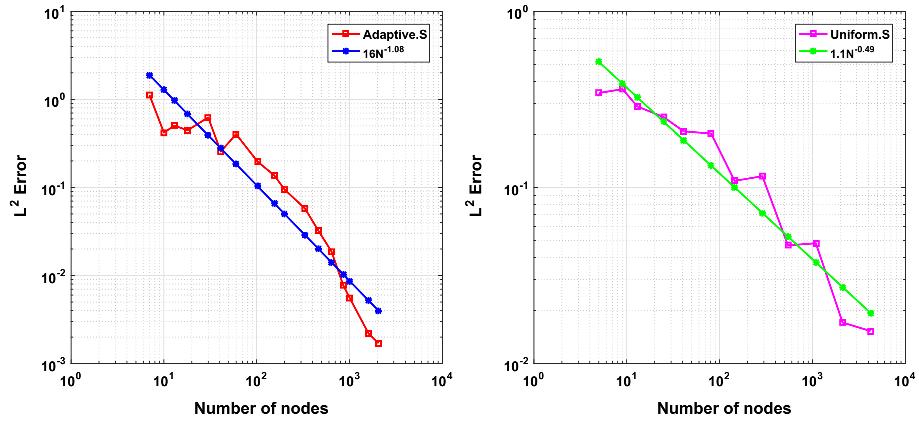

Fig. 9 Convergence rate by adaptive and uniform refined mesh with stabilization, Example 3

Table 8 Error of computed solution by uniform refinement, Example 3

Method	Level 4		Level 8		Level 12	
	N_t	$\|u-u_h\|_{L^2}$	N_t	$\|u-u_h\|_{L^2}$	N_t	$\|u-u_h\|_{L^2}$
Uniform	25	1.56e+0	289	1.12e−1	4225	1.03e−2
Uniform.S	25	5.29e−1	289	1.11e−1	4225	1.02e−2

Table 9 Error of computed solution on adaptive refined mesh, Example 3

Method	Level 5		Level 10		Level 15	
	N_t	$\|u-u_h\|_{L^2}$	N_t	$\|u-u_h\|_{L^2}$	N_t	$\|u-u_h\|_{L^2}$
Adaptive	25	1.32e+0	230	9.53e−2	1193	7.71e−3
Adaptive.S	30	1.24e+0	200	9.55e−2	1009	3.21e−3

Table 10 Relative error and CPU time of computed solution, Example 3

Adaptive.S				Uniform.S			
Level	N_t	$Re(u)$	Time	Level	N_t	$Re(u)$	Time
5	25	2.19e−1	1	4	25	1.18e−1	1
10	200	7.22e−3	5	8	289	7.02e−3	4
15	1009	1.08e−4	33	12	4225	2.38e−4	146

stabilized and unstabilized methods. However, here the results of using artificial diffusion (stabilized) and without this term (unstabilized) are not too different. The reason is that in this example ϵ can not be too small because even the exact solution is undefined for $\epsilon = 10^{-3}$ in the MATLAB double precision floating point system. Also, Fig. 9 shows convergence rate of presented methods in this example. Comparisons of adaptive and uniform refinements and effect of stabilization are presented in Tables 8, 9 and 10.

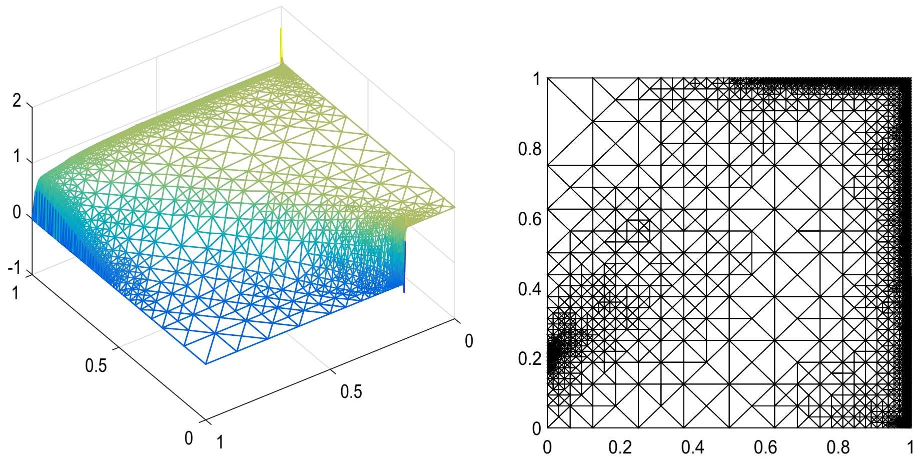

Fig. 10 The EFG solution by adaptive technique with $t = \frac{\pi}{4}$ (left) and the adapted mesh at level 20 (right) Example 4

4.4 Example 4

As the last example, consider the following example without exact solution, which involves discontinuous boundary conditions and causes not only a sharp layer, but also an internal sharp layer (Lin and Atluri 2000). In this case, $\epsilon = 10^{-3}$, the right hand side f and the reaction coffiecient c are zero. The vector $\mathbf{b} = [\cos t, \sin t]$ and Dirichlet boundary conditions are as follows:

$$g_D = \begin{cases} 1 & 0 \le x \le 1,\ y = 0 \\ 1 & x = 0,\ 0 \le y \le 0.2 \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

The numerical solution and last mesh obtained at level 20 of refinements for $t = \frac{\pi}{4}$ are shown in Fig. 10. A mild oscillation can be observed near the point (0.05,0.2). Results without using stabilization and adaptive technique are not acceptable and are not reported in the paper.

5 Conclusion

In this paper, an adaptive element free Galerkin (EFG) method is suggested for solving convection diffusion type equations. A posteriori error estimation based on the post-processed gradient is used. Also, to get the stability an artificial diffusion is considered. Here, it should be noted that based on the results reported in the tables, use of artificial diffusions has great effect on adaptive refinements, while this effect is insignificant for uniform refinements. To show efficiency of the proposed adaptive technique, some examples are solved numerically and the effectivity indices are computed in the examples with exact solution.

Acknowledgements The authors are very grateful to reviewers for carefully reading the paper and for their constructive comments and suggestions. This research was in part supported by a grant from IPM (No. 95650422).

References

Ainsworth M, Oden JT (1993) A unified approach to a posteriori error estimation using element residual methods. Numer Math 65:23–50

Bank RE, Weiser A (1985) Some a posteriori error estimotors for elliptic partial differential equations. Math Comp 44:283–301

Belytschko T, Krongauz Y, Fleming M, Organ D, Liu W (1996) Smoothing and accelerated computations in the element free Galerkin method. J Comput Appl Math 74:111–126

Belytschko T, Lu Y, Gu L (1994) Element free Galerkin methods. Int J Num Meth Eng 37:229–256

Burman E, Hansbo P (2004) Edge stabilization for Galerkin approximations of convection-diffusion-reaction problems. Comput. Methods Appl. Mech. Engrg. 193:1437–1453

Dehghan M, Abbaszadeh M (2018) Variational multiscale element-free Galerkin method combined with the moving Kriging interpolation for solving some partial differential equations with discontinuous solutions. Comp Appl Math 37:3869–3905

Dehghan M, Abbaszadeh M (2018) A reduced proper orthogonal decomposition (POD) element free Galerkin (POD-EFG) method to simulate two dimensional solute transport problems and error estimate. Appl Numer Math 126:92–112

Dehghan M, Narimani N (2018) An element free Galerkin meshless method for simulating the behavior of cancer cell invasion of surrounding tissue. Appl Math Model 59:500–513

Dolbow J, Belytschko T (1999) Numerical integration of the Galerkin weak form in meshfree methods. Comput Mech 23:219–230

Dolbow J, Belytschko T (1998) An introduction to programming the meshless element free Galerkin method. Arch Comput Methods Eng 5:207–241

Eriksson K, Estep D, Hansbo P, Johnson C (1995) Introduction to adaptive methods for differential equations. Acta Numer 105–158

Gockenbach MS (2006) Understanding and implementing the finite element method. SIAM

Gratsch T, Bathe K (2005) A posteriori error estimotion techniques in practical finite element analysis. Comput Struct 83:235–265

Jannesari Z, Tatari M (2017) A meshfree technique for numerical simulation of reaction–diffusion systems in developmental biology. Adv Appl Math Mech 9:1225–1249

Jannesari Z, Tatari M (2016) Element-free Galerkin method to the interface problems with application in electrostatic. Int J Numer Model 1089–1105

John V (2000) A numerical study of a posteriori error estimators for convection–diffusion equations. Comput Methods Appl Mech Eng 190:757–781

Lancaster P, Salkauskas K (1981) Surface generated by moving least squares methods. Math Comput 37:141–158

Larson MG, Bengzon F (2013) The finite element method. Theory, implementation and applications. Springer

Lin H, Atluri SN (2000) Meshless local-Petrov Galerkin (MLPG) methods for convection–diffusion problems. CMES 1:45–60

Liu GR (2003) Mesh free methods-moving beyond the finite element method. CRC Press LLC, London

Quarteroni A, Saleri F, Gervasio P (2014) Scientific Computing with MATLAB and Octave, 4th edn. Springer-Verlag, Berlin Heidelberg

Tang T, Trummer MR (1996) Boundary layer resolving pseudospectral methods for singular perturbation problems. SIAM J Sci Comput 17:430–438

Yaw L (2009) Introduction to moving least squares (MLS) shape functions. Walla Walla University, College Place

Zienkiewicz OC, Zhu JZ (1992) The superconvergent path recovery and a posteriori error estimotion. Part 1: the recovery technique. Int J Numer Methods Eng 33:1365–1382