Engineering Analysis with Boundary Elements 51 (2015) 52–63

Contents lists available at ScienceDirect

Engineering Analysis with Boundary Elements

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

Adaptive numerical integration in Element-Free Galerkin methods for elliptic boundary value problems

Grand Roman Joldes *, Adam Wittek, Karol Miller

Intelligent Systems for Medicine Laboratory, School of Mechanical and Chemical Engineering, The University of Western Australia, 35 Stirling Highway, Crawley/Perth WA 6009, Australia

ARTICLE INFO

Article history:
Received 25 March 2014
Received in revised form
22 July 2014
Accepted 19 October 2014
Available online 25 November 2014

Keywords:
Numerical integration
Element-Free Garlerkin
Irregular node distribution
Elliptic problems

ABSTRACT

In this paper we present a new numerical integration scheme for Element-Free Galerkin (EFG) methods used for solving elliptic problems. Integration points are distributed within the problem domain using an adaptive procedure, based on the characteristics of the shape functions. Existing numerical integration schemes for EFG methods do not offer any control over the integration accuracy. We devise a method of distributing the integration points which allows control over the integration accuracy for all elements of the stiffness matrix, while reducing the number of integration points required. The performance of the procedure is demonstrated on test problems in 1D and 2D.

1. Introduction

Elliptic boundary value problems (BVP) can describe physical phenomena from various fields: fluid dynamics, heat and electric conductivity, elasticity theory, geophysics, etc. Element-free Galerkin (EFG) methods offer an alternative solution method to well established solution algorithms (finite element method, finite difference method) for such problems. EFG methods do not require a mesh for shape function definition, and therefore are part of the larger class of mesh-free (MF) methods. Although EFG methods may offer important benefits over other solution methods (such as making possible the simulation of very large deformations in elasticity, eliminating the need for the construction of a high quality mesh), one important weakness that prevents EFG methods from being generally accepted is the difficulty in performing numerical integration.

While there are very well defined rules regarding numerical integration for the Finite Element Method (FEM), performing numerical integration for EFG methods is as much of an art as a science [1–3]. For FEM the integration cells are defined by the elements, and the number of integration points can be easily selected based on the characteristics of the shape functions, which are usually polynomials over the integration cells (elements). By knowing the degree of the shape function polynomials it is easy to select a Gaussian quadrature that can integrate exactly the shape functions, their derivatives or products of their derivatives. (Note: The above remark only applies for FEM using Updated Lagrangian formulation and polynomial shape functions. In the Total Lagrangian formulation the integrands are usually non-polynomials even when polynomial shape functions are used [4]). For MF methods the shape functions usually have a much larger support domain compared to FEM and they are not polynomials.

Gaussian quadrature over a background mesh is typically used for integration in EFG methods [1,3,5,6]. As stated in [3], there are two sources of error associated with these integration schemes: the shape functions in EFG methods are not polynomials and their local support may not align with the integration cells.

Specialized integration rules for circular support domains (or intersections of circular domains, also called lenses) are presented in the literature [7,8]. A simple adaptive integration procedure for such domains is proposed in [9]. In [3] the authors show that by aligning the integration cells with the support domains of the shape functions the integration accuracy can be increased. They call their method the bounding-box technique. Additional corrections may be required in the computation of stiffness matrices when quadrature rules based on intersections of shape functions support domains are used [10]. Also, specialized integration rules are restricted to specific types of support domains and the treatment of irregular boundaries is difficult.

Rather than integrating over the precise supports of the shape functions or develop more complicated quadrature rules, the use of nodal integration has been proposed as a possible solution. A comparison between nodal and Gaussian integration is presented in [11] for a few sample problems. Similar to under-integrated finite elements, MF methods based on nodal integration suffer from instability due to rank deficiency. A stabilized nodal

* Corresponding author. Tel.: +61 8 6488 3125.
E-mail address: grand.joldes@uwa.edu.au (G.R. Joldes).

http://dx.doi.org/10.1016/j.enganabound.2014.10.007
0955-7997/

integration algorithm for MF methods, using the strain smoothing stabilization technique, was proposed in [12].

There are some more exotic integration algorithms that have been proposed in the literature for MF methods, such as Quasi-Monte Carlo integration [13] or Genetic algorithms [14]. One common problem of these algorithms is the difficulty in estimating and controlling integration accuracy.

An adaptive numerical integration was proposed in [15] in the context of the Generalized Finite Element Method, where the DECUHR algorithm [16] was used to integrate the singular functions involved in the computation of the stiffness matrix elements. Hundreds of function evaluations per element are required to perform the integration [15].

In this paper we present a new adaptive quadrature algorithm. The new algorithm creates a distribution of integration points within the problem domain which allows the computation of integrals with controlled accuracy. The generated distribution of integration points is used for the computation of all integrals over the problem domain, including all elements of the stiffness matrix. The method imposes no constraints on the type of support domains which can be used.

The numerical solutions presented in this paper are obtained using the Element Free Galerkin (EFG) method [2] with Moving Least Squares (MLS) shape functions [1]. The results can be extended to most Galerkin MF methods and other types of shape functions.

The paper is organized as follows. In the following section we introduce the new integration procedure. Section 3 presents numerical experiments which highlight the properties of the method for one dimensional (1D) and two-dimensional (2D) elliptic boundary value problems. Section 4 contains a summary and some concluding remarks.

2. Adaptive numerical integration

The prevalent method for performing numerical integration in FEM and MF implementations is Gaussian quadrature [1]. The integration domain is split into a number of integration cells and then the quadrature is applied for each cell. It is a demonstrated fact that integration accuracy does not necessarily improve with the increase of number of Gauss points per integration cell [3]. A better approach is to increase the number of integration cells, which leads to convergence towards the real integral value. Unfortunately, a higher number of integration cells leads to proportionally higher computational time, and the number of integration cells required in order to obtain prescribed integration accuracy is still unknown.

To solve this problem we propose the use of an adaptive integration scheme. An adaptive quadrature algorithm is one that adapts itself to the behavior of the function being integrated. One example of such adaptive numerical quadrature is a process in which the integral of a function is approximated using static quadrature rules on adaptively refined subintervals of the integration domain, with the adaptive refinement being driven by the properties of the integrand. Such quadrature rules are well known in applied mathematics for integrating complicated or "badly-behaved" functions, including functions with singularities or discontinuities [17–20]. In [17] Rice estimates there are from 1 to 10 million adaptive quadrature algorithms that are potentially interesting and significantly different from one another.

2.1. Adaptive quadrature algorithms for univariate functions

As described in [20], a possible basic adaptive algorithm can be described by the following recursive procedure (for an univariate function f, an integration interval $[a\ b]$ and a desired accuracy τ):

Algorithm 1. :

```
Procedure integrate(f, [a b], τ)
  Q=quadrature(f, [a b]);
  ε=estimateError(f, [a b], I);
  if (ε > τ)
    m=(a+b)/2;
    Q=integrate(f, [a m], τ′)+integrate(f, [m b], τ′)
  endif
  return Q;
End
```

If the integrant is Riemann integrable and the error estimate is exact, the above algorithm will converge to the exact integral (as the tolerance $\tau \to 0$) [20]. The integration interval is recursively subdivided until the error estimate is smaller than the imposed tolerance for each sub-interval; the integration result is the sum of the integrals computed on each resulting sub-intervals.

The tolerance used for the recursive calls of the procedure must be adjusted to τ' to ensure that the required precision for the integral is obtained. One possibility is to use a proportional partitioning algorithm [17], which would lead to $\tau' = \tau/2$ for the adopted interval partitioning scheme (halving of the interval).

The main components of the above procedure are the *quadrature* rule used for integration and the procedure for estimating the integration error (*estimateError*). The quadrature rules we consider have the general form

$$\int_a^b f(x)dx = \sum_{i=1}^{n} w_i f(x_i) + E = Q_n[a,b] + E \tag{1}$$

where n is the number of points x_i in the interval $[a, b]$ used for function evaluation and w_i are their associated weights, Q_n is the approximation of the integral given by the quadrature rule and E is the integration error. The degree of precision of a quadrature formula (which we will simply call the degree of the quadrature) is N if and only if the error is zero for all polynomials of degree $k=0, 1,..N$ but is not zero for some polynomial of degree $N+1$.

The integration error for a quadrature of degree N is given by

$$E = k(b-a)^{N+2} f^{(N+1)}(\xi) \tag{2}$$

where k is a constant depending on the underlying quadrature rule and $f^{(N+1)}(x)$ is the $(N+1)$th derivative of the integrand at a point $\xi \in [a,b]$ [20].

We will use the notation $Q_n^m[a,b]$ for a quadrature applied on m equal size sub-intervals spanning interval $[a,b]$, therefore the *quadrature*$(f, [a,b])$ is $Q_n^1[a,b]$ in the above algorithm. Many other choices for the quadrature rule used are possible, such as Simpson's rule, Gauss-Legendre, Newton-Cotes, Clenshaw-Curtis or Gauss-Kronrod quadratures [17,20].

We want to use a quadrature that has a high degree for a given number of points, as the number of points increases exponentially for higher dimensions (curse of dimensionality). Therefore, the Gauss-Legendre quadrature will be used for univariate functions.

There are a variety of error estimators that can be used [20]. We select a linear error estimator based on the difference between two quadratures of the same degree (n) yet of different multiplicity (m):

$$\varepsilon \sim \left| Q_n^2[a,b] - Q_n^1[a,b] \right| \tag{3}$$

The use of such estimator in the decision of interval splitting is also referred to as "h-adaptivity".

The choice of the error estimator was dictated by the following considerations:

- The functions to be integrated are continuous and smooth (see following sections) – there is no need for more complicated estimators;
- The possibility to re-use information (quadrature value, weights and integration points) from one level to the next level of the recursive procedure;
- Easy extension to higher dimensions.

If the quadrature used has a degree N (can integrate exactly polynomials up to degree N) and the integrand f is sufficiently smooth, the value of the integral can be approximated by [20,21]:

$$I \cong Q_n^2[a,b] + \frac{\left|Q_n^2[a,b] - Q_n^1[a,b]\right|}{2^{N+1}-1} \tag{4}$$

Therefore, if $Q_n^2[a,b]$ is used as an approximation of the real integral, the error estimate given by (3) may be $2^{N+1}-1$ times larger than the actual integration error. Although this factor may seem large, in practice it is a good guard against bad estimates when the assumptions used for developing Eq. (4) are not met [20]. We also note that $Q_n^2[a,b]$, needed for the error estimation, is normally more accurate than $Q_n^1[a,b]$, and should therefore be used as an approximation of the integral.

The criterion used in the above algorithm to decide if further recursions are needed does not take into consideration the expected value of the integral. A better option is to use a criterion based on the relative error, such as [19]:

$$\varepsilon > \tau\left|Q_n^2[a,b]\right| \tag{5}$$

Such criterion may remove the need to adjust the tolerance used in the recursive calls of the integration procedure. Special care must be taken when integrating some functions which have zeroes within the interval, as the above criterion may always be met on at least one sub-interval and the recursive procedure may never end. This situation can be handled by imposing a limit on the number of recursion levels. A more detailed discussion on stopping criteria and computation of integrals to machine precision can be found in [19]. As shown in the following sections, our procedure requires integration of positive functions, and therefore does not have such problems.

Based on the above considerations, we define the following adaptive integration scheme for univariate functions:

Algorithm 2. :

```
Procedure [Q, xi, wi]=integrate(f, [a b], τ)
[Q1, xi, wi]=Q_n^1[a,b] // xi=integration points used,
  //wi=associated weights
Q=Q_n^2[a,b];
  if (|Q−Q1| > τ|Q|)
    m=(a+b)/2;
    [Q1, xi1, wi1]=integrate(f, [a m], τ);
    [Q2, xi2, wi2]=integrate(f, [m b], τ);
    Q=Q1+Q2; xi=concatenate(xi1, xi2); wi=concatenate(wi1, wi2);
  endif
  return Q, xi, wi;
End
```

One last aspect we want to highlight in this section is the concept of characteristic length, introduced in [17]. The characteristic length $\lambda(f)$ of the integrand $f(x)$ can be defined as an interval length such that the error bounds used in the adaptive procedure are valid for all intervals of length less than $\lambda(f)$. The characteristic length does not only depend on the integrand, but also on the adaptive algorithm used (quadrature, error estimation procedure). Calculating the characteristic length may be difficult in practice. Nevertheless, its consideration has an important practical implication that the integration interval for an adaptive procedure cannot be indiscriminately large, as it can lead to the failure of the adaptive procedure (incorrect results due to early termination of the recursive calls). Therefore, large integration intervals should be divided into smaller intervals and the adaptive procedure should be applied to each sub-interval (composite integration).

2.2. Adaptive quadrature algorithms in multiple dimensions

The direct extension of quadrature rules in multiple dimensions is through the use of tensor product of one-dimensional quadrature rules. We will restrict our discussion to 2D, although the results can be easily extended to higher dimensions. We will also consider that the same one-dimensional quadrature rule is used in each dimension (as the rate of convergence for a multi-dimensional tensor product rule is equal to the smallest rate of convergence of the one-dimensional quadratures used).

Tensor product rules are usually described over a square region (integration cell). This does not restrict their use; any quadrilateral or triangle can be obtained from a square through a coordinate transformation.

Therefore, for a one-dimensional quadrature rule with n points, of degree N, the following 2D integration formula can be derived:

$$\int_{-h/2}^{h/2}\int_{-h/2}^{h/2} f(x,y)dydx = \sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j f(x_i,y_j) + E_{2D} = Q_{n,n} + E_{2D} \tag{6}$$

The integration error can be obtained by using the one-dimensional results, as follows:

$$\int_{-\frac{h}{2}}^{\frac{h}{2}}\int_{-\frac{h}{2}}^{\frac{h}{2}} f(x,y)dydx = \int_{-\frac{h}{2}}^{\frac{h}{2}}\left(\sum_{j=1}^{n} w_j f\left(x,y_j\right) + E_j\right)dx =$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j f(x_i,y_j) + \sum_{j=1}^{n} w_j E_{ij} + \int_{-\frac{h}{2}}^{\frac{h}{2}} E_j dx \tag{7}$$

therefore

$$E_{2D} = \sum_{j=1}^{n} w_j E_{ij} + \int_{-\frac{h}{2}}^{\frac{h}{2}} E_j dx \tag{8}$$

with

$$E_j = kh^{N+2} f_{,y}^{(N+1)}(x,\xi_j)$$
$$E_{ij} = kh^{N+2} f_{,x}^{(N+1)}(\xi_{ij},y_j) \tag{9}$$

Under the assumption that the integrand is sufficiently smooth that the partial derivatives in Eq. (9) are approximately equal to a constant D, the 2D integration error becomes:

$$E_{2D} = \sum_{j=1}^{n} w_j kh^{N+2} D + \int_{-\frac{h}{2}}^{\frac{h}{2}} kh^{N+2} D dx = 2kh^{N+3} D \tag{10}$$

where we used the fact that the quadrature is exact for any constant:

$$\int_{-\frac{h}{2}}^{\frac{h}{2}} 1 dx = \sum_{j=1}^{n} w_j = h \tag{11}$$

The square with the edge length equal h can be divided into 4 smaller squares using its centroid and the middle of its edges. Using the notation $Q_{n,n}^4$ for the sum of integrals computed using the 2D quadrature rule on each of the obtained squares, and considering the relation between h and the error given by Eq. (10), an equation similar to (4) can be obtained for the 2D case:

$$I \cong Q_{n,n}^4 + \frac{\left|Q_{n,n}^4 - Q_{n,n}^1\right|}{2^{N+1}-1} \tag{12}$$

Therefore, an estimation of the error is given by:

$$\varepsilon \sim \left| Q^4_{n,n} - Q^1_{n,n} \right| \tag{13}$$

Quadrature rules in 2D defined as tensor product of a one-dimensional quadrature rule of degree N can integrate exactly all polynomials including the following monomial terms:

$$m(x,y) = cx^i y^j,\ i = 0..N,\ j = 0..N \tag{14}$$

In many applications the total order of the polynomials that need to be integrated is bounded (complete polynomials, having total order $\max(i+j) \le p$). Special rules have been developed for integrating such polynomials over triangular or rectangular domains [22–28]. The advantage of these quadrature rules is the reduction in the number of points required for exact integration of a complete polynomial (by taking advantage of the limited number of monomials that need to be integrated), as compared to a tensor product rule. Nevertheless, the integration error would be higher for functions which cannot be well approximated by a complete polynomial; therefore using such quadrature rules may not lead to a decrease in the total number of integration points required to meet a given integration accuracy.

2.3. Adaptive quadrature algorithm for elliptic boundary value problems

Finding the solution of elliptic boundary value problems using mesh-free methods usually involves the computation and assembly of a global stiffness matrix. As shown in the following sections, stiffness matrices involve the computation and assembly of local stiffness matrices defined based on the derivatives of shape functions, such as:

$$K_{ij} = \int_a^b \varnothing^i_{,x}(x)\varnothing^j_{,x}(x)dx \tag{15}$$

where i and j are indexes of the degrees of freedom of the discretization.

Shape functions (and their derivatives) have a compact support domain, so that only a limited number of nodes from the discretization influence their behavior. Over a given integration cell used for numerical integration only a small number of shape functions will have nonzero values. At the same time, for each integration point belonging to a cell, the number and indexes of the nonzero shape functions may be different. Since the nonzero shape functions spanning an integration cell are combined as shown in Eq. (15) to define the elements of the local stiffness matrix, many different functions need to be integrated over each integration cell. One may compute each of these integrals using an adaptive procedure; however, such approach is computationally expensive, as obtaining the shape functions at each integration point requires the solution of a minimization problem (as shown in the following section).

We propose a different approach to integration of the stiffness matrix elements. Our approach is based on the fact that the integration accuracy is related to the "smoothness" of the integrand. The "smoothness" of a function is defined by the degree of the polynomial that can approximate it accurately over the integration cell. The smoothness is expected to increase as the dimension of the integration cell decreases, which forms the basis of the h-adaptive integration procedure. If the adaptive integration method is able to accurately integrate a given integrand, it should also accurately integrate integrands which are "smoother". Using this observation, our proposed integration method has the following steps:

- Define an integrand which is less smooth than all the integrands used for stiffness matrix computation over the integration cell;
- Apply the adaptive integration procedure to integrate the above integrand to user-defined accuracy; this will generate a collection of integration points and weights over the integration cell;
- Use the integration points and weights generated above to compute each component of the stiffness matrix for the integration cell.

We propose the definition of the less smooth integrand, which drives the creation of the integration points and weights associated with an integration cell, using the following heuristics:

- If a function f can be approximated by a polynomial of degree n and a function g can be approximated by a polynomial of degree m, then their product fg can be approximated by a polynomial p of degree $m+n$. Because $m+n < \max(2m, 2n)$, either f^2 or g^2 requires for approximation a polynomial of degree higher than the degree of polynomial p and is therefore less smooth than fg.
- The degree of a polynomial approximation for f^2+g^2 is max $(2m, 2n)$ (unless the two polynomials have the same degree and the coefficients of the highest monomial in their polynomial approximation cancel each-other).

Considering the above, we define the function that drives the creation of the integration points and weights associated to an integration cell, to be used for the computation of stiffness matrices according to Eq. (15), as:

$$f(x) = \sum_i \left(\varnothing^i_{,x}\right)^2 \tag{16}$$

We note the fact that for a given integration point only a few shape functions (and their derivatives) are nonzero and they are all computed at the same time using the Moving Least Squares (MLS) minimization procedure. Therefore, the values for the function in Eq. (16) can be easily calculated at each integration point.

The above procedure produces a distribution of integration points which, for a given required accuracy, is defined only by the shape functions, and therefore dependent only on the discretization of the domain (number and position of nodes). The effectiveness of the proposed procedure will be demonstrated in the following section.

3. Numerical experiments

3.1. Moving least squares shape functions

The moving least squares (MLS) approximation is the method of choice for Mesh-Free methods [1,29], mainly because of the smoothness and continuity of the approximation field it generates. The MLS method was proposed by Lancaster and Salkauskas [30] for smoothing and interpolating data. Given n data points (nodes) located at positions $\mathbf{x}_j$, $j=1..\ n$, we can obtain a function $u^h(\mathbf{x})$ that approximates the given scalar values u^j at points $\mathbf{x}_j$ by minimizing the error functional

$$J(\mathbf{x}) = \sum_{j=1}^{n} [(u^h(\mathbf{x}_j) - u^j)^2\ w(\|\mathbf{x}-\mathbf{x}_j\|)] \tag{17}$$

where the error between the defined function and the given scalar values is weighted using the weighting function w based on the Euclidian distances between the evaluation point and the positions of the nodes. We use $\|.\|$ as the notation for Euclidian distance.

The function $u^h(\mathbf{x})$ is chosen as a polynomial

$$u^h(\mathbf{x}) = \sum_{i=1}^{m} p_i(\mathbf{x})a_i(\mathbf{x}) = \mathbf{p}^T(\mathbf{x})\mathbf{a}(\mathbf{x}) \tag{18}$$

where m is the number of terms in the basis $\mathbf{p}(\mathbf{x})$, and $a_i(\mathbf{x})$ are the coefficients that depend on spatial coordinates $\mathbf{x}$ (due to the weight functions which depend on $\mathbf{x}$). For example, commonly used basis and the corresponding coefficients in 2D are:

– linear basis:

$$\mathbf{p}^T(\mathbf{x}) = [1, x, y], \quad \mathbf{a}^T(\mathbf{x}) = [a_1, a_x, a_y] \tag{19}$$

– quadratic basis:

$$\mathbf{p}^T(\mathbf{x}) = [1, x, y, x^2, xy, y^2], \quad \mathbf{a}^T(\mathbf{x}) = [a_1, a_x, a_y, a_{x^2}, a_{xy}, a_{y^2}] \tag{20}$$

The coefficients $a_i(\mathbf{x})$ are obtained by minimizing the weighted least-square error functional $J(\mathbf{x})$. The approximation function can then be expressed as:

$$u^h(\mathbf{x}) = \sum_{j=1}^{n} \varnothing^j(\mathbf{x}) u^j \tag{21}$$

where $\varnothing^j(\mathbf{x})$ are the shape functions.

The weight function plays important roles in the formulation of MLS approximation: it provides weightings for the residuals at different nodes within the (compact) support domain and it ensures that nodes enter and leave the influence domain smoothly so that the shape functions satisfy the compatibility condition and the approximation is globally continuous. We use a quartic spline weight function to ensure the approximation has second order continuity, with circular support domains. The quartic spline weight function is defined based on the normalized distance

$$s = \frac{\|\mathbf{x} - \mathbf{x}_j\|}{R_j} \tag{22}$$

(where R_j is the radius of the influence domain of the node $\mathbf{x}_j$) as:

$$w(\|\mathbf{x} - \mathbf{x}_j\|) = w(s) = \begin{cases} 1 - 6s^2 + 8s^3 - 3s^4 & s \le 1 \\ 0 & s > 1 \end{cases}. \tag{23}$$

We will use linear basis for our MF method. The radius of influence for each node $\mathbf{x}_j$ is selected as:

$$R_j = M \cdot \max_{i \in N_j} (\|\mathbf{x}_i - \mathbf{x}_j\|) \tag{24}$$

where M is a scaling parameter greater than 1 and N_j is the set of indexes of neighboring nodes surrounding node $\mathbf{x}_j$ (therefore R_j is based on a measure of nodal spacing). This selection of the radius of influence will ensure that the moment matrix is not singular and the shape functions can be computed even for irregular node placement.

3.2. Integration accuracy

The elements of the stiffness matrices K_{ij} in Eq. (15) can be computed accurately by using a very large number of integration cells and Gauss points. The same integrals can be computed using the proposed adaptive integration scheme with different values for the imposed accuracy τ – we will use the notation K_{ij}^{*} for the obtained values. Because the diagonal terms of matrix K are always non-zero, while many off-diagonal elements are zero, we define the relative integration error for each element of K as:

$$E_{ij}^{rel} = \left| \frac{K_{ij}^{*} - K_{ij}}{K_{ii}} \right| = \left| \frac{dK_{ij}}{K_{ii}} \right| \tag{25}$$

The above error definition can be used to evaluate how well the parameter τ controls the integration accuracy for stiffness matrices.

3.3. Solution errors and convergence

The error in the solution of a problem with known analytical solution $u(\mathbf{x})$ is evaluated using the following relative error norms:

$$RENu = \frac{\|u - u^h\|_{L_2}}{\|u\|_{L_2}} = \frac{\left(\int_\Omega (u - u^h)^2 d\Omega\right)^{1/2}}{\left(\int_\Omega u^2 d\Omega\right)^{1/2}}. \tag{26}$$

$$RENux = \frac{\|u_{,x} - u_{,x}^h\|_{L_2}}{\|u_{,x}\|_{L_2}} = \frac{\left(\int_\Omega \left(u_{,x} - u_{,x}^h\right)^2 d\Omega\right)^{1/2}}{\left(\int_\Omega u_{,x}^2 d\Omega\right)^{1/2}}. \tag{27}$$

The integrals in the above relations are computed by dividing the integration domain into a large number of integration cells, and using a high order Gaussian quadrature for each integration cell.

Convergence studies are performed by studying the variation of the above error norms with the spacing between discretization points.

3.4. Number of integration points required

Apart from accuracy, we also study the efficiency of the integration scheme. We consider that an integration scheme is more efficient if it requires a lower number of integration points for a given accuracy.

For a given starting integration domain, if we consider that the procedure stops after n recursivity levels ($n=1$ means no subdivision of the integration domain required for integration, although a subdivision is still required for error estimation), at each level the domain is divided into s sub-domains, and N integration points are used for each sub-domain, the total number of evaluations of the function guiding the integration procedure is given by:

$$Neval = N + Ns + \ldots + Ns^n = N\frac{s^{n+1} - 1}{s - 1} \tag{28}$$

The number of sub-domains s depends on the integration domains used: in 1D the domains are intervals which can be divided into 2, so $s=2$. In 2D the domains can be quadrilaterals or triangles, and each can be subdivided into 4 ($s=4$) and in 3D the domains may be hexahedrons or tetrahedrons, which can be divided into 8 or 5 sub-domains ($s=8$ and $s=5$ respectively). Other subdivision schemes are possible.

The number of integration points per integration domain can vary, depending on the quadrature used. The number of recursivity levels required is related to the desired accuracy τ.

3.5. One-dimensional example

We consider the same 1D boundary value problem (BVP) as analyzed in [3]:

$$u_{,xx}(x) + g(x) = 0 \quad \text{in } \Omega = (0, 1), \tag{29}$$

in which comma denotes partial derivatives with respect to the following subscripts, with boundary conditions:

$$u(0) = u_0 \tag{30}$$

$$u_{,x}(1) = d_1 \tag{31}$$

3.5.1. Discretization of the BVP equation

The weak form for Eq. (29) can be written, using test functions v, as:

$$\int_0^1 u_{,xx}(x) v(x) dx + \int_0^1 g(x) v(x) dx = 0. \tag{32}$$

By using integration by parts of the first integral and re-arranging the terms, Eq. (32) can be re-written as:

$$\int_0^1 u_{,x}(x)v_{,x}(x)dx + u_{,x}(0)v(0) = u_{,x}(1)v(1) + \int_0^1 g(x)v(x)dx. \tag{33}$$

Eq. (33) is discretized with an approximation of the solution constructed using the MF shape functions as a basis defined over n points in the problem domain – Eq. (21). In the Galerkin method the shape functions are used as test functions. Therefore, Eq. (33) can be re-written (for each test function) as:

$$\sum_{k=1}^{n} u^k \int_0^1 \phi^k_{,x}(x)\phi^j_{,x}(x)dx + u_{,x}(0)\phi^j(0) = u_{,x}(1)\phi^j(1) + \int_0^1 g(x)\phi^j(x)dx, \quad j = 1..n \tag{34}$$

We notice that the natural boundary conditions (31) can be substituted directly in (34) while the essential boundary conditions (30) have to be enforced. A usual method of enforcing the essential boundary conditions is to require that all shape functions used in (34) are 0 on the essential boundaries (have Kronecker delta property). Such method does not increase the number of variables (as the term containing $u_{,x}(0)$ disappears) but it normally requires a modification of the MF shape functions. We avoid such modification by adding the equations describing the essential boundary conditions to the system of equations, while considering $u_{,x}(0) = d_0$ a secondary variable. Therefore, the final discretized system of equations becomes:

$$\begin{cases} \sum_{k=1}^{n} u^k \int_0^1 \phi^k_{,x}(x)\phi^j_{,x}(x)dx + d_0\phi^j(0) = d_1\phi^j(1) + \int_0^1 g(x)\phi^j(x)dx, \quad j = 1..n \\ \sum_{k=1}^{n} \phi^k(0)u^k = u_0 \end{cases} \tag{35}$$

Therefore, the integrals that need to be computed using a numerical quadrature are based on the derivatives of the shape functions

$$I_{kj} = \int_0^1 \phi^k_{,x}(x)\phi^j_{,x}(x)\,dx \tag{36}$$

as well as on the shape functions and function g:

$$F_j = \int_0^1 g(x)\phi^j(x)\,dx \tag{37}$$

All these integrals need to be computed accurately in order to obtain an accurate numerical solution.

3.5.2. Numerical integration procedure

We consider the non-uniform nodal distribution used in [3], with nodal spacing of 0.1 everywhere in the problem domain except the sub-domain (0.4, 0.6) where the nodal spacing is 0.01. The MLS shape functions constructed using a scaling factor $M=2$ for defining the radius of influence are presented in Fig. 1 for sub-domain (0.5, 0.8).

We define the function used to guide the subdivision of integration cells as:

$$f(x) = \begin{bmatrix} \sum_{k=1}^{n} \left(\phi^k_{,x}(x)\right)^2 \\ \sum_{k=1}^{n} \left(\phi^k(x)\right)^2 \end{bmatrix} \tag{38}$$

The function in (38) has multiple outputs, therefore the relative accuracy is computed and verified independently for each output inside the adaptive integration procedure. The outputs of this function are presented in Fig. 2.

Function g does not appear in the definition of function f. Because function g is the second order derivative of the solution, it should be smoother than the shape functions used for constructing the solution.

The proposed procedure for performing numerical integration using the adaptive integration scheme is described by the following algorithm:

Algorithm 3. :

Step 1: Pre-computation of integration points xi and associated weights wi:

- For each initial integration interval [ak bk]
  - Apply the recursive integration procedure (Algorithm 2) for the function used to drive the subdivision, Eq. (38): [Q, xik, wik]=integrate(f, [ak bk], τ);
  - Save the integration points and weights: xi=concatenate(xi, xik); wi=concatenate(wi, wik);
  - Save shape functions and their derivatives computed at integration points during the evaluation of function f(x) (to avoid re-computing them in the next step).

Step 2: Perform numerical integration

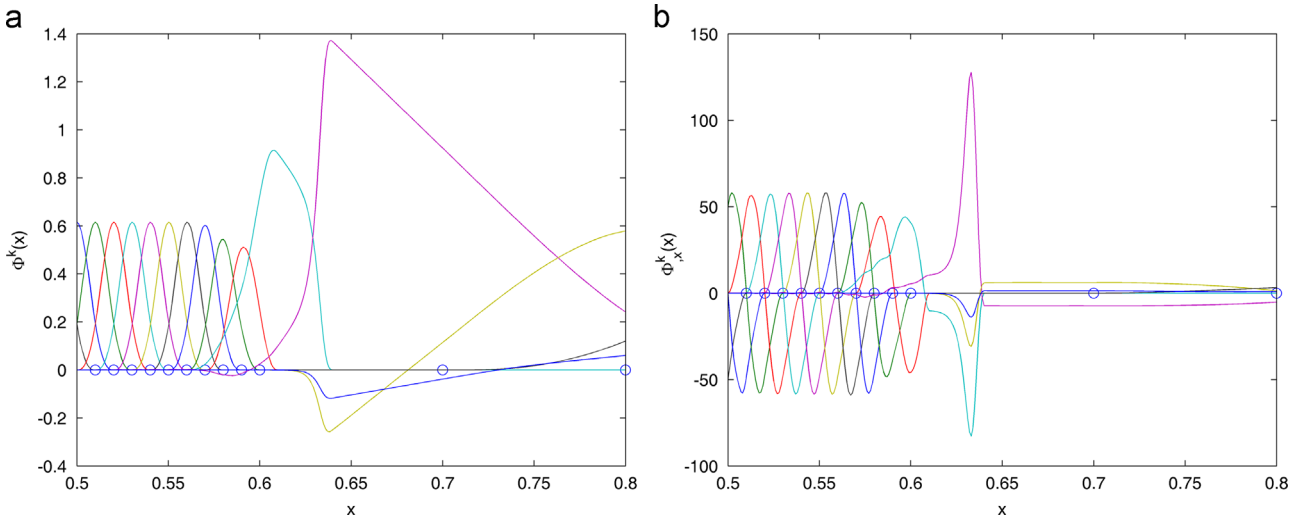

Fig. 1. One dimensional MLS shape functions (a) and their derivatives (b) for sub-domain (0.5, 0.8). We used a scaling factor $M=2$ for defining the radius of influence and linear basis functions. Each curve represents the MLS shape function (or its derivative) corresponding to one of the nodes (represented as cirles).

– Compute the integrals (Eqs. (36), (37)) over the domain by using in Eq. (1) the integration points and weights determined in Step 1.

3.5.3. Performance assessment

To assess the performance of the proposed integration scheme, we study the following problem taken from [3]:

$$g(x)=6x+\left(\frac{2}{\alpha^2}-\left(\frac{1-2x}{\alpha^2}\right)^2\right)\exp\left(-\left(\frac{x-a}{\alpha}\right)^2\right) \tag{39}$$

and

$$u(0)=\exp\left(-\frac{a^2}{\alpha^2}\right) \tag{40}$$

$$u_{,x}(1)=-3-2\left(\frac{1-a}{\alpha^2}\right)\exp\left(-\left(\frac{1-a}{\alpha}\right)^2\right) \tag{41}$$

with the solution

$$u(x)=-x^3+\exp\left(-\left(\frac{x-a}{\alpha}\right)^2\right). \tag{42}$$

By choosing the parameters $a=0.5$ and $\alpha=0.05$ the solution has a local character at $x=0.5$. The solution obtained using 5-point Gaussian quadrature, with the integration cells being the intervals between points, is presented in Fig. 3. These results match very well the results presented in [3]. The total number of integration points used is 140.

The solution obtained using the proposed adaptive cell subdivision scheme, starting from the same initial integration cells, combined with 3-point Gaussian quadrature is presented in Fig. 4. We imposed a relative accuracy $\tau=0.01$. The total number of integration points used is 144.

The adaptive integration scheme improved the relative error norms *RENu* from 0.0715 to 0.0088 and *RENux* from 0.0909 to 0.0363, while using almost the same number of integration points. The relative error norms were computed using a 10-point Gaussian quadrature and 1000 equal size integration cells.

The maximum and mean relative integration errors for the elements of the stiffness matrix versus the number of integration points are presented in Fig. 5. The curve marked "Gn; $n=2,3,\ldots$" is obtained using n-point Gaussian quadrature for each integration cell, with the integration cells being the intervals between points; n is increased at each step. The curve marked "G10; decrease h" is obtained by doubling the number of integration cells at each step (each integration cell used in one step is divided into two equal integration cells for the following step). The other curves are obtained by using the presented adaptive integration procedure with different numbers of Gauss points per integration cell; the horizontal lines indicate the levels of accuracy τ used for the adaptive procedure at each step.

From the results in Fig. 5 we can draw the following conclusions:

- There is a good correlation between the imposed accuracy τ and the obtained maximum error. **Note:** because we use a local relative error estimate (see Algorithm 2), the global relative error is not guaranteed to be less than the imposed accuracy [20].
- The proposed adaptive integration procedure can achieve significant increases in accuracy for a given number of integration points. For example, for approximately 1000 integration points, the proposed adaptive integration procedure using G10 quadrature increases accuracy by 8 digits as compared to a uniform increase in number of integration cells using the same quadrature (curve 2).
- The proposed adaptive integration procedure can integrate all elements of the stiffness matrix with high accuracy.
- For high integration accuracy, it is better to use a high order Gaussian quadrature (G10) with the adaptive procedure, as it leads to a lower total number of integration points.

A convergence study for the above problem is presented in Fig. 6. Refined discretizations are obtained by adding a node in between each two existing nodes. The grid spacing parameter h is taken as the maximum distance between any two adjacent nodes of the discretization.

The adaptive integration method ensures much better accuracy and increased convergence rates compared with the 5-point Gaussian quadrature while requiring slightly less integration points when the imposed accuracy is $\tau=10^{-3}$. When the integration accuracy is increased to $\tau=10^{-6}$, the number of integration points required increases while there is little change in the relative error norms. This illustrates the fact that, for any accuracy greater than $\tau=10^{-3}$, discretization error dominates the solution and any

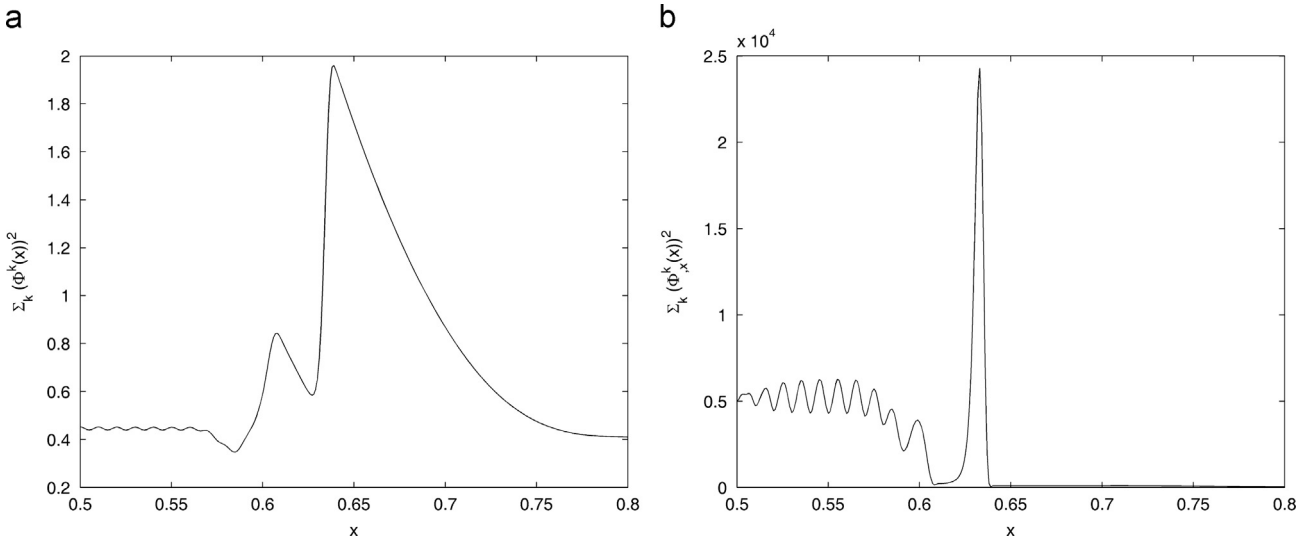

Fig. 2. The outputs of the function used to guide the adaptive cell division: (a) sum of shape functions (b) sum of shape functions' derivatives.

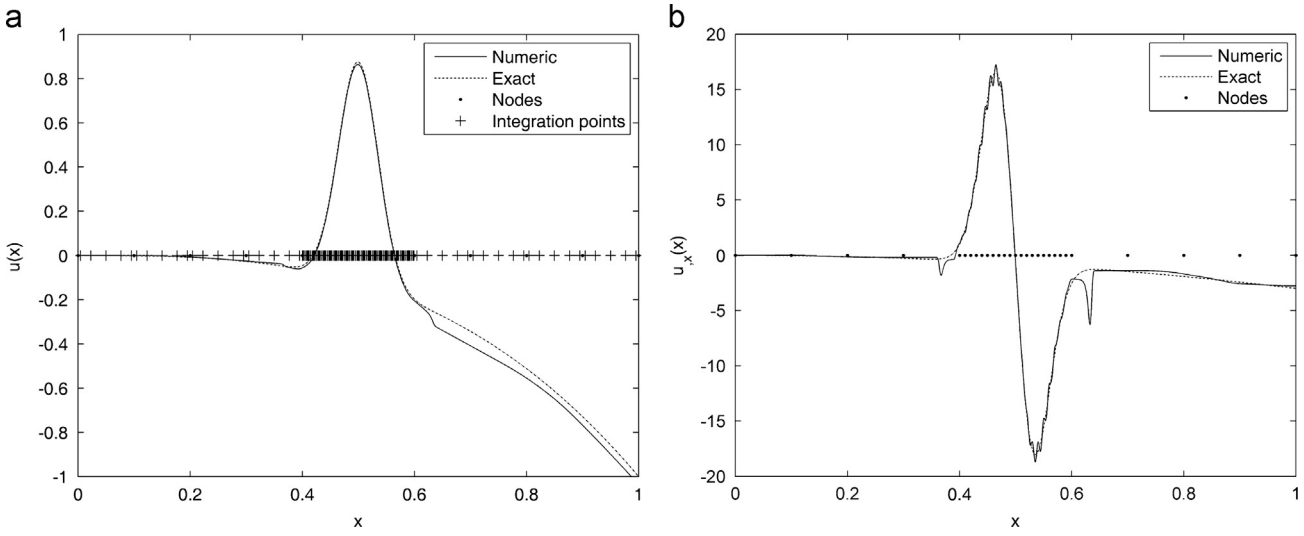

Fig. 3. Solution obtained using 5-point Gaussian quadrature. (a) The function $u(x)$ and (b) its gradient $u_{,x}(x)$.

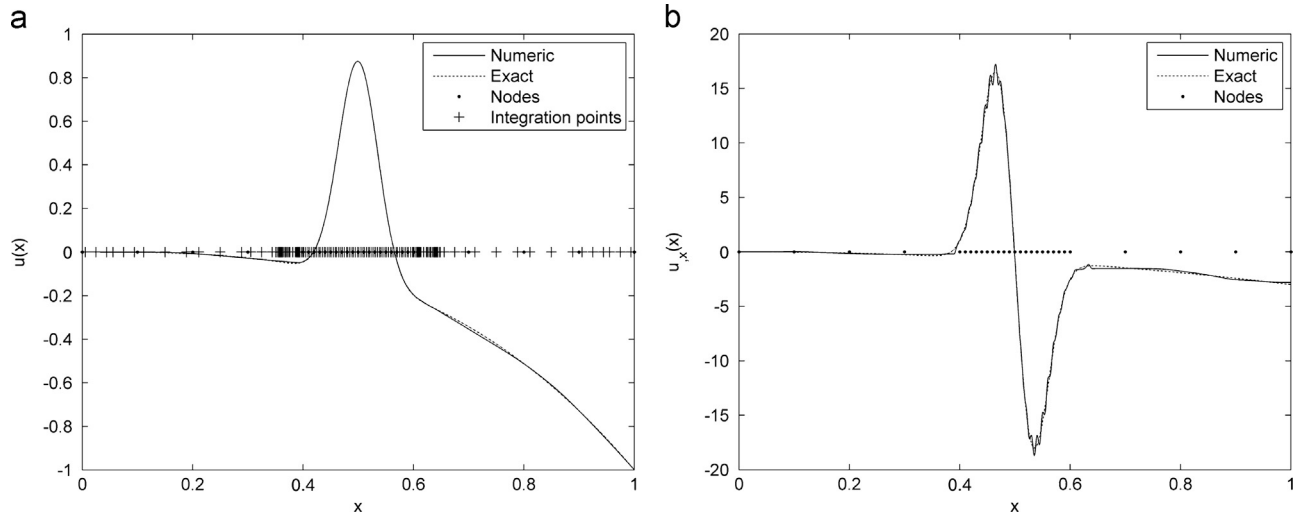

Fig. 4. Solution obtained using the adaptive interval subdivision scheme combined with 3-point Gaussian quadrature, for an accuracy $\tau=0.01$. (a) The function $u(x)$ and (b) its gradient $u_{,x}(x)$.

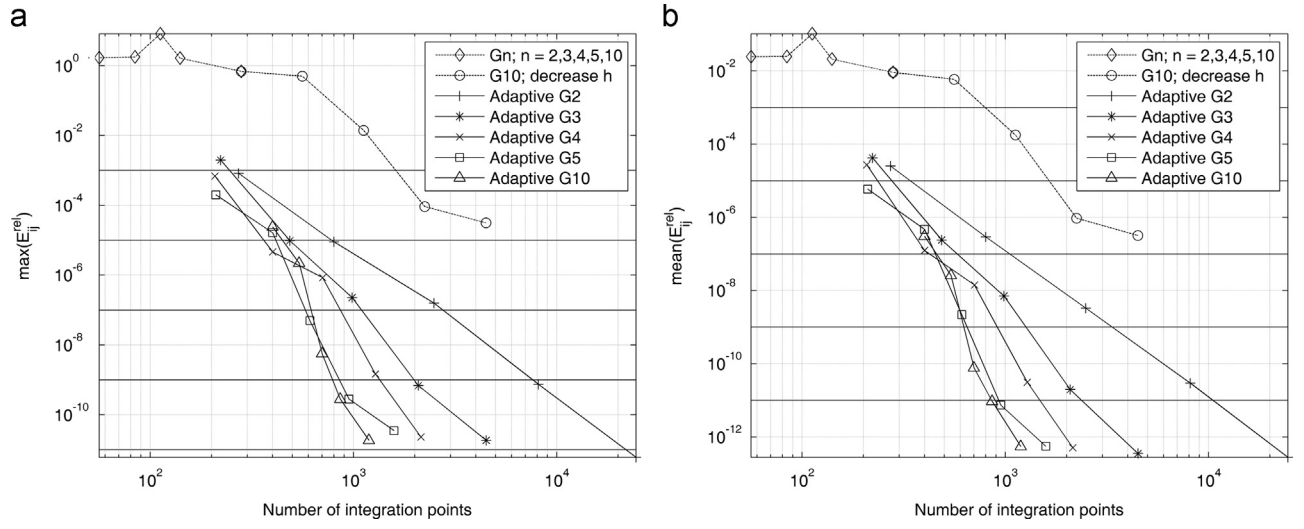

Fig. 5. Maximum (a) and mean (b) relative integration errors for the elements of the stiffness matrix versus the number of integration points. Accuracy of the proposed adaptive procedure is increased by changing the value of τ. The first curve shows the change in accuracy with the increase of Gauss points per integration cell. The second curve shows the change in accuracy with the decrease of integration cell size h. The horizontal lines indicate the levels of accuracy τ used for the adaptive procedure at each step.

further increase in integration accuracy has little influence on the solution accuracy.

3.6. Two-dimensional example

As an example of 2D BVP we analyze the Poisson equation:

$$u_{,xx}(\mathbf{x})+u_{,yy}(\mathbf{x})+g(\mathbf{x})=0 \quad \text{in } \Omega, \tag{43}$$

with the Dirichlet (essential) boundary condition

$$u(\mathbf{x})=\overline{u}(\mathbf{x}) \quad \text{on } \Gamma. \tag{44}$$

3.6.1. Discretization of the BVP equation

The weak form for Eq. (43) can be written, using test functions v, as:

$$\int_\Omega \Delta u v d\Omega+\int_\Omega g v d\Omega=0 \tag{45}$$

where $\Delta=\nabla^2$ is the Laplace operator. By using Green's first identity for the first integral and re-arranging the terms, Eq. (45) can be re-written as:

$$\int_\Omega \nabla u\cdot\nabla v d\Omega-\int_\Gamma(\nabla u\cdot\mathbf{n})v dS=\int_\Omega g v d\Omega \tag{46}$$

where $\mathbf{n}$ is the outward pointing unit normal of boundary element dS.

Eq. (46) is discretized with an approximation of the solution constructed using the MF shape functions as a basis defined over n points in the problem domain. By using the shape functions as test functions, Eq. (46) can be re-written (for each test function) as:

$$\sum_{k=1}^{n} u^k\int_\Omega \nabla\phi^k\cdot\nabla\phi^j d\Omega-\int_\Gamma(\nabla u\cdot\mathbf{n})\phi^j dS=\int_\Omega g\phi^j d\Omega, \quad j=1..n \tag{47}$$

Because the MF shape functions for the interior nodes do not vanish on the boundary, the second integral in Eq. (47) is not zero. We consider

$$q=\nabla u\cdot\mathbf{n} \tag{48}$$

as a secondary variable and discretize it on the boundary as:

$$q^h=\sum_{i=1}^{n_b}\psi^i(\mathbf{x})q^i \tag{49}$$

where n_b is the number of nodes on the boundary, q^i is the value of q at node i and ψ^i are linear shape functions defined for the boundary segments. By substituting the secondary variables in Eq. (47) and considering the essential boundary conditions (44) for the nodes on the boundary, we obtain the discretized system of equations:

$$\begin{cases}\displaystyle\sum_{k=1}^{n} u^k\int_\Omega \nabla\phi^k\cdot\nabla\phi^j d\Omega-\sum_{i=1}^{n_b} q^i\int_\Gamma\psi^i\phi^j dS=\int_\Omega g\phi^j d\Omega, j=1..n\\ \displaystyle\sum_{k=1}^{n}\phi^k(\boldsymbol{x}^i)u^k=\overline{u}^i, i=1..nb\end{cases} \tag{50}$$

As seen from (50), the integrals that need to be computed using a numerical quadrature are based on the derivatives of the shape functions

$$Ix_{kj}=\int_\Omega \phi^k_{,x}\phi^j_{,x}d\Omega \quad \text{and} \quad Iy_{kj}=\int_\Omega \phi^k_{,y}\phi^j_{,y}\,d\Omega \tag{51}$$

as well as on the shape functions:

$$F_j=\int_\Omega g\phi^j d\Omega \quad \text{and} \quad F_{ij}=\int_\Gamma\psi^i\phi^j\,dS. \tag{52}$$

3.6.2. Numerical integration procedure

We use either triangles or quadrilaterals as integration cells in 2D. Each integration cell is subdivided into 4 cells during the adaptive integration procedure using the mid-points of the edges. The function that guides the subdivision of integration cells is defined considering the integrals that need to be computed (51) and (52) as:

$$f(\mathbf{x})=\begin{bmatrix}\displaystyle\sum_{k=1}^{n}\left(\phi^k_{,x}(\mathbf{x})\right)^2\\ \displaystyle\sum_{k=1}^{n}\left(\phi^k_{,y}(\mathbf{x})\right)^2\\ \displaystyle\sum_{k=1}^{n}\left(\phi^k(\mathbf{x})\right)^2\end{bmatrix} \tag{53}$$

For the same reasons as in the 1D case, function g and the linear shape functions ψ^i do not appear in the definition of function f.

3.6.3. Performance assessment

To assess the integration performance we study problem (43) with:

$$g=0 \text{ in } \Omega=\{(x,y)|x^2+y^2<1\} \tag{54}$$

and

$$\overline{u}=x+y \tag{55}$$

with the solution

$$u=x+y. \tag{56}$$

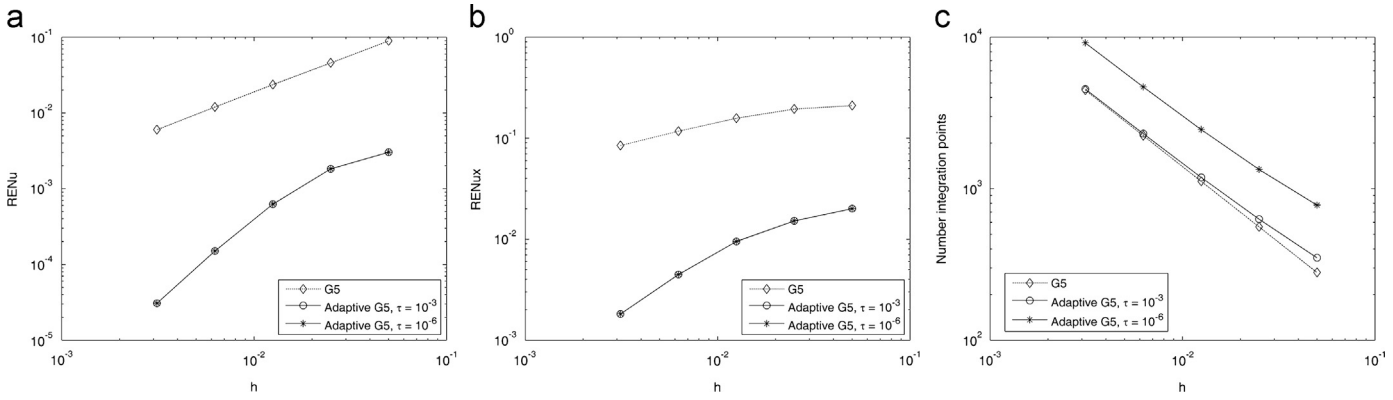

Fig. 6. Convergence study: relative error norms for the solution (a) and its derivative (b); number of integration points used (c). Note that the last two curves in plots (a) and (b) are so close they cannot be visually distinguished.

We use the scaling parameter $M=1.2$ for defining the nodal influence domains, which ensures sufficient nodes are allocated to each integration point for the computation of the shape functions. The integrals over boundary segments are computed using 10-point Gaussian quadrature. We used both quadrilateral (Fig. 7) and triangular cells for integration, the triangular cells being obtained by dividing the quadrilateral cells along the shortest diagonal.

The maximum and mean relative integration errors for the elements of the stiffness matrix versus the number of integration points are presented in Fig. 8. The curve marked "Tn; $n=3,6$" is obtained using 3-points and 6-points quadratures for each triangular integration cell as derived in [22]. The curve marked "Gn,n; $n=3,5$" is obtained using 3×3 and 5×5-points tensor product Gaussian quadratures for each quadrilateral integration cell. The curves marked "T6; decrease h" and "G5,5; decrease h" are obtained by dividing all integration cells into 4 smaller cells at each step and applying the selected quadrature (T6 and G5,5) with the resulting integration cells. The horizontal lines indicate the levels of accuracy τ used for the adaptive procedure at each step.

Similar conclusions as for the 1D case can be drawn from the integration error study. Our numerical experiments have shown that the benefits of the adaptive procedure, in terms of number of integration points required for a given accuracy, are higher for more irregular nodal distributions (as the shape functions have more complicated shapes and are more difficult to integrate).

Similar to the 1D case, solution accuracy does not significantly change once integration accuracy reaches a given value (Fig. 9). This result suggests it is not necessary to obtain very high integration accuracy, as it only leads to higher computational effort (number of integration points, and therefore number of function evaluations) without having any significant influence on the accuracy of the solution (due to a much higher discretization error). We propose that a convergence study should be performed to determine the best integration accuracy for a given discretization. Our proposed adaptive procedure is well suited for such study, as it offers control over the integration accuracy through parameter τ.

4. Discussion and conclusions

In this paper we propose an adaptive integration procedure for Mesh Free Galerkin Methods. A function constructed based on the shape functions guides an adaptive integration cell division process which stops only when the guiding function is integrated with a prescribed accuracy. The resulting distribution of integration points and corresponding weights can then be used to compute all integrals involved in the solution of the BVP over the problem domain.

The new adaptive quadrature has the following properties:

- The integration accuracy can be controlled by the analyst;
- The size of integration cells is automatically adjusted; the adaptive method detects areas where the shape functions have large variations and automatically increases the number of integration cells to maintain integration accuracy;
- It works for any shape and size of support domains;
- The adaptive procedure introduces new integration points only in the areas where the integration accuracy is not sufficient; therefore it creates an almost optimum number of integration points for a given accuracy;
- The resulting distribution of integration points depends only on the nodal discretization and the required accuracy;
- The procedure is especially effective in case of non-uniform nodal distributions.

We test the efficiency and accuracy of the proposed procedure for 1D and 2D BVPs and compare it with classical non-adaptive integration methods. The new method can ensure more accurate results using the same or lower number of integration points.

The initial integration cells used for the adaptive procedure do not have to be constructed based on the nodal discretization of the domain. We performed experiments with initial integration cells unrelated to the discretization and obtained similar results. Even if the initial integration cells are not related to the discretization, it is important that the size of these cells is closely related to the distance between nodes (see the discussion about characteristic length in Section 2.1).

The proposed integration method is evaluated for simple linear problems, in order to analyze its properties. For such problems the computation time is usually not greatly influenced by the number of integration points used, as the solution to the resulting system of equations takes a lot longer to compute. The solution algorithms that would greatly benefit from the proposed integration procedure are those that are greatly influenced by the number of integration points required. We intend to use the presented integration procedure for solving 3D elasticity problems by explicit algorithms in the context of a Total Lagrangian formulation [5,31], in which most of the solution time is spend on computing stresses at each integration point. Such explicit algorithms can also be used

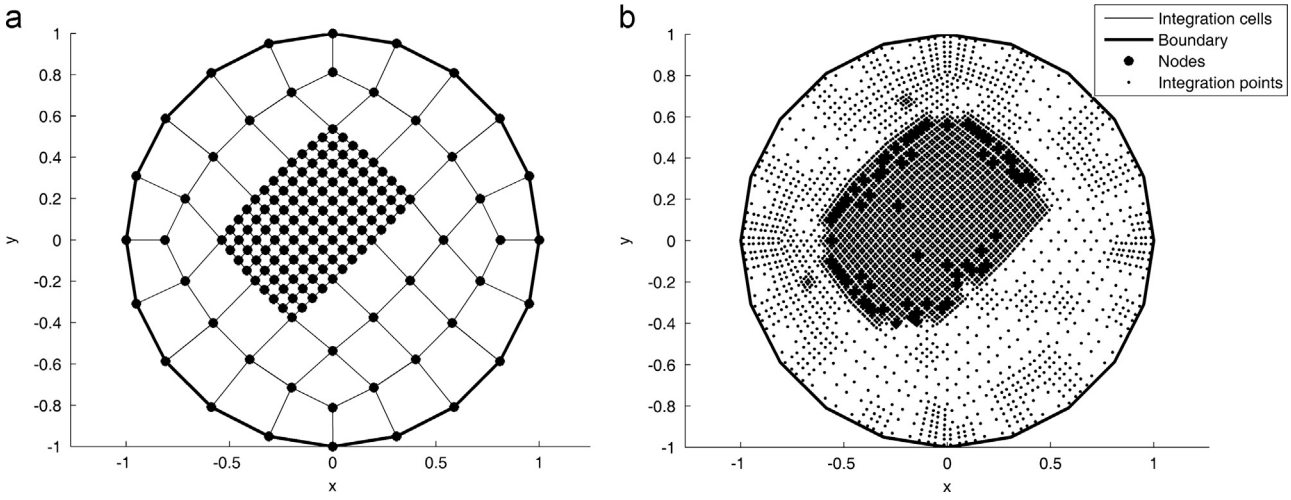

Fig. 7. (a) Point distribution used for discretizing the 2D domain and quadrilaterals used as initial integration cells. (b) Example integration point distribution, obtained using a Gaussian 2×2 tensor product quadrature for $\tau=0.01$.

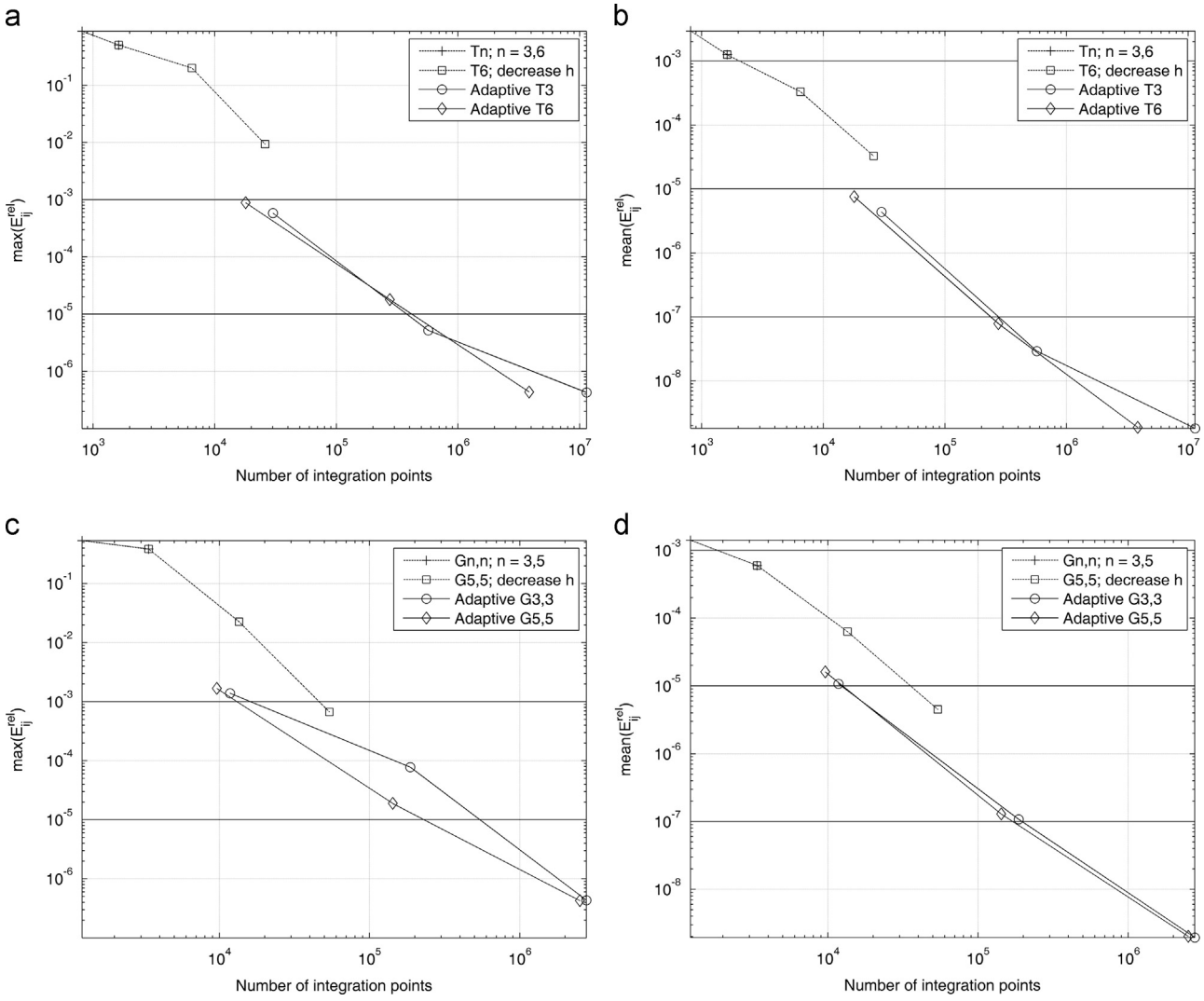

Fig. 8. Maximum and mean relative integration errors for the elements of the stiffness matrix versus the number of integration points for triangular (a, b) and quadrilateral (c, d) integration cells. Accuracy of the proposed adaptive procedure is increased by changing the value of τ. The first curves in these plots show the change in accuracy with the increase in number of integration points per cell. The second curve shows the change in accuracy with the decrease of integration cell size h. The horizontal lines indicate the levels of accuracy τ used for the adaptive procedure at each step.

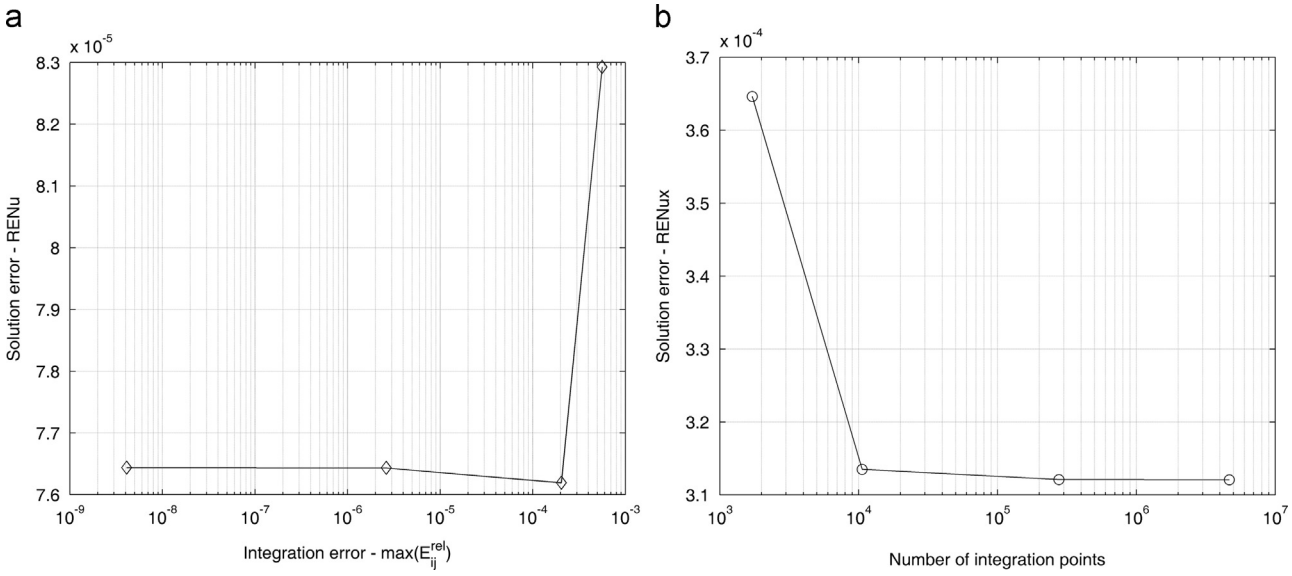

Fig. 9. Solution error norms versus integration error (a) and number of integration points used (b). The adaptive integration procedure was used with a T19 quadrature. Accuracy was increased by decreasing the required accuracy 100x at each step starting from τ=0.01.

for computing steady state solutions for non-linear problems when combined with Dynamic Relaxation [32]. The use of Total Lagrangian formulation means that the shape functions do not change during the solution process, and therefore the distribution of integration points used for numerical quadrature may not need to be updated.

The main benefit of the method over other integration methods consists in the possibility of controlling integration accuracy. Our results show that after achieving certain integration accuracy, the solution accuracy can no longer be significantly improved by increasing the number of integration points. This suggests a convergence study for the integration error may be appropriate, to find the optimum level of integration accuracy needed.

Acknowledgments

The financial support of the Australian Research Council (Discovery Grants no. DP1092893, DP120100402) and the National Health and Medical Research Council (Grant no. APP1006031) is gratefully acknowledged.

References

[1] Liu GR. Mesh Free Methods: Moving Beyond the Finite Element Method. Boca Raton: CRC Press; 2003.
[2] Belytschko T, Lu YY, Gu L. Element-free Galerkin methods. Int J Numer Meth Eng 1994;37:229–56.
[3] Dolbow J, Belytschko T. Numerical integration of the Garlekin weak form in meshfree methods. Comput Mech 1999;23:219–30.
[4] Miller K, Joldes GR, Lance D, Wittek A. Total Lagrangian Explicit Dynamics Finite Element Algorithm for Computing Soft Tissue Deformation. Commun Numer Methods Eng 2007;23:121–34.
[5] Horton A, Wittek A, Joldes GR, Miller K. A meshless total lagrangian explicit dynamics algorithm for surgical simulation. Int J Numer Method Biomed Eng 2010;26(8):977–98.
[6] Fries T-P, Matthies H-G. Classification and Overview of Meshfree Methods. Brunswick, Germany: Technische Universität Braunschweig; 2003.
[7] De S, Bathe K-J. The method of finite spheres with improved numerical integration. Comput Struct 2001;79:2183–96.
[8] Atluri SN, Kim H-G, Cho JY. A critical assessment of the truly Meshless Local Petrov-Galerkin (MLPG), and Local Boundary Integral Equation (LBIE) methods. Comput Mech 1999;24:348–72.
[9] Racz D, Bui TQ. Novel adaptive meshfree integration techniques in meshless methods. Int J Numer Methods Eng 2012;90:1414–34.
[10] Babuska I, Banerjee U, Osborn JE, Li Q. Quadrature for meshless methods. Int J Numer Methods Eng 2008;76:1434–70.
[11] Quak W, AHvd Boogaard, González D, Cueto E. A comparative study on the performance of meshless approximations and their integration. Comput Mech 2011;48:121–37.
[12] Chen J-S, Wu C-T, Yoon S, You Y. A stabilized conforming nodal integration for Galerkin mesh-free methods. Int J Numer Meth Eng 2001;50(2):435–66.
[13] Rosca VE, VMA Leitao. Quasi-Monte Carlo mesh-free integration for meshless weak formulations. Eng Anal Bound Elem 2008;32:471–9.
[14] BaniHani S, De S. Genetic algorithms for meshfree numerical integration. In: Griebel M, Schweitzer MA, editors. Meshfree Methods for Partial Differential Equations III. Berlin Heidelberg: Springer; 2007. p. 17–40.
[15] Strouboulis T, Babuska I, Copps K. The design and analysis of the Generalized Finite Element Method. Comput Methods Appl Mech Eng 2000;181:43–69.
[16] Espelid TO, Genz A. DECUHR: an algorithm for automatic integration of singular functions over a hyperrectangular region. Numer Algorithms 1994;8:201–20.
[17] Rice JR. A Metalgorithm for Adaptive Quadrature. J ACM 1975;22(1):61–82.
[18] Rice JR. Adaptive quadrature: convergence of parallel and sequencial algorithms. Bull Amer Math Soc 1974;80(6):1250–4.
[19] Gander W, Gautschi W. Adaptive Quadrature—Revisited. BIT. Numer Math 2000;40(1):84–101.
[20] Gonnet P. A review of error estimation in adaptive quadrature. ACM Comput Surv 2012;44(4):22.
[21] Garribba S, Quartapelle L, Reina G. SNIFF: efficient self-tuning algorithm for numerical integration. Computing 1978;20:363–75.
[22] Dunavant DA. High degree efficient symmetrical Gaussian quadrature rules for the triangle. Int J Numer Meth Eng 1985;21:1129–48.
[23] Smolyak SA. Quadrature and interpolation formulas for tensor products of certain classes of functions. Soviet Math Doklady 1963;4:240–3.
[24] Cools R. An encyclopedia of cubature formulas. Journal of Complexity 2003;19:445–53.
[25] Bungartz H-J, Griebel M. Sparse grids. Acta Numerica 2004;13:147–269.
[26] Gerstner T, Griebel M. Numerical integration using sparse grids. Numer Algorithms 1998;18:209–32.
[27] Gerstner T, Griebel M. Dimension-adaptive tensor-product quadrature. Computing 2003;71:65–87.
[28] Zhang L, Cui T, Liu H. A Set of Symmetric Quadrature Rules on Triangles and Tetrahedra. J Comput Math 2009;27(1):89–96.
[29] Li S, Liu WK. Meshfree Particle Methods. Berlin: Springer-Verlag; 2004.
[30] Lancaster P, Salkauskas K. Surfaces generated by moving least squares methods. Math Comput 1981;37:155.
[31] Miller K, Horton A, Joldes GR, Wittek A. Beyond finite elements: a comprehensive, patient-specific neurosurgical simulation utilizing a meshless method. J Biomech 2012;45:2698–701.
[32] Joldes GR, Wittek A, Miller K. An adaptive Dynamic Relaxation method for solving nonlinear finite element problems. Application to brain shift estimation. Int J Numer Methods Biomed Eng 2011;27(2):173–85.