Non-Negative Moment Fitting Quadrature Rules for Fictitious Domain Methods

Grégory Legrain

Non-Negative Moment Fitting Quadrature Rules for Fictitious Domain Methods

Grégory Legrain

gregory.legrain@ec-nantes.fr
GeM Institute, UMR CNRS 6183, École Centrale de Nantes, Université de Nantes, France

Fictitious domain methods enable to solve physical problems on unfitted grids, thereby avoiding time-consuming and error prone meshing phases. However, an accurate integration of the weak formulation is still mandatory, leading to the need for efficient quadrature strategies in the elements that are partially located in the physical domain. Various methodologies have been proposed to this end. Among them, the design of element-specific moment-fitting quadrature rules seems promising. However, some of the resulting weights are usually negative and the points located out of the domain which may lead to a lack quadrature stability. In this contribution, we aim to construct quadrature rules whose weights are all positive and points all located in the physical domain. Such rules can be constructed based on a non-negative least square resolution of the moment-fitting equations. The resulting quadrature formulas are compared to empirical quadrature rules and different variants of classical moment-fitting quadrature rules in both 1D and 2D. Benchmarks show that non-negative moment-fitting quadrature rules are robust and efficient although their setup cost can be significantly higher than classical moment-fitting. Finally, their application to linear elastic and small-strain elasto-plastic problems highlight their robustness for engineering applications.

Keywords: X-FEM; high-order; fictitious domain; quadrature rules; moment fitting; empirical interpolation; non-negative least square

1. Introduction

Advanced engineering applications such as those encountered in the aerospace industry rely on a tight relationship between CAD and CAE. Classical workflows make use of the finite element method which is a very robust and proven method. However, it is now admitted that the finite element method struggles to fully integrate with CAD, mainly due to the geometrical conversions that occur during the construction of the mesh. Geometries are affected in the process, and the topology of the model is lost. This is why various alternatives have been proposed in the literature in order to answer this issue: meshfree methods [1, 2], isogeometric analysis [3] and fictitious domains among others [4]. Such methods enable a more efficient transition between CAD and CAE. In the following, we will consider more specifically high-order fictitious domain methods such as the Finite Cell [5, 6, 7] or the high-order eXtended Finite Element Method [8, 9, 10]. These approaches can effectively uncouple geometry and approximation by extending the approximation domain away from the physical boundaries, avoiding time-consuming meshing and geometry de-featuring steps (see figure 1). In addition, the use of higher order basis functions lead to improved convergence rates and accuracy. Let us also mention that not all the difficulties are caused by the meshing process: usual B-REP CAD description only describes the

surface of the domains, and commonly contains hole, overlap and badly oriented surfaces [11]. Fictitious domain methods can be very robust in that regard, also simplifying the treatment of image-based applications [12, 13, 14, 15]. The flexibility and robustness of the method regarding geometrical as-

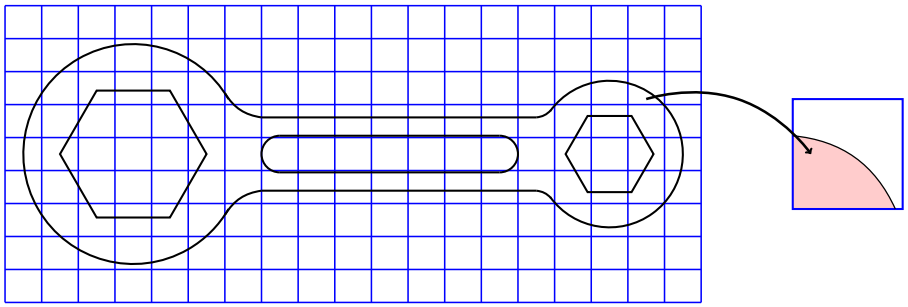
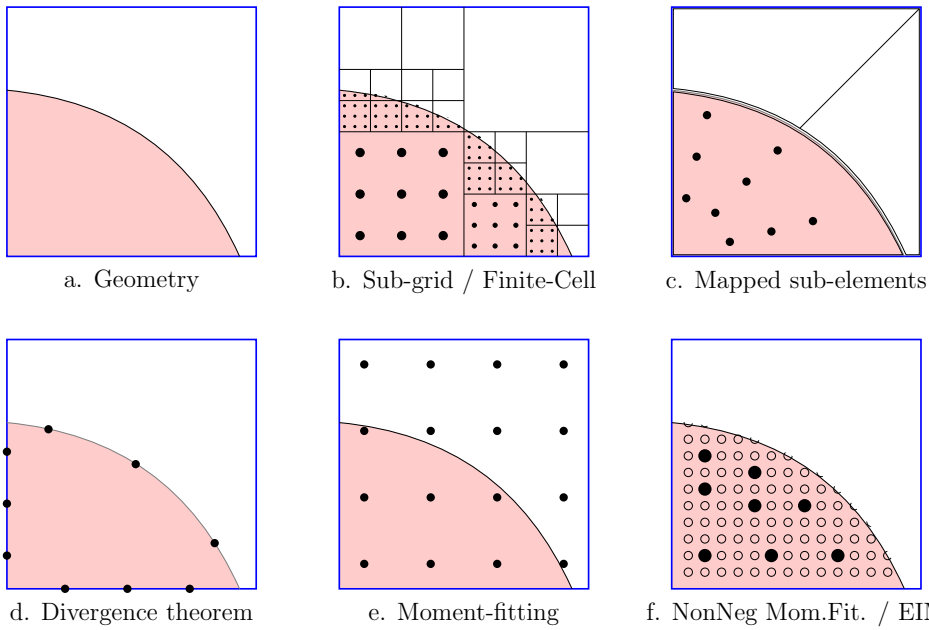


Figure 1: Setup of a mechanical problem using fictitious domain methods.

pects is balanced by the cost of the integration of the finite element operators. Indeed, approximation elements partially located in the physical domain have to be integrated solely in this part (see detail on figure 1). Various approaches have been proposed for the integration of these elements (see figure 2 for a general overview). The most common and robust methods are based the use of an embedded spatially adapted integration mesh which can be constructed based on space-tree structures (figure 2b), with [16, 17, 18] or without [5] any further tessellation. Note that such integration strategies can also be constructed based on specific mesh slicing approaches [19]. The main drawback of these methods lie in the resulting assembly cost, as numerous quadrature points are involved. Alternatively (figure 2c), coarse sub-elements can be mapped appropriately on the real geometry by means of specific mapping strategies [20, 21, 22, 23]. In the case of linear elastic problems, volume integrals can be recast into boundary integrals [24] (figure 2d), decreasing the number of quadrature points. Recently, the construction of ad-hoc quadrature rules based on the moment-fitting equations has been advocated (figure 2e) [25, 26, 27]. Quadrature points are usually positioned regarding the domain on which the integration takes place, and quadrature weights are calculated so that it is exact for a given polynomial order. The remaining difficulties with this method are twofold: (i) the weights may not be positive, which deteriorates the stability of the quadrature rule and (ii) the quadrature points may be located out of the domain which preclude their use for nonlinear problems with material variables.
The main purpose of this contribution is the following: (i) propose an approach to construct efficient and robust quadrature rules (i.e. with positive weights and quadrature points located strictly in the physical domain); (ii) assess the behaviour of moment-fitting quadrature rules more precisely, in particular with respect to the position of the quadrature points; (iii) evaluate the use of two other appealing quadrature rules: empirical quadratures [28] and "quality oriented"[1] moment fitting quadrature rules [29].
The rest or the paper is organized as follows: section 2 motivates this work by illustrating some drawbacks related to the use of negative quadrature rules, section 3 presents the theoretical background for constructing the various quadrature rules that will be compared (classical and quality oriented moment fitting quadrature rules, but also empirical quadrature rules). In particular, strategies for the derivation of sparse positive quadrature rules are presented. These quadrature rules are compared in section 4 in both 1D and 2D. The influence of the position of the quadrature points will be specifically considered. Finally, non-negative moment fitting quadrature rules are assessed in the context of fictitious domain methods.

[1]This terminology is proposed here, as the approach is not named in [29]



a. Geometry b. Sub-grid / Finite-Cell c. Mapped sub-elements

d. Divergence theorem e. Moment-fitting f. NonNeg Mom.Fit. / EIM

Figure 2: Integration over a domain of interest Ω embedded in an element. a. Geometry; b. Sub-grid/Finite-Cell: the element is subdivided recursively and quadrature rules are defined on the leaves of the subdivision; c. Mapped sub-element: a coarse triangulation of the domain is blended on the real geometry; d. Divergence theorem: volume integrals are recast as boundary integrals; e. Moment-fitting: quadrature points are arbitrarily chosen, and the moment-fitting equations are solved to compute quadrature weights; f. Non-Negative moment-fitting / EIM (Empirical Interpolation Method [28]): quadrature points are automatically selected among a set of tentative quadrature points, and the corresponding weights computed.

2. Motivation

In this section, we motivate the search for positive quadrature rules through illustrations of some drawbacks related to the use of negative quadrature rules. First, consider the integration of a 1D pulse-like function on a broken cell that mimics some localized thermal source effect (figure 3). An arbitrary set of quadrature points is chosen in order to define a seventh order rule and the corresponding weights are obtained through the moment-fitting method presented in section 3.2. The corresponding points and weights are given in table 1: Notice that four of them are negative. The resulting approximation of the integral turns out to be negative (-1.64637) whereas the exact integral is obviously positive (0.17725). On the contrary, using eight points Gauss-Legendre and non-negative moment-fitting[2] quadrature rules both lead to accurate approximated integrals. In real problems, this quadrature error would lead to misleading results in the area of interest.

As a second illustration, consider the quadrature of the Runge and abs functions $1/(1+25x^2)$ and $|x|$ on $[-1,1]$ with an increasing number of points. These two functions were chosen because the Runge function is notably difficult to interpolate on equispaced points, and the abs function is non-smooth. The relative quadrature errors of Gauss-Legendre and moment-fitting method on equidistant points are compared in figure 4. In this case, moment-fitting quadrature rules are negative for more than 9 points. It can be observed that while Gauss-Legendre quadrature converges with respect to the number of quadrature points, it is not the case for moment-fitting: This behaviour is due to the analyticity properties of the functions of interest (see [30]). On the contrary, moment-fitting rules with positive weights (see section 3.5) are seen to converge properly (better than Gauss-Legendre in the abs case).

[2]See section 3.5

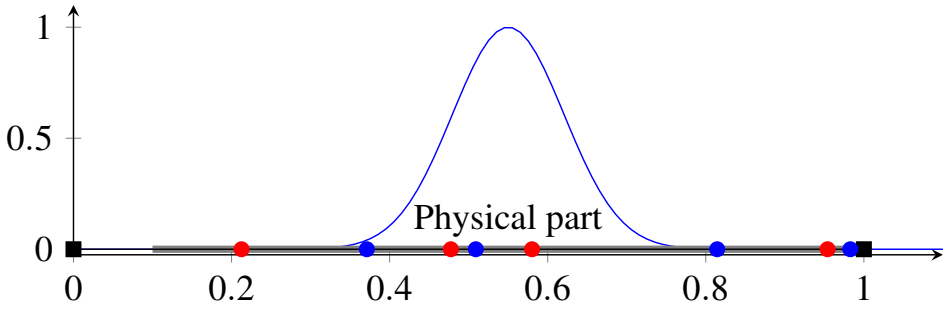


Figure 3: Integration of a pulse function on the physical part $[0.1, 1]$ of an element $[0, 1]$. Quadrature points are depicted as circles (red=positive weight, blue=negative weight)

Abscissa	Weight
0.21261261	0.40209976
0.37117117	-1.88397568
0.47747748	15.16705032
0.50900901	-17.50788951
0.58018018	4.67401643
0.81441441	-0.18383059
0.95405405	0.41874489
0.98288288	-0.18621563

Table 1: Integration of the pulse function $e^{-(x-0.55)^2/0.01}$ between 0.1 and 1. Order 7 moment-fitting rule points and weights.

As a last example, consider the mechanical problem consisting of a 1D bar of length 100 mm and section 0.75 mm^2, made of an elasto-plastic material with isotropic linear hardening (Young's modulus 10 000 MPa, tangent modulus 1000 MPa, yield stress 5 MPa) fixed on its left boundary and subjected to a uniform line load $f = 0.1$ MPa (see figure 5). We consider a fourth-order approximation using integrated Legendre shape functions on a five elements mesh. Gauss-Legendre quadrature is used in all the elements (4 points) but the middle one, where moment-fitting quadrature rules of varying order are built against evenly spaced points (see figure 5). In the linear-elastic regime, finite element operators are integrated exactly for an order $2 \times (p-1) = 6$ quadrature rule (presented in figure 5). Reference source-displacement and plastic strain curves using a Gauss-Legendre quadrature rules in all the elements are depicted in figure 6. The calculation is now run using quadrature rules with increasing order in the centre element ($6, 14, 16, 18$ and 20). The evolution of the maximum number of Newton-Raphson iterations along the 20 loading steps is presented in table 2. It is shown that the Newton-Raphson algorithm diverges when the quadrature order is increased. The iterative procedure is seen to converge even in some cases involving negative quadrature rules. However, the local plastic strain does not converge in the centre element, as presented in figure 7a. On the contrary, the approach advocated in this contribution (section 3.5) is seen to converge properly (figure 7b).

These three simple examples highlight the need for positive quadrature rules not only from mathematical arguments, but also from practical issues that may be encountered in both linear and nonlinear finite elements.

3. Efficient quadrature rules for fictitious domain methods

In this section, we introduce various approaches for the integration of finite element quantities on cut elements. We focus on approaches based on moment-fitting (figure 2e) and empirical quadratures

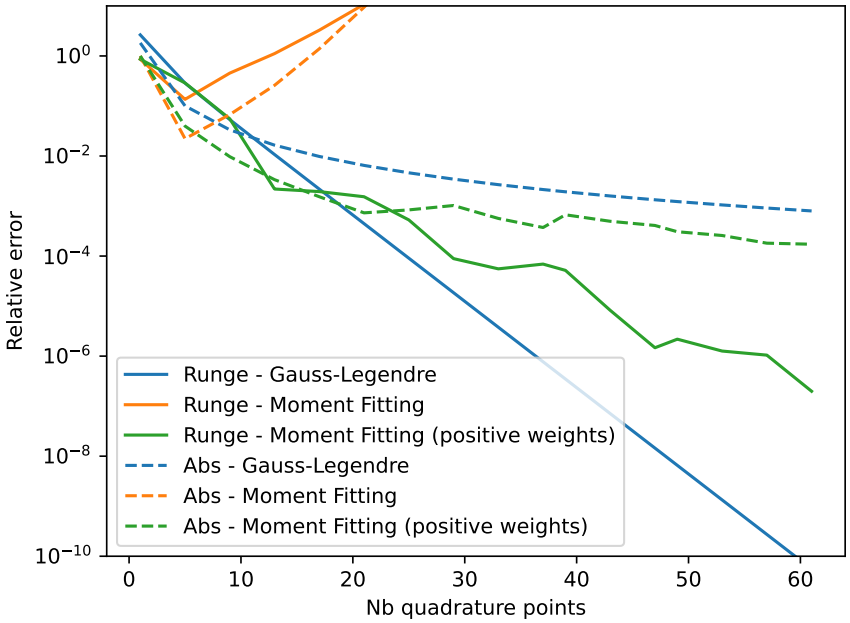


Figure 4: Relative quadrature error for Runge $(1/(1+25x^2))$ and abs functions on $[-1,1]$. Comparison between Gauss-Legendre quadrature and moment-fitting quadratures on equidistant points.

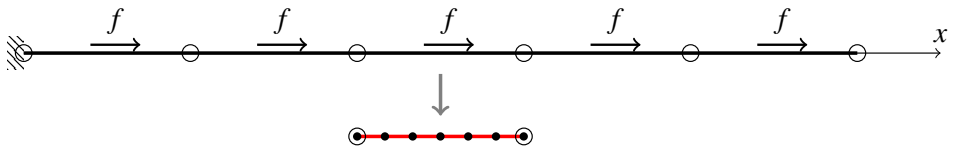


Figure 5: Bar subjected to a line load. Example of quadrature points in the centre element.

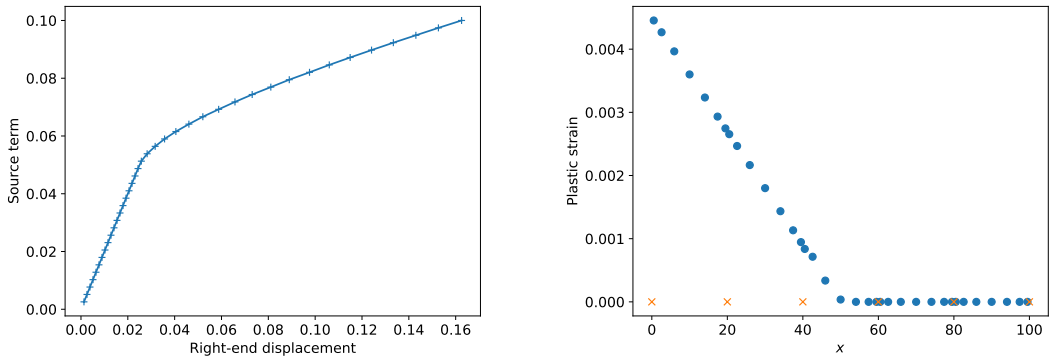
(figure 2f), but also propose a strategy for the construction of non-negative moment-fitting quadrature rules.

3.1. Quadrature rules for fictitious domain methods

The objective of the design of a quadrature rule is to approximate I, the integral of a function $f(\mathbf{x})$ over a given domain Ω (see figure 2a):

$$I = \int_{\Omega} f(\mathbf{x}) d\Omega \approx \sum_{j=1}^{n_q} f(\mathbf{x}_j) w_j \tag{1}$$

where n_q is the number of quadrature points located at $\mathbf{x}_j$ and w_j their weights. In this work, we aim to compare various quadrature construction methods with emphasis on their accuracy and the positivity of their weights which is a desirable property for robustness (stability of the quadrature rule) and non-linear computations (see section 2).



a. b.

Figure 6: Bar subjected to a line load. a. Right-end displacement versus source term; b. Plastic strain distribution and mesh nodes (×).

Order	Sign	Max. Iter
6	> 0	4
14	< 0	8
16	< 0	7
18	< 0	7
20	< 0	∞

Table 2: Bar subjected to a line load. Evolution of the number of iteration with respect to the quadrature order in the middle element (moment-fitting quadrature rules).

3.2. Moment fitting quadrature rules

Using the moment fitting method, it is assumed that the integrand $f(\mathbf{x})$ is well approximated on a given basis of m functions $\mathcal{H} = \{h_j\}_{j=1,\dots,m}$:

$$f(\mathbf{x}) \approx \sum_{j=1}^{m} \beta_j h_j(\mathbf{x}) \tag{2}$$

where β_j are the coefficients of the approximation on the basis. In this paper, we will consider two types of polynomial bases of order p_q in $\mathbb{R}^2$ namely monomial basis

$$\mathcal{H} = \left\{x^i y^j, i, j = 0, \dots, p_q\right\} \tag{3}$$

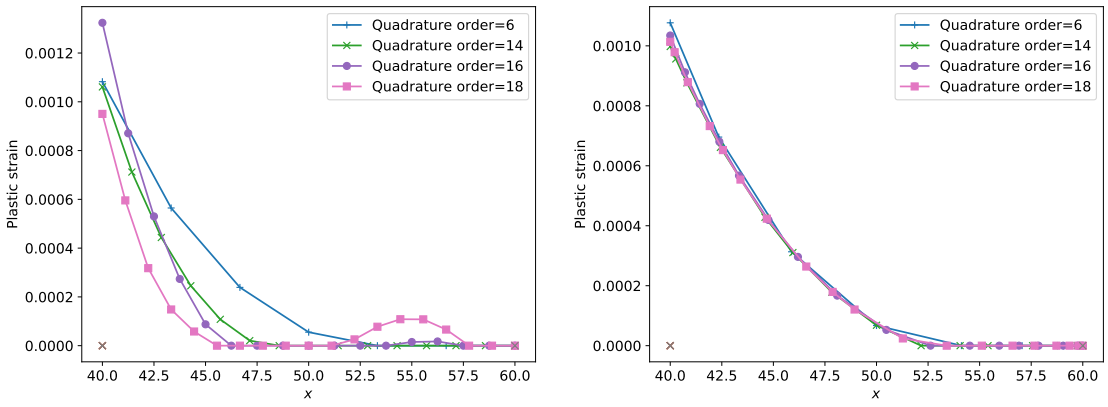
and Legendre basis

$$\mathcal{H} = \{L_i(x)L_j(y), i, j = 0, \dots, p_q\} \tag{4}$$

where $L_i(x)$ are unidimensional Legendre bases. In these expressions, full tensorial spaces are considered, but trunk spaces [31] will also be used. The integrand in I (eq.(1)) can be replaced by its polynomial approximation on $\mathcal{H}$. This leads to the so-called moment fitting equation that defines the set of weights that ensure a correct integration of the polynomial approximation of f:

$$\sum_{j=1}^{n_q} h_i(\mathbf{x}_j) w_j = \int_{\Omega} h_i(\mathbf{x}) d\Omega, \quad i = 1, \dots, m \tag{5}$$

This $m \times n_q$ system should be solved for the location of the quadrature points $\mathbf{x}_j$ and the corresponding weights w_j. The number of quadrature points is also a priori unknown, but as advocated in [27] we will



a. b.

Figure 7: Bar subjected to a line load. Plastic strain in the central element. a. Moment-fitting quadrature rules; b. Positive moment-fitting quadrature rules using the approach developed in this contribution (see section 3.5).

select $n_q \geq m$ in the following. Solving the moment fitting equations (5) with $\mathbf{x}_j$ as unknowns renders the problem nonlinear. In order to recover a linear problem, the location of the quadrature points is fixed in advance. The problem is thus solved using a classical least-square solver. It is possible to position the quadrature points only inside Ω [32], which is more convenient in the case where material variables have to be stored. Unfortunately, this choice seems to lead to a degradation of the conditioning of the least square problem, and thus to a decrease of the accuracy of the quadrature rules [32]. Alternatively, it was proposed to locate them at the Gauss-Legendre quadrature points of the approximation element, which leads to an improved robustness [32] to the price of points localized out of the physical domain Ω. Locating the quadrature points out of the physical domain can lead to severe issues in the nonlinear setting, especially for the update of the material variables [33]. To this end, specific strategies have been recently proposed [33, 34].
Constructing a moment fitting quadrature rule comes to the price of integrating the right-hand side of the moment-fitting equations and solve a least-square problem. The integration of the RHS of eqn.(5) (surface integral in 2D, volume integral in 3D) can be re-cast as a boundary integral thanks to the use of the divergence theorem [27], but only in the linear setting [33]. Otherwise, a sub-grid [5, 18, 16] quadrature rule can be used. Although this strategy can be quite expensive[3], it is very robust and can be amortized in the case of non-linear problems because of the repeated assemblies. Finally, it is worth mentioning that there is no way to make sure beforehand that the resulting quadrature weights will be positive, even if Tchakaloff [35] has proved that such solutions exist when $n_q = m$ in the case where the location of the quadrature points is free, and when $n_q \geq m$ if the points are fixed [36].

3.3. Empirical quadrature rules

Empirical quadrature rules have been first advocated in [28], as an application of the Empirical Interpolation Method. One can also cite [37] where the efficiency of the method was improved. The construction of the quadrature rule is based on the Empirical Interpolation Method which is first introduced.

3.3.1. The Empirical Interpolation Method

The Empirical Interpolation Method [28] (EIM in the following) is a general multi-purpose interpolation procedure. It constructs at the same time interpolation points and associated basis functions. Although

[3]Note that it is less expensive than the integration of the stiffness matrix as the integrands are cheaper to construct than finite element integrands.

not optimal, the method can achieve exponential convergence [28] (under reasonable conditions) and can be applied to any domain geometry. Finally, note that the method is hierarchical, which means that adding new interpolation points does not lead to the update of the existing set of interpolation points and basis functions. The method was originally introduced as a mean to deal with non-affine parametric functions in the Reduced Basis method (see [38] for a review). The method can be used to interpolate non-parametrized functions by the following reasoning [39]: Assume that we are interested in the interpolation of a function $f(\mathbf{x})$ that belongs to a space denoted as $\mathcal{U}$:

$$f(\mathbf{x}) \approx \mathcal{I}_M[f](\mathbf{x}) = \sum_{i=1}^{M} \beta_i q_i(\mathbf{x}) \tag{6}$$

where $\mathcal{I}_M[f](\mathbf{x})$ is the interpolation of f and $\{q_i\}$, $i = 1, \dots, M$ is a set of basis functions (at least C^0) that span a space $\mathcal{W}_M \subseteq \mathcal{U}$. The link with the EIM (which was introduced for *parametric* functions) is done through the introduction of a parametric generating function $\mathcal{G}(\cdot; \boldsymbol{\mu})$ that spans $\mathcal{U}$ by choosing $\boldsymbol{\mu}$ within a parameter domain $\mathcal{D}$: $\mathcal{U} = \text{span}\{\mathcal{G}(\cdot; \boldsymbol{\mu}) : \boldsymbol{\mu} \in \mathcal{D}\}$. The construction of $\mathcal{W}_M$ is done by choosing functions q_i within a finite dimensional subset of $\mathcal{U}$ spanned by $\mathcal{G}(\cdot; \boldsymbol{\mu})$: $\mathcal{W}_M = \text{span}\{\mathcal{G}(\cdot; \boldsymbol{\mu}) : \boldsymbol{\mu} \in \boldsymbol{\Xi}\}$, with $\boldsymbol{\Xi}$ of dimension M is a finite dimensional subset of $\mathcal{D}$. For example, the generating function can be constituted by the first M monomials or Legendre polynomials. To summarize, we look for the best[4] M terms polynomial interpolation of any polynomial function.

The EIM algorithm which is detailed in Algorithm 2 constructs in a greedy fashion interpolation points $\mathbf{x}_i^\star$ (also called magic points) and basis functions q_i. At each iteration of the algorithm, the next function and point are those that are the most poorly approximated by the current interpolation. It is to note that $\boldsymbol{B^M}$, the matrix of the linear system used to enforce the interpolation (see Algorithm 2 for its definition) is lower triangular with unit diagonal and therefore invertible.

Once the EIM magic points and basis functions are constructed, it is possible to compute the polynomial interpolation of any input function $f(\mathbf{x})$ by solving the interpolation system:

$$\boldsymbol{B^M}\boldsymbol{\beta} = \boldsymbol{f}^\star \tag{7}$$

where $\boldsymbol{f}_i{}^\star = f(\mathbf{x}_i^\star)$. Then, we can write $f \approx \mathcal{I}_m[f] = \sum_{i=1}^{M} \beta_i q_i(\mathbf{x})$

3.3.2. Empirical Quadrature Rules

Once the empirical interpolation (6) of f is known, its integration can be approximated by:

$$I = \int_\Omega f(\mathbf{x}) d\Omega \approx \int_\Omega \mathcal{I}_m[f](\mathbf{x}) d\Omega \tag{8}$$

$$\approx \sum_{i=1}^{M} \underbrace{\int_\Omega q_i(\mathbf{x}) d\Omega}_{\widetilde{w}_i} \beta_i \tag{9}$$

where $\tilde{w}_i$ are the weights of the empirical quadrature rule. The problem with this expression arises from the fact that $\boldsymbol{\beta}$ has to be computed for every new integrand through (7). Fortunately, it is possible to overcome this issue by recognizing that $\boldsymbol{B^M}$ is a (small) invertible matrix, and thus $\boldsymbol{\beta} = \boldsymbol{B^M}^{-1}\boldsymbol{f}^\star$. This operation can be done efficiently using the `dtrtri` routine from `Lapack` [40]. Writing the previous equation in a matricial way and inserting the new expression of $\boldsymbol{\beta}$ leads to:

$$I = \int_\Omega f(\mathbf{x}) d\Omega \approx \widetilde{\boldsymbol{w}}^T \boldsymbol{B^M}^{-1} \boldsymbol{f}^\star \tag{10}$$

$$\approx \boldsymbol{w}^T \boldsymbol{f}^\star \tag{11}$$

[4] The optimality condition considered here will be defined later.

Using w_i rather than $\widetilde{w}_i$ allows to write the quadrature rule in a usual manner. The quadrature points are chosen among a set of tentative points that can be located strictly in the physical domain. However, there is no way to ensure that the quadrature weights are positive. This drawback restricts the use of the empirical quadrature rules to linear elastic problems. Nevertheless, we will still consider this approach in the benchmarks presented in section 4 for comparison purpose.

3.4. Quality oriented moment fitting quadrature rules

This quadrature rule setup strategy was introduced in H. Sun's PhD thesis [41] as a robust integration approach for Cut-Cell methods[5]. In [41], this quadrature rule is used in conjunction with robust cell-cutting algorithms and is applied in [29] in a cut-cell adaptive Discontinuous Galerkin framework. The quadrature rule is an improvement of the moment fitting quadrature rules presented in section 3.2. The method works as follows: First, the projection of the integrand onto the polynomial basis is obtained through a weighted least square fit:

$$\boldsymbol{F} = \arg\min_{\boldsymbol{F}} \sum_{q=1}^{n_q} c_q \left(\sum_{i=1}^{m} F_i h_i(\mathbf{x}_q) - f(\mathbf{x}_q) \right)^2 \tag{12}$$

where $\boldsymbol{F}$ are the coefficients of the projection of f on $\mathcal{H}$ and c_q is the (approximate) volume of the Voronoi cell around $\mathbf{x}_q$. The solution of this problem writes:

$$\boldsymbol{F} = (\boldsymbol{V}\boldsymbol{C}\boldsymbol{V}^T)^{-1}\boldsymbol{V}\boldsymbol{C}\boldsymbol{f} \tag{13}$$

where $\boldsymbol{C} \in \mathbb{R}^{n_q \times n_q}$ so that $C_{qq} = c_q$, $\boldsymbol{V}$ is the $m \times n_q$ matrix associated to the moment fitting problem (5) and $\boldsymbol{f} \in \mathbb{R}^{n_q}$ contains $f(\mathbf{x}_q)$, $q = 1, \ldots, n_q$. Integrating the polynomial approximation of $f(\mathbf{x})$ leads to the following definition of the quadrature weights:

$$\boldsymbol{w} = \boldsymbol{C}\boldsymbol{V}^T(\boldsymbol{V}\boldsymbol{C}\boldsymbol{V}^T)^{-1}\boldsymbol{b} \tag{14}$$

where $\boldsymbol{w} \in \mathbb{R}^{n_q}$ contains the quadrature weights, and $\boldsymbol{b} \in \mathbb{R}^{n_q}$ corresponds to the right-hand side of the moment-fitting problem (5) (integral of the basis functions). Note that the weights prove to be independent of the integrand [41]. Once the quadrature points and weights are known, a quality measure of the quadrature rule Q is introduced:

$$Q = \boldsymbol{w}^T\boldsymbol{C}^{-1}\boldsymbol{w} \tag{15}$$

Quadrature rules defined as (14) are proved to converge, with Q also converging to one from above, and the quadrature weights w_q converging to $c_q > 0$ when $n_q \to \infty$. Sun et al. [41] proposed to choose properly the location and number of quadrature points in order to ensure that the quality of the quadrature rule is sufficient. In particular, magic points from the Empirical Interpolation Methods are used in this quality oriented quadrature rule whose algorithm is presented in Algorithm 3. In the following, the algorithm is modified in order to comply to both quality and positivity requirements (testing against both $Q > Q^{lim}$ and $\min_q\{w_q\} > 0$). Consequently, intrinsic judgement on the original method should not be made based on the undermentioned results.

3.5. Non-negative moment fitting quadrature rules

As an effort to construct positive quadrature rules, we propose a very simple approach to enforce the positivity of the solution of the moment fitting problem (5). A similar idea for the construction of quadrature rules was initially introduced in [30] in the 1D setting. First, let us note that it was proved in [35, 42, 36] that the (nonlinear) moment fitting equation is guaranteed to have solutions with $n_q = m$ quadrature points in Ω and all positive weights (Tchakaloff's theorem). Unfortunately, it may

[5]The method was not named in [41], so that 'Quality oriented' is proposed here.

not be possible to capture these solutions when solving the linear moment fitting equation (with fixed quadrature points). In this case, it is still possible to find such a solution, but with $n_q \geq m$ points [30, 43, 36]. In order to extract a positive solution, we recast (5) into a non-negative least square moment-fitting problem:

$$\text{Minimize} \|\boldsymbol{V}\boldsymbol{w} - \boldsymbol{d}\|_2^2 \quad \text{subject to } w_i \geq 0,\ i = 1, \ldots, n_q \tag{16}$$

where $\boldsymbol{V}$ are the matrix and right-hand side $\boldsymbol{d}$ which appear in (5). Such a solution can be obtained through the use of efficient constraint optimization solvers such as TRON (Trust Region Newton method) [44] or LBFGS-B (Limited-memory Broyden-Fletcher-Goldfarb-Shanno) [45]. Unfortunately, the number of quadrature points needed to ensure the existence of a positive solution may be large, which would hinder the efficiency of the quadrature rule (even for the "simple" case of equidistant points on a hypercube, it was shown that $n_q \gg m$ [43]).

Fortunately Davis [42, 46] proved theorem 1 which states that it is always possible to find a "sparse" [6] positive quadrature rule based on the resolution of the moment-fitting equations, provided that n_q is sufficiently large to ensure that a positive solution exists.

Theorem 1 (Davis [42]) *If a quadrature rule of order p_q exists with $n_q \geq m$ points and positive weights, then a quadrature rule of order p_q exists with the same points, but with only m non-zero and positive weights.*

We propose to obtain this solution by considering non-negative least square solvers such as `nnls` [47]. This solver is based on an active set strategy which is proved to be convergent, finite and produces a sparse solution (see algorithm 4 for an overview). Non-negative LASSO algorithms could be equally considered [48]. Note that if there are too few tentative quadrature points, solving (16) may fail or converge to a solution with a large residual (due to the fact that the positive solution exists when n_q is sufficiently larger than m). In the latter case, $\exists \epsilon > 0$ such that $0 < \|\boldsymbol{V}\boldsymbol{w} - \boldsymbol{d}\|_2 < \epsilon$, and the quadrature rule is not exact. Nevertheless, the quadrature rule can be sufficiently accurate if ϵ is sufficiently small. In the following, problem (16) will be solved using the `scipy`'s implementation [49] of the `nnls` algorithm presented in [47].
In practice, to make sure that the number of tentative points is sufficient, the non-negative solver can be embedded into an adaptive strategy that iteratively increases the number of tentative quadrature points if it fails to converge to an accurate solution. This approach and a recursive one targeted to fictitious domain methods are assessed in section 5. In the following section, the set of tentative quadrature points is built using a regular grid, but any random or quasi-random sampling of the geometry could be used. Like the empirical interpolation method, the approach selects at most m quadrature points from the tentative set and computes the associated weights (see figure 2f).

4. Quadrature rules benchmarks

We present here various benchmarks in order to compare the quadrature strategies detailed in the previous section. Both 1D and 2D benchmarks will be considered, and the accuracy of the integration of both polynomial and non-polynomial quantities is assessed. The quantity that is chosen to compare the methods is the relative error on the integration of an integrand function f :

$$e_r = \left| \frac{I_{ex} - I_q}{I_{ex}} \right| \tag{17}$$

where e_r will be called *Relative error* in the following section, I_{ex} is the exact value of the integral of f on the domain of interest, and I_q is the value of the integral obtained by one the methods that are compared. Note that in all the examples of this section, the integrals that are involved in the construction of the

[6] i.e. with m non-zero weights.

quadrature rules are obtained through symbolic calculations using the `sympy` python library[50]. This ensures that we are actually comparing the quadrature rule construction methods. In addition, the tolerance for the non-negative least square solver is set near to machine accuracy (10^{-10}).

4.1. 1D benchmarks

The 1D benchmarks considered hereunder serve the two following objectives: (i) verifying the performances of the four methods mentioned above in the case of polynomial integrands; and (ii) assessing the dependency of the accuracy of moment-fitting approaches (classical and non-negative) on the location of the quadrature points in the case of non-polynomial integrands.

4.1.1. Verification for polynomial functions

We consider the integration of polynomial functions of different orders $p = 1, \ldots, p_{\max}$ ($p_{\max} = 10$) on $[-1, 1]$. Quadrature rules of order $p_q = 7$ are defined. Moment-fitting based quadrature rules are obtained using both monomial and Legendre basis (cf eqns.(3) and (4)). $p_q + 1$ evenly spaced points are selected for building these rules. The relative integration error e_r with respect to p is plotted in figure 8. Note that random polynomial functions are used as integrands. As expected, all rules are exact up to machine precision as long as $p \leq p_q$. Note also that all the quadrature rules have positive weights for this example. Let us also stress that this desirable feature is enforced by construction in the case of non-negative moment fitting rules and for the quality oriented quadrature rule (denoted as *Sun et al.* in the remaining figures).

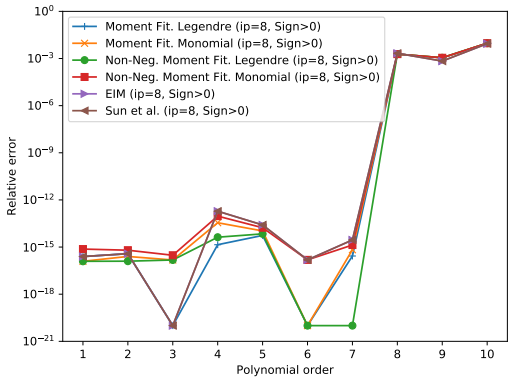


Figure 8: Evolution of the relative error e_r with respect to the polynomial order p of the integrand. All quadrature rules are of order 7.

4.1.2. Influence of the position of the quadrature points on the accuracy

In a second verification benchmark, we look for the integration of a non-polynomial function $f(x) = \sin(\pi x)$ in $[0, 1]$. A non-polynomial function is chosen here because, as highlighted in [41], despite the fact that a quadrature rule of order p_q can integrate exactly the first p_q terms of the polynomial approximation of f, the rule may magnify the error associated to the remaining higher order terms. One of the objectives of this benchmark is to study the interplay between the location of the quadrature points and the actual accuracy of the quadrature rule. A set of 1000 tentative evenly spaced quadrature points are first defined in $[0, 1]$. Then, n_q of them are randomly picked and used as quadrature points for the construction of classical and non-negative moment fitting quadratures (order $p_q = 7$ hereunder). These points also serve as candidate points for the EIM and quality oriented moment fitting. It is important to mention here that no matter the location of the quadrature points, the resulting quadrature rules integrate polynomial functions of order p_q within machine accuracy. We gradually increase the

Rule type	r	% positive	% inaccurate
Classical.	1	0.3	1.5
Classical.	5	66	0.
Classical.	10	92.1	0.
Non Neg.	1	100	99.8
Non Neg.	5	100	10.5
Non Neg.	10	100	0.5

Table 3: 1D scattering: Rules positivity and failure ratios.

amount of (tentative) points from $n_q = (p_q + 1)$ to $n_q = r \times (p_q + 1)$ $(r = 5, 10)$ in order to study the behaviour of the quadratures against this parameter. The corresponding results are depicted in figure 9 and compared to Gauss-Legendre, EIM and quality oriented moment-fitting (which are all positive here). We can observe that although the average accuracy is on par with the other methods, moment-fitting quadrature rules display a scattering between four and six orders of magnitude. Increasing r has a small influence on the average but clearly decreases the scattering. Non-negative moment-fitting quadratures also lead to such scattering, but to a smaller extent. This is due to the fact that only converged rules are kept: for lower r, the majority of the random draws lead to low accuracy rules as positive solutions do not exist, Tchakaloff's theorem being not satisfied for such a few tentative points. Positivity and failure ratios are displayed in table 3: one can observe that for low r, positivity is hard to achieve, but that the ratio growths significantly with r (which is consistent with Tchakaloff's theorem). Non-negative rules are always positives, but the lower the size of the tentative set, the harder the algorithm converges. The filtering of the non-converged solutions explains the low scattering for $r = 1$ in figure 9. Finally, it is noticeable that, for a given quadrature order, increasing the amount of quadrature points from its strict minimum $(r = 1)$ leads to more accurate rules for classical moment-fitting (two orders of magnitude from $r = 1$ to $r = 5$ here).

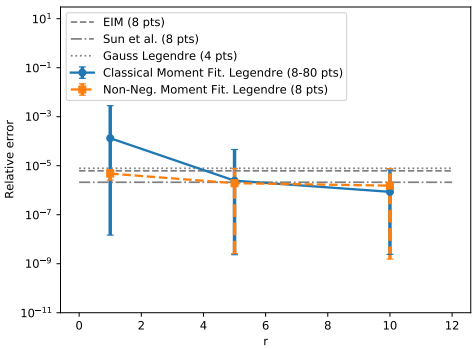


Figure 9: 1D scattering: Relative errors for 1000 evaluations with $n_q = r \times (p_q + 1)$ random quadrature points. Average and dispersion of the resulting moment-fitting rules are depicted. Gauss-Legendre, EIM and quality oriented moment fitting error are depicted as constant dashed lines (as they operate with already fixed quadrature points).

4.1.3. Partial conclusion

Based on these two simple 1D examples, important conclusions can be drawn. First, the accuracy of moment fitting quadrature rules (both classical and non-negative) can be strongly affected by the location of the quadrature points in the case where non-polynomial functions have to be integrated. The positivity of the classical moment fitting quadrature rules is seen to depend on the choice of the

quadrature points (number, but also location). Scattering and negativity of the quadrature rules can be decreased by increasing the number of points in the quadrature. Second, EIM and quality oriented quadratures are very accurate and automatically select the best set of quadrature points. Concerning non-negative moment-fitting, an increase of the amount of tentative points improves both scattering and error level (like classical moment fitting). Quite large sets of tentative points are necessary for the algorithm to converge with a proper failure ratio. In practice (see section 6), the residual is used as a criterion to check whether the number of tentative points has to be increased. Increasing the number of tentative points has no influence on the cost of the final quadrature rule, as the algorithm always selects the best $p_q + 1$ quadrature points (and even less in some cases, due to the tolerances in the algorithm). Finally, note that the second illustration presented in section 2 also highlighted the good properties of non-negative moment-fitting rules for more challenging functions (Runge and abs functions).

4.2. 2D benchmarks

In this section, we focus on the integration over partial domains that are encountered in the calculation of elementary quantities with fictitious domain methods. Three geometries are considered hereunder (see figure 10). They all represent the integration over a square element $(x, y) \in [0, 1]^2$ filled with different geometries. Again, we are interested in the accuracy of the approximated integral on the

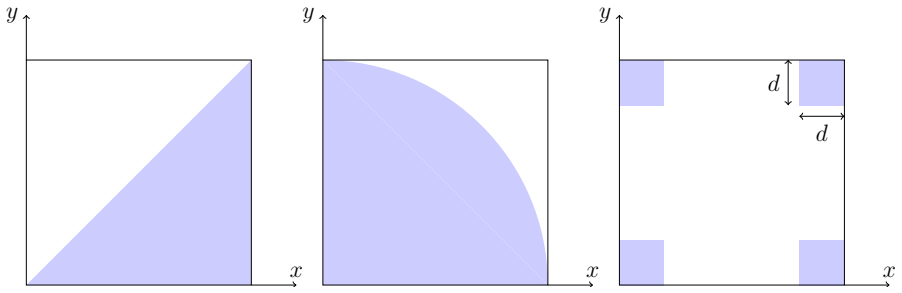


Figure 10: Integration domains for the 2D benchmarks. From left to right: triangular domain, quarter circle, corners ($d = 5\,10^{-2}$ here). The element side is 1 in all cases.

subdomains through its relative error e_r. We conducted benchmarks similar to those presented in 1D. As the objective is to use these quadrature rules for nonlinear problems, quadrature points were chosen exclusively inside the physical domain. These points are obtained by defining a regular grid of point, keeping only those located in the domain. These tests are not presented here[7] and led to the following conclusions for polynomial integrands:

- The accuracy of the moment fitting approaches strongly can depend on the geometry. It can be close to machine precision (triangle) or quite large (quarter circle and corners). This behaviour was also highlighted in [32] and imputed to the conditioning of the moment-fitting equations. This motivated the authors to locate their quadrature points at Gauss-Legendre quadrature points (and thus *out* of the domain of interest which is not convenient for nonlinear materials). Interestingly, the closer r to one, the worse the behaviour of the classical moment fitting. Larger r enables to recover machine accuracy;

- Non-negative moment fitting quadrature rules lead to accuracies similar to classical moment fitting provided that the set of tentative points is sufficiently large. The size of this set depends both on the order of the rule and on the geometry. Also, the higher the order of the rule the larger the size of the tentative set needed to converge;

[7] Some raw material is however given in B

- Quality oriented quadrature rules lead to positive quadrature schemes, but also a somewhat larger error level. This error level seems to be linked to the stability of the QR factorization that is needed for the construction of the quadrature rule;
- EIM seems to be the more robust approach, despite the fact that the quadrature scheme is negative for the three cases presented here;

Scattering benchmarks conducted with non-polynomial integrands led to the following conclusions:

- With respect to the results presented in 1D, more points were needed for Tchakaloff's theorem to hold ($r = 10$ led to less than 7% of positive rules for classical moment-fitting);
- A scattering in the accuracy of randomly generated rules is still noticeable;
- Larger tentative sets (with respect to 1D) may be needed so that the non-negative least square solver can converge to the requested accuracy, depending on the geometry;
- For a given quadrature order, increasing r lead to more accurate classical rules (between 3 and 6 orders of magnitude accurate in average by increasing r from 1 to 5). If converged, non-negative quadrature rules are not significantly influenced by the value of r.

4.2.1. Accuracy comparison with Gauss-Legendre moment-fitting

We now compare the accuracy of non-negative moment fitting quadratures to EIM quadratures and classical moment-fitting quadratures on Gauss-Legendre points (advocated in [32]). Indeed, this last approach is the state of the art for the use of moment-fitting quadrature rules in the context of fictitious domain methods. Eighth order quadrature rules are considered here, and figure 11 illustrates the location of the resulting quadrature points in the case of the triangular geometry presented in figure 10. In the case of the EIM and non-negative moment-fitting, the points are selected by the algorithms among the tentative set of points whose construction was presented in the previous section. Let us mention that tensor product bases are used here in order to compare the three methods with the same number of quadrature points (81).

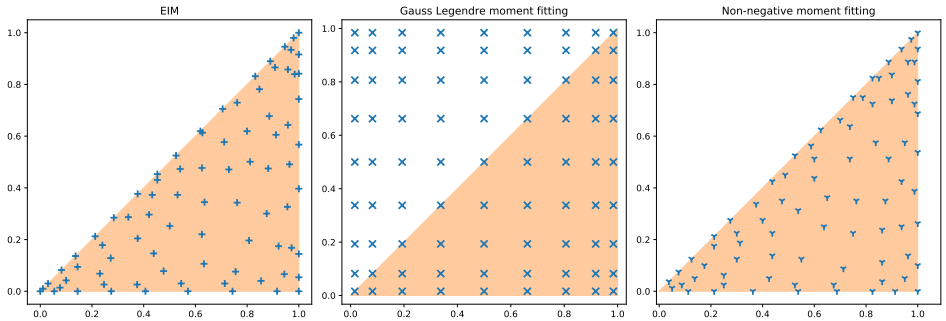


Figure 11: 2D domains, accuracy comparison with EIM and classical moment-fitting. Resulting points for quadrature rules of order 8. From left to right: EIM, moment fitting based on Gauss-Legendre points, non-negative moment fitting.

The error levels for polynomial and non-polynomial functions are given in table 4. We can observe that all the quadrature rules are exact up to machine precision if the integrand is a polynomial function. EIM and non-negative moment fitting quadratures are very consistent and usually lead to the best accuracy (although the EIM quadratures are not positive here). In addition to be negative, moment-fitting quadratures based on Gauss-Legendre points are not as consistent as they depend quite heavily on the geometry. In fact, this behaviour is in line with the conclusions of the previous section in terms of the influence of the location of the quadrature points. Only the triangular case leads to results that are

more accurate than EIM and non-negative moments fitting. This seems to be linked to the symmetry of the geometry: Gauss-Legendre quadrature points are optimal on the square (full) domain, leading to an accuracy of order $2p_q - 1$. This order of accuracy is preserved in the triangular case because of the symmetry of the geometry and quadrature points with respect to the diagonal of the square. In the case where the height of the triangle is halved and with the non-polynomial integrand, EIM and non-negative quadratures lead to an error of $1.12\,10^{-7}$ and $1.44\,10^{-7}$ respectively, whereas moment-fitting on Gauss-Legendre points leads to $3.69\,10^{-6}$. The convergence of the quadratures is finally monitored for this last geometry (1×0.5 square triangle) by increasing the quadrature order from 4 to 14 with a more complex (but regular) integrand: $f(x, y) = \sin(2.1\pi\, x)\, \sin(6.3\pi\, y)$. The corresponding results are presented in figure 12: it is shown that all approaches converge due to the regularity of the integrand. Gauss-Legendre moment-fitting and EIM converge at the same rate, but with a lower level for EIM (three orders of magnitude). Non-negative moment-fitting converges at a higher rate, further improving the error level.

Integrand	GL Moment Fitting	Non-Neg. Moment Fitting	EIM
		Triangular	
Polynomial	$7.3\,10^{-15}$	0.	$2.61\,10^{-13}$
Non-polynomial	$4.27\,10^{-9}$	$2.1\,10^{-4}$	$2.66\,10^{-4}$
		Quarter circle (Radius 1.0)	
Polynomial	$7.92\,10^{-15}$	$9.0\,10^{-16}$	$2.34\,10^{-14}$
Non-polynomial	$1.17\,10^{-9}$	$2.41\,10^{-9}$	$2.51\,10^{-8}$
		Quarter circle (Radius 0.2)	
Polynomial	$1.24\,10^{-16}$	$1.49\,10^{-15}$	$3.7\,10^{-11}$
Non-polynomial	$1.2\,10^{-8}$	$3.23\,10^{-12}$	$8.77\,10^{-13}$
		Corner	
Polynomial	$5.95\,10^{-15}$	$4.1\,10^{-16}$	$8.32\,10^{-11}$
Non-polynomial	$2.67\,10^{-3}$	$1.57\,10^{-7}$	$2.43\,10^{-8}$

Table 4: Comparison of the error levels of various quadrature rules of order 8. All the quadrature rules involve 81 points. Polynomial integrand: random polynomial of order 8. Non-polynomial: $\sin(\pi x)\sin(3\pi y)$ for triangular and corner geometry; $\frac{1}{25}(r^5 - 1) + r^2 cos(2\theta)$ for the quarter circle. Refer to figure 10 for the geometries.

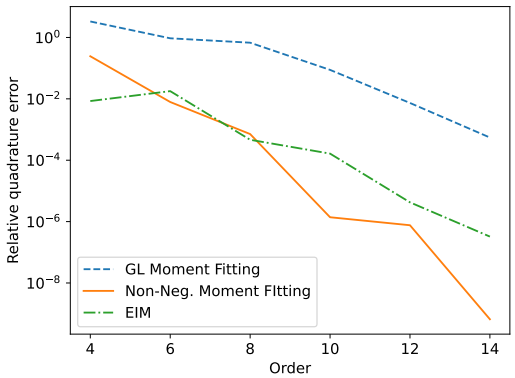


Figure 12: Convergence of EIM, Gauss-Legendre moment-fitting and non-negative moment-fitting for increasing quadrature order on a 1×0.5 square rectangle.

4.2.2. Partial conclusions

These 2D benchmarks allow to refine the conclusions drawn in 1D: First, it is increasingly complicated (if not almost impossible) to construct a positive moment-fitting quadrature rule with fixed quadrature points when the spatial dimension increases, due to a prohibitive number of quadrature points. Second, quality oriented moment-fitting quadrature rules are also more difficult to build if positivity has to bee enforced (we were not able to construct such rules for some geometry/order couples with orders close to $p_q = 18 - 20$). Third, EIM quadrature rules are robust and efficient but are usually not positive. Fourth, non-negative moment fitting quadrature rules lead to sparse and positive weights. Their accuracy depends on the number of tentative quadrature points that should be sufficiently large so that Tchakaloff's theorem holds. Once these requirements are taken into account, non-negative moment-fitting approaches lead to the best compromise in terms of quadrature efficiency (low number of quadrature points) and accuracy. However, the setup cost of the method has to be assessed, which is the objective of the next section.

5. Cost assessment and improvement

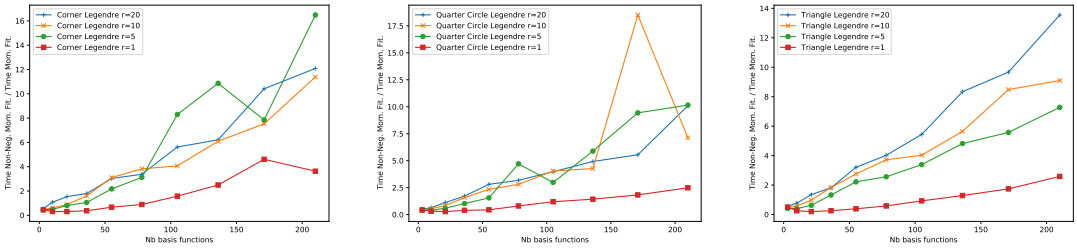
We now focus on the additional cost brought by the use of the non-negative moment-fitting method. The focus is made on the 2D examples introduced in the previous section. First, a micro-benchmark targeted at the cost of the use of `nnls` rather than a classical least square solver is considered. Second, an approach is proposed to ensure the satisfaction of Tchakaloff's theorem. Finally, the different methods are compared on the three 2D representative cases presented in section 4.2.

5.1. Cost comparison with classical moment-fitting

Compared to classical moment-fitting approaches, the main significant overheads are (i) the use of a non-negative least square solver while computing the quadrature weights and (ii) the extra size of the moment-fitting problem (due to the higher number of tentative points).

In order to quantify this overhead, the time spent to solve the moment-fitting problem is measured and compared for classical least-square and non-negative least-square solvers (respectively `numpy`'s `lstsq` and `scipy`'s `nnls`). More precisely, `lstsq` relies on Lapack's `dgelsd`, whereas `nnls` relies on the fortran implementation presented in [47] (which does not make use of any BLAS directive). This is why single-threaded timings will be considered in the following. The monitoring of the relative solving time (`nnls` time divided by `lstsq` time) is measured for increasing quadrature orders and numbers of quadrature points (tentative for `nnls`), see figure 13. The number of basis functions is used as x-axis rather than the order of the quadrature (which ranges from 1 to 19). This figure highlights the computational overhead associated to the use of the non-negative least square solver rather than the classical one. These curves underline a close to linear dependency of the overhead with respect to the number of basis functions. We can observe that this algorithmic change leads to an increase of the computational time of at most one order of magnitude in the most unfavourable case with large quadrature order and large r (defined as follows: $n_q = r \times m$ where n_q is the number of quadrature points and m the number of basis functions for a given quadrature order). On the contrary, if r is small (of the order of 1), then the overhead drops below 2 (provided that `nnls` converges for this small amount of tentative points). It is interesting to note that in this case `nnls` can even outperform `lstsq`.

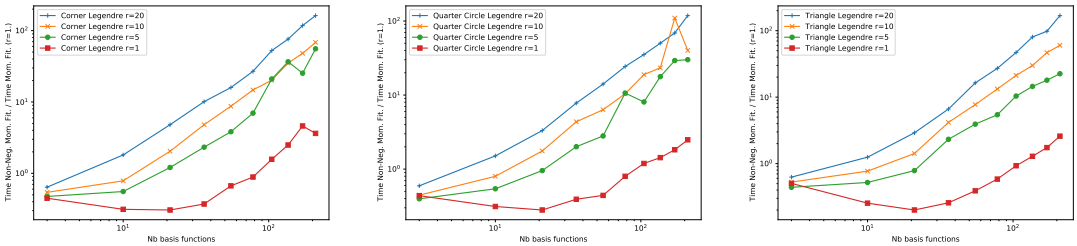
We finally consider the worse case scenario (figure 14) where the timings are normalized with respect to moment fitting quadratures with $n_q = m$ (i.e. the cheapest quadrature rule using this approach, which will inevitably be negative). Logarithmic scales are used here, so that a $m^{3/2}$ raise can be highlighted asymptotically, independently of r. With this new reference time, the cost penalty is very large and increases up to two orders of magnitude. Again, we can also observe that when r is moderate, we can actually expect a speedup for low order quadratures. These curves also enable to extrapolate the overhead for 3D problems (with *this* particular solver): it turns out that for an order 16 quadrature rule in 3D, we can estimate the overhead to range between 100 and 1000 times depending on r. These results motivate the use of ad-hoc strategies to improve the robustness and efficiency of the method.



a. b. c.

Figure 13: Cost comparison: relative time versus the number of basis function for order 1 to 19 rules. Coefficient r is defined as follows $n_q = r \times m$ where n_q is the number of quadrature points and m the number of basis functions for a given quadrature order. a. Corner, b. Quarter circle and c. Triangle. See figure 10 for an illustration of the geometries.

Note also that all the conclusions drawn in this section are closely related to the implementation of the non-negative least square solver. We also let the solver converge up to machine precision when possible which may be overkill for rules of moderate order. More efficient large scale solvers like ASA [51] or SBB [52] would mitigate this issue. In particular, Luo and Duraiswami proved that a careful implementation of the `nnls` algorithm together with multi-threaded BLAS could lead to remarkable speedups [53]. In addition, it is very important to bear in mind that (i) the setup of the quadrature is done once, while the quadrature itself may be used several times, especially for nonlinear problems and (ii) ths resolution of the moment fitting equations is one step in the creation of the rule: constructing the matrix and integrating the right-hand side of the system have a cost, so that the overhead depicted in these micro-benchmarks is pessimistic upper-bounds, as it will be shown in the next sections.



a. b. c.

Figure 14: Cost comparison: relative time versus the number of basis function for order 1 to 19. Coefficient r is defined as follows $n_q = r \times m$ where n_q is the number of quadrature points and m the number of basis functions for a given quadrature order. The overhead is measured with respect to the cheapest moment-fitting quadrature. a. Corner, b. Quarter circle and c. Triangle. See figure 10 for an illustration of the geometries.

Comments on matlab's lsqnonneg Matlab[8] provides an implementation of the non-negative least square solver proposed by [47]. We do not advocate the use of this implementation because of the following reasons: (i) While it seems to be relatively fast (sometimes faster that `nnls` Fortran implementation), its default accuracy is not sufficient for the class of problems considered here. (ii) Decreasing the tolerances makes the algorithm more accurate, but significantly slower than the Fortran implementation in our tests. This stems from the fact that the algorithm is not exactly the one used in `nnls`.

[8] R2018b here.

The latter implements partial updates of the unconstrained least square resolutions and also takes into account the use of finite precision arithmetic in the implementation.

5.2. Performances improvements

As presented in the last section, we aim at keeping the overall cost of the setup of the quadrature rule as small as possible.

5.2.1. Adaptive algorithm

One possible path is to keep the size of the moment-fitting problem as small as possible, but sufficiently large to cope with Tchakaloff's theorem. We propose here to select r arbitrarily (but small for efficiency purpose) then increase it until a converged solution is found (see algorithm 1). Extra resolutions may be involved if the solution is not found at the first try, but the number of quadrature points being small, the cost is expected to be moderate. The increase of the size of the tentative set can be done in many ways: for example through a regular grid of points of increasing resolution (similar to [32]), or by using the quadrature points resulting from rules of increasing orders built on the space-tree + level-set tessellation (see figure 15 a). Thanks to the developments detailed in the next subsection, the convergence of this second enlargement strategy is actually provable (although there is no mean to ensure that it is more efficient than the regular grid strategy).

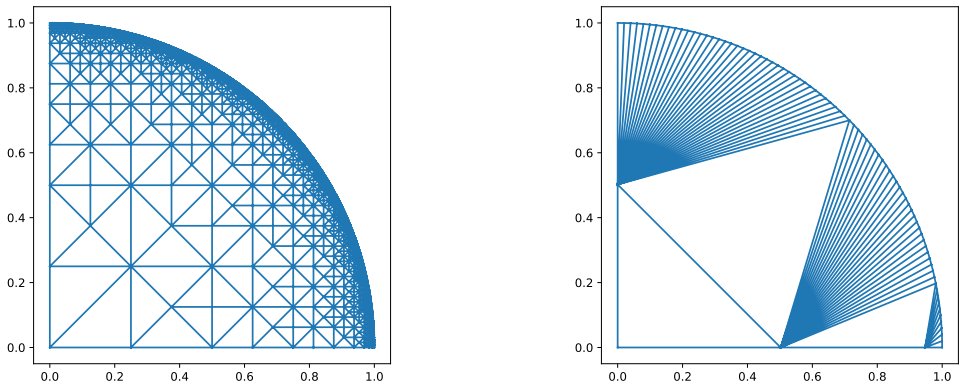
Data: Quadrature order p, Residual tolerance tol
Result: Non-negative quadrature rule in dimension d: $\{(x_i, w_i)\}_{i=1\cdots(p+1)^d}$
$r = 1.$;
Build the moment fitting right-hand side $\boldsymbol{h}$;
do
  Set $n_q \geq r(p+1)^d$ tentative points;
  Build the moment fitting operator $\boldsymbol{V}$;
  Find the non-negative least square solution of $\boldsymbol{V}\boldsymbol{w} = \boldsymbol{h}$;
  Compute the residual of the solution ε;
  Increase r;
while $\varepsilon > tol$;

Algorithm 1: Non-negative moment-fitting quadrature: adaptive algorithm.

5.2.2. A recursive algorithm

The main theoretical difficulty for constructing non-negative quadrature rules at first try is the necessity to satisfy Tchakaloff's theorem. It turns out that in the case of fictitious domain methods, an upper bound for r can be found. Let us consider the case of the quarter circle from figure 10. If the corresponding element is treated using a level-set representation, then a tessellation of the physical part of the element is already available from the space-tree + level-set quadrature although it contains too many sub-elements (see 15 a). Yet, the amount of sub-elements can be decreased through a local tessellation from the boundaries (see 15 b). If we attach standard (positive) quadrature rules of order p_q to these sub-elements, then the rule composed by the union of all the quadrature points and their associated weights is solution of the moment-fitting equations. All the weights being positives, then this set of points satisfies Tchakaloff's theorem, and from Davis' theorem (theorem 1) `nnls` will be able to extract a sparse and positive solution. This set of points thus gives a proven working set of quadrature points ready to be sparsified. However, this set of point must rather be considered as an upper bound given its large size: working directly with these points would be too expensive.

We thus propose a recursive approach that will render practical the work on this set of points. The procedure depicted in figure 16 works as follows: First, the quadrature mesh is partitioned hierarchically,



a. b.

Figure 15: Base quadrature meshes: a. sub-grid/space-tree; b. boundary tessellations

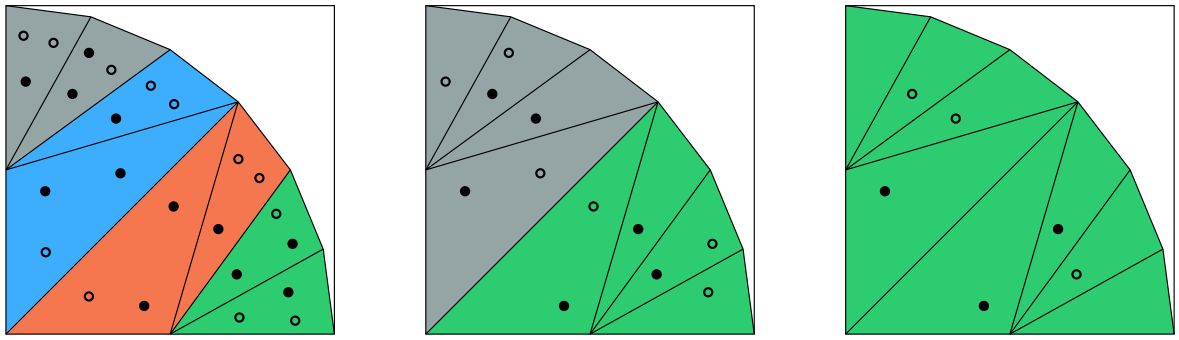


Figure 16: Different levels of recursion of the base quadrature mesh and associated quadrature rules. Circles represent the set of tentative points at each level. Filled circles are chosen by the non-negative least-square algorithm.

see the colours in figure 16 a. The quadrature points on the individual elements in the partition are used to construct a set of non-negative quadrature rules (one for each group of elements corresponding to a partition). The number of elements in the partitions of this first level is chosen so that the moment-fitting system is small and therefore solved efficiently by `nnls` (see previous section). In a second step the partitions of the first level are used to define second level partitions (see the colours in figure 16 b). Again, the union of the set of points obtained at the previous step satisfies the moment-fitting equations, making possible to use these points to find a sparse positive solution to the equation. The process is reproduced recursively until the final quadrature is obtained. By means of this strategy, it is possible to construct efficiently non-negative moment-fitting quadrature rules without solving any large problem and with a provable convergence.

5.3. Cost comparison for the overall construction of moment-fitting rules

In this section, we monitor the efficiency of the two methods proposed in section 5.2. The whole construction process is considered here, not only the resolution of the moment-fitting equations as done in section 4.2.1. We monitor the construction time for the three geometries presented in figure 10 in a

Geometry	Elts. in tesselation	$r_{\text{classical}}$	$t_{\text{adapt.}}/t_{class.}$	$t_{\text{recur.}}/t_{class.}$
Quart. circle	4308	9.78	11.49	3.02
Quart. circle coarse	43	1.17	5.67	7.49
Triangle	3208	7.27	10.49	3.65
Corners	1622	3.68	1.21	3.58
Sliver	1708	3.87	1.15	3.15

Table 5: Performance assessment for adaptive and recursive constructions of the quadrature rules. $t_{class.}$ corresponds to the classical moment-fitting approach, $t_{\text{adapt.}}$ to the adaptative algorithm from section 5.2.1 and $t_{\text{recur.}}$ to the recursive algorithm from section 5.2.2.

single threaded context. Note that all the algorithms are centred around the use of the tessellations for integrating the right-hand side of the moment-fitting equations and choosing the quadrature points. In particular: in the case of classical moment-fitting, the set of quadrature points is obtained by raising the order of the base quadrature rule (on the tessellation) until $r > 1$. This approach avoids the construction of voxelizations of the geometry as presented in [32]. The downside is that the number of quadrature points can increase rapidly and may lead to unnecessary large r. The same strategy is used for the adaptive algorithm 1. Thanks to Tchakaloff's upper bound, we know that a solution will be found at the latest when the order of the base rule reaches the order of the requested moment-fitting rule.

Order 20 quadrature rules are constructed using local tessellations built from a space-tree with 8 levels of recursion, and results are presented in table 5. Numbers show that the proposed point placement method is not optimal for classical moment-fitting as the number of quadrature points can be quite large ($r > 1$). It is however well adapted to the non-negative moment-fitting approach as it makes the recursive method possible and renders the increase of the tentative set easy in the adaptive algorithm. We can also see that in this case, the overhead due to the non-negative construction of the rule can be restrained with the use of the two proposed algorithms. From the different testcases, the recursive algorithm led to the most consistent results although it can be outperformed by the adaptive one in some cases.

6. Application to realistic testcases

Non-negative moment fitting quadrature rules are now applied to fictitious domain mechanical test-cases. Both linear elastic and elasto-plastic problems will be considered, with focus on the influence of the quadrature rules on the accuracy of the solution, and its robustness. Like with the verifications presented in the previous section, a set of n_{qt} tentative quadrature points is generated in the matter part of each element. The non-negative moment fitting approach presented in section 3.5 is then applied to select n_q quadrature points among the tentative points, and the associated *positive* weights. As presented in the last sections, an insufficient number of tentative quadrature points can lead to a failure of the non-negative least square solver. To ensure that the quadrature rule has a proper accuracy, the recursive strategy from section 5.2.2 is considered. In addition, it is worth noting that in this section the right-hand sides of the moment fitting equations are integrated on the quadtree quadrature sub-grid. The cost of this procedure is moderate as it corresponds to the integration of standard scalar valued functions, which is far less costly than integrating finite element quantities. In the linear elastic case, this part could be speeded-up by the use of the divergence theorem which enables to integrate on lower dimensional domains [32, 26, 25, 27]. In the elasto-plastic case however, this approach cannot be considered.

6.1. Linear elasticity

We first present a common verification example: consider an infinite plate with a circular hole in its centre which is loaded in uni-axial tension $\sigma_\infty = 1.0\,\text{MPa}$ along $\mathbf{x}$ axis (see Figure 17). The analytical

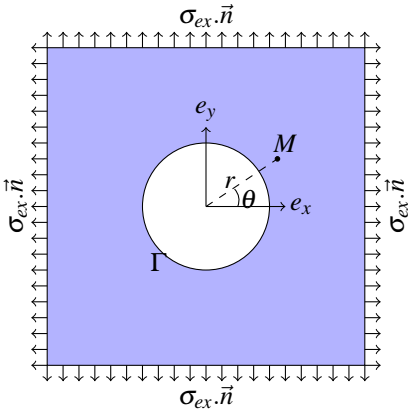
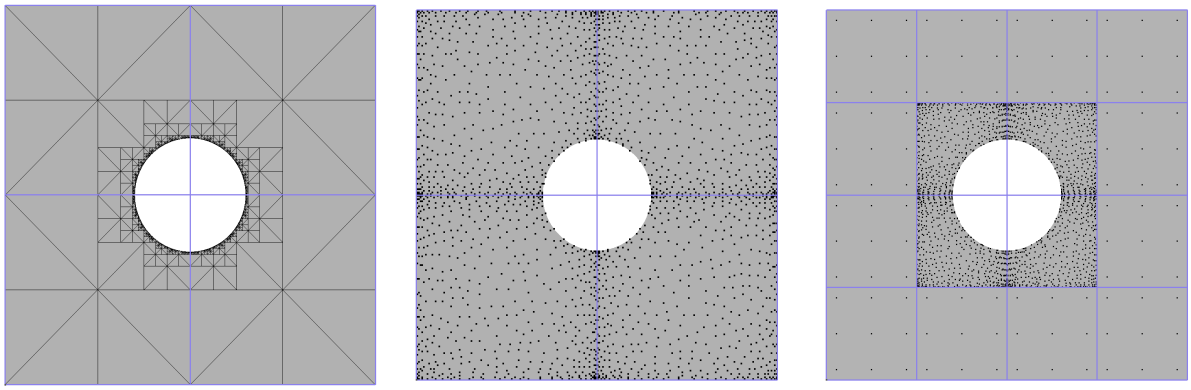


Figure 17: Infinite plate with a hole subjected to uni-axial tension: geometry and approximation mesh.

solution of this problem is given in [9] and is used here as a reference solution for error monitoring. A square of length $L = 2.0\,\text{mm}$ with a circular hole of radius $a = 0.3\,\text{mm}$ at its centre is considered. Finally, tractions computed from the exact solution are applied on the boundary of the domain, and three degrees of freedom (dofs in the following) are prescribed to zero in order to avoid rigid body motion. The exact solution of this finite problem is therefore the same as the exact solution inside the boundary. Young's modulus is set to 1.0 MPa, and Poisson's ratio to 0.3. A p-convergence study is performed for $p = 1$ to 12, and the error in the energy norm is monitored with respect to the number of dofs and quadrature points. The approximation mesh is composed of 4×4 quadrangular elements with Legendre hierarchical shape functions. A sub-grid quadrature rule is considered here [5, 18, 16], even if more efficient rules such as [19] are also available. After adaptation, the geometrical mesh size h_g is set to $h_g = h/128$ near the interface if h is the mesh size of the approximation mesh, see figure 18a. The 2116 quadrature points obtained by non-negative moment fitting for a quadrature order $p_q = 22$ are shown in figure 18b for a 2×2 approximation mesh (529 quadrature points per element). The case of the 4×4 mesh is also shown in figure 18 c: one can observe that a quadrature of order $p_q = 22$ is used in the cut elements whereas a tensor product quadrature rule of order $p_q = 2$ is used in the other elements ($p = 2$ for this particular illustration). This is due to the fact that it is more efficient to generate a quadrature rule once for all in the cut elements and use it for any p order used in the approximation. On the contrary, the quadrature rule is linked to the approximation order in the other elements. This strategy may seem inefficient, but it is not the case as the use of a sub-grid quadrature rule of order 1 already leads to 8892 quadrature points, whereas the one presented in figure 18c (which is accurate up to order 22 in the partial elements) involves less quadrature points (2224).

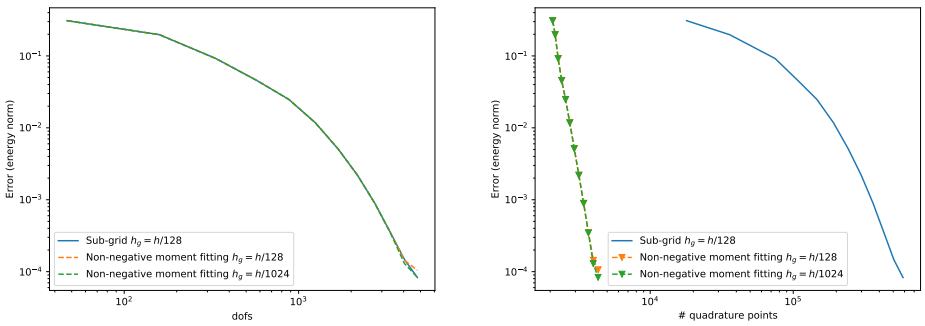
The evolution of the error in the energy norm with respect to the number of degrees of freedom is plotted in figure 19a for the mesh and quadrature points depicted in figure 18c. As expected, the convergence is exponential in the pre-asymptotic range, then levels-off for the coarser geometrical approximation as the geometrical error becomes to dominate. The error is plotted as a function of the number of quadrature points in figure 19b: it can be seen that, as expected, the non-negative moment fitting strategy is clearly more efficient than the sub-grid quadrature rule in terms of accuracy per quadrature point. The small increase of the number of quadrature points during the p convergence stems from the dependence of the quadrature rules to the polynomial order in the elements that are fully in the matter. It is also important to note that increasing the depth of the quadtree to improve the geometrical accuracy has no influence on the number of quadrature points, as they solely depend on the order of the quadrature rule (and not to the amount of integration cells in the geometrical mesh).

One should acknowledge that setting-up the quadrature rule has a cost. In the example depicted in figure 18c, the cost of a unique calculation with $p = 12$ (including the setup of the quadrature rule) is already 57% lower than the cost of a calculation with sub-grid quadrature. Most of this time is



a. b. c.

Figure 18: Plate with a hole: a. approximation mesh (blue lines) and integration cells ($h_g = h/128$); b. Quadrature points of order $p_q = 22$, 2×2 approximation mesh; c. Quadrature points of order $p_q = 22$ in the partially cut elements, and usual tensor product quadrature rule of order $p_q = 2$ elsewhere (4×4 approximation mesh)



a. b.

Figure 19: Plate with a hole: convergence. a. With respect to the number of dofs; b. With respect to the number of quadrature points.

Young's modulus [Pa]	$217.5\,10^9$
Poisson coefficient [-]	0.3
Hardening modulus [Pa]	$5\,10^9$
Yield stress [Pa]	$250\,10^6$

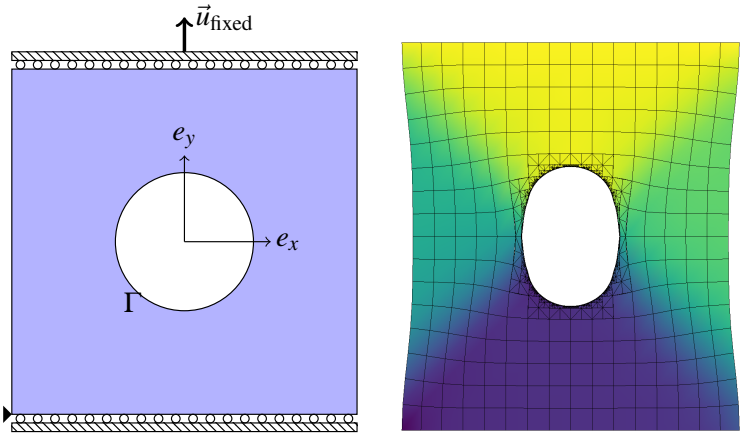
Table 6: Elasto-plastic material coefficients.

spent in the construction of the quadrature rule (98.8%). This means that the assembly phase can be quickly amortized if multiple assembly is necessary. Indeed, this is the case here as we are working in an adaptive environment: if the whole p convergence is considered as the timing of interest, then the non-negative moment fitting quadrature strategy is 571% faster than the sub-grid quadrature.

6.2. Elasto-plastic problems

The advantage of having positive quadrature rules is more visible for nonlinear problems, as the stability of the quadrature rule can have a noticeable influence on the convergence of the iterative algorithms (see section 2). Moreover, this application involves multiple system assembly even at fixed p: we expect the non-negative moment fitting strategy to be significantly faster than the sub-grid quadrature approach. We consider elasto-plastic problems with von Mises yield criterion (J_2) and linear isotropic hardening under the small strain assumption. The material properties used for all the examples are summarized in table 6.

6.2.1. Plate with a hole



a. b.

Figure 20: Plate with a hole subjected to uniaxial tension: a. geometry; b. final displacement.

The first elasto-plastic case uses the geometry of the previous section. The plate (1m×1m) is now subjected to a uniaxial traction in the vertical direction by means of a prescribed displacement on its top boundary (10 mm) (see figure 20a.). This top displacement is imposed gradually in 100 loading steps, leading to the deformed configuration depicted in figure 20b. A reference solution is computed using a linear approximation on a grid involving 256 elements along each direction (122 917 dofs). High order approximations on a 16×16 grid with $p = 2, 4$ and 8 are considered in order to monitor the convergence of the solution (resp 1 557, 4 093 and 15 021 dofs). The geometrical accuracy is fixed to $h_g = h/64$. The final plastic zone for $p = 8$ is presented in figure 21 for both sub-grid and non-negative moment fitting. A very good agreement can be observed although the first involves 585 600 integration points versus

only 26 734 for non-negative moment fitting. It is interesting to note here that the Newton-Raphson iterative process was not able to converge when using classical moment-fitting quadratures (with points inside the physical domain) due to the negativity of the weights. As highlighted in section 4.2, an increase of the amount of quadrature point tends to lead to positive rules which, in this case, allowed the Newton-Raphson algorithm to converge properly.

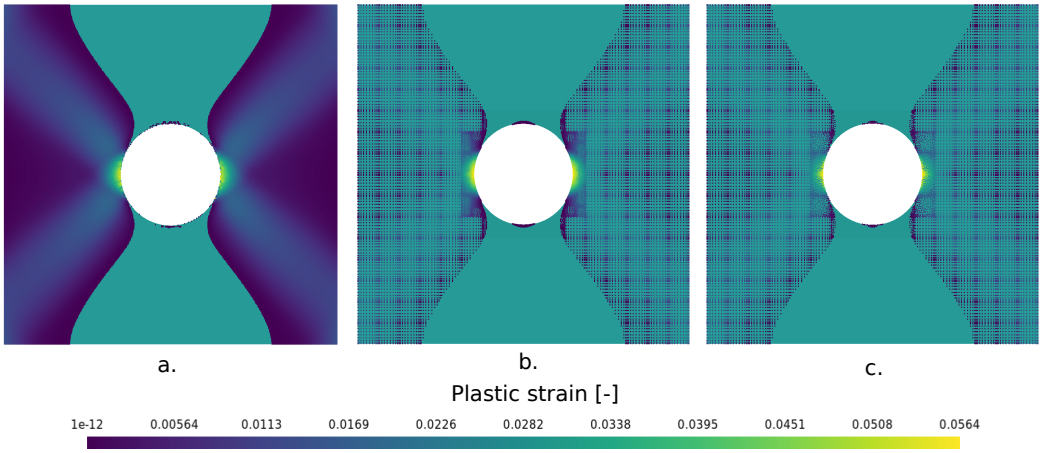
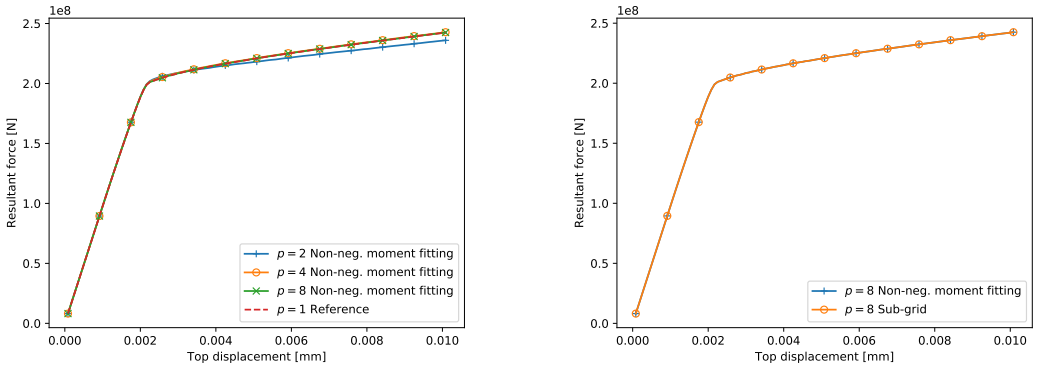


Figure 21: Plate with a hole subjected to uniaxial tension: Plastic strain. a. Reference linear computation; b. $p = 8$ sub-grid; c. $p = 8$ non-negative moment fitting. The grid pattern in b. and c. is linked to the structured nature of the underlying mesh.

The force-displacement curves are compared in figure 22. One can observe the convergence of the high-order results towards the reference solution when p increases (figure 22a). Also, the high order solution is as efficient as the linear solution as early as $p = 4$, with 30 times less dofs. Perfect accordance between sub-grid and non-negative moment fitting is also observed (figure 22b).

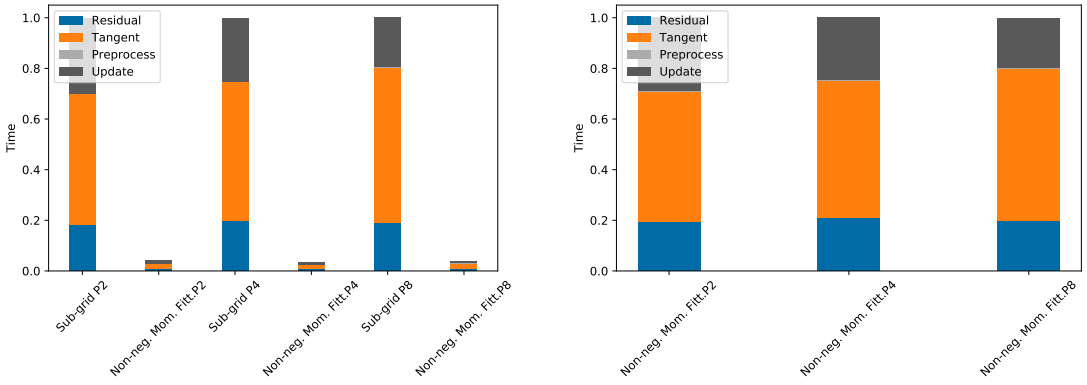


a. b.

Figure 22: Plate with a hole subjected to uniaxial tension: Force-displacement curve. a. comparison with the reference computation; b. comparison between sub-grid and non-negative moment fitting.

The costs of the computations are also compared in figure 23a where it can be observed that a tremendous amount of time can be saved thanks to the small number of quadrature points involved.

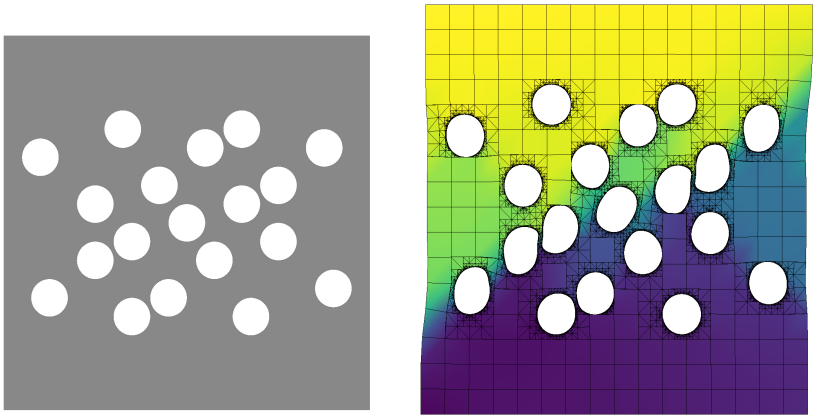
Looking more closely at the moment-fitting computations (figure 23b), we can conclude (i) that the preprocessing time is of negligible influence in the timings and (ii) that increasing the polynomial order mainly leads to an increase of the relative cost of the assembly of the finite element operators.



a. b.

Figure 23: Plate with a hole subjected to uniaxial tension: Timing comparison. Residual, Tangent, Preprocess and Update stand respectively for the cumulative time for the residual computation, the assembly of the tangent operator, the moment fitting processing time and the update of the material variables at the quadrature points. a. With respect to sub-grid (normalized time for each polynomial order); b. non-negative moment fitting, influence of the polynomial order (normalized time for each polynomial order).

6.2.2. Plate with multiple holes



a. b.

Figure 24: Plate with multiple holes subjected to uniaxial tension: a. Geometry; b. Final displacement.

We finish with an example involving multiple interfaces: a 2m×2m plate containing 19 holes of radius 100 mm (see figure 24a). It is still subjected to a uniaxial traction by means of a displacement of 10 mm on the top boundary, similar to figure 20a. This top displacement is imposed gradually in 120 loading steps, leading to the deformed configuration depicted in figure 24b. A reference solution is

computed using a quadratic computation on a grid involving 256 elements along each direction. High order approximations on a 16×16 grid with $p = 2, 4$ and 8 are considered in order to monitor the convergence of the solution (resp 1 599, 4 223 and 15 615 dofs). The geometrical accuracy is fixed to $h_g = h/64$. The final plastic zone for $p = 8$ is presented in figure 25 for both reference and non-negative moment fitting. Like for the last example, a very good agreement can be observed although the first computation involves 903 988 integration points versus only 35 109 for non-negative moment fitting.

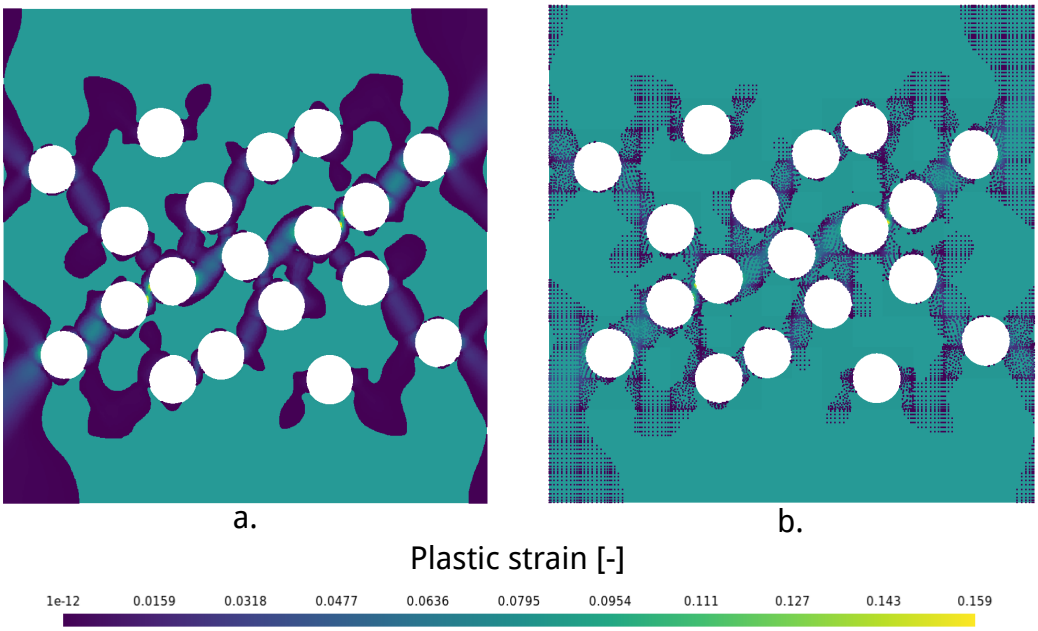
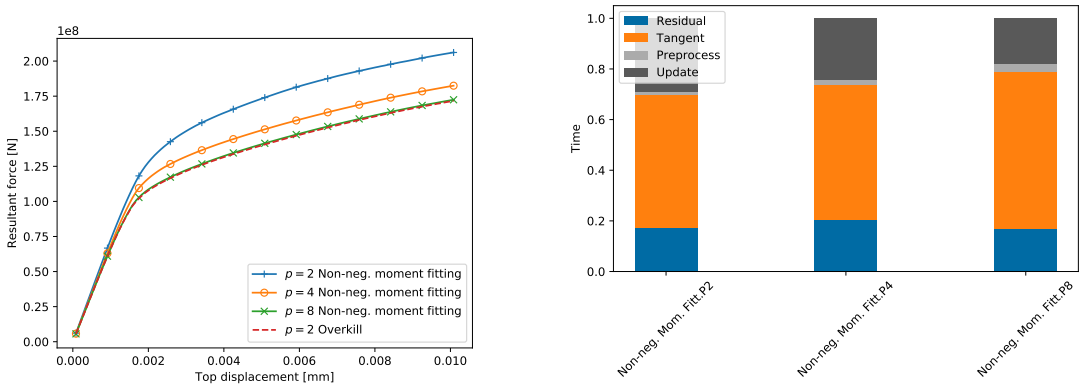


Figure 25: Plate with multiple holes subjected to uniaxial tension: Plastic strain. a. Reference quadratic computation; b. $p = 8$ non-negative moment fitting. The grid pattern in b. is linked to the structured nature of the underlying mesh.

The evolution of the force-displacement curve with increasing polynomial order is also presented in figure 26a. For this more complicated case, a 8^{th} order approximation is necessary to match the reference solution. Timing comparisons which are presented in figure 26b show similar trends compared to the previous example. One can yet notice a slight increase of the contribution of the preprocessing step due to the larger number of partitioned elements. The total computation time is 1.8 times faster than the reference computation.

7. Conclusions and Outlook

In this contribution, we proposed a strategy to build sparse and positive moment-fitting quadrature rules. The accuracy of this strategy was compared to various quadrature rule families in the context of fictitious domain. Emphasis was put on the positivity of the quadratures weights and the use of quadrature points strictly inside the physical domain so that the rules can remain robust in the context of nonlinear problems. Four methods were compared: (i) classical moment-fitting; (ii) quality oriented quadrature rules with positivity constraint; (iii) empirical quadrature rules and (iv) non-negative moment-fitting quadrature rules. We have shown that classical moment-fitting quadrature rules cannot lead to positive weights unless a very large number of quadrature points are considered. The negativity of the weights is usually not an issue for linear problems, but can be problematic with nonlinear computations. We also highlighted that the location of the quadrature points can have a



a. b.

Figure 26: Plate with multiple hole subjected to uni-axial tension. a. Convergence of the force-displacement curve with increasing polynomial order; b. Timing comparison (normalized time for each polynomial order). Residual, Tangent, Preprocess and Update stand respectively for the cumulative time for the residual computation, the assembly of the tangent operator, the moment fitting processing time and the update of the material variables at quadrature points.

large influence on the accuracy with non-polynomial integrands for the moment-fitting quadratures (classical and non-negative). This influence can be mitigated by increasing the number of quadrature points. Quality oriented quadrature rules are also proved to converge to positive weights, and were shown to lead to a solid accuracy. However, like for classical moment fitting, a larger number of quadrature points has to be considered to satisfy positivity. Positivity was thus difficult to attain for geometries representative of real cases. Empirical quadrature rules are robust and lead to the minimal number of quadrature points. Unfortunately, it was not possible to enforce the positivity of the weights which limits its use to linear problems. Finally, non-negative quadrature rules are sparse and accurate, provided that the number of tentative quadrature points is sufficient. Yet, the cost of building these quadrature rules is larger compared to classical moment-fitting (less than one order of magnitude slower in our 2D experiments), even considering the optimized strategies proposed in section 5. Fortunately, this additional cost can be amortized in the non-linear setting.

Applications of the non-negative quadrature rules to two-dimensional linear and nonlinear mechanical problems have shown that the method is very efficient, and that the overhead induced by the setup of the quadrature rule for each cut element was not significant with respect to the whole calculation process. Future work on this topic will focus on 3D elasto-plastic calculations based on scanned data, and the proper choice of efficient algorithms for solving the non-negative least square problem in case where the size of the system becomes significant.

A. Algorithms description

Data: $\mathcal{G}(\cdot;\boldsymbol{\mu})$, $\boldsymbol{\mu} \in \boldsymbol{\Xi}$, set of tentative points $\{\mathbf{x}_k\}$, $k = 1, \ldots, N$
Result: EIM Magic points $\mathbf{x}_i^\star$ and basis $q_i(\mathbf{x})$, $i = 1, \ldots, M$
1 `// Select the worse generating function`
2 $\boldsymbol{\mu}_1 = \arg\max_{\mu \in \boldsymbol{\Xi}} \|\mathcal{G}(\cdot;\boldsymbol{\mu})\|_{L^\infty(\Omega)}$
3 `// Select the worse tentative point`
4 $\mathbf{x}_1^\star = \arg\max_{\mathbf{x} \in \{\mathbf{x}_k\}} |\mathcal{G}(\mathbf{x};\boldsymbol{\mu}_1)|$
5 `// Construct the interpolation function`
6 $q_1(\mathbf{x}) = \mathcal{G}(\cdot;\boldsymbol{\mu}_1)/\mathcal{G}(\mathbf{x}_1^\star;\boldsymbol{\mu}_1)$
7 `// EIM Greedy loop`
8 **for** $m = 2$ **to** M **do**
9 `// Define` $\mathcal{I}_{m-1}[\mathcal{G}(\cdot;\boldsymbol{\mu})]$ `by solving for` $\boldsymbol{\beta}(\boldsymbol{\mu})$
10 $\boldsymbol{B}^{\boldsymbol{M}}\boldsymbol{\beta}(\boldsymbol{\mu}) = \mathcal{G}(\mathbf{x}^\star;\boldsymbol{\mu})$ with $\boldsymbol{B}^{\boldsymbol{M}}_{\boldsymbol{ij}} = q_j(\mathbf{x}_i^\star)$
11 `// Next` $\boldsymbol{\mu}_m$ `as the one associated to the worse parametric error`
12 $\boldsymbol{\mu}_m = \arg\max_{\mu \in \boldsymbol{\Xi}} \|\mathcal{G}(\cdot;\boldsymbol{\mu}) - \mathcal{I}_{m-1}[\mathcal{G}(\cdot;\boldsymbol{\mu})]\|_{L^\infty(\Omega)}$
13 `// Next` $\mathbf{x}_m$ `as the one associated to the worse spatial error`
14 $\mathbf{x}_m^\star = \arg\max_{\mathbf{x} \in \{\mathbf{x}_k\}} |\mathcal{G}(\mathbf{x};\boldsymbol{\mu}_m) - \mathcal{I}_{m-1}[\mathcal{G}(\cdot;\boldsymbol{\mu}_m)](\mathbf{x})|$
15 `// Construct the interpolation function`
16 $q_m(\mathbf{x}) = (\mathcal{G}(\cdot;\boldsymbol{\mu}_m) - \mathcal{I}_{m-1}[\mathcal{G}(\cdot;\boldsymbol{\mu}_m)]) \,/\, (\mathcal{G}(\mathbf{x}_m^\star;\boldsymbol{\mu}_m) - \mathcal{I}_{m-1}[\mathcal{G}(\cdot;\boldsymbol{\mu}_m)](\mathbf{x}_m^\star))$
17 **end**

Algorithm 2: Greedy construction of the EIM interpolation of $\mathcal{G}(\cdot;\boldsymbol{\mu})$

Data: Quadrature rule order p_q, Quality threshold Q^{lim}
Result: Quality oriented quadrature points $\{\mathbf{x}_q\}_{q=1,\ldots,n_q}$ and weights $\{w_q\}_{w=1,\ldots,n_q}$
1 `// Initial number of quadrature points`
2 Initialize $\{\mathbf{x}_q\}$ with $n_q = 0$ the number of quadrature points
3 `// Quadrature space (Legendre basis)`
4 Initialize $\mathcal{H}$ with $n_b = \dim(\mathcal{H})$
5 `// Basis for the EIM algorithm (monomial basis)`
6 Initialize $\mathcal{W}$ with $n_p = \dim(\mathcal{W}) = n_b$
7 `// Quality loop`
8 **while** $Q > Q^{lim}$ **do**
9 `// Select a new EIM magic point`
10 Compute a new magic point $\mathbf{x}^\star$ according to Algorithm 2
11 $\{\mathbf{x}_q\} \leftarrow \mathbf{x}^\star$
12 `// Enlarge` $\mathcal{W}$ `if needed`
13 **if** $n_q > 0.75 n_p$ **then**
14 $n_p += 1$
15 Update $\mathcal{W}$
16 **end**
17 **if** $n_q \geq n_b$ **then**
18 `// Compute the weights`
19 Compute the weights according to (14)
20 **end**
21 `// Compute the quality of the quadrature rule`
22 Compute Q according to (15)
23 **end**

Algorithm 3: Quality oriented quadrature strategy from [29]. Note that in this contribution, the test on line 8 was replaced by "$Q > Q^{lim}$ and $\min\{w_q\} \geq 0$" as we are interested in *positive* quadrature rules.

Data: Moment-fitting matrix and right-hand side $\boldsymbol{A}$ (dimension $m \times n$), $\boldsymbol{x}$ (dimension n), tolerance ϵ

Result: Non-negative least square solution $\boldsymbol{x}$ of $\boldsymbol{Ax} = \boldsymbol{b}$

```
// Initialization
```
$P = \emptyset$ (passive set, free indices)

$R = \{1, \ldots, n\}$ (active set, fixed to zero)

$\boldsymbol{x}$, feasible all zero vector of dimension n

Set $\boldsymbol{v} = \boldsymbol{A}^T(\boldsymbol{b} - \boldsymbol{Ax})$ (Lagrange multipliers)

```
// Main loop
```
while $R \neq \emptyset$ and $\max(\boldsymbol{v}) > \epsilon$ **do**

  $j = \arg\max_{n \in R}(\boldsymbol{v})$

  $P = P \cup \{j\}$

  $R = R \setminus \{j\}$

  ```
  // Restrict A to the passive set of unknowns
  ```
  $\boldsymbol{A}^P = [\boldsymbol{A}_{ij}]_{j \in P}$

  ```
  // Solve the restricted least square problem
  ```
  $\boldsymbol{s}$ vector of dimension n

  $\boldsymbol{A}^P \boldsymbol{s}^P = \boldsymbol{b}$

  ```
  // Set active set variables to zero
  ```
  $\boldsymbol{s}^R = 0$

  ```
  // Inner loop
  ```
  while $\min(\boldsymbol{s}^p) \leq 0$ **do**

    $\alpha = \min\left(\frac{x_i}{x_i - s_i}\right)$ for $i \in P$ where $s_i \leq 0$

    Set $\boldsymbol{x} = \boldsymbol{x} + \alpha(\boldsymbol{s} - \boldsymbol{x})$

    ```
    // Update R and P
    ```
    $R = R \cup \{j\}$ for j such as $x_j = 0$

    $P = P \setminus \{j\}$ for j such as $x_j = 0$

    ```
    // Solve the restricted least square problem
    ```
    $\boldsymbol{A}^P \boldsymbol{s}^P = \boldsymbol{b}$

    ```
    // Set active set variables to zero
    ```
    $\boldsymbol{s}^R = 0$

  end

  ```
  // Update x
  ```
  $\boldsymbol{x} = \boldsymbol{s}$

  ```
  // Update Lagrange multipliers
  ```
  $\boldsymbol{v} = \boldsymbol{A}^T(\boldsymbol{b} - \boldsymbol{Ax})$

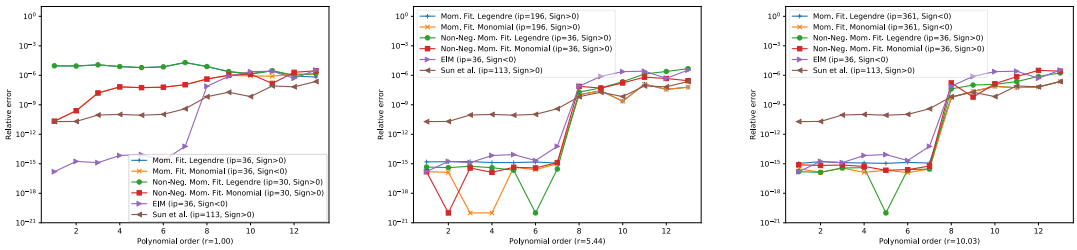
end

Algorithm 4: Non-negative least square algorithm from [47].

B. Additional 2D benchmarks

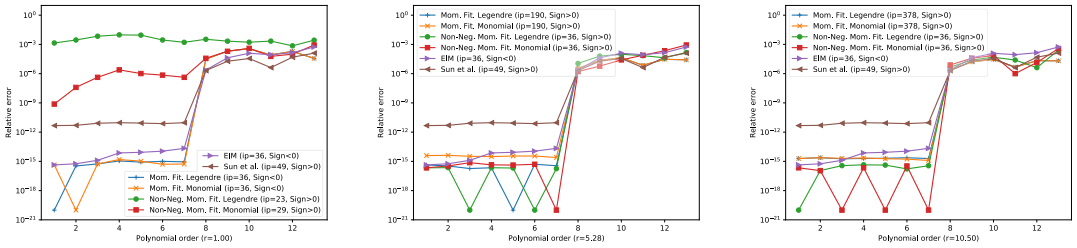
The benchmarks mentioned in the introduction of section 4.2 are presented here for the sake of completness.

B.1. Polynomial integrands



$n_q \simeq n_{\text{basis}}$ $n_q \simeq 5\, n_{\text{basis}}$ $n_q \simeq 10\, n_{\text{basis}}$

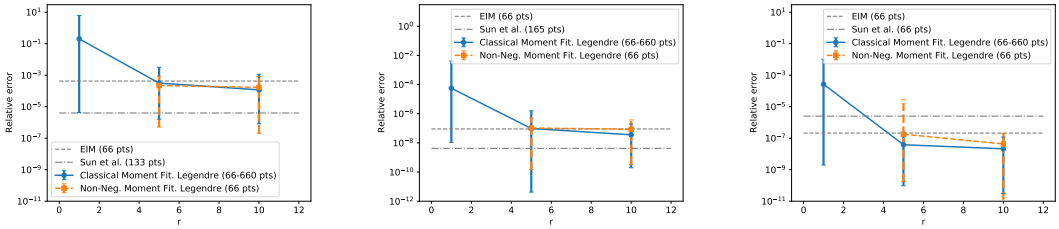
Figure 27: Accuracy for polynomial integrands of increasing order (order 7 rules) for the corner geometry (points located inside the domain).



$n_q \simeq n_{\text{basis}}$ $n_q \simeq 5\, n_{\text{basis}}$ $n_q \simeq 10\, n_{\text{basis}}$

Figure 28: Accuracy for polynomial integrands of increasing order (order 7 rules) for the triangular geometry (points located inside the domain). Refer to figure 10 for the geometry.

B.2. Non-polynomial integrands



Triangle Quarter circle Corner

Figure 29: 2D scattering: Relative errors for 1000 evaluations with $n_q = rn_{\text{basis}}$ random quadrature points. Average and dispersion of the resulting valid moment-fitting rules are depicted. EIM and quality oriented moment fitting error are depicted as constant dashed lines (as they operate with already fixed quadrature points). Integrand: $\sin(\pi x)\sin(3\pi y)$ for triangular and corner geometry; $\frac{1}{25}(r^5 - 1) + r^2 cos(2\theta)$ for the quarter circle. Refer to figure 10 for the geometries.

References

[1] A. Huerta, T. Belytschko, and T. Rabczuk. Meshfree Methods. pages 1–49. 2004.

[2] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl. Meshless methods: An overview and recent developments. Computer Methods in Applied Mechanics and Engineering, 139:3–47, 1998.

[3] J. Cottrell, T. J. R. Hughes, and Y. Bazilevs. Isogeometric Analysis. 2009.

[4] V. K. Saulèv. On the solution of some boundary value problems on high performance computers by fictitious domain method. Siberian Math. J., 4:912–925, 1963.

[5] J. Parvizian, A. Düster, and E. Rank. Finite cell method. Computational Mechanics, 41(1):121–133, 2007.

[6] A. Düster, J. Parvizian, Z. Yang, and E. Rank. The finite cell method for three-dimensional problems of solid mechanics. Computer Methods in Applied Mechanics and Engineering, 197(45):3768–3782, August 2008.

[7] E. Rank, S. Kollmannsberger, C. Sorger, and A. Düster. Shell Finite Cell Method: A high order fictitious domain approach for thin-walled structures. Computer Methods in Applied Mechanics and Engineering, 200(45-46):3200–3209, October 2011.

[8] N. Moës, J. E. Dolbow, and T. Belytschko. A finite element method for crack growth without remeshing. International Journal for Numerical Methods in Engineering, 46:131–150, 1999.

[9] N. Sukumar, D. L. Chopp, N. Moës, and T. Belytschko. Modeling holes and inclusions by level sets in the extended finite-element method. Computer Methods in Applied Mechanics and Engineering, 190(46-47):6183–6200, 2001.

[10] G. Legrain, C. Geuzaine, J. Remacle, N. Moës, P. Cresta, and J. Gaudin. Numerical simulation of CAD thin structures using the eXtended Finite Element Method and Level Sets. Finite Elements in Analysis and Design, 77, 2013.

[11] B. Wassermann, S. Kollmannsberger, S. Yin, L. Kudela, and E. Rank. Integrating CAD and numerical analysis: 'Dirty geometry' handling using the Finite Cell Method. Computer Methods in Applied Mechanics and Engineering, 351:808–835, July 2019.

[12] G. Legrain, P. Cartraud, I. Perreard, and N. Moës. An X-FEM and level set computational approach for image-based modelling: Application to homogenization. International Journal for Numerical Methods in Engineering, 86(7):915–934, 2011.

[13] A. Düster, H.-G. Sehlhorst, and E. Rank. Numerical homogenization of heterogeneous and cellular materials utilizing the finite cell method. Computational Mechanics, 50(4):413–431, January 2012.

[14] S. Heinze, M. Joulaian, and A. Düster. Numerical homogenization of hybrid metal foams using the finite cell method. Computers & Mathematics with Applications, 2015.

[15] G. Legrain, M. Chevreuil, and N. Takano. Prediction of apparent properties with uncertain material parameters using high-order fictitious domain methods and PGD model reduction. International Journal for Numerical Methods in Engineering, 109(3), 2017.

[16] K. Dréau, N. Chevaugeon, and N. Moës. Studied X-FEM enrichment to handle material interfaces with higher order finite element. Computer Methods in Applied Mechanics and Engineering, 199(29-32):1922–1936, June 2010.

[17] S. Groß and A. Reusken. An extended pressure finite element space for two-phase incompressible flows with surface tension. Journal of Computational Physics, 224(1):40–58, May 2007.

[18] G. Legrain, N. Chevaugeon, and K. Dréau. High order X-FEM and levelsets for complex microstructures: Uncoupling geometry and approximation. Computer Methods in Applied Mechanics and Engineering, 241-244:172–189, October 2012.

[19] G. Legrain and N. Moës. Adaptive anisotropic integration scheme for high-order fictitious domain methods: Application to thin structures. International Journal for Numerical Methods in Engineering, 2018.

[20] T. Strouboulis, K. Copps, and I. Babuška. The generalized finite element method. Computer Methods in Applied Mechanics and Engineering, 190(32-33):4081–4193, May 2001.

[21] G. Legrain. A NURBS enhanced extended finite element approach for unfitted CAD analysis. Computational Mechanics, 52(4):913–929, April 2013.

[22] L. Kudela, N. Zander, S. Kollmannsberger, and E. Rank. Smart octrees: Accurately integrating discontinuous functions in 3D. Computer Methods in Applied Mechanics and Engineering, 306:406–426, July 2016.

[23] T.-p. Fries, S. Omerović, D. Schöllhammer, and J. Steidl. Higher-order meshing of implicit geometries—Part I: Integration and interpolation in cut elements. Computer Methods in Applied Mechanics and Engineering, 2016.

[24] S. Duczek and U. Gabbert. Efficient integration method for fictitious domain approaches. Computational Mechanics, 56(4):725–738, 2015.

[25] S. E. Mousavi and N. Sukumar. Numerical integration of polynomials and discontinuous functions on irregular convex polygons and polyhedrons. Computational Mechanics, 47(5):535–554, December 2010.

[26] Y. Sudhakar and W. A. Wall. Quadrature schemes for arbitrary convex/concave volumes and integration of weak form in enriched partition of unity methods. Computer Methods in Applied Mechanics and Engineering, 258:39–54, May 2013.

[27] B. Müller, F. Kummer, and M. Oberlack. Highly accurate surface and volume integration on implicit domains by means of moment-fitting. International Journal for Numerical Methods in Engineering, 96(8):512–528, November 2013.

[28] Y. Maday, N. C. Nguyen, A. T. Patera, and G. S. H. Pau. A general multipurpose interpolation procedure: The magic points. Communications on Pure and Applied Analysis, 8(1):383–404, 2009.

[29] H. Sun and D. L. Darmofal. An adaptive simplex cut-cell method for high-order discontinuous Galerkin discretizations of elliptic interface problems and conjugate heat transfer problems. Journal of Computational Physics, 278:445–468, 2014.

[30] D. Huybrechs. Stable high-order quadrature rules with equidistant points. Journal of Computational and Applied Mathematics, 231(2):933–947, September 2009.

[31] B. Szabó and I. Babuška. Finite Element Analysis. John Wiley & Sons, first edition, 1991.

[32] M. Joulaian, S. Hubrich, and A. Düster. Numerical integration of discontinuities on arbitrary domains based on moment fitting. Computational Mechanics, 4(C), March 2016.

[33] H.-G. Bui, D. Schillinger, and G. Meschke. Efficient cut-cell quadrature based on moment fitting for materially nonlinear analysis. Computer Methods in Applied Mechanics and Engineering, page 30.

[34] S. Hubrich and A. Düster. Numerical integration for nonlinear problems of the finite cell method using an adaptive scheme based on moment fitting. Computers & Mathematics with Applications, December 2018.

[35] V. Tchakaloff. Formules de cubatures mécaniques à coefficients non négatifs. Bull. Sci. Math, 81(2):123–134, 1957.

[36] P. J. Davis. Approximate integration rules with nonnegative weights. Lectures in differential equations, 2:233–256, 1969.

[37] H. Antil, S. E. Field, F. Herrmann, R. H. Nochetto, and M. Tiglio. Two-step greedy algorithm for reduced order quadratures. Journal of Scientific Computing, 57(3):604–637, 2013.

[38] G. ROZZA. An Introduction To Reduced Basis Method For Parametrized PDEs. Applied and Industrial Mathematics in Italy III - Selected Contributions from the 9th SIMAI Conference, pages 508–519, 2010.

[39] S. Kristoffersen. The Empirical Interpolation Method. PhD thesis, 2013.

[40] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. LAPACK Users' Guide. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[41] H. Sun. A RObust Simplex Cut-Cell Method for Adaptive High-Order Discretizations of Aerodynamics and Multi-Physics Problems. PhD thesis, 2013.

[42] P. J. Davis. A Construction of Nonnegative Approximate Quadratures. Mathematics of Computation, 21(100):578–582, 1967.

[43] M. Wilson. Necessary and Sufficient Conditions for Equidistant Quadrature Formula. SIAM Journal on Numerical Analysis, 7(1):134–141, March 1970.

[44] C. Lin and J. Moré. Newton's Method for Large Bound-Constrained Optimization Problems. SIAM Journal on Optimization, 9(4):1100–1127, January 1999.

[45] R. Byrd, P. Lu, J. Nocedal, and C. Zhu. A Limited Memory Algorithm for Bound Constrained Optimization. SIAM Journal on Scientific Computing, 16(5):1190–1208, September 1995.

[46] E. Steinitz. Bedingt konvergente Reihen und konvexe Systeme. Journal für die reine und angewandte Mathematik, 143:128–176, 1913.

[47] C. L. Lawson and R. J. Hanson. Solving Least Squares Problems, volume 15. Siam, 1995.

[48] L. Wu, Y. Yang, and H. Liu. Nonnegative-lasso and application in index tracking. Computational Statistics & Data Analysis, 70:116–126, February 2014.

[49] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python. 2001.

[50] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, Š. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, and A. Scopatz. SymPy: Symbolic computing in Python. PeerJ Computer Science, 3:e103, January 2017.

[51] W. W. Hager and H. Zhang. A New Active Set Algorithm for Box Constrained Optimization. SIAM Journal on Optimization, 17(2):526–557, January 2006.

[52] D. Kim, S. Sra, and I. S. Dhillon. A non-monotonic method for large-scale non-negative least squares. Optimization Methods and Software, 28(5):1012–1039, October 2013.

[53] Y. Luo and R. Duraiswami. Efficient Parallel Nonnegative Least Squares on Multicore Architectures. SIAM Journal on Scientific Computing, 33(5):2848–2863, January 2011.