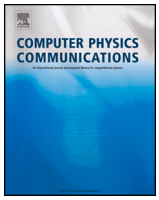
Quadrature of functions with endpoint singular and generalised polynomial behaviour in computational physics

Computer Physics Communications 299 (2024) 109124

Contents lists available at ScienceDirect

Computer Physics Communications

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



Computer Programs in Physics

Quadrature of functions with endpoint singular and generalised polynomial behaviour in computational physics ☆,☆☆,☆☆☆

Guido Lombardi *, Davide Papapicco

Department of Electronics and Telecommunications, Politecnico di Torino, Italy

ARTICLE INFO

Keywords:
Numerical integration
Gauss-Legendre quadrature
Asymptotic estimation
Generalised polynomials
Müntz polynomials
Monomial transformation
Floating-point machine precision

ABSTRACT

Fast and accurate numerical integration always represented a bottleneck in high-performance computational physics, especially in large and multiscale industrial simulations involving Finite (FEM) and Boundary Element Methods (BEM). The computational demand escalates significantly in problems modelled by irregular or endpoint singular behaviours which can be approximated with generalised polynomials of real degree. This is due to both the practical limitations of finite-arithmetic computations and the inefficient samples distribution of traditional Gaussian quadrature rules. We developed a non-iterative mathematical software implementing an innovative numerical quadrature which largely enhances the precision of Gauss-Legendre formulae (G-L) for integrands modelled as generalised polynomial with the optimal amount of nodes and weights capable of guaranteeing the required numerical precision. This methodology avoids to resort to more computationally expensive techniques such as adaptive or composite quadrature rules. From a theoretical point of view, the numerical method underlying this work was preliminary presented in [1] by constructing the monomial transformation itself and providing all the necessary conditions to ensure the numerical stability and exactness of the quadrature up to machine precision. The novel contribution of this work concerns the optimal implementation of said method, the extension of its applicability at run-time with different type of inputs, the provision of additional insights on its functionalities and its straightforward implementation, in particular FEM applications or other mathematical software either as an external tool or embedded suite. The open-source, cross-platform C++ library Monomial Transformation Quadrature Rule (MTQR) has been designed to be highly portable, fast and easy to integrate in larger codebases. Numerical examples in multiple physical applications showcase the improved efficiency and accuracy when compared to traditional schemes.

Program summary
Program title: MTQR
CPC Library link to program files: https://doi.org/10.17632/276f78wzsw.1
Developer's repository link: https://github.com/MTQR/MTQR
Licensing provisions: GNU General Public License 3
Programming language: C++ (C++17 standard)
Supplementary material: *User manual* (for installation and execution)
Nature of problem: Accuracy and time of execution of the current implementations of high-precision numerical integration routines for singular and irregular integrands modelled by generalised polynomials are restricted by: limitations of the floating-point (f.p.) finite-arithmetic of the machine; inability of classical Gaussian quadrature rules to efficiently capture irregular behaviours or end-point singularities using an optimal number of samples; relying on significantly expensive techniques as adaptive or composite quadrature rules that severely increase the number of steps necessary to converge to the desired accuracy threshold. However by precisely manipulating the G-L samples using an ad-hoc monomial transformation we achieve a one-shot, non-iterative, machine-precise quadrature rule with straightforward scalability in higher dimensions. The advantages brought by a non-adaptive

☆ This is a collaborative effort: the authors are arranged in alphabetical order. Individual contributions are listed in the CRediT authorship contribution statement.
☆☆ The review of this paper was arranged by Prof. Andrew Hazel.
☆☆☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author.
E-mail address: guido.lombardi@polito.it (G. Lombardi).

https://doi.org/10.1016/j.cpc.2024.109124
Received 16 June 2023; Received in revised form 23 January 2024; Accepted 6 February 2024
Available online 9 February 2024

technique are greatly emphasised whenever the problem on hand is characterised by a numerous set of integrand functions that behave similarly to sets of generalised polynomials.

Solution method: The underlying algorithm is an application of a monomial transformation of (real) order to the $n_{\min} \in \mathbb{N}$ quadrature samples (nodes and weights) of the G-L rule. The order of the transformation depends on the infimum ($\lambda_{\min}$) and supremum ($\lambda_{\max}$) of the Müntz sequence of real degrees of the integrands modelled by generalised polynomials. With an a-priori full analysis of the set of integrands to be integrated, the design and the action of such transformation ensures that the bounding monomials are integrated with machine-epsilon precision in double floating-point (f.p.) format [2] without resorting to less efficient schemes as adaptive or composite quadrature rules, singularity subtraction and cancellation methods with limited and uncontrolled precision. The effects of the action of the monomial transformation onto the original G-L samples are clearly visible by the clustering of the nodes around the endpoint singularity of the integrand function (see Fig. 1.2 of the *user manual* shipped with the source code and located in the *MTQR/doc* sub-directory as indicated in the root tree in Fig. 2.1 of said manual.)

Additional comments including restrictions and unusual features: To strictly secure the integration with finite-arithmetic precision at the order of the machine epsilon in double f.p. representation, higher-than-double data types are necessary to build the quadrature rule; furthermore to compute the optimal number of quadrature samples, the sole real root of a 7th degree polynomial [1, Equation (62)] has to be computed. For these reasons the proposed software depends on both the Boost's [3] Multiprecision header-only library and on the GNU Scientific Library, GSL [4]. Any source code using MTQR must be linked to those libraries which can be easily installed and configured for the system specific compiler in both Windows and Linux. Detailed information and assistance for successfully compiling and building applications with this library can be found in the aforementioned *user manual*.

1. Introduction

Efficient and accurate computations of definite integrals are a critical step across the entire scientific spectrum, especially in numerical methods for partial differential equations (PDEs) using spectral, finite and boundary elements. A well-known pitfall in these methods concerns the integration of functions featuring either irregular behaviours or endpoint (integrable) singularities [5–13]. While ubiquitous in boundary discretisations of problems in applied physics (acoustics, electromagnetics, elastodynamics...) where the irregular behaviour and the singularities are embedded in the field's potential associated to the operator and/or the unknown fields, this issue also affects surface and volume discretisations. In fact, critical field behaviours arising in e.g. wave propagation in 2 and 3-dimensional domains tend to degenerate in singularities in the close proximity of sharp material and geometrical discontinuities [5,11,13] especially at high wave number [14,15]. One particular numerical method that has established itself as quintessential in various disciplines in both science and engineering is the FEM. To properly address the aforementioned issues [16,17] in volume discretisations, adaptive conforming finite and spectral elements cannot be deployed as their convergence would require intensive computational effort. Extended (XFEM), generalised (GFEM) and enriched (ESG) formulations of the finite and spectral element methods [18–20] provided a rigorous framework for mixed and non-conforming Galerkin approximations of those singular fields by enriching the sub-spaces with non-classical polynomial basis functions [5,21–24]. Aside from the needs of numerical methods for PDEs, there are several other areas concerning computational physics where the necessity for highly-precise computation of singular integrals [25–27] is of paramount importance. In this framework, special attention is devoted to the numerical integration of sets of integrand functions modelling irregular behaviours. Motivated by this, a considerable wealth of numerical algorithms has been proposed throughout the years to address the general problem of approximating singular integrals and overcome the limited performance offered by standard Gaussian quadrature rule and/or the computational cost of adaptive quadrature rules. Most notably [28–31] have spawn an active research on the development of generalised adaptive techniques [32,33] to be exploited in scientific computing. Those algorithms have the advantage of being robust and general when applied to a wide range of irregular and singular integrand functions, especially those featuring endpoint logarithmic singularities or functions whose irregular part can be factorised as a known weight function $w(x)$. On the other hand, they entail a further computational load in large-scale simulations involving complex, high-dimensional domains as they rely on iterative constructions of appropriate nodes and weights of specialised quadrature rules for any given integrand. Improvements have been made [34,35] to ease the effort in constructing such specialised quadrature rules however the underlying concept of the algorithm remains unchanged. This constitutes a bottleneck in performance especially in those Galerkin approximations of problems that model the singular behaviour (and/or irregular behaviour) of the solution with generalised polynomials of the form

$$f(x) = \sum_{k=1}^{m} c_k x^{\lambda_k}, \quad \lambda_k \in (-1, \infty]. \tag{1}$$

Moreover, the implementation of a high-precision quadrature rule for singular and/or non-classical polynomials in (1) is also hindered by the finite precision and memory of the machines running the simulations; machine-epsilon thresholds in finite arithmetic limit in fact the accuracy and performance of the numerical algorithm to the user's computing power and data types. We also note that in numerical applications such is the case for FEM and BEM, adaptive or composite quadrature algorithms must be run for every element that is either enriched by singular or irregular basis functions or it is defined over a higher-order (curved) geometry. In recent years a number of efforts in designing optimal quadrature rules for the isogeometric analysis (IGA) in FEM have resulted in efficient algorithms [36–38] based on the integration of spline functions. It has been long established in fact [39–41] that piecewise (low-degree) spline interpolation lead to more accurate and more efficient quadratures when compared to traditional Gaussian rules. These works have been able to produce powerful algorithms that generalise and extend the element-wise spline interpolation beyond classical Gaussian integration by producing optimal quadrature rules [38,42,43]. While these methods, i.e. Gaussian and semi-Gaussian spline rules, can be successfully deployed in the quadrature of both smooth and high oscillatory functions, their extension to endpoint singular integrands is non-trivial and requires further ad-hoc investigation. In [1] a one-shot, non-adaptive and non-iterative method was proposed to solve this shortcoming while dealing with endpoint singularities (or irregular behaviour); in it a simple yet powerful quadrature rule was proposed which is capable to achieve arbitrary machine-precision with the original field of applications being to ease the computational task required by

numerical integration in generalised Galerkin methods for BVPs, with singular and/or non-classical polynomial modelling. This novel algorithm is based on the optimal design of an ad-hoc monomial transformation applied to selected G-L nodes and weights to capture the endpoint singularity (or irregular behaviour) of the generalised polynomial integrand in (1) according to the bounds of the (ordered) set $[\lambda_{\min}, \lambda_{\max}]$ of real-valued exponents. In this work we developed a fast, light and portable cross-platform C++ library implementing such Monomial Transformation Quadrature Rule (MTQR) which assures arbitrary machine-precision integration for these special cases while retaining the minimum (optimal) amount of samples for the G-L formula. MTQR has been programmed with an emphasis on flexibility (runtime polymorphism and templetisation) and simplicity (minimal user input) in order to provide a quick integration in larger and more complex open-source codebases s.a. well established finite element solvers [44–51] and other libraries in numerical mathematics [52–55]. Moreover, the mathematical software can be used as external tool that provides quadratures for sets of generalised polynomials. The algorithm's architecture and its library's implementation allow to straightforwardly adapt the code to more general, user-defined machine-precision [2], i.e. more or less restrictive than the double f.p. format specified above and throughout the rest of this work. The integration of MTQR with state-of-the-art numerical integration techniques based on Gaussian and semi-Gaussian spline rules should allow for the exact analysis of complex structures containing both singularities and smooth/high oscillatory behaviour however this task falls outside the scope of this work and shall be investigated in future endeavours.

In Section 1, we introduce the motivation and the scope of the present work and report on the state of the art related to the proposed algorithm. Section 2 presents the derivation of the algorithm itself with the necessary mathematical background, while Section 3 is devoted to its implementation as a mathematical software. Validation and efficacy of the proposed algorithm is demonstrated in Section 4 presenting two sets of multiple examples related to different sub-disciplines in computational physics, namely the integration of arbitrary products of Bessel's functions of the first kind of fractional order and the assembly of the mass matrix in 2.5-dimensional, additive, vector finite elements arising in electromagnetic scattering and propagation problems with diffraction phenomena. In this context, we compared the performances of our library against other, well-established adaptive algorithms implemented in widely-known open-source scientific software. Conclusions are addressed in Section 5 and the Appendix reports the primary module's source code of the proposed library. This work is supplemented by a detailed *user manual* which describes the implemented code, structure, installation and execution of the library in both Linux and Windows.

2. Derivation

In the following we will derive a consistent formulation of the numerical problem addressed by MTQR and provide the computational steps required to overcome the aforementioned limitations and contextualise its applications.

2.1. Motivation

The main purpose of MTQR is to provide users and developers of open-source scientific codes a straightforward and light numerical tool to perform arbitrarily accurate integration of functions that exhibit irregular or singular behaviour at either of the endpoints of the integration interval. In particular, if the integrand function $f : (a, b) \to \mathbb{R}$ has an integrable singularity in $x^* \in \{a, b\}$ that can be locally modelled by expanding $f(x^*)$ as a generalised polynomial (see Section 3.3 in [16]) of the form (1), then we are concerned with the computation of

$$I(f) = \int_a^b f(x)dx\,, \tag{2}$$

with an arbitrary (user-defined) machine precision in finite arithmetic (i.e. using f.p. approximations of the real numbers). Of course the analytic integration of polynomial functions is trivial, however there are several cases in numerical and computational mathematics in which those integrals are required to be calculated numerically. One straightforward case is when these functions are not known a-priori and thus need to be integrated during the execution of a simulation. Such instances are commonly encountered in several implementations of Galerkin methods for the approximation of PDEs where the local weak formulation of the model requires the numerical evaluation of (2) either when enriching the approximation sub-space with generalised polynomials basis functions or when the forcing term is itself modelled by a generalised or singular polynomial.

2.1.1. Generalised finite methods

In finite, spectral and boundary elements methods we seek for a numerical approximation $u_h(\mathbf{x})$ of a BVP defined over a d-dimensional domain $\Omega \subset \mathbb{R}^d$. The approximation sub-space is usually spanned by piecewise classical polynomials that locally interpolate the solution in every discretised node of the domain and the solution is then expressed as a linear combination of the basis functions of such sub-space. In the generalised finite methods introduced above the set of classical polynomial basis functions are enriched by generalised polynomials of non-integer degree defined in the proximity of the singular point $\mathbf{x}^*$. Therefore the sub-space is no longer spanned by classical polynomials (of integer degree) but instead we search for a solution that is (locally) well represented by a Müntz polynomial [56], i.e. a generalised polynomial function of type (1) for which the (increasing) sequence $\Lambda = \{-1 < \lambda_k < \lambda_{k+1}\}_{k=1,\dots,m-1}$ of real-valued exponents satisfies

$$\sum_{k=1}^{\infty} \frac{1}{\lambda_k} = \infty\,. \tag{3}$$

Equality (3) ensures that the Müntz space $M(\Lambda) = \text{span}\{x^{\lambda_k},\ \lambda_k \in \Lambda\}$ is dense in $C[(0, 1)]$ [57]. Practical examples in which singular solutions are well approximated by Müntz polynomials are propagation and scattering problems in the presence of corners, wedges, abrupt changes in the boundaries and any sharp discontinuity within the domain Ω. In the weak formulation of variational BVPs the linear forms often stem from inner products that involve the computation of the definite integral of some products of the solution $u_h(\mathbf{x})$ with some test functions $v_h(\mathbf{x})$. Proper employment of the Galerkin method then projects the weak form onto a finite dimensional sub-space which, in our case, has Müntz polynomials enriching the classical shape functions of integer degree. Elemental matrices for those elements $\mathcal{K} \in \mathbb{R}^d$ that contain the singular point $\mathbf{x}^*$ will now require an appropriate (accurate and efficient) computation of such scalar quantity

$$\int_{\mathcal{K}} p_\alpha(\mathbf{x})\, p_m(\mathbf{x})d\mathbf{x}\,, \quad p_m(\mathbf{x}) \in \mathbb{P}_m[x_1, \dots, x_d]\,,\ p_\alpha(\mathbf{x}) \in (M(\Lambda))^d\,,\ \alpha = \max(\Lambda)\,, \tag{4}$$

i.e. where at least one of the two basis functions is modelled by a non-classical polynomial.

We remark that a detailed and thorough digression on the weak form and operator discretisation is well outside the scope of the present work, which instead focuses on handling (4) specifically in code implementation of variational numerical methods for PDEs and we refer the discussion on Müntz-enriched finite methods to the aforecited [18,19,22].

2.2. Background on interpolatory Gaussian quadrature formulae

The development of MTQR starts with the precise approximation of $I(f) = \int_a^b f(x)dx$. Specifically it addresses the numerical computation of the definite integral using a quadrature rule that assures an arbitrary (user-defined) accuracy while being as efficient as possible by retaining the optimal number of nodes provided by a Gaussian formula (avoiding to resort to an adaptive or composite technique). In the following we will refer to the endpoint singular/irregular integrand in (1) where $\lambda_{\min} := \lambda_1 = \min(\Lambda) > -1$ and $\lambda_{\max} := \lambda_m = \max(\Lambda) < \infty$. In numerical analysis, the approximation $I_n(f)$ of a definite integral $I(f)$ can be achieved through several distinct approaches. One method, widely used in scientific computing, is the usage of interpolatory quadrature formulae. They are based on the substitution of the integrand $f : \mathcal{I} = (a,b) \to \mathbb{R}$ with a Lagrange polynomial $\mathcal{L}_n(x)$ interpolating its values along an ordered set of distinct points ($\mathcal{I} \ni x_j \neq x_k \iff j \neq k$) called nodes

$$\mathbb{P}_{n-1} \ni \mathcal{L}_n(x) := \sum_{j=1}^{n} f(x_j)\ell_j(x) \Rightarrow \mathcal{L}_n(x_j) \equiv f(x_j), \tag{5}$$

where $\ell_j(x) := \prod_{k=1,k\neq j}^{n} \frac{x-x_k}{x_j-x_k}$, $\forall j = 1,\dots,n$, is a set of n orthogonal polynomials of degree $n-1$ which forms the *Lagrangian basis* of $\mathbb{P}_{n-1}[x]$. An interpolatory quadrature formula is built from (5) by substituting the integrand with its Lagrangian polynomial. The resulting quadrature rule has the form

$$I_n(f) = \sum_{j=1}^{n} f(x_j)w_j, \quad I(f) = \int_a^b f(x)dx = I_n(f) + E_n(f), \tag{6}$$

where $w_j := \int_a^b \ell_j(x)dx$ are the so-called weights of the quadrature rule. We remark that to each choice of the partitioning nodes there is a unique set of quadrature weights; this property derives from the uniqueness of $\ell_j(x)$ given an ordered set of distinct nodes in $\mathcal{I}$. The term $E_n(f) = \int_a^b E_n'(f)dx$ quantifies the absolute error, often referred to as the remainder of the quadrature rule, through the interpolation error of $\mathcal{L}_n(x)$ of $f(x)$. From a computational point of view, it is ideal however to work with a relative error instead and so we identify, and henceforth use, the following normalised quantity

$$R_n := \frac{|E_n(f)|}{|I(f)|}, \tag{7}$$

associated to the quadrature formula. Of the many numerical quadrature technique, Gaussian rules assure the maximum degree of precision integrating exactly (i.e. $E_n(f) = 0$) any classical polynomial up to degree $d = 2n-1$, where n in the number of samples of the rule itself [58]. Those formulae are built from nodes that stem out as the roots of the associated orthogonal polynomials [59] from which they take the name. Due to their simplicity and the wide range of applicability, provided by the weight function $w(x) \equiv 1$, G-L quadrature rules are used to integrate sufficiently well-behaved functions in closed intervals with relatively small number of samples. Functions $f(x) = w(x)g(x)$ with endpoint singularities often are integrated using Gauss-Jacobi (G-J) quadrature rules by rewriting the non-regular weight function as $w(x) = (1-x)^\alpha(1+x)^\beta$ for specific $\alpha,\beta > -1$. The monomial transformation quadrature rule proposed in [1] on the other hand allows to retain the simplicity of the G-L rule and extend the range of applicability integrating sets of functions modelled by generalised singular/irregular polynomials with the improved precision at cost comparable to G-L quadrature and with multiple values of singular/irregular exponents of monomial terms.

2.3. Asymptotic error estimate

When implementing a Gaussian quadrature rule in a software, the accuracy of its result suffers from rounding errors that are embedded in the f.p. representation of real numbers. It is easy to verify such limitation by checking the exactness of a G-L formula, given by its degree of precision $d = 2n-1$, against its finite-arithmetic counterpart for a given number of quadrature nodes n. The bottleneck for the fidelity of a code implementing such rule would be the decimal precision with which nodes and weights of the formula are stored. Tabulated values of those parameters exist for various Gaussian quadrature rules with up to 32 decimal digits of precision [58]. The constraint on the user side is that traditional f.p. formats available for most of today's general-purpose processors and operating systems, s.a. double precision, operate with a much lower precision (approximately 16 decimal digits respectively). In practice, achieving acceptable results using the classical G-L formula is usually a trade-off between accuracy and computational cost based on the f.p. format with which the quadrature samples are stored.

This aspect exacerbates whenever the integrand is a generalised polynomial with endpoint singularities; with reference to Fig. 1 we can assess how non-integer degree monomials are integrated with less finite precision (concave oriented humps) than those of integer-valued degrees (downward pointing spikes). When integrating polynomials with a G-L quadrature rule it is ideal to achieve close to analytic exactness with the minimum possible number of nodes. To measure the accuracy of a given formula we can use an a-posteriori relative error for which the numerator of (7) is computed as the strictly positive difference between the analytic primitive of the integrand $I(f)$ and its numerical counterpart $I_n(f)$ computed using the selected quadrature rule with n nodes, i.e. $|E_n(f)| \equiv |I(f) - I_n(f)|$. This procedure is computationally inefficient as it involves the numerical evaluation of the quadrature rule; on the other hand with an a-priori relative error the only knowledge required is that of the integrand function $f(x)$. Accurate estimations of the a-priori form for the remainder $E_n(f)$ have been the subject of numerous efforts and constitute the cornerstone upon which MTQR is built.

Theorem 1. *Let $I_n(f)$ be a Gaussian quadrature rule with $P_n \in \mathbb{P}_n$ being a polynomial of degree n, orthogonal to a weight function $w(x)$ in the closed interval $\mathcal{I} = (a,b)$. If we consider the analytic continuation of f to the complex plane and we specify a contour C that encloses $\mathcal{I}$, then the remainder of $I_n(f)$ has the exact form of the following contour integral along C*

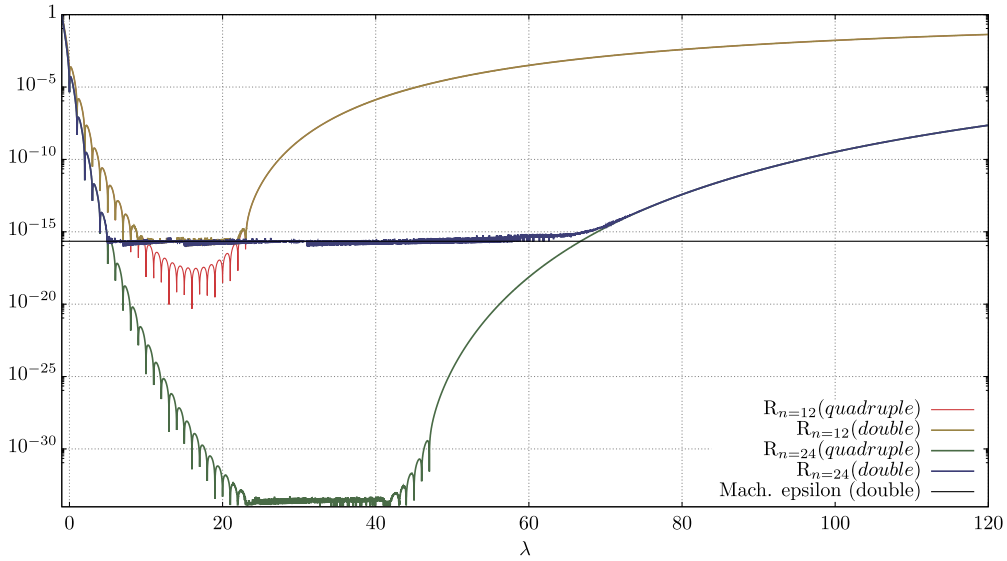


Fig. 1. Comparison of how truncation of decimal digits in finite-arithmetic representation of the samples affects the a-posteriori relative error R_n associated to a G-L quadrature rule integrating 10000 monomials of the form $x^\lambda \in \mathcal{I} = (0,1)$ with λ uniformly distributed in $(-1,120]$. The quadrature formula uses $n \in \{12,24\}$ nodes and weights stored in double and quadruple (float128) f.p. precision. We report the machine-epsilon $\epsilon = 2^{-52}$ in double f.p. format as a reference threshold. We observe how, after the specified degree of precision $d = 2n-1$ of each G-L quadrature rule, the error features a slow monotonic increase. Furthermore we notice that as the number of samples n for the G-L formula increases, the portion of the asymptotic a-posteriori relative error that falls below the numerical exactness threshold gets progressively larger. (For interpretation of the colours in the figure, the reader is referred to the web version of this article.)

$$E_n(f) = \frac{1}{2\pi i} \oint_C \frac{\Pi_n(z)}{P_n(z)} f(z) dz, \quad \Pi_n(z) := \int_a^b \frac{P_n(x)}{x - z} dx, \; z \in \mathbb{C}/\mathcal{I}. \tag{8}$$

Proof. See Barret [60]. □

The integrand function in the remainder estimate (8) of Theorem 1 is a meromorphic function (recall that P_n is a polynomial of degree n meaning that the reciprocal $\frac{1}{P_n}$ has n poles distributed along $\mathcal{I}$). Although elegant in its formulation, estimate (8) does not satisfy the rapid evaluation requirement that we specified in the numerical applications as it requires the explicit computation of a complex contour integral. Furthermore the explicit form of $\Pi_n(z)$ is complicated to obtain as it requires the evaluation of hyper-geometric functions [61]. It is obvious to see how, even for simplistic applications of the G-L quadrature rule to monomial integrand functions, the analytic evaluation of the a-priori estimate using (8) is too expensive, effectively nullifying the efforts of avoiding to compute $E_n(f)$ with the a-posteriori formula. For this reason, asymptotic analysis was introduced in [62] by expanding both $\Pi_n(z)$ and $P_n(z)$ in a normalised interval $\mathcal{I} = (0,1)$.

Proposition 1. *Let $f(x) = x^\lambda$ and $I(f) = \int_0^1 x^\lambda dx$. If the number of nodes n (and degree of Legendre polynomials $P_n(x)$) of the G-L formula $I_n(f)$ is large, then the following asymptotic equivalence holds*

$$\frac{\Pi_n(x)}{P_n(x)} \approx \frac{2\pi}{(x + (x^2-1)^{\frac{1}{2}})^{2n+1}}, \tag{9}$$

and thus the remainder in (8) reduces to

$$E_n(f) = -2^{-2\lambda}\, \lambda\, \sin(\pi\lambda) \left(\frac{B(2\lambda, 2n-\lambda)}{2n+\lambda} - \frac{B(2\lambda, 2+2n-\lambda)}{2+2n+\lambda} \right), \tag{10}$$

where $B(z,w) := \frac{\Gamma(z)\Gamma(w)}{\Gamma(z+w)}$.

Proof. See Lombardi [1]. □

The result in (10) is meaningful as it enables a fast and concise estimate of the remainder for integrating generalised monomials with real degree $\lambda \in (-1, +\infty)$ in $(0,1)$. In fact we observe how the new error estimate depends solely on the number of nodes n and the real-valued monomial degree λ. Nevertheless, as evinced in Fig. 1, analytic exactness for any finite-arithmetic representation of a quadrature rule implemented in code can never be achieved.

Therefore, to quantify the performance and accuracy of a numerical integration routine we must specify a threshold of accuracy as an alternative finite-precision equivalent exactness. In the following we assume such threshold to be, without loss of generality, the machine epsilon in double precision [2]; this assumption is, as a matter of fact, arbitrary in nature and can be easily changed to be more or less restrictive without impacting the workflow of the monomial transformation quadrature rule.

The desirable characteristics of a useful a-priori error estimation of a quadrature rule stem from the very task for which it would be used in case of polynomial integration, that is minimising the extensive computation time required for evaluating the λ-asymptotic behaviour of multiple G-L

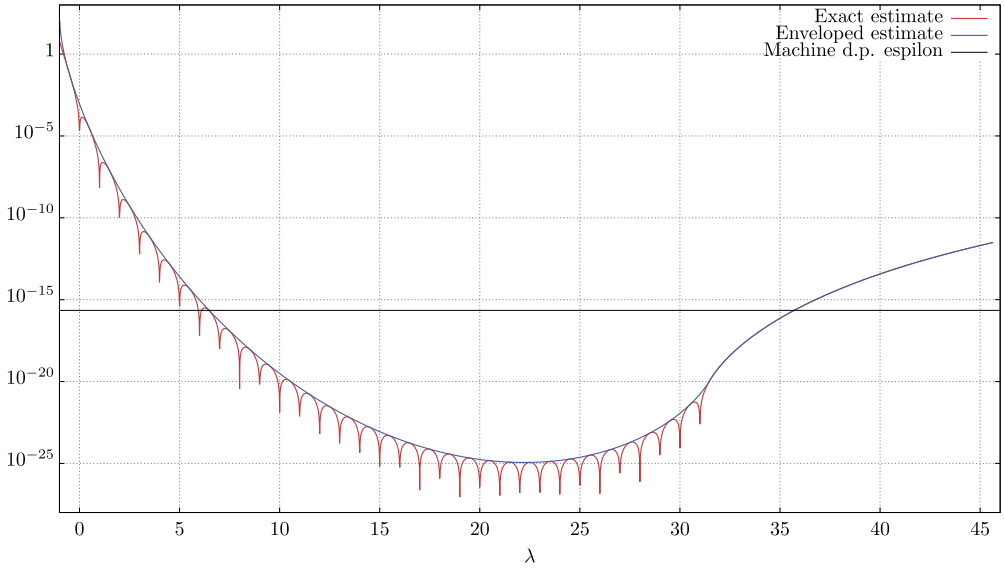


Fig. 2. The new "exact" a-priori estimate in (10) is used here to quickly evaluate the λ-asymptotic behaviour of the relative error of a G-L quadrature formula with $n = 16$ nodes integrating monomials x^λ, $\lambda \in (-1, 46]$ of non-integer degree in $\mathcal{I} = (0, 1)$. The humps-spike alternation matches the one partially featured by the exact a-posteriori relative error reported in Fig. 1. The enveloped estimate is obtained by omitting the term $\sin(\pi\lambda)$ from (10) when $\lambda < 2n - 1/2$. From the plot of the enveloped estimate we can derive an approximation of the minimum $\lambda = \beta_{\min} \approx 6.5463$ and maximum degree $\lambda = \beta_{\max} \approx 35.7774$ of a generalised polynomial to be integrated with finite-arithmetic (double precision) equivalent exact precision using such formula. (For interpretation of the colours in the figure, the reader is referred to the web version of this article.)

quadrature rules (i.e. for several values of n) applied to x^λ integrand by avoiding to compute I and I_n explicitly. In Fig. 2 we depict the asymptotic analysis performed using the new estimate in (10) for a $n = 16$ nodes G-L formula; a similar pattern is encountered, previously deduced from the a-posteriori plot, that a steady but somewhat slow degradation of the G-L accuracy is detected once the degree of precision of the formula is reached i.e. $\forall \lambda > d = 2n - 1$. In Fig. 2, the enveloped estimate is obtained by omitting the term $\sin(\pi\lambda)$ from (10) when $\lambda < 2n - 1/2$ and it will be used as the optimal upper error estimate for the design of each monomial transformation quadrature rule.

3. Implementation

Having briefly outlined the fundamental mathematical problem of numerical integration and error estimation we will now discuss how the implementation of MTQR addresses and ultimately overcomes the shortcomings of the currently available algorithms dealing with integrands modelled with generalise polynomials. From here onward we will consider $\mathcal{I} = (0, 1)$ as the reference interval of integration with $f(x)$ having a potential end-point singularity or irregular behaviour located at $x = 0$. This choice does not limit the applicability of the algorithm as any integrand defined in (a, b) with one end-point singularity can be easily mapped in $(0, 1)$ via an affine transformation. In case of multiple singular points, we can in fact simply partition (a, b) into as many sub-intervals, each having a singularity at one endpoint. This procedure is of fundamental importance to avoid numerical cancellation since the action of MTQR on classical G-L nodes causes their clustering near $x = 0$ as previously discussed.

3.1. Monomial transformation

The new a-priori estimate (10), alongside its enveloped version depicted in Fig. 2, is a tool that enables a much swifter analysis of the asymptotic behaviour of the G-L relative error that we previously computed while producing Fig. 1 using an exact a-posteriori estimate. For any given G-L formula with n nodes, we can rapidly evaluate what the minimum $\lambda = \beta_{\min}$ and maximum value $\lambda = \beta_{\max}$ are for the degree of a generalised polynomial to be integrated with finite-arithmetic equivalent exactness using a predefined accuracy threshold, for instance the double precision machine epsilon. We therefore learned that (10) can be effectively used to determine those critical constraints for any G-L formula to achieve (finite arithmetic equivalent) precise quadratures immediately realising that tabulated values of $\beta_{\min}(n)$, $\beta_{\max}(n)$ can be used to design appropriate transformations of the G-L samples. For instance, by performing the same λ-asymptotic analysis with increasingly greater values of n we immediately notice how the *bathtub* shaped region of Fig. 2 both deepens (the relative error gets smaller as higher and higher accuracy is achieved with an increasing number of samples) and widens (the sub-interval $\lambda \in [\beta_{\min}, \beta_{\max}]$ gets larger as the degree of precision $d = 2n - 1$ grows linearly with n). Furthermore, by recalling that the exact a-priori estimate (10) used in plotting Fig. 2 was derived for the specific λ-asymptotic analysis of generalised monomial functions of non-integer degrees in $(0, 1)$, we can map any generalised polynomial, including singular ones of real degree $\lambda > -1$, in such interval where the estimate holds for the G-L quadrature rule. For a given choice of n nodes G-L quadrature rule, if a certain monomial x^λ does not fall within the bathtub, i.e. the "exactness" region provided by the estimate (10), i.e. $\lambda \notin [\beta_{\min}, \beta_{\max}]$, then an ad-hoc designed monomial map of $(0, 1)$ onto itself can shift its exponent to fall within the desired interval. If all the $\beta_{\min}(n), \beta_{\max}(n)$ values are known or have been tabulated previously for a given range of $n \in \mathbb{N}$ samples, then the remaining pieces of information needed for constructing the map are the minimum and maximum non-integer exponents of the integrand function. The fundamental algorithm in MTQR is composed by efficiently coupling an ad-hoc designed monomial transformation with a G-L quadrature rule by selecting the optimal number of samples n. Let us start with the latter; given an input Müntz sequence of real exponents we are interested in fitting its minimum values $\lambda_{\min}$ and its maximum $\lambda_{\max}$ within the exactness region of a G-L quadrature rule associated to the (quasi) minimum number of nodes n possible.

Proposition 2. *Let (1) be a Müntz polynomial of degree λ_{max} and lowest degree $\lambda_{min} > -1$, then the minimum number n_{min} of G-L quadrature nodes to achieve finite-arithmetic equivalent exactness in terms of target double precision using the monomial map is the round toward ceiling of the unique real root n of*

$$(-4.0693 \cdot 10^{-3} + 4.1296 \cdot 10^{-4} n)[(8.8147 + 1.0123 \cdot 10^{-1} n^2) \cdot (1 + \lambda_{min}) - (1 + \lambda_{max})]^3 - (1 + \lambda_{max})^3 = 0, \tag{11}$$

obtained by a full study of the performance of G-L via linear regressions of the tabulated values of $\beta_{min}(n), \beta_{max}(n)$.

Proof. See Lombardi [1]. □

Once $n = n_{\min}$ is computed from (11), we can now turn our attention to the construction of the monomial transformation of order $r \in \mathbb{R}$

$$\gamma_r(x) = x^r, \tag{12}$$

acting on the $n_{\min}$ samples of the selected quadrature rule. MTQR refers to the tabulated values of $\beta_{\min}(n), \beta_{\max}(n)$ (or directly estimate them from the enveloped version of the estimate (10)) and supplies all the necessary data to compute the order r of the monomial transformation uniquely associated to the set of input polynomials defined by $\lambda_{\min}$ and $\lambda_{\max}$. From [1] we have

$$r_{\min} := \frac{1 + \beta_{min}(n)}{1 + \lambda_{min}} < r < \frac{1 + \beta_{max}(n)}{1 + \lambda_{max}} =: r_{\max}, \tag{13}$$

and in MTQR we compute the order r as the mean value between $r_{\min}$ and $r_{\max}$. By letting map (12) act on the definite integral of a generalised polynomial of the form (1) we obtain

$$I(f(x)) = \int_0^1 f(x)dx = \sum_{k=0}^{m} c_k \int_0^1 x^{\lambda_k} dx = \sum_{k=0}^{m} c_k \int_0^1 r x^{r(\lambda_k+1)-1} dx. \tag{14}$$

The numerical approximation for the prescribed machine-precision of (14) is then provided by the monomial transformation quadrature rule with n samples and weights x_j, w_j defined over the integration interval $(0, 1)$ as a result of the (quasi) optimal selection of $n = n_{min}$ and r

$$I_n = \sum_{k=0}^{m} c_k \sum_{j=1}^{n} (\tilde{w}_j r \tilde{x}_j^{r-1}) \tilde{x}_j^{r\lambda_k} = \sum_{k=0}^{m} c_k \sum_{j=1}^{n} w_j x_j^{\lambda_k}. \tag{15}$$

From (15) we deduce that the new, proposed numerical integration specified for the integrand $f(x)$ is uniquely determined by the ad-hoc manipulation of classical G-L quadrature rule's nodes and weights $\tilde{x}_j, \tilde{w}_j$ also defined in $(0, 1)$

$$x_j = \tilde{x}_j^r, \; w_j = r \tilde{x}_j^{r-1} \tilde{w}_j \quad \forall j = 1, ..., n. \tag{16}$$

From (14)-(16) we observe that the monomial transformation quadrature rule only requires the information regarding the lower (λ_{min}) and upper (λ_{max}) bounds of the Müntz sequence λ of the input polynomial to retrieve an ad-hoc transformation of the G-L quadrature's samples to achieve a finite-arithmetic equivalent exact approximation of the definite integral in $(0, 1)$. We hereby clarify that the evaluation of n_{min} via (11) is obtained by imposing that $r_{min} = r_{max}$ in (13) and considering regression formulae for $\beta_{min}(n)$, $\beta_{max}(n)$ (see (60) in [1]) with target double precision. Moreover, the Müntz space related to the (quasi) optimal quadrature of generalised polynomials in $M(\Lambda)$ with $\lambda \in (\lambda_{min}, \lambda_{max})$ is $M(B)$ with $\beta \in (\beta_{min}(n_{min}), \beta_{max}(n_{min}))$ via the monomial transformation (13) with $r \in (r_{min}, r_{max})$. Generalisation to arbitrary precision is straightforward and achieved by selecting appropriate regression formulae for $\beta_{min}(n)$, $\beta_{max}(n)$ which can be obtained from the asymptotic error estimation discussed in Section 2.3.

3.2. Software integration

MTQR is a C++ implementation of the monomial transformation quadrature rule derived thus far. It consists of a collection of highly-templatised methods and it is written in a procedural programming paradigm. Interaction with the library happens through the so-called primary module (see Appendix A), an overloaded method that is instantiated by the users by passing the parameters of the input polynomial and it returns optimised nodes and weights for the machine-precise numerical integration[1]. A numerical recipe detailing the main steps performed by the library when its primary module is instantiated is reported below. We prepared a comprehensive *user manual* for the correct installation of MTQR alongside its

[1] To allow MTQR to deal with extreme cases of high monomial transformation order our default datatype is the quadruple precision (float128) to export the new nodes and weights. If however double precision for those samples is sufficient for them reach the machine epsilon in the computation of the definite integral than the data type precision is purposely regressed; this allows the user of MTQR to optimise its data and memory management while retaining as much precision as possible. For more detailed information regarding this aspect we refer to the library's *user manual*.

dependencies [3,4] and running custom applications with it; we refer to such manual (located in the *MTQR/doc* sub-directory of the library) for more detailed information.

Data: Sequences $(c_1, \dots, c_m)$ and $(\lambda_1, \dots, \lambda_m)$ of polynomial integrand function of the form (1).
Result: Optimised nodes $\mathbf{x}_j$ and weights $\mathbf{w}_j$ of the monomial transformation quadrature formula.
Set accuracy threshold (e.g. double machine$-\epsilon = 2^{-52}$);
Extract $(\lambda_{\min}, \lambda_{\max})$ through a sorting algorithm;
Compute n_{min} from (11);
Derive $\beta_{\min}(n_{min})$ and $\beta_{\max}(n_{min})$ from tabulated pre-computed values;
Obtain the monomial transformation's order r ;
Pre-load affinely mapped samples $\tilde{\mathbf{x}}_j$ and $\tilde{\mathbf{w}}_j$ associated to the $n_{\min}$ G-L rule;
Monomial transformation: Map $\tilde{\mathbf{x}}_j$ and $\tilde{\mathbf{w}}_j$ in $(0,1)$ through γ_r (12) obtaining the new nodes $\mathbf{x}_j$ and weights $\mathbf{w}_j$;
Assemble and compute the monomial transformation quadrature formula with n_{min} nodes;
Compute the a-posteriori relative error R_n using the new nodes and weights;
while $R_n < \epsilon$ **do**
    Optimise the f.p. format of the new nodes and weights;
    Compute the a-posteriori relative error R_n using the degraded nodes and weights;
end
Export $\mathbf{x}_j$, $\mathbf{w}_j$ and the estimated values of the integral of the selected generalised polynomial for testing.

3.3. Computational costs in brief

For what concerns the computational cost of the algorithm, useful run time estimations are provided further in Section 4.1 where comparison with alternative adaptive quadrature algorithms is performed in terms of number of function evaluations. In brief, the computational cost of MTQR can be analysed from different points of view: 1) application of the designed MTQR rule to a series of integrals of generalised polynomials, 2) generation of the MTQR quadrature rule for a selected Müntz space and target precision, 3) background computations to develop 2). Once MTQR has generated the samples and weights for a selected Müntz space and target precision, the cost of computation of each integral is limited to the one of ordinary interpolatory quadrature rules in terms of the number of samples, i.e. simply made of integrand function evaluations, multiplications by weights and summation. In order to generate the MTQR quadrature rule for a selected Müntz space and target precision we apply the algorithm as described in Section 3.2. In this case, the cost of computation is related to identifying λ_{max} and λ_{min} of the set of integrands (i.e. the selected Müntz space), solve (11) or a similar equation for a target precision different from d.p. with an external tool, compute the monomial transformation of order r according to (13) and finally apply (15) to get the transformed samples and weights. The background computations are primarily referred to the development of regression formulae for $\beta_{max}(n)$ and $\beta_{min}(n)$ which are necessary to get (11) for a selected target precision, e.g. single, double or quadruple precision for the current state-of-the-art. The computational cost is related to the analysis reported in [1] and they are needed once a for all. In brief this analysis requires non-linear solvers for a given asymptotic error estimates of the G-L rule and a selected target precision machine epsilon as a function of the number of samples, see Fig. 2. Finally a rich database with G-L rules as a function of the number of samples needs to be available at the selected target precision.

4. Applications

Following the discussion of the previous sections we now propose a series of numerical tests with the twofold aim of validating the performance of MTQR, in terms of both accuracy and computational efficiency, and to provide substantial evidence in support of the previous claim with regards to its wide scope of applicability and integration in larger, multipurpose codebases. We highlight that the library itself is shipped with two executable scripts (located in the sub-directory *MTQR/tests*) that can be used to validate the successful installation of the software (for more information we refer to the *user manual*). The numerical experiments that follow are additional tests we implemented with the objective of achieving the aforementioned goals of efficient accuracy and usefulness in other numerical and computational physics libraries. Relating to the latter goal we propose two series of tests, each one related to a different discipline and thus targeting different applications in multiphysics. Specifically in Subsection 4.1 we evaluate MTQR's performance when dealing with products of Bessel's functions of the first kind of fractional order (featuring an endpoint polynomial singularity or irregular behaviour in $x = 0$); in Subsection 4.2 the library is used in the context of Galerkin methods integrating the elemental contributions to the mass matrix assembled in 2.5-dimensional additive vector finite elements arising in electromagnetic scattering and propagation problems.

4.1. Integrating arbitrary products of Bessel functions of real order

Indefinite integrals involving arbitrary products of Bessel functions of the first kind of the form

$$\int_0^\infty f(x) \prod_{j=1}^{n} J_{\nu_j}(a_j x)\, dx\,, \quad \nu_j > -1,\ a_j > 0\,, \tag{17}$$

are of great interest in a wide variety of applications ranging from theoretical and applied electromagnetics [63–67] to acoustics and elastodynamics [68–72], fluid dynamics [73–77], theoretical physics [78–82] and several more [83–87] to name but a few. In 1995 an algorithm [88] was proposed to compute (17) with one product of two Bessel functions of positive integer order (i.e. $n = 1$, $\nu_j \in \mathbb{N}$) and with $f(x)$ being any sufficiently smooth function. In 2014 an implementation of said algorithm [89] resulted in a MATLAB toolbox named *IIPBF* which generalised the previous algorithm to deal with Bessel functions of fractional order. In 2006 a different work [90] provided another MATLAB package, named *BESSELINT*, which is able to integrate products of an arbitrary number of Bessel functions (i.e. any $n \in \mathbb{N}$) albeit with the caveat of having $f(x) = x^m$ where $m \in \mathbb{R}$ s.t. $\sum_{j=1}^{n} \nu_j + m > -1$; two years later an extension of such package [91] allowed to deal with exponential $g(x) = e^{-cx} x^m$ and rational $f(x) = \frac{x^m}{x^2+u^2}$

kernels (with $u, c \in \mathbb{R}$, $c > 0$). Both programs provide accurate approximations of (17) by splitting the indefinite integral in a *finite* and *infinite* range integrals in the following matter

$$\int_0^\infty f(x)J_\nu(ax)J_\mu(bx)\,dx = \int_0^{x_0} f(x)J_\nu(ax)J_\mu(bx)\,dx + \int_{x_0}^\infty f(x)h(x)dx\,, \tag{18}$$

where we considered the case of a single product of two Bessel functions to comply with both methods. The *infinite* range integral, which is the main focus of the two algorithms, is effectively computed using an asymptotic expansion $h(x)$ of the product $J_\nu(ax)J_\mu(bx)$ for large x thereby obtaining a more well-behaved integrand. For the *finite* range integral, both algorithms resort to adaptive quadrature rules and in particular:

- *IIPBF* uses a Gauss-Kronrod (G-K) technique which is implemented in the toolbox's routine **dqagea** via MATLAB's built-in function **quadgk**;
- *BESSELINT* uses an adaptive hard-coded G-L quadrature rule with $\{5, 7, 11, 15, 19, 23, 27, 31\}$ samples implemented in the routine **fri**.

Alongside the numerical technique used to integrate the finite range integral, the two methods differ both in the asymptotic approximation for the infinite range integration and also in the computation of the breakpoint $x_0 \in (0, \infty)$ at which the interval decomposition occurs. We will not focus on these aspects as they are not entirely concerning the potential presence of an endpoint singularity in $x = 0$ which is instead what MTQR is purposely designed to deal with. As a matter of fact both [88,89] (concerning *IIPBF*) and [90,91] (concerning *BESSELINT*) lament a decrease in either accuracy (quantified in a loss of decimal digits of precision) or efficiency (quantified in an increased number of function evaluations to reach the specified machine precision) when the integrand $f(x)J_\nu(ax)J_\mu(bx)$ presents an irregular or singular behaviour in the left endpoint of the integration interval. Not being an adaptive technique, MTQR can easily overcome the cited limitations when dealing with the aforementioned cases providing a simple, fast and extremely robust framework that can be easily integrated in either the two toolboxes in lieu of the third-party computationally expensive adaptive methods that have been deployed by *IIPBF* and *BESSELINT*. To prove this claim numerically we propose four test cases with integrand $f(x)J_\nu(ax)J_\mu(bx)$, and fixed orders

$$f(x) = x^{-\frac{1}{2}}\,, \quad \nu = 0, \mu = 1,\ a = 1, b = \frac{3}{2}\,, \tag{19}$$

$$f(x) = x^{\frac{1}{6}}\,, \quad \nu = -\frac{1}{3}, \mu = 0,\ a = 1, b = 3\,, \tag{20}$$

$$f(x) = 1\,, \quad \nu = -\frac{1}{2}, \mu = -\frac{1}{3},\ a = 1, b = 1\,, \tag{21}$$

$$f(x) = 1\,, \quad \nu = 0, \mu = -\frac{\pi}{4},\ a = 1, b = 1\,, \tag{22}$$

followed by two further cases with varying parameters

$$f(x) = J_\eta(\sqrt{2}x)\,, \quad \nu = 0, \mu = 0,\ a = \sqrt{3}, b = \sqrt{4}\,, \tag{23}$$

$$f(x) = \frac{x^{\mu-\nu+1}}{x^2+u^2}\,, \quad \nu = \frac{1}{2}, \mu = -\frac{1}{2},\ a = 10^{-1}, b = 10\,. \tag{24}$$

In (23) we let η to vary in $(-1, 0)$ so to evaluate increasingly less strong singularities in $x = 0$; on the other hand in (24) we let u to vary in $(0.01, 0.1)$. In Table 1 we collect the performance showcased by the two algorithms and MTQR for the test cases (19) – (22); in particular we compare both the relative error (7) computed for each algorithm and the number of function evaluations required to reach such precision. To obtain the listed results we used the function **dqagea**, implementing the adaptive G-K (aG-K) technique, for *IIPBF* and the function **fri**, implementing several adaptive G-L (aG-L) quadrature rules, for *BESSELINT*; we remark that due to the fact that the two toolboxes work on different values for the breakpoint of the finite range integral, we used $x_0 = 1$ as a fixed reference value for the upper endpoint of the integration interval in all the test cases. From these results we can see how MTQR consistently provides more accurate approximations of the finite range integral with a significantly smaller number of functions evaluations. This last consideration stems from the advantage of MTQR not being an adaptive technique but rather a one-shot application of transformed G-L quadrature samples (using an ad-hoc monomial transformation) to better capture the singularity in $x = 0$.

The results for the tests (23) – (24) are depicted in Fig. 3 and Fig. 4 respectively. In (23) the kernel $f(x)$ is itself a Bessel function of the first kind whose (real) order η is sampled uniformly in $(-1, 0)$ while we keep the orders of the two Bessel functions in the product, ν and μ, at fixed (integer) equal values ($\nu = \mu = 0$). With this instance we circumvent both the limitations of *IIPBF* and *BESSELINT* since for the former we consider $f(x) = J_\mu(\sqrt{2}x)$ while for the latter we set $m = 0$ and $n = 3$ so that it is interpreted as a product of 3 Bessel functions. We notice how MTQR achieves consistently more accurate results (with one exception) and always with a significant smaller amount of numerical evaluations for the integrand function. This provides a tool for both *IIPBF* and *BESSELINT* to be potentially integrated in order to achieve a much more cost effective solution integrating whatever function with endpoint singularities.

In (24) we sample u uniformly in $(0.01, 0.1)$ for the kernel function while we keep fixed both the (fractional) orders of the two Bessel functions and the (real) coefficients of their arguments. Also in this case we can see how MTQR achieves more precise results compared only against aG-L of *BESSELINT* and only for the lower portion of the sampled interval whereas we notice similar precisions reached by aG-K of *IIPBF* albeit with an enormous amount of computational effort in contrast against MTQR. Finally we highlight how test case (24) is part of a different kind of problem than those for which MTQR was originally designed, as the integrand function is nearly singular rather than singular; this test case thus forces MTQR beyond its scope of applicability yet resulting in optimal performances in terms of precision and more efficient computations in terms of efficiency.

With this set of numerical tests we thus delivered a simple example of how MTQR can be incorporated in multipurpose software, tackling different areas of scientific computing, as an efficient, reliable and robust third-party accessory to improve the accuracy and computational efficiency when dealing with integrals featuring endpoint singular or irregular behaviour. In the following Subsection we provide further evidence of this by using MTQR in the context of Galerkin methods for PDEs.

Table 1
Comparison of the results obtained by aG-K (used in *IIPBF*) and aG-L (used in *BESSELINT*) with those achieved by MTQR on test cases (19) – (22). For each implementation we report the relative error $R_n(f) = \frac{|I(f)-I_n|}{|I(f)|}$, as defined in (7), associated to the numerical integral computed by the relevant algorithm; the word *exact* entails that the relative error falls below the machine epsilon in double f.p. format. The reference for the analytical value of the finite range integral $I = \int_0^1 f(x)J_\nu(ax)J_\mu(bx)dx$ is provided by means of symbolic computations using Wolfram Mathematica [92]. Furthermore, as a measure of efficiency, we report the number of function evaluations (under the columns labelled Fun. Eval.) performed by each method to reach the specified relative error. Here by *number of function evaluations* we refer to the number of times the integrand function $f(x)J_\nu(ax)J_\mu(bx)$ is evaluated. We reiterate that MTQR is not adaptive, as opposed to both aG-K and aG-L, and therefore its number of function evaluations will (always) coincide with the cardinality of the set of quadrature samples outputted by the algorithm.

	Results					
Test case	aG-K		aG-L		MTQR	
	$R_n(f)$	Fun. Eval.	$R_n(f)$	Fun. Eval.	$R_n(f)$	Fun. Eval.
(19)	4.02×10^{-15}	5000	9.71×10^{-16}	272	exact	20
(20)	1.76×10^{-15}	5000	3.91×10^{-16}	306	exact	30
(21)	3.39×10^{-15}	5000	4.23×10^{-16}	706	exact	50
(22)	1.47×10^{-15}	5000	1.47×10^{-15}	546	2.67×10^{-16}	44

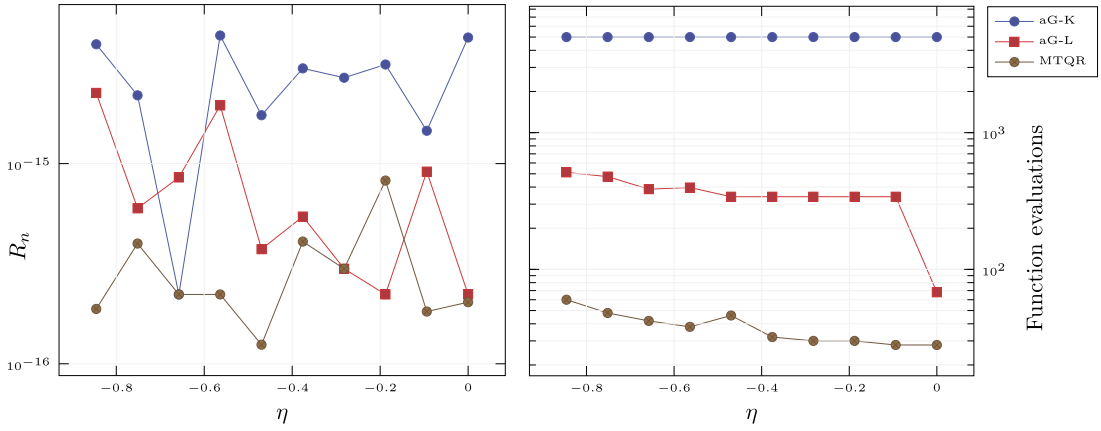


Fig. 3. Relative error (left) and number for function evaluations (right) obtained by aG-K (*IIPBF*), aG-L (*BESSELINT*) and MTQR for test case (23); both are in log-scale on the vertical axis. (For interpretation of the colours in the figure, the reader is referred to the web version of this article.)

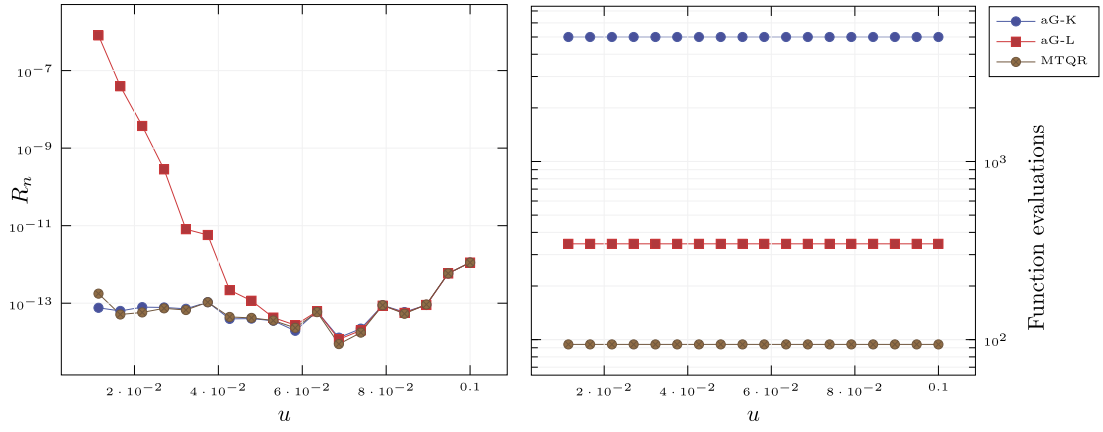


Fig. 4. Relative error (left) and number for function evaluations (right) obtained by aG-K (*IIPBF*), aG-L (*BESSELINT*) and MTQR for test case (24); both are in log-scale on the vertical axis. (For interpretation of the colours in the figure, the reader is referred to the web version of this article.)

4.2. Mass matrix computation in vector finite elements enriched by singular basis functions

We propose an effective application of the proposed algorithm to FEM in computational electromagnetics amenable of diffraction phenomena although the workflow we hereby present is easily generalised to other applications of computational physics. We particularly refer to 2.5-dimensional problems s.a. electromagnetic propagation in metallic waveguides in the presence of sharp material/geometrical discontinuities (septum, wedge,

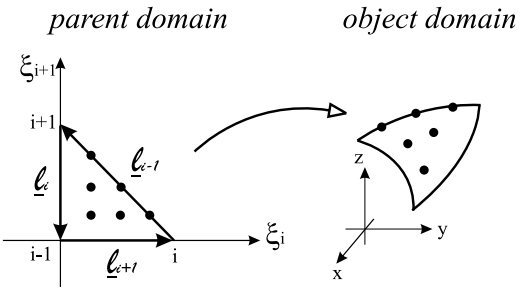


Fig. 5. Triangular cell of a meshed (object) domain and reference rectilinear triangular element in parent domain with parent coordinates $\xi_i + \xi_{i+1} + \xi_{i-1} = 1$ by applying a polynomial mapping between the domains $\mathbf{r}(\xi_i, \xi_{i+1}, \xi_{i-1})$, see [95,96].

etc...) that generates singular/irregular field behaviour. We recall that singular/irregular longitudinal and transverse field behaviours can be potentially excited near such structures:

$$E_z = j\omega\mu_0 A\rho^\nu \sin\nu\phi \tag{25}$$

$$H_t = \frac{\nu A}{\rho^{1-\nu}}\left(\sin\nu\phi\,\hat{\phi} - \cos\nu\phi\,\hat{\rho}\right) \tag{26}$$

$$H_z = j\omega\varepsilon_0 B\rho^\nu \cos\nu\phi + \text{constant} \tag{27}$$

$$E_t = -\frac{\nu B}{\rho^{1-\nu}}\left(\cos\nu\phi\,\hat{\phi} + \sin\nu\phi\,\hat{\rho}\right) \tag{28}$$

where A and B are appropriate coefficients and with singularity coefficient $\nu \geq 1/2$ [93]. In particular, we consider the elemental contributions of transverse field components to the elemental mass matrix assembled in 2.5-dimensional, additive, curl-conforming vector finite elements [94] using the Galerkin method and embedding singular behaviour to proper modelling such sharp discontinuities.

Table 2
Lowest-Order Triangular Curl-Conforming Bases [5] for transverse field components in 2.5 dimensional problems with subscripts counted modulo 3, and $i = 1, 2, 3$. On top regular bases, on bottom singular additive bases with $\nu \geq 1/2$.

Regular Functions
$\mathbf{\Omega}_\beta(\mathbf{r}) = \xi_{\beta+1}\nabla\xi_{\beta-1} - \xi_{\beta-1}\nabla\xi_{\beta+1}$
for $\beta = i, i \pm 1$
Wedge Functions
$\mathbf{\Omega}^s_{i\pm1}(\mathbf{r}) = \nabla\left[\xi_{i\mp1}\left(1-(1-\xi_i)^{\nu-1}\right)\right]$
$\mathbf{\Omega}^s_i(\mathbf{r}) = (1-\nu)\left((1-\xi_i)^\nu - 1\right)\mathbf{\Omega}_i(\mathbf{r})$

We hereby examine the results obtained by using parent-object domains discretised by triangular meshes and lowest-order triangular elements for transverse field components in parent coordinates on the reference rectilinear triangle as defined in [5], also readily available in Table 2 and Figs. 5 and 6 for simplicity.

In order to validate the performance and potential impact and contribution in FEM of MTQR we assemble the elemental mass matrix for the lowest order curl-conforming vector basis functions defined over both linear and curvilinear triangular elements. We consider the presence of a singularity in node i by applying the quadrature rule in the parent domain with change of variable $\chi = 1 - \xi_i$ to avoid numerical cancellation (mentioned previously in the context of node clustering in proximity of the singular point $\xi_i = 1$)

$$\int_0^1 \int_0^{1-\xi_i} \mathbf{\Psi}_\mathbf{k}(\mathbf{r}) \cdot \mathbf{\Psi}_\mathbf{h}(\mathbf{r}) J(\mathbf{r}) d\xi_{i+1} d\xi_i \,. \tag{29}$$

In (29) $J(\mathbf{r})$ is the Jacobian of the transformation from the object-space triangle to the parent coordinate reference triangle and $\mathbf{\Psi}_\mathbf{k}(\mathbf{r})$, $k = 1, 6$ are the six ordered basis function, i.e. (a) $\mathbf{\Omega}_i(\mathbf{r})$, (b) $\mathbf{\Omega}_{i+1}(\mathbf{r})$, (c) $\mathbf{\Omega}_{i-1}(\mathbf{r})$, (d) $\mathbf{\Omega}^s_i(\mathbf{r})$, (e) $\mathbf{\Omega}^s_{i+1}(\mathbf{r})$, (f) $\mathbf{\Omega}^s_{i-1}(\mathbf{r})$. We recall that due to the definitions of $\nabla\xi_\beta$ [95,96], the integrand of (29) shows a polynomial term $J(\mathbf{r})$ at the denominator for curvilinear triangular elements described by polynomial shape functions (in case of straight triangular elements $J(\mathbf{r})$ is constant).

4.2.1. Rectilinear triangular element

For the sake of reference and simplicity, here we compute the entries (29) to the elemental mass matrix for a rectilinear triangular element in the object domain coincident with the reference triangle (node i at $(1,0)$, node $i+1$ at $(1,0)$, node $i-1$ at $(0,0)$). We consider a singular node at

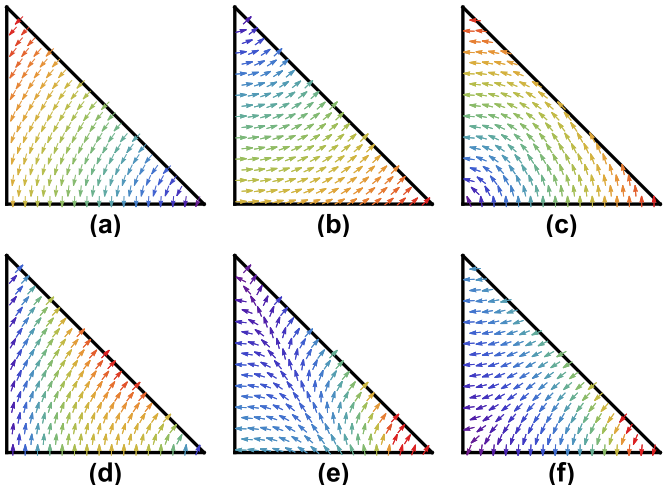


Fig. 6. Vector plots of Lowest-Order Triangular Curl-Conforming Bases [5] for transverse field components in 2.5 dimensional problems with reference to Fig. 5 and Table 2: (a) $\mathbf{\Omega}_i(\mathbf{r})$, (b) $\mathbf{\Omega}_{i+1}(\mathbf{r})$, (c) $\mathbf{\Omega}_{i-1}(\mathbf{r})$, (d) $\mathbf{\Omega}_i^s(\mathbf{r})$, (e) $\mathbf{\Omega}_{i+1}^s(\mathbf{r})$, (f) $\mathbf{\Omega}_{i-1}^s(\mathbf{r})$ and considering a singularity at node i. Temperature colour map is applied for intensity, normalised for each basis function. (For interpretation of the colours in the figure, the reader is referred to the web version of this article.)

i with sharp knife singularity, i.e. $\nu = 1/2$. With the numbering scheme reported above, the elemental mass matrix with exact, reference values, is computed using Wolfram Mathematica [92] and it reads

$$\begin{bmatrix} \frac{1}{3} & -\frac{1}{6} & 0 & -\frac{1}{54} & -\frac{1}{15} & -\frac{2}{15} \\ \times & \frac{1}{3} & 0 & \frac{13}{756} & \frac{1}{10} & \frac{1}{5} \\ \times & \times & \frac{1}{6} & \frac{1}{189} & \frac{1}{10} & \frac{1}{30} \\ \times & \times & \times & \frac{1}{540} & \frac{29}{2520} & \frac{19}{1260} \\ \times & \times & \times & \times & \frac{1}{4} & \frac{5}{24} \\ \times & \times & \times & \times & \times & \frac{1}{4} \end{bmatrix}. \tag{30}$$

To clarify the utility of a special quadrature, we report the matrix relative errors of the mass matrix computed via a G-L quadrature using 24×24 samples with respect to (30):

$$\begin{bmatrix} 5.96179 \cdot 10^{-17} & 5.96179 \cdot 10^{-17} & 1.21025 \cdot 10^{-18} & 6.63256 \cdot 10^{-12} & 1.98943 \cdot 10^{-7} & 8.19178 \cdot 10^{-8} \\ \times & 5.12593 \cdot 10^{-17} & 1.82846 \cdot 10^{-19} & 1.34229 \cdot 10^{-9} & 1.22758 \cdot 10^{-4} & 4.38496 \cdot 10^{-5} \\ \times & \times & 5.40269 \cdot 10^{-17} & 2.91604 \cdot 10^{-9} & 7.02045 \cdot 10^{-5} & 2.10543 \cdot 10^{-4} \\ \times & \times & \times & 6.63251 \cdot 10^{-11} & 5.74254 \cdot 10^{-7} & 3.61148 \cdot 10^{-7} \\ \times & \times & \times & \times & 9.81007 \cdot 10^{-5} & 7.56777 \cdot 10^{-5} \\ \times & \times & \times & \times & \times & 5.60575 \cdot 10^{-5} \end{bmatrix}. \tag{31}$$

As reference, we also propose the matrix relative errors computed via adaptive GSL's built-in routine QAGS [4], which implements a 21-point aG-K technique, with respect to (30)

$$\begin{bmatrix} \text{exact} & \text{exact} & 6.93889 \cdot 10^{-18} & 2.07959 \cdot 10^{-14} & 2.08167 \cdot 10^{-16} & 2.08167 \cdot 10^{-16} \\ \times & \text{exact} & 8.67362 \cdot 10^{-19} & 2.6229 \cdot 10^{-15} & \text{exact} & 2.77556 \cdot 10^{-16} \\ \times & \times & \text{exact} & 6.06546 \cdot 10^{-15} & 1.38778 \cdot 10^{-16} & 2.08167 \cdot 10^{-16} \\ \times & \times & \times & 9.25041 \cdot 10^{-15} & 1.50741 \cdot 10^{-16} & 4.60158 \cdot 10^{-16} \\ \times & \times & \times & \times & 4.44089 \cdot 10^{-16} & 1.33227 \cdot 10^{-16} \\ \times & \times & \times & \times & \times & 2.22045 \cdot 10^{-16} \end{bmatrix}. \tag{32}$$

Finally, we report the matrix relative errors computed by MTQR (which we reiterate being non-adaptive) with respect to (30)

$$\begin{bmatrix} 9.66863 \cdot 10^{-18} & 9.89643 \cdot 10^{-17} & 1.20703 \cdot 10^{-17} & 5.2196 \cdot 10^{-18} & 6.67898 \cdot 10^{-17} & 2.21996 \cdot 10^{-17} \\ \times & 7.72377 \cdot 10^{-17} & 2.41407 \cdot 10^{-18} & 6.55449 \cdot 10^{-17} & 7.77306 \cdot 10^{-17} & 3.88997 \cdot 10^{-17} \\ \times & \times & 4.10268 \cdot 10^{-17} & 1.35925 \cdot 10^{-16} & 1.22925 \cdot 10^{-17} & 5.10886 \cdot 10^{-17} \\ \times & \times & \times & 5.05374 \cdot 10^{-16} & 7.24108 \cdot 10^{-17} & 2.78953 \cdot 10^{-17} \\ \times & \times & \times & \times & 2.96314 \cdot 10^{-17} & 3.37762 \cdot 10^{-17} \\ \times & \times & \times & \times & \times & 1.79792 \cdot 10^{-17} \end{bmatrix}. \tag{33}$$

For what concerns the results in (33) the double integration (29) with solely regular bases is performed by a product of classical G-L quadrature rule with 2×2 samples. If the integrand of (29) contains irregular/singular bases we apply a product quadrature rule with 2×18 samples where the first 2 are related to a G-L quadrature rule (inner integral in ξ_{i+1}) and the remaining 18 samples are related to MTQR with $\lambda \in (0, 4)$ (outer integral in ξ_i). We observe that the selection of the number of samples in MTQR and G-L derives from the a-priori analytical study of the six integrands defined in (29) which are either classical or generalised Müntz polynomials for regular and singular bases respectively. In particular we note that the inner integration is performed in ξ_{i+1}, yielding a regularisation effect on outer integration in ξ_i.

By comparing (33) with (32) we observe the full convergence of the non-adaptive MTQR (with only 36 function evaluations per irregular integrand) with respect to the performance of the aG-K technique underlying GSL-QAGS. We also note that product quadrature rules using MTQR are computed once and for all elements of a FEM application, while adaptive integration algorithm must be run for each elements yielding an increasingly less efficient solution for large, multiscale simulations in computational physics.

4.2.2. Curvilinear triangular element

Often, finite element applications use polynomial shape functions to model curvilinear geometries. In this framework, the integrand of (29) shows a polynomial behaviour $J(\mathbf{r})$ at the denominator resulting in a rational integrand function which cannot be directly, globally and analytically represented in terms of either classical or generalised polynomials. For demonstrating purposes, in this test case we consider shape functions of order 2 for the geometry with control points in object domain located at $node\,i\,(1,0)$, $node\,i+1\,(0,1)$, $node\,i-1\,(0,0)$, and $(1/\sqrt{2},1/\sqrt{2})$, $(0,1/2)$, $(1/2,0)$. The resulting curvilinear triangle is a deformed version of the reference triangular element along the edge $i-1$ (see Fig. 5). We select a singular node at i with sharp knife singularity $\nu=1/2$.

With the usual numbering scheme adopted for the linear case, the elemental mass matrix with reference values is again computed using Wolfram Mathematica [92]

$$\begin{bmatrix} 0.2824683905612152 & -0.0799881369933545 & -0.0511288002782521 & -0.0134875272625489 & -0.0590162878603002 & -0.0675515579944396 \\ \times & 0.2824683905612152 & -0.0511288002782521 & 0.0087648058609067 & 0.0291436298001587 & 0.1136894684054350 \\ \times & \times & 0.1845734601172637 & 0.0052347248641515 & 0.0683970043032625 & -0.0045452822860420 \\ \times & \times & \times & 0.0012241429923909 & 0.0078405230975909 & 0.0084234049826403 \\ \times & \times & \times & \times & 0.1721840539890618 & 0.1098626273234278 \\ \times & \times & \times & \times & \times & 0.1397771801706621 \end{bmatrix}. \tag{34}$$

As reference, below we propose the matrix relative errors computed with QAGS's aG-K technique with respect to (34)

$$\begin{bmatrix} \text{exact} & 6.93992\cdot10^{-16} & 1.35714\cdot10^{-16} & 1.72347\cdot10^{-14} & 3.76243\cdot10^{-15} & 4.1088\cdot10^{-16} \\ \times & \text{exact} & 1.35714\cdot10^{-16} & 2.96879\cdot10^{-15} & 3.16664\cdot10^{-14} & 7.32405\cdot10^{-16} \\ \times & \times & 3.00754\cdot10^{-16} & 2.17059\cdot10^{-14} & 8.11602\cdot10^{-16} & 4.57984\cdot10^{-15} \\ \times & \times & \times & 1.95382\cdot10^{-13} & 8.62879\cdot10^{-15} & 4.11882\cdot10^{-16} \\ \times & \times & \times & \times & 1.61197\cdot10^{-16} & 1.26319\cdot10^{-16} \\ \times & \times & \times & \times & \times & 1.9857\cdot10^{-16} \end{bmatrix}. \tag{35}$$

Finally, we report the matrix relative errors computed using MTQR with respect to (34)

$$\begin{bmatrix} 6.99866\cdot10^{-17} & 6.74668\cdot10^{-16} & 4.26045\cdot10^{-16} & 4.38676\cdot10^{-16} & 3.23314\cdot10^{-15} & 7.64195\cdot10^{-16} \\ \times & 5.79219\cdot10^{-17} & 3.24033\cdot10^{-16} & 6.5069\cdot10^{-16} & 2.61723\cdot10^{-14} & 7.06715\cdot10^{-16} \\ \times & \times & 2.73857\cdot10^{-16} & 1.98204\cdot10^{-16} & 8.36839\cdot10^{-16} & 6.34363\cdot10^{-15} \\ \times & \times & \times & 4.44293\cdot10^{-16} & 6.9585\cdot10^{-15} & 7.96629\cdot10^{-16} \\ \times & \times & \times & \times & 7.62109\cdot10^{-17} & 9.69835\cdot10^{-17} \\ \times & \times & \times & \times & \times & 1.7804\cdot10^{-16} \end{bmatrix}. \tag{36}$$

For what concerns the results in (36), in this occasion, the double integration (29) with solely regular bases is performed by a product of classical G-L quadrature rule with 10×10 samples. If the integrand of (29) contains irregular/singular bases we then apply a product quadrature rule with 10×30 samples where the first 10 are related to a G-L quadrature rule (inner integral in ξ_{i+1}) and the remaining 30 samples are related to MTQR with $\lambda\in(0,14)$ (outer integral in ξ_i). We observe that the selection of the number of samples in MTQR and G-L derives from the a-priori analytical study of the six integrands defined in (29) for the selected case of curvilinear triangular element which are in general rational polynomials containing either a classical polynomial numerator or a generalised Müntz polynomial numerator respectively for regular and singular bases respectively.

We observe that, although the denominator of the integrand, i.e. the Jacobian $J(\mathbf{r})$, has a smooth polynomial behaviour for "regular" curvilinear elements, it has a strong impact on limiting the convergence of the numerical integration. In particular, it yields an increase of iterations in adaptive quadrature rules and an increase of λ-range in MTQR. By comparing (36) with (35) we observe the full convergence of the non-adaptive, more computationally efficient MTQR with respect to the performance of GSL-QAGS.

We rebate that product quadrature rules using MTQR are computed a-priori once and for all elements of a FEM application, while adaptive integration algorithm must be repeated for each element. Moreover, in the framework of 2.5D FEM applications with singular modelling, the proposed product quadrature exceeds in precision and efficiency frequently used traditional schemes of singular integration mainly based on singularity subtraction or cancellation techniques.

5. Conclusion

In this work we contribute to enrich the network of existing high-precision numerical integration software by issuing an additional fast, light, portable, and integrable C++ suite, that automatically optimises transformations capable of manipulating the traditional G-L quadrature rule to fully capture the endpoint singularities/irregularities in those integrands that are modelled by sets of generalised polynomials of non-integer degree. MTQR is a non-iterative mathematical software that guarantees the required numerical precision (we selected double precision) and it avoids to resort to computationally expensive algorithms such as adaptive quadrature rules. To the best of our knowledge, there is no other open-source alternative in existence that matches the precision of MTQR on such a large scale of integrand functions when combined with the minimum computational cost assured by the algorithm. In the aforementioned *user manual* the reader will find useful notions and instructions for its installations and modes of executions in both Linux and Windows. In this work we have shown reliability in MTQR's accuracy on a wide range of benchmarking tests in computational physics (dealing with arbitrary products of Bessel functions of real order and additive, curl-conforming basis functions in vector finite elements), compared its performance against well-known adaptive techniques and further proved its potential of being integrated in larger codebases as a useful third-party numerical tool. The range of application for MTQR is of particular interest in BEM and FEM formulations frequently arising in Computational Electromagnetics (CEM) and Fluid Dynamics (CFD), as well as structural analysis and

fracture mechanics or any other engineering or scientific applications modelling abrupt discontinuities. Finally we think that while the combination of MTQR with Gaussian and quasi-Gaussian (piecewise) spline rules, we argue, could lead to more refined, efficient and general purpose quadrature techniques, such implementation is not straightforward and will be investigated in future studies.

CRediT authorship contribution statement

Guido Lombardi: Conceptualization, Formal analysis, Funding acquisition, Methodology, Project administration, Supervision, Validation, Writing – original draft, Writing – review & editing. **Davide Papapicco:** Investigation, Software, Validation, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

This study was supported in part by the Italian Ministry of University and Research (MIUR) under the PRIN2017 Grant (2017NT5W7Z) *"Chipless radio frequency identification (RFID) for GREEN TAGging and Sensing - GREEN TAGS"* and by European Union under Next Generation EU - PNRR M4C2, Investimento 1.4 - Avviso n. 3138 del 16/12/2021 - CN00000013 National Centre for HPC, Big Data and Quantum Computing (HPC) - CUP E13C22000990001.

Appendix A. Primary module's source code

```
#include "mtqr.h"

// Global variable controlling the primary module mode of execution
bool loud_mode = true;

// LOUD MODE
template<typename T>
void mtqr(std::vector<T>& muntz_sequence, std::vector<T>& coeff_sequence)
{
  // Print initial message and selects user's inputs
  auto input_data = manageData(muntz_sequence, coeff_sequence);
  // Extract beta_min, beta_max and n_min
  auto monomial_data = streamMonMapData(std::get<0>(input_data));
  // Compute order of the monomial transformation
  double transf_order = computeMapOrder(std::get<1>(input_data), std::get<1>(monomial_data));
  // Compute the new nodes and weights of the Monomial Transformation Quadrature Rule
  auto quad_data = computeQuadParams(transf_order, std::get<0>(monomial_data), std::get<2>(monomial_data));
  // Cast the quadrature parameter in the most optimised f.p. format possible
  optimiseData(quad_data, muntz_sequence, coeff_sequence);
}
template void mtqr<float128>(std::vector<float128>& muntz_sequence, std::vector<float128>& coeff_sequence);
template void mtqr<double>(std::vector<double>& muntz_sequence, std::vector<double>& coeff_sequence);

// SILENT MODE
std::vector<std::vector<double>> mtqr(double lambda_min, double lambda_max)
{
  // Deactivate terminal's and files' output
  loud_mode = false;
  // Initialise input parameters of the Monomial transformation quadrature rule
  std::vector<double> muntz_sequence = {lambda_min, lambda_max};
  std::vector<double> coeff_sequence = {1.0, 1.0};
  // Print initial message and selects user's inputs
  auto input_data = manageData(muntz_sequence, coeff_sequence);
  // Extract beta_min, beta_max and n_min
  auto monomial_data = streamMonMapData(std::get<0>(input_data));
  // Compute order of the monomial transformation
  double transf_order = computeMapOrder(std::get<1>(input_data), std::get<1>(monomial_data));
  // Compute the new nodes and weights of the Monomial Transformation Quadrature Rule
  auto quad_data = computeQuadParams(transf_order, std::get<0>(monomial_data), std::get<2>(monomial_data));
  // Cast the quadrature parameter in the most optimised f.p. format possible
  optimiseData(quad_data, muntz_sequence, coeff_sequence);
  // Generate double-precise new nodes and weights and export them in memory as output
  std::vector<double> nodes = castVector(std::get<0>(quad_data), std::numeric_limits<double>::epsilon());
  std::vector<double> weights = castVector(std::get<1>(quad_data), std::numeric_limits<double>::epsilon());
  return std::vector<std::vector<double>> {nodes, weights};
}
```

References

[1] G. Lombardi, Design of quadrature rules for Müntz and Müntz-logarithmic polynomials using monomial transformation, Int. J. Numer. Methods Eng. 80 (13) (2009) 1687–1717, https://doi.org/10.1002/nme.2684.

[2] IEEE standard for floating-point arithmetic, IEEE Std 754-2019 (Revision of IEEE 754-2008, https://doi.org/10.1109/IEEESTD.2019.8766229, 2019.
[3] Boost C++ Libraries, https://www.boost.org/, 1999.
[4] B. Gough, GNU Scientific Library Reference Manual, Network Theory Ltd., 2009.
[5] R. Graglia, G. Lombardi, Singular higher order complete vector bases for finite methods, IEEE Trans. Antennas Propag. 52 (7) (2004) 1672–1685, https://doi.org/10.1109/TAP.2004.831292.
[6] P. Johnston, D. Elliott, A sinh transformation for evaluating nearly singular boundary element integrals, Int. J. Numer. Methods Eng. 62 (4) (2005) 564–578, https://doi.org/10.1002/nme.1208.
[7] G. Monegato, L. Scuderi, Numerical integration of functions with endpoint singularities and/or complex poles in 3D Galerkin boundary element methods, Publ. Res. Inst. Math. Sci. 41 (4) (2005) 869–895, https://doi.org/10.2977/PRIMS/1145474599.
[8] M. Carley, Numerical quadratures for singular and hypersingular integrals in boundary element methods, SIAM J. Sci. Comput. 29 (3) (2007) 1207–1216, https://doi.org/10.1137/060666093.
[9] L. Scuderi, On the computation of nearly singular integrals in 3D BEM collocation, Int. J. Numer. Methods Eng. 74 (11) (2008) 1733–1770, https://doi.org/10.1002/nme.2229.
[10] D. Elliott, B. Johnston, P. Johnston, Clenshaw–Curtis and Gauss–Legendre quadrature for certain boundary element integrals, SIAM J. Sci. Comput. 31 (1) (2008) 510–530, https://doi.org/10.1137/07070200X.
[11] R. Graglia, G. Lombardi, Singular higher order divergence-conforming bases of additive kind and moments method applications to 3D sharp-wedge structures, IEEE Trans. Antennas Propag. 56 (12) (2008) 3768–3788, https://doi.org/10.1109/TAP.2008.2007390.
[12] R. Graglia, G. Lombardi, Machine precision evaluation of singular and nearly singular potential integrals by use of Gauss quadrature formulas for rational functions, IEEE Trans. Antennas Propag. 56 (4) (2008) 981–998, https://doi.org/10.1109/TAP.2008.919181.
[13] G. Lombardi, R.D. Graglia, Modeling junctions in sharp edge conducting structures with higher order method of moments, IEEE Trans. Antennas Propag. 62 (11) (2014) 5723–5731, https://doi.org/10.1109/TAP.2014.2355855.
[14] H. Wu, J. Zou, Finite element method and its analysis for a nonlinear Helmholtz equation with high wave numbers, SIAM J. Numer. Anal. 56 (3) (2018) 1338–1359, https://doi.org/10.1137/17M111314X.
[15] T. Chaumont-Frelet, Mixed finite element discretizations of acoustic Helmholtz problems with high wavenumbers, Calcolo 56 (2019), https://doi.org/10.1007/s10092-019-0346-z.
[16] Y. Ordokhani, P. Rahimkhani, A numerical technique for solving fractional variational problems by Müntz—Legendre polynomials, J. Appl. Math. Comput. 58 (2018) 75–94, https://doi.org/10.1007/s12190-017-1134-z.
[17] D. Hou, M.T. Hasan, C. Xu, Müntz spectral methods for the time-fractional diffusion equation, Comput. Methods Appl. Math. 18 (1) (2018) 43–62, https://doi.org/10.1515/cmam-2017-0027.
[18] T. Fries, T. Belytschko, The extended/generalized finite element method: An overview of the method and its applications, Int. J. Numer. Methods Eng. 84 (3) (2010) 253–304, https://doi.org/10.1002/nme.2914.
[19] I. Babuška, U. Banerjee, Stable generalized finite element method (SGFEM), Comput. Methods Appl. Mech. Eng. 201–204 (2012) 91–111, https://doi.org/10.1016/j.cma.2011.09.012.
[20] S. Chen, J. Shen, Enriched spectral methods and applications to problems with weakly singular solutions, J. Sci. Comput. 77 (2018) 1468–1489, https://doi.org/10.1007/s10915-018-0862-z.
[21] L. Marin, D. Lesnic, V. Mantič, Treatment of singularities in Helmholtz-type equations using the boundary element method, J. Sound Vib. 278 (1) (2004) 39–62, https://doi.org/10.1016/j.jsv.2003.09.059.
[22] J. Shen, Y. Wang, Müntz–Galerkin methods and applications to mixed Dirichlet–Neumann boundary value problems, SIAM J. Sci. Comput. 38 (4) (2016) A2357–A2381, https://doi.org/10.1137/15M1052391.
[23] J. Jiang, M.S. Mohamed, M. Seaid, Hongqiu Li, Identifying the wavenumber for the inverse Helmholtz problem using an enriched finite element formulation, Comput. Methods Appl. Mech. Eng. 340 (2018) 615–629, https://doi.org/10.1016/j.cma.2018.06.014.
[24] E. Benvenuti, A. Chiozzi, G. Manzini, N. Sukumar, Extended virtual element method for the Laplace problem with singularities and discontinuities, Comput. Methods Appl. Mech. Eng. 356 (2019) 571–597, https://doi.org/10.1016/j.cma.2019.07.028.
[25] A. Jablonski, An effective algorithm for calculating the Chandrasekhar function, Comput. Phys. Commun. 183 (8) (2012) 1773–1782, https://doi.org/10.1016/j.cpc.2012.02.022.
[26] A. Fowlie, A fast C++ implementation of thermal functions, Comput. Phys. Commun. 228 (2018) 264–272, https://doi.org/10.1016/j.cpc.2018.02.015.
[27] A. Jablonski, The Chandrasekhar function for modeling photoelectron transport in solids, Comput. Phys. Commun. 235 (2019) 489–501, https://doi.org/10.1016/j.cpc.2018.07.005.
[28] W. Gautschi, Numerical quadrature in the presence of a singularity, SIAM J. Numer. Anal. 4 (3) (1967) 357–362, https://doi.org/10.1137/0704031.
[29] G. Monegato, Numerical evaluation of hypersingular integrals, J. Comput. Appl. Math. 50 (1) (1994) 9–31, https://doi.org/10.1016/0377-0427(94)90287-9.
[30] J. Ma, V. Rokhlin, S. Wandzura, Generalized Gaussian quadrature rules for systems of arbitrary functions, SIAM J. Numer. Anal. 33 (3) (1996) 971–996, https://doi.org/10.1137/0733048.
[31] P. Kolm, V. Rokhlin, Numerical quadratures for singular and hypersingular integrals, Comput. Math. Appl. 41 (3) (2001) 327–352, https://doi.org/10.1016/S0898-1221(00)00277-7.
[32] K. Pachucki, M. Puchalski, V. Yerokhin, Extended Gaussian quadratures for functions with an end-point singularity of logarithmic type, Comput. Phys. Commun. 185 (11) (2014) 2913–2919, https://doi.org/10.1016/j.cpc.2014.06.018.
[33] G. Milovanović, T. Igić, D. Turnić, Generalized quadrature rules of Gaussian type for numerical evaluation of singular integrals, J. Comput. Appl. Math. 278 (2015) 306–325, https://doi.org/10.1016/j.cam.2014.10.009.
[34] N. Hale, A. Townsend, Fast and accurate computation of Gauss–Legendre and Gauss–Jacobi quadrature nodes and weights, SIAM J. Sci. Comput. 35 (2) (2013) A652–A674, https://doi.org/10.1137/120889873.
[35] I. Bogaert, Iteration-free computation of Gauss–Legendre quadrature nodes and weights, SIAM J. Sci. Comput. 36 (3) (2014) A1008–A1026, https://doi.org/10.1137/140954969.
[36] T. Hughes, A. Reali, G. Sangalli, Efficient quadrature for NURBS-based isogeometric analysis, Comput. Methods Appl. Mech. Eng. 199 (5) (2010) 301–313, https://doi.org/10.1016/j.cma.2008.12.004.
[37] M. Bartoň, V.M. Calo, Gauss–Galerkin quadrature rules for quadratic and cubic spline spaces and their application to isogeometric analysis, Comput. Aided Des. 82 (2017) 57–67, https://doi.org/10.1016/j.cad.2016.07.003.
[38] K.A. Johannessen, Optimal quadrature for univariate and tensor product splines, Comput. Methods Appl. Mech. Eng. 316 (2017) 84–99, https://doi.org/10.1016/j.cma.2016.04.030, Special Issue on Isogeometric Analysis: Progress and Challenges.
[39] H. Curry, I. Schoenberg, On Pólya frequency functions IV: The fundamental spline functions and their limits, J. Anal. Math. 17 (1) (1966) 71–107, https://doi.org/10.1007/BF02788653.
[40] M.J. Marsden, An identity for spline functions with applications to variation-diminishing spline approximation, J. Approx. Theory 3 (1) (1970) 7–49, https://doi.org/10.1016/0021-9045(70)90058-4.
[41] C. de Boor, On calculating with b-splines, J. Approx. Theory 6 (1) (1972) 50–62, https://doi.org/10.1016/0021-9045(72)90080-9.
[42] R.R. Hiemstra, F. Calabrò, D. Schillinger, T.J. Hughes, Optimal and reduced quadrature rules for tensor product and hierarchically refined splines in isogeometric analysis, Comput. Methods Appl. Mech. Eng. 316 (2017) 966–1004, https://doi.org/10.1016/j.cma.2016.10.049, Special Issue on Isogeometric Analysis: Progress and Challenges.
[43] M. Bartoň, R. Ait-Haddou, V.M. Calo, Gaussian quadrature rules for C1 quintic splines with uniform knot vectors, J. Comput. Appl. Math. 322 (2017) 57–70, https://doi.org/10.1016/j.cam.2017.02.022.
[44] A. Logg, G. Wells, DOLFIN: automated finite element computing, ACM Trans. Math. Softw. 37 (2010), https://doi.org/10.1145/1731022.1731030.
[45] R. Kirby, Algorithm 839: FIAT, a new paradigm for computing finite element basis functions, ACM Trans. Math. Softw. 30 (2004) 502–516, https://doi.org/10.1145/1039813.1039820.
[46] R. Anderson, J. Andrej, A. Barker, J. Bramwell, J.-S. Camier, J. Cerveny, V. Dobrev, Y. Dudouit, A. Fisher, T. Kolev, W. Pazner, M.S.V. Tomov, I. Akkerman, J. Dahm, D. Medina, S. Zampini, MFEM: A modular finite element methods library, in: Development and Application of Open-Source Software for Problems with Numerical PDEs, Comput. Math. Appl. 81 (2021) 42–74, https://doi.org/10.1016/j.camwa.2020.06.009.
[47] P.C. Africa, lifex: A flexible, high performance library for the numerical solution of complex finite element problems, SoftwareX 20 (2022) 101252, https://doi.org/10.1016/j.softx.2022.101252.

[48] D. Arndt, W. Bangerth, D. Davydov, T. Heister, L. Heltai, M. Kronbichler, M. Maier, J.-P. Pelteret, B. Turcksin, D. Wells, The deal.II finite element library: Design, features, and insights, in: Development and Application of Open-Source Software for Problems with Numerical PDEs, Comput. Math. Appl. 81 (2021) 407–422, https://doi.org/10.1016/j.camwa.2020.02.022.
[49] S. Badia, A.F. Martín, A tutorial-driven introduction to the parallel finite element library FEMPAR v1.0.0, Comput. Phys. Commun. 248 (2020) 107059, https://doi.org/10.1016/j.cpc.2019.107059.
[50] F. Verdugo, S. Badia, The software design of Gridap: A finite element package based on the Julia JIT compiler, Comput. Phys. Commun. 276 (2022) 108341, https://doi.org/10.1016/j.cpc.2022.108341.
[51] M.W. Scroggs, I. Baratta, C. Richardson, G. Wells, Basix: a runtime finite element basis evaluation library, J. Open Sour. Softw. 7 (73) (2022) 3982, https://doi.org/10.21105/joss.03982.
[52] Z.-G. Yan, Y. Pan, G. Castiglioni, K. Hillewaert, J. Peiró, D. Moxey, S.J. Sherwin, Nektar++: Design and implementation of an implicit, spectral/hp element, compressible flow solver using a jacobian-free Newton–Krylov approach, in: Development and Application of Open-Source Software for Problems with Numerical PDEs, Comput. Math. Appl. 81 (2021) 351–372, https://doi.org/10.1016/j.camwa.2020.03.009.
[53] F. Xu, Q. Xiong, V. Aizinger, G. Ducrozet, Development and application of open-source software for problems with numerical PDEs, in: Development and Application of Open-Source Software for Problems with Numerical PDEs, Comput. Math. Appl. 81 (2021) 1–2, https://doi.org/10.1016/j.camwa.2020.12.002.
[54] G. Stabile, G. Rozza, Finite volume POD-Galerkin stabilised reduced order methods for the parametrised incompressible Navier-Stokes equations, Comput. Fluids (2018), https://doi.org/10.1016/j.compfluid.2018.01.035.
[55] M.W. Hess, A. Lario, G. Mengaldo, G. Rozza, Reduced order modeling for spectral element methods: current developments in Nektar++ and further perspectives, 2022.
[56] V. Milovanovic, G.V. Gradimir, Müntz orthogonal polynomials and their numerical evaluation, in: Applications and Computation of Orthogonal Polynomials, 1999, pp. 179–194.
[57] J. Almira, Müntz type theorems I, Surv. Approx. Theory 3 (2007) 152–194.
[58] A. Stroud, D. Secrest, Gaussian Quadrature Formulas, Prentice-Hall, 1966.
[59] G. Szegő, Orthogonal Polynomials, Colloquium Publications, vol. 23, American Mathematical Society, 1939.
[60] W. Barrett, Convergence properties of gaussian quadrature formulae, Comput. J. 3 (4) (1961) 272–277, https://doi.org/10.1093/comjnl/3.4.272.
[61] D. Elliott, Uniform asymptotic expansions of the Jacobi polynomials and an associated function, Math. Comput. 25 (114) (1971) 309–315, https://doi.org/10.2307/2004926.
[62] J.D. Donaldson, D. Elliott, A unified approach to quadrature rules with asymptotic estimates of their remainders, SIAM J. Numer. Anal. 9 (4) (1972) 573–602, https://doi.org/10.1137/0709051.
[63] M. Holmes, R. Kashyap, R. Wyatt, Physical properties of optical fiber sidetap grating filters: free-space model, IEEE J. Sel. Top. Quantum Electron. 5 (5) (1999) 1353–1365, https://doi.org/10.1109/2944.806761.
[64] G. Fikioris, P. Cottis, A. Panagopoulos, On an integral related to biaxially anisotropic media, J. Comput. Appl. Math. 146 (2) (2002) 343–360, https://doi.org/10.1016/S0377-0427(02)00368-0.
[65] R. Golubović, A.G. Polimeridis, J.R. Mosig, The weighted averages method for semi-infinite range integrals involving products of Bessel functions, IEEE Trans. Antennas Propag. 61 (11) (2013) 5589–5596, https://doi.org/10.1109/TAP.2013.2280048.
[66] K.A. Michalski, J.R. Mosig, Efficient computation of Sommerfeld integral tails – methods and algorithms, J. Electromagn. Waves Appl. 30 (3) (2016) 281–317, https://doi.org/10.1080/09205071.2015.1129915.
[67] M.P.S. dos Santos, J.A.F. Ferreira, J.A.O. Simões, R. Pascoal, J. Torrão, X. Xue, E.P. Furlani, Magnetic levitation-based electromagnetic energy harvesting: a semi-analytical non-linear model for energy transduction, Sci. Rep. 6 (2016), https://doi.org/10.1038/srep18579.
[68] J.M. Roesset, Nondestructive dynamic testing of soils and pavements, J. Appl. Sci. Eng. 1 (1999) 61–81, https://doi.org/10.6180/jase.1998.1.2.01.
[69] R. Craster, Scattering by cracks beneath fluid-solid interfaces, J. Sound Vib. 209 (2) (1998) 343–372, https://doi.org/10.1006/jsvi.1997.1252.
[70] N. Robinson, An isotropic elastic medium containing a cylindrical borehole with a rigid plug, Int. J. Solids Struct. 39 (19) (2002) 4889–4904, https://doi.org/10.1016/S0020-7683(02)00414-6.
[71] M. Bevis, E. Pan, H. Zhou, F. Han, R. Zhu, Surface deformation due to loading of a layered elastic half-space: constructing the solution for a general polygonal load, Acta Geophys. 63 (2016) 957–977, https://doi.org/10.1515/acgeo-2015-0034.
[72] M.A. Ceballos, Numerical evaluation of integrals involving the product of two Bessel functions and a rational fraction arising in some elastodynamic problems, J. Comput. Appl. Math. 313 (2017) 355–382, https://doi.org/10.1016/j.cam.2016.09.043.
[73] A.M.J. Davis, Drag modifications for a sphere in a rotational motion at small, non-zero Reynolds and Taylor numbers: wake interference and possibly Coriolis effects, J. Fluid Mech. 237 (1992) 13–22, https://doi.org/10.1017/S002211209200332X.
[74] J.P. Tanzosh, H.A. Stone, Motion of a rigid particle in a rotating viscous flow: an integral equation approach, J. Fluid Mech. 275 (1994) 225–256, https://doi.org/10.1017/S002211209400234X.
[75] D.M. Tartakovsky, J.D. Moulton, V.A. Zlotnik, Kinematic structure of minipermeameter flow, Water Resour. Res. 36 (9) (2000) 2433–2442, https://doi.org/10.1029/2000WR900178.
[76] J. Sherwood, Optimal probes for withdrawal of uncontaminated fluid samples, Phys. Fluids 17 (2005), https://doi.org/10.1063/1.2006128.
[77] G. Ledder, V.A. Zlotnik, Evaluation of oscillatory integrals for analytical groundwater flow and mass transport models, Adv. Water Resour. 104 (2017) 284–292, https://doi.org/10.1016/j.advwatres.2017.04.007.
[78] S. Groote, J. Körner, A. Pivovarov, On the evaluation of sunset-type Feynman diagrams, Nucl. Phys. B 542 (1) (1999) 515–547, https://doi.org/10.1016/S0550-3213(98)00812-8.
[79] J.T. Conway, Analytical solutions for the newtonian gravitational field induced by matter within axisymmetric boundaries, Mon. Not. R. Astron. Soc. 316 (3) (2000) 540–554, https://doi.org/10.1046/j.1365-8711.2000.03523.x.
[80] M. Mobilia, Competition between homogeneous and local processes in a diffusive many-body system, J. Stat. Mech. Theory Exp. 04 (2005), https://doi.org/10.1088/1742-5468/2005/04/P04003.
[81] J. Salo, H.M. El-Sallabi, P. Vainikainen, Statistical analysis of the multiple scattering radio channel, IEEE Trans. Antennas Propag. 54 (11) (2006) 3114–3124, https://doi.org/10.1109/TAP.2006.883964.
[82] A. Kisselev, Approximate formula for the total cross section for a moderately small eikonal function, Theor. Math. Phys. 201 (2019) 1484–1502, https://doi.org/10.1134/S0040577919100064.
[83] M. Ikonomou, P. Köhler, A.F. Jacob, Computation of integrals over the half-line involving products of Bessel functions, with application to microwave transmission lines, Z. Angew. Math. Mech. 75 (11) (1995) 917–926, https://doi.org/10.1002/zamm.19950751109.
[84] H.A. Stone, H.M. McConnell, Hydrodynamics of quantized shape transitions of lipid domains, Proc. Math. Phys. Sci. 448 (1932) (1995) 97–111.
[85] X. You-Sheng, L. Ji, L. Hua-Mei, W. Feng-Min, Analysis for the potential function of the digital microstructure image of porous media, Commun. Theor. Phys. 40 (4) (2003) 393, https://doi.org/10.1088/0253-6102/40/4/393.
[86] N.P. Singh, T. Mogi, Electromagnetic response of a large circular loop source on a layered earth: a new computation method, Pure Appl. Geophys. 162 (2005) 181–200, https://doi.org/10.1007/s00024-004-2586-2.
[87] E.P. Petrov, P. Schwille, Translational diffusion in lipid membranes beyond the Saffman-Delbrück approximation, Biophys. J. 94 (5) (2008) L41–L43, https://doi.org/10.1529/biophysj.107.126565.
[88] S. Lucas, Evaluating infinite integrals involving products of Bessel functions of arbitrary order, J. Comput. Appl. Math. 64 (3) (1995) 269–282, https://doi.org/10.1016/0377-0427(95)00143-3.
[89] J.T. Ratnanather, J.H. Kim, S. Zhang, A.M.J. Davis, S.K. Lucas, Algorithm 935: IIPBF, a MATLAB toolbox for infinite integral of products of two Bessel functions, ACM Trans. Math. Softw. 40 (2) (mar 2014), https://doi.org/10.1145/2508435.
[90] J.V. Deun, R. Cools, Algorithm 858: Computing infinite range integrals of an arbitrary product of Bessel functions, ACM Trans. Math. Softw. 32 (4) (2006) 580–596, https://doi.org/10.1145/1186785.1186790.
[91] J. Van Deun, R. Cools, Integrating products of Bessel functions with an additional exponential or rational factor, Comput. Phys. Commun. 178 (8) (2008) 578–590, https://doi.org/10.1016/j.cpc.2007.11.010.
[92] W. R. Inc., Mathematica, Version 13.2, Champaign, IL, 2022, https://www.wolfram.com/mathematica, 2023.
[93] J.V. Bladel, Singular Electromagnetic Fields and Sources, Oxford University Press, 1991.

[94] J. Nedelec, Mixed finite elements in $\mathbb{R}^3$, Numer. Math. 35 (1980) 315–341, https://doi.org/10.1007/BF01396415.
[95] R. Graglia, D. Wilton, A. Peterson, Higher order interpolatory vector bases for computational electromagnetics, IEEE Trans. Antennas Propag. 45 (3) (1997) 329–342, https://doi.org/10.1109/8.558649.
[96] R.D. Graglia, A.F. Peterson, Higher-order techniques in computational electromagnetics, in: Electromagnetic Waves, Institution of Engineering and Technology, 2015.