Turkish Journal of Electrical Engineering and Computer Sciences

Volume 29 | Number 2 Article 45

1-1-2021

An adaptive element division algorithm for accurate evaluation of singular andnear singular integrals in 3D

HAKAN BAYINDIR

BESİM BARANOĞLU

ALİ YAZICI

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2021) 29: 1187 – 1206

doi:10.3906/elk-1911-57

An adaptive element division algorithm for accurate evaluation of singular and near singular integrals in 3D

Hakan BAYINDIR[1,*], **Besim BARANOĞLU**[2], **Ali YAZICI**[3]
[1]Network Technologies Department, TÜBİTAK ULAKBİM, Ankara, Turkey
[2]Metal Forming Center of Excellence, Atılım University, Ankara, Turkey
[3]Department of Software Engineering, Faculty of Engineering, Atılım University, Ankara, Turkey

Received: 07.11.2019 • **Accepted/Published Online:** 21.08.2020 • **Final Version:** 30.03.2021

Abstract: An adaptive algorithm for evaluation of singular and near singular integrals in 3D is presented. The algorithm is based on successive adaptive/selective subdivisions of the element until a prescribed error criteria is met. For evaluating the integrals in each subdivision, Gauss quadrature is applied. The method is computationally simple, memory efficient and can be applied for both triangular and quadrilateral elements, including the elements with nonplanar and/or curved surfaces. To assess the method, several examples are discussed. It has shown that the algorithm performs well for singular and near-singular integral examples presented in the paper and evaluates the integrals with very high accuracy.

Key words: Near-singular integral, triangular elements, boundary element method, 3D

1. Introduction

Accurate evaluation of field variables on and near the boundary of the solution domain is an important issue for many applications of boundary element method. Near-singular integrals may stem from elements that are very close to each other (e.g., in the case of particle flow problems when the moving particle is very close to the channel wall), or in case of a small element being a neighbor with a considerably large element (e.g., when a fine mesh is attached to a neighboring coarse mesh), or in evaluation of internal quantities near the boundary (e.g., in elastoplastic analysis using dual reciprocity method, in case of computing stress components at collocation points near the boundary).

The general approach to evaluating such singular and near-singular integrals can be classified into four main topics: (i) increasing the quadrature order, (ii) analytical or semianalytical solution of the integral, (iii) transformation of the coordinates to a new set of parameters in which the singularity is canceled or reduced with the aid of the Jacobian of the transformation, and (iv) subdivision of the element to smaller elements and summing up the contributions from each subdivision.

Increasing quadrature order is a simple solution to the problem, yet, computationally inefficient and not practical in coding. Therefore, in literature this alternative is mostly disregarded, unless combined with another algorithm, such as element subdivision [1], and/or adaptive Gaussian quadrature algorithms [2–4]. In Section 3, an example on the effect of increasing the number of Gauss points is presented, which displays why this method is discouraged.

*Correspondence: hakan@bayindir.org

Analytical or semianalytical solution of the singular or near-singular integral is widely used in 2D applications, especially for element geometries that are linear [5, 6], but obtaining an analytical solution to integrals appearing in 3D is not an easy task. Therefore, analytical approach to singularity is limited to simple element geometries and/or simple function forms [7]. Examples would be the exact integrations for the Laplace equation [8] and elasticity equation [9].

In solving singular integrals, series expansion of integral kernel into a prescribed number of terms and obtaining an approximate analytical solution using this expansion (which can be considered as a semianalytical method) is a common procedure for solution in several problems [10, 11]. In some problems, separation of singular and nonsingular parts of the kernel is possible, which gives the possibility to analytically obtain a solution for the singular part and solving the nonsingular part with Gauss quadrature [12–14]. Another method for obtaining approximate analytical solutions is to approximate the function with its limiting value (e.g., $\lim_{x\to 0}(1+x)^q = 1+qx$, [15]) or use a simpler function with identical singularity, as in using Kelvin solutions in place of Helmholz solutions since

$$[\forall\omega]\,(\mathbf{x}\to\mathbf{y}) : \left\{ \begin{array}{l} G_{ij}(\mathbf{x},\mathbf{y},\omega) - G_{ij}(\mathbf{x},\mathbf{y}) = O(1) \\ H_{ij}(\mathbf{x},\mathbf{y},\omega) - H_{ij}(\mathbf{x},\mathbf{y}) = O(1) \end{array} \right\} \tag{1}$$

where $G_{ij}(\mathbf{x},\mathbf{y},\omega)$ and $H_{ij}(\mathbf{x},\mathbf{y},\omega)$ are the Helmholz fundamental solutions and $G_{ij}(\mathbf{x},\mathbf{y})$ and $H_{ij}(\mathbf{x},\mathbf{y})$ are the Kelvin fundamental solutions [16].

In all cases, to obtain an analytical (or semi/approximate analytical) solution for a singular integral, there is a need for simple element geometries, such as isoparametric elements of linear or quadratic order. The need for a simple element geometry and simple forms of functions is also true for the analytical or semianalytical evaluation of near-singular integrals. For example, in [17], near-singular integral is evaluated by transforming the domain integral into line integrals over a linear isoparametric triangle, and solving the resulting line integrals numerically. The same procedure is followed in [18] for semianalytical solution of near-singular integrals appearing in Galerkin surface integrals. As in [17], in [18] linear isoparametric triangular elements are employed. For higher order elements, such as quadratic quadrilateral elements, this is possible with the approximation of the distance function [19].

Parameter transformation is a widely used strategy for evaluating singular or near-singular integrals. Several transformations are proposed in literature, starting from polynomial transformations [20–22] to sigmoidal [23] or sinh [24] transformations, or combinations of these in different directions, like sinh transformation in the radial direction and sigmoidal transformation in the angular direction [25]. Basic idea is to obtain a transformation that has a Jacobian which approaches zero faster than the kernel of the integral approaches maximum value [26]. The major drawback (for near-singular cases) is the need for the calculation of the projected point onto the plane of the element. The position of this point, especially if it lies close to the boundary of the element or is outside the domain of the element, requires special care [27]. Also, the selection of an effective transformation is an important issue and not independent from the function form [28].

Note that, in all above cited studies for evaluating near-singular integrals, location of the near-singular point with respect to the integrated element is of crucial importance. Also, in most cases, the given solution is limited to the presented kernel function or similar functions.

A general approach to evaluate singular or near-singular integrals is to divide the element being integrated into the prescribed number of sub-elements. Note that, there are infinite choices of subdivision strategies that

can be utilized on simplexes. In this study, a subdivision algorithm based on triangulations is proposed. Integrals over nontriangular elements, such as quadrilateral elements, can also be evaluated with this method by subdividing the element into triangles before the process is applied. In the proposed algorithm, triangulations do not alter the geometrical properties of the original element – in fact, all subtriangles are similar triangles to the original element, therefore no regularization in the geometry of sub-triangles are needed. Also, any Gauss quadrature scheme can be employed within the algorithm. The main advantage of the algorithm is its self-adaptivity independently from the location of the source point to the integrated element. Furthermore, a selective subdivision is used, increasing resolution at the regions needed, and stopping the subdivision process at the regions where a prescribed error criterion is reached. The algorithm is not recursive, but stack-based, also it is designed to work depth-first. The depth-first approach prevents large queues since only one triangle's integral is calculated at once. These design decisions lead to two major properties: a very small memory footprint and effortless parallelization. As a result, the algorithm can be run in a parallel manner without the need of complex synchronization structures and works with very small memory consumption costs. The proposed algorithm is flexible in the sense that it can be applied to any type of element including nonplanar geometries, and any kernel function within the integral.

2. Adaptive element subdivision algorithm

The element subdivision with triangulation, in literature, is mainly performed in two ways: either the element is subdivided into three triangles from its center, or the projection of the source point onto the plane of the integrated element is used as a pivot to subdivide the element [28]. In such cases, the newly formed subelements mostly become distorted since one of the angles become too large with respect to the others, which requires a secondary subdivision to overcome this problem and regularize the subelements (see Figure 1).

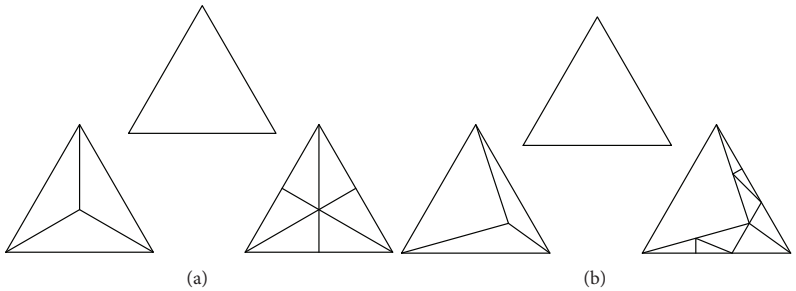

Figure 1. Two commonly applied subdivision technique for triangular shapes, (a) the division based on the center point, (b) division based on the projected point. The left division is the first iteration, the right is the redivision to obtain regular triangles.

It is important to note at this point that, none of the subelements formed after these types of subdivisions are similar to the original element being integrated. Application of an adaptive technique to nonsimilar triangles is not practical since the algorithm needs to take distortions in the triangles into account and make corrective subdivisions if necessary. This adds more processing steps and may cause unnecessary or impractical subdivisions.

An exception to this subdivision technique appears in [29] where the triangular element is divided into similar triangles by folding the triangle – as given in Figure 2. The method can be considered to be adaptive, in the sense that it continues the subdivision process until a given convergence criterion is achieved. The major

disadvantage is that each subdivision process doubles the number of triangles, making 4^k subtriangles for a process of k-fold. Also, the subdivision is not sensitive to the location of singularity or near singularity.

In the proposed algorithm in this study, the element subdivision is made based on side midpoints of the triangle, which results in four similar triangles as demonstrated in Figure 3a. To present the algorithm, we first define a tree structure with elements being triangles that are generated as the result of the subdivision process. The tree originates from the original triangle that integration will be evaluated on. Each element of the tree will be composed of a parent and four children (Figure 3b).

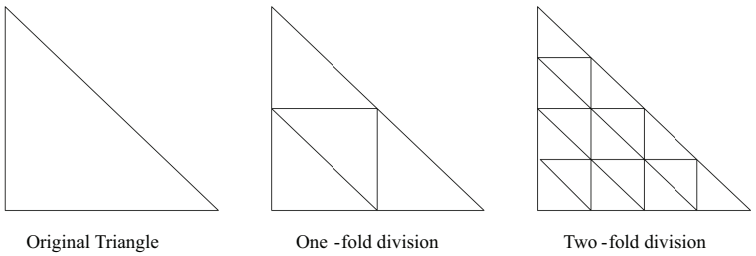

Figure 2. Subdivision by folding as presented in [29] using Romberg integration scheme.

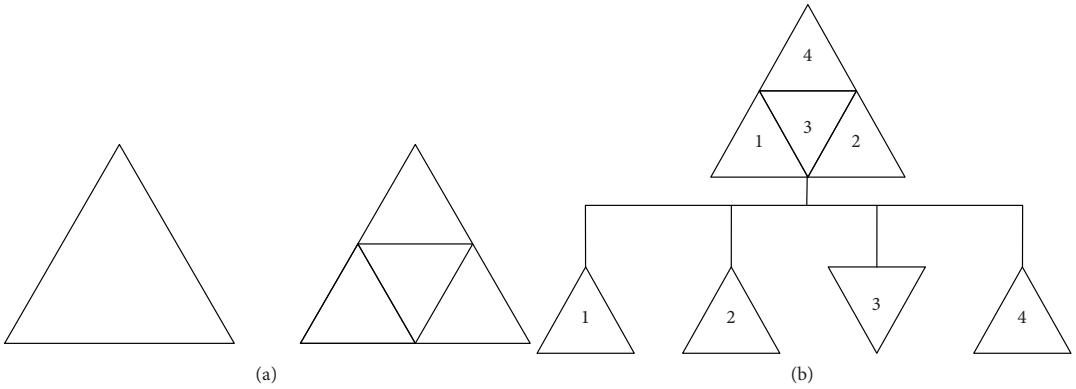

Figure 3. The subdivision of the triangular element (a) proposed subdivision technique applied to a triangular element, and (b) the parent and its four children in tree structure.

The algorithm mainly depends on the comparison of intermediate approximate values of two integrals. These two integrals are computed on the same triangle, but in a different manner. First the so-called parent triangle is considered, then this triangle is subdivided into four which are are integrated independently to add up a composite value of the integral. As a last step, these two values are compared using the stopping criterion (SC) given as

$$SC = \left| \frac{I_{evalulated} - I_{exact}}{I_{exact}} \right| \times 100\% \tag{2}$$

If the SC is equal or lower than the predeterminded value, integration process is terminated. Otherwise, child triangles are appended to the queue for further subdivision, integration and comparison processes. This process of subdividing the triangles continues for every triangle, until the required SC is satisfied for all.

When a subdivision and comparison process yields a result with the desired SC, all five triangles are discarded and result obtained from the sum of divided triangles is kept. This simple notion directs the algorithm to divide sections of the triangles with more singularity into smaller, more fine grained triangles and abandon sufficiently accurate parts earlier. With this behavior, the algorithm automatically concentrates subdivided triangles to the region of the integration domain where near-singularity is mostly affected. This is done inherently within the algorithm without any preprocessing or guesswork and shows a naive but highly accurate adaptive behavior.

The algorithm is implemented with simplicity in mind. For efficiency and resiliency under large loads, the algorithm is designed without recursion. The triangles to be processed are stored in a stack, which has a LIFO (last in, first out) behavior. This stack makes the algorithm to work in depth-first fashion, and as a result the algorithm processes a single triangle until it calculates the required integral. The flowchart of the algorithm is given in Figure 4. Also, complete computation pipeline is consciously designed to prevent subnormal floating point numbers, which create measurable slowdowns during operation of the algorithm. Subnormal floating point numbers are the numbers which fill underflow gap around zero in floating point arithmetic and require special processing by the CPU, hence slowing down computations from one to three orders of magnitude depending on the number's closeness to 0.

This simple and lightweight subdivision algorithm adaptively focuses on hard parts of an integration without any pre or post processing. This adaptive behavior converts Gaussian quadrature from a fixed time, variable accuracy to variable time, almost constant accuracy method. As a result, resulting computation becomes very accurate which, will be detailed in the next section.

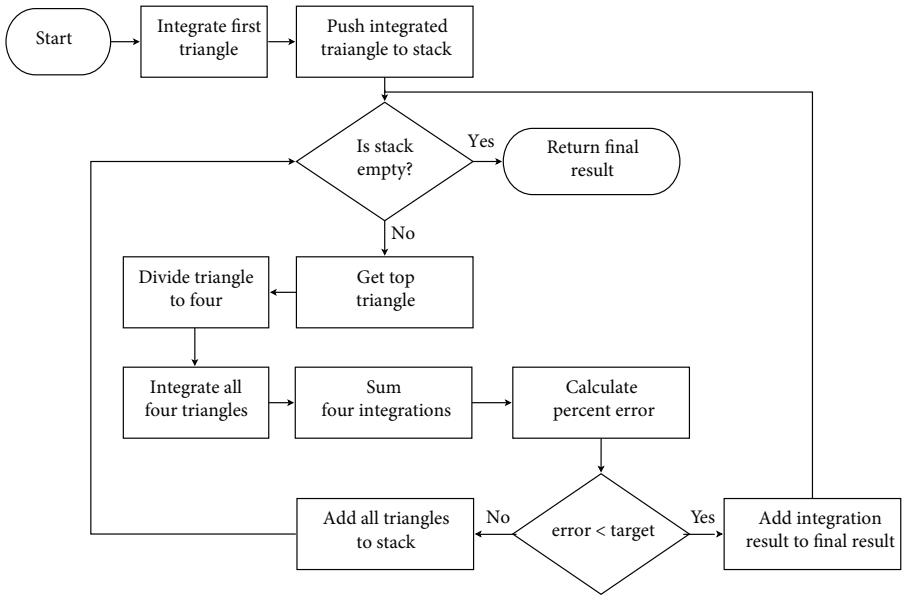

Figure 4. The adaptive subdivision algorithm flowchart.

An example triangulation is presented in Figure 5. Here, the original triangle $\overset{\triangle}{ABC}$ is first divided into four similar triangles $\overset{\triangle}{AEF}$, $\overset{\triangle}{EBD}$, $\overset{\triangle}{EDF}$ and $\overset{\triangle}{FDC}$ for which the integral values are summed up and compared with the original triangle. Since the SC is not met, all four sub-triangles are pushed to stack, the upper most

triangle $\overset{\triangle}{FDC}$ is taken from the stack as the parent triangle. This triangle is subdivided into four children and the integral value is compared with the parent. Since the comparison yielded a SC that is below the prescribed bound, the integral value of the $\overset{\triangle}{FDC}$ is updated as the value of the sum of the integrals of the children, and the triangles are popped out. The next integral on top of the stack is $\overset{\triangle}{EBD}$, which is divided into four children. Since the comparison did not yield to a SC that is lower than the bound, all children are also divided into four and compared with their respective parents. When the prescribed value is obtained between a parent and its four children, the corresponding triangles are popped out of the stack. The process goes on with the other triangles until there is no triangles left on the stack.

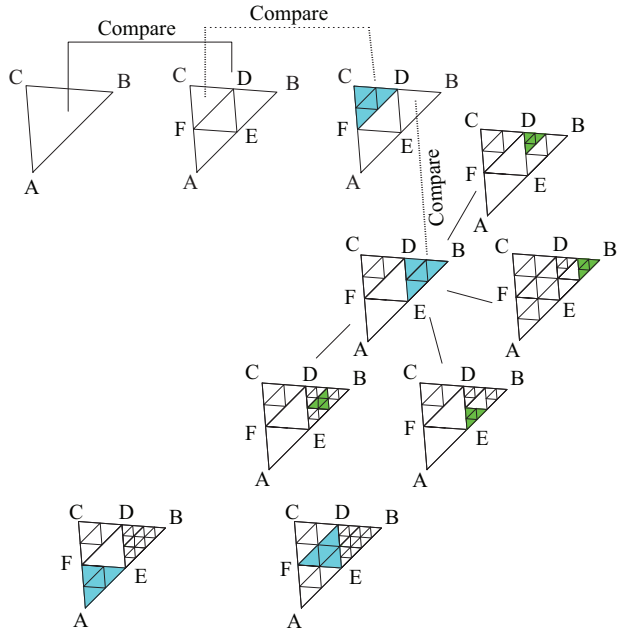

Figure 5. The subdivision of the integration region into similar triangles. Each subdivision level is represented with a different color. The self-localization behavior of the algorithm can be traced in the triangle region BED.

Since the algorithm uses Gaussian quadrature during evaluation of integrals in every step, utilization of different number of Gaussian quadrature points have an effect on speed of evaluation. Since a higher order Gaussian quadrature is inherently more accurate in approximation of the integral value, using a higher order Gaussian quadrature results in more accurate evaluations with fewer number of triangles. This effect can be clearly observed in the results presented in the Section 3.

Another advantage of the proposed algorithm is its independence of the integral kernel. This independence means the algorithm does not need to be changed or tuned with respect to the problem being evaluated. As a result, the algorithm can be integrated into a general purpose application and be used for evaluation of any singular or near-singular integral. This feature separates the proposed algorithm from other methods and algorithms which needs to be tuned or partially reimplemented in order to evaluate singular and near-singular integrals efficiently and accurately.

Last but not the least, the proposed algorithm's design is targeted towards real world high performance computing applications. This targeting enables it to be able to exploit all the computing resources provided by

modern multicore and multiprocessor systems. During execution, the algorithm isolates all its work data and does not need any information besides the target triangle, source point and a SC at start and during execution. Due to these properties, multiple copies of the algorithm can be run in parallel, evaluating multiple triangles at once, hence accelerating the evaluation process in real world problems greatly.

One possible caveat of the algorithm is its tendency to overshoot prescribed error. The reason for this phenomenon is the calculation of the SC on the last five (one parent and four siblings) triangles. When the algorithm tries to satisfy SC on smaller triangles that yield smaller integration values, this translates to an even lower SC value on the whole integration process. This issue may be averted in a variety of ways, but most of these methods will complicate the algorithm, hence result in a more computationally costly algorithm with lower accuracy. To keep the algorithm simple and fast, the authors did not pursue a solution to eliminating this overshooting phenomenon.

3. Case studies

To display the efficiency of the proposed algorithm, three benchmarks are presented in this section.

3.1. Solution of a simple singular kernel over linear parametric triangular element

As a first benchmark, the first example from [17] is studied. Here, the integral

$$I = \int_T \frac{1}{r^n} dA \tag{3}$$

is evaluated and compared with the exact values. In Equation 3, n represents the order of singularity and r is the Euclidean distance between the source (fixed) point and the varied (integration) point on the integrated element.

The triangle that the integral is evaluated over is shown in Figure 6 along with the position of the source point. It is obvious that, as the value of z approaches to zero, the integral becomes highly near singular.

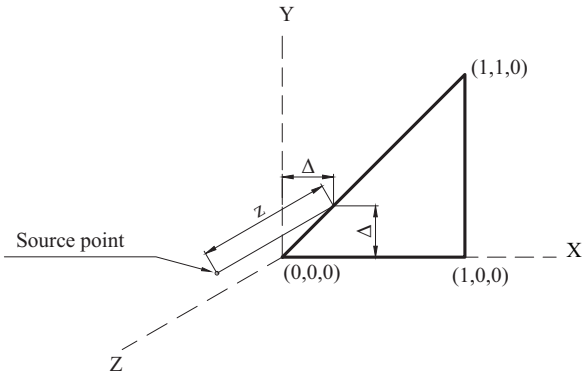

Figure 6. The location of the source point and the integrated triangular element for the benchmark problem. The triangle is a right-angled triangle at xy-plane and the source point is located at (Δ, Δ, z).

To display inefficiency of increasing only the Gauss quadrature order (in a conventional Gaussian integration algorithm, without element subdivision), we first present Table 1. It can be easily seen that, as the point gets closer to the integrated element, e.g., as $z \to 0$, increasing the number of Gauss points, to even 4 million

points, is not effective enough (for $z = 0.001$, $\%error = 1.85$). Note at this point that, for a single integrated element, the storage requirement for 4 million points (including the corresponding weights) is approximately 90 MB (in contrast, solution of a 1000 constant elements elastostatic problem, including the evaluation of internal points consumes 139.27 MB at most when solved using the framework presented in [30] utilizing the shown method in this study). This makes Gaussian quadrature inefficient from both computational and resulting percent error aspects.

Table 1. Effect of increasing the number of Gauss points in a conventional algorithm ($n = 5$ and $\Delta = 0.6$).

z	Exact solution	# of Gauss points	% error
0.1	1.03964997638965e + 03	10^2	1.369e + 01
		250^2	4.570e − 12
		500^2	4.570e − 12
		1000^2	4.570e − 12
		2000^2	2.380e − 11
0.01	1.04718947671873e + 06	10^2	9.766e + 01
		250^2	9.000e − 02
		500^2	1.830e − 04
		1000^2	1.780e − 10
		2000^2	7.710e − 11
0.001	1.04719754311651e + 09	10^2	1.000e + 02
		250^2	8.932e + 01
		500^2	4.754e + 01
		1000^2	3.830e + 00
		2000^2	1.850e + 00

In the present study, 3-, 6- and 7-point Gauss quadratures are used in evaluating the integrals in the parent and divided elements (all Gauss quadrature tables are presented in the Appendix with respective references). For comparison, the study of Niu et al. is taken into consideration [17].

Firstly, 7-point Gauss quadrature results are presented along with the data given in the implied reference [17]. Values labeled as *conventional* are obtained by evaluating the same integral with 8-point Gaussian quadrature (also present in [17]). Note that, in [17] the same definition for the percent error given in Equation 2 is used. In all presented results, unless otherwise is explicitly stated, the SC is prescribed as $SC = 10^{-4}$.

Table 2 presents the comparison of the percent error. It can be seen that the results of the current study present almost-constant accuracy independently from the location of the source point whereas the results of [17] highly depend on the location of the near-singular point. It is also worth noting that even with a relaxed stopping criteria of $SC = 10^{-4}$, the obtained results are comparable with [17].

To assess the memory footprint of the developed algorithm, a stripped-down C++ implementation[i] is memory profiled using Valgrind Massif. Massif is a heap and stack profiler which, in practice, measures the memory usage of an application by encapsulating it and monitoring its memory allocation behavior. The profiling is done for every test case in Table 2 for stopping criteria $SC = 10^{-4}$ and $SC = 10^{-8}$ using 7-point

[i]A special application containing only the integration algorithms is developed during the preparation of this paper to test and verify the case studies. The algorithms are identical to [30].

Gaussian quadrature. The relevant results can be seen in Table 3[ii]. Since there are no implementation details provided in [17], it is not possible to make any accurate predictions on the memory usage of the algorithm developed by the researchers because of the vast number of possible ways to design and implement such an algorithm. As can be seen in Table 3, the complete program's memory footprint is extremely small and can be considered independent of integration difficulty. This property allows this algorithm to be implemented without any noticeable resource impact even in massively multicore scenarios.

Table 2. Comparison of the 7-point Gauss quadrature results from the current study with the results presented in [17]. The singularity order is $n = 5$ and the stopping criteria is 1e-4.

Point			Percent error		
z	Δ	Exact Sol'n	Conventional	Niu	7 pt. adapt.
0.001	0.01	1.046e + 09	2.60e + 01	4.61e − 12	1.032e − 06
	0.1	1.047e + 09	7.54e + 01	3.42e − 14	7.772e − 08
	0.6	1.047e + 09	9.93e + 01	0.00e + 00	7.772e − 08
0.01	0.01	8.737e + 05	1.62e + 01	4.80e − 05	7.772e − 08
	0.1	1.046e + 06	1.06e + 01	4.76e − 12	1.376e − 08
	0.6	1.047e + 06	2.64e + 01	5.56e − 14	3.738e − 09
0.1	0.01	3.327e + 02	2.54e + 00	2.93e − 01	3.846e − 09
	0.1	8.734e + 02	9.12e − 01	2.07e − 05	3.154e − 08
	0.6	1.039e + 03	7.58e − 02	3.69e − 11	4.357e − 08

Numbers are truncated. Computation precision is 14 significant digits.

Table 3. Comparison of the memory usage of the algorithm in different cases in both $SC = 10^{-4}$ and $SC = 10^{-8}$ using 7-point Gauss quadrature. The singularity order is $n = 5$.

Point			Memory usage (bytes)	
z	Δ	Exact Sol'n	$SC \rightarrow \mathrm{e} = 10^{-4}$	$SC \rightarrow \mathrm{e} = 10^{-8}$
0.001	0.01	1.046e + 09	92, 384	95, 960
	0.1	1.047e + 09	92, 384	94, 456
	0.6	1.047e + 09	91, 448	94, 456
0.01	0.01	8.737e + 05	90, 288	94, 456
	0.1	1.046e + 06	91, 448	94, 456
	0.6	1.047e + 06	90, 288	94, 456
0.1	0.01	3.327e + 02	88, 360	92, 384
	0.1	8.734e + 02	88, 360	92, 384
	0.6	1.039e + 03	90, 288	93, 320

Numbers are truncated. Computation precision is 14 significant digits.

Further analysis can be made considering the number of Gauss quadrature points on each triangle in the evaluation. The simplest is the 1-point GQ, where the evaluation is based on the center point of the triangle. This case is not covered in this study since for small values of z the computation time becomes extremely long. In this study, we present the results obtained with 3- and 7-point GQ. The points are symmetrically distributed

[ii]The profiling is done over 100 runs and reported maximum value is used for tabulation.

on the element as proposed by [31]. The results of the comparisons are given in Tables 4 and 5 where the error, number of triangles used in evaluating the integral and average solution time of 5 runs[iii] is tabulated respectively. It can be seen that 7-point GQ gives comparable results with those of 3-point GQ in accuracy with much less number of triangles. Furthermore, it can be seen that all evaluations give an almost constant accuracy regardless of the position of the source point, and this accuracy can be preset by the application developer or user in accordance with the requirements of the problem solved. Also, a set of benchmarks are done to observe the effect of using different number of GQ points in the integration process. In the benchmarks, exactly the same integration is evaluated on the triangle shown in Figure 6, and number of GQ points have changed. The solution times of these benchmarks is tabulated in Table 6[iv]. Tables for GQ points can be found in Appendix A.

The effect of utilizing a different number of GQ points in the evaluation of integrals can be observed in Tables 4–6. In every case, an increase in the number of GQ points reduces total computation time and the number of triangles evaluated to reach the prescribed error bound. In these benchmarks, stopping criteria is selected as $SC = 10^{-8}$ to push the algorithm to its limits and highlight the effect of increasing GQ point count clearly. This reduction of time and number of evaluations are obtained with a negligible increase in computation resource requirement. If desired, the GQ points can be increased further. Increasing points will accelerate computation up to a certain extent without degrading the result accuracy. Since the algorithm will evaluate at least 5 integrals to compare the results between the parent and divided triangles, accelerating the process will not be possible after a certain GQ point count. In any case, the result accuracy will not degrade.

Table 4. Comparison of GQ order concerning percent error and number of triangles used in evaluating the integral ($n = 5$).

Point		3-point GQ		7-point GQ	
z	Δ	% Error	Triangle count	% Error	Triangle Count
0.001	0.01	1.695e − 11	159309125	6.946e − 13	157333
	0.1	2.663e − 12	229474789	1.216e − 11	225909
	0.6	6.875e − 12	260216069	1.289e − 11	255957
	0.01	1.798e − 12	71031493	5.729e − 13	69925
0.01	0.1	2.068e − 12	128665733	2.202e − 12	86641
	0.6	8.448e − 13	159180101	2.089e − 12	153477
	0.01	7.875e − 12	26970245	6.884e − 12	26213
0.1	0.1	2.642e − 12	36464661	2.394e − 12	37285
	0.6	8.485e − 12	55231125	4.570e − 12	54805

[1]Numbers are truncated. Computation precision is 14 significant digits.
[2]Stopping criteria is chosen as $e = 10^{-8}$.

For completeness, one last analysis can be presented for evaluating the performance of the proposed algorithm with different orders of singularity. For this, n is changed from 2 to 5. In the presented evaluations, 7-point GQ is used in triangular integration. It can be seen in Table 7 that for all singularity orders, n, the proposed method gives a good approximation with the exact value. For comparison, results obtained from 46-point (order 14 [32]) GQ evaluation are included (Appendix A).

[iii]The time measurements are done on a Dell PowerEdge R720 server with two 2.0 GHz Intel Xeon 2650 eight-core processors and 392 GB of RAM.
[iv]Benchmarks in Tables 6 and 7 are done with a Mac Book Pro (Mid 2014 model with 2.8GHz Intel Core i7 CPU and 16GB RAM). Timing is an average of 100 runs.

Table 5. Comparison of GQ order concerning average time to obtain solution ($n = 5$).

Point		Solution time (ms)	
z	Δ	3-point	7-point
0.001	0.01	51732	110
	0.1	74380	159
	0.6	83891	179
0.01	0.01	22836	48
	0.1	41324	86
	0.6	50870	105
0.1	0.01	8654	18
	0.1	11651	29
	0.6	17473	38

Stopping criteria is chosen as $e = 10^{-8}$.

Table 6. Comparison of GQ order concerning average time to obtain solution ($n = 5,\ \Delta = 0.1$).

	Solution time (ms)		
z	3-point	5-point	7-point
5	0.508	0.215	0.015
1	30.468	3.018	0.184
0.5	141.797	7.989	0.410
0.1	795.135	43.747	2.424
0.01	3015.818	141.509	7.422
0.001	5273.177	260.369	13.778

Table 7. Comparison for different singularity orders ($\Delta = 0.1$).

z	Order (n)	Exact solution	# of triangles	Solution time (μs)	Error	46-point GQ error
0.1	2	3.410e + 00	761	0.277	3.860e − 11	3.500e − 01
0.01	2	1.012e + 01	2509	0.928	4.790e − 10	1.632e + 01
0.001	2	1.734e + 01	4449	1.694	3.305e − 10	4.904e + 01
0.1	3	1.748e + 01	1065	1.115	1.226e − 08	6.700e − 01
0.01	3	2.963e + 02	3621	3.947	3.804e − 10	5.086e + 01
0.001	3	3.124e + 03	6417	7.240	3.624e − 10	9.484e + 01
0.1	4	1.161e + 02	1477	1.764	1.760e − 10	5.700e − 01
0.01	4	1.564e + 04	4945	5.535	1.360e − 10	7.767e + 01
0.001	4	1.570e + 06	8813	10.066	6.540e − 10	9.974e + 01
0.1	5	8.734e + 02	37285	2.424	2.394e − 12	9.092e + 01
0.01	5	1.046e + 06	86641	7.422	2.202e − 12	1.000e + 02
0.001	5	1.047e + 09	225909	13.778	1.216e − 11	9.999e + 01

3.2. Solution of a simple singular kernel over curved elements

A second benchmark is aimed to assess the performance of the proposed algorithm over nonplanar and/or curved elements. For this, a 3D six-noded quadratic isoparametric element is considered. To be consistent with the previous benchmark, the three vertex nodes of the quadratic triangular element and the definition of the fixed point are kept as the same in Figure 6. The coordinates of the nodes in order are given in Table 8. The change of geometry is displayed in Figure 7 for three different values of M.

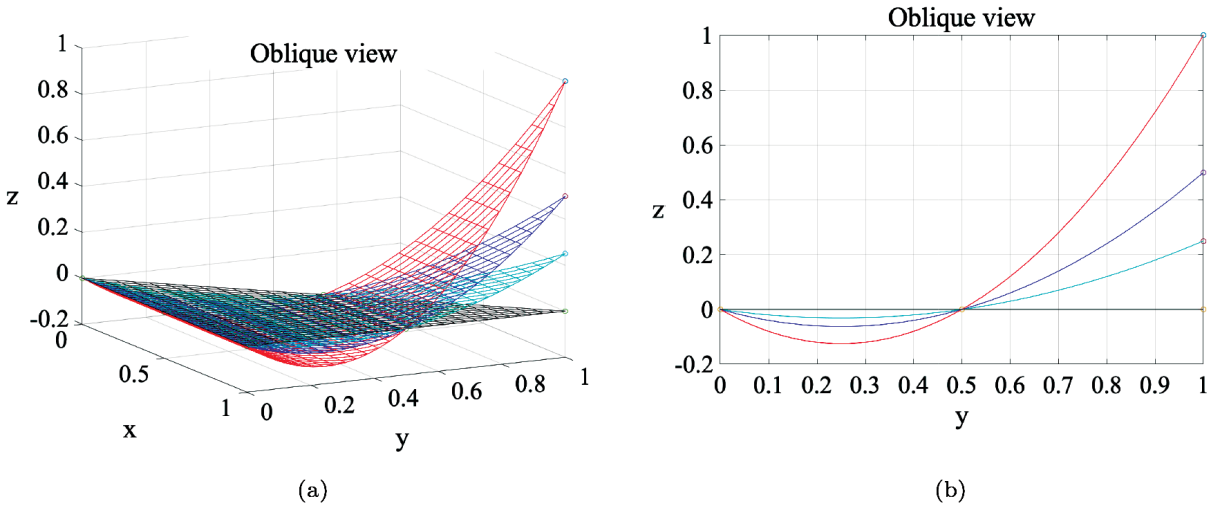

Figure 7. The curvature of the isoparametric quadratic six-noded element. The tip z-coordinate values are: $M = 1$ (red), $M = 0.5$ (blue), $M = 0.25$ (green) and $M = 0$ (black). Note that for $M = 1$ the element is planar and identical to three-noded linear isoparametric element.

Table 8. Coordinates of the nodes for quadratic element in benchmark 2. The curvature can be controlled by changing the parameter M in z-coordinate of the third node. For $M = 0$, the surface is identical to that given in Figure 6.

Node #	x-coordinaate	y-coordinate	z-coordinate
1	0	0	0
2	1	0	0
3	0	1	M
4	1/2	0	0
5	1/2	1/2	0
6	0	1/2	0

The integral is given as in Equation 3. A semianalytical solution to this integral is possible. For this, first the quadratic transformation is performed as:

$$I = \int_T f(x_i) dA = \int_0^{+1} \int_0^{1-\xi} f(\xi, \eta) J(\xi, \eta) d\xi d\eta \tag{4}$$

where

$$x_i = N_j x_i^j \tag{5}$$

with x_i^j are the coordinates of the i^{th} node and N_j are the corresponding base functions. In Equation 4, $J(\xi,\eta)$ is the Jacobian which is given by

$$J(\xi,\eta) = \left\| \frac{\partial x_i}{\partial \xi} \times \frac{\partial x_i}{\partial \eta} \right\| \tag{6}$$

Later, a second quadratic transformation is proposed (as in [31]) with

$$\xi = \frac{1+u}{2} \quad \text{and} \quad \eta = \frac{(1-u)(1+v)}{4} \tag{7}$$

which transforms the integral to

$$I = \int_T f(x_i)dA = \int_0^{+1}\int_0^{1-\xi} f(\xi,\eta)J(\xi,\eta)d\xi d\eta = \int_{-1}^{+1}\int_{-1}^{+1} f(u,v)J(u,v)(1-u)dudv \tag{8}$$

which in the final form can be solved by bisection to obtain a very accurate solution.

A first analysis considers the change of the singular point for a constant curvature of the element setting $M = 1$. Table 9 presents the obtained results. For different closely located fixed points, the outcome of the proposed method is compared with the semianalytical results along with the comparison of 100-point GQ. It can be seen that the results present an almost constant accuracy.

Table 9. Comparison of the proposed method with the semianalytical solution for $M = 1$.

z	Δ	Semianalytical	% Error present	% Error 100-pt GQ
0.001	0.01	7.316e + 00	6.679e − 08	1.182e + 01
	0.1	5.550e + 00	7.016e − 07	1.898e + 00
	0.6	8.246e + 00	2.057e − 06	1.578e + 01
	0.01	5.048e + 00	1.465e − 06	2.450e + 00
0.01	0.1	5.173e + 00	3.355e − 07	1.148e + 00
	0.6	8.559e + 00	6.340e − 07	1.825e + 01
	0.01	2.120e + 00	3.997e − 06	1.442e − 02
0.1	0.1	2.941e + 00	2.031e − 07	2.803e − 03
	0.6	1.462e + 01	2.067e − 06	2.448e + 01

[1]Numbers are truncated. Computation precision is 14 significant digits.
[2]Stopping criteria is chosen as $e = 10^{-8}$.

As a second assessment to the proposed method, the z-coordinate of the third node (i.e. M) is varied to change the curvature of the element. The near-singular point is selected to be (0.6, 0.6, 0.001). It should be noted that this point is close to the sixth node of the element.

It is expected to obtain the exact result evaluated for the 3-point linear isoparametric element when $M = 0$. In other cases, the semianalytical results can be evaluated. These values are compared with the values obtained from the present study and also with 100-point GQ (Table 10). The results show that the accuracy of the method is satisfactory, especially when compared with the very bad approximations by the 100-point GQ.

Table 10. Comparison of the proposed method with the semianalytical solution for fixed point location $\Delta = 0.6$ and $z = 0.001$.

M	Semianalytical	% Error present	% Error 100-pt GQ
0	1.047e + 09	6.85e − 12	9.999e + 01
0.10	9.087e + 05	3.685e − 07	9.306e + 01
0.25	6.487e + 04	1.604e − 06	2.855e + 01
0.50	1.289e + 04	3.504e − 07	7.698e + 01
0.75	6.314e + 03	2.475e − 07	1.124e + 02
1.00	4.298e + 03	9.492e − 07	1.210e + 02
2.00	2.491e + 03	4.501e − 06	5.020e + 01

[1]Numbers are truncated. Computation precision is 14 significant digits.
[2]Stopping criteria is chosen as $e = 10^{-8}$.

3.3. Solution of elastostatic fundamental solutions over constant triangular element with linear isoparametric geometry

As a third benchmark, elastostatic first and second fundamental solutions which are given by [33]:

$$G_{ij}(B,P) = \frac{1}{16\pi\mu(1-\nu)r}\left[(3-4\nu)\delta_{ij} - r_i r_j\right] \tag{9}$$

and

$$H_{ij} = -\frac{1}{8\pi(1-\nu)r^2}\left\{\begin{array}{c} \dfrac{\partial r}{\partial n}\left((1-2\nu)\delta_{ij} + 3r_i r_j\right) \\ +(1-2\nu)(n_i r_j - n_j r_i) \end{array}\right\} \tag{10}$$

are evaluated. The exact solution when the source point is on the integrated triangular element is given by [34] and later by [35]. In this study, the integrated element is selected as the same in Figure 6, with the change that the source point is now located at the center point of the triangle, e.g., at $(\frac{2}{3}, \frac{1}{3}, 0)$. A nondimensional analysis is performed replacing r with $\bar{r} = \frac{r}{a}$ where a is the characteristic length and $\bar{\sigma} = \frac{\sigma}{\mu}$ where μ is the shear modulus of chosen material. The analytical solutions are obtained through the formulation presented by [35] and presented in Table 11 with nondimensional material properties selected as shear modulus, $\bar{\mu} = 1.0$ and Poisson's ratio, $\nu = 0.2$. The error, in this matrix case, is defined as the maximum of the Frobenius norm of the difference between the evaluated and the exact matrices:

$$e = \max\left(\left\{\left[G_{ij}^{ex} - G_{ij}^{ev}\right]\left[G_{ij}^{ex} - G_{ij}^{ev}\right]\right\}^{1/2}, \left\{\left[H_{ij}^{ex} - H_{ij}^{ev}\right]\left[H_{ij}^{ex} - H_{ij}^{ev}\right]\right\}^{1/2}\right). \tag{11}$$

Table 11. Exact values of the G matrix for the given benchmark triangle.

G matrix		
0.161629822	0.003846090	0
0.003846090	0.161629822	0
0	0	0.131698374

H matrix		
0.5	0	0.007360794
0	0.5	–0.007360794
–0.007360794	0.007360794	0.5

Due to geometry, the G matrix is symmetric and H matrix is antisymmetric (in the sense that off-diagonal elements are negative of each other side, yet the diagonal should be $\frac{1}{2}$, see Table 11). Therefore, only nonzero independent components are compared, which are G_{11}, G_{12}, G_{22}, G_{33} and H_{11}, H_{13} and H_{23}. Firstly, the error in the G matrix is presented in Table 12 compared with the prescribed SC. It can be seen that even with a very relaxed SC of 10^{-3} the procedure gives very accurate results. Secondly, the error in the H matrix is presented in Table 13 compared with the prescribed SC. Similarly, very accurate results are obtained. Note that, the number of triangles used in the evaluation is $951,700$ for $SC = 10^{-3}$, $3,875,716$ for $SC = 10^{-4}$ and $28,317,812$ for $SC = 10^{-5}$ giving the solution times of approximately $0.37s$, $1.42s$ and $10.86s$ respectively.

Table 12. Comparison of % error in independent components of G for different prescribed error bounds.

$SC \rightarrow$	10^{-3}	10^{-4}	10^{-5}
G_{11}	2.909e − 04	8.093e − 05	1.305e − 05
G_{12}	3.454e − 04	1.218e − 04	2.301e − 05
G_{22}	2.950e − 04	8.074e − 05	1.287e − 05
G_{33}	2.927e − 04	8.056e − 05	1.276e − 05
max	3.454e − 04	1.218e − 04	2.301e − 05

Table 13. Comparison of % error in independent components of H for different prescribed error bounds.

$SC \rightarrow$	10^{-3}	10^{-4}	10^{-5}
H_{11}	2.460e − 05	8.800e − 07	5.600e − 07
H_{13}	1.795e − 03	2.367e − 05	7.325e − 05
H_{23}	1.202e − 03	4.342e − 04	4.707e − 05
max	1.795e − 03	4.342e − 04	7.325e − 05

The last benchmark, using the same triangle geometry is presented below. In this scenario, the source point is moved out of the triangle's plane, in the direction of its outer unit direction. The new location of the source point is changed through a positive scalar value, s (see Figure 8). The near singularity increases as $s \rightarrow 0$ thus, the components of the G and H matrices shall approach to the singular exact values presented in Table 11. G_{11} and H_{11} are presented with respect to changing s in the Figure 9. Validating the expectations, values approach to the exact singular values as the value of s decreases.

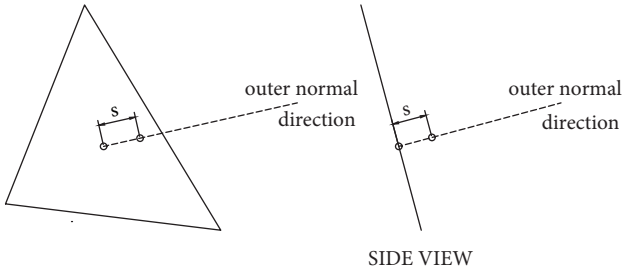

Figure 8. The change of position of the singular point on a triangular element.

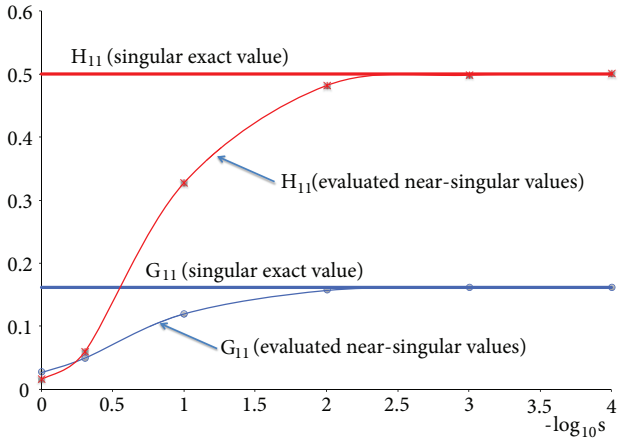

Figure 9. The change of position of the singular point on a triangular element.

4. Conclusion and discussion

A new adaptive method based on element subdivision for obtaining three-dimensional singular and near singular integrals is presented. The method proves well for the examples studied. The advantages of the presented method lie in the following discussion:

- The algorithm continues to operate until given SC is achieved. This ensures that the accuracy of the algorithm is guaranteed.
- Since the subdivision of the triangles continues until given SC is achieved, the subdivision process concentrates on the location of the point of singularity (or near-singularity), making more divisions around that point.
- The proposed method operates without approximating the distance of the source point to the element or without projecting said point to the element hence, it operates regardless of the source point with respect to the element being evaluated.
- Since all triangles obtained from the original triangles are similar triangles, it is possible to compute the coordinates of the Gaussian points on the new triangles without complex transformations. This feature reduces the computational complexity and cost.
- Using the similar triangle property pointed above, it is easier to calculate the areas of the divided triangles with a simple division by 4. This, again, reduces computational complexity and cost.
- Simplicity of the algorithm allows efficient implementation in terms of both space and time. This claim is materialized [30] where an elastostatic problem containing 1000 constant elements has been solved including internal point evaluation and, the whole application has consumed 139.27 MB at most during the solution process. Furthermore, memory profiling of the algorithm alone has shown that the memory footprint of the algorithm is extremely small (less than 100 Kib) and can be scaled to many cores without any significant resource requirement.

It is also important to note here that, the presented algorithm can be used along with any parallelization algorithm for evaluating the system matrices since it presents an independent element-based integration. Also,

a considerable speed-up can be obtained when the integrations of the child triangles are evaluated on different compute nodes.

One major advantage of the algorithm is the applicability of the method to three-dimensional nonplanar and/or curved elements. In this study, as an example, the isoparametric six-noded quadratic triangular element is studied and the obtained accuracy is found highly satisfactory. The procedure can easily be extended to other nonplanar and/or curved elements, e.g., isogeometric elements. Also, the method presents applicability to any fundamental solution. These two properties make it possible to be used as a common integration algorithm for package programs that are designed to analyze different problems appearing in different mathematical models.

A. Gaussian quadrature points

In this section, the Gaussian quadrature points and weights referenced in the paper and used in the algorithm are presented. To save space, some tables are represented in a compact form. To obtain the full set of points, one shall write all permutations of all *alpha*, *beta*, and *gamma* triplets, then eliminate the identical sets from the result. This process yields one point if *alpha*, *beta*, and *gamma* have the same values, three points if two values of the triplet are identical, or six points if all values of the triplet are different. The same weight shall be used for all points generated from a single line in the table. If the table is generated right, the sum of all weights after the elimination of identical triplets shall be equal to 1.

The order 14/46-point Gaussian quadrature is reproduced from [32] and displayed in Table 17. Table should be read as follows: The orbit S_3 has a single abscissa, $a = \frac{1}{3}$, and represents a single point (a, a, a) in local coordinates. The orbits S_{21} each have single abscissas, namely a and each corresponds to three points with local coordinates being combinations of $(a, a, 1-2a)$. The orbits S_{111} each have double abscissas, namely (a, b) and these correspond to six points with local coordinates being the combinations of $(a, b, 1-a-b)$. Other useful Gaussian quadratures consisting 3-, 6- and 7-points are also provided in Tables 14–16, respectively for further reference.

Table 14. Three-point Gaussian quadrature points and weights used in the algorithm.

Weight	*Alpha*	*Beta*	*Gamma*
0.333333333333333	0.666666666666667	0.166666666666667	0.166666666666667

Table 15. Six-point Gaussian quadrature points and weights used in the algorithm.

Weight	*Alpha*	*Beta*	*Gamma*
0.223381589678011	0.108103018168070	0.445948490915965	0.445948490915965
0.109951743655322	0.815847572980459	0.091576213509771	0.091576213509771

Table 16. Seven-point Gaussian quadrature points and weights used in the algorithm.

Weight	*Alpha*	*Beta*	*Gamma*
0.225000000000000	0.333333333333333	0.333333333333333	0.333333333333333
0.132394152788510	0.059715871789780	0.470142064105110	0.470142064105110
0.125939180544830	0.797426985353090	0.101286507323460	0.101286507323460

Table 17. Forty-six-point Gauss quadrature points and weights used for comparisons in Table 7.

Orbit	*Abcissas*	*Weight*
S_3	0.33333333333333333333333333333333	0.058596285226028594127893806347756 0
S_{21}	0.00997976080645843241529352958205 24	0.00173515122972526756806186388080 94
S_{21}	0.47997789352118838981055286508838 99	0.02616378255861452177782885918197 83
S_{21}	0.15381195917696690000000000000000 00	0.00391972924240182909652082757014 54
S_{21}	0.07402347711698781000000000000000 00	0.01224735975694086609728698992625 05
S_{21}	0.13035468250333000000000000000000 00	0.02819962850325796010736630715156 57
S_{21}	0.23061722602665313429960537009838 31	0.05088708718595948529603482754545 40
S_{21}	0.42233208341914782411440871379139 39	0.05045343990160359919102089713411 89
S_{111}	0.78623738593466100332962211403309 00 0.19061636003190090424614328286530 34	0.01706364421223345129002539938494 72
S_{111}	0.63055214366060744162240907556881 29 0.36232313774354714461832673435977 29	0.00968346642550660040752096309341 94
S_{111}	0.62657732985630631423351231375342 65 0.29077120588366741502481681748167 32	0.03638575592848500562201132776427 17
S_{111}	0.91420998492962541223996709938504 69 0.07116571087775076254759245029243 36	0.00696466337351841242539972250424 13

References

[1] Gao X, Davies T. Adaptive integration in elasto-plastic boundary element analysis. Journal of the Chinese Institute of Engineers 2000; 23 (3): 349-356. doi: 10.1080/02533839.2000.9670555

[2] Pinheiro JC, Chao EC. Efficient Laplacian and adaptive Gaussian quadrature algorithms for multilevel generalized linear mixed models. Journal of Computational and Graphical Statistics 2006; 15 (1): 58-81. doi 10.1198/106186006X96962

[3] Patterson MA, Rao AV. GPOPS-II: a MATLAB software for solving multiple-phase optimal control problems using hp-adaptive Gaussian quadrature collocation methods and sparse nonlinear programming. ACM Transactions on Mathematical Software (TOMS) 2014; 41 (1): 1-37. doi: 10.1145/2558904

[4] Rizopoulos D. Fast fitting of joint models for longitudinal and event time data using a pseudo-adaptive Gaussian quadrature rule. Computational Statistics & Data Analysis 2012; 56 (3): 491-501. doi: 10.1016/j.csda.2011.09.007

[5] Wang j, Tsay TK. Analytical evaluation and application of the singularities in boundary element method. Engineering Analysis with Boundary Elements 2005; 29 (3): 241-256. doi: 10.1016/j.enganabound.2004.12.008

[6] Singh KM, Tanaka M. Analytical integration of weakly singular integrals in boundary element analysis of Helmholtz and advection–diffusion equations. Computer Methods in Applied Mechanics and Engineering 2000; 189 (2): 625-640. doi: 10.1016/S0045-7825(99)00316-3

[7] Zhou H, Niu Z, Cheng C, Guan Z. Analytical integral algorithm applied to boundary layer effect and thin body effect in BEM for anisotropic potential problems. Computers & Structures 2008; 86 (15-16): 1656-1671. doi: 10.1016/j.compstruc.2007.10.002

[8] JabŁoński P. Integral and geometrical means in the analytical evaluation of the BEM integrals for a 3D Laplace equation. Engineering Analysis with Boundary Elements 2010; 34 (1): 264-273. doi: 10.1016/j.enganabound.2009.10.002

[9] Fata SN. Explicit expressions for three-dimensional boundary integrals in linear elasticity. Journal of Computational and Applied Mathematics 2011; 235 (15): 4480-4495. doi: 10.1016/j.cam.2011.04.017

[10] Mahmoudi Y. Taylor expansion method for integrals with algebraic-logarithmic singularities. International Journal of Computer Mathematics 2011; 88 (12): 2618-2624. doi: 10.1080/00207160.2011.553673

[11] Ren Y, Zhang B, Qiao H. A simple Taylor-Series expansion method for a class of second kind integral equations. Journal of Computational and Applied Mathematics 1999; 110 (1): 15-24. doi: S0377-0427(99)00192-2

[12] Pearson LW. A separation of the logarithmic singularity in the exact kernel of the cylindrical antenna integral equation. IEEE Transactions on antennas and propagation 1975; 23 (2): 256-258. doi: 10.1109/TAP.1975.1141048

[13] Liviu Marin. Treatment of singularities in the method of fundamental solutions for two-dimensional Helmholz-type equations. Applied Mathematical Modeling 2010; 34 (6): 1615-1633. doi: 10.1016/j.apm.2009.09.009

[14] Polishchuk OD. On the separation of singularities in the numerical solution of integral equations of the potential theory. Journal of Mathematical Sciences 2016; 212 (1): 27-37. doi: 10.1007/s10958-015-2646-4

[15] Xie G, Zhong Y. Zhou F, Du W, Li H et al. Singularity cancellation method for time-domain boundary element formulation of elastodynamics: A Direct Approach. Applied Mathematical Modelling 2020; 80: 647-667. doi: 10.1016/j.apm.2019.11.053

[16] Dangla P, Semblat JF, Xiao H, Deléphine N. A simple and efficient regularization method for 3D BEM: application to frequency-domain elastodynamics. Bulletin of the Seismological Society of America 2005; 95 (5): 1916-1927 doi: 10.1785/0120050012

[17] Niu Z, Wendland W, Wang X, Zhou H. A semi-analytical algorithm for the evaluation of the nearly singular integrals in three-dimensional boundary element methods. Computer Methods in Applied Mechanics and Engineering 2005; 194 (9-11): 1057-1074. doi: 10.1016/j.cma.2004.06.024

[18] Fata SN. Semi-analytic treatment of nearly-singular Galerkin surface integrals. Applied Numerical Mathematics 2010; 60 (10): 974-993. doi: 10.1016/j.apnum.2010.06.003

[19] Hu Z, Niu Z, Cheng C. A new semi-analytic algorithm of nearly singular integrals on higher order element in 3D potential BEM. Engineering analysis with boundary elements 2016; 63: 30-39. doi: 10.1016/j.enganabound.2015.11.001

[20] Telles JCF. A self adaptive co-ordinate transformation for efficient numerical evaluation of general boundary element integrals. International Journal For Numerical Methods in Engineering 1987; 24 (5): 959-973.

[21] Johnston PR, Elliott D. A generalisation of Telles' method for evaluating weakly singular boundary element integrals. Journal of Computational and Applied Mathematics 2001; 131 (1-2): 223-241. doi: 10.1016/S0377-0427(00)00273-9

[22] Johnston PR, Elliott D. Transformations for evaluating singular boundary element integrals. Journal of Computational and Applied Mathematics 2002; 146 (2): 231-251. doi: 10.1016/S0377-0427(02)00357-6

[23] Johnston P. Application of sigmoidal transformations to weakly singular and near-singular boundary element integrals. International Journal For Numerical Methods in Engineering 1999; 45 (10): 1333-1348. doi: 10.1002/(SICI)1097-0207(19990810)45:10<1333::AID-NME632>3.0.CO;2-Q

[24] Gu Y, Chen W, Zhang C. The sinh transformation for evaluating nearly singular boundary element integrals over high-order geometry elements. Engineering Analysis with Boundary Elements 2013; 37 (2): 301-308. doi: 10.1016/j.enganabound.2012.11.011

[25] Johnston BM, Johnston PR, Elliott D. A new method for the numerical evaluation of nearly singular integrals on triangular elements in the 3D boundary element method. Journal of Computational and Applied Mathematics 2013; 245: 148-161. doi: 10.1016/j.cam.2012.12.018

[26] Xie G, Zhang J, Huang C, Lu C, Li G. Calculation of nearly singular boundary element integrals in thin structures using an improved exponential transformation. Computer Modeling in Engineering & Sciences 2013; 94 (2): 139-157.

[27] Ma H, Kamiya N. Distance transformation for the numerical evaluation of near singular boundary integrals with various kernels in boundary element method. Engineering Analysis with Boundary Elements 2002; 26 (4): 329-339. doi: 10.1016/S0955-7997(02)00004-8

[28] Miao Y, Li W, Lv JH, Long XH. Distance transformation for the numerical evaluation of nearly singular integrals on triangular elements. Engineering Analysis with Boundary Elements 2013; 37 (10): 1311-1317. doi: 10.1016/j.enganabound.2013.06.009

[29] Yazici A. On the subdivision sequences of the extrapolation method of quadrature over a triangular region. METU Journal of Pure and Applied Sciences 1990: 23: 35-51.

[30] Bayindir H. Development of a parallel-extensible generic boundary element method application framework. PhD, Atılım University, Ankara, Turkey, 2017.

[31] Dunavant D. High degree efficient symmetrical Gaussian quadrature rules for the triangle. International Journal For Numerical Methods in Engineering 1985; 21 (6): 1129-1148. doi: 10.1002/nme.1620210612

[32] Zhang L, Cui T, Liu H. A set of symmetric quadrature rules on triangles and tetrahedra. Journal of Computational Mathematics 2009; 89-96.

[33] Becker AA. The Boundary Element Method in Engineering - A Complete Course. New York, NY, USA: McGraw-Hill, 1992.

[34] Cruse T. Numerical solutions in three dimensional elastostatics. International Journal of Solids and Structures 1969; 5 (12): 1259-1274. doi: 10.1016/0020-7683(69)90071-7

[35] Banerjee PK. The Boundary Element Methods in Engineering. 2 ed. New York, NY, USA: McGraw-Hill, 1981.