Journal of Computational Physics 397 (2019) 108831

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Quaternion and octonion-based finite element analysis methods for computing multiple first order derivatives

Mauricio Aristizabal [a,b], Daniel Ramirez-Tamayo [a], Manuel Garcia [b,c], Andres Aguirre-Mesa [a,b], Arturo Montoya [a], Harry Millwater [a,*]

[a] University of Texas at San Antonio, USA
[b] Universidad EAFIT, Medellín, Colombia
[c] Angelo State University, San Angelo, TX, USA

ARTICLE INFO

Article history:
Received 6 March 2019
Received in revised form 26 June 2019
Accepted 13 July 2019
Available online 24 July 2019

Keywords:
Quaternions
Cayley-Dickson numbers
Numerical differentiation
First order derivatives
Complex step

ABSTRACT

The complex Taylor series expansion method for computing accurate first order derivatives is extended in this work to quaternion, octonion and any order Cayley-Dickson algebra. The advantage of this new approach is that highly accurate multiple first order derivatives can be obtained in a single analysis. Quaternion and octonion-based finite element analysis methods were developed in order to compute up to three (quaternion) and up to seven (octonion) first order derivatives of shape, material properties, and/or loading conditions in a single analysis. The traditional finite element formulation was modified such that each degree-of-freedom was augmented with three or seven additional imaginary nodes. The quaternion and octonion-based methods were integrated within the Abaqus commercial finite element code through a user element subroutine. Numerical examples are presented for thermal conductivity and linear elasticity; however, the methodology is general. The results indicate that the quaternion and octonion-based methods provide derivatives of the same high accuracy as the complex finite element method but are significantly more efficient. A Fortran code to solve a simple seven variable quaternion example is given in the Appendix.

1. Introduction

The computation and application of derivatives from finite element models play an important role in sensitivity analysis, optimization, and reliability analysis, among others. Hence, methods to efficiently compute highly accurate shape, material and loading sensitivities are under continuous development. The complex variable finite element method is a recent highly effective method to compute first order derivatives. By extending the underlying complex Taylor series method to quaternion, octonion and any other Cayley-Dickson algebra, multiple first order derivatives can be computed in an efficient manner.

The complex Taylor series expansion (CTSE) method has been shown to be an effective procedure for obtaining accurate numerical first order derivatives from arbitrary holomorphic functions [1,2]. The method is simple in concept and

* Corresponding author.
E-mail addresses: maristi7@eafit.edu.co (M. Aristizabal), Daniel.Ramirez@my.utsa.edu (D. Ramirez-Tamayo), manuel.garcia@angelo.edu (M. Garcia), Andres.Aguirremesa@my.utsa.edu (A. Aguirre-Mesa), Arturo.Montoya@utsa.edu (A. Montoya), Harry.Millwater@utsa.edu (H. Millwater).

https://doi.org/10.1016/j.jcp.2019.07.030
0021-9991/

Nomenclature

a	Acceleration.
$\mathbf{b}$	Body force.
$\bar{b}$	Prescribed heat flux.
Bi	Biot number.
$\mathbf{B}_H$	Gradient matrix of heat transfer problem.
$\mathbf{B}_M$	Strain-displacement matrix.
$\mathbf{d}$	Nodal displacement vector.
$\mathbf{D}_H$	Thermal conductivity matrix.
$\mathbf{D}_M$	Elastic material matrix.
E	Elastic modulus.
$\mathbf{f}^*$	Global vector of independent coefficients.
$\mathbf{f}_s^{e*}$	Force vector due to a heat source.
$\mathbf{f}_{h_c}^{e*}$	Force vector due to convection.
$\mathbf{f}_b^{e*}$	Force vector due to a prescribed heat flux.
$f_{,x}$	Partial derivative of f with respect to x, $\partial f/\partial x$.
$f_{,xy}$	Mixed second order partial derivative of f with respect to x and y, $\partial^2 f/\partial x \partial y$.
H	Height of beam.
h	Step size.
h_c	Convection coefficient.
$\mathbf{J}$	Jacobian matrix.
k	Thermal conduction coefficient.
$\mathbf{K}^*$	Global system matrix.
$\mathbf{K}_{h_c}^{e*}$	Element convection matrix.
$\mathbf{K}_c^{e*}$	Element heat conduction matrix.
L	Length of beam.
$\mathbf{n}$	Unit normal.
$\mathbf{N}$	Shape function matrix.
$\mathcal{O}(\cdot)$	Order of the error term.
P	Applied load.
r	Radius of cylinder.
R	Normalized radius.
s	Heat generation rate.
$\bar{\mathbf{t}}$	Surface traction.
T	Temperature field.
$\bar{T}$	Prescribed temperature.
T_∞	Ambient temperature.
$\mathbf{T}$	Nodal temperature vector.
$\mathbf{u}$	Displacement field, nodal solution of a finite element analysis.
$\bar{\mathbf{u}}$	Prescribed displacement.
W	Integration weight.
x^*	Hypercomplex number.
α	Thermal Conductivity.
Γ	Boundary of domain.
ϵ	Dual imaginary direction.
ν	Poisson ratio.
σ	Stress vector.
ρ	Normalized outer radius.
ξ, η, γ	Natural Coordinates.
$\hat{\theta}$	Non-dimensional temperature.
$\bar{\boldsymbol{\tau}}$	Cauchy stress tensor.
Ω	Analysis domain.
∇	Gradient operator.
∇_s	Symmetric gradient operator.
$\mathbb{R}$	Real number set.
$\mathbb{C}$	Complex number set.
$\mathbb{H}$	Quaternion number set.
$\mathbf{Re}[x^*]$	Real coefficient of x^*.
$\mathbf{Im}_i[x^*]$	Coefficient of the i'th imaginary direction of x^*.
BS	Block-solver.
CD	Cayley-Dickson.
CDi	Central Differences.
CTSE	Complex Taylor Series Expansion.
CDTSE	Cayley-Dickson Taylor Series Expansion.
CFEM	Complex Finite Element Method.
CR	Cauchy-Riemann.
FD	Forward Differences.
FE	Finite Elements.
H.O.T.	High Order Terms.
LU	Lower-Upper decomposition.
OFEM	Octonion Finite Element Method.
OTSE	Octonion Taylor Series Expansion.
QFEM	Quaternion Finite Element Method.
QTSE	Quaternion Taylor Series Expansion.
UEL	Abaqus user element.
$(\cdot)_i$	Quantity at the inner surface.
$(\cdot)_o$	Quantity at the outer surface.

straightforward to implement: convert a real-valued code to complex-valued and introduce a small perturbation along the imaginary axis of the parameter of interest. The derivative of each degree-of-freedom with respect to the parameter of interest is then contained within its imaginary component. Derivatives of auxiliary results can then be obtained using standard post-processing methods such as obtaining strains and stresses and their derivatives within an elasticity analysis. This method obtains results with machine precision accuracy if the step size is sufficiently small, say $\leq 10^{-10}$ times the parameter of interest. CTSE is subtractive cancellation error free, meaning that derivatives can be computed with an arbitrarily small step size. In contrast, finite differences requires a selection of the step size, which is problem dependent and thus often a challenging task.

Successful implementation of CTSE within engineering software applications include heat transfer [3,4], structural dynamics [5], thermoelasticity [6], solid mechanics [7], plasticity [8], fluid dynamics [9], aerodynamics and aero-structural analysis [10,11], dynamic system optimization [12], pseudospectral [13] and eigenvalue sensitivity methods [14], among others. In [4], CTSE was used to improve the performance of the semi-analytic method in order to reduce the computational time expense and to reduce the sensitivity to the perturbation size compared with the standard finite-difference-driven semi-analytic method. A particularly successful application area of CTSE is to determine the energy release rate for linear

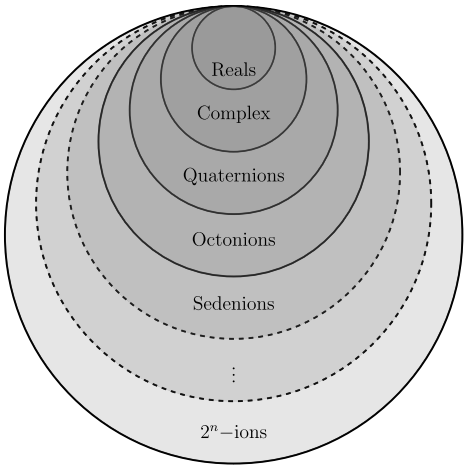


Fig. 1. Cayley-Dickson doubling process.

[15] and nonlinear fracture mechanics [16]. The energy release rate is the derivative of the strain energy with respect to a crack extension.

Dual numbers provide an equivalent method to CTSE regarding the computation of first order sensitivities [17]. In dual algebra, the square of the imaginary direction is $\epsilon^2 = 0$, and first order sensitivities are obtained by perturbing the variable in the imaginary direction ϵ by h. In contrast to CTSE, dual algebra analyses are insensitive to the step-size h and thus the derivatives are always machine error accurate. However, dual libraries are not standard in most programming languages, while complex types are more widely available to the end user and most programming languages have efficient complex implementations.

The extension of CTSE or dual methods to compute multiple first order derivatives or higher order derivatives require the use of hypercomplex algebra such as multicomplex [18] or hyperduals [19] (referred herein as multiduals). For example, computing two first order derivatives requires a bicomplex or bidual implementation, three derivatives requires tricomplex or tridual, etc. This approach is accurate but becomes grossly inefficient as more derivatives are added because the resulting size of the matrix representation increases by a factor of 2 with each increase in dimension. That is, a bicomplex matrix is 4 times larger and a tricomplex is 8 times larger than a real-valued analysis. As a result, using multi-complex/dual methods to compute multiple first order derivatives is not advantageous within a finite element formulation as the size of the element and global stiffness matrices becomes intractable. A better solution is to perform multiple complex/dual analyses, or, as shown here, use Cayley-Dickson algebras.

Quaternion algebra was introduced by Hamilton in 1843 to represent the mechanics of motion in three-dimensional space [20,21]. Quaternion numbers contain a real coefficient plus three coefficients along three imaginary directions. Their main application today is in three-dimensional computer graphics and computer vision, where they are used to compute three-dimensional rigid-body rotations [22]. Quaternions have also been used within the finite element method in order to represent rotations for beam element formulations [23–25].

Quaternion numbers are a subset of the Cayley-Dickson (CD) hypercomplex numbers [26]. CD algebras are formed by doubling a member of CD numbers, which contains the duplication of the number of dimensions of the initial algebra. For example, complex algebra (2 dimensions) is the result of doubling the real numbers, quaternion algebra (4 dimensions) is the result of doubling the complex numbers, etc. This process generates the set of CD numbers, which includes Octonions (8), Sedenions (16), etc., and associated algebraic rules. Fig. 1 shows the algebraic hierarchy available from the Cayley-Dickson doubling process. The number of real values required to describe a number for each case is doubled for each new algebra; Real (1), Complex (2), Quaternions (4), etc. In each case, $n-1$ imaginary units are available for computing derivatives in a manner analogous to the complex Taylor series expansion method; Real (0), Complex (1), Quaternions (3), Octonions (7), Sedenions (15), etc.

Turner presented a method that used quaternions to compute up to three first order derivatives [27]. The work used all three imaginary directions with small perturbations, such that when using the equivalent quaternion operations, the result contained the gradient of the function with respect to the perturbed variables, stating quadratic convergence with respect to the step size h. Turner also presented a method to solve systems of linear quaternion equations, which required programming a quaternion-based solver.

The present work extends Turner's work through the use of higher dimensional algebras such as octonions. It also shows that quadratic convergence of the derivatives with respect to the step size is only achieved under certain conditions and that linear convergence, not quadratic, is the expected convergence rate of the derivative results.

Moreover, the present work incorporates the quaternion and octonion Taylor Series expansion method into the Finite Element Method. A quaternion-based finite element formulation (QFEM), as described in this article, is capable of computing up to three first order derivatives in a single analysis without increasing the size of the stiffness matrix and an octonion-

based finite element method (OFEM) can compute up to seven first order derivatives. This approach can be extended to sedenions (15 derivatives) and higher dimensional CD algebras in a similar manner, thus generalized in the CDFEM method. The advantage of these methods is that the QFEM, OFEM, etc, can obtain multiple first order derivatives in a single analysis far more efficiently than an equivalent hypercomplex implementation.

Also, the present work introduces two methods for solving systems of linear hypercomplex equations using standard real-based solvers consisting of: *i*) a method based on the Cauchy-Riemann equivalent matrix form of the hypercomplex algebra, and *ii*) a new block solver method that subdivides the hypercomplex system into multiple real-only systems of equations that can be solved using multiple right hand sides with a single factorization of the real-valued stiffness matrix.

2. Methodology

2.1. Quaternion Taylor series expansion method

The quaternion Taylor series method (QTSE) is an extension of the complex Taylor series method (CTSE). CTSE [1,2] is a methodology to compute the sensitivity of a real function with respect to a variable of interest that is analogous to the finite difference (FD) method in the sense that a perturbation is applied to the parameter of interest. However, in CTSE the perturbation of the variable of interest x is applied along the imaginary axis, becoming $x^* = x_0 + hi$, where x_0 represents the point at which the derivative of the function is required and h is the perturbation step, addressed after Equation (3). The typical procedure consists of replacing every operation in the real function of interest f by its complex-equivalent operation (a process known as complexification of f), e.g. real multiplication is replaced by complex multiplication, etc. Using the complex-variable Taylor series, a function F can be expressed as

$$F(x^*) = F(x_0 + hi) = f(x_0) + f_{,x}(x_0)h\,i - \frac{1}{2!}f_{,x^2}(x_0)h^2 - \frac{1}{3!}f_{,x^3}(x_0)h^3\,i + \mathcal{O}(h^4) \tag{1}$$

where F is the result of complexifying f. Also, $f_{,x}$ represents the first order derivative with respect to x, $f_{,x^2}$ the second order derivative, etc. Note that the negative signs in Equation (1) come from factoring terms with i^2 and replacing it by the complex identity $i^2 = -1$. Taking the real and imaginary parts of both sides, Re[·] and Im[·] respectively; and solving for the function and its first order derivative, the following is obtained

$$f(x_0) = \quad \mathrm{Re}\left[F(x^*)\right] + \mathcal{O}(h^2) \tag{2}$$

$$f_{,x}(x_0) = \frac{1}{h}\mathrm{Im}\left[F(x^*)\right] + \mathcal{O}(h^2) \tag{3}$$

Note that the perturbation step size h can be made arbitrarily small with no concern about subtraction cancellation error. Hence, higher order effects of $\mathcal{O}(h^2)$ can be made negligible through the use of a small h. A typical value for h is 10^{-10} times the value of the parameter being perturbed.

The Quaternion Taylor Series Expansion (QTSE) method uses quaternion algebra (see Appendix A) to compute up to 3 first order derivatives of a real function evaluated at $x = x_0$, $y = y_0$, $z = z_0$. The variables x, y and z are perturbed by a small step size h in imaginary directions i, j and k respectively:

$$\begin{aligned} x^* &= x_0 + \boldsymbol{hi} + 0j + 0k, \\ y^* &= y_0 + 0i + \boldsymbol{hj} + 0k, \\ z^* &= z_0 + 0i + 0j + \boldsymbol{hk} \end{aligned} \tag{4}$$

The quaternion-variable Taylor series expansion (see Appendix C) adapted for the perturbations as shown in Equation (4), becomes

$$\mathcal{F}(x^*, y^*, z^*) = f(x_0, y_0, z_0) + h\left(f_{,x}(x_0, y_0, z_0)i + f_{,y}(x_0, y_0, z_0)j + f_{,z}(x_0, y_0, z_0)k\right) + \mathcal{O}(h^2) \tag{5}$$

where $\mathcal{F}$ is the quaternion-equivalent function of the real function f, which corresponds to the quaternionization of the operations in f. The result of $\mathcal{F}(x^*, y^*, z^*)$ is a quaternion containing the real function result and first order derivatives of f with respect to x, y and z, one lying along each imaginary direction:

$$f(x_0, y_0, z_0) \approx \mathrm{Re}\left[\mathcal{F}(x^*, y^*, z^*)\right] \tag{6}$$

$$f_{,x}(x_0, y_0, z_0) \approx \mathrm{Im}_i\left[\mathcal{F}(x^*, y^*, z^*)\right]/h \tag{7}$$

$$f_{,y}(x_0, y_0, z_0) \approx \mathrm{Im}_j\left[\mathcal{F}(x^*, y^*, z^*)\right]/h \tag{8}$$

$$f_{,z}(x_0, y_0, z_0) \approx \mathrm{Im}_k\left[\mathcal{F}(x^*, y^*, z^*)\right]/h \tag{9}$$

where $\mathrm{Re}[\cdot]$ extracts the real coefficient of the quaternion number, $\mathrm{Im}_i[\cdot]$ extracts the coefficient associated with the imaginary direction i, etc. See Appendix C for a more detailed elaboration.

Since no subtractions are performed during quaternion algebra operations, the method does not suffer from subtraction cancellation error. In practical numerical applications, the minimum possible error is usually bounded by machine error.

2.1.1. Convergence with respect to the step-size

Although Turner [27] expressed that when using quaternions to compute derivatives the convergence of the first order derivatives is quadratic with respect to the step size (as in CTSE), it can be shown that it is not the case when quaternion multiplication is present.

For the most general case, the function value converges quadratically with respect to the step-size h whereas the derivatives, Equations (7)–(9), converge linearly. These convergence rates are shown in the numerical examples section. In some special cases but not generally, quadratic convergence may be obtained in the first order derivatives. See Appendix C for more details.

On the other hand, CTSE converges quadratically on both function, Equation (2), and its derivative, Equation (3). However, since both complex and quaternion methods are subtraction cancellation error free, both will converge to the same error when using a sufficiently small h with no implication in performance.

2.1.2. Extension to higher dimensional hypercomplex algebras

In order to use higher dimensional CD algebras to compute multiple first order derivatives, e.g. octonions [28], sedenions [29], etc., each variable is associated with and perturbed independently along a unique imaginary direction, analogous to Equation (4). The maximum number of variables that the algebra is capable to compute derivatives with respect to in a single analysis is given by the total number of imaginary directions. This forms the generalized Cayley-Dickson Taylor series expansion method (CDTSE).

In the case of octonions (see Appendix B), a total of 8 coefficients are present in the algebra, corresponding to the real and 7 imaginary directions. Therefore, the algebra can be used to compute up to 7 first order derivatives in a single analysis. In addition, the 16-dimensional algebra of sedenions (one real and 15 imaginary directions) is able to compute 15 derivatives in a single analysis.

For octonions, forming the octonion Taylor series expansion (OTSE) method, the following perturbation scheme is used for a 7 variable function $f : \mathbb{R}^7 \to \mathbb{R}$, $f(x, y, z, a, b, c, d)$:

$$\begin{aligned}
x^* &= x_0 + \boldsymbol{hi} + 0j + 0k + 0e' + 0i' + 0j' + 0k', \quad & b^* &= b_0 + 0i + 0j + 0k + 0e' + \boldsymbol{hi'} + 0j' + 0k',\\
y^* &= y_0 + 0i + \boldsymbol{hj} + 0k + 0e' + 0i' + 0j' + 0k', \quad & c^* &= c_0 + 0i + 0j + 0k + 0e' + 0i' + \boldsymbol{hj'} + 0k',\\
z^* &= z_0 + 0i + 0j + \boldsymbol{hk} + 0e' + 0i' + 0j' + 0k', \quad & d^* &= d_0 + 0i + 0j + 0k + 0e' + 0i' + 0j' + \boldsymbol{hk'}\\
a^* &= a_0 + 0i + 0j + 0k + \boldsymbol{he'} + 0i' + 0j' + 0k'
\end{aligned} \tag{10}$$

Derivatives are obtained by evaluating f at $(x^*, y^*, z^*, a^*, b^*, c^*, d^*)$, which results in a hypercomplex octonion number, then extracting the imaginary coefficients and dividing each by the perturbation h. In the case of the perturbed variables in Equation (10), the evaluated function gives

$$f(x_0, y_0, z_0, a_0, b_0, c_0, d_0) \approx \mathrm{Re}\left[f(x^*, y^*, z^*, a^*, b^*, c^*, d^*)\right] \tag{11}$$

$$f_{,x}(x_0, y_0, z_0, a_0, b_0, c_0, d_0) \approx \mathrm{Im}_i\left[f(x^*, y^*, z^*, a^*, b^*, c^*, d^*)\right]/h \tag{12}$$

$$f_{,y}(x_0, y_0, z_0, a_0, b_0, c_0, d_0) \approx \mathrm{Im}_j\left[f(x^*, y^*, z^*, a^*, b^*, c^*, d^*)\right]/h \tag{13}$$

$$f_{,z}(x_0, y_0, z_0, a_0, b_0, c_0, d_0) \approx \mathrm{Im}_k\left[f(x^*, y^*, z^*, a^*, b^*, c^*, d^*)\right]/h \tag{14}$$

$$f_{,a}(x_0, y_0, z_0, a_0, b_0, c_0, d_0) \approx \mathrm{Im}_{e'}\left[f(x^*, y^*, z^*, a^*, b^*, c^*, d^*)\right]/h \tag{15}$$

$$f_{,b}(x_0, y_0, z_0, a_0, b_0, c_0, d_0) \approx \mathrm{Im}_{i'}\left[f(x^*, y^*, z^*, a^*, b^*, c^*, d^*)\right]/h \tag{16}$$

$$f_{,c}(x_0, y_0, z_0, a_0, b_0, c_0, d_0) \approx \mathrm{Im}_{j'}\left[f(x^*, y^*, z^*, a^*, b^*, c^*, d^*)\right]/h \tag{17}$$

$$f_{,d}(x_0, y_0, z_0, a_0, b_0, c_0, d_0) \approx \mathrm{Im}_{k'}\left[f(x^*, y^*, z^*, a^*, b^*, c^*, d^*)\right]/h \tag{18}$$

Similar to QTSE, using CDTSE methods, the convergence with respect to the step size is quadratic for the function value and linear for the derivatives. All results are also subtractive cancellation error free.

2.2. Numerical example: analytic 7 variable function

The proposed sensitivity method is applied to an analytic function of seven variables

$$f(a, b, c, d, x, y, z) = \sin(a^3 b^2 c^3 d^4 x^3 y^2 z^4) \tag{19}$$

to compute $f_{,a}$, $f_{,b}$, $f_{,c}$, $f_{,d}$, $f_{,x}$, $f_{,y}$ and $f_{,z}$ at the values of $a = 0.1$, $b = 0.2$, $c = 0.3$, $d = 0.4$, $x = 0.5$, $y = 0.6$ and $z = 0.7$. In particular, this example will be computed using seven CTSE, three QTSE and one OTSE analyses to serve for comparison purposes.

Table 1
Perturbation scheme for 7 complex Taylor series expansion analyses to compute 7 first order derivatives.

Variable	Evaluation						
	1	2	3	4	5	6	7
a^*	$0.1 + hi$	0.1	0.1	0.1	0.1	0.1	0.1
b^*	0.2	$0.2 + hi$	0.2	0.2	0.2	0.2	0.2
c^*	0.3	0.3	$0.3 + hi$	0.3	0.3	0.3	0.3
d^*	0.4	0.4	0.4	$0.4 + hi$	0.4	0.4	0.4
x^*	0.5	0.5	0.5	0.5	$0.5 + hi$	0.5	0.5
y^*	0.6	0.6	0.6	0.6	0.6	$0.6 + hi$	0.6
z^*	0.7	0.7	0.7	0.7	0.7	0.7	$0.7 + hi$

Table 2
Perturbation scheme for 3 quaternion Taylor series expansion analysis to compute 7 first order derivatives.

Variable	Evaluation		
	1	2	3
a^*	$0.1 + hi$	0.1	0.1
b^*	$0.2 + hj$	0.2	0.2
c^*	$0.3 + hk$	0.3	0.3
d^*	0.4	$0.4 + hi$	0.4
x^*	0.5	$0.5 + hj$	0.5
y^*	0.6	$0.6 + hk$	0.6
z^*	0.7	0.7	$0.7 + hi$

Table 3
Perturbation scheme for an octonion Taylor series expansion analysis to compute 7 first order derivatives.

Variable	Evaluation
	1
a^*	$0.1 + hi$
b^*	$0.2 + hj$
c^*	$0.3 + hk$
d^*	$0.4 + he'$
x^*	$0.5 + hi'$
y^*	$0.6 + hj'$
z^*	$0.7 + hk'$

2.2.1. Complex Taylor series expansion

Using CTSE, seven analyses must be performed to obtain the desired derivatives. In this case, each analysis computes the first order derivative with respect to one variable. Table 1 shows the perturbations required per function evaluation in order to obtain all seven derivatives.

According to Table 1, the first evaluation computes the first order derivative $f_{,a}$. The second evaluation computes $f_{,b}$, the third $f_{,c}$, etc.

2.2.2. Quaternion Taylor series expansion

Three evaluations are required to compute all seven derivatives using quaternion algebra. The perturbation schemes per evaluation are described in Table 2.

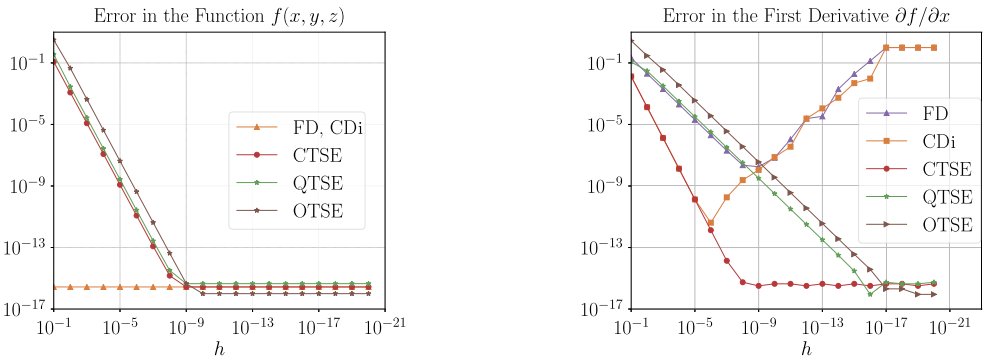
According to Table 2, the first evaluation computes $f_{,a}$, $f_{,b}$ and $f_{,c}$, contained in the imaginary directions i, j and k respectively. The second evaluation computes $f_{,d}$, $f_{,x}$ and $f_{,y}$. Finally, the third evaluation computes $f_{,z}$.

2.2.3. Octonion Taylor series expansion

Octonion algebra is formed by 7 imaginary directions, namely i, j, k, e', i', j' and k'. For a description of the algebra see Appendix B.

In order to evaluate the function in Equation (19) with octonions, only one analysis is required in order to compute all 7 derivatives. The perturbations are done as shown in Table 3.

Convergence of the relative error for first order derivatives and function results with respect to the step-size are shown in Fig. 2. Notice that the convergence rate of the derivative for the quaternion and octonion analyses is of first order. Also, all CDTSE methods evaluated: CTSE, QTSE and OTSE reached machine precision accuracy using step sizes h equal to 10^{-8}, 10^{-16} and 10^{-17} respectively. In practical terms, all analyses can be evaluated with a step-size $h = 10^{-17}$ or smaller. The source code for this example is given in Appendix D.



(a) Relative error of the function $f(a, b, c, d, x, y, z)$ value with respect to the step size h.

(b) Relative error of the first derivative $\partial f/\partial x$ with respect to the stepsize h.

Fig. 2. Convergence of the relative error of function $f(a, b, c, d, x, y, z)$ and first derivative $\partial f/\partial x$ for forward differences (FD), central differences (CDi), CTSE, QTSE and OTSE analyses.

2.3. The truncated Taylor series approach to compute functions of Cayley-Dickson numbers

The equations to compute typical functions of quaternions are well known [27] and it is possible to use them within the QTSE method described in section 2.1. However, the computational complexity of quaternion functions is usually high, and the methods for evaluation of other CD functions are not widely available. Hence a new method to evaluate functions of quaternions, octonions, and other hypercomplex algebras is presented here for use with the CD numbers containing very small imaginary components. The method is based on a Taylor series expansion of the CD function, extending the multicomplex and multidual function evaluation method proposed in [30] to CD numbers, with the condition that in the context of CDTSE, the step size h is small so that high order terms can be neglected.

For a single variable function $f(x) : \mathbb{R} \to \mathbb{R}$, e.g. $\sin(x)$, the proposed truncated Taylor series approach consists of the following. Consider a hypercomplex function $f(q^*)$. Its second order Taylor series expansion is

$$f(q^*) = f(q_{Re}) + f_{,x}(q_{Re})\ v^* + \frac{1}{2} f_{,xx}(q_{Re})\ \left(v^*\right)^2 \tag{20}$$

where q_{Re} is the real coefficient of the number and v^* is the whole imaginary part of q^*. For example, in the case of a quaternion q^*,

$$q^* = q_{Re} + q_i i + q_j j + q_k k \tag{21}$$

v^* is equal to the same quaternion number with real coefficient equal to zero, i.e.

$$v^* = q^* - q_{Re} = q_i i + q_j j + q_k k \tag{22}$$

Since v^* is only imaginary, and in particular to CDTSE applications all imaginary coefficients are multiplied by small values of h, its square and higher powers will result in terms multiplied by h^2 and higher. Thus, only for CDTSE applications, higher order terms may be neglected. Therefore, a function for CDTSE applications is evaluated as:

$$f(q^*) = f(q_{Re}) + f_{,x}(q_{Re})\ v^* \tag{23}$$

For instance, evaluating the sine function of a CD number q^* can be evaluated as:

$$\sin(q^*) = \sin(q_{Re}) + \cos(q_{Re}) v^* \tag{24}$$

where $\sin(q_{Re})$ and $\cos(q_{Re})$ are the standard real trigonometric functions, evaluated at the real-value q_{Re}.

The error induced by the higher order terms is negligible due to the accompanied power of the imaginary part of q^*. Hence, the effect of higher order terms can be made negligible through the use of a small h. The result is that functions of quaternions, octonions, etc., can be evaluated using one evaluation of a function of a real variable. Table 4 summarizes some commonly used functions.

2.4. Quaternion finite element method

The Quaternion Finite Element Method (QFEM) consists of replacing the real-valued variables with quaternion variables in order to compute first order derivatives with respect to a maximum of three variables. For simplicity, a heat transfer and a linear elasticity finite element implementations will be used to show the methodology. However, any other physics can also be implemented similarly.

Table 4
Alternative truncated Taylor form to evaluate functions of Cayley-Dickson numbers for CDTSE.

Function	Symbol	Equivalence
Sine	$\sin(q^*)$	$\sin(q_{Re}) + \cos(q_{Re})v^*$
Cosine	$\cos(q^*)$	$\cos(q_{Re}) - \sin(q_{Re})v^*$
Square root	$\sqrt{q^*}$	$\sqrt{q_{Re}} + \frac{1}{2\sqrt{q_{Re}}}v^*$
Power	$(q^*)^m$	$q_{Re}^m + mq_{Re}^{(m-1)}v^*$
Exponential	e^{q^*}	$e^{q_{Re}} + e^{q_{Re}}v^*$
Natural logarithm	$\ln(q^*)$	$\ln(q_{Re}) + \frac{v^*}{q_{Re}}$

Table 5
Input variables that define the heat transfer problem for isotropic materials.

Input parameter	Variable ϕ
Geometry	(x, y, z)
Thermal conductivity	α
Heat generation rate	s
Prescribed temperature	$\bar{T}$
Prescribed heat flux	$\bar{b}$
Convection coefficient	h_c
Ambient temperature	T_∞

2.4.1. Boundary value problem: heat transfer

The conservation of thermal energy for steady-state conditions can be expressed as

$$\begin{aligned} -\nabla^T(\mathbf{D}_H\nabla T) &= s && \text{in } \Omega \\ T &= \bar{T} && \text{on } \Gamma_1 \\ \mathbf{D}_H\nabla T\cdot\mathbf{n} &= \bar{b} && \text{on } \Gamma_2 \\ \mathbf{D}_H\nabla T\cdot\mathbf{n} &= h_c(T-T_\infty) && \text{on } \Gamma_3 \end{aligned} \tag{25}$$

where Ω is the region over which the problem is defined, T is the temperature field in Ω, $\mathbf{D}_H$ is the thermal conductivity matrix, ∇ is the gradient operator and s is the inner heat generation rate per unit volume. In the case of isotropic materials $\mathbf{D}_H = \alpha\mathbf{I}$, where α is the thermal conductivity. The boundary $\partial\Omega$ is decomposed into three regions: Γ_1 represents the region with essential (Dirichlet) conditions where the temperature $\bar{T}$ is prescribed; Γ_2 represents the region with natural (Neumann) conditions where the normal heat flux $\bar{b}$ is prescribed; and Γ_3 represents the region with mixed (Robin) conditions where convection is developed, hence h_c is the convection coefficient, T is the unknown temperature at the solid/fluid interface Γ_3, and T_∞ is the ambient temperature. $\mathbf{n}$ is the unit normal vector to the corresponding boundary.

Table 5 summarizes the input parameters that define the stationary heat transfer problem for isotropic materials.

2.4.2. Boundary value problem: linear elasticity

Static equilibrium for linear elastic materials can be expressed as

$$\begin{aligned} \nabla_s^T(\mathbf{D}_M\nabla_s\mathbf{v}) &= \mathbf{b} && \text{in } \Omega \\ \mathbf{u} &= \bar{\mathbf{u}} && \text{on } \Gamma_1 \\ \mathbf{t} = \bar{\boldsymbol{\tau}}\mathbf{n} &= \bar{\mathbf{t}} && \text{on } \Gamma_2 \end{aligned} \tag{26}$$

where ∇_s represents the symmetric gradient operator [31], $\mathbf{D}_M$ is the elastic material matrix that is determined by the elastic modulus, E, and the Poisson ratio, ν, for the case of isotropic materials. The variable $\mathbf{u}$ is the displacement vector field, $\bar{\mathbf{u}}$ is a prescribed displacement on Dirichlet boundary Γ_1, $\bar{\mathbf{t}}$ is the traction acting on Neumann boundary Γ_2 and $\bar{\boldsymbol{\tau}}$ is the Cauchy stress tensor.

Table 6 summarizes the input parameters that define the static equilibrium linear elastic problem for isotropic materials.

2.4.3. Finite element formulation

The finite element formulation is described in terms of quaternion variables but can be similarly extended to octonions, etc. The Quaternion Finite Element Method (QFEM) has been developed based on the QTSE method described in section 2.1. The finite element computations are conducted using quaternion variables in order to compute first order derivatives with respect to up to three variables of interest ϕ_1, ϕ_2 and ϕ_3 per QFEM analysis (see Table 5 or Table 6). In order to perform the quaternion analysis, these variables are perturbed in specific imaginary directions, e.g.

Table 6
Input variables that define the linear elastic problem for isotropic materials.

Input parameter	Variable ϕ
Geometry	(x, y, z)
Elastic modulus	E
Poisson ratio	ν
Body force	$\mathbf{b}$
Surface displacements	$\bar{\mathbf{u}}$
Surface traction	$\bar{\mathbf{t}}$

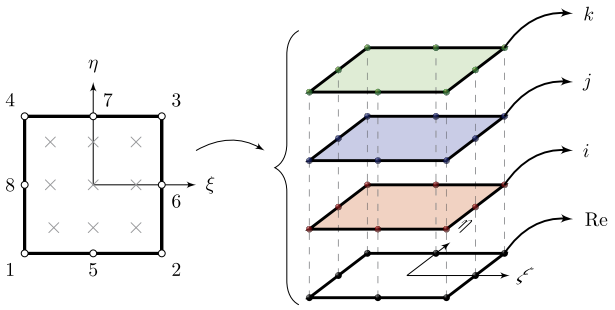


Fig. 3. 2D 8-noded quaternion quadrilateral element with the corresponding duplicates given by the imaginary directions i, j and k in natural coordinates and the 3×3 full integration scheme. A 3D 20-noded brick element with 20 DOF can be similarly constructed.

$$\phi_1^* = \phi_1 + \boldsymbol{hi} + 0j + 0k \tag{27}$$

$$\phi_2^* = \phi_2 + 0i + \boldsymbol{hj} + 0k \tag{28}$$

$$\phi_3^* = \phi_3 + 0i + 0j + \boldsymbol{hk} \tag{29}$$

where the step-size for a quaternion analysis is typically $h = 10^{-17}$ or smaller times the value of the perturbed parameter for a double precision analysis due to the linear convergence of QTSE. The three perturbed variables may be any combination of the input parameters of the problem, which may come from either Table 5 or Table 6. For example, to obtain the sensitivity in a heat transfer analysis of the temperature solution with respect to the thermal conductivity k, heat generation rate s and convection coefficient h_c, the variables are perturbed as: $k^* = k + hi$, $s^* = s + hj$ and $h_c^* = h_c + hk$. Note that the real part of the quaternion input variables corresponds to the input values of the conventional finite element analysis.

Due to the quaternion inputs, additional degrees of freedom have to be defined in order to obtain the derivative information. As an example, Fig. 3 shows an 8-noded quaternion quadrilateral element appropriate for a thermal analysis with 4 degrees of freedom per node ($\mathbf{u}$, $\mathbf{u}_{,\phi_1}h$, $\mathbf{u}_{,\phi_2}h$ and $\mathbf{u}_{,\phi_3}h$) with a 3×3 integration scheme in natural coordinates with a node for each hypercomplex direction.

Due to the quaternion inputs, the QFEM analysis generates a quaternion system of equations:

$$\mathbf{K}^*\mathbf{u}^* = \mathbf{f}^* \tag{30}$$

where $\mathbf{u}^*$ is the nodal solution vector which represents either the nodal temperature if solving a heat transfer analysis or the nodal displacements if solving a linear elastic problem. Regardless of the problem being solved, the result of the QFEM analysis is composed of quaternion numbers and thus it is formed by the sum of vectors directed along imaginary directions:

$$\mathbf{u}^* = \mathbf{u}_{Re} + \mathbf{u}_i i + \mathbf{u}_j j + \mathbf{u}_k k \tag{31}$$

where $\mathbf{u}_{Re}$ is equivalent to the conventional finite element nodal solution vector (e.g. nodal temperature or nodal displacements), $\mathbf{u}_i$ is a vector containing the derivative of nodal solution with respect to parameter ϕ_1, $\mathbf{u}_j$ is a vector with the derivative with respect to ϕ_2 and $\mathbf{u}_k$ contains the derivative with respect to ϕ_3, that is:

$$\mathbf{u}_{Re} \approx \mathbf{u} \tag{32}$$

$$\mathbf{u}_i \approx h\mathbf{u}_{,\phi_1} \tag{33}$$

$$\mathbf{u}_j \approx h\mathbf{u}_{,\phi_2} \tag{34}$$

$$\mathbf{u}_k \approx h\mathbf{u}_{,\phi_3} \tag{35}$$

As in the conventional finite element method, traditional real-valued element shape functions are used in QFEM for interpolation of the nodal solution within a finite element:

$$u^*(x, y, z) = \mathbf{N}\mathbf{u}^* \tag{36}$$

where $\mathbf{N}$ is the matrix of real-valued element shape functions. Consequently, the derivatives of the solution field $u(x, y, z)$ with respect to the input parameters are approximated using the shape function matrix $\mathbf{N}$ as

$$u(x, y, z) \approx \quad \mathbf{N}\mathbf{u}_{Re} \tag{37}$$

$$u_{,\phi_1}(x, y, z) \approx \frac{1}{h}\mathbf{N}\mathbf{u}_i \tag{38}$$

$$u_{,\phi_2}(x, y, z) \approx \frac{1}{h}\mathbf{N}\mathbf{u}_j \tag{39}$$

$$u_{,\phi_3}(x, y, z) \approx \frac{1}{h}\mathbf{N}\mathbf{u}_k \tag{40}$$

The quaternion global system matrix $\mathbf{K}^*$ is a matrix whose elements are quaternion numbers, and $\mathbf{f}^*$ is the force vector which is also formed by quaternion elements. Each $\mathbf{K}^*$ and $\mathbf{f}^*$ can be separated by its imaginary coefficients as follows:

$$\mathbf{K}^* = \mathbf{K}_{Re} + \mathbf{K}_i i + \mathbf{K}_j j + \mathbf{K}_k k \tag{41}$$

$$\mathbf{f}^* = \mathbf{f}_{Re} + \mathbf{f}_i i + \mathbf{f}_j j + \mathbf{f}_k k \tag{42}$$

where $\mathbf{K}_{Re}$ and $\mathbf{f}_{Re}$ are the conventional global system matrix and global vector of independent terms, $\mathbf{K}_i$ and $\mathbf{f}_i$ contain approximations of the derivatives with respect to ϕ_1, i.e. $h\mathbf{K}_{,\phi_1}$ and $h\mathbf{f}_{,\phi_1}$ respectively. Analogously, the matrices and vectors in j and k directions contain the derivatives with respect to parameters ϕ_2 and ϕ_3 respectively. Both $\mathbf{K}^*$ and $\mathbf{f}^*$ depend on the input parameters of the problem being solved. They are formed using an element-by-element approach as done in a traditional FE method albeit now using quaternion degrees-of-freedom.

The Jacobian matrix $\mathbf{J}$ depends on the coordinates (x, y, z) of the nodes that define the element in the original domain. Since the nodal coordinates are input parameters, they may be perturbed in any imaginary direction and thus they are quaternion-valued inputs. As a consequence, $\mathbf{J}$ becomes a matrix of quaternion numbers and is computed as follows:

$$\mathbf{J}^* = \begin{bmatrix} N_{1,\xi} & N_{2,\xi} & \cdots \\ N_{1,\eta} & N_{2,\eta} & \cdots \\ N_{1,\gamma} & N_{2,\gamma} & \cdots \end{bmatrix} \begin{bmatrix} x_1^* & y_1^* & z_1^* \\ x_2^* & y_2^* & z_2^* \\ \vdots & \vdots & \vdots \end{bmatrix} \tag{43}$$

where the elemental shape functions $(N_1, N_2, ...)$ correspond to the conventional FEM shape functions for the corresponding element interpolation. The quaternion nodal coordinates are defined for the p'th node as:

$$x_p^* = x_{p_{Re}} + x_{pi}i + x_{pj}j + x_{pk}k, \quad y_p^* = y_{p_{Re}} + y_{pi}i + y_{pj}j + y_{pk}k, \quad z_p^* = z_{p_{Re}} + z_{pi}i + z_{pj}j + z_{pk}k \tag{44}$$

Nodal coordinates are perturbed when shape derivatives are required using the imaginary nodal coordinates [15]. For instance, consider a 2D analysis with a domain that has a circular hole. If the derivative with respect to the location of the hole is required, say in the horizontal direction, a perturbation is applied to the nodes adjacent to the circle boundary in the x coordinates as shown in Fig. 4a. If the sensitivity with respect to the radius of the circle is required, then a perturbation is applied to all adjacent nodes in the radial direction as shown in Fig. 4b, representing virtual displacements of the nodes in the imaginary direction. In QFEM, for example, imaginary directions i and j can be used to compute sensitivities with respect to the location of the hole in the x and y directions respectively. Meanwhile the k imaginary direction can be used to compute the sensitivity with respect to the radius of the hole.

In particular to the heat transfer problem, the element system matrix is formed by

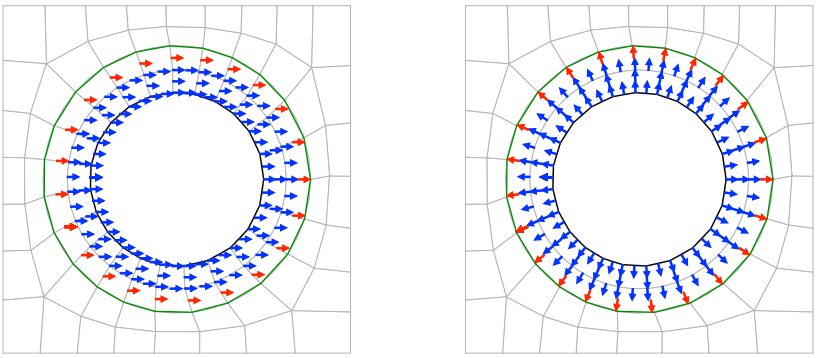
$$\mathbf{K}^{e*} = \mathbf{K}_c^{e*} + \mathbf{K}_{h_c}^{e*} \tag{45}$$

where $\mathbf{K}_c^{e*}$ is the portion given by heat conduction and $\mathbf{K}_{h_c}^{e*}$ is the contribution of convection to the problem.

The element heat conduction matrix is computed as:

$$\mathbf{K}_c^{e*} = \int_{\Omega_e} \mathbf{B}^{*T}_H \mathbf{D}_H^* \mathbf{B}_H^* \, |\mathbf{J}^*| \, d\Omega \tag{46}$$

where $\mathbf{D}_H^*$ is the quaternion-valued thermal conductivity matrix. The temperature gradient matrix $\mathbf{B}_H^*$ also becomes a matrix of quaternion numbers because it depends on the Jacobian $\mathbf{J}$:



(a) Perturbation to obtain the sensitivity with respect to the horizontal location of the hole.

(b) Perturbation to obtain the sensitivity with respect to the radius of the hole.

Fig. 4. Examples of the nodal perturbations to obtain shape derivatives of a circular feature in a domain. The green circle marks the perturbed region inside which all nodes are perturbed, while the blue and red arrows determine the magnitude of the perturbation, h and $h/2$ respectively. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

$$\mathbf{B}_H^* = \left[\mathbf{J}^*\right]^{-1} \begin{bmatrix} N_{1,\xi} & N_{2,\xi} & \cdots \\ N_{1,\eta} & N_{2,\eta} & \cdots \\ N_{1,\gamma} & N_{2,\gamma} & \cdots \end{bmatrix} \tag{47}$$

The determinant of the Jacobian matrix $\mathbf{J}^*$ is computed using the standard matrix determinant operations and thus the result is a scalar quaternion value. As an example, the determinant of a two dimensional analysis with a 2×2 Jacobian matrix is calculated as follows:

$$|\mathbf{J}^*| = \begin{vmatrix} J_{11}^* & J_{12}^* \\ J_{21}^* & J_{22}^* \end{vmatrix} = J_{11}^* J_{22}^* - J_{12}^* J_{21}^* \tag{48}$$

In standard applications, Equation (46) is carried out using the Gaussian integration scheme for each element type. In the case of a quaternion implementation such integration is carried out as:

$$\mathbf{K}_c^{e*} \approx \sum_{r}^{n_{gp_\gamma}} \sum_{l}^{n_{gp_\eta}} \sum_{p}^{n_{gp_\eta}} W_r W_l W_p \, \mathbf{B}^*{}_H(\xi_p, \eta_l, \gamma_r)^T \mathbf{D}_H^* \mathbf{B}_H^*(\xi_p, \eta_l, \gamma_r) \, |\mathbf{J}^*(\xi_p, \eta_l, \gamma_r)| \tag{49}$$

where W_r, W_l, W_p and $(\xi_p,\ \eta_l,\ \gamma_r)$ are the conventional real-valued weights and evaluation points for the corresponding quadrature.

Similarly, taking into account the quaternion-valued convection coefficient h_c^*, the element convection matrix is formulated as

$$\mathbf{K}_{h_c}^{e*} = \int_{\partial\Omega_e} h_c^* \mathbf{N}^T \mathbf{N} \, |\mathbf{J}^*| \, d\Gamma \tag{50}$$

and is integrated using real-valued Gaussian integration weights and points.

The element force vector is composed by

$$\mathbf{f}^{e*} = \mathbf{f}_b^{e*} + \mathbf{f}_{h_c}^{e*} + \mathbf{f}_s^{e*} \tag{51}$$

where $\mathbf{f}_b^{e*}$ is the force vector due to a prescribed heat flux, $\mathbf{f}_{h_c}^{e*}$ elemental heat conduction matrix is the force vector due to convection and $\mathbf{f}_s^{e*}$ is the force vector due to a heat source. The force vectors are defined as

$$\mathbf{f}_b^{e*} = \int_{\partial\Omega_e} b^* \mathbf{N}^T \, |\mathbf{J}^*| \, d\Gamma \tag{52}$$

$$\mathbf{f}_{h_c}^{e*} = \int_{\partial\Omega_e} h_c^* T_\infty^* \mathbf{N}^T \, |\mathbf{J}^*| \, d\Gamma \tag{53}$$

$$\mathbf{f}_s^{e*} = \int_{\Omega_e} s^* \mathbf{N}^T \, |\mathbf{J}^*| \, d\Omega \tag{54}$$

For the linear elastic problem, the element system matrix $\mathbf{K}^{e*}$ is formed by

$$\mathbf{K}^{e*} = \int_{\Omega_e} \mathbf{B}^{*T}_M \mathbf{D}_M^* \mathbf{B}_M^* \, |\mathbf{J}^*| \, d\Omega \tag{55}$$

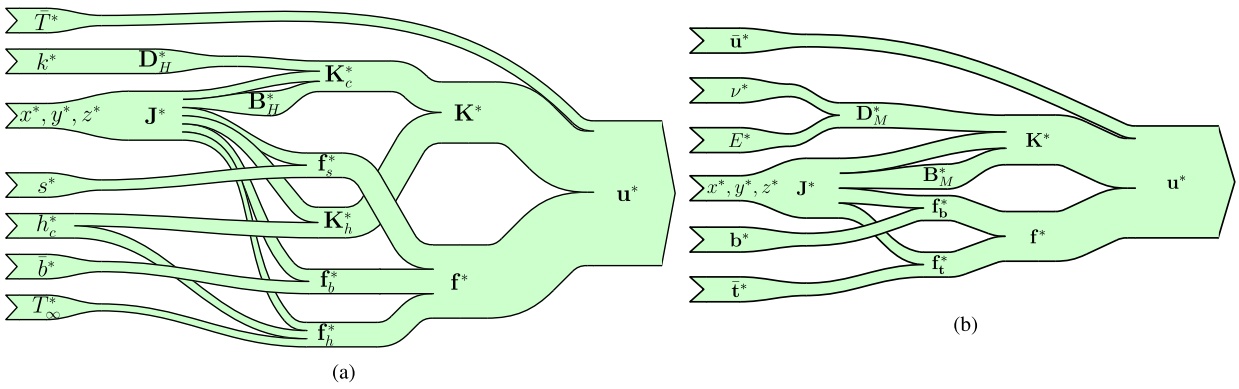


Fig. 5. Dependency of the intermediate variables of the finite element problem with respect to the input variables for (a) heat transfer and (b) linear elasticity problems.

where $\mathbf{B}_M^*$ is the strain-displacement matrix. The element force vector is composed by

$$\mathbf{f}_M^{e*} = \mathbf{f}_\mathbf{t}^{e*} + \mathbf{f}_\mathbf{b}^{e*} \tag{56}$$

were $\mathbf{f}_\mathbf{t}^{e*}$ is the force vector due to surface tractions and $\mathbf{f}_\mathbf{b}^{e*}$ is the force vector due to body forces which are calculated as

$$\mathbf{f}_\mathbf{t}^{e*} = \int_{\partial\Omega_e} \mathbf{N}^T \bar{\mathbf{t}}^* \, |\mathbf{J}^*| \, d\Gamma \tag{57}$$

$$\mathbf{f}_\mathbf{b}^{e*} = \int_{\Omega_e} \mathbf{N}^T \mathbf{b}^* \, |\mathbf{J}^*| \, d\Omega \tag{58}$$

2.4.4. Dependency of intermediate variables

Until now, this paper has discussed the hypercomplexification of all variables and intermediate variables in the finite element analysis in order to provide the capability of computing sensitivities with respect to any input variable for any analysis. However, only certain elements of the code may need to be hypercomplexified in order to provide the correct capabilities for computing the desired derivatives, and hence, increasing computational performance.

Fig. 5a summarizes the dependency of the intermediate variables with respect to the input variables for the heat transfer problem and Fig. 5b analogously summarizes the dependency of the intermediate variables of the linear elastic finite element problem. The nodal temperatures $\mathbf{u}^*$, solution of the QFEM analysis, will always be hypercomplex. However, if the desired analysis does not require sensitivities with respect to shape variables nor nodal coordinates (x, y, z), then neither the Jacobian matrix $\mathbf{J}^*$ nor the gradient matrix $\mathbf{B}^*$ are required to be hypercomplexified, hence they can be used as standard real-only matrices. Moreover, e.g., if the sensitivity with respect to the heat conduction coefficient k is required, then it is mandatory that the heat conduction matrix $\mathbf{K}_c^*$ and the global system matrix $\mathbf{K}^*$ be defined as hypercomplex-typed arrays.

2.4.5. Assembly and solution of the global system of equations

The quaternion/octonion finite element method produces element matrices and vectors formed by quaternion/octonion numbers. Instead of implementing a quaternion/octonion solver, this paper describes two methods to solve the system of equations using real-based solvers: *i*) convert the quaternion/octonion system of equations to an equivalent Cauchy-Riemann form of all real numbers (see Appendix A.2), and then use a conventional direct solver, or *ii*) reduce the system of equations to lower triangular then use a block solver method, as discussed below. Both solution methods provide the same result; however, the block solver method is more efficient.

For method *i*) mentioned above, a single system of equations with all-real coefficients is formed based on the corresponding matrix-vector forms of quaternion/octonion numbers. The system matrix $\mathbf{K}^*$ is transformed into its matrix form and vectors $\mathbf{u}^*$ and $\mathbf{f}^*$ are transformed into vector forms, and thus an equivalent system of equations composed of all real numbers is formulated as:

$$\overbrace{\begin{bmatrix} \mathbf{K}_{\text{Re}} & -\mathbf{K}_i & -\mathbf{K}_j & -\mathbf{K}_k \\ \mathbf{K}_i & \mathbf{K}_{\text{Re}} & -\mathbf{K}_k & \mathbf{K}_j \\ \mathbf{K}_j & \mathbf{K}_k & \mathbf{K}_{\text{Re}} & -\mathbf{K}_i \\ \mathbf{K}_k & -\mathbf{K}_j & \mathbf{K}_i & \mathbf{K}_{\text{Re}} \end{bmatrix}}^{\mathbf{T}(\mathbf{K}^*)} \overbrace{\begin{bmatrix} \mathbf{u}_{\text{Re}} \\ \mathbf{u}_i \\ \mathbf{u}_j \\ \mathbf{u}_k \end{bmatrix}}^{\mathbf{t}(\mathbf{u}^*)} = \overbrace{\begin{bmatrix} \mathbf{f}_{\text{Re}} \\ \mathbf{f}_i \\ \mathbf{f}_j \\ \mathbf{f}_k \end{bmatrix}}^{\mathbf{t}(\mathbf{f}^*)} \tag{59}$$

where $\mathbf{T}(\cdot)$ and $\mathbf{t}(\cdot)$ are the matrix and vector transformations of the quaternion algebra (see Appendix A.2), respectively. Notice that this conversion can be performed at the local element level before assembly or at the global level after assembly. The resulting system of equations (59) is unsymmetric. However, in the case of symmetric problems, e.g. heat transfer and linear elasticity, where $\mathbf{K}_{\mathrm{Re}}$ is symmetric; the Cauchy-Riemann matrix of Equation (59) can be transformed to symmetric by changing the sign of all the elements of its upper triangular portion, Equation (60). This does not affect the overall result of the QFEM/OFEM analysis since the product between imaginary terms, e.g., $\mathbf{K}_i\ \mathbf{u}_j$, are of order h^2 and thus insignificant to the QFEM/OFEM analysis. Therefore, a symmetric solver is used.

$$\begin{bmatrix} \mathbf{K}_{\mathrm{Re}} & -\mathbf{K}_i & -\mathbf{K}_j & -\mathbf{K}_k \\ \mathbf{K}_i & \mathbf{K}_{\mathrm{Re}} & -\mathbf{K}_k & \mathbf{K}_j \\ \mathbf{K}_j & \mathbf{K}_k & \mathbf{K}_{\mathrm{Re}} & -\mathbf{K}_i \\ \mathbf{K}_k & -\mathbf{K}_j & \mathbf{K}_i & \mathbf{K}_{\mathrm{Re}} \end{bmatrix} \mapsto \begin{bmatrix} \mathbf{K}_{\mathrm{Re}} & \mathbf{K}_i & \mathbf{K}_j & \mathbf{K}_k \\ \mathbf{K}_i & \mathbf{K}_{\mathrm{Re}} & \mathbf{K}_k & -\mathbf{K}_j \\ \mathbf{K}_j & \mathbf{K}_k & \mathbf{K}_{\mathrm{Re}} & \mathbf{K}_i \\ \mathbf{K}_k & -\mathbf{K}_j & \mathbf{K}_i & \mathbf{K}_{\mathrm{Re}} \end{bmatrix} \tag{60}$$

The system of equations for a quaternion implementation forms a system of equations that is 16 times larger than the real-only system $\mathbf{K}_{Re}\mathbf{u}_{Re} = \mathbf{f}_{Re}$. In the case of octonions, the equivalent system of equations is 64 times the size of the real-only system.

In the case of method *ii*) as mentioned above, the system of equations (59) can be expanded and subdivided into simpler blocks in order to optimize its solution. First, expand Equation (59) to form four systems of equations, then remove the terms of order h^2 as shown below

$$\mathbf{K}_{\mathrm{Re}}\mathbf{u}_{\mathrm{Re}} - \cancelto{0}{\mathbf{K}_i\ \mathbf{u}_i} - \cancelto{0}{\mathbf{K}_j\ \mathbf{u}_j} - \cancelto{0}{\mathbf{K}_k\ \mathbf{u}_k} = \mathbf{f}_{\mathrm{Re}} \tag{61}$$

$$\mathbf{K}_i\ \mathbf{u}_{\mathrm{Re}} + \mathbf{K}_{\mathrm{Re}}\mathbf{u}_i - \cancelto{0}{\mathbf{K}_k\ \mathbf{u}_j} + \cancelto{0}{\mathbf{K}_j\ \mathbf{u}_k} = \mathbf{f}_i \tag{62}$$

$$\mathbf{K}_j\ \mathbf{u}_{\mathrm{Re}} + \cancelto{0}{\mathbf{K}_k\ \mathbf{u}_i} + \mathbf{K}_{\mathrm{Re}}\mathbf{u}_j - \cancelto{0}{\mathbf{K}_i\ \mathbf{u}_k} = \mathbf{f}_j \tag{63}$$

$$\mathbf{K}_k\ \mathbf{u}_{\mathrm{Re}} - \cancelto{0}{\mathbf{K}_j\ \mathbf{u}_i} + \cancelto{0}{\mathbf{K}_i\ \mathbf{u}_j} + \mathbf{K}_{\mathrm{Re}}\mathbf{u}_k = \mathbf{f}_k \tag{64}$$

Using the QTSE method, each coefficient in the imaginary directions i, j and k contains a factor h, e.g. $\mathbf{K}_i = h\mathbf{K}_{,\phi_1}$. Therefore, each multiplication between two imaginary components will become a term of order h^2, e.g. $\mathbf{K}_i\mathbf{u}_i = h^2\mathbf{K}_{,\phi_1}\mathbf{u}_{,\phi_1}$. As a consequence, the terms formed by the multiplication between two imaginary coefficients can be neglected such as $\mathbf{K}_i\mathbf{u}_i$ or $\mathbf{K}_j\mathbf{u}_k$. Using this reduction method, the system of equations can be rewritten as

$$\mathbf{K}_{\mathrm{Re}}\mathbf{u}_{\mathrm{Re}} = \mathbf{f}_{\mathrm{Re}} \tag{65}$$

$$\mathbf{K}_{\mathrm{Re}}\mathbf{u}_i = \mathbf{f}_i - \mathbf{K}_i\mathbf{u}_{\mathrm{Re}} \tag{66}$$

$$\mathbf{K}_{\mathrm{Re}}\mathbf{u}_j = \mathbf{f}_j - \mathbf{K}_j\mathbf{u}_{\mathrm{Re}} \tag{67}$$

$$\mathbf{K}_{\mathrm{Re}}\mathbf{u}_k = \mathbf{f}_k - \mathbf{K}_k\mathbf{u}_{\mathrm{Re}} \tag{68}$$

where the solution of the system (65) solves the standard real system of equations, obtaining the conventional finite element solution $\mathbf{u}_{\mathrm{Re}}$. The real stiffness matrix $\mathbf{K}_{\mathrm{Re}}$, the real solution $\mathbf{u}_{\mathrm{Re}}$ and the imaginary components of $\mathbf{f}^*$ and $\mathbf{K}^*$ are used to solve the imaginary coefficients of $\mathbf{u}^*$, Equations (66)–(68). All systems (65)–(68) share the same matrix of coefficients $\mathbf{K}_{\mathrm{Re}}$, the conventional stiffness matrix of the finite element formulation, and thus Lower-Upper (LU) decomposition or Cholesky decomposition may be used to take advantage of the formulation with multiple right-hand sides. This approach does not require the formation of the full Cauchy-Riemann matrix form for the system. The Cauchy-Riemann form was used only to deduce the method. It only requires the matrices and vectors described in Equations (66)–(68).

3. Abaqus implementation

The quaternion finite element formulations for heat transfer and linear elasticity analyses of 2D bodies were implemented within the Abaqus commercial finite element software through the implementation of user element subroutines (UELs).

To convert a traditional, real-only Abaqus finite element code into a quaternion-based one, a support library is required to define the quaternion number type and overload its operators. The minimal operations to be supported and overloaded are given in Table 7. Although matrix and vector functions are optional, in the general case the code is simplified by writing it in matrix form, thus these operations are included in Table 7.

Both approaches for solving the system of equations discussed in section 2.4.5, *i*) the direct Cauchy-Riemann approach and *ii*) the Block Solver approach were implemented in the Abaqus software.

Table 7
Basic operations required to hypercomplexify a Finite Element code.

	Operation
Scalar	Addition/Subtraction Multiplication/Division Power
Vector	Dot product Addition/Subtraction
Matrix	Matrix multiplication Matrix transpose Matrix inverse

3.1. Direct Cauchy-Riemann approach

The Cauchy-Riemann solution approach requires the least modification of a conventional real UEL code compared to the Block-Solver method. In this paper we extend the strategy proposed previously [15] to quaternions and octonions. To implement this approach, the traditional real Abaqus UEL code is quaternionized/octonionized by defining every intermediate variable (see Fig. 5) as quaternion/octonion type instead of real. In addition to this, the input and output parameters should also be defined as quaternion/octonion variables. Since Abaqus is restricted to real-only parameters, the Cauchy-Riemann vector and matrix forms are used within the UEL. In this case, the UEL is defined with $4\times$ the number of degrees-of-freedom (DOFs) of the real element to contain the imaginary coefficients of the quaternion algebra and $8\times$ the number of DOFs for the octonion algebra. For example, an 8-noded quadrilateral element with one DOF per node will become a 32-noded element (see Fig. 3) with a total of 32-DOFs; each DOF has 4 values at each node, e.g. u, $\partial u/\partial \phi_1$, $\partial u/\partial \phi_2$, $\partial u/\partial \phi_3$ for quaternions. The real coefficients of the nodes and their degrees of freedom correspond to the traditional FEM. Each additional imaginary DOF contains derivative information with respect to the perturbed input parameters. In Abaqus, increasing the number of DOFs automatically generates a system with $4\times$ the number of equations and thus a matrix of $16\times$ the size of that of the real analysis for quaternions and with $8\times$ the number of equations and $64\times$ the matrix size of the real analysis for the case of octonions.

Increasing the number of DOFs requires the definition of additional nodes, which therefore requires the definition of the corresponding coordinates. Since all new nodes correspond to imaginary directions, the coordinate values of the added nodes correspond to the perturbation applied to the real associated node in the corresponding imaginary direction in x, y and z respectively (see Fig. 4). Other input parameters such as the modulus of elasticity, thermal conductivity, etc., are defined as input parameters to the UEL containing its imaginary coefficients as well. Each input parameter is converted from a set of 4/8 independent values to quaternion/octonion variables at the start of the UEL. Boundary conditions are also expanded to 4/8 inputs per condition each, where consistency is required while associating boundary conditions to the corresponding imaginary nodes.

After the quaternionized/octonionized UEL computes the stiffness matrix and load vectors, the UEL converts the system matrix and load vector to the real-only Cauchy-Riemann corresponding forms following the structure defined in Equation (59) with the symmetric matrix as defined in Equation (60). Afterwards, no further modification to the Abaqus conventional routines is required, thus the assembly procedure is performed by conventional Abaqus calls and the solution is found using conventional real-only solvers.

After the analysis, post-processing is required to collect the derivative information, which is contained within the nodal solutions of the DOFs associated to the imaginary nodes as described before.

3.2. Block-solver approach

As previously discussed in Section 2.4.5, the quaternion/octonion system of equations can be reduced to lower triangular and then subdivided into multiple subsystems. This approach results in a more efficient implementation than the direct Cauchy-Riemann method.

The implementation of the block solver solution scheme within Abaqus is composed of two steps. First, an initial step is performed as a conventional finite element analysis to solve the real system of equations, Equation (65). Then, a second and final step is used to form and solve the subsequent systems (see Equations (66)–(68)), using the "multiple load case analysis" in Abaqus, which is used to study the linear response of a structure subjected to distinct sets of loads and boundary conditions. In the second step, the UEL uses the solution from the previous step ($\mathbf{u}_{Re}$) and quaternion/octonion algebra to form the right hand sides of the system of equations. The total number of load cases (right hand sides) corresponds to the number of imaginary directions of the algebra, three for quaternions and seven for octonions. Each load case is uniquely associated to a right hand side and thus to a unique imaginary direction of the algebra. The boundary conditions of a particular load case represent the perturbation of that boundary condition in the corresponding imaginary direction. Perturbations to the nodal coordinates are defined as field variables and the perturbations to the modulus of elasticity, thermal conductivity, etc., are defined as input parameters to the UEL.

The advantage of using the multiple load case approach is that Abaqus shares the stiffness matrix for all right hand sides. This is possible due to the fact that systems (66)–(68) share the same stiffness matrix, $\mathbf{K}_{\mathrm{Re}}$. Notice that both steps generate systems with the same number of DOFs as the conventional real-only finite element analysis.

This approach results in significant advantages over the direct Cauchy-Riemann approach where the number of DOFs grows 4× (quaternions) and 8× (octonions); and the system of equations grows to 16× (quaternions) and 64× (octonions) the size of the real-only system. Then the system is assembled element by element, in which each element matrix is transformed to its Cauchy-Riemann equivalent system.

4. Numerical examples

4.1. Finite element Abaqus example: hollow cylinder with internal heat generation and convective surfaces

A two dimensional axisymmetric thermal sensitivity analysis was conducted for a hollow cylinder with internal heat generation in which the inner and outer surfaces are under convective conditions. This example possesses an analytical solution presented by [32]. Also, [6] presented a complex-valued FEM for sensitivity analysis of thermoelastic problems in which the same example was analyzed using CTSE. For each sensitivity, a complex-valued finite element analysis was required. However, using a quaternion/octonion based FEM, three/seven different sensitivities can be obtained in a single analysis. The quaternion and octonion FEM were implemented using an Abaqus User Element (UEL).

The non-dimensional analytical temperature through the thickness of the cylinder presented by [32] is defined as:

$$\hat{\theta}(R) = \frac{T(R) - T_i}{T_i - T_o} = -\frac{1}{4}\hat{s}R^2 + \frac{\rho \ln(R)}{4}K_c + L_c \tag{69}$$

where R is a normalized radius defined as the ratio between the radial coordinate r, and the inner radius r_i, i.e. $R = r/r_i$. R lies within the interval $[1, \rho]$, where ρ is defined as r_o/r_i with r_o as the outer radius. The term $\hat{s}$ in Equation (69) is a normalized heat generation parameter, defined as

$$\hat{s} = \frac{s r_i^2}{\alpha(T_i - T_o)} \tag{70}$$

where T_i and T_o are the temperature at the inner and outer surfaces respectively. Also, the terms K_c and L_c from Equation (69) are given by:

$$K_c = \frac{2Bi_o\hat{s} - Bi_iBi_o\hat{s} + 2Bi_i\hat{s}\rho - 4Bi_iBi_o + Bi_iBi_o\hat{s}\rho^2}{Bi_iBi_o\rho\ln(\rho) + Bi_i + Bi_o\rho} \tag{71}$$

$$L_c = \frac{1}{4}\frac{2\hat{s}\rho^2 - 4Bi_o\rho + Bi_o\hat{s}\rho^3 - 2Bi_o\hat{s}\rho\ln(\rho) + Bi_iBi_o\hat{s}\rho\ln(\rho) - 2\hat{s} + Bi_i\hat{s}}{Bi_iBi_o\rho\ln(\rho) + Bi_i + Bi_o\rho} \tag{72}$$

where Bi_i and Bi_o are the Biot numbers at the inner and outer surfaces, defined as follows:

$$Bi_i = \frac{h_{ci}r_i}{\alpha} \tag{73}$$

$$Bi_o = \frac{h_{co}r_i}{\alpha} \tag{74}$$

where h_{ci} and h_{co} are the convection coefficients at the inner and outer surfaces, respectively.

The analyses were computed using Abaqus 6.12, and were performed using a computer containing a compute node with an Intel(R) Xeon(R) CPU E5-2650 v2 processor @ 2.60 GHz and 377.80 GB of RAM. User element subroutines were implemented to perform the analyses. For the CFEM case, the user element used the standard complex type of the Fortran language and for the QFEM and OFEM user element subroutines, two libraries based on operator overloading were developed and used as described in section 3.

4.1.1. Results

The cylinder was modeled using a QFEM 8-noded 32-DOF axisymmetric element. The non-dimensional parameters were $\rho = 1.5$, $Bi_i = 0.1$, $Bi_o = 1$, $T_o/T_i = 0.7$ and $\hat{s} = 5$. The finite element model contained a total of 20 32-noded (8 real and 24 imaginary) axisymmetric elements. A schematic of the problem and the finite element mesh are shown in Fig. 6.

Fig. 7 shows the comparison of QFEM against the values obtained using the analytical solution for the normalized temperature, $\hat{\theta}$, and the three requested derivatives, $\partial\hat{\theta}/\partial h_{ci}$, $\partial\hat{\theta}/\partial h_{co}$ and $\partial\hat{\theta}/\partial T_i$. When using octonions, in which seven imaginary directions are available, seven sensitivities were computed in a single OFEM analysis. In this case, additional to the QFEM sensitivities, four extra derivatives, $\partial\hat{\theta}/\partial T_o$, $\partial\hat{\theta}/\partial r_o$, $\partial\hat{\theta}/\partial\hat{s}$ and $\partial\hat{\theta}/\partial\alpha$, were obtained. The sensitivity results are the same for complex, quaternion or octonion FE analyses.

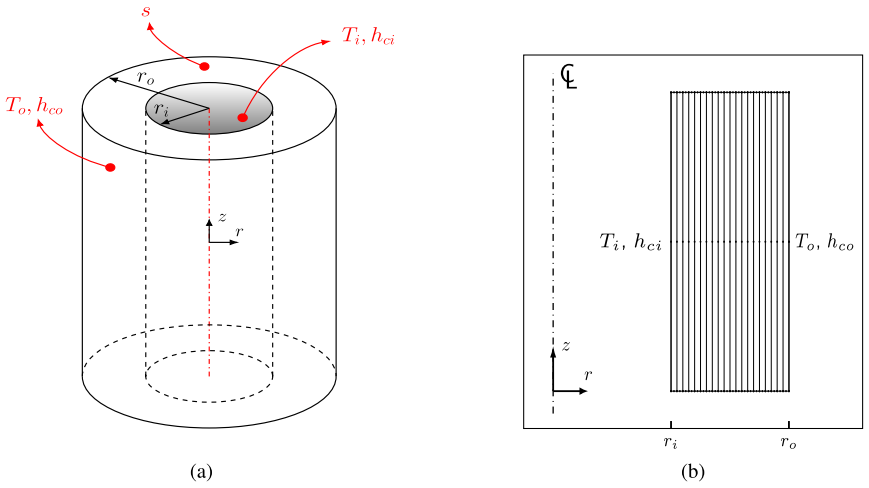


Fig. 6. (a) Problem schematic (b) Finite element mesh for axisymmetric model.

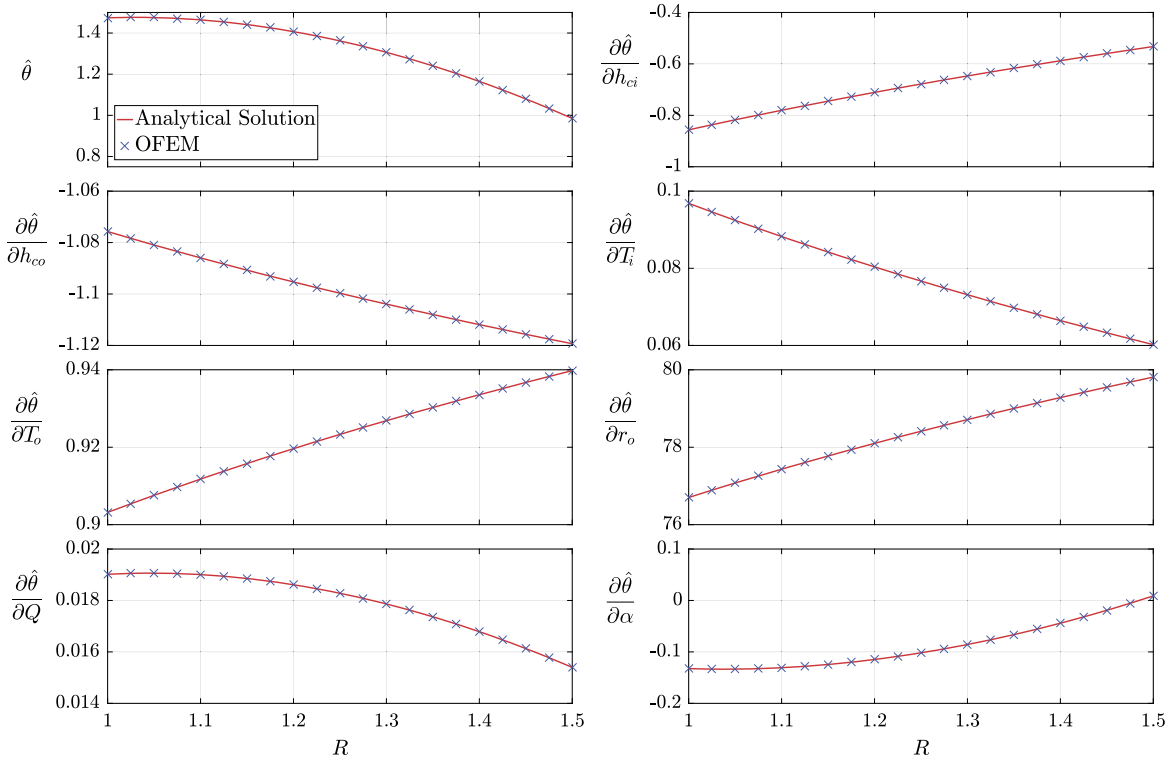


Fig. 7. CFEM, QFEM and OFEM dimensionless nodal temperature sensitivities with respect to h_{ci}, h_{co}, T_i, T_o, r_0, $\hat{s}$ and α.

4.2. Finite element Abaqus example: cantilever beam

The purpose of this analysis is to compare the performance of the methods for solving the system of equations, direct Cauchy-Riemann versus block-solver approach. Therefore, a two dimensional elastic analysis was performed for a cantilever beam fixed at the left end and subjected to body forces defined by accelerations in the x and y directions and a contact force at the unsupported end as shown in Fig. 8.

A set of CFEM, QFEM and OFEM analyses was performed in order to compute one, three and seven derivatives respectively, with different meshes ranging from 320 to 1,310,720 elements. The following derivatives were computed: Derivative of the displacement vector field $\mathbf{u}$ (u_x and u_y) with respect to: the modulus of elasticity E, the Poisson ratio ν, the accelerations a_x and a_y, the applied load P, the height H and the length L of the beam. All derivatives were computed using a step size $h = 10^{-17}$ for all algebras.

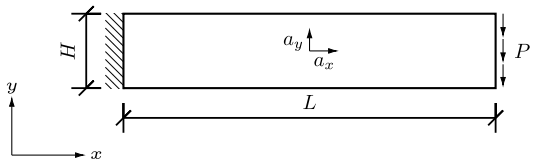


Fig. 8. Sketch of a cantilever beam anchored at the left end with a vertical load at the right end.

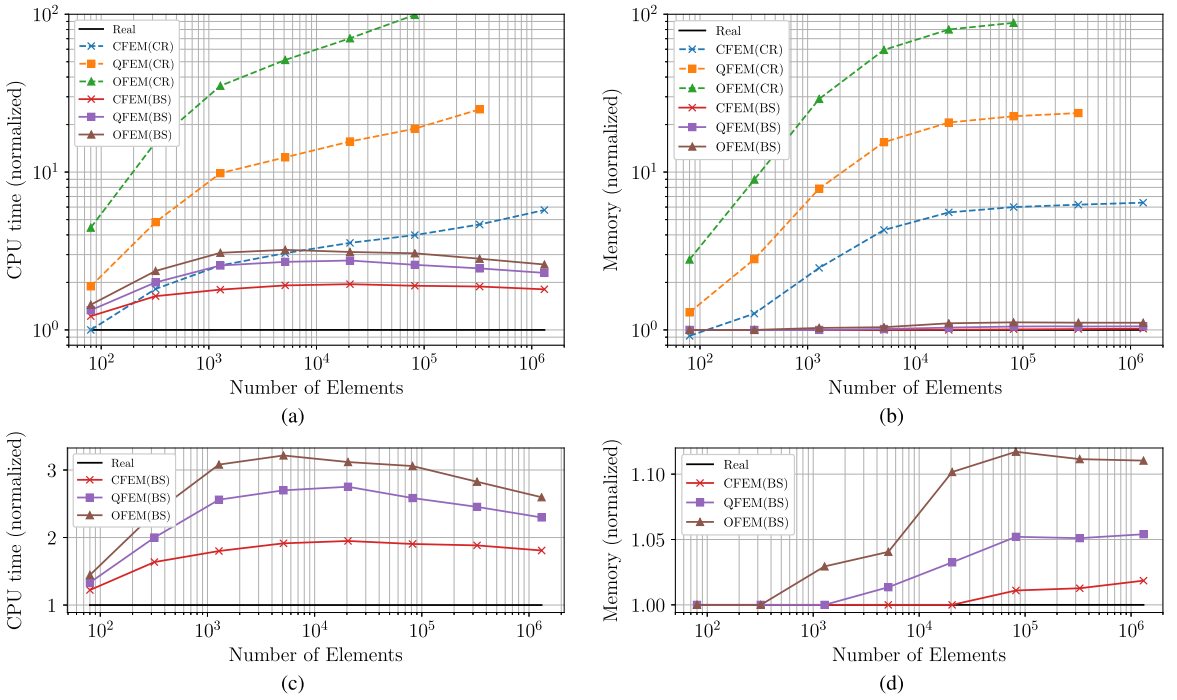


Fig. 9. Performance results of the linear elastic problem with different mesh densities showing (a) CPU time and (b) memory normalized with respect to real analysis of CFEM (1 derivative), QFEM (3 derivatives) and OFEM (7 derivatives) approaches using both the direct Cauchy Riemann (CR) approach and the block-solver (BS) method. A closeup is presented showing the normalized (c) CPU time and (d) memory of the block-solver approach only.

The analyses were computed using Abaqus 6.12, and were performed using a computer containing a compute node with an Intel(R) Xeon(R) CPU E5-2650 v2 processor @ 2.60 GHz and 377.80 GB of RAM. User element subroutines were implemented to perform the analyses. For the CFEM case, the user element used the standard complex type of the Fortran language and for the QFEM and OFEM user element subroutines, two libraries based on operator overloading were developed and used as described in section 3.

4.2.1. Results

The performance results for both CPU time and memory consumption are summarized in Fig. 9. The results are normalized with respect to the CPU and memory consumption of a conventional real analysis for the same mesh, with no derivative computation. The direct Cauchy-Riemann (CR) solution was both the most time and memory consuming approach. The CR approach constantly increased the compute time relative to a real-only analysis as the number of degrees-of-freedom is increased. For example, an OFEM analysis using a CR approach may take longer than 100× the CPU time and 100× the memory of a real-only solution.

The Block Solver (BS) approach, on the other hand, is far more efficient than the CR method. To compute one derivative, CFEM using the BS approach used the least computational resources of up to 2× the CPU time and up to 4% extra memory compared to a real-only analysis. To compute three derivatives, QFEM used up to 2.8× the CPU time and 6% of the memory of a real-only analysis. To compute seven derivatives, OFEM used up to 3.3× the CPU time and 13% of the memory of a real-only analysis (see Fig. 9c and 9d).

Table 8 summarizes the computational expense relative to a real-only analysis on a per derivative basis. Using CFEM, computing one derivative takes up to 95% of the time of the real solution. Using QFEM, the derivatives are computed using only 58% the time of the real solution and using OFEM the derivatives are computed using only 32% the time of the real solution. OFEM is most efficient on a per derivative basis because the cost of solving the real system is amortized across a larger number of derivatives.

Table 8
CPU time overhead per derivative (1 for CFEM, 3 for QFEM and 7 for OFEM) normalized to a real analysis.

Number of elements	Block-solver			Cauchy-Riemann		
	CFEM	QFEM	OFEM	CFEM	QFEM	OFEM
320	0.64	0.33	0.20	0.82	1.27	2.07
1280	0.80	0.52	0.30	1.56	2.94	4.89
5120	0.91	0.56	0.32	2.07	3.80	7.18
20480	0.95	0.58	0.30	2.56	4.87	9.91
81920	0.90	0.53	0.29	2.99	5.94	14.00
327680	0.88	0.49	0.26	3.65	7.99	N/A
1310720	0.81	0.43	0.23	4.75	N/A	N/A

Table 9
Computational cost relative to a real-only analysis required to compute derivatives using a block-solver solution scheme for a mesh of 81920 elements.

Number of derivatives	Number of analysis required				Relative computational cost			
	CFEM	QFEM	OFEM	Optimal	CFEM	QFEM	OFEM	Optimal
1	1C	1Q	1O	1C	1.90	2.59	3.03	1.90
2	2C	1Q	1O	1Q	3.80	2.59	3.03	2.59
3	3C	1Q	1O	1Q	5.70	2.59	3.03	2.59
4	4C	2Q	1O	1O	7.60	5.18	3.03	3.03
5	5C	2Q	1O	1O	9.50	5.18	3.03	3.03
6	6C	2Q	1O	1O	11.40	5.18	3.03	3.03
7	7C	3Q	1O	1O	13.30	7.77	3.03	3.03
8	8C	3Q	2O	1C+1O	15.20	7.77	6.06	4.93
9	9C	3Q	2O	1Q+1O	17.10	7.77	6.06	5.62
10	10C	4Q	2O	1Q+1O	19.00	10.36	6.06	5.62
11	11C	4Q	2O	2O	20.90	10.36	6.06	6.06
12	12C	4Q	2O	2O	22.80	10.36	6.06	6.06
13	13C	5Q	2O	2O	24.70	12.95	6.06	6.06
14	14C	5Q	2O	2O	26.60	12.95	6.06	6.06

In Table 9, the number of Complex (C), Quaternion (Q) and Octonion (O) finite element analyses required to compute up to fourteen derivatives and the relative computational expense compared to a real-only analysis is shown for each case. The results were obtained for a mesh containing a total of 81920 elements. For multiple derivatives, CFEM is the most inefficient because only one derivative can be computed per complex analysis. However, if only one derivative is required by the user, a CFEM analysis is most efficient. QFEM is the most efficient method to compute 2 or 3 derivatives, whereas OFEM is most efficient if 4-7 derivatives are required.

For more than seven derivatives, a combination of analyses is required. For example, to compute ten derivatives, one can use 10 CFEM, 4 QFEM, or 2 OFEM analyses; however, the optimal approach is to use 1 QFEM and 1 OFEM. In that case, the run time will be five times longer than a real-only analysis. To compute eleven derivatives, the optimal approach is to use 2 OFEM analyses whereas only 11 are requested. The column marked optimal shows the combination of analyses that can compute the required number of derivatives in minimum computer time.

Fig. 10 shows the most efficient combination of analyses to compute up to 28 derivatives for the same FE mesh. The most efficient sequence of analyses is the one that uses the most number of full OFEM analyses, this is, computing 7 derivatives per run. Then, the remaining derivatives are computed using an extra CFEM analysis if one more derivative is required, an extra QFEM analysis if 2-3 derivatives are required or an extra OFEM analysis if 4-7 derivatives are required. For example, to compute 23 derivatives, three OFEM analyses compute 21 derivatives and then the remaining 2 derivatives are computed using a QFEM analysis.

5. Discussion

The complex Taylor series expansion method (CTSE) was extended to Cayley-Dickson algebras (CDTSE) in order to compute multiple first order derivatives in a single analysis. In particular, quaternion (3 derivatives) and octonion (7 derivatives) - based examples were demonstrated. It is well known that a property is lost in each doubling of the algebra, commutativity is lost with quaternion algebra, associativity is lost with octonion, etc. However, the loss of these properties does not affect CDTSE in that for small step sizes the effect of the loss of properties is contained in the higher order terms and these terms are driven to below machine precision through the use of a sufficiently small step size.

Although the method is analogous to CTSE, the truncation error for the derivative is of order h, not h^2, as in CTSE. This is clearly seen in Fig. 2b. The implications are that a smaller step size is needed for QTSE or OTSE than CTSE. However there are no numerical implications in practice as the truncation error can be driven below machine precision in all cases through the use of a sufficiently small step size.

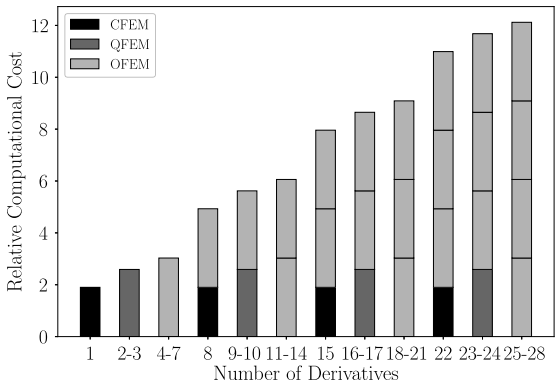


Fig. 10. Optimal selection of analyses type based on the required number of derivatives for a mesh of 81920 elements.

Fig. 2b also shows that the relative error for octonions is slightly higher than quaternions, however preserving the same convergence rate. This is explained by the increase in the number of operations octonions will perform in comparison with quaternions. Sedenions will have a larger truncation error than octonions, etc. However, this is an academic topic since the truncation error can be driven below machine precision in practice because reducing the step size h does not generate subtraction cancellation errors.

Both the direct Cauchy-Riemann and block-solver approaches solved for the same value and number of outputs. The block-solver method is a better approach in terms of its computational performance because it required several orders of magnitude less CPU time and memory than the direct approach. The block solver method lies within the same order of magnitude of CPU performance metrics as the conventional real analysis, even though the number of derivatives increased up to a number of 7 in the OFEM case. The direct method, however, was easier to implement since it was all performed within one step of analysis. However, the paradigm of the implementation in a commercial software such as Abaqus is still open, since the multiple right-hand-side solvers are usually available only to linear analyses.

Using the block-solver method, QFEM shows a better performance per derivative than CFEM. QFEM is at least $1.54\times$ faster per derivative than CFEM. In the case of OFEM, it is at least $1.71\times$ faster per derivative than QFEM. However, the total CPU time of QFEM is longer than CFEM. Similarly, the total CPU time of OFEM is slower than QFEM. It is therefore more efficient to use CFEM to compute one derivative, QFEM to compute 2-3 derivatives and OFEM to compute 4-7. It is expected that higher dimensional algebras, e.g. sedenions, will be more efficient for larger number of derivatives, than multiple CFEM, QFEM or OFEM analyses.

The Taylor series method for computing functions of quaternions and octonions is a convenient tool because the required operations are the function and its real-derivative evaluated at the real value. Hence, a formal comprehensive method of computing functions of quaternions and octonions is not required.

The proposed methodology is far superior in run times and memory consumption than equivalent methods using multi-complex or multi-dual implementations. However, CDTSE is limited to first order derivatives, unlike multi-complex or multi-dual implementations which can compute first, second, or higher order derivatives.

6. Conclusions

The complex Taylor series expansion (CTSE) method has been extended to Cayley-Dickson algebras (CDTSE) in order to compute multiple first order derivatives. In particular, quaternion algebra can compute 3 first order derivatives, octonion algebra can compute 7, and sedenion algebra can compute 15. This strategy can be extended to any number of first order derivatives in powers of 2.

Similarly to CTSE, CDTSE does not require subtraction of terms in order to compute a derivative. As a result, the truncation error can be driven below machine precision through the use of a very small step size. Contrary to CTSE, the error for CDTSE is of order h whereas for CTSE the error is of order h^2. However, this difference in error is of no practical importance once a sufficiently small h is used for numerical computations.

Quaternion and octonion algebras were implemented within the commercial finite element code Abaqus as a user element in an analogous manner as CTSE. The key elements were to a) define a set of imaginary nodes added to each real node, b) use the coordinates of the imaginary nodes to define the perturbation in the geometry, c) compute the element stiffness matrix using mathematical operations that support the Cayley-Dickson algebra (quaternion, octonion, etc.), d) convert the stiffness matrix to all real values before returning the results to Abaqus, e) solve the system of equations using the Abaqus built-in direct solver or use the block solver approach, and f) post process the derivatives of the degrees-of-freedom to obtain auxiliary results.

The conversion of a traditional real valued finite element routine to quaternion and octonion algebra was facilitated through the definition of quaternion and octonion types, implementing operator overloading of addition/subtraction, multiplication/division, and the implementation of an approximate but accurate method for computing functions of quaternion-

s/octonions using a truncated Taylor series. The truncated Taylor series approach was shown to have the same accuracy as the traditional quaternion/octonion functions but it was simpler to program and faster to execute.

The accuracy, versatility and performance of CDTSE in the finite element method was shown in two numerical examples implemented in Abaqus through the use of user element subroutines. The use of QFEM and OFEM user elements allowed one to compute accurate three and seven first order derivatives respectively in heat transfer and linear elasticity problems. Two solution methods were discussed and compared: a direct Cauchy Riemann and a block solver approach. The block solver approach was significantly more efficient than the direct approach, although both resulted in the same output values.

Acknowledgements

This work was supported by the Department of Defense through grant No. W911NF-15-1-0456 and by Departamento Administrativo de Ciencia, Tecnología e Innovación Colciencias through the PhD. scholarships 614-2013 and 647-2014.

Appendix A. Quaternions

Quaternion numbers, $\mathbb{H}$, are a subset of hypercomplex numbers. They extend the complex numbers to three imaginary units. A number q^* is a quaternion when

$$q^* = q_{Re} + q_i i + q_j j + q_k k \tag{A.1}$$

where q_{Re}, q_i, q_j and q_k are real coefficients and i, j, k are imaginary units governed by the following conditions

$$i^2 = j^2 = k^2 = ijk = -1 \tag{A.2}$$

$$ij = -ji = k \tag{A.3}$$

$$jk = -kj = i \tag{A.4}$$

$$ki = -ik = j \tag{A.5}$$

A.1. Algebra

Quaternion addition is defined by the addition of real coefficients of the same imaginary units, whereas product operation is defined by imaginary conditions in Equation (A.2). Assume two quaternion numbers q^* and p^* are defined as follows:

$$\begin{aligned} q^* &= q_{Re} + q_i i + q_j j + q_k k \\ p^* &= p_{Re} + p_i i + p_j j + p_k k \end{aligned} \tag{A.6}$$

Addition of p^* and q^* is calculated as:

$$q^* + p^* = (q_{Re} + p_{Re}) + (q_i + p_i)i + (q_j + p_j)j + (q_k + p_k)k \tag{A.7}$$

and the product of q^* and p^* results in

$$\begin{aligned} q^* p^* = &(q_{Re}p_{Re} - q_i p_i - q_j p_j - q_k p_k) + \\ &(q_{Re}p_i + q_i p_{Re} + q_j p_k - q_k p_j)i + \\ &(q_{Re}p_j - q_i p_k + q_j p_{Re} + q_k p_i)j + \\ &(q_{Re}p_k + q_i p_j - q_j p_i + q_k p_{Re})k \end{aligned} \tag{A.8}$$

As seen in Equation (A.8), p^*q^* differs from q^*p^*. Therefore, this algebra is non-commutative.

Multiplication of a quaternion by a scalar, i.e. a quaternion with zero imaginary coefficients, is defined as

$$q^*\alpha = \alpha q^* = \alpha q_{Re} + \alpha q_i i + \alpha q_j j + \alpha q_k k \tag{A.9}$$

The conjugate $\bar{q^*}$ of a Quaternion number q^* is

$$\bar{q^*} = q_{Re} - q_i i - q_j j - q_k k \tag{A.10}$$

such that multiplication of $\bar{q^*}q^*$ eliminates all imaginary directions

$$\bar{q^*}q^* = q^*\bar{q^*} = q_{Re}^2 + q_i^2 + q_j^2 + q_k^2 \tag{A.11}$$

Consider a quaternion number $p^* \in \mathbb{H}$ with its conjugate as defined in Equation (A.10). The norm of p^* is defined as

$$||p^*|| = \sqrt{p^*\bar{p^*}} \tag{A.12}$$

As indicated in [26], Quaternion is a division algebra, because an inverse operation can be defined:

$$\frac{1}{p^*} = p^{*-1} = \frac{\bar{p^*}}{||p^*||^2} \tag{A.13}$$

Therefore, division is performed as

$$\frac{q^*}{p^*} = q^* \frac{1}{p^*} = \frac{q^* \bar{p^*}}{||p^*||^2} \tag{A.14}$$

A.2. Matrix and vector representation

Quaternion algebra is isomorphic to matrix algebra. This implies that quaternion numbers can be transformed to an equivalent matrix such that operations like addition and multiplication are equivalent to its corresponding quaternion algebra operations. In general, given the transformation $\mathbf{T}(\cdot)$ from quaternion to matrix space and two quaternion numbers q^* and p^* as Equation (A.6), then

$$\mathbf{T}(q^* + p^*) = \mathbf{T}(q^*) + \mathbf{T}(p^*) \tag{A.15}$$

$$\mathbf{T}(q^* p^*) = \mathbf{T}(q^*)\mathbf{T}(p^*) \tag{A.16}$$

Transformation $\mathbf{T}(\cdot)$ is not unique; however, there is no preferential form if quaternions are used for computation of derivatives. A possible matrix form of a quaternion q^* as in Equation (A.1) is the following:

$$\mathbf{T}(q^*) = \begin{bmatrix} q_{\mathrm{Re}} & -q_i & -q_j & -q_k \\ q_i & q_{\mathrm{Re}} & -q_k & q_j \\ q_j & q_k & q_{\mathrm{Re}} & -q_i \\ q_k & -q_j & q_i & q_{\mathrm{Re}} \end{bmatrix} \tag{A.17}$$

Notice the transformation generates an asymmetric matrix composed by real coefficients. Therefore, this transformation allows one to use real-only algebra with no imaginary concepts involved. This can be useful in a computational implementation.

A vector form also exists for quaternions. For quaternion q^*, its vector form is

$$\mathbf{t}(q^*) = \begin{bmatrix} q_{\mathrm{Re}} & q_i & q_j & q_k \end{bmatrix}^T \tag{A.18}$$

The transformation from quaternion to vector space, $\mathbf{t}(\cdot)$, relates to quaternion operations as follows

$$\mathbf{t}(q^* + p^*) = \mathbf{t}(q^*) + \mathbf{t}(p^*) \tag{A.19}$$

$$\mathbf{t}(q^* p^*) = \mathbf{T}(q^*)\,\mathbf{t}(p^*) \tag{A.20}$$

Notice that only transformation of the multiplication operation requires the matrix form of the multiplier (left term), meanwhile the multiplicand (right term) is in its vector form. Moreover, addition may be done using the corresponding vector forms. The advantage of using matrix and vector representation is that arithmetic operations can be computed using linear algebra libraries.

In summary, quaternion algebra can be represented by vector and matrix forms. As shown in Equations (A.19) and (A.20), quaternion addition can be computed using the vector form, and multiplication can be computed using a matrix-vector product.

Appendix B. Octonions

Also known as Cayley numbers [26], octonions are also a subset of hypercomplex numbers. They are considered a doubling of the algebra of quaternions, because the total number of directions (including real direction) is doubled. They extend complex numbers to a total of 7 imaginary units. A number o^* is an octonion when

$$o^* = o_{Re} + o_i i + o_j j + o_k k + o_{e'} e' + o_{i'} i' + o_{j'} j' + o_{k'} k' \tag{B.1}$$

where o_{Re}, o_i, o_j, o_k, $o_{e'}$, $o_{i'}$, $o_{j'}$ and $o_{k'}$ are real coefficients and i, j, k, e', i', j', k' are imaginary units with imaginary conditions summarized by the multiplication table shown in Table 10. According to Table 10, the result of multiplying, e.g. k times j' is given in the corresponding k-row and j'-column, i.e. $kj' = i'$.

Table 10
Multiplication table of Octonion imaginary units.

		Right factor							
		1	i	j	k	e'	i'	j'	k'
Left factor	1	1	i	j	k	e'	i'	j'	k'
	i	i	-1	k	$-j$	i'	$-e'$	$-k'$	j'
	j	j	$-k$	-1	i	j'	k'	$-e'$	i'
	k	k	j	$-i$	-1	k'	$-j'$	i'	$-e'$
	e'	e'	$-i'$	$-j'$	$-k'$	-1	i	j	k
	i'	i'	e'	$-k'$	j'	$-i$	-1	$-k$	j
	j'	j'	k'	e'	$-i'$	$-j$	k	-1	$-i$
	k'	k'	$-j'$	i'	e'	$-k$	$-j$	i	-1

B.1. Algebra

Octonion addition is similar to quaternion addition. Assume two octonion numbers o^* and r^*,

$$o^* = o_{Re} + o_i i + o_j j + o_k k + o_{e'} e' + o_{i'} i' + o_{j'} j' + o_{k'} k' \tag{B.2}$$

$$r^* = r_{Re} + r_i i + r_j j + r_k k + r_{e'} e' + r_{i'} i' + r_{j'} j' + r_{k'} k' \tag{B.3}$$

addition of o^* and r^* results in

$$\begin{aligned} o^* + r^* = {} & (o_{Re} + r_{Re}) + (o_i + r_i) i + (o_j + r_j) j + (o_k + r_k) k + \\ & (o_{e'} + r_{e'}) e' + (o_{i'} + r_{i'}) i' + (o_{j'} + r_{j'}) j' + (o_{k'} + r_{k'}) k' \end{aligned} \tag{B.4}$$

Multiplication of octonions is performed as follows:

$$\begin{aligned} o^* r^* = {} & (o_{Re} r_{Re} - o_i r_i - o_j r_j - o_k r_k - o_{e'} r_{e'} - o_{i'} r_{i'} - o_{j'} r_{j'} - o_{k'} r_{k'}) + \\ & (o_{Re} r_i + o_i r_{Re} - o_j r_k + o_k r_j - o_{e'} r_{i'} + o_{i'} r_{e'} + o_{j'} r_{k'} - o_{k'} r_{j'}) i + \\ & (o_{Re} r_j + o_i r_k + o_j r_{Re} - o_k r_i - o_{e'} r_{j'} - o_{i'} r_{k'} + o_{j'} r_{e'} + o_{k'} r_{i'}) j + \\ & (o_{Re} r_k - o_i r_j + o_j r_i + o_k r_{Re} - o_{e'} r_{k'} + o_{i'} r_{j'} - o_{j'} r_{i'} + o_{k'} r_{e'}) k + \\ & (o_{Re} r_{e'} + o_i r_{i'} + o_j r_{j'} + o_k r_{k'} + o_{e'} r_{Re} - o_{i'} r_i - o_{j'} r_j - o_{k'} r_k) e' + \\ & (o_{Re} r_{i'} - o_i r_{e'} + o_j r_{k'} - o_k r_{j'} + o_{e'} r_i + o_{i'} r_{Re} + o_{j'} r_k - o_{k'} r_j) i' + \\ & (o_{Re} r_{j'} - o_i r_{k'} - o_j r_{e'} + o_k r_{i'} + o_{e'} r_j - o_{i'} r_k + o_{j'} r_{Re} + o_{k'} r_i) j' + \\ & (o_{Re} r_{k'} + o_i r_{j'} - o_j r_{i'} - o_k r_{e'} + o_{e'} r_k + o_{i'} r_j - o_{j'} r_i + o_{k'} r_{Re}) k' \end{aligned} \tag{B.5}$$

Multiplication by a scalar real number α results in

$$o^* \alpha = \alpha o^* = \alpha o_{Re} + \alpha o_i i + \alpha o_j j + \alpha o_k k + \alpha o_{e'} e' + \alpha o_{i'} i' + \alpha o_{j'} j' + \alpha o_{k'} k' \tag{B.6}$$

Besides being non-commutative, octonion multiplication is non-associative. Consider the triplet ije', where the result varies according to the order of associative operations:

$$(ij)e' = ke' = k' \tag{B.7}$$

meanwhile

$$i(je') = ij' = -k' \tag{B.8}$$

$$\therefore (ij)e' \neq i(je') \tag{B.9}$$

The conjugate $\bar{o^*}$ of an octonion number o^* is

$$\bar{o^*} = o_0 - o_i i - o_j j - o_k k - o_{e'} e' - o_{i'} i' - o_{j'} j' - o_{k'} k' \tag{B.10}$$

such that multiplication $\bar{o^*} o^*$ eliminates all imaginary directions

$$\bar{o^*} o^* = o^* \bar{o^*} = o_{Re}^2 + o_i^2 + o_j^2 + o_k^2 + o_{e'}^2 + o_{i'}^2 + o_{j'}^2 + o_{k'}^2 \tag{B.11}$$

Further algebraic operations are described in section 2.3.

B.2. Matrix and vector representation

Matrix representation of an octonion number (see [33]) is given by

$$\mathbf{T}(o^*) = \begin{bmatrix} o_{Re} & -o_i & -o_j & -o_k & -o_{e'} & -o_{i'} & -o_{j'} & -o_{k'} \\ o_i & o_{Re} & -o_k & o_j & -o_{i'} & o_{e'} & o_{k'} & -o_{j'} \\ o_j & o_k & o_{Re} & -o_i & -o_{j'} & -o_{k'} & o_{e'} & o_{i'} \\ o_k & -o_j & o_i & o_{Re} & -o_{k'} & o_{j'} & -o_{i'} & o_{e'} \\ o_{e'} & o_{i'} & o_{j'} & o_{k'} & o_{Re} & -o_i & -o_j & -o_k \\ o_{i'} & -o_{e'} & o_{k'} & -o_{j'} & o_i & o_{Re} & o_k & -o_j \\ o_{j'} & -o_{k'} & -o_{e'} & o_{i'} & o_j & -o_k & o_{Re} & o_i \\ o_{k'} & o_{j'} & -o_{i'} & -o_{e'} & o_k & o_j & -o_i & o_{Re} \end{bmatrix} \tag{B.12}$$

and its vector representation is

$$\mathbf{t}(o^*) = \begin{bmatrix} o_{Re} & o_i & o_j & o_k & o_{e'} & o_{i'} & o_{j'} & o_{k'} \end{bmatrix}^T \tag{B.13}$$

Addition and multiplication of octonions can be performed using vector and matrix forms analogously to quaternions, see Equations (A.19) and (A.20).

Appendix C. Quaternion Taylor series expansion

The aim of this section is to derive the quaternion Taylor Series Expansion for computing derivatives, shown in Equation (5), and to show that the convergence of the approximated derivatives is linear with respect to the step size. Consider a quaternion function, $\mathcal{F}: \mathbb{H}^3 \rightarrow \mathbb{H}$, that depends on three quaternion inputs x^*, y^* and z^*. From a real function perspective, $\mathcal{F}$ is formed by four real functions,

$$\mathcal{F}(x^*, y^*, z^*) = f_{\mathrm{r}}(\mathbf{x}) + f_i(\mathbf{x})i + f_j(\mathbf{x})j + f_k(\mathbf{x})k \tag{C.1}$$

where, $x^* = x_{\mathrm{r}} + x_i i + x_j j + x_k k$, $y^* = y_{\mathrm{r}} + y_i i + y_j j + y_k k$, $z^* = z_{\mathrm{r}} + z_i i + z_j j + z_k k$, $\mathbf{x}$ is a vector formed by all real coefficients of the quaternion inputs of $\mathcal{F}$, namely

$$\mathbf{x} = \begin{bmatrix} x_{\mathrm{r}} & x_i & x_j & x_k & y_{\mathrm{r}} & y_i & y_j & y_k & z_{\mathrm{r}} & z_i & z_j & z_k \end{bmatrix}^T \tag{C.2}$$

where the number of real input parameters corresponds to four times the number of input quaternion variables of the function $\mathcal{F}$. Also, $f_{\mathrm{r}}(\mathbf{x})$, $f_i(\mathbf{x})$, $f_j(\mathbf{x})$ and $f_k(\mathbf{x})$ are real functions that generate the real, i'th, j'th and k'th components of $\mathcal{F}$, respectively. It is considered that the function $\mathcal{F}$ is isomorphic to a real function f, i.e., when evaluated at $x^* = x + 0i + 0j + 0k$, $y^* = y + 0i + 0j + 0k$ and $z^* = z + 0i + 0j + 0k$, the function behaves as

$$\mathcal{F}(x, y, z) = f(x, y, z) + 0i + 0j + 0k \tag{C.3}$$

The derivative of a quaternion function with respect to a real parameter t is given by

$$\mathcal{F}_{,t}(t) = f_{\mathrm{r},t}(t) + f_{i,t}(t)i + f_{j,t}(t)j + f_{k,t}(t)k \tag{C.4}$$

where $\mathcal{F}_{,t}(t)$ is $d\mathcal{F}/dt$, $f_{\mathrm{r},t}(t)$ is df_{r}/dt, etc.

The Taylor series expansion of the quaternion function can be achieved by generating the Taylor series expansions of its four real functions with respect to all real variables in $\mathbf{x}$,

$$f_{\mathrm{r}}(\mathbf{x}_0 + \mathbf{h}) = f_{\mathrm{r}}(\mathbf{x}_0) + \nabla f_{\mathrm{r}}(\mathbf{x}_0)^T \mathbf{h} + \frac{1}{2}\mathbf{h}^T \mathbf{H}_{\mathrm{r}}(\mathbf{x}_0)\mathbf{h} + \mathbf{H.O.T.} \tag{C.5}$$

$$f_i(\mathbf{x}_0 + \mathbf{h}) = f_i(\mathbf{x}_0) + \nabla f_i(\mathbf{x}_0)^T \mathbf{h} + \frac{1}{2}\mathbf{h}^T \mathbf{H}_i(\mathbf{x}_0)\mathbf{h} + \mathbf{H.O.T.} \tag{C.6}$$

$$f_j(\mathbf{x}_0 + \mathbf{h}) = f_j(\mathbf{x}_0) + \nabla f_j(\mathbf{x}_0)^T \mathbf{h} + \frac{1}{2}\mathbf{h}^T \mathbf{H}_j(\mathbf{x}_0)\mathbf{h} + \mathbf{H.O.T.} \tag{C.7}$$

$$f_k(\mathbf{x}_0 + \mathbf{h}) = f_k(\mathbf{x}_0) + \nabla f_k(\mathbf{x}_0)^T \mathbf{h} + \frac{1}{2}\mathbf{h}^T \mathbf{H}_k(\mathbf{x}_0)\mathbf{h} + \mathbf{H.O.T.} \tag{C.8}$$

where $\mathbf{h}$ contains the perturbations of all variables in $\mathbf{x}$,

$$\mathbf{h} = \begin{bmatrix} h_{x_{\mathrm{r}}} & h_{x_i} & h_{x_j} & h_{x_k} & h_{y_{\mathrm{r}}} & h_{y_i} & h_{y_j} & h_{y_k} & h_{z_{\mathrm{r}}} & h_{z_i} & h_{z_j} & h_{z_k} \end{bmatrix}^T \tag{C.9}$$

$\mathbf{x}_0$ is a vector containing the center or reference points of all Taylor series expansions

$$\mathbf{x}_0 = \begin{bmatrix} x_{0\mathrm{r}} & x_{0i} & x_{0j} & x_{0k} & y_{0\mathrm{r}} & y_{0i} & y_{0j} & y_{0k} & z_{0\mathrm{r}} & z_{0i} & z_{0j} & z_{0k} \end{bmatrix}^T \tag{C.10}$$

the matrices $\mathbf{H}_{\mathrm{r}}(\mathbf{x})$, $\mathbf{H}_i(\mathbf{x})$, $\mathbf{H}_j(\mathbf{x})$ and $\mathbf{H}_k(\mathbf{x})$ are the Hessian matrices of the real functions f_{r}, f_i, f_j and f_k respectively, with respect to all parameters in $\mathbf{x}$, and the term **H.O.T.** contains the higher order terms from the Taylor series.

Using quaternion notation, Equations (C.5)–(C.8) can be simplified in the following manner

$$\mathcal{F}(x_0^* + \Delta x^*, y_0^* + \Delta y^*, z_0^* + \Delta z^*) = \mathcal{F}(x_0^*, y_0^*, z_0^*) + \nabla\mathcal{F}(x_0^*, y_0^*, z_0^*)^T\mathbf{h} + \frac{1}{2}\mathbf{h}^T\boldsymbol{\mathcal{H}}(x_0^*, y_0^*, z_0^*)\mathbf{h} + \mathbf{H.O.T.} \tag{C.11}$$

where $\boldsymbol{\mathcal{H}}$ contains the Hessian matrices of the functions in $\mathcal{F}$

$$\boldsymbol{\mathcal{H}}(x_0^*, y_0^*, z_0^*) = \mathbf{H}_{\mathrm{r}}(\mathbf{x}_0) + \mathbf{H}_i(\mathbf{x}_0)i + \mathbf{H}_j(\mathbf{x}_0)j + \mathbf{H}_k(\mathbf{x}_0)k \tag{C.12}$$

x_0^*, y_0^*, z_0^* and Δx^*, Δy^*, Δz^*, respectively correspond to the quaternion center points and perturbations:

$$\begin{aligned} x_0^* &= x_{0\mathrm{r}} + x_{0i}i + x_{0j}j + x_{0k}k, & \Delta x^* &= h_{x_{\mathrm{r}}} + h_{x_i}i + h_{x_j}j + h_{x_k}k, \\ y_0^* &= y_{0\mathrm{r}} + y_{0i}i + y_{0j}j + y_{0k}k, & \Delta y^* &= h_{y_{\mathrm{r}}} + h_{y_i}i + h_{y_j}j + h_{y_k}k, \\ z_0^* &= z_{0\mathrm{r}} + z_{0i}i + z_{0j}j + z_{0k}k, & \Delta z^* &= h_{z_{\mathrm{r}}} + h_{z_i}i + h_{z_j}j + h_{z_k}k \end{aligned} \tag{C.13}$$

In the context of Quaternion Taylor series for computation of derivatives of real functions as described in section 2.1, the center points and perturbations are

$$\begin{aligned} x_0^* &= x_0 + 0i + 0j + 0k, & \Delta x^* &= 0 + hi + 0j + 0k, \\ y_0^* &= y_0 + 0i + 0j + 0k, & \Delta y^* &= 0 + 0i + hj + 0k, \\ z_0^* &= z_0 + 0i + 0j + 0k, & \Delta z^* &= 0 + 0i + 0j + hk \end{aligned} \tag{C.14}$$

Taking the values of Equation (C.14) and inserting them in Equation (C.11), leads to

$$\begin{aligned} \mathcal{F}(x_0 + hi, y_0 + hj, z_0 + hk) = \mathcal{F}(x_0, y_0, z_0) &+ h\left(\mathcal{F}_{,x_i} + \mathcal{F}_{,y_j} + \mathcal{F}_{,z_k}\right) \\ &+ \frac{1}{2}h^2\left(\mathcal{F}_{,x_i^2} + \mathcal{F}_{,y_j^2} + \mathcal{F}_{,z_k^2} + \mathcal{F}_{,x_iy_j} + \mathcal{F}_{,x_iz_k} + \mathcal{F}_{,y_jx_i} + \mathcal{F}_{,z_jx_i} + \mathcal{F}_{,y_jz_k} + \mathcal{F}_{,z_ky_j}\right) + \mathcal{O}(h^3) \end{aligned} \tag{C.15}$$

where $\mathcal{F}_{,\alpha}$ represents the derivative of $\mathcal{F}$ with respect to α, $\partial\mathcal{F}/\partial\alpha$. Similarly, $\mathcal{F}_{,\alpha^2}$ and $\mathcal{F}_{,\alpha\beta}$ represent the second order derivatives $\partial^2\mathcal{F}/\partial\alpha^2$ and $\partial^2\mathcal{F}/\partial\alpha\partial\beta$ respectively. All derivatives are evaluated at $(x^*, y^*, z^*) = (x_0, y_0, z_0)$. Notice that the imaginary coefficients of the evaluation points are zero, and the derivatives of $\mathcal{F}$ are with respect to real terms, thus calculated using Equation (C.4).

Since $\mathcal{F}$ is isomorphic to f (see Equation (C.3)), it can be shown that $\mathcal{F}$ and its first order derivatives $\mathcal{F}_{,x_i}$, $\mathcal{F}_{,y_j}$ and $\mathcal{F}_{,z_k}$ evaluated at (x_0, y_0, z_0) correspond to:

$$\mathcal{F}(x_0, y_0, z_0) = f(x_0, y_0, z_0) + 0i + 0j + 0k \tag{C.16}$$

$$\mathcal{F}_{,x_i}(x_0, y_0, z_0) = 0 + f_{,x}(x_0, y_0, z_0)i + 0j + 0k \tag{C.17}$$

$$\mathcal{F}_{,y_j}(x_0, y_0, z_0) = 0 + 0i + f_{,y}(x_0, y_0, z_0)j + 0k \tag{C.18}$$

$$\mathcal{F}_{,z_k}(x_0, y_0, z_0) = 0 + 0i + 0j + f_{,z}(x_0, y_0, z_0)k \tag{C.19}$$

and, similarly, the second order derivatives $\mathcal{F}_{,x_i^2}$, $\mathcal{F}_{,y_j^2}$ and $\mathcal{F}_{,z_k^2}$ are related to the second order derivatives of f as

$$\mathcal{F}_{,x_i^2}(x_0, y_0, z_0) = -f_{,x^2}(x_0, y_0, z_0) + 0i + 0j + 0k \tag{C.20}$$

$$\mathcal{F}_{,y_j^2}(x_0, y_0, z_0) = -f_{,y^2}(x_0, y_0, z_0) + 0i + 0j + 0k \tag{C.21}$$

$$\mathcal{F}_{,z_k^2}(x_0, y_0, z_0) = -f_{,z^2}(x_0, y_0, z_0) + 0i + 0j + 0k \tag{C.22}$$

Notice that the derivatives of the quaternion function $\mathcal{F}$, Equations (C.16)–(C.22), are directly related to the derivatives of its isomorphic real function f. This, however, is not the case for the second order mixed derivatives of $\mathcal{F}$ because in general, they are result of operations between terms with two different imaginary directions, thus being influenced by the distribution of terms in $\mathcal{F}$ due to the non-commutativity quaternion property. This makes the magnitude of the second order derivative dependent on the distribution and arrangement of terms in the quaternion function and therefore it may differ from the second order mixed derivative of f.

The mixed second order derivative result lays only on the direction of the product between the imaginary directions of the derivands, e.g. $\mathcal{F}_{,\alpha_k\beta_j}$ lies on i and so forth. Consequently, the general case of the mixed second order quaternion derivatives evaluated at (x_0, y_0, z_0) will be non-zero and directed along the imaginary directions i, j and k as shown below.

$$\mathcal{F}_{,z_k y_j} + \mathcal{F}_{,y_j z_k} = 0 + 2\gamma_i i + 0j + 0k \tag{C.23}$$

$$\mathcal{F}_{,x_i z_k} + \mathcal{F}_{,z_k x_i} = 0 + 0i + 2\gamma_j j + 0k \tag{C.24}$$

$$\mathcal{F}_{,x_i y_j} + \mathcal{F}_{,y_j x_i} = 0 + 0i + 0j + 2\gamma_k k \tag{C.25}$$

where γ_i, γ_j and γ_k are real functions that depend on the input coefficients $\mathbf{x}$.

Consequently, the quaternion Taylor series expansion for the applications discussed in section 2.1 is

$$\begin{aligned}\mathcal{F}(x_0 + hi, y_0 + hj, z_0 + hk) = f(x_0, y_0, z_0) - \frac{1}{2}h^2\left(f_{,x^2} + f_{,y^2} + f_{,z^2}\right) \\ + \left(hf_{,x} + h^2\gamma_i\right)i + \left(hf_{,y} + h^2\gamma_j\right)j + \left(hf_{,z} + h^2\gamma_k\right)k + O(h^3)\end{aligned} \tag{C.26}$$

and taking the real and imaginary coefficients of Equation (C.26) and then solving for the function value and its first order derivatives, the following is obtained

$$f(x_0, y_0, z_0) = \mathrm{Re}\left[\mathcal{F}(x^*, y^*, z^*)\right] + h^2\left(f_{,x^2} + f_{,y^2} + f_{,z^2}\right)/2 + \mathcal{O}(h^3) \tag{C.27}$$

$$f_{,x}(x_0, y_0, z_0) = \mathrm{Im}_i\left[\mathcal{F}(x^*, y^*, z^*)\right]/h - h\gamma_i(\mathbf{x}) + \mathcal{O}(h^2) \tag{C.28}$$

$$f_{,y}(x_0, y_0, z_0) = \mathrm{Im}_j\left[\mathcal{F}(x^*, y^*, z^*)\right]/h - h\gamma_j(\mathbf{x}) + \mathcal{O}(h^2) \tag{C.29}$$

$$f_{,z}(x_0, y_0, z_0) = \mathrm{Im}_k\left[\mathcal{F}(x^*, y^*, z^*)\right]/h - h\gamma_k(\mathbf{x}) + \mathcal{O}(h^2) \tag{C.30}$$

which shows that QTSE approximates the function quadratically, but approximates the first order derivatives linearly with respect to h.

Reorganizing terms of Equation (C.26) by first order (h) and then second order terms (h^2)

$$\begin{aligned}\mathcal{F}(x_0 + hi, y_0 + hj, z_0 + hk) = f(x_0, y_0, z_0) + h\left(f_{,x} + f_{,y} + f_{,z}\right) \\ + h^2\left(-\frac{1}{2}\left(f_{,x^2} + f_{,y^2} + f_{,z^2}\right) + \gamma_i i + \gamma_j j + \gamma_k k\right) + O(h^3)\end{aligned} \tag{C.31}$$

leads to Equation (5) by collecting terms with $\mathcal{O}(h^2)$ and higher.

This justification extrapolates to higher dimensional Cayley Dickson algebras such as octonions, sedenions, etc. The only cases where quadratic convergence is obtained in the derivative approximation are those cases whose second order mixed derivatives of the function $\mathcal{F}$ have magnitude zero when evaluated at (x_0, y_0, z_0).

Appendix D. Fortran support library

It is straightforward to develop a library to support quaternion/octonion operations needed for finite element methods hereby presented. Operator overloading is used to simplify the development and modification of current programs in order to use quaternion algebra. The library is composed by a single Fortran module that defines the type structure, functions and operator overloads. It contains zero (scalar), one (vector) and two (matrix) dimensional array support.

Scalar operations are the minimal requirement to apply quaternion and octonion algebras in the computation of derivatives. In addition, vector and matrix support is provided in order to simplify coding of finite element subroutines. The functions currently implemented in the library are listed in Table 11. Operations TOMATRIX, UNFOLD and FOLD are extrinsic functions. All other implemented operations are overloads to intrinsic Fortran functions.

D.1. Scalar type

Table 12 lists the minimal functions needed to support scalar operations of a hypercomplex algebra. The library implements operations that follow the algebra as defined in Appendix A as well as functions of quaternions using the truncated Taylor series approach. (see section 2.3).

Operator overloads are linked to the quaternion type so that manipulation of the quaternion variables are straightforward and also to reduce the modifications required to adapt current codes to quaternion variables. Overloaded operators were "+", "−", "*", "/", "**" as well as functions such as sine, cosine, etc.

A minimal version of the quaternion module is given below in order to provide the fundamental functions and operator overloads required to evaluate the example listed in section 2.2.

The Fortran source code to perform the numerical example listed in section 2.2 using quaternion numbers is given below. The code computes the derivatives of $f(a, b, c, d, x, y, z) = \sin(a^3b^2c^3d^4x^3y^2z^4)$ with respect to a, b, c, d, x, y and z using three quaternion evaluations of the function f.

Table 11
List of supported operations and functions available in the Fortran library.

Operation	Fortran operator/function
Addition	+
Multiplication	*
Division	/
Power	**
Sine	SIN
Cosine	COS
Tangent	TAN
Exponential	EXP
Logarithm	LOG
Vector dot product	DOT_PRODUCT
Matrix multiplication	MATMUL
Matrix transpose	TRANSPOSE
Conversion to equiv. matrix form	TOMATRIX
Expand quaternion matrix to equiv. matrix form	UNFOLD
Convert from expanded form to standard form	FOLD

Table 12
List of minimum operations and functions available in the Quaternion Fortran package.

Operation	Fortran operator
Addition	+
Multiplication	*
Division	/
Power	**

D.2. Vector/matrix support

The library supports vector/matrix of quaternion type.

```
TYPE(QTRN)        :: QA(N,   N) ! Matrix A*
TYPE(QTRN)        :: QC(N, 2*N) ! Matrix C*
TYPE(QTRN)        :: QF(N)      ! Vector f*
```

Element-wise operator overloads are supported with: "+", "−" and "∗". Also, operations of scalar-vector/scalar-matrix of quaternions types are supported using "+", "−", "∗" and "/".

Matrix operations such as matrix multiplications and transpose are supported through the overloaded TRANSPOSE and MATMUL functions. Vector dot product is performed using the DOT_PRODUCT functio2n. An example of its use is given below.

```
.
.
! Type definitions.
TYPE(QTRN)        :: QB(N, N)    ! Matrix of quaternion numbers
TYPE(QTRN)        :: QD(N, N)    ! Matrix of quaternion numbers

! Result quaternion variables
TYPE(QTRN)        :: QK(N, N)    ! Matrix of quaternion numbers
TYPE(QTRN)        :: QF(N)       ! Vector of quaternion numbers

! Real value equivalents
DOUBLE PRECISION :: F(4*N)
DOUBLE PRECISION :: K(4*N,4*N)

.
.
! Perform  K* = B*^T D* B*
QK = MATMUL(TRANSPOSE(QB),MATMUL(QD,QB))

! Extract the results into real valued
K = UNFOLD(QK)
F = UNFOLD(QF)

! - or equivalently -

F(    1:  N) = QF%RE ! Real coefficients
F(  N+1:2*N) = QF%I  !  "i" coefficients
F(2*N+1:3*N) = QF%J  !  "j" coefficients
```

```
  F(3*N+1:4*N) = QF%K  !  "k" coefficients
  :
  :
```

Quaternion matrix/vector specific functions such as expand operations to equivalent vector and matrix forms are provided using the functions FOLD and UNFOLD.

D.3. Source code

```
MODULE QUATERNIONS
! A MODULE FOR BASIC QUARTERNION OPERATIONS
  IMPLICIT NONE

  INTEGER, PARAMETER :: REAL_KIND = 8

  TYPE QTRN
     REAL(REAL_KIND) :: RE  ! REAL COMPONENT
     REAL(REAL_KIND) :: I   !-
     REAL(REAL_KIND) :: J   ! | - IMAGINARY COMPONENTS
     REAL(REAL_KIND) :: K   !-
  END TYPE QTRN

INTERFACE OPERATOR(*)          ! QUARTERNION PRODUCT OPERATOR
   MODULE PROCEDURE QPROD
END INTERFACE

INTERFACE OPERATOR(**)         ! QUARTERNION POWER OPERATOR
   MODULE PROCEDURE QPOW
END INTERFACE

INTERFACE SIN                  ! QUARTERNION SIN (TAYLOR SERIES)
  MODULE PROCEDURE TSIN
END INTERFACE

CONTAINS

  FUNCTION QPROD(Q1,Q2) RESULT(QN) ! RETURNS THE QUATERNION PRODUCT
    IMPLICIT NONE
    TYPE(QTRN), INTENT(IN) :: Q1, Q2
    TYPE(QTRN) :: QN

    QN%RE = Q1%RE*Q2%RE- Q1%I*Q2%I - Q1%J*Q2%J - Q1%K*Q2%K
    QN%I  = Q1%RE*Q2%I + Q1%I*Q2%RE+ Q1%J*Q2%K - Q1%K*Q2%J
    QN%J  = Q1%RE*Q2%J - Q1%I*Q2%K + Q1%J*Q2%RE+ Q1%K*Q2%I
    QN%K  = Q1%RE*Q2%K + Q1%I*Q2%J - Q1%J*Q2%I + Q1%K*Q2%RE

  END FUNCTION QPROD

  FUNCTION TSIN(Q) RESULT(RES) ! RETURNS TRUNCATED TAYLOR SERIES SINE
    IMPLICIT NONE
    TYPE(QTRN),   INTENT(IN) :: Q
    TYPE(QTRN)               :: RES

    RES%RE = SIN(Q%RE)
    RES%I  = COS(Q%RE)*Q%I
    RES%J  = COS(Q%RE)*Q%J
    RES%K  = COS(Q%RE)*Q%K

  END FUNCTION

  FUNCTION QPOW(Q,N) RESULT(QN) ! RETURNS POWER OF A QUATERNION
    IMPLICIT NONE
    INTEGER, INTENT(IN) :: N
    INTEGER :: I
    TYPE(QTRN), INTENT(IN) :: Q
    TYPE(QTRN) :: QN

    QN%RE = 1.0D0
    QN%I  = 0.0D0
    QN%J  = 0.0D0
    QN%K  = 0.0D0

    DO I=1,N
      QN = QN*Q
    END DO

  END FUNCTION QPOW
```

```
END MODULE QUATERNIONS


PROGRAM SEVEN_VAR_EXAMPLE
    ! ================ DECLARATIONS ==============================
    USE QUATERNIONS
    IMPLICIT NONE

    ! DEFINE STEP SIZE:
    REAL(REAL_KIND) :: H = 1D-15

    ! DEFINE THE REAL COORDINATES
    REAL(REAL_KIND) :: A0=0.1D0, B0=0.2D0, C0=0.3D0, D0=0.4D0,   &
                       X0=0.5D0, Y0=0.6D0, Z0=0.7D0


    ! DEFINE QUATERNION VERSIONS OF VARIABLES
    TYPE(QTRN)      :: A, B, C, D, X, Y, Z

    ! RESULT HOLDERS
    TYPE(QTRN)      :: QF
    REAL(REAL_KIND) ::    F, DFDA, DFDB, DFDC, &
                       DFDD, DFDX, DFDY, DFDZ
    ! ============================================================

    ! PERTURB THE COORDINATES IN QUATERNION IMAGINARY DIRECTIONS,
    A = QTRN( A0,  H , 0.0, 0.0 )
    B = QTRN( B0, 0.0,  H , 0.0 )
    C = QTRN( C0, 0.0, 0.0,  H  )
    D = QTRN( D0, 0.0, 0.0, 0.0 )
    X = QTRN( X0, 0.0, 0.0, 0.0 )
    Y = QTRN( Y0, 0.0, 0.0, 0.0 )
    Z = QTRN( Z0, 0.0, 0.0, 0.0 )


    ! EVALUATE THE SIN FUNCTION USING PERTURBED VARIABLES.
    QF = SIN(A**3 * B**2 * C**3 * D**4 * X**3 * Y**2 * Z**4)

    F    = QF%RE  !
    DFDA = QF%I/H !
    DFDB = QF%J/H ! EXTRACT THE RESULTS
    DFDC = QF%K/H !

    ! EVALUATE SECOND TIME
    A = QTRN( A0, 0.0, 0.0, 0.0 )
    B = QTRN( B0, 0.0, 0.0, 0.0 )
    C = QTRN( C0, 0.0, 0.0, 0.0 )
    D = QTRN( D0,  H , 0.0, 0.0 )
    X = QTRN( X0, 0.0,  H , 0.0 )
    Y = QTRN( Y0, 0.0, 0.0,  H  )
    Z = QTRN( Z0, 0.0, 0.0, 0.0 )

    ! EVALUATE THE SIN FUNCTION USING PERTURBED VARIABLES.
    QF = SIN(A**3 * B**2 * C**3 * D**4 * X**3 * Y**2 * Z**4)

    DFDD = QF%I/H !
    DFDX = QF%J/H ! EXTRACT THE RESULTS
    DFDY = QF%K/H !

    ! EVALUATE THIRD TIME
    A = QTRN( A0, 0.0, 0.0, 0.0 )
    B = QTRN( B0, 0.0, 0.0, 0.0 )
    C = QTRN( C0, 0.0, 0.0, 0.0 )
    D = QTRN( D0, 0.0, 0.0, 0.0 )
    X = QTRN( X0, 0.0, 0.0, 0.0 )
    Y = QTRN( Y0, 0.0, 0.0, 0.0 )
    Z = QTRN( Z0,  H , 0.0, 0.0 )


    ! EVALUATE THE SIN FUNCTION USING PERTURBED VARIABLES.
    QF = SIN(A**3 * B**2 * C**3 * D**4 * X**3 * Y**2 * Z**4)

    DFDZ = QF%I/H

    PRINT*, '== RESULT OF QUATERNION ANALYSIS =='
    PRINT*, 'F:    ',    F, 'DFDA:', DFDA, 'DFDB:', DFDB, 'DFDC:', DFDC
    PRINT*, 'DFDD:', DFDD, 'DFDX:', DFDX, 'DFDY:', DFDY, 'DFDZ:', DFDZ

END PROGRAM SEVEN_VAR_EXAMPLE
```

References

[1] J.R. Martins, P. Sturdza, J.J. Alonso, The complex-step derivative approximation, ACM Trans. Math. Softw. 29 (3) (2003) 245–262.
[2] W. Squire, G. Trapp, Using complex variables to estimate derivatives of real functions, SIAM Rev. 40 (1) (1998) 110–112.
[3] J.F. Monsalvo, M.J. García, H. Millwater, Y. Feng, Sensitivity analysis for radiofrequency induced thermal therapies using the complex finite element method, Finite Elem. Anal. Des. 135 (2017) 11–21.
[4] J. Grisham, A. Akbariyeh, W. Jin, B.H. Dennis, B.P. Wang, Application of the semi-analytic complex variable method to computing sensitivities in heat conduction, J. Heat Transf. 140 (8) (2018) 082006.
[5] J. Garza, H. Millwater, Multicomplex newmark-beta time integration method for sensitivity analysis in structural dynamics, AIAA J. 53 (5) (2015) 1188–1198.
[6] A. Montoya, H. Millwater, Sensitivity analysis in thermoelastic problems using the complex finite element method, J. Therm. Stresses 40 (3) (2017) 302–321.
[7] A. Voorhees, H. Millwater, R. Bagley, Complex variable methods for shape sensitivity of finite element models, Finite Elem. Anal. Des. 47 (10) (2011) 1146–1156.
[8] A. Montoya, R. Fielder, A. Gomez-Farias, H. Millwater, Finite-element sensitivity for plasticity using complex variable methods, J. Eng. Mech. 141 (2) (2014) 04014118.
[9] W.K. Anderson, J.C. Newman, D.L. Whitfield, E.J. Nielsen, Sensitivity analysis for Navier-Stokes equations on unstructured meshes using complex variables, AIAA J. 39 (1) (2001) 56–63.
[10] D.L. Whitfield, J.C. Newman, W.K. Anderson, Step-size independent approach for multidisciplinary sensitivity analysis, J. Aircr. 40 (3) (2003) 566–573.
[11] C.O.E. Burg, J.C. Newman III, Computationally efficient, numerically exact design space derivatives via the complex Taylor's series expansion method, Comput. Fluids 32 (3) (2003) 373–383.
[12] J. Kim, D.G. Bates, I. Postlethwaite, Nonlinear robust performance analysis using complex-step gradient approximation, Automatica 42 (1) (2006) 177–182.
[13] L.I. Cerviño, T.R. Bewley, On the extension of the complex-step derivative technique to pseudospectral algorithms, J. Comput. Phys. 187 (2) (2003) 544–549.
[14] B.P. Wang, A.P. Apte, Complex variable method for eigensolution sensitivity analysis, AIAA J. 44 (12) (2006) 2958–2961.
[15] H. Millwater, D. Wagner, A. Baines, A. Montoya, A virtual crack extension method to compute energy release rates using a complex variable finite element method, Eng. Fract. Mech. 162 (2016) 95–111.
[16] A. Montoya, D. Ramirez-Tamayo, H. Millwater, M. Kirby, A complex-variable virtual crack extension finite element method for elastic-plastic fracture mechanics, Eng. Fract. Mech. (2018).
[17] R.E. Spall, W. Yu, Imbedded dual-number automatic differentiation for computational fluid dynamics sensitivity analysis, J. Fluids Eng. 135 (1) (2013) 014501.
[18] G. Lantoine, R.P. Russell, T. Dargent, Using multicomplex variables for automatic computation of high-order derivatives, ACM Trans. Math. Softw. 38 (3) (2012) 16.
[19] J.A. Fike, J.J. Alonso, The development of hyper-dual numbers for exact second-derivative calculations, AIAA Pap. 886 (2011) 124.
[20] W.R. Hamilton, XI. On quaternions; or on a new system of imaginaries in algebra, Philos. Mag. 33 (219) (1848) 58–60.
[21] W.R. Hamilton, Researches respecting quaternions: first series, Trans. R. Ir. Acad. 21 (part 1) (1848) 199–296.
[22] J. Vince, Quaternions for Computer Graphics, Springer Science & Business Media, 2011.
[23] M. Geradin, A. Cardona, Kinematics and dynamics of rigid and flexible mechanisms using finite elements and quaternion algebra, Comput. Mech. 4 (2) (1988) 115–135.
[24] I. Romero, The interpolation of rotations and its application to finite element models of geometrically exact rods, Comput. Mech. 34 (2) (2004) 121–133.
[25] H. Zhong, R. Zhang, N. Xiao, A quaternion-based weak form quadrature element formulation for spatial geometrically exact beams, Arch. Appl. Mech. 84 (12) (2014).
[26] I.L. Kantor, A.S. Solodovnikov, Hypercomplex Numbers: An Elementary Introduction to Algebras, Springer, 1989.
[27] J. Turner, Quaternion-based partial derivative and state transition matrix calculations for design optimization, in: 40th AIAA Aerospace Sciences Meeting & Exhibit, 2002, p. 448.
[28] J. Baez, The octonions, Bull. Am. Math. Soc. 39 (2) (2002) 145–205.
[29] K. Imaeda, M. Imaeda, Sedenions: algebra and analysis, Appl. Math. Comput. 115 (2–3) (2000) 77–88.
[30] A.M. Aguirre-Mesa, M.J. Garcia, H. Millwater, MultiZ: a library for computation of high order derivatives using multicomplex or multidual numbers, ACM Trans. Math. Softw. (2019), submitted for publication.
[31] J. Fish, T. Belytschko, A First Course in Finite Elements, vol. 1, John Wiley & Sons, New York, 2007.
[32] A. Aziz, M. Torabi, Thermal stresses in a hollow cylinder with convective boundary conditions on the inside and outside surfaces, J. Therm. Stresses 36 (10) (2013) 1096–1111.
[33] Y. Tian, Matrix representations of octonions and their applications, Adv. Appl. Clifford Algebras 10 (1) (2000) 61–90.