Variational Framework for FIC Formulations **i**n Continuum Mechanics: High Order Tensor-Derivative **T**ransformations and Invariants

Carlos A. Felippa[*], **Eugenio Oñate**[†] **and Sergio R. Idelsohn**[‡]

[*] Professor, Department of Aerospace Engineering Sciences,
University of Colorado at Boulder, Boulder, CO 80309-0429, USA
carlos.felippa@colorado.edu

[†] Professor, Universidad Politécnica de Cataluña and Director of CIMNE, Barcelona 08034, Spain

[‡] ICREA Research Professor at CIMNE, Universidad Politécnica de Cataluña, Barcelona 08034, Spain

July 16, 2017

Contributed to *Archives For Computational Methods In Engineering*

ABSTRACT

This is part of an article series on a variational framework for continuum mechanics based on the Finite Increment Calculus (FIC). That formulation utilizes high order derivatives of the classical fields of continuum mechanics integrated over control regions to construct stabilizing modification terms. Fields include displacements, body forces, strains, stresses, pressure and volumetric strains. To support observer-invariant FIC formulations, we have catalogued field transformation equations as well as sets of linear and quadratic invariants of fields and of their derivatives up to appropriate order. Attention is focused on the two-dimensional case of a body in plane strain under the drilling-rotation transformation group. Results are presented for displacement and body-force derivatives of orders up to 4, and for stress, strain, pressure and volumetric strain derivatives of order up to 3. The material assembled here is self-contained because this catalog is believed to be useful for non-FIC applications; for example gradient-based, nonlocal constitutive models of multiscale mechanics and physics that involve finite characteristic dimensions analogous to FIC steplengths.

Key Words: continuum mechanics, tensor field derivatives, invariants, observer invariance, objectivity, frame indifference, characteristic length, Finite Increment Calculus

1 Introduction

The Finite Increment Calculus (FIC) is a residual-based formulation of computational methods in science and engineering. It was proposed in 1998 [35]. For applications surveys see [37, 46].

The formulation process is flowcharted in Figure 1. The governing equation(s) are generally partial differential equations in space and time, which are placed in residual form. Those residuals are expanded in Taylor series in the space variables and truncated to given orders. The truncated expansions are integrated over *control regions* (volumes, areas or segments) to build *modified residuals*, which may be interpreted as *generalized balance equations*. The integration process naturally introduces *steplenghts* linked to the dimensions of the control region. These may be viewed as to-be-chosen-later parameters.

The modified residuals are "functionalized" to produce either variational or weak forms. For the latter the Galerkin method is often adopted, although other instances of the Method of Weighted Residuals (MWR) [10] may in fact be used. The functional form is then discretized by the Finite Element Method (FEM), a process that necessarily requires numerically instantiating steplengths. Finally, the FEM equations are solved to produce the numerical solution.

Why are the governing residuals Taylor expanded? The original motivation was *stabilization* of flow problems involving advection and diffusion. The unique feature of FIC is that this process is done at the *governing equation level*. This is in contrast to other residual based methods, in which such terms are introduced *a posteriori* in the governing functional or weak form. The use of steplengths to improve accuracy has also received attention [8, 39, 45].

FIC has been particularly successful for stabilization of advective (transport) incompressible flow processes [39–41, 43–45]. The method has been also tried for quasi-incompressible elastic solid mechanics but in Galerkin form [36, 38] since models used therein were not encompassed by the conventional framework of variational calculus. The idea of placing FIC in a Ritz variational framework was first explored in [8] for diffusion and Helmholtz problems. The key idea is to get the modified residuals as Euler-Lagrange equations of the FIC functional.

Recent successes of Particle Finite Element Methods (PFEM) for gravity dominated flows [25] as well as fluid-structure interaction (FSI) problems [13, 26, 42] has opened the possibility of considering a unified variational formulation that relies on the FIC concept while encompassing compressible and incompressible media. This goal becomes attainable because the full-Lagrangian kinematic description of PFEM eliminates advective terms. Such formulation would offer advantages in software reuse, since discarding stabilization terms for compressible solids reduces the FIC functional to a conventional one. Thus, existing FEM libraries for linear and nonlinear analysis could be utilized for modeling the compressible portion of FSI problems. Another practical advantage of a standard variational formulation is clarification of tricky boundary conditions for program users.

2 Motivation and Related Aplications

As noted above, the FIC variational formulation of problems in continuum mechanics leads naturally to functionals that contain *higher order spatial derivatives*; for instance, displacement second derivatives

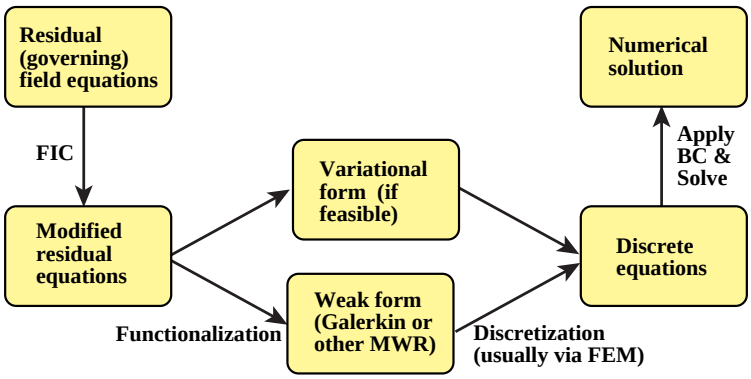

Figure 1: Steps in a FIC-based analysis process.

and/or pressure first derivatives. The same is true for weak forms obtained by the Galerkin method or its MWR variants. For the FEM discretiztion such derivatives can be conveniently recast as vectors, which appear as components of quadratic and/or bilinear forms. The kernels of such forms are matrices whose entries depend on the FIC steplengths. Here the question of invariance arises: would those forms be invariant under change of material axes? This "observer invariance" is important for comparing results produced by different researchers and teams. Studies of this nature are hampered by the lack of readily available information on high-order derivative transformations and invariants in the matrix framework favored by FEM discretizations.

To facilitate those studies, this paper collects results on 2D rotational transformations as well as associated linear and quadratic invariants of tensor derivatives of order up to 4 for displacements and body forces, and up to 3 for strains, stresses, volumetric strain, and pressure. The material is intended to primarily support an article on FIC variational functionals under preparation [9].

As noted in the abstract, the results may have potential side uses in nonlocal constitutive theories that model size effects, couple-stresses, or actions-at-a-distance. Those topics are subsumed by the term *generalized continuum mechanics* or GCM. Following the prescient but long ignored contribution by the Cosserat brothers in the early XX Century [6], GCM was revived by Mindlin and Toupin in the 1960s [30–32,50] through the introduction of coupled field and polar theories that involved first gradients of strains. It was generalized to arbitrary tensor derivative order by Green and Rivlin [17, 18]. General invariant forms of such tensors are summarized in Section B.II of [49] in the form of representation theorems.

GCM largely remained an academic curiosity until the late 1990s, when it acquired renewed vitality prompted by rising interest into micro- and nano-mechanics as well as multiscale modeling. Recent surveys may be found in [7, 29]. Observer invariance remains important in GCM constitutive models, despite some ongoing controversies. It was first clearly stated in the expository memoir [48] as the "principle of material indifference." (Alternative names: "frame indifference," "objectivity" and "isotropy of space".) For a recent review of its history and current status, see [14].

Our chief motivation for cataloging results in one place is the difficulty of finding a coherent body of information of transformations and invariance available in the matrix framework favored by computational implementations such as FEM. It should be noted that Web searches are not great help. For example a Google search on "strain gradient invariants" as of May 2017 returns 8 hits, mostly connected with

nonlocal plasticity theories developed by Fleck and Hutchinson [11, 12], as well as Gurtin's viscoelastoplastic model [20]. Trying "pressure derivative invariants" gave no matches.

By contrast, the abstract theory of algebraic and differential invariants is much older. It was well established since its creation by Hilbert in the 1890s [23], and is abundantly accessible in textbooks and monographs, e.g., [15,19,33,34,51,52]. Those expositions, however, tend to be disconnected from actual engineering application needs. For example, classical results on binary polynomial forms for conformal mapping are of little help to a computational mechanician or GCM modeler struggling with second order strain derivatives constrained by compatibility equations.

To streamline the organization, fields are grouped into three conjugate pairs: (1) displacements and body forces, (2) strains and stresses, (3) pressure and volumetric strains. The material is presented in a self-contained manner. No prior knowledge of tensorial analysis or classical invariance theory is assumed.

3 Problem and Notation Summary

We consider an isotropic, linearly elastic, solid or fluid body in plane strain from which a "slice" of unit thickness is virtually extracted as shown in Figure 2(a,b). Notation and sign conventions for the fields considered here are defined in Figure 3. All displacements are rotations are considered infinitesimal. The position (global) coordinate axes are denoted by $\{x_1, x_2, x_3\}$, which form a right-handed system. Rotated axes $\{\bar{x}_1, \bar{x}_2, \bar{x}_3\}$ are defined as pictured in Figure 2(c). The rotation angle about x_3 is denoted by φ, positive counterclockwise. The following abbreviations for sines and cosines of integer multiples of that angle are used throughout:

$$c = \cos\varphi, \quad s = \sin\varphi, \quad c_n = \cos\, n\varphi, \quad s_n = \sin\, n\varphi, \qquad n = 2, 3, \ldots \tag{1}$$

Terms "C-vector" and "DS-orthogonal" are abbreviations for *cast vector* and *diagonally-similar orthogonal*, respectively, as defined below. All quantities are assumed real until otherwise noted.

A partial derivative respect to x_i is abbreviated by writing i as subscript following a comma, and repeating as necessary. For example

$$u_{1,122} = \frac{\partial^3 u_1}{\partial x_1\, \partial x_2\, \partial x_2}. \tag{2}$$

Equality of cross derivatives to any order will be assumed; e.g., $u_{1,122} = u_{1,212} = u_{1,221}$. The Laplacian operator is denoted by Δ; for instance $\Delta p = p_{,11} + p_{,22}$.

An *invariant* is an expression $J(x_1, x_2)$ that keeps a constant value for all rotation angles φ. Two vectors, say $\mathbf{p}(x_1, x_2, \varphi)$ and $\mathbf{q}(x_1, x_2, \varphi)$, are said to be *conjugate* if $\mathbf{p}^T\,\mathbf{q}$ is invariant, that is, independent of φ. (This dot product often has interpretation of work when $\mathbf{p}$ and $\mathbf{q}$ are zero-order quantities, but that meaning is lost once derivatives are taken.) A similar definition and interpretation holds for quadratic forms $\mathbf{p}^T\,\mathbf{W}\,\mathbf{p}$, in which $\mathbf{W}$ is a symmetric matrix.

The identity matrix is generically denoted by $\mathbf{I}$. If showing its order n is important, it is written $\mathbf{I}_n$.

A square matrix $\mathbf{Q}$ is said to be *orthogonal* if $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$. If is *properly orthogonal* if $\det\mathbf{Q} = +1$. If $\mathbf{Q}^T\,\mathbf{Q} = \mathbf{D}$, in which $\mathbf{D}$ is a positive-definite diagonal matrix, $\mathbf{Q}$ is said to be *diagonally-scaled orthogonal*, or *DS-orthogonal* for short.

All field tensors, whatever the order, are mapped to one-dimensional arrays. The operation is called *casting* and the result is a *cast vector* or *C-vector*. Casting rules are stated in 4.1 for the displacement-and-body-force field pair. These rules hold for other conjugate field pairs: strain and stress, and pressure-and-volumetric-strain, with minor changes described where appropriate.

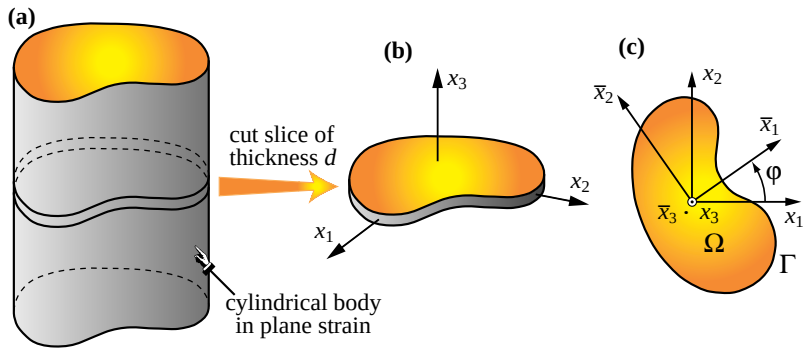

Figure 2: Cylindrical body in plane strain: (a) body, (b) generic slice, (c) inplane axes rotation.

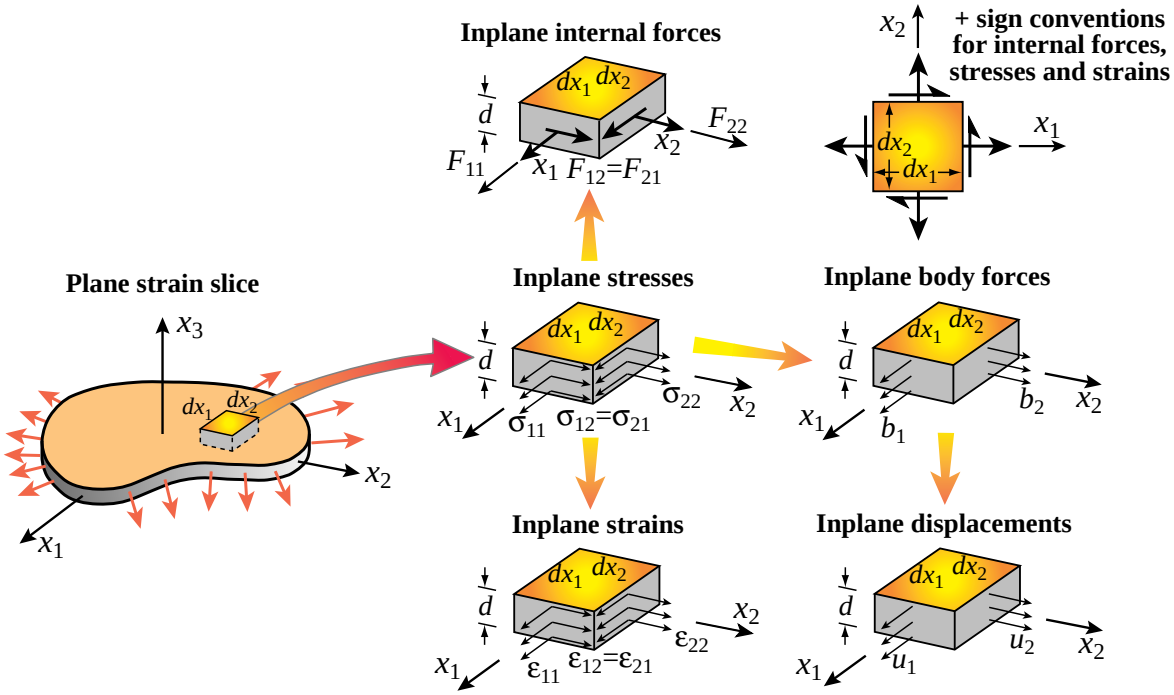

Figure 3: Notational and sign conventions.

As regards quantities in the x_3 direction, all fields with a 3 subscript are assumed constant over the body. In addition the following fields vanish identically: u_3, ϵ_{33}, $\epsilon_{13} = \epsilon_{31}$, $\epsilon_{23} = \epsilon_{32}$, $\sigma_{13} = \sigma_{31}$, and $\sigma_{23} = \sigma_{32}$. Results may be transcribed to plane stress with minor modifications.

4 Displacements and Body Forces

We first consider transformations of 2D displacement and body force tensor fields and their first 4 derivatives, grouped since they are dot-product conjugate. The derivative order is denoted by $m \geq 0$. This index appears as a parentheses-enclosed superscript to avoid potential confusion with exponents; for example $\mathbf{u}^{(3)}$ is an 8-vector that collects third-order displacement derivatives. If $m = 0$ the superscript is often omitted.

4.1 Vector Casting Rules

Tensor components of any order are cast into C-vectors to facilitate matrix notation. For orders $m = 0$ and $m = 1$, those vectors are unique. For orders 2 and higher, general C-vectors may carry weighting coefficients in the mixed derivatives as shown in the lists below.

$$\mathbf{u} = \mathbf{u}^{(0)} = [\, u_1 \quad u_2 \,]^T , \tag{3}$$

$$\mathbf{u}^{(1)} = [\, u_{1,1} \quad u_{1,2} \quad u_{2,1} \quad u_{2,2} \,]^T , \tag{4}$$

$$\mathbf{u}^{(2)}(k) = [\, u_{1,11} \quad k\, u_{1,12} \quad u_{1,22} \quad u_{2,11} \quad k\, u_{2,12} \quad u_{2,22} \,]^T , \tag{5}$$

$$\mathbf{u}^{(3)}(k) = [\, u_{1,111} \quad k\, u_{1,112} \quad k\, u_{1,122} \quad u_{1,222} \quad u_{2,111} \quad k\, u_{2,112} \quad k\, u_{2,122} \quad u_{2,222} \,]^T , \tag{6}$$

$$\begin{aligned} \mathbf{u}^{(4)}(k_1, k_2) = [\, & u_{1,1111} \quad k_1\, u_{1,1112} \quad k_2\, u_{1,1122} \quad k_1\, u_{1,1222} \quad u_{1,2222} \\ & u_{2,1111} \quad k_1\, u_{2,1112} \quad k_2\, u_{2,1122} \quad k_1\, u_{2,1222} \quad u_{2,2222} \,]^T , \end{aligned} \tag{7}$$

$$\mathbf{b} = \mathbf{b}^{(0)} = [\, b_1 \quad b_2 \,]^T , \tag{8}$$

$$\mathbf{b}^{(1)} = [\, b_{1,1} \quad b_{1,2} \quad b_{2,1} \quad b_{2,2} \,]^T , \tag{9}$$

$$\mathbf{b}^{(2)}(k) = \left[\, b_{1,11} \quad \frac{2}{k}\, b_{1,12} \quad b_{1,22} \quad b_{2,11} \quad \frac{2}{k}\, b_{2,12} \quad b_{2,22} \,\right]^T , \tag{10}$$

$$\mathbf{b}^{(3)}(k) = \left[\, b_{1,111} \quad \frac{3}{k}\, b_{1,112} \quad \frac{3}{k}\, b_{1,122} \quad b_{1,222} \quad b_{2,111} \quad \frac{3}{k}\, b_{2,112} \quad \frac{3}{k}\, b_{2,122} \quad b_{2,222} \,\right]^T , \tag{11}$$

$$\begin{aligned} \mathbf{b}^{(4)}(k_1, k_2) = \Big[\, & b_{1,1111} \quad \frac{4}{k_1}\, b_{1,1112} \quad \frac{6}{k_2}\, b_{1,1122} \quad \frac{4}{k_1}\, b_{1,1222} \quad b_{1,2222} \\ & b_{2,1111} \quad \frac{4}{k_1}\, b_{2,1112} \quad \frac{6}{k_2}\, b_{2,1122} \quad \frac{4}{k_1}\, b_{2,1222} \quad b_{2,2222} \,\Big]^T . \end{aligned} \tag{12}$$

In the second through fourth derivative forms, k, k_1 and k_2 are positive real numbers known as *mixed derivative casting coefficients*, or MDCC for short. If those coefficients are left arbitrary, they appear as arguments. Three specializations used in this paper: orthogonal, binomial and flat casting, are obtained by selecting the MDCC as shown in Table 1. That table also introduces notation for the three forms, which are further discussed in 4.2.

.

Table 1: **Instantiation of Displacement and Body Force C-Vectors for Orders 2,3,4.**

Field	**Order**	**Orthogonal Casting**		**Binomial Casting**		**Flat Casting**	
	m	Symbol	MDCC	Symbol	MDCC	Symbol	MDCC
Displacement	2	$\widehat{\mathbf{u}}^{(2)}$	$k{=}\sqrt{2}$	$\mathbf{u}^{(2)}$	$k{=}2$	$\bar{\mathbf{u}}^{(2)}(1)$	$k{=}1$
Displacement	3	$\widehat{\mathbf{u}}^{(3)}$	$k{=}\sqrt{3}$	$\mathbf{u}^{(3)}$	$k{=}3$	$\bar{\mathbf{u}}^{(3)}(1)$	$k{=}1$
Displacement	4	$\widehat{\mathbf{u}}^{(4)}$	$k_1{=}2, k_2{=}\sqrt{6}$	$\mathbf{u}^{(4)}$	$k_1{=}4, k_2{=}6$	$\bar{\mathbf{u}}^{(4)}(1,1)$	$k_1{=}k_2{=}1$
Body force	2	$\widehat{\mathbf{b}}^{(2)}$	$k{=}\sqrt{2}$	$\bar{\mathbf{b}}^{(2)}(1)$	$k{=}1$	$\mathbf{b}^{(2)}$	$k{=}2$
Body force	3	$\widehat{\mathbf{b}}^{(3)}$	$k{=}\sqrt{3}$	$\bar{\mathbf{b}}^{(3)}(1)$	$k{=}1$	$\mathbf{b}^{(3)}$	$k{=}3$
Body force	4	$\widehat{\mathbf{b}}^{(4)}$	$k_1{=}2, k_2{=}\sqrt{6}$	$\bar{\mathbf{b}}^{(4)}(1,1)$	$k_1{=}k_2{=}1$	$\mathbf{b}^{(4)}$	$k_1{=}4, k_2{=}6$

Conjugate pairs: $\mathbf{u}^{(m)}$ with $\mathbf{b}^{(m)}$, $\widehat{\mathbf{u}}^{(m)}$ with $\widehat{\mathbf{b}}^{(m)}$, and $\overline{\mathbf{u}}^{(m)}$ with $\overline{\mathbf{b}}^{(m)}$.
Orthogonal $\widehat{\mathbf{u}}^{(m)}$ and $\widehat{\mathbf{b}}^{(m)}$ are self-conjugate.

4.2 Orthogonal, Binomial and Flat Casting

Orthogonal casting builds C-vectors, denoted by $\widehat{\mathbf{u}}^{(m)}$ and $\widehat{\mathbf{b}}^{(m)}$ for displacement and body forces, respectively, which produce *orthogonal* transformation matrices. They are identified with a superposed hat. For $m = 0, 1$, they agree with the general form, whereas for $m = 2, 3, 4$ they are obtained by appropriately setting the MDCC as shown in Table 1. For example: $\widehat{\mathbf{u}}^{(1)} = \mathbf{u}^{(1)}$, $\widehat{\mathbf{u}}^{(3)} = \mathbf{u}^{(3)}(\sqrt{3})$, and $\widehat{\mathbf{b}}^{(4)} = \mathbf{b}^{(4)}(2, \sqrt{6})$. Orthogonal casting preserves Frobenius tensor norms, but has the inconvenience of polluting expressions with square roots, as Table 1 makes plain.

Binomial casting for displacements paired with *flat casting* for body forces, is a common choice. It has the advantage that all MDCC are integers, although strict transformation orthogonality is lost.

Binomial displacement C-vectors will be denoted simply as $\mathbf{u}^{(m)}$ for all m, omitting MDCC arguments if $m \geq 2$. Those coefficients are set to mixed-derivative permutation counts that are binomial coefficients, whence the name. For example, the second entry of $\mathbf{u}^{(3)}(k)$ collects $\binom{3}{2} = 3$ mixed derivatives: $u_{1,112}$, $u_{1,121}$ and $u_{1,211}$; thus $k = 3$. Specific values are listed in Table 1 for $m = 2, 3, 4$, with no change needed if $m = 0, 1$. The detailed configurations are

$$\mathbf{u} = \mathbf{u}^{(0)} = [\, u_1 \quad u_2 \,]^T , \tag{13}$$

$$\mathbf{u}^{(1)} = \nabla \mathbf{u} = [\, u_{1,1} \quad u_{1,2} \quad u_{2,2} \quad u_{2,2} \,]^T , \tag{14}$$

$$\mathbf{u}^{(2)} = [\, u_{1,11} \quad 2u_{1,12} \quad u_{1,22} \quad u_{2,11} \quad 2u_{2,12} \quad u_{2,22} \,]^T , \tag{15}$$

$$\mathbf{u}^{(3)} = [\, u_{1,111} \quad 3u_{1,112} \quad 3u_{1,122} \quad u_{1,222} \quad u_{2,111} \quad 3u_{2,112} \quad 3u_{2,122} \quad u_{2,222} \,]^T , \tag{16}$$

$$\mathbf{u}^{(4)} = [\, u_{1,1111} \quad 4u_{1,1112} \quad 6u_{1,1122} \quad 4u_{1,1222} \quad u_{1,2222} \quad u_{2,1111} \quad 4u_{2,1112} \quad 6u_{2,1122} \quad 4u_{2,1222} \quad u_{2,2222} \,]^T , \tag{17}$$

Body force C-vectors conjugate to binomially cast displacement vectors are constructed by *flat casting*; that is, all MDCC are set to one for $m \geq 2$ in (10) through (12):

$$\mathbf{b}^{(2)} = \mathbf{b}^{(2)}(1), \quad \mathbf{b}^{(3)} = \mathbf{b}^{(3)}(1), \quad \mathbf{b}^{(4)} = \mathbf{b}^{(4)}(1,1), \tag{18}$$

with no change if $m = 0, 1$. The detailed configurations are obvious and not shown. Flat-cast displacement C-vectors are denoted by $\overline{\mathbf{u}}^{(m)}$ whereas binomially cast body force C-vectors are denoted by $\overline{\mathbf{b}}^{(m)}$. The general conjugation rules are: $\mathbf{u}^{(m)}$ with $\mathbf{b}^{(m)}$, $\widehat{\mathbf{u}}^{(m)}$ with $\widehat{\mathbf{b}}^{(m)}$, and $\overline{\mathbf{u}}^{(m)}$ with $\overline{\mathbf{b}}^{(m)}$. These dual pairings explain the choice of symbols. The last combination (flat displacement C-vectors with binomial body force C-vectors) is less common in actual FEM implementations.

4.3 Rotational Transformations

We study how C-vectors $\mathbf{u}^{(m)}$ or $\mathbf{b}^{(m)}$ transform under rotation φ (+CCW about x_3, cf. Figure **??**). To bring the angle into play the notation of (13) through (17) is expanded with an additional argument or an extra subscript:

$$\begin{cases} \text{Original C-vector } (\varphi = 0)\text{:}\ \ \mathbf{u}^{(m)} \equiv \mathbf{u}_0^{(m)} \equiv \mathbf{u}^{(m)}(0), \\ \text{Rotated C-vector (arbitrary } \varphi)\text{:}\ \ \mathbf{u}_\varphi^{(m)} \equiv \mathbf{u}^{(m)}(\varphi), \end{cases} \tag{19}$$

and likewise for body forces. The additional argument is preferred over the extra subscript when MDCC arguments must be explicitly shown, or functions of φ appear, as in (21) below. Here is the notation used for C-vectors of order $m = 0$ through 4 with arbitrary MDCC:

$$\begin{aligned} &\mathbf{u}^{(m)}(\varphi) = \mathbf{T}_u^{(m)}(\varphi)\ \mathbf{u}^{(m)},\ \ m = 0, 1, \quad \mathbf{u}^{(m)}(\varphi, k) = \mathbf{T}_u^{(m)}(\varphi, k)\ \mathbf{u}^{(m)}(k),\ \ m = 2, 3, \\ &\mathbf{u}^{(4)}(\varphi, k_1, k_2) = \mathbf{T}_u^{(4)}(\varphi, k_1, k_2)\ \mathbf{u}^{(4)}(k_1, k_2), \\ &\mathbf{b}^{(m)}(\varphi) = \mathbf{T}_b^{(m)}(\varphi)\ \mathbf{b}^{(m)},\ \ m = 0, 1, \quad \mathbf{b}^{(m)}(\varphi, k) = \mathbf{T}_b^{(m)}(\varphi, k)\ \mathbf{b}^{(m)}(k),\ \ m = 2, 3, \\ &\mathbf{b}^{(4)}(\varphi, k_1, k_2) = \mathbf{T}_b^{(4)}(\varphi, k_1, k_2)\ \mathbf{b}^{(4)}(k_1, k_2).\text{‘} \end{aligned} \tag{20}$$

For brevity, argument φ may be converted to an extra subscript of the transformation matrix if no confusion may arise. For example, $\mathbf{T}_{u\varphi}^{(1)} \equiv \mathbf{T}_u^{(1)}(\varphi)$, $\mathbf{T}_{u\varphi}^{(4)}(k_1, k_2) \equiv \mathbf{T}_u^{(4)}(\varphi, k_1, k_2)$. From the definitions (20) angle-addition is associated with matrix multiplication: $\mathbf{T}_u^{(m)}(\varphi_1{+}\varphi_2) = \mathbf{T}_u^{(m)}(\varphi_1)\,\mathbf{T}_u^{(m)}(\varphi_2)$. in which MDCC for $m \geq 2$ are omitted for brevity. Setting $\varphi_1 = \varphi$ and $\varphi_2 = -\varphi$ one gets $\mathbf{T}_{u0}^{(m)} = \mathbf{I} = \mathbf{T}_u^{(m)}(\varphi)\,\mathbf{T}_u^{(m)}(-\varphi)$, whence

$$\left(\mathbf{T}_u^{(m)}(\varphi)\right)^{-1} = \mathbf{T}_u^{(m)}(-\varphi). \tag{21}$$

Thus, the inverse of a transformation matrix is obtained simply by reversing the angle argument. That sign change negates all sine terms while keeping constant and cosine terms invariant.

If orthogonal casting is used, $\mathbf{u}^{(m)}$ is denoted by $\widehat{\mathbf{u}}^{(m)}$ (cf. Table 1) and transformation matrices are tagged with a hat; for example $\widehat{\mathbf{T}}_u^{(4)}(\varphi) = \mathbf{T}_u^{(4)}\left(\varphi, 2, \sqrt{6}\right)$ In that case, the matrices are proper orthogonal: $\left(\widehat{\mathbf{T}}_u^{(m)}\right)^T \widehat{\mathbf{T}}_u^{(m)} = \mathbf{I}_{2m+2}$. The same property holds for body forces.

4.4 Transformations for General C-Vectors

A list of displacement transformation matrices for arbitrary MDCC follows.

$$\mathbf{T}_u^{(0)}(\varphi) = \mathbf{T}_u(\varphi) = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}. \tag{22}$$

$$\mathbf{T}^{(1)}_{u\varphi} = \mathbf{T}^{(1)}_{u}(\varphi) = \frac{1}{2}\begin{bmatrix} 1+c_2 & s_2 & s_2 & 1-c_2 \\ -s_2 & 1+c_2 & -1+c_2 & s_2 \\ -s_2 & -1+c_2 & 1+c_2 & s_2 \\ 1-c_2 & -s_2 & -s_2 & 1+c_2 \end{bmatrix}, \tag{23}$$

$$\mathbf{T}^{(2)}_{u\varphi}(k) = \mathbf{T}^{(2)}_{u}(\varphi,k) = \frac{1}{4}\begin{bmatrix} a_1 & 2b_3/k & a_4 & b_3 & 2a_4/k & b_2 \\ -k\,b_3 & 2a_3 & k\,b_3 & -k\,a_4 & -2b_4 & k\,a_4 \\ a_4 & -2b_3/k & a_1 & b_2 & -2a_4/k & b_3 \\ -b_3 & -2a_4/k & -b_2 & a_1 & 2b_3/k & a_4 \\ k\,a_4 & 2b_4 & -k\,a_4 & -k\,b_3 & 2a_3 & k\,b_3 \\ -b_2 & 2a_4/k & -b_3 & a_4 & -2b_3/k & a_1 \end{bmatrix}, \tag{24}$$

$$\mathbf{T}^{(3)}_{u\varphi}(k) = \mathbf{T}^{(3)}_{u}(\varphi,k) = \frac{1}{8}\begin{bmatrix} d_1 & \frac{3\,e_1}{k} & \frac{3\,d_2}{k} & e_2 & e_1 & \frac{3\,d_2}{k} & \frac{3\,e_2}{k} & d_3 \\ -k\,e_1 & d_4 & e_3 & k\,d_2 & -k\,d_2 & -e_4 & -d_5 & k\,e_2 \\ k\,d_2 & -e_3 & d_4 & k\,e_1 & k\,e_2 & d_5 & -e_4 & k\,d_2 \\ -e_2 & \frac{3\,d_2}{k} & -\frac{3\,e_1}{k} & d_1 & -d_3 & \frac{3\,e_2}{k} & -\frac{3\,d_2}{k} & e_1 \\ -e_1 & -\frac{3\,d_2}{k} & -\frac{3\,e_2}{k} & -d_3 & d_1 & \frac{3\,e_1}{k} & \frac{3d_2}{k} & e_2 \\ k\,d_2 & e_4 & d_5 & -k\,e_2 & -k\,e_1 & d_4 & e_3 & k\,d_2 \\ -k\,e_2 & -d_5 & e_4 & -k\,d_2 & k\,d_2 & -e_3 & d_4 & k\,e_1 \\ d_3 & -\frac{3\,e_2}{k} & \frac{3\,d_2}{k} & -e_1 & -e_2 & \frac{3d_2}{k} & -\frac{3\,e_1}{k} & d_1 \end{bmatrix}. \tag{25}$$

$$\mathbf{T}^{(4)}_{u\varphi}(k_1,k_2) = \mathbf{T}^{(4)}_{u}(\varphi,k_1,k_2) =$$

$$\frac{1}{16}\begin{bmatrix} f_1 & 4\frac{g_1}{k_1} & 6\frac{f_2}{k_2} & 4\frac{g_2}{k_1} & f_3 & g_1 & 4\frac{f_2}{k_1} & 6\frac{g_2}{k_2} & 4\frac{f_3}{k_1} & g_3 \\ -k_1g_1 & 4f_4 & 6\frac{k_1g_4}{k_2} & 4f_5 & k_1g_2 & -k_1f_6 & -4g_5 & 6\frac{k_1f_7}{k_2} & -4g_6 & k_1f_3 \\ k_2f_6 & -4\frac{k_2g_4}{k_1} & 2f_8 & 4\frac{k_2g_4}{k_1} & k_2f_6 & k_2g_2 & -4\frac{k_2f_7}{k_1} & 2g_7 & 4\frac{k_2f_7}{k_1} & k_2g_2 \\ -k_1g_2 & 4f_5 & -6\frac{k_1g_4}{k_2} & 4f_4 & k_1g_1 & -k_1f_3 & -4g_6 & -6\frac{k_1f_7}{k_2} & -4g_5 & k_1f_6 \\ f_3 & -4\frac{g_2}{k_1} & 6\frac{f_2}{k_2} & -\frac{4g_1}{k_1} & f_1 & g_3 & -4\frac{f_3}{k_1} & 6\frac{g_2}{k_2} & -4\frac{f_6}{k_1} & g_1 \\ -g_1 & -4\frac{f_6}{k_1} & -6\frac{g_2}{k_2} & -4\frac{f_3}{k_1} & -g_3 & f_1 & \frac{4g_1}{k_1} & \frac{6f_2}{k_2} & 4\frac{g_2}{k_1} & f_3 \\ k_1f_6 & 4g_5 & -6\frac{k_1f_7}{k_2} & 4g_6 & -k_1f_3 & -k_1g_1 & 4f_4 & 6\frac{k_1g_4}{k_2} & 4f_5 & k_1g_2 \\ -k_2g_2 & 4\frac{k_2f_7}{k_1} & -2g_7 & -4\frac{k_2f_7}{k_1} & -k_2g_2 & k_2f_6 & -4\frac{k_2g_4}{k_1} & 2f_8 & 4\frac{k_2g_4}{k_1} & k_2f_6 \\ k_1f_3 & 4g_6 & 6\frac{k_1f_7}{k_2} & 4g_5 & -k_1f_6 & -k_1g_2 & 4f_5 & -6\frac{k_1g_4}{k_2} & 4f_4 & k_1g_1 \\ -g_3 & 4\frac{f_3}{k_1} & -6\frac{g_2}{k_2} & 4\frac{f_2}{k_1} & -g_1 & f_3 & -4\frac{g_2}{k_1} & 6\frac{f_2}{k_2} & -\frac{4g_1}{k_1} & f_1 \end{bmatrix}. \tag{26}$$

Auxiliary variables used in foregoing matrices:

$$
\begin{array}{llll}
a_1 = 3c + c_3, & a_2 = 3c - c_3, & a_3 = c + c_3, & a_4 = c - c_3, \\
b_1 = 3s + s_3, & b_2 = 3s - s_3, & b_3 = s + s_3, & b_4 = s - s_3, \\
d_1 = 3 + 4c_2 + c_4, & d_2 = 1 - c_4, & d_3 = 3 - 4c_2 + c_4, & \\
d_4 = 1 + 4c_2 + 3c_4, & d_5 = 1 - 4c_2 + 3c_4, & & \\
e_1 = 2s_2 + s_4, & e_2 = 2s_2 - s_4, & e_3 = 2s_2 + 3s_4, & e_4 = 2s_2 - 3s_4, \\
f_1 = 10c + 5c_3 + c_5, & f_2 = 2c - c_3 - c_5, & f_3 = 2c - 3c_3 + c_5, & f_4 = c + 2c_3 + c_5, \\
f_5 = c - c_5, & f_6 = 2c - c_3 - c_5, & f_7 = c_3 - c_5, & f_8 = 2c + 3c_3 + 3c_5, \\
g_1 = 2s + 3s_3 + s_5, & g_2 = 2s + s_3 - s_5, & g_3 = 10s - 5s_3 + s_5, & g_4 = s_3 + s_5, \\
g_5 = s - s_5, & g_6 = s - 2s_3 + s_5, & g_7 = 2s - 3s_3 + 3s_5. &
\end{array}
\tag{27}
$$

diag

4.5 MDCC Scaling and Spectral Invariance

If $m \geq 2$, displacement transformation matrices with different MDCC values can be related through simple similarity transformations. To display that property, introduce the diagonal matrices

$$
\begin{aligned}
\widehat{\mathbf{D}}_u^{(2)} &= \mathbf{diag}\,[\,1 \quad k/\sqrt{2} \quad 1 \quad 1 \quad k/\sqrt{2} \quad 1\,], \\
\widehat{\mathbf{D}}_u^{(3)} &= \mathbf{diag}\,[\,1 \quad k/\sqrt{3} \quad k/\sqrt{3} \quad 1 \quad 1 \quad k/\sqrt{3} \quad k/\sqrt{3} \quad 1\,], \\
\widehat{\mathbf{D}}_u^{(4)} &= \mathbf{diag}\,[\,1 \quad k_1/2 \quad k_2/\sqrt{6} \quad k_1/2 \quad 1 \quad 1 \quad k_1/2 \quad k_2/\sqrt{6} \quad k_1/2 \quad 1\,],
\end{aligned}
\tag{28}
$$

which may be completed with $\widehat{\mathbf{D}}_u^{(0)} = \mathbf{I}_2$ and $\widehat{\mathbf{D}}_u^{(1)} = \mathbf{I}_4$. Then

$$
\mathbf{T}_{u\varphi}^{(m)} = \widehat{\mathbf{D}}_u^{(m)}\, \widehat{\mathbf{T}}_{u\varphi}^{(m)}\, \left(\widehat{\mathbf{D}}_u^{(m)}\right)^{-1}, \quad \widehat{\mathbf{T}}_{u\varphi}^{(m)} = \left(\widehat{\mathbf{D}}_u^{(m)}\right)^{-1}\, \mathbf{T}_{u\varphi}^{(m)}\, \widehat{\mathbf{D}}_u^{(m)},
\tag{29}
$$

in which $\mathbf{T}_{u\varphi}^{(m)}$ may have arbitrary MDCC for $m \geq 2$. A consequence of (29) is that *eigenvalues of displacement transformation matrices are independent of the MDCC.* Since $\widehat{\mathbf{T}}_{u\varphi}^{(m)}$ is orthogonal, its eigenvalues lie on the unit circle, a property inherited by all $\mathbf{T}_{u\varphi}^{(m)}$. Only the eigenvectors change.

In the derivation of body force transformations in 4.6, squares of the scaling matrices (28) appear. These are denoted by

$$
\begin{aligned}
\widehat{\mathbf{J}}_u^{(2)} &= \widehat{\mathbf{D}}_u^{(2)} \widehat{\mathbf{D}}_u^{(2)} = \mathbf{diag}\,[\,1 \quad k^2/2 \quad 1 \quad 1 \quad k^2/2 \quad 1\,], \\
\widehat{\mathbf{J}}_u^{(3)} &= \widehat{\mathbf{D}}_u^{(3)} \widehat{\mathbf{D}}_u^{(3)} = \mathbf{diag}\,[\,1 \quad k^2/3 \quad k^2/3 \quad 1 \quad 1 \quad k^2/3 \quad k^2/3 \quad 1\,], \\
\widehat{\mathbf{J}}_u^{(4)} &= \widehat{\mathbf{D}}_u^{(4)} \widehat{\mathbf{D}}_u^{(4)} = \mathbf{diag}\,[\,1 \quad k_1^2/4 \quad k_2^2/6 \quad k_1^2/4 \quad 1 \quad 1 \quad k_1^2/4 \quad k_2^2/6 \quad k_1^2/4 \quad 1\,].
\end{aligned}
\tag{30}
$$

In some computations (e.g., finding invariants) it is convenient to use flat displacement vectors, in which all MDCC are one. To transform to other MDCC, introduce the diagonal matrices

$$
\begin{aligned}
\overline{\mathbf{D}}_u^{(2)} &= \mathbf{diag}\,[\,1 \quad k \quad 1 \quad 1 \quad k \quad 1\,], \\
\overline{\mathbf{D}}_u^{(3)} &= \mathbf{diag}\,[\,1 \quad k \quad k \quad 1 \quad 1 \quad k \quad k \quad 1\,], \\
\overline{\mathbf{D}}_u^{(4)} &= \mathbf{diag}\,[\,1 \quad k_1 \quad k_2 \quad k_1 \quad 1 \quad 1 \quad k_1 \quad k_2 \quad k_1 \quad 1\,],
\end{aligned}
\tag{31}
$$

completed with $\mathbf{I}_2$ and $\mathbf{I}_4$ for $m = 0, 1$. Then

$$\mathbf{T}^{(m)}_{u\varphi} = \overline{\mathbf{D}}^{(m)}_u \, \overline{\mathbf{T}}^{(m)}_{u\varphi} \, \left(\overline{\mathbf{D}}^{(m)}_u\right)^{-1}, \tag{32}$$

The matrix on the left hand side may have arbitrary MDCC for $m \ge 2$. x

4.6 Body Force Transformations

By stipulation $\mathbf{u}^{(m)}$ and $\mathbf{b}^{(m)}$ are conjugate, whence $(\mathbf{b}^{(m)}_\varphi)^T \, \mathbf{u}^{(m)}_\varphi = (\mathbf{b}^{(m)}_0)^T \, \mathbf{u}^{(m)}_0$ for any m and φ. On inserting $\mathbf{u}^{(m)}_\varphi = \mathbf{T}^{(m)}_{u\varphi} \, \mathbf{u}^{(m)}_0$ and $\mathbf{b}^{(m)}_\varphi = \mathbf{T}^{(m)}_{b\varphi} \, \mathbf{b}^{(m)}_0$ one finds that $(\mathbf{T}^{(m)}_{b\varphi})^T \, \mathbf{T}^{(m)}_{u\varphi} = \mathbf{I}_{2m+2}$, whence

$$\mathbf{T}^{(m)}_{b\varphi} = \left(\mathbf{T}^{(m)}_{u\varphi}\right)^{-T}. \tag{33}$$

If both displacement and body-force C-vectors are orthogonal, the transformation matrices coincide: $\widehat{\mathbf{T}}^{(m)}_{b\varphi} = \widehat{\mathbf{T}}^{(m)}_{u\varphi}$, because $\widehat{\mathbf{T}}^{(m)}_{u\varphi}$ is orthogonal. This is always the case for $m = 0, 1$. As the inverse of $\mathbf{T}^{(m)}_{u\varphi}$ can be obtained by replacing φ by $-\varphi$, for $m \ge 2$ we may use the rule

$$\mathbf{T}^{(m)}_b(\varphi) = \left(\mathbf{T}^{(m)}_u(-\varphi)\right)^T, \quad \mathbf{T}^{(m)}_u(\varphi) = \left(\mathbf{T}^{(m)}_b(-\varphi)\right)^T. \tag{34}$$

A more complicated relation that uses diagonal scaling is

$$\mathbf{T}^{(m)}_{b\varphi} = \left(\widehat{\mathbf{J}}^{(m)}_u\right)^{-1} \mathbf{T}^{(m)}_{u\varphi} \, \widehat{\mathbf{J}}^{(m)}_u, \quad \mathbf{T}^{(m)}_{u\varphi} = \widehat{\mathbf{J}}^{(m)}_u \, \mathbf{T}^{(m)}_{b\varphi} \left(\widehat{\mathbf{J}}^{(m)}_u\right)^{-1}, \tag{35}$$

in which the $\widehat{\mathbf{J}}^{(m)}_u$ are given by (30). Since (35) are similarity transformations, $\mathbf{T}_{u\varphi}$ and $\mathbf{T}_{b\varphi}$ have the same eigenvalues for any MDCC choice.

4.7 Transformations for Binomial C-Vectors

Pairing a binomial C-vector for displacements with a flat C-vector for body forces is useful in FEM work as it avoids carrying square roots along. The transformation matrices for $m = 0$ and $m = 1$ are still given by (22) and (23). For $m = 2$ through 4 they are listed below.

$$\mathbf{T}^{(2)}_u = \mathbf{T}^{(2)}_u(2) = \tfrac{1}{4} \begin{bmatrix} a_1 & b_3 & a_4 & b_3 & a_4 & b_2 \\ -2b_3 & 2a_3 & 2b_3 & -2a_4 & -2b_4 & 2a_4 \\ a_4 & -b_3 & a_1 & b_2 & -a_4 & b_3 \\ -b_3 & -a_4 & -b_2 & a_1 & b_3 & a_4 \\ 2a_4 & 2b_4 & -2a_4 & -2b_3 & 2a_3 & 2b_3 \\ -b_2 & a_4 & -b_3 & a_4 & -b_3 & a_1 \end{bmatrix}, \tag{36}$$

$$\mathbf{T}^{(3)}_u = \mathbf{T}^{(3)}_u(3) = \tfrac{1}{8} \begin{bmatrix} d_1 & e_1 & d_2 & e_2 & e_1 & d_2 & e_2 & d_3 \\ -3\,e_1 & d_4 & e_3 & 3\,d_2 & -3\,d_2 & -e_4 & -d_5 & 3\,e_2 \\ 3\,d_2 & -e_3 & d_4 & 3\,e_1 & 3\,e_2 & d_5 & -e_4 & 3\,d_2 \\ -e_2 & d_2 & -e_1 & d_1 & -d_3 & e_2 & -d_2 & e_1 \\ -e_1 & -d_2 & -e_2 & -d_3 & d_1 & e_1 & d_2 & e_2 \\ 3\,d_2 & e_4 & d_5 & -3\,e_2 & -3\,e_1 & d_4 & e_3 & 3\,d_2 \\ -3\,e_2 & -d_5 & e_4 & -3\,d_2 & 3\,d_2 & -e_3 & d_4 & 3\,e_1 \\ d_3 & -e_2 & d_2 & -e_1 & -e_2 & d_2 & -e_1 & d_1 \end{bmatrix}, \tag{37}$$

$$\mathbf{T}_u^{(4)} = \mathbf{T}_u^{(4)}(4,6) =$$

$$= \tfrac{1}{16}\begin{bmatrix} f_1 & g_1 & f_2 & g_2 & f_3 & g_1 & f_2 & g_2 & f_3 & g_3 \\ -4g_1 & 4f_4 & 4g_4 & 4f_5 & 4g_2 & -4f_6 & -4g_5 & 4f_7 & -4g_6 & 4f_3 \\ 6f_6 & -6g_4 & 2f_8 & 6g_4 & 6f_6 & 6g_2 & -6f_7 & 2g_7 & 6f_7 & 6g_2 \\ -4g_2 & 4f_5 & -4g_4 & 4f_4 & 4g_1 & -4f_3 & -4g_6 & -4f_7 & -4g_5 & 4f_6 \\ f_3 & -g_2 & f_2 & -g_1 & f_1 & g_3 & -f_3 & g_2 & -f_6 & g_1 \\ -g_1 & -f_6 & -g_2 & -f_3 & -g_3 & f_1 & g_1 & f_2 & g_2 & f_3 \\ 4f_6 & 4g_5 & -4f_7 & 4g_6 & -4f_3 & -4g_1 & 4f_4 & 4g_4 & 4f_5 & 4g_2 \\ -6g_2 & 6f_7 & -2g_7 & -6f_7 & -6g_2 & 6f_6 & -6g_4 & 2f_8 & 6g_4 & 6f_6 \\ 4f_3 & 4g_6 & 4f_7 & 4g_5 & -4f_6 & -4g_2 & 4f_5 & -4g_4 & 4f_4 & 4g_1 \\ -g_3 & f_3 & -g_2 & f_2 & -g_1 & f_3 & -g_2 & f_2 & -g_1 & f_1 \end{bmatrix}. \tag{38}$$

The auxiliary variables a_i through g_i are listed in (27). For body force transformations use (33) adjusting the MDCC as appropriate.

4.8 Linear Invariants

A *linear invariant* of the displacement derivative tensor of order m is a scalar-valued, linear combination of its C-vector components, expressable as a dot product:

$$I_u^{(m)} = \left(\mathbf{w}_u^{(m)}\right)^T \mathbf{u}^{(m)}, \quad \mathbf{w}_u^{(m)} \neq \mathbf{0}, \tag{39}$$

that stays unchanged when $\mathbf{u}^{(m)} = \mathbf{u}_0^{(m)}$ is replaced by $\mathbf{u}_\varphi^{(m)} = \mathbf{T}_{u\varphi}^{(m)}\,\mathbf{u}_0^{(m)}$, for any φ. Here $\mathbf{w}_u^{(m)}$ is a *invariant weight vector* of dimension $2m+2$, which is independent of φ and may be scaled by a nonzero factor. If the only solution is $\mathbf{w}_u^{(m)} = \mathbf{0}$, there are no linear invariants for order m.

If $\mathbf{w}_u^{(m)}$ has n_I independent free parameters, denoted as η_i, $i = 1, \ldots n_I$, an *invariant basis* can be established by assigning n_I sets of numerical values. The process, called *instantiation*, results in n *base invariants* denoted by $I_{u\textcircled{i}}^{(m)}$, in which the circled number is an invariant index. Base invariants must be linearly independent, whence any other invariant follows by linear combination. The goal should be to select base invariants that possess as much physical significance as possible.

What happens for $m \geq 2$ if the MDCC in $\mathbf{u}^{(m)}$ are changed? Then $\mathbf{w}^{(m)}$ changes so that $(\mathbf{w}_u^{(m)})^T\,\mathbf{u}^{(m)}$ remains invariant; this is easily verified using the scaling matrices introduced in 4.5. Thus orthogonal, binomial or flat C-vectors may be indifferently used in (39). For the ensuing computtaions we will select *flat* vectors for displacements: $\mathbf{u}^{(m)} = \overline{\mathbf{u}}^{(m)}$, and *binomial* vectors for body forces, $\mathbf{b}^{(m)} = \overline{\mathbf{b}}^{(m)}$, with the overbars omitted to reduce clutters. This has two advantages: the $\mathbf{w}^{(m)}$ solutions are identical, and square roots are avoided.

The generic algebraic statement for displacements is $(\mathbf{w}_u^{(m)})^T\,\mathbf{T}_{u\varphi}^{(m)}\mathbf{u}_0^{(m)} = (\mathbf{w}_u^{(m)})^T\mathbf{u}_0^{(m)}$. Because $\mathbf{u}_0^{(m)}$ is arbitrary, we must have

$$\left(\mathbf{T}_{u\varphi}^{(m)}\right)^T\,\mathbf{w}_u^{(m)} = \mathbf{w}_u^{(m)}. \tag{40}$$

This can be translated to a spectral condition: $\mathbf{T}_{u\varphi}^{(m)}$ *must have a positive real unit eigenvalue*. If so, the corresponding eigenvector is $\mathbf{w}_u^{(m)}$ up to a scaling factor. If the $+1$ eigenvalue has multiplicity $n > 1$,

$\mathbf{w}_u^{(m)}$ is a linear combination of the corresponding eigenvectors, and there are $n_{Iu}^{(m)} = n$ independent linear invariants.

Inspection of the eigenvalues (144) in Appendix A shows that only orders $m = 1$ and $m = 3$ possess linear invariants. In both cases $+1$ is a double eigenvalue. Invariant weighting vectors are readily built from the corresponding eigenvectors give in (146)–(152), by looking at the first two rows of $\mathbf{V}_L^{(1)}$ and $\mathbf{V}_L^{(3)}$ or, alternatively, the first two columns of $\mathbf{V}_R^{(1)}$ and $\mathbf{V}_R^{(3)}$. Upon applying appropriate scale factors we get

$$\begin{aligned}\mathbf{w}_u^{(1)} &= [\,\eta_1 \quad -\eta_2 \quad \eta_2 \quad \eta_1\,]^T,\\ \mathbf{w}_u^{(3)} &= [\,\eta_1 \quad -\eta_2 \quad \eta_1 \quad -\eta_2 \quad \eta_2 \quad \eta_1 \quad \eta_2 \quad \eta_1\,]^T,\end{aligned} \tag{41}$$

in which η_i are free parameters. Physically meaningful base invariants are obtained by instantiating $\eta_1 = 1, \eta_2 = 0$ and $\eta_1 = 0, \eta_2 = 1$. This gives

$$\begin{aligned}I_{u\textcircled{1}}^{(1)} &= u_{1,1} + u_{2,2} = \mathrm{Tr}(\boldsymbol{\epsilon}) = \epsilon_v, \quad I_{u\textcircled{2}}^{(1)} = -u_{1,2} + u_{2,1} = 2\omega_3,\\ I_{u\textcircled{1}}^{(3)} &= u_{1,111} + u_{1,122} + u_{2,112} + u_{2,222} = \Delta\, I_{u\textcircled{1}}^{(1)} = \Delta\,\epsilon_v,\\ I_{u\textcircled{2}}^{(3)} &= -u_{1,112} - u_{1,222} + u_{2,111} + u_{2,122} = \Delta\, I_{u\textcircled{2}}^{(1)} = 2\Delta\,\omega_3.\end{aligned} \tag{42}$$

These have well known interpretations in terms of deformations. $I_{u\textcircled{1}}^{(1)}$ is the trace of the infinitesimal plane strain tensor, which is also the volumetric strain e in terms of displacements. $I_{u\textcircled{2}}^{(1)}$ is twice the infinitesimal rotation about x_3. Finally, $I_{u\textcircled{1}}^{(3)}$ and $I_{u\textcircled{2}}^{(3)}$ are their Laplacians. Exactly the same four base invariants emerge for body forces using binomial C-vectors, the only change being the replacement of u by b. Their physical meaning, however, is less obvious.

4.9 Quadratic Invariants

A *quadratic invariant* of the displacement derivative tensor of order m is a scalar-valued combination of its C-vector components expressible as a quadratic form

$$J_u^{(m)} = \left(\mathbf{u}^{(m)}\right)^T \mathbf{W}_u^{(m)}\, \mathbf{u}^{(m)}, \quad \text{where} \quad \mathbf{W}_u^{(m)} \neq \mathbf{0}, \tag{43}$$

that remains unchanged when $\mathbf{u}^{(m)}$ is replaced by $\mathbf{u}_\varphi^{(m)} = \mathbf{T} - \varphi^{(m)}\mathbf{u}_0^{(m)}$ for any φ. Here $\mathbf{W}_u^{(m)}$ is a nonzero *symmetric* square matrix called an *invariant weight matrix*, which is independent of φ. Again the choice of MDCC in $\mathbf{u}^{(m)}$ is irrelevant, and for convenience we often use flat vectors with overbars suppressed. Replacing $\mathbf{u}_\varphi^{(m)} = \mathbf{T}_\varphi^{(m)}\, \mathbf{u}^{(m)}$ in (43) and noting that $\mathbf{u}^{(m)}$ is arbitrary gives

$$\left(\mathbf{T}_{u\varphi}^{(m)}\right)^T \mathbf{W}_u^{(m)}\, \mathbf{T}_{u\varphi}^{(m)} - \mathbf{W}_u^{(m)} = \mathbf{0}, \quad \text{or} \quad \left(\mathbf{T}_{u\varphi}^{(m)}\right)^T \mathbf{W}_u^{(m)}\, \mathbf{T}_{u\varphi}^{(m)} = \mathbf{W}_u^{(m)}. \tag{44}$$

This is a homogeneous matrix equation of Sylvester type [5, 16, 22] that appears in control theory, with $\mathbf{W}^{(m)} \neq \mathbf{0}$ as unknown. As in the case of linear invariants, the general solution $\mathbf{W}^{(m)}$ may contain n_J free parameters that can be instantiated to exhibit an invariant basis. If there are many parameters, showing the solution $\mathbf{W}^{(m)}$ is often sufficient, leaving instantiation to further work.

Nothing so far dictates that the entries of $\mathbf{W}^{(m)}$ must be real. But if complex, decompose as $\mathbf{W}^{(m)} = \mathbf{W}_r^{(m)} + j\,\mathbf{W}_c^{(m)}$, in which $\mathbf{W}_r^{(m)}$ and $\mathbf{W}_i^{(m)}$ are symmetric real. Substitution into (44) shows that both components separately satisfy the equation. So it suffices to consider real entries.

The nontrivial solution of (44) contain free parameters labeled η_i, $i = 1, \ldots n_J$, in which $n_J = 1, 4, 5, 8, 9$ for $m = 0, 1, 2, 3, 4$, respectively. The following results were obtained by splitting $\mathbf{W}_u^{(m)} = \mathbf{W}_{uB}^{(m)} + \mathbf{W}_{uX}^{(m)}$, with entry patterns `"B"` and `"X"`, respectively (see Table 3), and combining.

$$\mathbf{W}_u^{(0)} = \begin{bmatrix} \eta_1 & 0 \\ 0 & \eta_1 \end{bmatrix}, \tag{45}$$

$$\mathbf{W}_u^{(1)} = \begin{bmatrix} \eta_1+\eta_2+\eta_3 & \eta_4 & -\eta_4 & \eta_1 \\ \eta_4 & \eta_2 & \eta_3 & \eta_4 \\ -\eta_4 & \eta_3 & \eta_2 & -\eta_4 \\ \eta_1 & \eta_4 & -\eta_4 & \eta_1+\eta_2+\eta_3 \end{bmatrix} \tag{46}$$

$$\mathbf{W}_u^{(2)} = \begin{bmatrix} \eta_1+\eta_2+\eta_3 & \eta_5 & \eta_4 & -\eta_5 & \eta_1 & 0 \\ \eta_5 & \eta_1+\eta_2+2\eta_3-2\eta_4 & \eta_5 & \eta_2 & 0 & \eta_1 \\ \eta_4 & \eta_5 & \eta_3 & 0 & \eta_2 & \eta_5 \\ -\eta_5 & \eta_2 & 0 & \eta_3 & -\eta_5 & \eta_4 \\ \eta_1 & 0 & \eta_2 & -\eta_5 & \eta_1+\eta_2+2\eta_3-2\eta_4 & -\eta_5 \\ 0 & \eta_1 & \eta_5 & \eta_4 & -\eta_5 & \eta_1+\eta_2+\eta_3 \end{bmatrix} \tag{47}$$

$$\mathbf{W}_u^{(3)} = \begin{bmatrix} \eta_1 & \eta_8 & \eta_2 & -\eta_9 & \eta_3 & -\eta_8 & \eta_4 & \eta_9 & \xi_{16} & 0 \\ \eta_8 & \xi_{11} & \xi_{12} & \xi_{13} & -\eta_9 & \eta_5 & -\xi_{14} & \xi_{15} & 0 & \xi_{16} \\ \eta_2 & \xi_{12} & \xi_{17} & \xi_{12} & \eta_6 & \eta_9 & \xi_{18} & 0 & \xi_{15} & -\eta_9 \\ -\eta_9 & \xi_{13} & \xi_{12} & \xi_{19} & \eta_8 & \eta_7 & 0 & \xi_{18} & \xi_{14} & \eta_4 \\ \eta_3 & -\eta_9 & \eta_6 & \eta_8 & \xi_{20} & 0 & \eta_7 & -\eta_9 & \eta_5 & \eta_8 \\ -\eta_8 & \eta_5 & \eta_9 & \eta_7 & 0 & \xi_{20} & -\eta_8 & \eta_6 & \eta_9 & \eta_3 \\ \eta_4 & -\xi_{14} & \xi_{18} & 0 & \eta_7 & -\eta_8 & \xi_{19} & \xi_{21} & \xi_{13} & \eta_9 \\ \eta_9 & \xi_{15} & 0 & \xi_{18} & -\eta_9 & \eta_6 & \xi_{21} & \xi_{17} & \xi_{21} & \eta_2 \\ \xi_{16} & 0 & \xi_{15} & \xi_{14} & \eta_5 & \eta_9 & \xi_{13} & \xi_{21} & \xi_{11} & -\eta_8 \\ 0 & \xi_{16} & -\eta_9 & \eta_4 & \eta_8 & \eta_3 & \eta_9 & \eta_2 & -\eta_8 & \eta_1 \end{bmatrix}, \tag{48}$$

$$\mathbf{W}_u^{(4)} = \begin{bmatrix} \eta_1 & \eta_8 & \eta_2 & -\eta_9 & \eta_3 & -\eta_8 & \eta_4 & \eta_9 & \xi_{16} & 0 \\ \eta_8 & \xi_{11} & \xi_{12} & \xi_{13} & -\eta_9 & \eta_5 & -\xi_{14} & \xi_{15} & 0 & \xi_{16} \\ \eta_2 & \xi_{12} & \xi_{17} & \xi_{12} & \eta_6 & \eta_9 & \xi_{18} & 0 & \xi_{15} & -\eta_9 \\ -\eta_9 & \xi_{13} & \xi_{12} & \xi_{19} & \eta_8 & \eta_7 & 0 & \xi_{18} & \xi_{14} & \eta_4 \\ \eta_3 & -\eta_9 & \eta_6 & \eta_8 & \xi_{20} & 0 & \eta_7 & -\eta_9 & \eta_5 & \eta_8 \\ -\eta_8 & \eta_5 & \eta_9 & \eta_7 & 0 & \xi_{20} & -\eta_8 & \eta_6 & \eta_9 & \eta_3 \\ \eta_4 & -\xi_{14} & \xi_{18} & 0 & \eta_7 & -\eta_8 & \xi_{19} & \xi_{21} & \xi_{13} & \eta_9 \\ \eta_9 & \xi_{15} & 0 & \xi_{18} & -\eta_9 & \eta_6 & \xi_{21} & \xi_{17} & \xi_{21} & \eta_2 \\ \xi_{16} & 0 & \xi_{15} & \xi_{14} & \eta_5 & \eta_9 & \xi_{13} & \xi_{21} & \xi_{11} & -\eta_8 \\ 0 & \xi_{16} & -\eta_9 & \eta_4 & \eta_8 & \eta_3 & \eta_9 & \eta_2 & -\eta_8 & \eta_1 \end{bmatrix} \tag{49}$$

Auxiliary variable in the last two matrices:

$$\begin{array}{lll}
\xi_1 = 3\eta_1{-}2\eta_2{-}\eta_3{-}\eta_6, & \xi_2 = 2\eta_7 + \eta_8, & \xi_3 = -\eta_7{-}2\eta_8, \\
\xi_4 = \eta_2{+}2\eta_3{-}3\eta_4{-}\eta_5, & \xi_5 = 3\eta_1{-}2\eta_3{-}2\eta_5{-}2\eta_6, & \xi_6 = -2\eta_2{+}3\eta_4{+}2\eta_5{+}2\eta_6, \\
\xi_7 = \eta_7{+}2\eta_8, & \xi_8 = \eta_1{-}\eta_3{-}\eta_6, & \xi_9 = \eta_2{-}\eta_4{-}\eta_5, \\
\xi_{10} = -2\eta_7{-}\eta_8, & \xi_{11} = 4\eta_1{-}2\eta_2{-}\eta_4{-}\eta_5, & \xi_{12} = 3\eta_8{+}\eta_9, \\
\xi_{13} = \eta_2{-}4\eta_3{+}\eta_6, & \xi_{14} = 2(\eta_8{+}\eta_9), & \xi_{15} = -2\eta_2{+}3\eta_4{+}2\eta_6{+}3\eta_7, \\
\xi_{16} = \eta_2{-}\eta_6{-}\eta_7, & \xi_{17} = 6\eta_1{-}2\eta_2{+}6\eta_3{-}3\eta_4{-}3\eta_5{-}2\eta_6, & \xi_{18} = \eta_2{+}3\eta_5{-}\eta_6{-}3\eta_7, \\
\xi_{19} = 4\eta_1{-}3\eta_4{-}3\eta_5{-}2\eta_6, & \xi_{20} = \eta_1{-}\eta_4{-}\eta_5, & \xi_{21} = -3\eta_8{-}\eta_9.
\end{array} \tag{50}$$

For $m{=}0$ we have $\mathbf{W}_u^{(0)} = \eta_1\,\mathbf{I}_2$; see (45). Setting $\eta_1 = 1$ gives the squared Euclidean length of the displacement vector as the only zero order invariant:

$$J_{u\textcircled{1}}^{(0)} = u_1^2 + u_2^2. \tag{51}$$

For $m{=}1$, the $\mathbf{W}_u^{(1)}$ of (46) has 4 free parameters. Setting those to $\{1,0,0,0\}$, $\{0,1,-1,0\}$, $\{0,0,0,-\frac{1}{2}\}$ and $\{\frac{1}{2},0,-\frac{1}{2},0\}$ gives the base invariants

$$\begin{aligned}
J_{u\textcircled{1}}^{(1)} &= (u_{1,1} + u_{2,2})^2 = I_{u\textcircled{1}}^{(1)}\, I_{u\textcircled{1}}^{(1)} = \epsilon_v^2, \\
J_{u\textcircled{2}}^{(1)} &= (-u_{1,2} + u_{2,1})^2 = I_{u\textcircled{2}}^{(1)}\, I_{u\textcircled{2}}^{(1)} = 4\omega_3^2, \\
J_{u\textcircled{3}}^{(1)} &= (u_{1,1} + u_{2,2})(-u_{1,2} + u_{2,1}) = I_{u\textcircled{1}}^{(1)}\, I_{u\textcircled{2}}^{(1)} = 2\epsilon_v\,\omega_3 \\
J_{u\textcircled{4}}^{(1)} &= u_{1,1}u_{2,2} - u_{1,2}u_{2,1} = \det(\boldsymbol{\epsilon}).
\end{aligned} \tag{52}$$

The first three are related to the two first order invariants listed in (42) The last one is the determinant of the infinitesimal 2D strain tensor.

For $m{=}2$ the matrix $\mathbf{W}_u^{(2)}$ of (47) has 5 free parameters, Setting those to $\{0,0,1,1,0\}$, $\{0,0,1,-1,0\}$, $\{\frac{1}{2},\frac{1}{2},-1,-\frac{1}{2},0\}$, $\{0,0,0,0,-\frac{1}{2}\}$, and $\{0,0,1,0,0\}$ gives the base invariants

$$\begin{aligned}
J_{u\textcircled{1}}^{(2)} &= (u_{1,11} + u_{1,22})^2 + (u_{2,11} + u_{2,22})^2 = \Delta u_1 \Delta u_1 + \Delta u_2 \Delta u_2, \\
J_{u\textcircled{2}}^{(2)} &= (u_{1,12} - u_{2,11})^2 + (u_{1,22} - u_{2,12})^2 = 4\,(\omega_{3,1}^2 + \omega_{3,2}^2), \\
J_{u\textcircled{3}}^{(2)} &= (u_{1,11} + u_{1,22})(u_{2,12} - u_{1,22}) - (u_{2,11} + u_{2,22})(u_{2,11} - u_{1,12}) \\
&\quad = 2\,(\Delta\,u_1\,\omega_{3,2} - \Delta\,u_2\,\omega_{3,1}), \\
J_{u\textcircled{4}}^{(2)} &= (u_{1,11} + u_{1,22})(u_{2,11} - u_{1,12}) + (u_{2,11} + u_{2,22})(u_{2,12} - u_{1,22}) \\
&\quad = 2\,(\Delta\,u_1\,\omega_{3,1} + \Delta\,u_2\,\omega_{3,2}), \\
J_{u\textcircled{5}}^{(2)} &= u_{1,11}^2 + 2u_{1,12}^2 + u_{1,22}^2 + u_{2,11}^2 + 2u_{2,12}^2 + u_{2,22}^2.
\end{aligned} \tag{53}$$

in which Δ denotes the Laplacian. The first four have obvious connections to physics. The fifth one lacks obvious meaning, but is nonetheless interesting in that it is generated by a diagonal weighting matrix: **diag** $[\,1\;\;2\;\;1\;\;1\;\;2\;\;1\,]$. Another instance linked to volumetric strain is $\epsilon_{v,1}^2 + \epsilon_{v,2}^2$. This is not linearly independent, however, because it can be expressed as $J_{u\textcircled{1}}^{(2)} + J_{u\textcircled{2}}^{(2)} - J_{u\textcircled{4}}^{(2)}$.

Physical instantiation of the invariants for $m = 3$ and $m = 4$ has not been investigated because of the increased complexity of the $\mathbf{W}$ matrices. The foregoing expressions can be used to formally derive equivalent body force invariants, although their physical interpretation is nebulous.

5 Strains and Stresses

Strains and stresses are treated together as conjugate entities. Derivatives of order up to 3 are investigated since that meets FIC functional needs. The striking difference with displacements are *strain compatibility constraints* that emerge for derivative orders 2 and higher. These can be introduced through reduced C-vectors as shown in 5.2 below. In the ensuing derivations all derivatives up to the indicated order are assumed to exist, and all mixed derivatives commute.

5.1 C-Vectors of Strains and Stresses

In what follows $\gamma_{12} = \epsilon_{12} + \epsilon_{21} = 2\epsilon_{12}$ denotes the engineering shear strain. Standard C-vectors of the infinitesimal strain field and its first 3 derivatives are denoted by

$$\boldsymbol{\epsilon} = \boldsymbol{\epsilon}^{(0)} = [\, \epsilon_{11} \quad \epsilon_{22} \quad 2\epsilon_{12} \,]^T = [\, \epsilon_{11} \quad \epsilon_{22} \quad \gamma_{12} \,]^T , \tag{54}$$

$$\boldsymbol{\epsilon}^{(1)} = [\, \epsilon_{11,1} \quad \epsilon_{11,2} \quad \epsilon_{22,1} \quad \epsilon_{22,2} \quad \gamma_{12,1} \quad \gamma_{12,2} \,]^T , \tag{55}$$

$$\boldsymbol{\epsilon}^{(2)}(g) = [\, \epsilon_{11,11} \quad g\epsilon_{11,12} \quad \epsilon_{11,22} \quad \epsilon_{22,11} \quad g\epsilon_{22,12} \quad \epsilon_{22,22} \quad \gamma_{12,11} \quad g\gamma_{12,12} \quad \gamma_{12,22} \,]^T , \tag{56}$$

$$\begin{aligned} \boldsymbol{\epsilon}^{(3)}(g) = [\, &\epsilon_{11,111} \quad g\epsilon_{11,112} \quad g\epsilon_{11,122} \quad \epsilon_{11,222} \quad \epsilon_{22,111} \quad g\epsilon_{22,112} \quad g\epsilon_{22,122} \quad \epsilon_{22,222} \\ &\gamma_{12,111} \quad g\gamma_{12,112} \quad g\gamma_{12,122} \quad \gamma_{12,222} \,]^T . \end{aligned} \tag{57}$$

The corresponding C-vectors of the Cauchy stress field and its first 3 derivatives are

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}^{(0)} = [\, \sigma_{11} \quad \sigma_{22} \quad \sigma_{12} \,]^T , \tag{58}$$

$$\boldsymbol{\sigma}^{(1)} = [\, \sigma_{11,1} \quad \sigma_{11,2} \quad \sigma_{22,1} \quad \sigma_{22,2} \quad \sigma_{12,1} \quad \sigma_{12,2} \,]^T , \tag{59}$$

$$\boldsymbol{\sigma}^{(2)}(g) = \big[\, \sigma_{11,11} \quad \frac{2}{g}\sigma_{11,12} \quad \sigma_{11,22} \quad \sigma_{22,11} \quad \frac{2}{g}\sigma_{22,12} \quad \sigma_{22,22} \quad \sigma_{12,11} \quad \frac{2}{g}\sigma_{12,12} \quad \sigma_{12,22} \,\big]^T , \tag{60}$$

$$\begin{aligned} \boldsymbol{\sigma}^{(3)}(g) = \big[\, &\sigma_{11,111} \quad \frac{3}{g}\sigma_{11,112} \quad \frac{3}{g}\sigma_{11,122} \quad \sigma_{11,222} \quad \sigma_{22,111} \quad \frac{3}{g}\sigma_{22,112} \quad \frac{3}{g}\sigma_{22,122} \quad \sigma_{22,222} \\ &\sigma_{12,111} \quad \frac{3}{g}\sigma_{12,112} \quad \frac{3}{g}\sigma_{12,122} \quad \sigma_{12,222} \,\big]^T . \end{aligned} \tag{61}$$

In C-vectors of orders 2 and 3, g is a positive real number known as a *mixed derivative casting coefficient*, or MDCC for short. Three useful specializations: orthogonal, binomial and flat casting, are obtained by selecting g as shown in Table 5.1. This table also summarizes pertinent notation. For $m \le 1$, g does not appear.

Inner products of binomial $\boldsymbol{\epsilon}^{(m)}$ and flat $\boldsymbol{\sigma}^{(m)}$ are conjugate; e.g., for $m = 0$:

$$\begin{aligned} \boldsymbol{\sigma}^T \boldsymbol{\epsilon} &= \sigma_{11}\epsilon_{11} + \sigma_{22}\epsilon_{22} + \sigma_{12}\gamma_{12} = \sigma_{11}\epsilon_{11} + \sigma_{22}\epsilon_{22} + 2\,\sigma_{12}\epsilon_{12} \\ &= \sigma_{11}\epsilon_{11} + \sigma_{22}\epsilon_{22} + \sigma_{12}\epsilon_{12} + \sigma_{21}\epsilon_{21} = \sigma_{\alpha\beta}\epsilon_{\alpha\beta}. \end{aligned} \tag{62}$$

This is twice the strain energy density from a stress-free state. That kind of physical interpretation does not hold for $m \ge 1$.

Table 2: **Instantiation of Strain and Stress C-Vectors for Orders 2,3.**

Field	**Order**	**Orthogonal Casting**		**Binomial Casting**		**Flat Casting**	
	m	Symbol	MDCC	Symbol	MDCC	Symbol	MDCC
Strain	2	$\widehat{\boldsymbol{\epsilon}}^{(2)}$	$g=\sqrt{2}$	$\boldsymbol{\epsilon}^{(2)}$	$g=2$	$\bar{\boldsymbol{\epsilon}}^{(2)}$	$g=1$
Strain	3	$\widehat{\boldsymbol{\epsilon}}^{(3)}$	$g=\sqrt{3}$	$\boldsymbol{\epsilon}^{(3)}$	$g=3$	$\bar{\boldsymbol{\epsilon}}^{(3)}$	$g=1$
Stress	2	$\widehat{\boldsymbol{\sigma}}^{(2)}$	$g=\sqrt{2}$	$\bar{\boldsymbol{\sigma}}^{(2)}$	$g=2$	$\boldsymbol{\sigma}^{(2)}$	$g=1$
Stress	3	$\widehat{\boldsymbol{\sigma}}^{(3)}$	$g=\sqrt{3}$	$\bar{\boldsymbol{\sigma}}^{(3)}$	$g=3$	$\boldsymbol{\sigma}^{(3)}$	$g=1$
Conjugate pairs: $\boldsymbol{\epsilon}^{(m)}$ with $\boldsymbol{\sigma}^{(m)}$, $\widehat{\boldsymbol{\epsilon}}^{(m)}$ with $\widehat{\boldsymbol{\sigma}}^{(m)}$, and $\bar{\boldsymbol{\epsilon}}^{(m)}$ with $\bar{\boldsymbol{\sigma}}^{(m)}$. Orthogonal $\widehat{\boldsymbol{\epsilon}}^{(m)}$ and $\widehat{\boldsymbol{\sigma}}^{(m)}$ are self-conjugate.							

5.2 Augmented, Constrained and Reduced Strain C-Vectors

To render the $m = 0$ strain-displacement transformation matrix square, $\boldsymbol{\epsilon}^{(0)}$ is *augmented* with the infinitesimal rigid-body rotation about x_3:

$$\boldsymbol{\epsilon}_a^{(0)} = [\,\epsilon_{11}\;\;\epsilon_{22}\;\;\gamma_{12}\;\;2\omega_3\,]^T\,. \tag{63}$$

in which $\omega_3 = \frac{1}{2}(u_{2,1} - u_{1,2})$. The scaling by 2 eliminates the $\frac{1}{2}$ fraction. To accomplish a similar equilibration for $m = 2, 3$, it is necessary to *remove* entries. This can be done using kinematic constraints. For $m = 2$, the shear strain mixed derivative $\gamma_{12,12}$ is not linearly independent of the remaining ones when strains are computed from displacement derivatives:

$$\gamma_{12,12} = u_{1,122} + u_{2,112} = \epsilon_{11,22} + \epsilon_{22,11}. \tag{64}$$

This is the well known 2D strain compatibility condition. Constraints for $m = 3$ emerge on differentiating (64) with respect to x_1 and x_2:

$$\gamma_{12,112} = u_{1,1122} + u_{2,1112} = \epsilon_{11,122} + \epsilon_{22,111}, \quad \gamma_{12,122} = u_{1,1222} + u_{2,1122} = \epsilon_{11,222} + \epsilon_{22,112}. \tag{65}$$

Imposing these contraints explicitly on (56) and (57) yields *constrained* strain vectors, which are identified with a c subscript:

$$\boldsymbol{\epsilon}_c^{(2)}(g) = [\,\epsilon_{11,11}\;\;g\epsilon_{11,12}\;\;\epsilon_{11,22}\;\;\epsilon_{22,11}\;\;g\epsilon_{22,12}\;\;\epsilon_{22,22}\;\;\gamma_{12,11}\;\;g(\epsilon_{11,22}+\epsilon_{22,11})\;\;\gamma_{12,22}\,]^T\,, \tag{66}$$

$$\begin{aligned}\boldsymbol{\epsilon}_c^{(3)}(g) = [\,&\epsilon_{11,111}\;\;g\epsilon_{11,112}\;\;g\epsilon_{11,122}\;\;\epsilon_{11,222}\;\;\epsilon_{22,111}\;\;g\epsilon_{22,112}\;\;g\epsilon_{22,122}\\ &\epsilon_{22,222}\;\;\gamma_{12,111}\;\;g(\epsilon_{11,122}+\epsilon_{22,111})\;\;g(\epsilon_{11,222}+\epsilon_{22,112})\;\;\gamma_{12,222}\,].\end{aligned} \tag{67}$$

Next, the mixed-derivative shear strain entries are removed. This produces *reduced* strain vectors, which are identified with a r subscript:

$$\boldsymbol{\epsilon}_r^{(2)}(g) = [\,\epsilon_{11,11}\;\;g\epsilon_{11,12}\;\;\epsilon_{11,22}\;\;\epsilon_{22,11}\;\;g\epsilon_{22,12}\;\;\epsilon_{22,22}\;\;\gamma_{12,11}\;\;\gamma_{12,22}\,]^T\,, \tag{68}$$

$$\boldsymbol{\epsilon}_r^{(3)}(g) = [\,\epsilon_{11,111}\;\;g\epsilon_{11,112}\;\;g\epsilon_{11,122}\;\;\epsilon_{11,222}\;\;\epsilon_{22,111}\;\;g\epsilon_{22,112}\;\;g\epsilon_{22,122}\;\;\epsilon_{22,222}\;\;\gamma_{12,111}\;\;\gamma_{12,222}\,]. \tag{69}$$

The reduced C-vectors of order m have the same length as the corresponding displacement C-vectors, which have order $m+1$. Reduced and unconstrained strain C-vectors can be linked as

$$\boldsymbol{\epsilon}^{(m)}(g) = \mathbf{H}_{\epsilon r}^{(m)}\, \boldsymbol{\epsilon}_r^{(m)}(g), \quad \boldsymbol{\epsilon}_r^{(m)}(g) = \mathbf{H}_{r\epsilon}^{(m)}\, \boldsymbol{\epsilon}^{(m)}(g), \tag{70}$$

in which $\mathbf{H}_{\epsilon r}^{(m)}$ are sparse rectangular matrices dimensioned 9×8 and 12×10 for $m=2$ and $m=3$, respectively, whereas $\mathbf{H}_{r\epsilon}^{(m)}$ are their Moore-Penrose generalized inverses [4, 47]. For arbitrary g they are

$$\mathbf{H}_{\epsilon r}^{(2)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & g & g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{H}_{r\epsilon}^{(2)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & g_1 & g_2 & 0 & 0 & 0 & g_3 & 0 \\ 0 & 0 & g_2 & g_1 & 0 & 0 & 0 & g_3 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{71}$$

$$\mathbf{H}_{\epsilon r}^{(3)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & g & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & g & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{H}_{r\epsilon}^{(3)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & g_4 & 0 & -g_5 & 0 & 0 & 0 & 0 & g_6 & 0 & 0 \\ 0 & 0 & 0 & g_7 & 0 & -g_5 & 0 & 0 & 0 & 0 & g_5 & 0 \\ 0 & 0 & -g_5 & 0 & g_7 & 0 & 0 & 0 & 0 & g_5 & 0 & 0 \\ 0 & 0 & 0 & -g_5 & 0 & g_4 & 0 & 0 & 0 & 0 & g_6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{72}$$

in which the auxiliary variables are $g_1 = (1+g^2)/(1+2g^2)$, $g_2 = -g^2/(1+2g^2)$, $g_3 = g/(1+2g^2)$, $g_4 = (1+g^2)/(2+g^2)$, $g_5 = g/(2+g^2)$, $g_6 = 1/(2+g^2)$, and $g_7 = 2/(2+g^2)$.

Note that $\mathbf{H}_{r\epsilon}^{(2)}\,\mathbf{H}_{\epsilon r}^{(2)} = \mathbf{I}_8$ and $\mathbf{H}_{r\epsilon}^{(3)}\,\mathbf{H}_{\epsilon r}^{(3)} = \mathbf{I}_{10}$, but reversing the factors does not gives identity matrices. Of course, the explicit form of $\mathbf{H}_{r\epsilon}^{(m)}$ depends on the choice of generalized inverse.

5.3 Augmented, Constrained and Reduced Stress C-Vectors

The stress C-vectors given by (58)–(61) must be adjusted when paired to augmented and reduced strain C-vectors. For notational convenience we take $\boldsymbol{\sigma}_c^{(m)} = \boldsymbol{\sigma}^{(m)}$ for all m, since no *a priori* constraints are imposed on stresses. The stress C-vector associated with the augmented strain C-vector for $m=0$ is obviously

$$\boldsymbol{\sigma}_a^{(0)} = [\,\sigma_{11}\;\;\sigma_{22}\;\;\sigma_{12}\;\;0\,]^T, \tag{73}$$

in which the zero entry results from ω_3 being a rigid motion amplitude. For $m = 2, 3$ the conjugacy condition is

$$(\boldsymbol{\sigma}^{(m)})^T\,\boldsymbol{\epsilon}_c^{(m)} = (\boldsymbol{\sigma}_c^{(m)})^T\,\boldsymbol{\epsilon}_c^{(m)} = (\boldsymbol{\sigma}_r^{(m)})^T\boldsymbol{\epsilon}_r^{(m)}. \tag{74}$$

It is important to use here $\boldsymbol{\epsilon}_c^{(m)}$ instead of $\boldsymbol{\epsilon}^{(m)}$ to get the correct result. Application of (74) gives the reduced stress C-vectors as

$$\boldsymbol{\sigma}_r^{(2)}(g) = \begin{bmatrix}\sigma_{11,11} & \frac{2}{g}\sigma_{11,12} & \tilde{\sigma}_{11,22} & \tilde{\sigma}_{22,11} & \frac{2}{g}\sigma_{22,12} & \sigma_{22,22} & \sigma_{12,11} & \sigma_{12,22}\end{bmatrix}^T, \tag{75}$$

$$\boldsymbol{\sigma}_r^{(3)}(g) = \begin{bmatrix}\sigma_{11,111} & \frac{3}{g}\sigma_{11,112} & \frac{3}{g}\tilde{\sigma}_{11,122} & \tilde{\sigma}_{11,222} & \tilde{\sigma}_{22,111} & \frac{3}{g}\tilde{\sigma}_{22,112} & \frac{3}{g}\sigma_{22,122} & \sigma_{22,222} & \sigma_{12,111} & \sigma_{12,222}\end{bmatrix}^T. \tag{76}$$

The modified stresses in the foregoing expressions are

$$\begin{aligned}
\tilde{\sigma}_{11,22} &= \sigma_{11,22} + 2\sigma_{12,12}, & \tilde{\sigma}_{22,11} &= \sigma_{22,11} + 2\sigma_{12,12},\\
\tilde{\sigma}_{11,122} &= \sigma_{11,122} + \sigma_{12,112}, & \tilde{\sigma}_{11,222} &= \sigma_{11,222} + 3\sigma_{12,122},\\
\tilde{\sigma}_{22,111} &= \sigma_{22,111} + 3\sigma_{12,112}, & \tilde{\sigma}_{22,112} &= \sigma_{22,112} + \sigma_{12,122}.
\end{aligned} \tag{77}$$

By analogy with (70), the linkage of the reduced stress vectors (75) and (76) to the full ones can be expressed in matrix form as

$$\boldsymbol{\sigma}^{(m)}(g) = \mathbf{H}_{\sigma r}^{(m)}\,\boldsymbol{\sigma}_r^{(m)}(g), \quad \boldsymbol{\sigma}_r^{(m)}(g) = \mathbf{H}_{r\sigma}^{(m)}\,\boldsymbol{\sigma}^{(m)}(g), \tag{78}$$

The linkage matrices in (78) are related to those of (70) by

$$\mathbf{H}_{\sigma r}^{(m)} = \left(\mathbf{H}_{r\epsilon}^{(m)}\right)^T, \quad \mathbf{H}_{r\sigma}^{(m)} = \left(\mathbf{H}_{\epsilon r}^{(m)}\right)^T, \tag{79}$$

both of which follow directly from (74).

5.4 Strain-Displacement Relations

A strain derivative C-vector $\boldsymbol{\epsilon}^{(m)}$ ($m \ge 0$) can be linked to the corresponding displacement derivative C-vector $\mathbf{u}^{(m+1)}$ through matrix transformations. These transformations and their inverses are denoted as

$$\boldsymbol{\epsilon}^{(m)} = \mathbf{T}_{\epsilon u}^{(m)}\,\mathbf{u}^{(m+1)}, \quad \mathbf{u}^{(m+1)} = \mathbf{T}_{u\,\epsilon}^{(m)}\,\boldsymbol{\epsilon}^{(m)}, \tag{80}$$

for $m \ge 0$. For augmented or reduced strain C-vectors the notation is slightly altered to

$$\boldsymbol{\epsilon}_a^{(m)} = \mathbf{T}_{\epsilon u a}^{(m)}\,\mathbf{u}^{(m+1)}, \quad \mathbf{u}^{(m+1)} = \mathbf{T}_{u\,\epsilon a}^{(m)}\,\boldsymbol{\epsilon}_r^{(m)}, \quad \boldsymbol{\epsilon}_r^{(m)} = \mathbf{T}_{\epsilon u r}^{(m)}\,\mathbf{u}^{(m+1)}, \quad \mathbf{u}^{(m+1)} = \mathbf{T}_{u\,\epsilon r}^{(m)}\,\boldsymbol{\epsilon}_r^{(m)}, \tag{81}$$

in which the subscript a or r flags an augmented or reduced C-vector, respectively. For $m = 0$ through 3, $\boldsymbol{\epsilon}^{(m)}$ has dimension 3, 6, 9 and 12, whereas $\mathbf{u}^{(m+1)}$ has dimension 4, 6, 8 and 10. The matrices $\mathbf{T}_{\epsilon u}^{(m)}$ of (80) are dimensioned 3×4, 6×6, 9×8, and 12×10, respectively, so they are not square for $m = 0$ and $m \ge 2$. To get their inverses, which are needed for further derivations later, generalized inverses would be required. This can be avoided by using the augmented strain vector (73) for $m = 0$ and the reduced strain vectors (75) and (76) for $m = 2, 3$. The strain-displacement transformation matrices and their

inverses obtained after those modifications follow. (In these equations, k, k_1 and k_2 are the displacement C-vector MDCC introduced in 4.1.)

For order $m = 0$, using the augmented strain vector of (73):

$$\mathbf{T}^{(0)}_{\epsilon ua} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \end{bmatrix}, \quad \mathbf{T}^{(0)}_{u\,\epsilon a} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & -\frac{1}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 1 & 0 & 0 \end{bmatrix}, \tag{82}$$

For order $m = 1$ no augmentation or reduction is necessary:

$$\mathbf{T}^{(1)}_{\epsilon u} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{k} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{k} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & \frac{1}{k} & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & \frac{1}{k} & 0 \end{bmatrix}, \quad \mathbf{T}^{(1)}_{u\,\epsilon} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & k & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & k & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \tag{83}$$

For $m = 2$, using the reduced strain vector of (75):

$$\mathbf{T}^{(2)}_{\epsilon ur} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{g}{k} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{k} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{k} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{g}{k} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & \frac{1}{k} & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & \frac{1}{k} & 0 \end{bmatrix}, \; \mathbf{T}^{(2)}_{u\,\epsilon r} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{k}{g} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & k & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{g} & 0 & 0 & 1 \\ 0 & -\frac{1}{g} & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & k & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{k}{g} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{84}$$

For $m = 3$, using the reduced strain vector of (76):

$$\mathbf{T}^{(3)}_{\epsilon ur} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{g}{k_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{g}{k_2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{k_1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{k_1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{g}{k_2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{g}{k_1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & \frac{1}{k_1} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \frac{1}{k_1} & 0 \end{bmatrix}, \quad \mathbf{T}^{(3)}_{u\,\epsilon r} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{k_1}{g} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{k_2}{g} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & k_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{g} & 0 & 0 & 1 \\ 0 & -\frac{1}{g} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & k_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{k_2}{g} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{k_1}{g} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{85}$$

For $m = 2, 3$, full strain vectors can be recovered using the transformation matrices in (70).

5.5 Rotated Strain Transformations

Vector-cast strain and stress derivatives of order m at a rotated angle φ are denoted by $\boldsymbol{\epsilon}^{(m)}_{\varphi}$ and $\boldsymbol{\sigma}^{(m)}_{\varphi}$, respectively. If $\varphi = 0$, the angle subscript may be omitted. For $m = 2, 3$ a generic MDCC can be optionally shown as an argument, as in $\boldsymbol{\epsilon}^{(m)}_{\varphi}(g)$. The transformation from $\varphi = 0$ to arbitrary φ is written

$$\boldsymbol{\epsilon}^{(m)}_{\varphi} = \mathbf{T}^{(m)}_{\epsilon\varphi}\,\boldsymbol{\epsilon}^{(m)}, \quad \boldsymbol{\sigma}^{(m)}_{\varphi} = \mathbf{T}^{(m)}_{\sigma\varphi}\,\boldsymbol{\sigma}^{(m)}. \tag{86}$$

Here $\boldsymbol{\epsilon}^{(m)}_{\varphi}$ and $\boldsymbol{\epsilon}^{(m)}_{\varphi}$ denotes the full strain and stress C-vectors, respectively. Superscripts may be omitted if $m = 0$. Additional subscripts are inserted for reduced and augmented C-vectors. E.g., for the reduced strain and stress C-vectors used if $m = 2, 3$:

$$\boldsymbol{\epsilon}^{(m)}_{\varphi r} = \mathbf{T}^{(m)}_{\epsilon\varphi r}\,\boldsymbol{\epsilon}^{(m)}_{r}, \quad \boldsymbol{\sigma}^{(m)}_{\varphi r} = \mathbf{T}^{(m)}_{\sigma\varphi r}\,\boldsymbol{\sigma}^{(m)}_{r}. \tag{87}$$

Sometimes it is necessary to show the rotation angle explicitily as argument. For example to state the angle-sum and angle-reversal properties:

$$\mathbf{T}^{(m)}_{\epsilon r}(\varphi_1 + \varphi_2) = \mathbf{T}^{(m)}_{\epsilon r}(\varphi_1)\,\mathbf{T}^{(m)}_{\epsilon r}(\varphi_2), \quad \Rightarrow \quad \mathbf{T}^{(m)}_{\epsilon r}(\varphi)\,\mathbf{T}^{(m)}_{\epsilon r}(-\varphi) = \mathbf{I}. \tag{88}$$

The latter shows that the inverse is obtained simply by reversing φ. For symbolic computations it is convenient to reuse the displacement derivative transformations listed in 5.4 VarFICHOT:subsec:StrainDisplacemen This is done by expressing $\mathbf{T}^{(m)}_{\epsilon}$ as

$$\mathbf{T}^{(m)}_{\epsilon\varphi} = \mathbf{T}^{(m)}_{\epsilon u}\,\mathbf{T}^{(m+1)}_{u\,\varphi}\,\left(\mathbf{T}^{(m)}_{\epsilon\,u}\right)^{-1} = \mathbf{T}^{(m)}_{\epsilon u}\,\mathbf{T}^{(m+1)}_{u\,\varphi}\,\mathbf{T}^{(m)}_{u\,\epsilon}. \tag{89}$$

in which only $\mathbf{T}^{(m+1)}_{u\,\varphi}$ depends on φ. If $m = 0$ we use the augmented strain vector whereas if $m \geq 2$ we use the reduced strain vectors, so as to keep all $\mathbf{T}_{\epsilon u}$ matrices square and possessing ordinary inverses. If matrices depend on MDCC such as k, k_1, k_2 and g, it is important to display the appropriate coefficients in all matrices. Any MDDC coefficients in $\mathbf{T}^{(m+1)}_{u\,\varphi}$ disappear on executing the transformation, and only g remains for $m = 2, 3$. A list of strain transformation matrices follow that account for arbitrary g if $m \geq 2$.

For $m = 0$, using the augmented strain vector (63):

$$\mathbf{T}^{(0)}_{\epsilon a\varphi} = \mathbf{T}_{\epsilon\varphi} = \tfrac{1}{2}\begin{bmatrix} 1+c_2 & 1-c_2 & s_2 & 0 \\ 1-c_2 & 1+c_2 & -s_2 & 0 \\ -2s_2 & 2s_2 & 2c_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{90}$$

The configuration of the fourth row and column reflects the fact that ω_3 is invariant under z rotation. The upper 3×3 principal minor of $\mathbf{T}_{\epsilon\varphi}$ is the "strain rosette transformation" of experimental mechanics [3]. The next ones are not so well known. For $m = 1$:

$$\mathbf{T}^{(1)}_{\epsilon} = \tfrac{1}{4}\begin{bmatrix} a_1 & b_3 & a_4 & b_2 & b_3 & a_4 \\ -b_3 & a_1 & -b_2 & a_4 & -a_4 & b_3 \\ a_4 & b_2 & a_1 & b_3 & -b_3 & -a_4 \\ -b_2 & a_4 & -b_3 & a_1 & a_4 & -b_3 \\ -2\,b_3 & -2\,a_4 & 2\,b_3 & 2\,a_4 & 2\,a_3 & -2\,b_4 \\ 2\,a_4 & -2\,b_3 & -2\,a_4 & 2\,b_3 & 2\,b_4 & 2\,a_3 \end{bmatrix}, \tag{91}$$

in which

$$\begin{array}{llll} a_1 = 3c + c_3, & a_2 = 3c - c_3, & a_3 = c + c_3, & a_4 = c - c_3, \\ b_1 = 3s + s_3, & b_2 = 3s - s_3, & b_3 = s + s_3, & b_4 = s - s_3. \end{array} \tag{92}$$

For $m = 2$:

$$\mathbf{T}^{(2)}_{\epsilon r} = \tfrac{1}{8}\,\mathbf{D}^{(2)}_g \begin{bmatrix} d_1 & e_1 & 3\,d_2 & 3\,d_2 & e_2 & d_3 & e_1 & e_2 \\ -2\,e_1 & 2\,d_6 & 2\,e_3 & -2\,e_4 & -2\,d_7 & 2\,e_2 & -2\,d_2 & 2\,d_2 \\ d_2 & -e_1 & d_4 & d_5 & -e_2 & d_2 & e_2 & e_1 \\ d_2 & e_2 & d_5 & d_4 & e_1 & d_2 & -e_1 & -e_2 \\ -2\,e_2 & -2\,d_7 & 2\,e_4 & -2\,e_3 & 2\,d_6 & 2\,e_1 & 2\,d_2 & -2\,d_2 \\ d_3 & -e_2 & 3\,d_2 & 3\,d_2 & -e_1 & d_1 & -e_2 & -e_1 \\ -2\,e_1 & -2\,d_2 & -2\,e_4 & 2\,e_3 & 2\,d_2 & 2\,e_2 & 2\,d_6 & -2\,d_7 \\ -2\,e_2 & 2\,d_2 & -2\,e_3 & 2\,e_4 & -2\,d_2 & 2\,e_1 & -2\,d_7 & 2\,d_6 \end{bmatrix} \left(\mathbf{D}^{(2)}_g\right)^{-1}, \tag{93}$$

in which $\mathbf{D}^{(2)}_g = \mathbf{diag}[1, g/2, 1, 1, g/2, 1, 1, 1]$, and the auxiliary variables are

$$\begin{array}{llll} d_1 = 3 + 4c_2 + c_4, & d_2 = 1 - c_4, & d_3 = 3 - 4c_2 + c_4, & d_4 = 1 + 4c_2 + 3c_4, \\ d_5 = 1 - 4c_2 + 3c_4, & d_6 = 1 + 2c_2 + c_4, & d_7 = 1 - 4c_2 + c_4, & d_8 = 1 + c_4, \\ e_1 = 2s_2 + s_4, & e_2 = 2s_2 - s_4, & e_3 = 2s_2 + 3s_4, & e_4 = 2s_2 - 3s_4. \end{array} \tag{94}$$

For $m = 3$:

$$\mathbf{T}^{(3)}_{\epsilon r} = \tfrac{1}{16}\,\mathbf{D}^{(3)}_g \begin{bmatrix} f_1 & g_1 & 2f_2 & 4g_2 & 4f_2 & 2g_2 & f_3 & g_3 & g_1 & f_3 \\ -3g_1 & f_9 & 6g_4 & 12f_5 & -12g_5 & 6f_7 & -g_8 & 3f_3 & -3f_6 & 3g_2 \\ 3f_2 & -g_9 & 2f_8 & 12g_4 & -12f_7 & 2g_7 & -f_{11} & 3g_2 & 3g_2 & 3f_2 \\ -g_2 & f_2 & -2g_4 & 4f_4 & -4g_6 & -2f_7 & -g_2 & f_6 & -f_3 & g_1 \\ f_6 & g_2 & -2f_7 & 4g_6 & 4f_4 & 2g_4 & f_2 & g_2 & -g_1 & -f_3 \\ -3g_2 & -f_{11} & -2g_7 & -12f_7 & -12g_4 & 2f_8 & g_9 & 3f_2 & 3f_2 & -3g_2 \\ 3f_3 & g_8 & 6f_7 & 12g_5 & 12f_5 & -6g_4 & f_9 & 3g_1 & -3g_2 & -3f_6 \\ -g_3 & f_3 & -2g_2 & 4f_2 & -4g_2 & 2f_2 & -g_1 & f_1 & f_3 & -g_1 \\ -2g_1 & -2f_6 & -4g_5 & -4f_{12} & 4g_{10} & 4f_5 & 2g_2 & 2f_3 & 2f_{13} & -2g_{11} \\ 2f_3 & -2g_2 & 4f_5 & -4g_{10} & -4f_{12} & 4g_5 & -2f_6 & 2g_1 & 2g_{11} & 2f_{13} \end{bmatrix} \left(\mathbf{D}^{(3)}_g\right)^{-1}, \tag{95}$$

in which $\mathbf{D}^{(3)}_g = \textbf{diag}[1, g/3, g/3, 1, 1, g/3, g/3, 1, 1, 1]$, and the auxiliary variables are

$$\begin{array}{llll} f_1 = 10c + 5c_3 + c_5, & f_2 = 2c - c_3 - c_5, & f_3 = 2c - 3c_3 + c_5, & f_4 = c + 2c_3 + c_5, \\ f_5 = c - c_5, & f_6 = 2c - c_3 - c_5, & f_7 = c_3 - c_5, & f_8 = 2c + 3c_3 + 3c_5, \\ f_9 = 6c + 7c_3 + 3c_5, & f_{10} = c_3 - c_5, & f_{11} = 2c - 5c_3 + 3c_5, & f_{12} = c - 3c_3 + 2c_5, \\ f_{13} = 4c + 3c_3 + c_5, & g_1 = 2s + 3s_3 + s_5, & g_2 = 2s + s_3 - s_5, & g_3 = 10s - 5s_3 + s_5, \\ g_4 = s_3 + s_5, & g_5 = s - s_5, & g_6 = s - 2s_3 + s_5, & g_7 = 2s - 3s_3 + 3s_5, \\ g_8 = 6s - 7s_3 + 3s_5, & g_9 = 2s + 5s_3 + 3s_5, & g_{10} = s + 3s_3 + 2s_5, & g_{11} = 4s - 3s_3 + s_5, \end{array} \tag{96}$$

If the strain C-vector is binomial, that is $g = m$ for $m = 2, 3$, the $\mathbf{D}^{(m)}_g$ diagonal scaling matrices in (93) and (95) reduce to the identity and may be dropped.

To pass from reduced to full strain C-vectors for $m = 2, 3$, use the first of (70) to get $\mathbf{T}^{m)}_{\epsilon} = \mathbf{H}^{(m)}_{\epsilon r}\,\mathbf{T}^{(m)}_{\epsilon r}\mathbf{H}^{(m)}_{r\epsilon}$ for $m = 2, 3$. The expanded transformation matrix depends on the choice of generalized inverse. Alternatively (89) could be used.

5.6 Rotated Stress Transformations

Strain and stress transformation matrices are linked by conjugacy constraints that require

$$\left(\mathbf{T}^{(m)}_{\sigma\varphi}\right)^T\,\mathbf{T}^{(m)}_{\epsilon\varphi} = \mathbf{I}, \quad \Rightarrow \quad \mathbf{T}^{(m)}_{\sigma\varphi} = \left(\mathbf{T}^{(m)}_{\epsilon\varphi}\right)^{-1}. \tag{97}$$

with an analogous constraint for reduced and augmented vectors. So if $\mathbf{T}^{(m)}_{\epsilon\varphi}$ is known, to get the corresponding stress transformation, replace φ by $-\varphi$ and transpose it. The transverse stress σ_{33} is invariant with respect toi φ and is not considered in the transformations since it depends on constitutive properties. This is further studied in the section dealing with pressure.

For $m = 0$, using unaugmented stress-strain C-vectors and the upper $3x3$ principal minor of (90), we get

$$\mathbf{T}^{(0)}_{\sigma\varphi} = \mathbf{T}_{\sigma\varphi} = \tfrac{1}{2}\begin{bmatrix} 1 + c_2 & 1 - c_2 & 2s_2 \\ 1 - c_2 & 1 + c_2 & -2s_2 \\ -s_2 & s_2 & c_2 \end{bmatrix}, \tag{98}$$

which is the plane stress transformation of structural mechanics [3] (note that $\frac{1}{2}(1+c_2)=\cos^2\varphi$, and $\frac{1}{2}(1-c_2)=\sin^2\varphi$.) The sum $\sigma_{11}+\sigma_{22}$ is invariant. For $m=1$ we get

$$\mathbf{T}^{(1)}_{\sigma\varphi}=\tfrac{1}{4}\begin{bmatrix} a_1 & b_3 & a_4 & b_2 & 2b_3 & 2a_4\\ -b_3 & a_1 & -b_2 & a_4 & -2a_4 & 2b_3\\ a_4 & b_2 & a_1 & b_3 & -2b_3 & -2a_4\\ -b_2 & a_4 & -b_3 & a_1 & 2a_4 & -2b_3\\ -b_3 & -a_4 & b_3 & a_4 & 2a_3 & 2b_4\\ a_4 & -b_3 & -a_4 & b_3 & -2b_4 & 2a_3 \end{bmatrix} \tag{99}$$

in which a_i and b_i are given by (92). Comparing $\mathbf{T}^{(0)}_{\sigma\varphi}$ and $\mathbf{T}^{(1)}_{\sigma\varphi}$ to the corresponding strain transformations, an obvious pattern emerges: the sine-cosine entries repeat but their coefficients transpose. For $m=2,3$ the same pattern holds but is complicated by the presence of g.

5.7 Linear Invariants

By analogy with 4.8, a linear invariant of the strain derivative tensor of order m is a scalar

$$I^{(m)}_{\epsilon}=\left(\mathbf{w}^{(m)}_{\epsilon}\right)^T\boldsymbol{\epsilon}^{(m)},\quad \mathbf{w}^{(m)}_{u}\neq\mathbf{0}, \tag{100}$$

that remains unchanged when $\boldsymbol{\epsilon}^{(m)}$ is rotated. Here $\mathbf{w}^{(m)}_{\epsilon}$ is a *invariant weight vector* that is independent of φ. If the only solution is $\mathbf{w}^{(m)}_{u}=\mathbf{0}$, there are no linear invariants for order m. Because of (89), linear strain invariants of order m are expected to be closely related to those of displacement derivatives of order $m+1$, which were covered in 4.8. So only a brief summary of results is given. For $m\in[0,3]$ linear invariants only exist for $m=0,2$, and there are two of each:

$$\begin{aligned} I^{(0)}_{\epsilon\textcircled{1}}&=\epsilon_{11}+\epsilon_{22}=\epsilon_v,\quad I^{(0)}_{\epsilon\textcircled{2}}=2\omega_3,\\ I^{(2)}_{\epsilon\textcircled{1}}&=\epsilon_{11,11}+\epsilon_{11,22}+\epsilon_{22,11}+\epsilon_{22,22}=\Delta\,\epsilon_v,\\ I^{(2)}_{\epsilon\textcircled{2}}&=2\epsilon_{11,12}-2\epsilon_{22,12}+\gamma_{12,11}+\gamma_{12,22}=2\Delta\,\omega_3. \end{aligned} \tag{101}$$

These are in one-to-one correspondence with those in (42). Of these, $I^{(0)}_{\epsilon\textcircled{2}}$ only appears because of the strain C-vector augmentation (73), since ω_3 is not a deformational strain. If the unaugmented 3-vector $\boldsymbol{\epsilon}^{(0)}$ is used, that invariant disappears.

5.8 Quadratic Invariants

A *quadratic invariant* of the strain derivative tensor of order m is a quadratic form scalar

$$J^{(m)}_{\epsilon}=\left(\boldsymbol{\epsilon}^{(m)}\right)^T\mathbf{W}^{(m)}_{\epsilon}\,\boldsymbol{\epsilon}^{(m)}, \tag{102}$$

that remains unchanged when $\boldsymbol{\epsilon}^{(m)}$ is rotated. Here $\mathbf{W}^{(m)}_{\epsilon}$ is a nonzero *invariant weight matrix*, which is independent of φ. The necessary developments to compute these invariants are covered in 4.9, and need

not be repeated. The Sylvester equation equivalent to (44) is $\left(\mathbf{T}^{(m)}_{\epsilon\varphi}\right)^T \mathbf{W}^{(m)}_{\epsilon}\, \mathbf{T}^{(m)}_{\epsilon\varphi} = \mathbf{W}^{(m)}_{\epsilon}$ to be solved for $\mathbf{W}^{(m)}_{\epsilon} \neq \mathbf{0}$. If displacement quadratic invariants are available, a solution can be obtained through a congruential transformation: $\mathbf{W}^{(m)}_{\epsilon} = (\mathbf{T}^{(m)}_{\epsilon u})^T\, \mathbf{W}^{(m+1)}_{u}\, \mathbf{T}^{(m)}_{u\epsilon}$ which preserves symmetry, using weight matrices $\mathbf{W}^{(m)}_{u}$ of 4.9 and the strain-displacement matrices listed in 5.4. But the method was found unreliable, as the results may contain superfluous free parameters.

To get optimally sparse solutions in the sense defined in B.3.7, (102) was solved using stochastic variable choices. The following expressions result from using that software with *flat* strain C-vectors.

$$\mathbf{W}^{(0)}_{\epsilon a} = \begin{bmatrix} \eta_1 & \eta_1 - 2\eta_2 & 0 & \eta_3 \\ \eta_1 - 2\eta_2 & \eta_1 & 0 & \eta_3 \\ 0 & 0 & \eta_2 & 0 \\ \eta_3 & \eta_3 & 0 & \eta_4 \end{bmatrix}, \tag{103}$$

If the strain vector is not augmented, $\mathbf{W}^{(0)}_{\epsilon}$ is the upper 3×3 principal minor of the matrix (103), and has only 2 free parameters.

$$\mathbf{W}^{(1)}_{\epsilon} = \begin{bmatrix} \xi_1 & 2\eta_2 & \eta_3 & 0 & -\eta_2 & \eta_4 \\ 2\eta_2 & \xi_2 & 0 & \eta_3 & \eta_1 & \eta_2 \\ \eta_3 & 0 & \xi_2 & -2\eta_2 & -\eta_2 & \eta_1 \\ 0 & \eta_3 & -2\eta_2 & \xi_1 & \eta_4 & \eta_2 \\ -\eta_2 & \eta_1 & -\eta_2 & \eta_4 & \eta_5 & 0 \\ \eta_4 & \eta_2 & \eta_1 & \eta_2 & 0 & \eta_5 \end{bmatrix} \tag{104}$$

$$\mathbf{W}^{(2)}_{\epsilon r} = \begin{bmatrix} \eta_1 & 2\eta_2 & \eta_3 & \xi_3 & 2\eta_4 & \xi_4 & -\eta_2 & -\eta_4 \\ 2\eta_2 & \xi_5 & 2\eta_2 & -2\eta_4 & \xi_6 & -2\eta_4 & \eta_6 & \eta_5 \\ \eta_3 & 2\eta_2 & \xi_7 & \xi_8 & 2\eta_4 & \xi_3 & \eta_4 & \eta_2 \\ \xi_3 & -2\eta_4 & \xi_8 & \xi_7 & -2\eta_2 & \eta_3 & -\eta_2 & -\eta_4 \\ 2\eta_4 & \xi_6 & 2\eta_4 & -2\eta_2 & \xi_5 & -2\eta_2 & \eta_5 & \eta_6 \\ \xi_4 & -2\eta_4 & \xi_3 & \eta_3 & -2\eta_2 & \eta_1 & \eta_4 & \eta_2 \\ -\eta_2 & \eta_6 & \eta_4 & -\eta_2 & \eta_5 & \eta_4 & \eta_8 & \eta_7 \\ -\eta_4 & \eta_5 & \eta_2 & -\eta_4 & \eta_6 & \eta_2 & \eta_7 & \eta_8 \end{bmatrix} \tag{105}$$

$$\mathbf{W}^{(3)}_{\epsilon r} = \begin{bmatrix} \eta_1 & 2\eta_2 & \eta_3 & \eta_4 & \eta_5 & -\eta_4 & \xi_9 & 0 & -\eta_2 & \eta_6 \\ 2\eta_2 & \xi_{10} & 3\eta_2 & \eta_7 & \xi_{11} & \xi_{12} & 0 & \xi_9 & \xi_{13} & \eta_4 \\ \eta_3 & 3\eta_2 & \xi_{14} & \xi_{15} & \xi_{16} & 0 & \xi_{12} & \eta_4 & -\eta_4 & \eta_8 \\ \eta_4 & \eta_7 & \xi_{15} & \xi_{18} & 0 & \xi_{16} & \xi_{17} & \eta_5 & \xi_{19} & \eta_2 \\ \eta_5 & \xi_{11} & \xi_{16} & 0 & \xi_{18} & \xi_{20} & \eta_7 & -\eta_4 & -\eta_2 & \xi_{19} \\ -\eta_4 & \xi_{12} & 0 & \xi_{16} & \xi_{20} & \xi_{14} & -3\eta_2 & \eta_3 & \eta_8 & \eta_4 \\ \xi_9 & 0 & \xi_{12} & \xi_{17} & \eta_7 & -3\eta_2 & \xi_{10} & -2\eta_2 & -\eta_4 & \xi_{13} \\ 0 & \xi_9 & \eta_4 & \eta_5 & -\eta_4 & \eta_3 & -2\eta_2 & \eta_1 & \eta_6 & \eta_2 \\ -\eta_2 & \xi_{13} & -\eta_4 & \xi_{19} & -\eta_2 & \eta_8 & -\eta_4 & \eta_6 & \eta_9 & 0 \\ \eta_6 & \eta_4 & \eta_8 & \eta_2 & \xi_{19} & \eta_4 & \xi_{13} & \eta_2 & 0 & \eta_9 \end{bmatrix} \tag{106}$$

The auxiliary variables in the last three matrices are

$$
\begin{array}{ll}
\xi_1 = \eta_1 + \eta_3 + \eta_4 + 2\eta_5 & \xi_2 = -\eta_1 + \eta_3 - \eta_4 + 2\eta_5, \\
\xi_3 = \eta_1 - \eta_6 - 2\eta_8, & \xi_4 = \eta_3 - \eta_5 - 2\eta_7, \\
\xi_5 = 2\eta_1 - 2(\eta_3 + \eta_6), & \xi_6 = 2\eta_1 - 2(\eta_3 + \eta_6) + 4\eta_7 - 4\eta_8, \\
\xi_7 = \eta_1 - 2\eta_5 - 2\eta_7 + 2\eta_8, & \xi_8 = \eta_3 - \eta_5 + 2\eta_6 - 4\eta_7 + 2\eta_8, \\
\xi_9 = 3\eta_6 + \eta_7 - 2\eta_8, & \xi_{10} = \eta_1 - 2\eta_3 + 2\eta_5 + 4\eta_9, \\
\xi_{11} = -\eta_2 + 2\eta_4, & \xi_{12} = \eta_3 + 3\eta_5 - 12\eta_6 - 3\eta_7 + 4\eta_8, \\
\xi_{13} = \eta_1 - \eta_5 - 2\eta_9, & \xi_{14} = 3\eta_1 - 2\eta_3 + 6\eta_6 - 2\eta_8 + 3\eta_9, \\
\xi_{15} = 3\eta_2 - \eta_4, & \xi_{16} = 3\eta_1 - 2\eta_3 - 3\eta_5 + 12\eta_6 + 3\eta_7 - 4\eta_8 - 3\eta_9, \\
\xi_{17} = \eta_2 - 2\eta_4, & \xi_{18} = \eta_1 - 2\eta_8 + 3\eta_9 -, \\
\xi_{19} = \eta_3 - 4\eta_6 - \eta_7 + \eta_8, & \xi_{20} = -3\eta_2 + \eta_4.
\end{array}
\tag{107}
$$

If the strain C-vector is not flat, diagonal scaling should be applied as necessary.

Instantiation of $\mathbf{W}_{\epsilon a}^{(0)}$ by setting the η_i to $\{1,0,0,0\}$, $\{1,\frac{1}{2},0,0\}$, $\{0,0,1,0\}$, and $\{0,0,0,1\}$ in turn yields well known invariants:

$$
J_{\epsilon\textcircled{1}}^{(0)} = (\epsilon_{11}+\epsilon_{22})^2 = e^2, \quad J_{\epsilon\textcircled{2}}^{(0)} = \epsilon_{11}\epsilon_{22} - \tfrac{1}{4}\gamma_{12}^2, \quad J_{\epsilon\textcircled{3}}^{(0)} = e\,\omega_3 . J_{\epsilon\textcircled{4}}^{(0)} = \omega_3^2, \tag{108}
$$

in which e denotes the volumetric strain. The first two are the squared trace and determinant, respectively, of the 2×2 strain tensor, and are the strain equivalents of (52). The last two disappear if the unaugmented strain C-vector (54) is used. The "Von Mises-Odqvist strain invariant" (also called equivalent strain) $\bar{\epsilon} = \frac{2}{3}\sqrt{\epsilon_{ij}\epsilon_{ij}}$ that shows up in the simplest strain-hardening plasticity model [24,28] is related to the first two of (108) through $\epsilon_{ij}\epsilon_{ij} = \epsilon_{11}^2 + \epsilon_{22}^2 + \frac{1}{2}\gamma_{12}^2 = J_{\epsilon\textcircled{1}}^{(0)} - 2J_{\epsilon\textcircled{2}}^{(0)}$.

For $m = 1$ the task of producing compact base invariants is a bit more involved. Setting the five η_i to $\{0,0,1,0,0\}$, $\{0,0,2,0,-1\}$, $\{-1,0,0,2,0\}$, $\{1,0,0,0,0\}$, and $\{0,0,0,0,0,\frac{1}{2}\}$ gives

$$
\begin{aligned}
J_{\epsilon\textcircled{1}}^{(1)} &= (\epsilon_{111}+\epsilon_{221})^2 + (\epsilon_{112}+\epsilon_{222})^2 = (\mathbf{div}\, e)^2, \\
J_{\epsilon\textcircled{2}}^{(1)} &= 4\,\epsilon_{111}\,\epsilon_{221} + 4\,\epsilon_{112}\,\epsilon_{222} - \gamma_{121}^2 - \gamma_{122}^2, \\
J_{\epsilon\textcircled{3}}^{(1)} &= 2\,(\gamma_{122}\,(\epsilon_{111}-\epsilon_{221}) + \gamma_{121}\,(-\epsilon_{112}+\epsilon_{222})), \\
J_{\epsilon\textcircled{4}}^{(1)} &= \epsilon_{111}^2 - \epsilon_{112}^2 - \epsilon_{221}^2 + \epsilon_{222}^2 + 2\,\gamma_{121}\,\epsilon_{112} + 2\,\gamma_{122}\,\epsilon_{221}, \\
J_{\epsilon\textcircled{5}}^{(1)} &= \epsilon_{111}^2 + \epsilon_{112}^2 + \epsilon_{221}^2 + \epsilon_{222}^2 + \tfrac{1}{2}\,\gamma_{121}^2 + \tfrac{1}{2}\,\gamma_{122}^2.
\end{aligned}
\tag{109}
$$

These can be shown to be linear combinations of the five Toupin invariants [50]: $\epsilon_{ijj}\,\epsilon_{ikk}$, $\epsilon_{iik}\,\epsilon_{kjj}$, $\epsilon_{iik}\,\epsilon_{jjk}$, $\epsilon_{ijk}\,\epsilon_{ijk}$, and $\epsilon_{ijk}\,\epsilon_{kji}$, in which i,j,k range over 1,2. But the forms (109) have fewer terms while the first one has a simple physical meaning.

No attempt was made to find compact base invariants for $m = 2$ and $m = 3$.

5.9 Stress Invariants

Stress derivatives do not occur directly in single-field FIC functionals, and appear only indirectly through the pressure in mixed functionals for incompressible or quasi-incompressible media. For that reason their

invariants are not tabulated here. Should they be required, previous results for strains can be reused if *conjugate* stress C-vectors are used; this is particularly important for orders $m \geq 2$ to account for reduced strain forms. If that is done, the reuse consists of formally replacing ϵ_{11}, ϵ_{22} and $\frac{1}{2}\gamma_{12}$ by σ_{11}, σ_{22} and σ_{12}, respectively, whereas ω_3, if it appears, should be replaced by zero. For example: $I^{(0)}_{\sigma\textcircled{1}} = \sigma_{11} + \sigma_{22}$, $J^{(0)}_{\sigma\textcircled{1}} = (\sigma_{11} + \sigma_{22})^2$, $J^{(0)}_{\sigma\textcircled{2}} = \sigma_{11}\sigma_{22} - \sigma^2_{12}$. The more imprtant pressure derivatives are fully covered in the next section.

6 Pressure and Volumetric Strain

The volumetric strain e and pressure p (also called mean stress) are defined as the scalars

$$e = \epsilon_{11} + \epsilon_{22} + \epsilon_{33} = \epsilon_{11} + \epsilon_{22}, \quad p = \tfrac{1}{3}(\sigma_{11} + \sigma_{22} + \sigma_{33}), \tag{110}$$

They are considered together in this Section because they appear as conjugate quantities in mixed variational principles of Herrmann type [21]. For isotropic materials, which are the only ones considered here, the pressure transformations in terms of inplane stresses necesssarily involve Poisson's ratio, since $\sigma_{33} = -\nu\,(\sigma_{11} + \sigma_{22})$, whence

$$p = \tfrac{1}{3}\,(1 - \nu)\,(\sigma_{11} + \sigma_{22}). \tag{111}$$

For pressure derivatives it is assumed that ν is constant throughout, whence it does not appear in the pressure transformations listed in 6.2

For order zero the C-vectors are simply $\mathbf{e}^{(0)} = [e$ and $\mathbf{p}^{(0)} = [p]$. For orders $m = 1, 2, 3$ their configurations are

$$\mathbf{e}^{(1)}_v = [\,e_1 \quad \epsilon_2\,]^T, \tag{112}$$

$$\mathbf{e}^{(2)}_v = [\,e_{11} \quad g\,e_{12} \quad e_{22}\,]^T, \tag{113}$$

$$\mathbf{e}^{(3)}_v = [\,e_{,111} \quad g\,e_{,112} \quad g\,e_{,122} \quad e_{,222}\,]^T, \tag{114}$$

$$\mathbf{p}^{(1)} = [\,p_{,1} \quad p_{,2}\,]^T, \tag{115}$$

$$\mathbf{p}^{(2)} = \big[\,p_{,11} \quad \frac{2}{g}\,p_{,12} \quad p_{,22}\,\big]^T, \tag{116}$$

$$\mathbf{p}^{(3)} = \big[\,p_{,111} \quad \frac{3}{g}\,p_{,112} \quad \frac{3}{g}\,p_{,122} \quad p_{,222}\,\big]^T, \tag{117}$$

The value of g for $m = 2, 3$ must the same as that chosen for strains and stresses; the most common choice being $g = m$.

Transformations of $\boldsymbol{\epsilon}^{(m)}$ and $\mathbf{p}^{(m)}$ under a rotation angle φ are denoted by

$$\mathbf{e}^{(m)}_\varphi = \mathbf{T}^{(m)}_e\,\mathbf{e}^{(m)}, \quad \mathbf{p}^{(m)}_\varphi = \mathbf{T}^{(m)}_p\,\mathbf{p}^{(m)}, \tag{118}$$

in which $\mathbf{T}_e^{(m)}$ and $\mathbf{T}_p^{(m)}$ has order $m+1$. These matrices can be easily computed by doing congruential transformations on the strain and stress transformations. For that introduce the matrices

$$\mathbf{Z}^{(0)} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1 & 0\end{bmatrix}, \quad \mathbf{Z}^{(1)} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 0 & 1 & 0 & 0 & 0\\ 0 & 1 & 0 & 1 & 0 & 0\end{bmatrix},$$
$$\mathbf{Z}^{(2)} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0\end{bmatrix}, \tag{119}$$
$$\mathbf{Z}^{(3)} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0\end{bmatrix},$$

These verify the orthonormality property $\left(\mathbf{Z}^{(m)}\right)^T\,\mathbf{Z}^{(m)} = \mathbf{I}_{m+1}$. Then for $m = 1, 2, 3$ we get

$$\mathbf{T}_e^{(m)} = \mathbf{Z}^{(m)}\,\mathbf{T}_\epsilon^{(m)}\,\left(\mathbf{Z}^{(m)}\right)^T, \quad \mathbf{T}_p^{(m)} = \mathbf{Z}^{(m)}\,\mathbf{T}_\sigma^{(m)}\,\left(\mathbf{Z}^{(m)}\right)^T. \tag{120}$$

in which the $\mathbf{T}_\epsilon^{(m)}$ and $\mathbf{T}_\sigma^{(m)}$ matrices are listed in 5.5 and 5.6, respectively. For $m = 0$ the unaugmented versions are used, whereas for $m = 2, 3$ the reduced forms are used.

6.1 Volumetric Strain Transformations

Applying the factors (119) as per the first of (120) gives $\mathbf{T}_{ev}^{(0)} = [1]$ as expected, and

$$T_e^{(1)} = \begin{bmatrix}c & s\\ -s & c\end{bmatrix}, \tag{121}$$

$$T_e^{(2)}(g) = \tfrac{1}{2}\begin{bmatrix}1+c_2 & 2s_2/g & 1-c_2\\ -s_2\,g & 2c_2 & s_2\,g\\ 1-c_2 & -2s_2/g & 1+c_2\end{bmatrix}, \tag{122}$$

$$T_e^{(3)}(g) = \tfrac{1}{4}\begin{bmatrix}3c+c_3 & 3(s+s_3)/g & 3(c-c_3)/g & 3s-s_3\\ -(s+s_3)g & c+3c_3 & -s+3s_3 & (c-c_3)g\\ (c-c_3)g & s-3s_3 & c+3c_3 & (s+s_3)g\\ -3s+s_3 & 3(c-c_3)/g & -3(s+s_3)/g & 3c+c_3\end{bmatrix}. \tag{123}$$

6.2 Pressure Transformations

From the second of (120) we trivially obtain $\mathbf{T}_p^{(0)} = [1]$ and

$$\mathbf{T}_p^{(1)} = \begin{bmatrix}c & s\\ -s & c\end{bmatrix}, \tag{124}$$

$$\mathbf{T}_p^{(2)}(g) = \tfrac{1}{2}\begin{bmatrix} 1+c_2 & s_2\,g & 1-c_2 \\ -s_2/g & 2\,c_2 & 2s_2/g \\ 1-c_2 & -s_2\,g & 1+c_2 \end{bmatrix}, \tag{125}$$

$$\mathbf{T}_p^{(3)}(g) = \tfrac{1}{4}\begin{bmatrix} (3\,c+c_3) & (s+s_3)\,g & (c-c_3)\,g & 3\,s-s_3 \\ -3\,(s+s_3)/g & c+3\,c_3 & -s+3\,s_3 & 3\,(c-c_3)/g \\ 3\,(c-c_3)/g & s-3\,s_3 & c+3\,c_3 & 3\,(s+s_3)/g \\ -3\,s+s_3 & (c-c_3)\,g & -(s+s_3)\,g & 3\,c+c_3 \end{bmatrix}. \tag{126}$$

The conjugacy condition is

$$\left(\mathbf{T}_p^{(m)}\right)^T\,\mathbf{T}_e^{(m)} = \mathbf{I}_{m+1}, \tag{127}$$

which can be directly verified to hold for all m and any $g \neq 0$.

6.3 Volumetric Strain Invariants

For the following calculations the binomial form of $\mathbf{e}^m$ was used for $m \geq 2$. As regards linear invariants, for $m = 0$ there is obviously the scalar e. In the range $m \in [1, 3]$, only the Laplacian of e appears at $m = 2$:

$$\mathbf{w}_e^{(2)} = \eta_1\,[\,1 \quad 0 \quad 1\,] \quad \Rightarrow \quad I_{\textcircled{1}}^{(2)} = e_{,11} + e_{,22} = \Delta\,e. \tag{128}$$

The set of quadratic invariants is richer. For $m = 0$ there is the trivial one e^2. For $m = 1$ there is one:

$$\mathbf{W}_e^{(1)} = \eta_1\,\mathbf{I}_2 \quad \Rightarrow \quad J_{\textcircled{1}}^{(1)} = \epsilon_{v,1}^2 + \epsilon_{v,2}^2. \tag{129}$$

which is the Euclidean norm of the gradient ∇e. For $m = 2$ the solution has two free parameters:

$$\mathbf{W}_e^{(2)} = \begin{bmatrix} \eta_1 & 0 & \eta_1 - 2\eta_2 \\ 0 & \eta_2 & 0 \\ \eta_1 - 2\eta_2 & 0 & \eta_1 \end{bmatrix}. \tag{130}$$

Setting $\eta_1 = 1, \eta_2 = 0$, and $\eta_1 = 0, \eta_2 = 1$ we get

$$J_{\textcircled{1}}^{(2)} = e_{,11}^2 + 2e_{,11}\,e_{,22} + e_{,22}^2 = \big(\Delta e\big)^2, \quad J_{\textcircled{2}}^{(2)} = 4(e_{,11}\,e_{,22} - e_{,12}^2). \tag{131}$$

The second one is four times the determinant of the Hessian of e.

For $m = 3$ we get

$$\mathbf{W}_e^{(3)} = \tfrac{1}{2}\begin{bmatrix} \eta_1 & 0 & \eta_1 - 3\eta_2 & 0 \\ 0 & \eta_2 & 0 & \eta_1 - 3\eta_2 \\ \eta_1 - 3\eta_2 & 0 & \eta_1 & 0 \\ 0 & \eta_1 - 3\eta_2 & 0 & \eta_1 \end{bmatrix}. \tag{132}$$

Setting $\eta_1 = 1, \eta_2 = 0$, and $\eta_1 = 0, \eta_2 = 1$ we get

$$J_{\textcircled{1}}^{(3)} = e_{,111}^2 + 3e_{,111}e_{,122} + 3e_{112}e_{222} + e_{,222}^2, J_{\textcircled{2}}^{(3)} = 9(e_{,112}^2 - e_{,111}e_{,122} - \ e_{,112}e_{222} + e_{,122}^2), \tag{133}$$

the physical significance of which has not been worked out.

6.4 Pressure Invariants

The computations for pressures use the flat form of the C-vector. Results are identical to those found for volumetric strain — only the replacement of e by p is needed. The base invariants are tabulated for convenience:

$$
\begin{aligned}
I^{(0)}_{\textcircled{1}} &= p, \\
I^{(2)}_{\textcircled{1}} &= p_{,11} + p_{,22} = \Delta\, p, \\
J^{(1)}_{\textcircled{1}} &= p_{,1}^2 + p_{,2}^2 = ||\nabla p||_2^2, \\
J^{(2)}_{\textcircled{1}} &= p_{,11}^2 + 2p_{,11}\, p_{,22} + p_{,22}^2 = \big(\Delta\, p\big)^2, \\
J^{(2)}_{\textcircled{2}} &= p_{,11}\, p_{,22} - p_{,12}^2 = \det\big(\mathcal{H}(p)\big). \\
J^{(3)}_{\textcircled{1}} &= p_{,111}^2 + 3p_{,111}p_{,122} + 3p_{112}p_{222} + p_{,222}^2, \\
J^{(3)}_{\textcircled{2}} &= 9(p_{,112}^2 - p_{,111}p_{,122} - \, p_{,112}p_{222} + p_{,122}^2).
\end{aligned}
\tag{134}
$$

in which $\mathcal{H}(p)$ denotes the Hessian of p excluding the third dimension.

7 Bilinear And Quadratic FIC Forms

In the derivation of FIC functionals, stabilization terms configured as bilinear and quadratic forms repeatedly emerge:

$$
\text{B-forms: } \mathbf{v}_1^T\, \mathbf{Q}\, \mathbf{v}_2, \quad \text{Q-forms: } \mathbf{v}^T\, \mathbf{Q}\, \mathbf{v}, \tag{135}
$$

in which $\mathbf{v}$ are field C-vectors, and $\mathbf{Q}$ are real-valued kernel matrices. In bilinear (B) forms, $\mathbf{v}_1$ and $\mathbf{v}_2$ must be conjugate, for example displacements and body force derivatives of the same order. (Normally a factor of $\frac{1}{2}$ appears in front of a quadratic form, but it will be omitted for brevity.) For stabilization effectiveness, kernel matrices are usually taken to be symmetric; if so $\mathbf{v}_1$ and $\mathbf{v}_2$ can be interchanged in a bilinear form.

To reduce the effort in choosing a specific $\mathbf{Q}$, it is convenient to constraint its entries so that the associated form value does not change under rotation. If so $\mathbf{Q}$ is called an *invariant kernel*. For commonly occuring field combinations, invariant kernels are catalogued below for derivative orders 1 and 2, which are those that occur in FIC functionals truncated to $O(h^2)$ steplengths. They were computed using the software presented in Appendix B. Subscripts S and D tag symmetric and diagonal versions of the kernel matrix, respectively, whereas η_i denote free parameters.

7.1 Forms Involving Displacements and/or Body Force Derivatives

Here k denotes the MDCC for $m = 2$. Quadratic form $(\mathbf{u}^{(1)})^T\,\mathbf{Q}^{(1)}\,\mathbf{u}^{(1)}$:

$$\mathbf{Q}_S^{(1)} = \begin{bmatrix} \eta_1 & -\eta_2 & \eta_2 & \eta_1-\eta_3-\eta_4 \\ & \eta_3 & \eta_4 & -\eta_2 \\ & & \eta_3 & \eta_2 \\ symm & & & \eta_1 \end{bmatrix}, \quad \mathbf{Q}_D^{(1)} = \eta_1\,\mathbf{I}_4. \tag{136}$$

Quadratic form $(\mathbf{u}^{(2)})^T\,\mathbf{Q}^{(2)}\,\mathbf{u}^{(2)}$:

$$\mathbf{Q}_S^{(2)} = \begin{bmatrix} \eta_1 & -\eta_2 & \xi_1 & k\eta_2 & \xi_2 & 0 \\ & \eta_3 & -\eta_2 & \eta_5 & 0 & \xi_2 \\ & & \eta_4 & 0 & \eta_5 & -k\eta_2 \\ & & & \eta_4 & \eta_2 & \xi_1 \\ & & & & \eta_3 & \eta_2 \\ symm & & & & & \eta_1 \end{bmatrix}, \; \mathbf{Q}_D^{(2)} = \eta_1\,\mathbf{diag}\,[1,\,2/k^2,1,\,1,\,2/k^2,\,1]. \tag{137}$$

in which $\xi_1 = \frac{1}{2}(\eta_1 - k^2\eta_3 + \eta_4)$ and $\xi_2 = \eta_1/k - \eta_4/k - \eta_5$. For the bilinear form $(\mathbf{b}^{(1)})^T\,\mathbf{Q}^{(1)}\,\mathbf{u}^{(1)}$, $\mathbf{Q}_S^{(1)}$ and $\mathbf{Q}_D^{(1)}$ are the same as those in (136). Bilinear form $(\mathbf{b}^{(2)})^T\,\mathbf{Q}^{(2)}\,\mathbf{u}^{(2)}$:

$$\mathbf{Q}_S^{(2)} = \begin{bmatrix} \eta_1 & 0 & \eta_1-\eta_2 & 0 & 0 & 0 \\ & \eta_2 & 0 & 0 & 0 & 0 \\ & & \eta_1 & 0 & 0 & 0 \\ & & & \eta_1 & 0 & \eta_1-\eta_2 \\ & & & & \eta_2 & 0 \\ symm & & & & & \eta_1 \end{bmatrix}, \quad \mathbf{Q}_D^{(2)} = \eta_1\,\mathbf{I}_6. \tag{138}$$

Quadratic forms involving only body force derivatives do not occur in statics.

7.2 Forms Involving Pressure and/or Volumetric Strain Derivatives

Here g denotes the MDCC for $m = 2$. Quadratic form $(\mathbf{p}^{(m)})^T\,\mathbf{Q}^{(m)}\,\mathbf{p}^{(m)}$ for $m = 1, 2$:

$$\mathbf{Q}_S^{(1)} = \eta_1\,\mathbf{I}_2, \quad \mathbf{Q}_S^{(2)} = \begin{bmatrix} \eta_1 & 0 & \eta_1 - \dfrac{2\eta_2}{g^2} \\ 0 & \eta_2 & 0 \\ \eta_1 - \dfrac{2\eta_2}{g^2} & 0 & \eta_1 \end{bmatrix}, \quad \mathbf{Q}_D^{(2)} = \eta_1\,\mathbf{diag}\,[1,\,\tfrac{1}{2}g^2,\,1]. \tag{139}$$

Quadratic form $(\mathbf{e}^{(m)})^T\,\mathbf{Q}^{(m)}\,\mathbf{e}^{(m)}$ for $m = 1, 2$:

$$\mathbf{Q}_S^{(1)} = \eta_1\,\mathbf{I}_2, \quad \mathbf{Q}_S^{(2)} = \begin{bmatrix} \eta_1 & 0 & \eta_1 - \frac{1}{2}g^2\eta_2 \\ 0 & \eta_2 & 0 \\ \eta_1 - \frac{1}{2}g^2\eta_2 & 0 & \eta_1 \end{bmatrix}, \quad \mathbf{Q}_D^{(2)} = \eta_1\,\mathbf{diag}\,[1,\,2/g^2,\,1]. \tag{140}$$

Bilinear form $(\mathbf{p}^{(m)})^T\,\mathbf{Q}^{(m)}\,\mathbf{e}^{(m)}$ for $m = 1, 2$:

$$\mathbf{Q}_S^{(1)} = \eta_1\,\mathbf{I}_2, \quad \mathbf{Q}_S^{(2)} = \eta_1\,\mathbf{I}_3. \tag{141}$$

8 Conclusions

The present paper has been devoted to building a catalog of field transformations and associated invariants for higher order spatial derivatives that appear in variational and weak formulations of Finite Increment Calculus (FIC). The main contributions of this study are as follows.

(I) All derivations are carried out in a matrix framework regardless of derivative order. This is accomplished by vector casting all fields. That framework is completely natural for FEM implementations based on either functional or weak FIC forms.

(II) For fields of order two or higher, mixed derivative casting coefficients (MDCC) are introduced in the C-vectors. This is the first study in which such coefficients are kept symbolic, allowing for adjusting their choice as convenient; e.g., to avoid carrying square roots along.

(III) Explicit consideration of compatibility equations in forming strain C-vectors. Those constraints, which emerge for second and higher strain and stress derivatives, have been largely ignored in the literature.

(IV) The strain-displacement relations for strain derivative orders ≥ 2 are believed to be new.

(V) The calculation of invariants is carried out entirely in a matrix framework, avoiding any prior knowledge of tensor calculus. Once a rotational frame transformation is established, linear and quadratic invariants follow by solving matrix equations well studied in control theory. The method also works, with minor modifications, if the field is not a classical vector or tensor.

(VI) The spectral analysis of transformations presented in Appendix A is shown to explicitly illuminate the underlying invariant structure. Such symbolic eigenanalysis would be more difficult in a non-matrix framework.

The chief limitation is the restriction to two dimensions. This was done primarily for timeliness, spurred by the goal of applying results to 2D FIC models and quickly assessing the importance of invariance in the choice of steplengths. While extension to 3D is conceptually straightforward, it is expected to require more computational resources to do symbolic computations. The computer implementation presented in Appendix B should provide a starting point to achieve that goal.

Minor limitations involve the exclusion of rotational derivatives in the strain C-vectors as well as of terms that couple strain or displacement derivatives of different order. Such terms do appear in some multipolar constitutive models, but are not believed to be important for variational FIC.

Conflict of interest

There is not a conflict of interest statement in the manuscript.

A Spectral Properties

This Appendix present information on eigenvalues and eigenvectors of tensor field transformations. The results are useful in providing understanding regarding invariance properties of those transformations.

A.1 Displacements And Body Forces

A.1.1 Characteristic Polynomials

Denote $\mathcal{P}(\mathbf{T}^{(m)}) = \det\left(\mathbf{T}^{(m)} - \lambda\,\mathbf{I}_{2m+2}\right)$, $j = \sqrt{-1}$, and use tag (*) to flag a multiplicity-2 root pair. Formulas tabulated below apply equally to $\mathbf{T}_u^{(m)}$ and $\mathbf{T}_b^{(m)}$:

m	$\mathcal{P}(\mathbf{T}^{(m)})$	Roots of $\mathcal{P}(\mathbf{T}^{(m)}) = 0$
0	$1+\lambda^2-2\lambda c$	$c \pm js$
1	$(1-\lambda)^2\,(1+\lambda^2-2\lambda c_2)$	$1(*),\ c_2 \pm js_2$
2	$(1+\lambda^2-2\lambda c)^2\,(1+\lambda^2-2\lambda c_3)$	$c \pm js$ **(*)**, $c_3 \pm js_3$
3	$(1-\lambda)^2\ (1+\lambda^2-2\lambda c_2)^2\,(1+\lambda^2-2\lambda c_4)$	$1(*),\ c_2 \pm js_2$**(*)**, $c_4 \pm js_4$
4	$(1+\lambda^2-2\lambda c)^2\,(1+\lambda^2-2\lambda c_3)^2\,(1+\lambda^2-2\lambda c_5)$	$c \pm js$**(*)**, $c_3 \pm js_3$**(*)**, $c_5 \pm js_5$

(142)

Because of (29), all expressions are MDCC independent.

A.1.2 Spectral Decompositions

Three equivalent forms of the spectral decomposition of displacement transformation matrices are

$$\mathbf{T}_u^{(m)}\,\mathbf{V}_{uR}^{(m)} = \mathbf{V}_{uR}^{(m)}\,\mathbf{S}_u^{(m)},\ \ \mathbf{S}_u^{(m)} = \mathbf{V}_L^{(m)}\,\mathbf{T}_u^{(m)}\,\mathbf{V}_R^{(m)},\ \ \mathbf{T}_u^{(m)} = \mathbf{V}_{uR}^{(m)}\,\mathbf{S}_u^{(m)}\,\mathbf{V}_{uL}^{(m)},\ \text{ with }\ \mathbf{V}_R^{(m)}\,\mathbf{V}_L^{(m)} = \mathbf{I}. \tag{143}$$

Here $\mathbf{S}^{(m)}$ is the diagonal matrix of eigenvalues of $\widehat{\mathbf{T}}_u^{(m)}$, $\mathbf{V}_{uR}^{(m)}$ the matrix of right eigenvectors stored as columns, and $\mathbf{V}_{uL}^{(m)}$ the matrix of left eigenvectors stored as rows. The diagonal entries of $\mathbf{S}^{(m)}$ are the characteristic polynomial roots listed in (142). Since stacking order is important, they are specified in further detail next:

$$\begin{aligned}
\mathbf{S}^{(0)} &= \mathbf{diag}[\lambda_1^{(0)}, \lambda_2^{(0)}],\\
\mathbf{S}^{(1)} &= \mathbf{diag}[1, 1, \lambda_3^{(1)}, \lambda_4^{(1)}],\\
\mathbf{S}^{(2)} &= \mathbf{diag}[\lambda_1^{(2)}, \lambda_2^{(2)}, \lambda_3^{(2)}, \lambda_4^{(2)}, \lambda_5^{(2)}, \lambda_6^{(2)}],\\
\mathbf{S}^{(3)} &= \mathbf{diag}[1, 1, \lambda_3^{(3)}, \lambda_4^{(3)}, \lambda_5^{(3)}, \lambda_6^{(3)}, \lambda_7^{(3)}, \lambda_8^{(3)}],\\
\mathbf{S}^{(4)} &= \mathbf{diag}[\lambda_1^{(4)}, \lambda_2^{(4)}, \lambda_3^{(4)}, \lambda_4^{(4)}, \lambda_5^{(4)}, \lambda_6^{(4)}, \lambda_7^{(4)}, \lambda_8^{(4)}, \lambda_9^{(4)}, \lambda_{10}^{(4)}].
\end{aligned} \tag{144}$$

in which

$$\begin{aligned}
&\lambda_1^{(0)}=\cos\varphi-j\sin\varphi,\quad \lambda_2^{(0)}=\cos\varphi+j\sin\varphi,\\
&\lambda_3^{(1)}=\cos 2\varphi-j\sin 2\varphi,\quad \lambda_4^{(1)}=\cos 2\varphi+j\sin 2\varphi,\\
&\lambda_1^{(2)}=\lambda_2^{(2)}=\cos 2\varphi-j\sin 2\varphi,\quad \lambda_3^{(2)}=\lambda_4^{(2)}=\cos 2\varphi+j\sin 2\varphi,\\
&\lambda_3^{(3)}=\lambda_4^{(3)}=\cos 2\varphi+j\sin 2\varphi,\quad \lambda_5^{(3)}=\cos 4\varphi-j\sin 4\varphi,\quad \lambda_6^{(3)}=\cos 4\varphi+j\sin 4\varphi,\\
&\lambda_1^{(4)}=\lambda_2^{(4)}=\cos\varphi-j\sin\varphi,\quad \lambda_3^{(4)}=\lambda_4^{(4)}=\cos\varphi+j\sin\varphi,\quad \lambda_5^{(4)}=\lambda_6^{(4)}=\cos 3\varphi-j\sin 3\varphi,\\
&\lambda_7^{(4)}=\lambda_8^{(4)}=\cos 3\varphi+j\sin 3\varphi,\quad \lambda_9^{(4)}=\cos 5\varphi-j\sin 5\varphi,\quad \lambda_{10}^{(4)}=\cos 5\varphi+j\sin 5\varphi.
\end{aligned} \tag{145}$$

The eigenvector matrices listed below were obtained through *Mathematica*. Their vectors are not normalized. Two linearly independent vectors are shown for eigenvalues of multiplicity two; any linear combination of which is also an eigenvector. For $m \in [2,4]$ arbitrary MDCC are allowed.

$$\mathbf{V}_L^{(3)} = \begin{bmatrix} -j & 1 \\ j & 1 \end{bmatrix}, \quad \mathbf{V}_R^{(2)} = \tfrac{1}{2}\begin{bmatrix} j & -j \\ 1 & 1 \end{bmatrix}, \tag{146}$$

$$\mathbf{V}_L^{(1)} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & -1 & 1 & 0 \\ -1 & -j & -j & 1 \\ -1 & j & j & 1 \end{bmatrix}, \quad \mathbf{V}_R^{(1)} = \tfrac{1}{4}\begin{bmatrix} 2 & 0 & -1 & -1 \\ 0 & -2 & j & -j \\ 0 & 2 & j & -j \\ 2 & 0 & 1 & 1 \end{bmatrix}, \tag{147}$$

$$\mathbf{V}_L^{(2)}(k) = \begin{bmatrix} 3 & 2j/k & 1 & j & 2/k & 3j \\ -1 & -2j/k & 1 & j & -2/k & -j \\ 3 & -2j/k & 1 & -j & 2/k & -3j \\ 1 & -2j/k & -1 & j & 2/k & -j \\ 1 & 2j/k & -1 & j & -2/k & -j \\ -j & -2/k & j & -1 & 2j/k & 1 \end{bmatrix}, \quad \mathbf{V}_R^{(2)}(k) = \tfrac{1}{8}\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & j \\ 0 & jk & 0 & jk & -jk & -k \\ 1 & 2 & 1 & -2 & -1 & -j \\ -j & -2j & j & -2j & -j & -1 \\ 0 & -k & 0 & k & -k & -jk \\ -j & 0 & j & 0 & j & 1 \end{bmatrix}, \tag{148}$$

$$\mathbf{V}_L^{(3)}(k) = \begin{bmatrix} 2 & 0 & 2/k & 0 & 0 & 2/k & 0 & 2 \\ 0 & 2/k & 0 & 2 & -2 & 0 & -2/k & 0 \\ 4 & 6j/k & 0 & 2j & 2j & 0 & 6j/k & -4 \\ -2j & 4/k & 2j/k & 0 & 0 & -2j/k & 4/k & 2j \\ 4 & -6j/k & 0 & -2j & -2j & 0 & -6j/k & -4 \\ 2j & 4/k & -2j/k & 0 & 0 & 2j/k & 4/k & -2j \\ 1 & 3j/k & -3/k & -j & j & -3/k & -3j/k & 1 \\ 1 & -3j/k & -3/k & j & -j & -3/k & 3j/k & 1 \end{bmatrix}, \tag{149}$$

$$\mathbf{V}_R^{(3)}(k) = \frac{1}{16}\begin{bmatrix} 3 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & k & 0 & k & 0 & k & -jk & jk \\ k & 0 & k & -2jk & k & 2jk & -k & -k \\ 0 & 3 & -2j & -3 & 2j & -3 & j & -j \\ 0 & -3 & -2j & -3 & 2j & -3 & -j & j \\ k & 0 & -k & 2jk & -k & -2jk & -k & -1 \\ 0 & -k & 0 & k & 0 & k & jk & -jk \\ 3 & 0 & -1 & 0 & -1 & 0 & 1 & 1 \end{bmatrix}, \tag{150}$$

$$\mathbf{V}_L^{(4)}(k_1,k_2)=\begin{bmatrix} 2 & 0 & 4/k_2 & 0 & 2 & 2j & 0 & 4j/k_2 & 0 & 2j \\ -2j & 4/k_1 & 0 & 4/k_1 & 2j & -2 & -4j/k_1 & 0 & -4j/k_1 & 2 \\ 2 & 0 & 4/k_2 & 0 & 2 & -2j & 0 & -4j/k_2 & 0 & -2j \\ 2j & 4/k_1 & 0 & 4/k_1 & -2j & -2 & 4j/k_1 & 0 & 4j/k_1 & 2 \\ -1 & -4j/k_1 & 6/k_2 & 4j/k_1 & -1 & j & -4/k_1 & -6j/k_2 & 4/k_1 & j \\ -3j & 8/k_1 & 6j/k_2 & 0 & j & 1 & 0 & 6/k_2 & 8j/k_1 & -3 \\ -1 & 4j/k_1 & 6/k_2 & -4j/k_1 & -1 & -j & -4/k_1 & 6j/k_2 & 4/k_1 & -j \\ 3j & 8/k_1 & -6j/k_2 & 0 & -j & 1 & 0 & 6/k_2 & -8j/k_1 & -3 \\ 1 & 4j/k_1 & -6/k_2 & -4j/k_1 & 1 & j & -4/k_1 & -6j/k_2 & 4/k_1 & j \\ j & 4/k_1 & -6j/k_2 & -4/k_1 & j & 1 & -4j/k_1 & -6/k_2 & 4j/k_1 & 1 \end{bmatrix}, \tag{151}$$

$$\mathbf{V}_R^{(4)}(k_1,k_2)=\frac{1}{32}\begin{bmatrix} 3 & 2j & 3 & -2j & 1 & 2j & 1 & -2j & 1 & -j \\ 0 & k_1 & 0 & k_1 & 0 & k_1 & 0 & k_1 & -jk_1 & k_1 \\ k_2 & 0 & k_2 & 0 & k_2 & 0 & k_2 & 0 & -k_2 & jk_2 \\ 0 & k_1 & 0 & k_1 & -2jk_1 & k_1 & 2jk_1 & k_1 & jk_1 & -k_1 \\ 3 & -2j & 3 & 2j & -3 & -2j & -3 & 2j & 1 & -j \\ -3j & -2 & 3j & -2 & -3j & 2 & 3j & 2 & -j & 1 \\ 0 & jk_1 & 0 & -jk_1 & -2k_1 & -jk_1 & -2k_1 & jk_1 & -k_1 & jk_1 \\ -jk_2 & 0 & jk_2 & 0 & jk_2 & 0 & -jk_2 & 0 & jk_2 & -k_2 \\ 0 & jk_1 & 0 & -jk_1 & 0 & -jk_1 & 0 & jk_1 & k_1 & -jk_1 \\ -3j & 2 & 3j & 2 & j & -2 & -j & -2 & -j & 1 \end{bmatrix}. \tag{152}$$

It is remarkable that the only dependence on φ is through the eigenvalues listed in (144).

Eigenvectors corresponding to complex eigenvalues have the property that the dot product with itself is zero. Such vectors are called *isotropic*, and are important in the study of symplectic spaces; see [1, 2].

A.1.3 Spectral Form Of Quadratic Invariants

Substituting $\mathbf{T}_u^{(m)} = \mathbf{V}_L^{(m)}\,\mathbf{S}_\varphi^{(m)}\,\mathbf{V}_R^{(m)}$, in which $\mathbf{V}_R^{(m)} = \big(\mathbf{V}_L^{(m)}\big)^{-1}$, premultiplying by $(\mathbf{V}_L^{(m)})^T$ and postmultiplying by $(\mathbf{V}_R^{(m)})^T$, yields the spectral form of (44):

$$\mathbf{D}\,\mathbf{Y}^{(m)}\,\mathbf{D}-\mathbf{Y}^{(m)}=\mathbf{0},\quad \text{in which } \mathbf{D}=\mathbf{S}_\varphi^{(m)} \text{ and } \mathbf{Y}^{(m)}=\big(\mathbf{V}_L^{(m)}\big)^T\,\mathbf{W}^{(m)}\,\mathbf{V}_L^{(m)}. \tag{153}$$

The only dependence on φ is through the diagonal matrix $\mathbf{D}$, which stacks the eigenvalues listed in (144). We seek nontrivial solutions $\mathbf{Y}^{(m)}\neq\mathbf{0}$, that are symmetric but not necessarily real. Symbolically

processing the matrix equations for $m = [0, 4]$ through *Mathematica* yields

$$\mathbf{Y}^{(0)} = \begin{bmatrix} 0 & Y_{12} \\ Y_{12} & 0 \end{bmatrix}, \quad \mathbf{Y}^{(1)} = \begin{bmatrix} Y_{11} & Y_{12} & 0 & 0 \\ Y_{12} & Y_{22} & 0 & 0 \\ 0 & 0 & 0 & Y_{34} \\ 0 & 0 & Y_{34} & 0 \end{bmatrix}, \quad \mathbf{Y}^{(2)} = \begin{bmatrix} 0 & 0 & Y_{13} & Y_{14} & 0 & 0 \\ 0 & 0 & Y_{23} & Y_{24} & 0 & 0 \\ Y_{13} & Y_{23} & 0 & 0 & 0 & 0 \\ Y_{14} & Y_{24} & 0 & 0 & 0 & 0 \\ ,0 & 0 & 0 & 0 & 0 & Y_{56} \\ 0 & 0 & 0 & 0 & Y_{56} & 0 \end{bmatrix},$$

$$\mathbf{Y}^{(3)} = \begin{bmatrix} Y_{11} & Y_{12} & 0 & 0 & 0 & 0 & 0 & 0 \\ Y_{12} & Y_{22} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & Y_{35} & Y_{36} & 0 & 0 \\ 0 & 0 & 0 & 0 & Y_{45} & Y_{46} & 0 & 0 \\ 0 & 0 & Y_{35} & Y_{45} & 0 & 0 & 0 & 0 \\ 0 & 0 & Y_{36} & Y_{46} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & Y_{78} \\ 0 & 0 & 0 & 0 & 0 & 0 & Y_{78} & 0 \end{bmatrix}, \tag{154}$$

$$\mathbf{Y}^{(4)} = \begin{bmatrix} 0 & 0 & Y_{13} & Y_{14} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & Y_{23} & Y_{24} & 0 & 0 & 0 & 0 & 0 & 0 \\ Y_{13} & Y_{23} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ Y_{14} & Y_{24} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & Y_{57} & Y_{58} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & Y_{67} & Y_{68} & 0 & 0 \\ 0 & 0 & 0 & 0 & Y_{57} & Y_{67} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & Y_{58} & Y_{68} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & Y_{9\,10} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & Y_{9\,10} & 0 \end{bmatrix}.$$

The pattern of nonzero elements, dictated by the eigenvalue configuration in $\mathbf{S}_{\varphi}^{(m)}$, continues through higher orders. From inspection the number of free parameters is

$$N_{uJ}^{(m)} = 2m + 2 - \text{mod}(m, 2) = \text{floor}\big((4/3) \times \text{floor}(3n/2)\big). \tag{155}$$

The physical weighting matrix of (154) may be recovered as $\mathbf{W}_u^{(m)} = \mathbf{V}_R^T \mathbf{Y}_u^{(m)} \mathbf{V}_R$.

Comparison of the *Mathematica* performance in using (44) versus (153) for invariant computations indicates that use of (44) is preferable. Transforming the spectral solutions back to physical field values runs into serious complications: matrices $\mathbf{V}_R$ and $\mathbf{V}_L$ have complex entries, and the occurrence of multiple eigenvalues for $m \geq 1$ disrupts the uniqueness of the eigenvector basis. The spectral solutions (154) have the visual advantage, however, that their pattern is readily visualized for any derivative order.

A.2 Strains And Stresses

A.2.1 Spectral Decomposition

The triple product (89) for $m > 0$ is a similarity transformation. Consequently

1. $\mathbf{T}^{(m)}_{\epsilon\varphi}$ and $\mathbf{T}^{(m+1)}_{u\,\varphi}$ have the same eigenvalues, which were studied in A.1.2.

2. The right eigenvectors of $\mathbf{T}^{(m+1)}_{u\,\varphi}$ are rotated by $\mathbf{T}^{(m)}_{\epsilon\,u}$.

For $m = 0$ an extra zero eigenvalue, associated with the infinitesimal rigid rotation, is introduced.

The spectral decompostion of $\mathbf{T}^{(m)}$ is denoted as

$$\mathbf{T}^{(m)}_{\epsilon} = \mathbf{V}^{(m)}_{\epsilon L}, \mathbf{D}^{m}_{\epsilon}\, \mathbf{V}^{(m)}_{\epsilon R}, \tag{156}$$

This is closely linked to that of the displacemnent transformations studied in A.1.1, since

$$\mathbf{D}^{(m)}_{\epsilon} = \mathbf{D}^{(m+1)}_{u}, \quad \mathbf{V}^{(m)}_{\epsilon L} = \mathbf{V}^{(m+1)}_{uL}\, \mathbf{T}^{(m)}_{\epsilon}, \quad \mathbf{V}^{(m)}_{\epsilon R} = \mathbf{V}^{(m+1)}_{uR}\, \mathbf{T}^{(m)}_{\epsilon}. \tag{157}$$

Since the eigenvalues repeat, except for the adjustment of m, those need not be given. The eigenvector matrices are listed below for $m = 1, 2$.

$$\mathbf{V}^{(1)}_{\epsilon L} = \begin{bmatrix} 0 & -2j & -2 & 0 & j & 1 \\ -j & 1 & -j & 1 & 0 & 0 \\ 0 & 2j & -2 & 0 & -j & 1 \\ j & 1 & j & 1 & 0 & 0 \\ -1 & -j & 1 & j & -j & 1 \\ -1 & j & 1 & -j & j & 1 \end{bmatrix}, \quad \mathbf{V}^{(1)}_{\epsilon R} = \begin{bmatrix} 1 & 3j & 1 & -3j & -1 & -1 \\ j & 1 & -j & 1 & j & -j \\ -1 & j & -1 & -j & 1 & 1 \\ -j & 3 & j & 3 & -j & j \\ -2j & 2 & 2j & 2 & 2j & -2j \\ 2 & 2j & 2 & -2j & 2 & 2 \end{bmatrix}. \tag{158}$$

For $m = 2$, the reduced strain vector is used:

$$\mathbf{V}^{(2)}_{\epsilon L} = \begin{bmatrix} 0 & 1 & 0 & 0 & -1 & 0 & -1 & 0 & 1 \\ 0 & 0 & -2 & -2 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & -2j & 2j & -1 & 0 & 1 & 0 & 1 \\ -1 & -j & 1 & -1 & -j & 1 & 0 & 0 & 0 \\ 0 & -1 & 2j & -2j & -1 & 0 & 1 & 0 & 1 \\ -1 & j & 1 & -1 & j & 1 & 0 & 0 & 0 \\ j & -1 & -j & -j & 1 & j & -1 & -j & 1 \\ -j & -1 & j & j & 1 & -j & -1 & j & 1 \end{bmatrix}, \; \mathbf{V}^{(2)}_{\epsilon R} = \tfrac{1}{16} \begin{bmatrix} 0 & 2 & 6 & 2j & -4 & -2j & -4 & -j & j \\ 4 & 0 & 0 & 0 & 4j & 0 & -4j & -2 & -2 \\ 0 & -2 & 2 & 2j & 0 & -2j & 0 & j & -j \\ 0 & -2 & 2 & -2j & 0 & 2j & 0 & j & -j \\ -4 & 0 & 0 & 0 & 4j & 0 & -4j & 2 & 2 \\ 0 & 2 & 6 & -2j & 4 & 2j & 4 & -j & j \\ -4 & 0 & 0 & 4 & 4j & 4 & -4j & -2 & -2 \\ 0 & 8 & 8 & 0 & 0 & 0 & 0 & 4j & -4j \\ 4 & 0 & 0 & 4 & 4j & 4 & -4j & 2 & 2 \end{bmatrix} \tag{159}$$

in which

$$\mathbf{D} = \mathbf{diag}\big[1,\ 1,\ c_2 - js_2,\ c_2 - js_2,\ c_2 + js_2,\ c_2 + js_2,\ c_4 - js_4,\ c_4 + js_4\ \big]$$

$$\mathbf{Y}^{T} = \begin{bmatrix} 0 & 1 & 0 & 0 & -1 & 0 & -1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & -1 & -2j & 2j & -1 & 0 & 1 & 1 \\ -1 & -j & 1 & -1 & -j & 1 & 0 & 0 \\ 0 & -1 & 2j & -2j & -1 & 0 & 1 & 1 \\ -1 & j & 1 & -1 & j & 1 & 0 & 0 \\ j & -1 & -3j & -3j & 1 & j & -1 & 1 \\ -j & -1 & 3j & 3j & 1 & -j & -1 & 1 \end{bmatrix}, \quad \mathbf{X} = \tfrac{1}{16} \begin{bmatrix} 0 & 6 & 2j & -4 & -2j & -4 & -j & j \\ 4 & 0 & 0 & 4j & 0 & -4j & -2 & -2 \\ 0 & 2 & 2j & 0 & -2j & 0 & j & -j \\ 0 & 2 & -2j & 0 & 2j & 0 & j & -j \\ -4 & 0 & 0 & 4j & 0 & -4j & 2 & 2 \\ 0 & 6 & -2j & 4 & 2j & 4 & -j & j \\ -4 & 0 & 4 & 4j & 4 & -4j & -2 & -2 \\ 4 & 0 & 4 & 4j & 4 & -4j & 2 & 2 \end{bmatrix} \tag{160}$$

A.3 Volumetric Strain And Pressure

A.3.1 Spectral Decomposition

The spectral decomposition of $\mathbf{T}_e^{(m)}$ and $\mathbf{T}_p^{(m)}$ are denoted as

$$\mathbf{T}_e^{(m)} = \mathbf{V}_{eL}^{(m)}\,\mathbf{D}_e^{(m)}\,\mathbf{V}_{eR}^{(m)}, \quad \mathbf{T}_p^{(m)} = \mathbf{V}_{pL}^{(m)}\,\mathbf{D}_p^{(m)}\,\mathbf{V}_{pR}^{(m)}. \tag{161}$$

Here $\mathbf{D}^{(m)}$ is the diagonal matrix of eigenvalues, whereas $\mathbf{V}_{*L}^{(m)}$ and $\mathbf{V}_{*R}^{(m)}$ are matrices of left and right eigenvectors, respectively. The eigenvaues of $\mathbf{T}_e^{(m)}$ and $\mathbf{T}_p^{(m)}$ are the same:

$$\mathbf{D}^{(1)} = \mathbf{diag}[c - j\,s, c + j\,s], \tag{162}$$

$$\mathbf{D}^{(2)} = \mathbf{diag}[1, c_2 - j\,s_2, c_2 + j\,s_2], \tag{163}$$

$$\mathbf{D}^{(3)} = \mathbf{diag}[c - j\,s, c + j\,s, c_3 - j\,s_3, c_3 + j\,s_3], \tag{164}$$

The eigenvectors for the volumetric strain transformation matrices are

$$\mathbf{V}_{eL}^{(1)} = \begin{bmatrix} j & -j \\ 1 & 1 \end{bmatrix}, \quad \mathbf{V}_{eR}^{(1)} = \tfrac{1}{2}\begin{bmatrix} -j & 1 \\ j & 1 \end{bmatrix}, \tag{165}$$

$$\mathbf{V}_{eL}^{(2)} = \begin{bmatrix} 1 & -1 & -1 \\ 0 & jg & -j\,g \\ 1 & 1 & 1 \end{bmatrix}, \quad \mathbf{V}_{eR}^{(2)} = \tfrac{1}{4}\begin{bmatrix} 2 & 0 & 2 \\ -1 & -2j/g & 1 \\ -1 & 2j/g & 1 \end{bmatrix}, \tag{166}$$

$$\mathbf{V}_{eL}^{(3)} = \tfrac{1}{3}\begin{bmatrix} 3j & -3j & -3j & 3j \\ g & g & -3g & -3g \\ j\,g & -j\,g & 3j\,g & -3j\,g \\ 3 & 3 & 3 & 3 \end{bmatrix}, \quad \mathbf{V}_{eR}^{(3)} = \tfrac{1}{8}\begin{bmatrix} -3j & 3/g & -3j/g & 3 \\ 3j & 3/g & 3j/g & 3 \\ j & -3/g & -3j/g & 1 \\ -j & -3/g & 3j/g & 1 \end{bmatrix}, \tag{167}$$

The eigenvectors for the pressure transformation matrices are

$$\mathbf{V}_{pL}^{(1)} = \begin{bmatrix} -j & -j \\ 1 & 1 \end{bmatrix}, \quad \mathbf{V}_{pR}^{(1)} = \tfrac{1}{2}\begin{bmatrix} -j & 1 \\ j & 1 \end{bmatrix}, \tag{168}$$

$$\mathbf{V}_{pL}^{(2)} = \begin{bmatrix} 1 & -1 & -1 \\ 0 & 2j/g & -2j/g \\ 1 & 1 & 1 \end{bmatrix}, \quad \mathbf{V}_{pR}^{(2)} = \tfrac{1}{4}\begin{bmatrix} 2 & 0 & 2 \\ -1 & -j\,g & 1 \\ -1 & jg & 1 \end{bmatrix}, \tag{169}$$

$$\mathbf{V}_{pL}^{(3)} = \begin{bmatrix} j & -j & -j & j \\ 1/g & 1/g & -3/g & -3/g \\ j/g & -j/g & 3j/g & -3j/g \\ 1 & 1 & 1 & 1 \end{bmatrix}, \quad \mathbf{V}_{pR}^{(3)} = \tfrac{1}{8}\begin{bmatrix} -3j & g & -j\,g & 3 \\ 3j & g & jg & 3 \\ j & -g & -j\,g & 1 \\ -j & -g & jg & 1 \end{bmatrix}, \tag{170}$$

B Mathematica Implementation

The Appendix presents a *Mathematica* implementation for obtaining invariance results reported in the main body of the paper. This may be of interest as departing point for extending those calculations to three dimensional fields, a task that will likely require use of computer algebra on a high performance computer.

B.1 Constructors

Constructors generate objects, typically matrices and vectors, which are later manipulated by computational modules called processors. They return only one object, thus fitting the classical definition of function. Two types are distinguished: primitive and interface. Primitive constructors are self-contained. Interface constructors provide covers to lower level primitives. Constructrs should never be changed except to correct mistakes or streamline some operations.

B.1.1 Field Cast Vector

Primitive constructor `FCVector`, listed in Figures 4–5, returns a symbolic cast vector (C-vector) for a specified 2D field as well as its spatial derivatives up to the implemented order. The module is invoked as

```
v=FCVector[field]
```
(171)

Argument `field` is a 4-item list: { `fld,m,tag,mdcc` }, which defines the field as follows:

- `fld` A one-character string that specifies the field type. `"u"`: displacements, `"b"`: body forces, `"`ϵ`"`: strains, `"`σ`"`: stresses, `"e"`: volumetric strain, or `"p"`: pressure.
- `m` Field derivative order: up to 4 for displacements and body forces; up to 3 for strains, stresses, volumetric strains and pressures. If `m=0`, the field proper is returned.
- `tag` A one character string that distinguishes C vector variants for strains and stresses:
  `"a"`: augmented strains or stresses (used for `m=0`).
  `"r"`: reduced strains or stresses (used for `m=2,3`),
  `"c"`: conjugate strains or stresses (used for `m=2,3`).
  Ignored for fields other than strains or stresses, in which case `" "` is recommended.
- `mdcc` The mixed-derivative casting coefficient(s) (MDCC) to be used for `m=2,3,4`. If `m=2,3` there is only one coefficient and this argument is a scalar, such as `mdcc=k` for displacements and body forces, or `mdcc=g` for strains, stresses, volumetric strain or pressure. If `m=4`, which is only pertinent for displacements and body forces, a 2-entry list must be provided: `mdcc=`{ `k1,k2` }. Coefficients may be numeric or symbolic. This argument is ignored if `m=0,1`, but a dummy argument (such as 0 or 1) must appear.

The function returns the C-vector with symbolic components assigned standard names. For example,

`v=CVector[{"u",2," ",k}]` returns `v={u111,k*u112,u122,u211,k*u212,u222}`

Table 3: **Square Matrix Entry Pattern Identifier**

First char of `mxpat`	Entry Pattern
`"U"`	General unsymmetric
`"S"`	General symmetric
`"B"`	Bisymmetric: symmetric about both diagonals
`"X"`	Symmetric about main diagonal, antisymmetric about anti-diagonal
`"D"`	Diagonal
`"E"`	Antidiagonal

`v=CVector[{"p",3," ",g}]` returns `v={p111,(3/g)*p112,(3/g)*p122,p222}`

If `fld` is `"p"` or `"e"`, and `m=0`, a one-entry vector is returned although the field is scalar. If one or more items are incorrect or not implemented, the function returns `Null`. All symbolic components of the returned C-vector are globally initialized inside the module; thus `FCVector` should be called only before actual processing starts.

B.1.2 Symbolic Array Constructors

Figure 6 lists two primitives: `SymbVector` and `SymbSquareMatrix`, which construct symbolic vectors and square matrices, respectively. They are referenced as

```
v=SymbVector[letter,n,ib]                (172)
S=SymbSquareMatrix[letter,n,mxpat]       (173)
```

The arguments are:

- `letter` The first letter (Roman or Greek) of the generated entry names.
- `n` Vector or matrix order.
- `ib` For `SymbVector`, entry index base. Usually 0. (-1 is OK.) See example below.
- `mxpat` For `SymbMatrixOrder`, a character string whose first character specifies the entry pattern. See Table 3. The second character is ignored by this module.

The function returns the constructed vector or matrix. Generated matrix entries have two integers: row and column index, which follow the specified letter. The exemptions are patterns that generated diagonal or anti-diagonal matrices, in which case only the row index is appended to the letter.

Examples. `SymbVector["w",4,0]` returns `{w1,w2,w3,w4}` whereas `SymbVector["w",4,2]` returns `{w3,w4,w5,w6}`. The call `SymbSquareMatrix["A",2,"S"]` returns the symmetric matrix `{{A11,A12},{A12,A22}}`.

Figure 6 lists two companion modules, which are referenced as

```
v=SquareMatrixEntries[S,mxpat]           (174)
```

```
v=SquareMatrixSymbols[S,mxpat]
```
(175)

The arguments for both are

`S` A square matrix, usually symbolic.

`mxpat` The matrix entry pattern identifier defined in Table 3.

Module `SquareMatrixEntries` returns a list of matrix entries from the unique portion of the matrix; for example the upper triangle if `maxpat="S"`. Module `SquareMatrixSymbols` returns only unique symbolic entries (that is, entries with `Symbol` head, while eliminating any duplicates.

Example: suppose `W` is the bisymmetric matrix `{{W11,0,W13},{0,W22,0},{W13,0,W11}}`. Then `SquareMatrixSymbols[W,"B"]` returns `{W11,W13,W22}`.

B.1.3 Rotated Field Transformation Matrix

Field transformation matrices that depend on the rotation angle φ may be obtained by calling the interface constructor module `RFTMatrix`, which is listed in Figure 7. (RFT is the acronym for Rotated Field Transformation.) The module is invoked as

```
T=RFTMatrix[field,φ]
```
(176)

The arguments are:

`field` A 4-item list that specifies the field. See B.1.1.

φ The rotation angle, which may be numeric or symbolic.

The function returns the appropriate transformation matrix. Examples:

`Tu=TMatrix["u",2," ",k,φ]` returns the matrix (24).

`Tϵ=TMatrix["ϵ",3," ",g,φ]` returns the matrix (95).

If one or more arguments are incorrect or unimplemented, the function returns `Null`.

```
FCVector[{fld_,m_,tag_,mdcc_}]:=Module[{v=Null,k,k1,k2,g,h,h1,h2,
   atag=tag===“a”,btag=tag===“ ”,ctag=tag===“c”,rtag=tag===“r”},
   If [m==2||m==3, k=g=mdcc; h=m/k];
   If [m==4,{k1,k2}=mdcc; h1=4/k1; h2=6/k2];
   If [fld==“u”, ClearAll[u1,u2,u11,u12,u21,u22,u111,u112,u122,
       u211,u212,u222,u1111,u1112,u1122,u1222,u2111,u2112,u2122,
       u2222,u11111,u11112,u11122,u11222,u12222,u21111,u21112,
       u21122,u21222,u22222];
       If [m==0, v={u1,u2}];
       If [m==1, v={u11,u12,u21,u22}];
       If [m==2, v={u111,k*u112,u122,u211,k*u212,u222}];
       If [m==3, v={u1111,k*u1112,k*u1122,u1222,
                    u2111,k*u2112,k*u2122,u2222}];
       If [m==4, v={u11111,k1*u11112,k2*u11122,k1*u11222,u12222,
                    u21111,k1*u21112,k2*u21122,k1*u21222,u22222}];
      ];
   If [fld==“b”, ClearAll[b1,b2,b11,b12,b21,b22,b111,b112,b122,
       b211,b212,b222,b1111,b1112,b1122,b1222,b2111,b2112,b2122,
       b2222,b11111,b11112,b11122,b11222,b12222,b21111,b21112,
       b21122,b21222,b22222];
       If [m==0, v={b1,b2}];
       If [m==1, v={b11,b12,b21,b22}];
       If [m==2, v={b111,h*b112,b122,b211,h*b212,b222}];
       If [m==3, v={b1111,h*b1112,h*b1122,b1222,
                    b2111,h*b2112,h*b2122,b2222}];
       If [m==4, v={b11111,h1*b11112,h2*b11122,h1*b11222,b12222,
                    b21111,h1*b21112,h2*b21122,h1*b21222,b22222}];
      ];
   If [fld==“ε”, ClearAll[ε11,ε22,γ12,ω3,ε111,ε112,ε221,ε222,
       γ121,γ122,ε1111,ε1112, ε1122,ε2211,ε2212,ε2222,γ1211,γ1212,
       γ1222,ε11111,ε11112,ε11122,ε11222,ε22111,ε22112,ε22122,
       ε22222,γ12111,γ12112,γ12122,γ12222];
       If [m==0 && btag, v={ε11,ε22,γ12}];
       If [m==0 && atag, v={ε11,ε22,γ12,2*ω3}];
       If [m==1, v={ε111,ε112,ε221,ε222,γ121,γ122}];
       If [m==2 && btag, v={ε1111,g*ε1112,ε1122,ε2211,g*ε2212,
                            ε2222,γ1211,γ1212,γ1222}];
       If [m==2 && ctag, v={ε1111,g*ε1112,ε1122,ε2211,g*ε2212,
                            ε2222,γ1211,g*(ε1122+ε2211),γ1222}];
       If [m==2 && rtag, v={ε1111,g*ε1112,ε1122,ε2211,g*ε2212,
                            ε2222,γ1211,γ1222}];
       If [m==3 && btag, v={ε11111,g*ε11112,g*ε11122,ε11222,ε22111,
                            g*ε22112,g*ε22122,ε22222,γ12111,
                            g*γ12112,g*γ12122,γ12222}];
       If [m==3 && ctag, v={ε11111,g*ε11112,g*ε11122,ε11222,
                   ε22111,g*ε22112,g*ε22122,ε22222,γ12111,
                   g*(ε11122+ε22111),g*(ε11222+ε22112),γ12222}];
       If [m==3 && rtag, v={ε11111,g*ε11112,g*ε11122,ε11222,
           ε22111,g*ε22112,g*ε22122,ε22222,γ12111,γ12222}];
       ];
```

Figure 4: Field C-vector constructor module `FCVector`, Part 1.

```
If [fld=="σ", ClearAll[σ11,σ22,σ12,m3,σ111,σ112,σ221,σ222,
    σ121,σ122,σ1111,σ1112,σ1122,σ2211,σ2212,σ2222,σ1211,σ1212,
    σ1222,σ11111,σ11112,σ11122,σ11222,σ22111,σ22112,σ22122,
    σ22222,σ12111,σ12112,σ12122,σ12222];
    If [m==0 && btag, v={σ11,σ22,σ12}];
    If [m==0 && atag, v={σ11,σ22,σ12,m3}];
    If [m==1, v={σ111,σ112,σ221,σ222,σ121,σ122}];
    If [m==2 && btag, v={σ1111,h*σ1112,σ1122,σ2211,h*σ2212,
                         σ2222,σ1211,h*σ1212,σ1222}];
    If [m==2 && ctag, v={σ1111,h*σ1112,σ1122,σ2211,h*σ2212,
                         σ2222,σ1211,h*σ1212,σ1222}];
    If [m==2 && rtag, v={σ1111,h*σ1112,σ1122+2*σ1212,
                    σ2211+2*σ1212,h*σ2212,σ2222,σ1211,σ1222}];
    If [m==3 && (btag||ctag), v={σ11111,h*σ11112,h*σ11122,σ11222,
                                 σ22111,h*σ22112,h*σ22122,σ22222,
                                 σ12111,h*σ12112,h*σ12122,σ12222}];
    If [m==3 && rtag, v={σ11111,h*σ11112,h*(σ11122+σ12112),
             σ11222+3*σ12122,σ22111+3*σ12112,h*(σ22112+σ12122),
             h*σ22122,σ22222,σ12111,σ12222}]
    ];
If [fld=="p", ClearAll[p,p1,p2,p11,p12,p22,p111,p112,p122,p222];
    If [m==0, v={p}]; If [m==1, v={p1,p2}];
    If [m==2, v={p11,h*p12,p22}];
    If [m==3, v={p111,h*p112,h*p122,p222}];
    ];
If [fld=="e", ClearAll[e,e1,e2,e11,e12,e22,e111,e112,e122,e222];
    If [m==0, v={e}]; If [m==1, v={e1,e2}];
    If [m==2, v={e11,g*e12,e22}];
    If [m==3, v={e111,g*e112,g*e122,e222}];
   ];  Return[v]];
```

Figure 5: Field C-vector constructor module `FCVector`, Part 2.

```
SymbVector[letter_,n_,ib_]:=Module[{i,s,v=Table[0,{n}]},
  Off[General::spell]; For [i=1,i<=n,i++, s=letter<>ToString[ib+i];
  Clear[Evaluate[s]]; v[[i]]=Symbol[s]]; Return[v]];

SymbSquareMatrix[letter_,n_,mxpat_]:=Module[{i,j,v,f,pat,z=" ",
   zEven,zOdd,Sij,S=Table[0,{n},{n}]},
   pat=StringTake[mxpat,1];
   If [pat=="U",
      For [i=1,i<=n,i++, For [j=1,j<=n,j++,
           s=letter<>ToString[i]<>ToString[j]; Clear[Evaluate[s]];
           S[[i,j]]=Symbol[s] ]]];
   If [pat=="S",
      For [i=1,i<=n,i++, For [j=i,j<=n,j++,
           s=letter<>ToString[i]<>ToString[j]; Clear[Evaluate[s]];
           S[[i,j]]=S[[j,i]]=Symbol[s] ]]];
   If [pat=="B",
      For [i=1,i<=n,i++, For [j=i,j<=n,j++,
           If [i+j<n+2,s=letter<>ToString[i]<>ToString[j],
                       s=letter<>ToString[n-j+1]<>ToString[n-i+1]];
           Clear[Evaluate[s]]; S[[i,j]]=S[[j,i]]=Symbol[s] ]]];
   If [pat=="X",
      For [i=1,i<=n,i++, For [j=i,j<=n,j++, If[i+j==n+1, Continue[]];
           If [i+j<n+1,s=letter<>ToString[i]<>ToString[j]; f=1,
               s=letter<>ToString[n-j+1]<>ToString[n-i+1]; f=-1];
           Clear[Evaluate[s]]; Sij=Symbol[s];
           S[[i,j]]=S[[j,i]]=Sij*f]]];
   If [pat=="D",
      S=DiagonalMatrix[SymbVector[letter,n,0]]];
   If [pat=="E",
      v=SymbVector[letter,n,0]; Do[S[[i,n-i+1]]=v[[i]],{i,n}]];
   Return[S]];

SquareMatrixEntries[S_,mxpat_]:=Module[{i,j,k,n,pat,var={}},
  n=Length[S]; pat=StringTake[mxpat,1]; k=Floor[(n+1)/2];
  If [pat=="U",var=Table[S[[i,j]],{i,n},{j,n}]];
  If [pat=="S",var=Table[S[[i,j]],{i,n},{j,i,n}]];
  If [pat=="B",var=Table[S[[i,j]],{i,k},{j,i,n-i+1}]];
  If [pat=="X",var=Table[S[[i,j]],{i,k},{j,i,n-i+1}]];
  If [pat=="D",var=Table[S[[i,i]],{i,n}]];
  If [pat=="E",var=Table[S[[i,n-i+1]],{i,n}]];
  Return[Flatten[var]]];

SquareMatrixSymbols[S_,mxpat_]:=Module[{var},
  var=SquareMatrixEntries[S,mxpat];
  var=Complement[var,DeleteCases[var,_Symbol]];
  Return[var]];
```

Figure 6: Symbolic matrix and vector constructors.

```
RFTMatrix[{fld_,m_,tag_,mdcc_},φ_]:=Module[{Tu=Tb=Tε=Tσ=Te=Tp=Null,k,g,
  k1,k2,atag=tag==="a",btag=tag==" ",ctag=tag==="c",rtag=tag==="r"},
   If [m==2||m==3, k=g=mdcc]; If [m==4,{k1,k2}=mdcc];
   If [fld=="u",
       If [m==0, Tu=TuD0[φ]];      If [m==1, Tu=TuD1[φ]];
       If [m==2, Tu=TuD2[φ,k]];    If [m==3, Tu=TuD3[φ,k]];
       If [m==4, Tu=TuD4[φ,{k1,k2}]];
       Return[Tu]];
   If [fld=="b",
       If [m==0, Tb=TuD0[φ]];      If [m==1, Tb=TuD1[φ]];
       If [m==2, Tb=Transpose[TuD2[-φ,k]]];
       If [m==3, Tb=Transpose[TuD3[-φ,k]]];
       If [m==4, Tb=Transpose[TuD4[-φ,{k1,k2}]]];
       Return[Tb]];
   If [fld=="ε",
       If [m==0 && atag, Tε=TεaD0[φ]];
       If [m==0 && btag, Tε=TεD0[φ]];
       If [m==1, Tε=TεD1[φ]];
       If [m==2 && rtag, Tε=TεrD2[φ,g]];
       If [m==3 && rtag, Tε=TεrD3[φ,g]];
       Return[Tε]];
   If [fld=="σ",
       If [m==0 && atag, Tσ=Transpose[TεaD0[-φ]]];
       If [m==0 && btag, Tσ=Transpose[TεD0[-φ]]];
       If [m==1, Tσ=Transpose[TεD1[-φ]]];
       If [m==2 && rtag, Tσ=Transpose[TεrD2[-φ,g]]];
       If [m==3 && rtag, Tσ=Transpose[TεrD3[-φ,g]]];
       Return[Tσ]];
   If [fld=="e",
       If [m==0, Te=TeD0[φ]];   If [m==1, Te=TeD1[φ]];
       If [m==2, Te=TeD2[φ,g]]; If [m==3, Te=TeD3[φ,g]];
       Return[Te]];
   If [fld=="p",
       If [m==0, Tp=TeD0[φ]];   If [m==1, Tp=TeD1[φ]];
       If [m==2, Tp=Transpose[TeD2[-φ,g]]];
       If [m==3, Tp=Transpose[TeD3[-φ,g]]];
       Return[Tp]];
   ];
```

Figure 7: Module that returns rotated field transformation matrices.

```
SDTMatrix[{fld_,m_,tag_,mdccε_,mdccu_}]:=Module[{Tεu=Tuε=Null,k,g,
  k1,k2,atag=tag==="a",btag=tag==" ",ctag=tag==="c",rtag=tag==="r"},
   If [m==2, k=mdccu; g=mdccε]; If [m==3, {k1,k2}=mdccu; g=mdccε];
   If [fld=="εu",
       If [m==0 && atag, Tεu= TεauD01[]];
       If [m==0 && btag, Tεu= TεuD01[]];
       If [m==1, Tεu=TεuD12[k]];
       If [m==2 && rtag, Tεu=TεruD23[g,k]];
       If [m==2 && btag, Tεu=TεuD23[g,k]];
       If [m==3 && rtag, Tεu=TεruD34[g,k1,k2]];
       If [m==3 && btag, Tεu=TεuD34[g,k1,k2]];
       Return[Tεu]];
   If [fld=="uε",
       If [m==0 && atag, Tuε=TuεaD10[]];
       If [m==0 && btag, Tuε=TuεD10[]];
       If [m==1, Tuε=TuεD21[k]];
       If [m==2 && rtag, Tuε=TuεrD32[g,k]];
       If [m==2 && btag, Tuε=TuεD32[g,k]];
       If [m==3 && rtag, Tuε=TuεrD43[g,k1,k2]];
       If [m==3 && btag, Tuε=TuεD43[g,k1,k2]];
   Return[Tuε]]];

CVTMatrix[{fld_,m_,tag_,mdcc_}]:=Module[H=Null,g=mdcc,
   ctag=tag==="c",rtag=tag==="r"},
   If [fld=="ε",
       If [m==2 && rtag, H=HεrcD2[g]];
       If [m==2 && ctag, H=HεcrD2[g]];
       If [m==3 && rtag, H=HεrcD3[g]];
       If [m==3 && ctag, H=HεcrD3[g]];
   Return[H]]];
```

Figure 8: Modules that return strain-displacement and C-vector version transformations.

B.1.4 Strain-Displacement Transformation Matrix

The interface module `SDTMatrix` listed in Figure 8 constructs transformation matrices that relate strains and displacements, as well as their derivatives up to an implemented order.

As described in 5.4, strains derivatives of order `m` can be linked to displacement derivatives of order `m+1`. The present `SDTMatrix` implementation covers the strain derivative range `m=0,1,2,3`, which corresponds to the displacement derivative range `m=1,2,3,4`, respectively.

Both strain-displacement matrices as well as their displacement-strain inverses may be requested. If the strain-displacement matrix is square, so is its inverse. But if it is rectangular, the returned inverse transformation is its pseudoinverse (the so-called Moore-Penrose inverse), which is also rectangular. This topic is discussed in 5.4.

The module is invoked as

```
T=SDTMatrix[fieldx].
```
(177)

The only argument is a 5-item list similar to (but not identical) to that of `FCVector` described in B.1.1. It is configured as { `fld,m,tag,mdccϵ,mdccu` }, in which

`fld` A two-character string that specifies the fields to be connected:
`"ϵu"`: Get matrix linking strains or strain derivatives to displacement derivatives
`"uϵ"`: Get matrix linking displacements derivatives to strains or strain derivatives.

`m` Strain derivative order: 0 to 3. Note that displacement derivative order is `m+1`

`tag` A one character string that distinguishes C-vector variants for strains. The effect depends on the value of `m`:
If `m=0` and `tag="a"` use augmented strain C-vector; if `tag=" "` use standard vector.
If `m=2,3` and `tag="r"` use reduced strain C-vector to get the transformation matrix, but if `tag=" "` use the complete strain C-vector.

`mdcc`ϵ Specifies the MDCC value g to be used for strains if `m=2,3`. Ignored if `m=0,1`, but a dummy argument must be supplied.

`mdccu` Specifies the MDCC value(s) to be used for displacements if `m=1,2,3`. Ignored if `m=0`. If `m=1,2` this argument is the scalar `k` whereas if `m=3` it is the 2-entry list { `k1,k2` }.

Example: the call `T`ϵ`u1=SDTMatrix["`ϵ`u",1," "," "," "]` returns the 4×4 matrix (83).

B.1.5 C-Vector Version Transformation Matrix

Module `CVTMatrix`, also listed in Figure 8, returns the matrix that link two different versions of C-vectors for the same field. These are identified as **H** matrices in the text. The current implementation is restricted to strains. The module is invoked as

```
H=CVTMatrix[{ fld,m,tag,mdcc }]
```
(178)

`fld` This must be the one-character string "ϵ".

`m` Strain derivative order: 2 or 3, else ignored.

`tag` If `"r"`: return the matrix that links the reduced strain C-vector to the complete one.
If `"c"`: return the matrix that links the complete strain C-vector to the reduced one.

`mdcc` Specifies the MDCC value g to be used for the strain C-vectors.

The function returns the rectangular `H` matrix that transforms the complete to the reduced strain C-vector, or its pseudoinverse.

Example: `H`ϵ`r2=CVTMatrix["`ϵ`",2,"c",g]` returns the 9×8 matrix $\mathbf{H}_{\epsilon r}^{(2)}$ given in (71).

B.2 Transformation Primitives

Constructors `RFTMatrix`, `SDTMatrix` and `CVTMatrix` are interface modules that call lower level primitives that return the specified matrices. Since it is occasionally convenient to invoke those primitives directly, they are briefly described in the following subsections.

B.2.1 Displacement Transformation Primitives

Figure 9 lists primitive modules that return transformation matrices for displacement and their derivatives up to order 4, stored as C-vectors.

There are five of them: `TuD0` through `TuD4`. They take care of derivative orders 0 through 4, respectively, Orders 0 and 1 have only one argument: the rotation angle φ. Orders 2 through 4 have two arguments: the rotation angle φ followed by the MDCC specification. The latter is the scalar `k` for `m=2,3`, or the 2-entry list { `k1,k2` } for `m=4`. Each module returns one object: the transformation matrix. Arguments may be numeric or symbolic.

Transformation matrices for body forces and their derivatives are derived from these primitives by reversing the angle and taking the transpose. See the code of `RFTMatrix` listed in Figure 7.

B.2.2 Strain Transformation Primitives

Figure 10 lists primitive modules that return transformation matrices for strains and their derivatives up to order 3, stored as C-vectors.

Five modules are provided: `T`ϵ`aD0`, `T`ϵ`D0`, `T`ϵ`D1`, `T`ϵ`rD2`, and `T`ϵ`rD3`. The last character of the name is the order m. Orders 0 and 1 have only one argument: the rotation angle φ. Orders 2 and 3 have two arguments: the rotation angle φ followed by the MDCC argument g. Each module returns one function value: the transformation matrix. Arguments may be numeric or symbolic.

Two versions for `m=0` are provided: `T`ϵ`aD0` works with the augmented, 4-component strain C-vector (63) whereas `T`ϵ`D0` works with the standard, 3-component strain C-vector (54). The letter `r` in `T`ϵ`rD2` and `T`ϵ`rD3` indicates that the reduced strain C-vectors (68) and (69) are used.

Transformation matrices for stresses and their derivatives are derived from these primitives by reversing the angle and taking the transpose. See the code of `RFTMatrix` listed in Figure 7.

The listing of Figure 10 also shows less important modules that return auxiliary matrices $\mathbf{H}$ that can be used to recover the complete strain C-vector from the reduced ones.

B.2.3 Volumetric Strain Transformation Primitives

Figure 11 lists primitive modules that return transformation matrices for the volumetric strains and its derivatives up to order 3. There are four of them: `TeD0` through `TeD3`, which take care of orders 0 through 3, respectively, Orders 0 and 1 have only one argument: the rotation angle φ. Orders 2 and 3 have two arguments: the rotation angle φ followed by the MDCC argument g. Each module returns one function value: the transformation matrix. Arguments may be numeric or symbolic.

Transformation matrices for pressure and its derivatives are derived from these primitives by reversing the angle and taking the transpose. See the code of `RFTMatrix` listed in Figure 7.

```
TuD0[φ_]:=Module[{c=Cos[φ],s=Sin[φ]},Return[{{c,s},{-s,c}}]];

TuD1[φ_]:=Module[{c2=Cos[2*φ],s2=Sin[2*φ],Tu},
 Tu={{1+c2,s2,s2,1-c2},{-s2,1+c2,-1+c2,s2},
     {-s2,-1+c2,1+c2,s2},{1-c2,-s2,-s2,1+c2}}/2;
 Return[Tu]];

TuD2[φ_,k_]:=Module[{c=Cos[φ],s=Sin[φ],c3=Cos[3*φ],s3=Sin[3*φ],
 a1,a2,a3,a4,b1,b2,b3,b4,Tu}, a1=3*c+c3; a2=3*c-c3; a3=c+c3;
 a4=c-c3; b1=3*s+s3; b2=3*s-s3; b3=s+s3; b4=s-s3;
 Tu={{   a1, 2*b3/k,   a4,   b3, 2*a4/k,  b2},
     {-k*b3,   2*a3, k*b3,-k*a4,  -2*b4,k*a4},
     {   a4,-2*b3/k,   a1,   b2,-2*a4/k,  b3},
     {  -b3,-2*a4/k,  -b2,   a1, 2*b3/k,  a4},
     { k*a4,   2*b4,-k*a4,-k*b3,   2*a3,k*b3},
     {  -b2, 2*a4/k,  -b3,   a4,-2*b3/k,  a1}}/4;
 Return[Tu]];

TuD3[φ_,k_]:=Module[{c2=Cos[2*φ],s2=Sin[2*φ],c4=Cos[4*φ],
 s4=Sin[4*φ],d1,d2,d3,d4,d5,e1,e2,e3,e4,Tu}, d1=3+4*c2+c4;
 d2=1-c4; d3=3-4*c2+c4; d4=1+4*c2+3*c4; d5=1-4*c2+3*c4;
 e1=2*s2+s4; e2=2*s2-s4; e3=2*s2+3*s4; e4=2*s2-3s4;
 Tu={{   d1, 3*e1/k, 3*d2/k,   e2,   e1,3*d2/k, 3*e2/k,  d3},
     {-k*e1,     d4,     e3, k*d2,-k*d2,   -e4,    -d5,k*e2},
     { k*d2,    -e3,     d4, k*e1, k*e2,    d5,    -e4,k*d2},
     {  -e2, 3*d2/k,-3*e1/k,   d1,  -d3,3*e2/k,-3*d2/k,  e1},
     {  -e1,-3*d2/k,-3*e2/k,  -d3,   d1,3*e1/k, 3*d2/k,  e2},
     { k*d2,     e4,     d5,-k*e2,-k*e1,    d4,     e3,k*d2},
     {-k*e2,    -d5,     e4,-k*d2, k*d2,   -e3,     d4,k*e1},
     {   d3,-3*e2/k, 3*d2/k,  -e1,  -e2,3*d2/k,-3*e1/k,  d1}}/8;
 Return[Tu]];

TuD4[φ_,{k1_,k2_}]:=Module[{c=Cos[φ],s=Sin[φ],c3=Cos[3*φ],
  s3=Sin[3*φ],c5=Cos[5*φ],s5=Sin[5*φ],f1,f2,f3,f4,f5,f6,f7,f8,
  g1,g2,g3,g4,g5,g6,g7,Tu},
  f1=10*c+5*c3+c5; f2=2*c-c3-c5; f3=2*c-3*c3+c5; f4=c+2*c3+c5;
  f5=c-c5; f6=2*c-c3-c5; f7=c3-c5; f8=2*c+3*c3+3*c5;
  g1=2*s+3*s3+s5; g2=2*s+s3-s5; g3=10*s-5*s3+s5; g4=s3+s5;
  g5=s-s5; g6=s-2*s3+s5; g7=2*s-3*s3+3*s5;
  Tu={{     f1,    4*g1/k1,    6*f2/k2,    4*g2/k1,    f3,
            g1,    4*f2/k1,    6*g2/k2,    4*f3/k1,    g3},
      { -k1*g1,       4*f4, 6*k1*g4/k2,       4*f5, k1*g2,
        -k1*f6,      -4*g5, 6*k1*f7/k2,      -4*g6, k1*f3},
      {  k2*f6,-4*k2*g4/k1,       2*f8, 4*k2*g4/k1, k2*f6,
         k2*g2,-4*k2*f7/k1,       2*g7, 4*k2*f7/k1, k2*g2},
      { -k1*g2,       4*f5,-6*k1*g4/k2,       4*f4, k1*g1,
        -k1*f3,      -4*g6,-6*k1*f7/k2,      -4*g5, k1*f6},
      {     f3,   -4*g2/k1,    6*f2/k2,   -4*g1/k1,    f1,
            g3,   -4*f3/k1,    6*g2/k2,   -4*f6/k1,    g1},
      {    -g1,   -4*f6/k1,   -6*g2/k2,   -4*f3/k1,   -g3,
            f1,    4*g1/k1,    6*f2/k2,    4*g2/k1,    f3},
      {  k1*f6,       4*g5,-6*k1*f7/k2,       4*g6,-k1*f3,
        -k1*g1,       4*f4, 6*k1*g4/k2,       4*f5, k1*g2},
      { -k2*g2, 4*k2*f7/k1,      -2*g7,-4*k2*f7/k1,-k2*g2,
         k2*f6,-4*k2*g4/k1,       2*f8, 4*k2*g4/k1, k2*f6},
      {  k1*f3,       4*g6, 6*k1*f7/k2,       4*g5,-k1*f6,
        -k1*g2,       4*f5,-6*k1*g4/k2,       4*f4, k1*g1},
      {    -g3,    4*f3/k1,   -6*g2/k2,    4*f2/k1,    -g1,
            f3,   -4*g2/k1,    6*f2/k2,   -4*g1/k1,     f1}}/16;
  Return[Tu]];
```

Figure 9: Transformation primitives for displacements and their first 4 derivatives.

```
Tε aD0[φ_]:=Module[{c=Cos[φ],s=Sin[φ],c2=Cos[2*φ],s2=Sin[2*φ],Tε},
  Tε={{ 1+c2,1-c2,  s2,0},{1-c2,1+c2,-s2,0},
      {-2*s2,2*s2,2*c2,0},{0,0,0,2}}/2; Return[Tε]];

Tε D0[φ_]:=Module[{c=Cos[φ],s=Sin[φ],c2=Cos[2*φ],s2=Sin[2*φ],Tε},
  Tε={{ 1+c2,1-c2,  s2},{1-c2,1+c2, -s2},
      {-2*s2,2*s2,2*c2}}/2; Return[Tε]];

Tε D1[φ_]:=Module[{c=Cos[φ],s=Sin[φ],c3=Cos[3*φ],
  s3=Sin[3*φ],a1,a2,a3,a4,b1,b2,b3,b4,Tε},
  a1=3*c+c3; a2=3*c-c3; a3=c+c3; a4=c-c3;
  b1=3*s+s3; b2=3*s-s3; b3=s+s3; b4=s-s3;
  Tε={{   a1,   b3,   a4,  b2,  b3,   a4},
      {  -b3,   a1,  -b2,  a4, -a4,   b3},
      {   a4,   b2,   a1,  b3, -b3,  -a4},
      {  -b2,   a4,  -b3,  a1,  a4,  -b3},
      {-2*b3,-2*a4, 2*b3,2*a4,2*a3,-2*b4},
      { 2*a4,-2*b3,-2*a4,2*b3,2*b4, 2*a3}}/4; Return[Tε]];

Tε rD2[φ_,g_]:=Module[{c=Cos[φ],s=Sin[φ],c2=Cos[2*φ],s2=Sin[2*φ],
  c4=Cos[4*φ],s4=Sin[4*φ],d1,d2,d3,d4,d5,d6,d7,d8,
  e1,e2,e3,e4,Dg,Tε},
  d1=3+4*c2+c4; d2=1-c4; d3=3-4*c2+c4; d4=1+4*c2+3*c4;
  d5=1-4*c2+3*c4; d6=1+2*c2+c4; d7=1-2*c2+c4; d8=1+c4;
  e1=2*s2+s4; e2=2*s2-s4; e3=2*s2+3*s4; e4=2*s2-3s4;
  Tε={{   d1,   e1, 3*d2, 3*d2,   e2,  d3,   e1,   e2},
      {-2*e1, 2*d6, 2*e3,-2*e4,-2*d7,2*e2,-2*d2, 2*d2},
      {   d2,  -e1,   d4,   d5,  -e2,  d2,   e2,   e1},
      {   d2,   e2,   d5,   d4,   e1,  d2,  -e1,  -e2},
      {-2*e2,-2*d7, 2*e4,-2*e3, 2*d6,2*e1, 2*d2,-2*d2},
      {   d3,  -e2, 3*d2, 3*d2,  -e1,  d1,  -e2,  -e1},
      {-2*e1,-2*d2,-2*e4, 2*e3, 2*d2,2*e2, 2*d6,-2*d7},
      {-2*e2, 2*d2,-2*e3, 2*e4,-2*d2,2*e1,-2*d7, 2*d6}}/8;
  If [g==2, Return[Tε]];
  Dg=DiagonalMatrix[{1,g/2,1,1,g/2,1,1,1}];
  Tε=Dg.Tε.Inverse[Dg]; Return[Tε]];

Tε rD3[φ_,g_]:=Module[{c=Cos[φ],s=Sin[φ],c3=Cos[3*φ],s3=Sin[3*φ],
  c5=Cos[5*φ],s5=Sin[5*φ],f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,
  f11,f12,f13,g1,g2,g3,g4,g5,g6,g7,g8,g9,g10,g11,Dg,Tε},
  f1=10*c+5*c3+c5; f2=2*c-c3-c5; f3=2*c-3*c3+c5; f4=c+2*c3+c5;
  f5=c-c5; f6=2*c-c3-c5; f7=c3-c5; f8=2*c+3*c3+3*c5;
  f9=6*c+7*c3+3*c5; f10=c3-c5; f11=2*c-5*c3+3*c5; f12=c-3*c3+2*c5;
  f13=4*c+3*c3+c5; g1=2*s+3*s3+s5; g2=2*s+s3-s5; g3=10*s-5*s3+s5;
  g4=s3+s5; g5=s-s5; g6=s-2*s3+s5; g7=2*s-3*s3+3*s5; g8=6*s-7*s3+3*s5;
  g9=2*s+5*s3+3*s5; g10=s+3*s3+2*s5; g11=4*s-3*s3+s5;
  Tε={{   f1,   g1, 2*f2,  4*g2,  4*f2, 2*g2,   f3,  g3,   g1,    f3},
      {-3*g1,   f9, 6*g4, 12*f5,-12*g5, 6*f7,  -g8,3*f3,-3*f6,  3*g2},
      { 3*f2,  -g9, 2*f8, 12*g4,-12*f7, 2*g7, -f11,3*g2, 3*g2,  3*f2},
      {  -g2,   f2,-2*g4,  4*f4, -4*g6,-2*f7,  -g2,  f6,  -f3,    g1},
      {   f6,   g2,-2*f7,  4*g6,  4*f4, 2*g4,   f2,  g2,  -g1,   -f3},
      {-3*g2, -f11,-2*g7,-12*f7,-12*g4, 2*f8,   g9,3*f2, 3*f2, -3*g2},
      { 3*f3,   g8, 6*f7, 12*g5, 12*f5,-6*g4,   f9,3*g1,-3*g2, -3*f6},
      {  -g3,   f3,-2*g2,  4*f2, -4*g2, 2*f2,  -g1,  f1,   f3,   -g1},
      {-2*g1,-2*f6,-4*g5,-4*f12, 4*g10, 4*f5, 2*g2,2*f3,2*f13,-2*g11},
      { 2*f3,-2*g2, 4*f5,-4*g10,-4*f12, 4*g5,-2*f6,2*g1,2*g11, 2*f13}}/16;
  If [g==3,Return[Tε]];
  Dg=DiagonalMatrix[{1,g/3,g/3,1,1,g/3,g/3,1,1,1}];
  Tε=Dg.Tε.Inverse[Dg]; Return[Tε]];
```

Figure 10: Transformation primitives for strains and their first 3 derivatives.

```
TeD0[φ_]:=Module[{}, Return[{{1}}]];

TeD1[φ_]:=Module[{c=Cos[φ],s=Sin[φ]},
          Return[{{c,s},{-s,c}}]];

TeD2[φ_,g_]:=Module[{c=Cos[φ],s=Sin[φ],c2=Cos[2*φ],
     s2=Sin[2*φ],Te}, Te={{1+c2,2*s2/g,1-c2},
        {-g*s2,2*c2,g*s2},{1-c2,-2*s2/g,1+c2}}/2;
     Return[Te]];

TeD3[φ_,g_]:=Module[{c=Cos[φ],s=Sin[φ],c3=Cos[3*φ],
     s3=Sin[3*φ],Te},
     Te={{3*c+c3,3*(s+s3)/g,3*(c-c3)/g,3*s-s3},
          {(-s-s3)*g,c+3*c3,-s+3*s3,(c-c3)*g},
          {(c-c3)*g,s-3*s3,c+3*c3,(s+s3)*g},
          {-3*s+s3,3*(c-c3)/g,-3*(s+s3)/g,3*c+c3}}/4;
     Return[Te]];
```

Figure 11: Transformation primitives for volumetric strain and its first 3 derivatives.

```
TεauD01[]:=Module[{Tεu},
   Tεu={{1,0,0,0},{0,0,0,1},{0,1,1,0},{0,-1,1,0}};
   Return[Tεu]];

TεuD01[]:=Module[{Tεu},
   Tεu={{1,0,0,0},{0,0,0,1},{0,1,1,0}};
   Return[Tεu]];

TεuD12[k_]:=Module[{c1=1/k,Tεu},
   Tεu={{1,0,0,0,0,0},{0,c1,0,0,0,0},{0,0,0,0,c1,0},
        {0,0,0,0,0,1},{0,c1,0,1,0,0},{0,0,1,0,c1,0}};
   Return[Tεu]];

TεruD23[g_,k_]:=Module[{c1=1/k,c2=g/k,Tεu},
   Tεu={{1,0,0,0,0,0,0,0}, {0,c2,0,0,0,0,0,0},
        {0,0,c1,0,0,0,0,0},{0,0,0,0,0,c1,0,0},
        {0,0,0,0,0,0,c2,0},{0,0,0,0,0,0,0,1},
        {0,c1,0,0,1,0,0,0},{0,0,0,1,0,0,c1,0}};
   Return[Tεu]];

TεruD34[g_,k1_,k2_]:=Module[{c1=1/k1,c2=g/k1,c3=g/k2,Tεu},
   Tεu={{1,0,0,0,0,0,0,0,0,0}, {0,c2,0,0,0,0,0,0,0,0},
        {0,0,c3,0,0,0,0,0,0,0},{0,0,0,c1,0,0,0,0,0,0},
        {0,0,0,0,0,0,c1,0,0,0},{0,0,0,0,0,0,0,c3,0,0},
        {0,0,0,0,0,0,0,0,c2,0},{0,0,0,0,0,0,0,0,0,1},
        {0,c1,0,0,0,1,0,0,0,0},{0,0,0,0,1,0,0,0,c1,0}};
   Return[Tεu]];

TεuD23[g_,k_]:=Module[{c1=1/k,c2=g/k,Tεu},
   Tεu={{1,0,0,0,0,0,0,0}, {0,c2,0,0,0,0,0,0},
        {0,0,c1,0,0,0,0,0},{0,0,0,0,0,c1,0,0},
        {0,0,0,0,0,0,c2,0},{0,0,0,0,0,0,0,1},
        {0,c1,0,0,1,0,0,0},{0,0,c2,0,0,c2,0,0},
        {0,0,0,1,0,0,c1,0}};
   Return[Tεu]];

TεuD34[g_,k1_,k2_]:=Module[{c1=1/k1,c2=g/k1,c3=g/k2,Tεu},
   Tεu={{1,0,0,0,0,0,0,0,0,0},  {0,c2,0,0,0,0,0,0,0,0},
        {0,0,c3,0,0,0,0,0,0,0}, {0,0,0,c1,0,0,0,0,0,0},
        {0,0,0,0,0,0,c1,0,0,0}, {0,0,0,0,0,0,0,c3,0,0},
        {0,0,0,0,0,0,0,0,c2,0}, {0,0,0,0,0,0,0,0,0,1},
        {0,c1,0,0,0,1,0,0,0,0}, {0,0,c3,0,0,0,c2,0,0,0},
        {0,0,0,c2,0,0,0,c3,0,0},{0,0,0,0,1,0,0,0,c1,0}};
   Return[Tεu]];
```

Figure 12: Primitive modules for strain-displacement transformation matrices

B.2.4 Strain Displacement Transformation Primitives

Figures 12 and 13 lists the primitive constructors used by `SDTMatrix`. They are often used directly bypassing the interface.

B.2.5 C-Vector Variants Transformation Primitives

Figure 14 lists the primitive constructors used by `CVTMatrix`. Again these may be called directly for convenience.

```
Tuε aD10[]:=Module[{Tuε},
  Tuε={{1,0,0,0},{0,0,1/2,-1/2},{0,0,1/2,1/2},{0,1,0,0}};
  Return[Tuε]];

Tuε D10[]:=Module[{Tuε},
  Tuε={{1,0,0},{0,0,1/2},{0,0,1/2},{0,1,0}};
  Return[Tuε]];

Tuε D21[k_]:=Module[{Tuε},
  Tuε={{1,0,0,0,0,0}, {0,k,0,0,0,0},{0,0,-1,0,0,1},
       {0,-1,0,0,1,0},{0,0,k,0,0,0},{0,0,0,1,0,0}};
  Return[Tuε]];

Tuε rD32[k_,g_]:=Module[{c1=-1/g,c2=k/g,Tuε},
  Tuε={{1,0,0,0,0,0,0,0}, {0,c2,0,0,0,0,0,0},
       {0,0,k,0,0,0,0,0}, {0,0,0,0,c1,0,0,1},
       {0,c1,0,0,0,0,1,0},{0,0,0,k,0,0,0,0},
       {0,0,0,0,c2,0,0,0},{0,0,0,0,0,1,0,0}};
  Return[Tuε]];

Tuε rD43[k1_,k2_,g_]:=Module[{c1=-1/g,c2=k1/g,c3=k2/g,Tuε},
  Tuε={{1,0,0,0,0,0,0,0,0,0}, {0,c2,0,0,0,0,0,0,0,0},
       {0,0,c3,0,0,0,0,0,0,0},{0,0,0,k1,0,0,0,0,0,0},
       {0,0,0,0,0,0,c1,0,0,1},{0,c1,0,0,0,0,0,0,1,0},
       {0,0,0,0,k1,0,0,0,0,0},{0,0,0,0,0,c3,0,0,0,0},
       {0,0,0,0,0,0,c2,0,0,0},{0,0,0,0,0,0,0,1,0,0}};
  Return[Tuε]];

Tuε D32[k_,g_]:=Module[{c1=1+g^2,c2=k/(1+2*g^2),Tuε},
  Tuε={{1,0,0,0,0,0,0,0,0,0},{0,k/g,0,0,0,0,0,0,0,0},
       {0,0,c1*c2,-g^2*c2,0,0,0,g*c2,0},
       {0,0,0,0,-1/g,0,0,0,1},{0,-1/g,0,0,0,0,1,0,0},
       {0,0,-g^2*c2,c1*c2,0,0,0,g*c2,0},
       {0,0,0,0,k/g,0,0,0,0},{0,0,0,0,0,1,0,0,0}};
  Return[Tuε]];

Tuε D43[k1_,k2_,g_]:=Module[{c1=1+g^2,c2=2+g^2,
  c3=g*(2+g^2),c4=-1/g,c5=k1/g,Tuε},
  Tuε={{1,0,0,0,0,0,0,0,0,0,0,0},{0,c5,0,0,0,0,0,0,0,0,0,0},
       {0,0,k2*c1/c3,0,-k2/c2,0,0,0,0,k2/c3,0,0},
       {0,0,0,2*k1/c2,0,-g*k1/c2,0,0,0,0,g*k1/c2,0},
       {0,0,0,0,0,0,c4,0,0,0,0,1},{0,c4,0,0,0,0,0,0,1,0,0,0},
       {0,0,-g*k1/c2,0,2*k1/c2,0,0,0,0,g*k1/c2,0,0},
       {0,0,0,-k2/c2,0,k2*c1/c3,0,0,0,0,k2/c3,0},
       {0,0,0,0,0,0,c5,0,0,0,0,0},{0,0,0,0,0,0,0,1,0,0,0,0}};
  Return[Tuε]];
```

Figure 13: Primitive modules for displacement-strain transformation matrices.

B.3 Invariant Finders

Three computational processes were implemented to obtain the results presented in the main text:

1. Formulation of rotated and interfield transformations using straightforward pattern matching. Symbolic outputs were inlined using `InputForm` to equip the primitive constructor modules listed in the previous subsections.
2. Determination of the spectral decomposition of rotation transformation matrices. This relied largely on built in functions to get eigenvalues and eigenvectors of symbolic matrices.
3. Finding linear, bilinear and quadratic invariants.

Only the software for invariant determination is presented below, since its logic is less trivial than for the other two tasks. This implementation may serve as guide for future extensions to 3D.

```
HεcrD2[g_]:=Module[{H2=Table[0,{9},{8}]},
   For [i=1,i<=7,i++, H2[[i,i]]=1];
   H2[[9,8]]=1; H2[[8,3]]=H2[[8,4]]=g;
   Return[H2]];

HεcrD3[g_]:=Module[{H3=Table[0,{12},{10}]},
   For [i=1,i<=9,i++, H3[[i,i]]=1]; H3[[12,10]]=1;
   H3[[10,3]]=H3[[11,6]]=1; H3[[10,5]]=H3[[11,4]]=g;
   Return[H3]];

HεrcD2[g_]:=Module[{c,g1,g2,g3,H2=Table[0,{8},{9}]},
   c=1/(1+2*g^2); g1=(1+g^2)*c; g2=-g^2*c; g3=g*c;
   For [i=1,i<=7,i++, H2[[i,i]]=1]; H2[[8,9]]=1;
   H2[[3,3]]=H2[[4,4]]=g1; H2[[3,4]]=H2[[4,3]]=g2;
   H2[[3,8]]=H2[[4,8]]=g3; Return[H2]];

HεrcD3[g_]:=Module[{c,g4,g5,g6,g7,H3=Table[0,{10},{12}]},
   c=1/(2+g^2); g4=(1+g^2)*c; g5=g*c; g6=c; g7=2*c;
   For [i=1,i<=9,i++, H3[[i,i]]=1]; H3[[10,12]]=1;
   H3[[3,3]]=H3[[6,6]]=g4; H3[[4,11]]=H3[[5,10]]=g5;
   H3[[3,5]]=H3[[4,6]]=H3[[5,3]]=H3[[6,4]]=-g5;
   H3[[3,10]]=H3[[6,11]]=g6; H3[[4,4]]=H3[[5,5]]=g7;
   Return[H3]];
```

Figure 14: Transformation primitives for H matrices

B.3.1 Weight Array Initialization

Figure 15 lists two modules that initialize vectors and/or matrices for subsequent processing. They are invoked as

$$\{\texttt{v,w,var}\}=\texttt{WeightVectorSetup[field]} \tag{179}$$

$$\{\texttt{v,W,var}\}=\texttt{WeightMatrixSetup[field,mxpat]} \tag{180}$$

Modules `WeightVectorSetup` and `WeightMatrixSetup` set up unknown vectors and matrices used in linear and quadratic invariant calculations, respectively. The arguments are

- field — the 4-item field specification described in B.1.1.
- mxpat — For `WeightMatrixSetup`: the matrix entry pattern identifier defined in Table 3. This matrix is always symmetric (`mxpat="S"`). If the field is displacement or body force, it is convenient to split it as the sum of two matrices with (`mxpat="B"`) and (`mxpat="X"`, respectively. In addition a "custom" pattern may be set by `WeightMatrixSetup` for certain fields if the second character of of `mxpat` is `"C"`; see code listing.

The returned objects are

- v — The field C-vector returned by `FCVector`. Its length is `nv`.
- w — For `WeightVectorSetup`: the initialized weight vector of length `nv`.
- W — For `WeightMatrixSetup`, the initialized weight matrix of order `nv`.
- var — For `WeightVectorSetup`, a copy of `w` to allow its manipulation without touching `w`.

  For `WeightMatrixSetup`, a list of independent entries in `W` built by `SquareMatrixEntries`, which was described in B.1.2.

Entries in `w` and `W` are generated by the constructors `SymbVector` and `SymbSquareMatrix`, respectively, described in B.1.2.

```
WeightVectorSetup[field_]:=Module[{n,v,w,var={}},
  v=FCVector[field]; If [v==Null, Print["WeightVectorSetup: ",
    "bad field data"]; Return[{Null,Null,Null}]];
  n=Length[v]; w=var=SymbVector["w",n,0]; Return[{v,w,var}]];

WeightMatrixSetup[field_,mxpat_]:=Module[{n,v,fld,m,v1,v2,Z,W,var},
  fld=field[[1]]; m=field[[2]];
  v=FCVector[field]; If [v==Null, Print["WeightMatrixSetup: ",
    "bad field data"]; Return[{Null,Null,Null}]]; n=Length[v];
  W=SymbSquareMatrix["W",n,mxpat];
  If [(fld=="u" || fld=="b") && (mxpat=="B" || mxpat=="X"),
      v1={{1,0},{1,0,0,1},{1,0,1,0,1,0},{1,0,1,0,0,1,0,1},
          {1,0,1,0,1,0,1,0,1,0}};
      v2={{0,1},{0,1,1,0},{0,1,0,1,0,1},{0,1,0,1,1,0,1,0},
          {0,1,0,1,0,1,0,1,0,1}};
      Z=Transpose[{v1[[m+1]]}].{v1[[m+1]]}+
        Transpose[{v2[[m+1]]}].{v2[[m+1]]};
      If [mxpat=="X", Z=Table[1,{2*m+2},{2*m+2}]-Z]; W=W*Z];
  If [(fld=="e" || fld=="p"),
      v1={{1/Sqrt[2]},{1,0},{1,0,1},{1,0,1,0}};
      v2={{1/Sqrt[2]},{0,1},{0,1,0},{0,1,0,1}};
      Z=Transpose[{v1[[m+1]]}].{v1[[m+1]]}+
        Transpose[{v2[[m+1]]}].{v2[[m+1]]};
      W=W*Z];
  If [(fld==" " || fld==" ") && m==0,
      W={{W11,W12,0},{W12,W11,0},{0,0,W33}};
      ];
  If [(fld==" " || fld==" ") && m==1,
      W={{W11,W12,W13,  0,W15,W16},{W12,W22,  0,W24,W25,W26},
         {W13,  0,W22,W34,W35,W36},
         {  0,W24,W34,W11,W45,W46},{W15,W25,W35,W45,W55,  0},
         {W16,W26,W36,W46,  0,W55}};
      ];
  var=SquareMatrixSymbols[W,mxpat]; Return[{v,W,var}];
  Return[{v,W,var}]];
```

Figure 15: Modules that initialize weight unknowns.

Example 1: `field={"u",2," ",2}` specifies displacement second derivatives with `k=2`. Then `WeightVectorSetup[field]` returns {{`u111,2*u112,u122,u211,2*u212,u222`}, {`w1,w2,w3,w4,w5,w6`},{`w1,w2,w3,w4,w5,w6`}}. Here `var=w`.

Example 2: `field={"p",2," ",g}` specifies pressure second derivatives with symbolic `g`. Then `WeightMatrixSetup[field,"BE"]` returns {{`p11,(2/g)*p12,p22`},{{`W11,0,W13`},

{`0,W22,0`},{`W13,0,W11`}},{`W11,W13,W22`}}. Here `var`$\neq$`W`.

B.3.2 Invariance Residual Equations

Module `InvResidualEqs`, listed in Figure 16, receives information required to set up and return the invariance equations for both linear and quadratic problems. The module is invoked as

$$\{\texttt{T,R,eqs}\}=\text{InvResidualEqs[field,}\varphi\text{,Wm]} \tag{181}$$

The arguments are

`field` A 4-item list that specifies the field. See B.1.1.

φ The rotation angle symbol.

```
InvResidualEqs[field_,φ_,Wm_]:=Module[{fld=field[[1]],m=field[[2]],
  T,w,W,R={},R0,Rf,Rc,d=Length[Dimensions[Wm]],
  scale=1,n,rep,nmax=10,i,c,clist,eqs={}},
  T=RTFMatrix[field,φ]; If [T==Null, Return[{T,R,eqs}]];
  If [d==1, w=Wm; R=TrigReduce[Simplify[Transpose[T].w-w]]];
  If [d==2, W=Wm; R=TrigReduce[Transpose[T].W.T-W]];
  rep=Flatten[Table[{Sin[n*φ]->0,Cos[n*φ]->0},{n,nmax}]]; R0=R/.rep;
  Rf=Flatten[{R0,Table[{Coefficient[R,Sin[n*φ]],
                        Coefficient[R,Cos[n*φ]]},{n,nmax}]}];
  If [fld=="u"||fld=="b",scale={{1,2,4,8,16},{1,4,16,64,512}}[[d,m+1]]];
  If [fld=="ε"||fld=="σ",scale={{2,4,8,16},{8,32,128,512}}[[d,m+1]]];
  If [fld=="e"||fld=="p",scale={{1,1,2,4},{1,1,8,16}}[[d,m+1]]];
  Rc=DeleteCases[Flatten[Rf],0]; Rc=Simplify[scale*Rc];
  nc=Length[Rc]; If [nc<=0, Return[{T,R,eqs}]]; clist={};
  For [i=1,i<=nc,i++, c=Rc[[i]]; If [c==0,Continue[]];
       If [Length[Position[clist, c]]==0&&
           Length[Position[clist,-c]]==0,
           AppendTo[eqs,c==0]; AppendTo[clist,c]]];
  Clear[Rc,Rf,clist]; Return[{T,R,eqs}]];
```

Figure 16: Module that sets up invariance constraint equations from appropriate residual.

`Wm` For the linear invariant problem, the vector `w` of unknown weights set up by `WeightVectorSetup` via (179). For the quadratic problem, the matrix `W` of unknown weights set up by `WeightMatrixSe` via (180).

The function return is

`T` The rotated field transformation matrix. (`Null` if an error is detected.)

`R` The residual formed by either $\mathbf{R} = \mathbf{w}^T\,\mathbf{T} - \mathbf{w}$ or $\mathbf{R} = \mathbf{T}^T\,\mathbf{W}\,\mathbf{T} - \mathbf{W}$ for the linear or quadratic problem, respectively. Note the use of `TrigReduce` to express all sines and cosines in `R` as multiples of φ.

`eqs` A list of equations suitable for subsequent use of `Solve`. They follow from the condition that each residual component must vanish for any φ. Duplicates (such as { `w1-w2==0,-w1+w2==0` }) and trivial conditions (such as `0==0`) are weeded out. If the number of nontrivial equations happens to be zero, the empty list is returned.

Note: as implemented, `InvResidualEqs` assumes that the largest angle multiple that may appear in sines and cosines is `10*`φ, which occurs for displacement and body force fourth derivatives. To change this value, adjust `nmax` in the local variable list.

Example: for `field=`{ `"p",2," ",1` } and `w=`}`w1,w2,w3` }, the residual is
`R=`{ `(-w1+w3+(w1-w2)*Cos[2*`φ`]-2*w2*Sin[2*`φ`])/2,(-2*w2+2*w2*Cos[2*`φ`]`
`-(w1-w3)*Sin[2*`φ`])/2,(w1-w3+(-w1+w3)*Cos[2*`φ`]-2*w2*Sin[2*`φ`])/2` }.
The invariance equations are `eqs=`{ `-w1+w3==0,-2*w2==0` }.

B.3.3 Invariant Kernel Solver

One variation of `InvResidualEqs` is worth describing here. Module `InvKernelSolver`, listed in Figure 17. This one returns an invariant kernel matrix to be used in stabilization forms of FIC functionals. as covered in a previous section. The module is invoked as

```
Q=InvKernelSolver[field1,field2,mxpat,letter]          (182)
```

```
InvKernelSolver[field1_,field2_,φ_,mxpat_,letter_]:=Module[{v1,v2,
  T1,T2,n1,n2,W,R={},R0,Rf,Rc,n,rep,nmax=10,i,c,clist={},eqs={},sol,Wm,
  modnam="QuadFormInvKernel: "}, v1=FCVector[field1]; n1=Length[v1];
  v2=FCVector[field2]; n2=Length[v2];
  If [v1==Null||v2==Null, Print[modnam,"bad field(s)"]; Return[Null]];
  If [n1!=n2, Print[modnam,"unequal C-vector lengths"]; Return[Null]];
  T1=RTFMatrix[field1,φ]; T2=RTFMatrix[field2,φ];
  W=SymbSquareMatrix[letter,n1,mxpat]; var=SquareMatrixEntries[W,mxpat];
  If [W==Null, Print[modnam,"illegal mxpat"]; Return[Null]];
  R=Flatten[TrigReduce[Transpose[T1].W.T2-W]];
  rep=Flatten[Table[{Sin[n*φ]->0,Cos[n*φ]->0},{n,nmax}]]; R0=R/.rep;
  Rf=Table[{Coefficient[R,Sin[n*φ]],Coefficient[R,Cos[n*φ]]},{n,nmax}];
  Rc=DeleteCases[Flatten[{R0,Rf}],0]; nc=Length[Rc];
  If [nc<=0, Print[modnam,"empty sol"]; Return[Null]];
  For [i=1,i<=nc,i++, c=Rc[[i]]; If [c==0,Continue[]];
       If [Length[Position[clist, c]]==0&&
           Length[Position[clist,-c]]==0,
           AppendTo[eqs,c==0]; AppendTo[clist,c]]];
  Off[Solve::"svars"]; sol=Solve[eqs,var]; Wm=W/.sol[[1]];
  Clear[Rc,Rf,clist]; Return[Wm]];
```

Figure 17: Module that returns invariant kernel for bilinear and quadratic forms of FIC functionals.

```
LinInvSolver[field_,φ_,pvlist_,prt_]:=Module[
 {v,w,var,wsol,T,R,eqs,neq,sol,prtw,prtT,prtR,prtneq,prteqs,
  prtvar,prtsol,nvd=Length[pvlist],modnam="LinInvSolver: "},
 {prtw,prtT,prtR,prtneq,prteqs,prtvar,prtsol}={MemberQ[prt,"w"],
    MemberQ[prt,"T"],  MemberQ[prt,"R"],  MemberQ[prt,"neq"],
    MemberQ[prt,"eqs"],MemberQ[prt,"var"],MemberQ[prt,"sol"]};
  {v,w,var}=WeightVectorSetup[field];
  If [v==Null, Print[modnam,"bad field"]; Return[Null]];
  If [prtw, Print["w=",w]]; If [nvd>0, var=Delete[var,pvlist]];
  {T,R,eqs}=InvResidualEqs[field,φ,w]; neq=Length[eqs];
  If [T==Null, Print[modnam,"Null T"]; Return[Null]];
  If [prtT, Print["T=",T//MatrixForm]];
  If [prtR, Print["R=",R//MatrixForm]];
  If [prtneq, Print[neq," equations formed"]];
  If [prteqs, Print["eqs=",eqs]]; If [prtvar, Print["var=",var]];
  Off[Solve::"svars"]; sol=Simplify[Solve[eqs,var]];
  If [prtsol, Print["solution=",sol]];
  If [Length[sol]<=0, Print[modnam,"empty sol"]; Return[Null]];
  wsol=w/.sol[[1]]; Return[wsol]];
```

Figure 18: Module that solves for linear invariants.

The arguments are

`field1`	4-item list identifying the premultiplication field
`field2`	4-item list identifying the postmultiplication field
φ	The rotation angle symbol
`mxpat`	The desired kernel matrix pattern, e.g., `"S"` for symmetric or `"D"` for diagonal
`letter`	The letter used to label matrix entries

The only output is

`Q`	Kernel matrix that meets invariant conditions if nonzero. A returned null matrix means that those conditions cannot be met by the specified pattern.

The output matrix may be subsequently processed by the cleaning module of B.3.8 if desirable.

```
QuadInvOnePassSolver[field_,φ_,mxpat_,pvlist_,prt_]:=Module[{v,W,
  var,T,R,eqs,neq,sol,prtW,prtT,prtR,prtneq,prteqs,prtvar,prtsol,
  nvd=Length[pvlist],modnam="QuadInvOnePassSolver: "},
 {prtW,prtT,prtR,prtneq,prteqs,prtvar,prtsol}={MemberQ[prt,"W"],
    MemberQ[prt,"T"],  MemberQ[prt,"R"],  MemberQ[prt,"neq"],
    MemberQ[prt,"eqs"],MemberQ[prt,"var"],MemberQ[prt,"sol"]};
  {v,W,var}=WeightMatrixSetup[field,mxpat];
  If [v==Null, Print[modnam,"bad field"]; Return[Null]];
  If [prtW, Print["W=",W//MatrixForm]];
  If [nvd>0, var=Delete[var,pvlist]];
  {T,R,eqs}=InvResidualEqs[field,φ,W]; neq=Length[eqs];
  If [T==Null, Print[modnam,"Null T"]; Return[Null]];
  If [prtT,   Print["T=",T//MatrixForm]];
  If [prtR,   Print["R=",R//MatrixForm]];
  If [prtneq, Print[neq," equations formed"]];
  If [prteqs, Print["eqs=",eqs]]; If [prtvar, Print["var=",var]];
  Off[Solve::"svars"]; sol=Simplify[Solve[eqs,var]];
  If [prtsol,  Print["solution=",sol]];
  If [Length[sol]<=0, Print[modnam,"empty sol"]; Return[Null]];
  W=W/.sol[[1]];  Return[W]];
```

Figure 19: Quadratic invariant solver: one pass implementation..

B.3.4 Linear Invariant Solver

Module `LinInvSolver`, listed in Figure 18, solves for the linear invariants of a specified field under inplane rotations. The module is invoked as

$$\texttt{w=LinInvSolver[field,}\varphi\texttt{,pvlist,prt]} \tag{183}$$

The arguments are

`field`	4-item list defining the field to be investigated for invariants. See B.1.1
φ	The rotation angle symbol.
`pvlist`	Optionally, a position list of passive variables to be removed from `var`. Must be formatted as expected by the `Delete` function. Example: if `var=`{`w1,w2,w3,w4`} and `pvlist=`{{`2`},{`4`}}, `w2` and `w4` are dropped so the variable list passed to `Solve` is {`w1,w3`}. Note that `pvlist=`{`2,4`} would be an error. Ignored if an empty list.
`prt`	A list of internal print requests specified as character strings: `"w"`: print unknown weight vector, `"T"`: print transformation matrix, `"R"`: print residual vector, `"eqs"`: print constraint equations, `"neq"`: print number of constraint equations, `"var"`: print list of active weight variables, `"rep"`: print invariant residual equation solution as replacement list. These may appear in any order. If `prt` is the empty list, nothing is printed.

The function returns

w	The linear invariant weight vector $\mathbf{w}$ if not identically null. If the null vector, no linear invariant was found.

If `w` is nonzero, the generic invariant is the dot product `w.v`. See 4.8 regarding instance retrieval.

The implementation is straightforward. Module `WeightVectorSetup` is called to get the field C-vector $\mathbf{v}$ in `v` and the weight unknown list $\mathbf{w}$ in `w`, which is copied to `var`. Selected `var` entries (the so-called passive variables) may be dropped as per argument `pvlist`. Module `InvResidualEqs` is

```
PVC[nv_,npv_,ntrials_,seed_]:=Module[{i,j,k,n,c,ci,nc,
  sc,nsc,sclen,base=64}, SeedRandom[seed];
  c=Table[0,{ntrials}]; c[[1]]=Range[1,npv];
  For [i=2,i<=ntrials,i++,
       ci=Sort[Table[Random[Integer,{1,nv}],{npv}]];
       If [Length[Split[ci]]==npv, c[[i]]=ci]];
  c=DeleteCases[c,0]; nc=Length[c];
  c=Sort[Table[FromDigits[c[[i]],base],{i,nc}]]; sc=Split[c];
  nsc=Length[sc];  sclen=Table[Length[sc[[i]]],{i,nsc}];
  If [nc>nsc, k=0;
      For [i=1,i<=nsc,i++, k++; n=sclen[[i]]-1;
           If [n>0, For [j=k,j<=k+n-1,j++, c[[j+1]]=0]; k+=n]];
  c=DeleteCases[c,0]]; nc=Length[c]; Clear[sc,scl];
  c=Table[IntegerDigits[c[[i]],base],{i,nc}];
  Return[c]];

PVCombs[nv_,pvdata_]:=Module[{npv,c,nc,npv1,npv2,ncmax,
  ntrials,seed,pvtab={}}, {npv1,npv2,ncmax,ntrials,seed}=pvdata;
  For [npv=npv1,npv<=npv2,npv++, c=PVC[nv,npv,ntrials,seed];
       nc=Length[c]; If [nc>ncmax, c=Part[c,Range[ncmax]]];
       AppendTo[pvtab,c]; Clear[c]]; pvtab=Flatten[pvtab,1];
  Return[pvtab]];
```

Figure 20: Auxiliary modules for quadratic invariant multipass solver.

called to get the equations in `eqs`. (`T` and `R` are returned only to be optionally printed.) The built-in solver function `Solve` is called with `eqs` and `var` as arguments, and the solution placed in `sol`. If the result is not null, `w` is replaced by the solution and returned as function value.

Example 1. Find the linear invariants of pressure second derivatives. Use `g=1` (flat C-vector) and request printing of equations, variables and solution:

```
Clear[φ]; field={"p",2," ",1}; prt={"eqs","var","sol"};
letter=pvlist={}; w=LinInvSolver[field,φ,letter,pvlist,prt];
```

Output shows variables `var={w1,w2,w3}`, equations `eqs={(-w1+w3)/2==0,w2==0,w2/2==0}`, and solution `sol={{w1->w3,w2->0}}`. The module returns `w=var/.sol[[1]]={w3,0,w3}`. The field C-vector is `v=FCVector[field]={p11,p12,p22}`. The dot product `v.w` yields the generic invariant `(p11+p22)w3`, which has only one free parameter. Setting `w3=1` we find that the Hessian trace (the pressure Laplacian) `p11+p22`, or $p_{,11} + p_{,22}$, is the only linear invariant.

The call is now changed to `w=LinInvSolver[field,φ,pvlist,prt]`, in which `pvlist={{1}}` is nonempty. Passive variable `w1` is dropped so `var={w2,w3}` but `w` is not changed. This causes `w1` to survive in the replacement list returned by `Solve`, which is `sol={{w3->w1,w2->0}}`, and the module returns `w={w1,0,w1}`. Here this is a cosmetic change. But the ability to specify passive variables becomes important for the multipass quadratic invariant analysis described in B.3.7.

Example 2. Find the linear invariants of displacement second derivatives. Use flat C-vector, no passive variables and no internal printing:

```
Clear[φ]; field={"u",2," ",1}; w=LinInvSolver[field,φ,{},{}];
```

This returns the null 6-vector `w={0,0,0,0,0,0}`. Consequently there are no linear invariants.

B.3.5 Quadratic Invariant One Pass Solver

Module `QuadInvOnePassSolver`, listed in Figure 19, solves for the quadratic invariants of a specified field under plane rotations. It is a one pass solver as opposed to that described in B.3.7. The module is invoked as

```
W=QuadInvOnePassSolver[field,φ,mxpat,pvlist,prt]
```
(184)

The argument sequence is quite similar to that of `LinInvSolver`, described in B.3.4, whence only the differences are noted below.

`field,`φ Same as in B.3.4.

`mxpat` A list of matrix entry patterns that defines the initial weight matrix and set of original variables; see B.3.1 and Table 3. For instance, { `"S"` } specifies the `"S"` (full symmetric) pattern, which is the most general one. On the other hand, { `"B","X"` } specifies the weight matrix split as the sum of two sparser patterns: `"B"` and `"X"`.

`pvlist` A position list of passive variables to be removed from `var`. The order of these variables may be observed by printing `"var"` in a trial run. Ignored if empty. Note: if the optimal set of passsive variables is obtained from previous runs using `QuadInvMultiPassSolver`, it may be specified here to save time.

`prt` As in B.3.4, except that `"W"` requests print of the initial weight matrix `W`.

The function return is

`W` The quadratic invariant weight matrix.

The logic is virtually identical to that of `LinInvSolver`. The main difference occurs in the residual evaluation within the subordinate module `InvResidualEqs`.

Example. Find quadratic invariants of displacement derivatives, constraining $\mathbf{W}$ to be bisymmetric:

```
Clear[φ]; f={"u",1," ",0}; W=QuadInvOnePassSolver[f,φ,{"B"},{}];
```

This returns the matrix `W={{W14+W22+W23,0,0,W14},{0,W22,W23,0},{0,W23,W22,0},{W14,0,0,W14+W22+W23}}`, which has 3 free parameters: {`W14,W22,W23`}. This result in matrix notation reads

$$\mathbf{W}_u^{(2)} = \begin{bmatrix} W_{14}+W_{22}+W_{23} & 0 & 0 & W_{14} \\ 0 & W_{22} & W_{23} & 0 \\ 0 & W_{23} & W_{22} & 0 \\ W_{14} & 0 & 0 & W_{14}+W_{22}+W_{23} \end{bmatrix}, \tag{185}$$

which is bisymmetric. The field C-vector is `v={u11,u12,u21,u22}`, and the generic invariant may be obtained by forming `v.W.v`.

B.3.6 Auxiliary Modules for Multipass Solver

A difficulty with a result such as (185) is that $\mathbf{W}$ is not unique: any linear combination of rows (or columns) that maintains rank and pattern is a valid solution. The number of such admissible $\mathbf{W}$ grows

```
QuadInvMultPassSolver[field_,φ_,mxpat_,pvdata_,prt_]:=Module[
 {v,W,var,T,R,eqs,neq,nv,nvar,nc,i,pvc,varact,sol,Rch,Wsol,score,
  prtW,prtT,prtR,prtneq,prteqs,prtvar,prtrep,prtpvt,prtsco,
  bestscore=10^30,Wbest,modnam="QuadInvMultPassSolver: "},
  MxScore[M_]:=10^6*Length[DeleteCases[Flatten[M],0]]+LeafCount[M];
 {prtW,prtT,prtR,prtneq,prteqs,prtvar,prtrep,prtsco,prtpvt}=
   {MemberQ[prt,"W"],  MemberQ[prt,"T"],  MemberQ[prt,"R"],
    MemberQ[prt,"neq"],MemberQ[prt,"eqs"],MemberQ[prt,"var"],
    MemberQ[prt,"rep"],MemberQ[prt,"sco"],MemberQ[prt,"pvt"]};
  {v,W,var}=WeightMatrixSetup[field,mxpat]; nv=Length[v];
  If [v==Null, Print[modnam,"bad field"]; Return[Null]];
  If [prtW, Print["W=",W//MatrixForm]];
  If [prtvar, Print["original var=",var]];
  {T,R,eqs}=InvResidualEqs[field,φ,W]; neq=Length[eqs];
  If [T==Null, Print[modnam,"Null T"]; Return[Null]];
  If [prtT,   Print["T=",T//MatrixForm]];
  If [prtR,   Print["R=",R//MatrixForm]];
  If [prtneq, Print[neq," equations formed"]];
  If [prteqs, Print["eqs=",eqs]]; nvar=Length[var];
  pvtab=PVCombs[nvar,pvdata]; nc=Length[pvtab];
  If [prtpvt, Print["pvtab=",pvtab]]; Wbest=Table[0,{nv},{nv}];
  Print[nc," passive var combs to be tried"]; Off[Solve::"svars"];
  For [i=1,i<=nc,i++, pvc=pvtab[[i]]; varact=Delete[var,Split[pvc]];
      sol=Solve[eqs,varact];     If [Length[sol]<=0, Continue[]];
      Rch=Simplify[R/.sol[[1]]]; If [Union[Rch]=!={0}, Continue[]];
      If [prtrep, Print["solution=",sol," for pvc=",pvc]];
      Wsol=W/.sol[[1]]; score=MxScore[Wsol];
      If [score>=bestscore, Continue[]];
      bestscore=score; Wbest=Wsol;
      If [prtsco, Print["best score=",bestscore," for pvc=",pvc]];
      ]; Return[Wbest]];
```

Figure 21: Quadratic invariant solver: multipass implementation.

rapidly as the length of the C-vector increases. This poses two challenges: (i) characterize the "best" $\mathbf{W}$, and (ii) find it in reasonable computer time.

To answer (i) we rate each $\mathbf{W}$ solution with a "matrix score," which is a positive integer. Like in golf, a lower score is better. That of a candidate solution matrix `W` is taken to be `10^6*Nnz+LeafCount[W]`, in which `Nnz` is the number of nonzero entries of `W`, whereas `LeafCount` is a *Mathematica* integer measure of the number of operations needed to evaluate a symbolic matrix (more precisely, the number of subexpressions in the matrix evaluation "tree"). The latter is obtained through an eponymous function: `LeafCount[W]`. The score evaluation can be implemented as a one-line internal function:

```
MxScore[W_]:=10^6*Length[DeleteCases[Flatten[W],0]]+LeafCount[W]
```

(Another matrix score measure tried was `10^6*Nnz+LeafCount[v.W.v]` in which `v` is the field C-vector; this one did not work too well.)

To answer (ii) a reasonable number of passive variable combinations are generated by a statistical sampling process implemented in the module PVCombs, which is listed in Figure 20. It is invoked as

$$\texttt{pvtab=PVCombs[nv,pvdata]} \tag{186}$$

where

- nv — The number of weight variables, which is the length of `var` returned by `WeightMatrixSetup`.
- pvdata — A five integer list: {`npv1,npv2,ncmax,ntrials,seed`}. Here `npv1` and `npv2` specify the range of number of variables to delete, `nc` is a generation bound; `ntrial` is

```
CleanSolution[Ws_,mxpat_,oldlet_,newlet_,ib_,reord_]:=Module[
 {i,j,p,d=Length[Dimensions[Ws]],n=Length[Ws],nr=Length[reord],
  vi,pat=mxpat,W,Wlist,oldvar={},newvar={},nv,ordvar,rep,Wc},
  If [d==1, Wlist=Ws; W=SymbVector[oldlet,n,0]];
  If [d==2, Wlist=SquareMatrixSymbols[Ws,mxpat];
     W=SquareMatrixSymbols[SymbSquareMatrix[oldlet,n,pat],pat]];
  For [i=1,i<=Length[W],i++, vi=W[[i]]; p=Position[Wlist,vi];
       If [Length[p]>0, AppendTo[oldvar,vi]]]; Clear[W,Wlist];
  nv=Length[oldvar]; If [nv<=0, Return[{Ws,{}}]];
  newvar=ordvar=SymbVector[newlet,nv,ib];
  rep=Table[oldvar[[i]]->newvar[[i]],{i,nv}]; Wc=Ws/.rep;
  If [nr==nv, {Wc,newvar}=ReorderFreePars[Wc,ordvar,ib,reord]];
  Return[{Wc,Sort[newvar]}]];

ReorderFreePars[Ws_,oldvar_,ib_,reord_]:=Module[{i,j,s,
  nr=Length[reord],nv=Length[oldvar],midvar={},newvar=oldvar,
  oldlet,midlet="ζ",newlet,rep1,rep2,Wc=Ws},
  If [nv<=0||nr!=nv, Return[{Wc,newvar}]]; s=ToString[oldvar[[1]]];
  newlet=oldlet=StringTake[s,1]; If [oldlet=="ζ", midlet="β"];
  midvar=SymbVector[midlet,nv,ib]; newvar=SymbVector[newlet,nv,ib];
  Do [j=reord[[i]]; newvar[[i]]=oldvar[[j]],{i,nv}];
  rep1=Table[oldvar[[i]]->midvar[[i]],{i,nv}];
  rep2=Table[midvar[[i]]->newvar[[i]],{i,nv}];
  Wc=Simplify[(Wc/.rep1)/.rep2];
  Return[{Wc,Sort[newvar]}]];
```

Figure 22: Modules to clean up invariant solutions and reorder free parameters..

the maximum number of combinations to try for each variable grouping, and `seed` is the initial seed for the built-in random generator. For procedural details the code logic should be studied.

Combinations are built and filtered by the primitive constructor `PVC`, which is listed in Figure 20. The module returns distinct passive variable combinations produced by the random generator.

B.3.7 Quadratic Invariant Multipass Solver

Module `QuadInvOnePassSolver`, listed in Figure 21, solves for the quadratic invariants of a specified field under plane rotations. It is a multipass solver that computes a large number of solution weight matrices, and returns that with best score. The module is invoked as

$$\texttt{Wbest=QuadInvMultiPassSolver[field,}\varphi\texttt{,mxpat,pvdata,prt]} \tag{187}$$

The argument sequence is identical to that of `QuadInvOnePassSolver`, described in B.3.5. with exception of the last two arguments:

pvdata — Passive variable stochastic generation data to be passed to `PVCombs`; see B.3.6.

prt — Same options as for the one-pass solver in B.3.5, with an addition: `"sco"` prints best matrix score updates while cycling over passive variable combinations.

The function return is

`Wbest` — The weight matrix solution with best score.

```
ShowLinInvariants[field_,reord_,prt_]:=Module[{i,iv,φ,
  v,var,nvar,pvlist={},ivar,Inv,vi,ws,wc},
  Print["field=",field//InputForm]; v=FCVector[field];
  ws=LinInvSolver[field,φ,pvlist,prt];
 {wc,var}=CleanSolution[ws,{},"w","η",0,reord]; nvar=Length[var];
  Print["clean sol=",wc,"  ",nvar," free pars: ",var];
  If [MemberQ[prt,"inv"], Inv=Simplify[wc.v];
      For [iv=1,iv<=nvar,iv++, vi=var[[iv]];
           Print["I"<>ToString[iv]<>"=",Coefficient[Inv,vi]]]];
  If [MemberQ[prt,"chk"], T=RTFMatrix[field,φ];
      Print["chk=",Simplify[Transpose[T].wc-wc]]];
  Return[{wc,var}]];

ShowQuadInvariants[field_,pvlist_,mpats_,reord_,prt_]:=Module[{i,iv,
  φ,ib=0,pat,npat,mxpat,v,nv,var,nvar,Inv,vi,T,Ws,Wc,Wout,varout},
  Print["field=",field//InputForm]; v=FCVector[field]; nv=Length[v];
  Wout=Table[0,{nv},{nv}]; varout={}; npat=Length[mpats];
  For [pat=1,pat<=npat,pat++, mxpat=mpats[[pat]];
       Ws=QuadInvOnePassSolver[field,φ,mxpat,pvlist,prt];
      {Wc,var}=CleanSolution[Ws,mxpat,"W","η",ib,reord];
       nvar=Length[var]; Inv=Simplify[v.Wc.v];
       Print["clean sol=",Wc//MatrixForm,"\n",nvar," free pars: ",var];
       If [MemberQ[prt,"inv"], Inv=Simplify[v.Wc.v];
           For [iv=1,iv<=nvar,iv++, vi=var[[iv]];
                Print["J"<>ToString[iv]<>"=",Coefficient[Inv,vi]]]];
       If [MemberQ[prt,"chk"], T=RTFMatrix[field,φ];
           Print["chk=",Simplify[Transpose[T].Wc.T-Wc]//MatrixForm]];
       Wout+=Wc; varout=Flatten[{varout,var}]; ib+=nvar];
  Return[{Wout,varout}]];

ShowBestQuadInvariants[field_,pvdata_,mpats_,reord_,prt_]:=Module[
 {i,iv,φ,ib=0,pat,npat,mxpat,v,nv,var,nvar,Inv,vi,T,Ws,Wc,Wout,varout},
  Print["field=",field//InputForm]; v=FCVector[field]; nv=Length[v];
  Wout=Table[0,{nv},{nv}]; varout={}; npat=Length[mpats];
  For [pat=1,pat<=npat,pat++, mxpat=mpats[[pat]];
       Ws=QuadInvMultPassSolver[field,φ,mxpat,pvdata,prt];
      {Wc,var}=CleanSolution[Ws,mxpat,"W","η",ib,reord];
       nvar=Length[var];
       Print["clean sol=",Wc//MatrixForm,"\n",nvar," free pars: ",var];
       If [MemberQ[prt,"inv"], Inv=Simplify[v.Wc.v];
           For [iv=1,iv<=nvar,iv++, vi=var[[iv]];
                Print["J"<>ToString[iv]<>"=",Coefficient[Inv,vi]]]];
       If [MemberQ[prt,"chk"], T=RTFMatrix[field,φ];
           Print["chk=",Simplify[Transpose[T].Wc.T-Wc]//MatrixForm]];
       Wout+=Wc; varout=Flatten[{varout,var}]; ib+=nvar];
  Return[{Wout,varout}]];
```

Figure 23: Three driver modules that integrate invariant computations.

B.3.8 Solution Cleanup

Module `CleanSolution`, listed in Figure 22, beautifies the solution (vector or matrix) provided by an invariant solver. This action facilitates visualization of the free parameters as well as transfer to a typeset manuscript. The module is invoked as

$$\{\texttt{Wc,newvar}\}\texttt{=CleanSolution[Ws,mxpat,oldlet,newlet,ib,reord]} \qquad (188)$$

The arguments are

Ws	The solution (vector or matrix) to be cleaned.
mxpat	If `Ws` is a matrix solution, the entry pattern identifier. Ignored if a vector.
oldlet	The character string used as first letter of the entries of `Ws`.

newlet The character string to be substituted as first letter of the cleaned solution entries.

ib The index base for the cleaned solution entries. Usually 0.

reord See description of last argument of `ReorderPar` below.

The output is the list { `Wc,newvar` }, in which `Wc` is the cleaned solution and `newvar` is the sorted list of renamed variables. To illustrate the effect, suppose that `Ws` is the matrix (185). Then `CleanSolution[Ws,"BE","W","`η`",0,{ }]` returns `Wc` as { { η`1+`η`2+`η`3,0,0,`η`2 },{ 0,`η`2,`η`3,0 },{ 0,`η`3,`η`2,0 },{ `η`1,0,0,`η`1+`η`2+`η`3 } } whereas the exit `newvar` is { $\eta$`1,`$\eta$`2,`$\eta$`3 }. The cleaned solution in matrix notation is

$$\mathbf{W}_u^{(2)} = \begin{bmatrix} \eta_1+\eta_2+\eta_3 & 0 & 0 & \eta_2 \\ 0 & \eta_2 & \eta_3 & 0 \\ 0 & \eta_3 & \eta_2 & 0 \\ \eta_2 & 0 & 0 & \eta_1+\eta_2+\eta_3 \end{bmatrix}, \tag{189}$$

Here the three free parameters are easier to discern at a glance.

Figure 22 lists also the subordinate module `ReorderFreePars`, which is called by `CleanSolution` to reorder the free parameter list indices as specified through argument `reord`. The module is invoked as

{ `Wc,newvar` }=`ReorderFreePars[Wc,oldvar,ib,reord]` (190)

The arguments are

Wc A solution (vector or matrix) already cleaned by `CleanSolution`.

oldvar List of free parameters to be reordered.

ib Same as for `CleanSolution`.

reord An integer list that reorders the free parameter indices as shown by an example. Suppose that `oldlet` and `newlet` are `"w"` and "η", respectively, and that `Ws` on entry is { `w1,w2,w3` }. The default replacement is { `w1`$\rightarrow$$\eta$`1,w2`$\rightarrow$$\eta$`2,w3`$\rightarrow$$\eta 3$ }. But if `reord` is { `3,1,2` }, it will be changed to { `w1`$\rightarrow$$\eta$`3,w2`$\rightarrow$$\eta$`1,w3`$\rightarrow$$\eta 2$ }. If the length of `reord` does not match the number of different nonzero entries in `Ws`, no change is made. Since that number is typically unknown before a solution is obtained, this argument is normally set *a posteriori*.

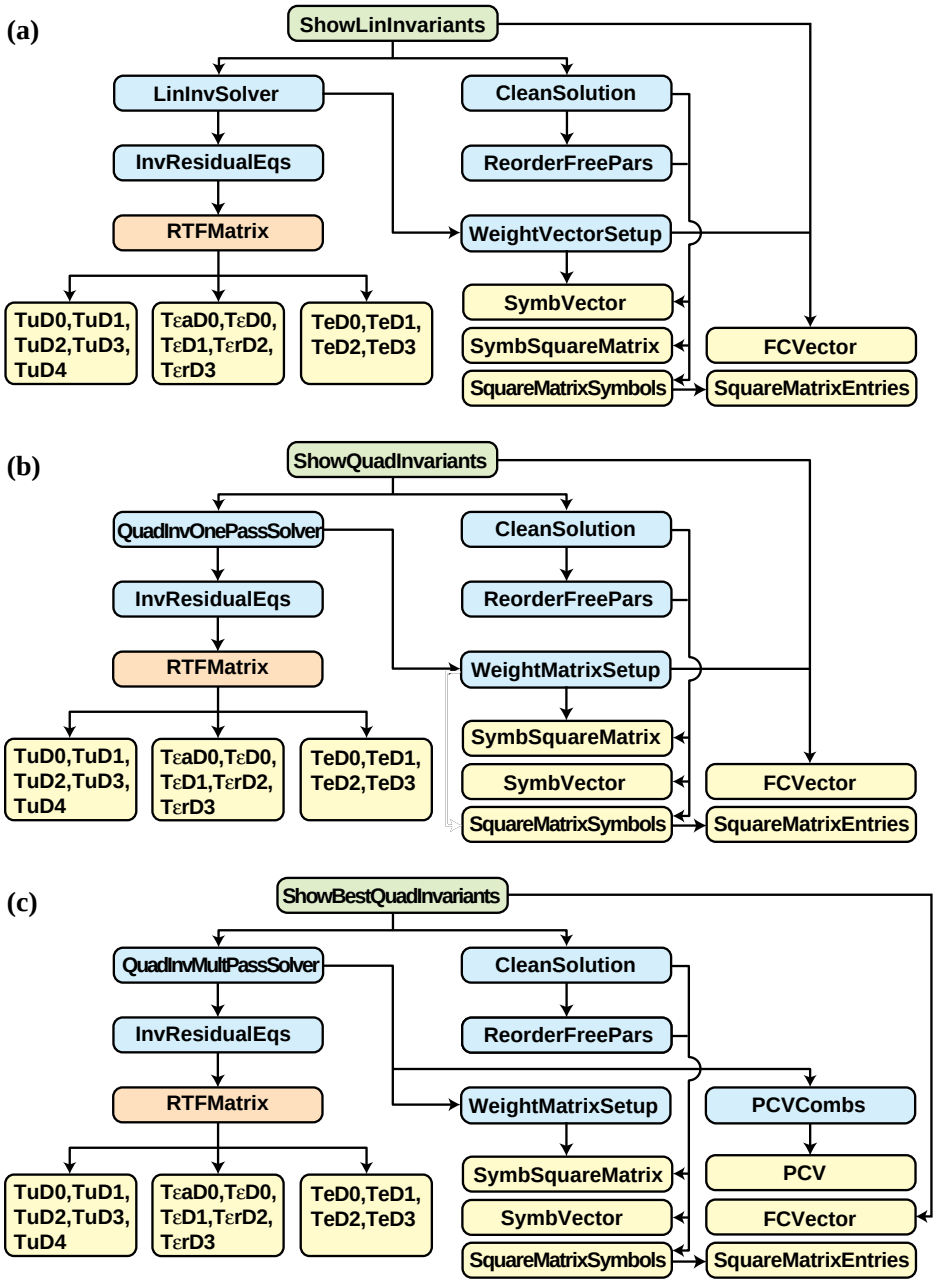

Figure 24: Hierarchical flow diagrams for invariant computations: (a) linear invariants, (b) quadratic invariants with one-pass solution, and (c) quadratic invariants multipass solutions. Box color code: yellow for primitive constructors; beige for interface constructors; blue for processors; green for drivers. $\boxed{A} \rightarrow \boxed{B}$ means that A calls B..

B.3.9 Solve-and-Clean Drivers

To expedite scripting, it is convenient to get-it-and-clean-it invariant solutions in one shot. This is facilitated by the three "show me" driver modules listed in Figure 23. These are invoked as

```
{wc,var}=ShowLinInvariants[field,reord,prt]                    (191)

{Wc,var}=ShowQuadInvariants[field,pvlist,mpats,reord,prt]      (192)

{Wc,var}=ShowBestQuadInvariants[field,pvdata,mpats,reord,prt]  (193)
```

The listed drivers are used to run `LinInvSolver`, `QuadInvOnePassSolver`, and `QuadBestInvSolve` respectively, followed by invoking `CleanSolution`. The arguments are those previously described for those modules, and need not be restated here. The only exception is

`prt` Controls printout of the pertinent solver as described previously. In addition two more specs may be entered: `"inv"` to print individual invariants, and "chk" to print a zero-residual check.

Each driver returns a 2-item list: the clean solution `Wc` and the list of free parameters `var`. Note: if `mpats` contains multiple patterns for the quadratic invariant solvers — for example `mpats={"B","X"}` — the output `Wc` is the sum of the solutions obtained for each pattern while `var` is their union.

Example. Find linear invariants of pressure and its derivatives for $m \leq 3$ using flat C-vectors. Type `For[m=0,m<=3,m++,Print[ShowLinInvariants[{"p",m," ",1},φ,{},{"sol"}]]];` in an input cell and execute.

Figure 24 displays the hierarchical organization of software capped by the foregoing three drivers. Some modules previously described are not present in those diagrams. Those are used for other tasks such as the first two listed at the start of B.3.

Acknowledgement

The work of the first author has been supported through a Visiting Scholar fellowships awarded by the Spanish Ministerio of Educación y Cultura to visit the International Center for Numerical Methods in Engineering (CIMNE), at Barcelona during summer periods from 2007 through 2010.

References

[1] R. Abrahams and J. E. Marsden, *Foundations of Mechanics*, Benjamin Cummins, London, 1978.

[2] E. Artin, *Geometric Algebra*, Interscience Publishers, Geneva, 1957.

[3] F. P. Beer and E. R. Johnston, *Mechanics of Materials*, 2nd ed., McGraw-Hill, NewYork, 1992.

[4] A. Ben-Israel and T. N. E. Greville, *Generalized Inverses: Theory and Applications* Springer-Verlag, New York, 2^{nd} ed., 2003.

[5] G. Birkhoff and S. A. MacLane, *A Survey of Modern Algebra*, 3rd ed., Chelsea Pub. Co., 1988.

[6] E. Cosserat and F. Cosserat, *Théorie des corpes deformables*, Hermann et Fils, Paris, 1909.

[7] A. C. Eringen, *Nonlocal Continuum Feld Theories* Springer-Verlag, Berlin, 2002.

[8] C. A. Felippa and E. Oñate, Nodally exact Ritz discretizations of 1D diffusion-absorption and Helmholtz equations by variational FIC and modified equation methods, *Comput. Mech.*, **39**, 91–112, 2007.

[9] C. A. Felippa, E. Oñate, and S. R. Idelsohn, FIC variational framework for continuum mechanics: variational principles, in preparation.

[10] B. A. Finlayson, *The Method of Weighted Residuals and Variational Principles*, Academic Press, 1972, reprinted by SIAM, 2014

[11] N. A. Fleck and J. W. Hutchinson, Strain gradient plasticity, in J. W. Hutchinson and T. Y. Wu (eds.), *Advances in Applied Mechanics*, Academic Press, **33**, 296–361, 1997.

[12] N. A. Fleck and J. W. Hutchinson, A reformulation of strain gradient plasticity, *J. Mech. Phys. Solids*, **42**, 295–361, 2002.

[13] A. Franci, E. Oñate and J. M. Carbonell, Unified Lagrangian formulation for fluid and solid mechanics and FSI problems, *Comp. Meths. Appl. Mech. Engrg.*, **298**, 520–547, 2006.

[14] M. Frewer, More clarity on the concept of frame indifference in classical continuum mechanics, *Acta Mech.*, **202**, 213–246, 2009.

[15] J. H. Grace and A. Young, *The Algebra of Invariants*, Cambridge Univ. Press, Cambridge, U.K., 1903.

[16] A. Graham, *Kronecker Products and Matrix Calculus with Applications*, Ellis Horwood Ltd., Chichester, UK, 1981.

[17] A. E. Green and R. S. Rivlin, Simple force and stress multipoles, *Arch. Rational Mech. Anal.*, **16**, 325–353, 1964.

[18] A. E. Green and R. S. Rivlin, Multipolar continuum mechanics, *Arch. Rational Mech. Anal.*, **17**, 113–147, 1964.

[19] H. Guggenheimer, *Differential Invariants*, Dover, New York, 1977.

[20] M. E. Gurtin, A gradient theory of single-crystal viscoplasticity that accounts for geometrically necessary dislocations, *J. Mech. Phys. Solids*, **50**, 5–32, 2002.

[21] L. R. Herrmann, Elasticity equations for nearly incompressible materials by a variational theorem, *AIAA J.*, **3**, 1896–1900, 1965.

[22] N. J. Higham, *Functions of Matrices: Theory and Computation*, SIAM, Philadelphia, 2008.

[23] D. Hilbert, *Theory of Algebraic Invariants*, Cambridge Univ. Press, Cambridge, U.K., 1999; a reprint of Hilbert's 1897 lectures at Göttingen.

[24] R. Hill, *The Mathematical Theory of Plasticity*, Oxford Univ. Press, London, 1950.

[25] S. R. Idelsohn, E. Oñate and F. Del Pin, The Particle Finite Element Method: A powerful tool to solve incompressible flows with free surfaces and breaking waves, *Int. J. Numer. Meth. Engrg.*, **61**, 964–989, 2004.

[26] S. .R. Idelsohn, F. Del Pin, R. Rossi and E. Oñate, Fluid-structure interaction problems with strong added-mass effect, by S.R. Idelsohn, F. Del Pin, R. Rossi, E. Oate. *Int. J. Numer. Meth. Engrg.*, **10**, 1261–1294, 2009.

[27] M. Kouhi and E. Oñate, A stabilized finite element method for high-speed inviscid compressible flows using finite calculus, *Int. J. Numer. Meth. Fluids*, **12**, 872–897, 2014.

[28] J. Lubliner, *Plasticity Theory*, Macmillan Pub. Co., New York, 1990; reprinted by Dover, 2008.

[29] G. A. Maugin and A. V. Metrikine (eds.) *Mechanics of Generalized Continua*, Springer, New York, 2010.

[30] R. D. Mindlin and H. F. Tiersten, Effect of coupled-stresses in linear elasticity, *Arch. Rational Mech. Anal.*, **11**, 415–422, 1962.

[31] R. D. Mindlin, Micro-structure in linear elasticity, *Arch. Rational Mech. Anal.*, **16**, 51–78, 1964.

[32] R. D. Mindlin, Second gradient of strain and surface tension in linear elasticity, *Int. J. Solids Struc.*, **1**, 417–438, 1965.

[33] P. J. Olver, *Applications of Lie Groups to Differential Equations*, Sp[ringer, Berlin and New York, 1993.

[34] P. J. Olver, *Classical Invariant Theory*, Cambridge Univ. Press, Cambridge, U.K., 1999.

[35] E. Oñate, Derivation of the stabilization equations for advective-diffusive fluid transport and fluid flow problems. *Comp. Meths. Appl. Mech. Engrg.,* **151**, 233–267, 1998.

[36] E. Oñate, R. L. Taylor, O. C. Zienkiewicz and J. Rojek, A residual correction method based on finite calculus, *Engrg. Comput.*, **20**, 629–638, 2003.

[37] E. Oñate, Possibilities of finite calculus in computational mechanics, *Int. J. Numer. Meth. Engrg.,* **60**, 255–281, 2004.

[38] E. Oñate, J. Rojek, R. L. Taylor, and O. C Zienkiewicz, Finite calculus formulation for incompressible solids using linear triangles and tetrahedra, *Int. J. Numer. Meth. Engrg.*, **59**, 1473–1500, 2004.

[39] E. Oñate, F. Zarate, and S. R. Idelsohn, Finite element formulation for the convective-diffusive problem with sharp gradients using finite calculus, *Comp. Meths. Appl. Mech. Engrg.*, **195**, 1793–1825, 2006.

[40] E. Oñate, J. Miquel, and G. Hauke, A stabilized finite element method for the one-dimensional advection diffusion-absorption equation using finite calculus, *Comp. Meths. Appl. Mech. Engrg.*, **195**, 3926–3946, 2006.

[41] E. Oñate, A. Valls, and J. Garcia, Modeling incompressible flows at low and high Reynolds numbers via a finite calculus-finite element approach, *J. Comp. Physics*, **224**,332–351, 2007.

[42] E. Oñate, S. R. Idelsohn, M. Celigueta, and R. Rossi, Advances in the particle finite element method for the analysis of fluid-multibody interaction problems and bed erosion in free surface flows, *Comp. Meths. Appl. Mech. Engrg.*, **197**, 1777–1800, 2008.

[43] E. Oñate, S. R. Idelsohn and C. A. Felippa, Consistent pressure Laplacian stabilization for incompressible continua via higher-order finite calculus, *Int. J. Numer. Meth. Engrg.,* **87**, 171–195, 2011.

[44] E. Oñate, P. Nadukandi, S. R. Idelsohn, and C. A. Felippa, A family of residual-based stabilized finite element methods for Stokes flow, *Int. J. Numer. Meth. Engrg.*, **65**, 106–134, 2011.

[45] E. Oñate, A. Franci, and J. Carbonell, Lagrangian formulation for finite element analysis of quasi-incompressible fluids with reduced mass losses, *Int. J. Numer. Meth. Fluids*, **74**, 699-731, 2014.

[46] E. Oñate, Finite Increment Calculus (FIC): a framework for deriving enhanced computational methods in mechanics, *Advanced Modeling and Simulation in Engineering Sciences*, Springer Open Journal, 3–14, 2016.

[47] C. R. Rao and S. K. Mitra, *Generalized Inverse of Matrices and its Applications*, Wiley, New York, 1971.

[48] C. Truesdell and R. A. Toupin, The classical field theories, in: S. Flgge (ed.) *Handbuch der Physik*, vol. III/1, 226–790, Springer Verlag, Berlin, 1960.

[49] C. Truesdell and W. Noll, The nonlinear field theories of mechanics, in: S. Flgge (ed.), *Handbuch der Physik*, Vol. III/3, Springer-Verlag, 1965.

[50] R. A. Toupin, Elastic materials with coupled stresses, *Arch. Rational Mech. Anal.*, **11**, 385–414, 1962.

[51] H. W. Turnbull, *The Theory of Determinants, Matrices and Invariants*, Blackie and Sons, London, 1929. Expanded reprint by Dover, 1960.

[52] H. Weyl, *The Classical Groups, Their Invariants and Representations*, Princeton University Press, Princeton, N.J., 1939.