Left or right handed potential data?

Horst Holstein[1*], Des FitzGerald[2], Matt Zengerer[2] and Andy Starr[1] highlight the nature of the ambiguity, and suggest a unified approach to encompass any mixture of coordinate conventions.

The representation of potential vector and tensor fields by 3 × 1 and 3 × 3 matrices of Cartesian components is ambiguous unless the coordinate directions are also specified, and in particular, whether the system is left- or right handed. In this paper we highlight the nature of the ambiguity, and suggest a unified approach to encompass any mixture of coordinate conventions. Failure to observe the correct conventions can lead to incorrect interpretation of the potential data, and the suggested protocols are a step towards data integrity.

Tensors of rank one and of rank two, commonly referred to as vectors and tensors in geophysical contexts, are often recorded as data sets of 3 × 1 and 3 × 3 matrices of numerical Cartesian components. Surveys from different sources may, however, use different coordinate conventions, or the client may be unaware of the coordinate conventions assumed by the provider.

In essence, vector and tensor array data are incomplete without also being accompanied by information stating the directions and ordering of the axes of the employed coordinate system. A commonly used system makes the use of North (*N*), East (*E*) and Down (*D*) coordinate directions. But is the ordering of the vector components *END* or *NED*? The first is for a left-handed and the second for a right-handed co-ordinate system. The choice will affect the meaning of the vector and tensor component data. Moreover, standard vector operation software (in particular, cross product routines) assume right-handed reference systems, and can yield incorrect results for left-handed systems unless explicitly adjusted.

To ensure consistent handling of potential data by various application programmes, we suggest an initial protocol during which the user is asked to declare the coordinate conventions to be used for a data source, before any processing of that data is undertaken (see Figure 1). This commits the user to a specific convention, rather than allowing processing to proceed under default assumptions. After this stage, any vector or tensor data from that source are converted into a standard right-handed system for internal working. An exit protocol reverts the internal form back to the conventions of an external data form. This will ensure a consistent treatment of vector and tensor processing. The user need not be aware of the internal convention, as all data interfaces are carried out via the user's declared protocols.

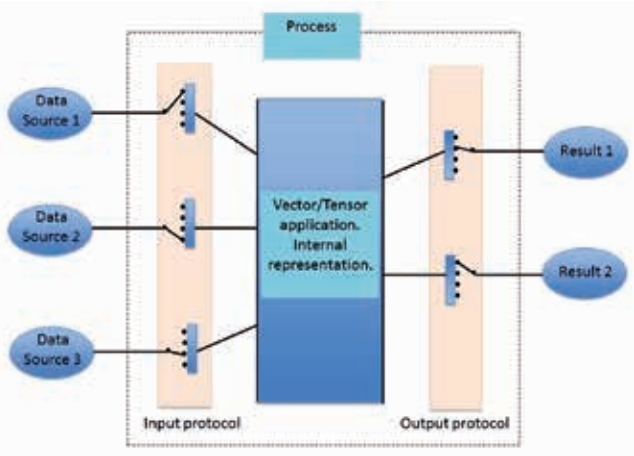

Figure 1 *Schematic diagram, for suggested integrity maintenance of vector and tensor handling. External vector and tensor data sources, on being accessed, are assigned appropriate methods for conversion into an internal form prior to processing. This ensures that data using different reference conventions are handled consistently. Results are exported through an output protocol by methods that convert the internal form to an appropriate external form.*

In the sections below, we first give an intuitive derivation of standard results in vector and tensor transformations for the case of three common Cartesian coordinate conventions. This will allow us to demonstrate the forward and inverse transformations between the external and internal standard reference systems. We then extend our derivations to include the usual matrix-based transformation formulae, and show how they may be included in the protocols. In the discussion and conclusions, we list some scenarios in which data coordinate protocols are important, and consider possible overheads from the suggested protocols.

Vector and tensor Cartesian representation

Given a potential function $\varphi\,(x_1, x_2, x_3)$ defined at points (x_1, x_2, x_3) of a Cartesian coordinate system, the components of the associated vector field **F** and tensor field **G** can be written in matrix form as

$$\mathbf{F} = \begin{bmatrix} \varphi_1 \\ \varphi_2 \\ \varphi_3 \end{bmatrix}, \mathbf{G} = \begin{bmatrix} \varphi_{11} & \varphi_{12} & \varphi_{13} \\ \varphi_{21} & \varphi_{22} & \varphi_{23} \\ \varphi_{31} & \varphi_{32} & \varphi_{33} \end{bmatrix} \quad (1)$$

[1] *Aberystwyth University.*
[2] *Intrepid Geophysics.*
[*] *Corresponding author, E-mail: hoh@aber.ac.uk*

where

$$\varphi_i = \frac{\partial \varphi}{\partial x_i}\ ,\ \varphi_{ij} = \varphi_{ji} = \frac{\partial^2 \varphi}{\partial x_i \partial x_j} \quad . \tag{2}$$

The usual condition of continuous second derivatives ensures that the tensor components φ_{ij} defined in equation (2) form a symmetric matrix representation for **G** in equation (1).

Although the notation in equation (1) reflects common usage, it is imprecise, as it does not draw attention to the directions of the coordinate axes used. An array of components can only represent a vector or tensor if the reference system's basis vectors are also indicated or implied. Below, we give such indications via suffixed arrays.

Let $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$ be three orthogonal unit vectors directed along the axes of the employed coordinate system. Compared to equation (1), a more explicit vector and tensor definition is provided via the notation

$$\mathbf{F} = \varphi_1 \mathbf{x}_1 + \varphi_2 \mathbf{x}_2 + \varphi_3 \mathbf{x}_3\ , \tag{3}$$

$$\begin{aligned} \mathbf{G} = \quad & \varphi_{11}\mathbf{x}_1\mathbf{x}_1 + \varphi_{12}\mathbf{x}_1\mathbf{x}_2 + \varphi_{13}\mathbf{x}_1\mathbf{x}_3 \\ & + \varphi_{21}\mathbf{x}_2\mathbf{x}_1 + \varphi_{22}\mathbf{x}_2\mathbf{x}_2 + \varphi_{23}\mathbf{x}_2\mathbf{x}_3 \\ & + \varphi_{31}\mathbf{x}_3\mathbf{x}_1 + \varphi_{32}\mathbf{x}_3\mathbf{x}_2 + \varphi_{33}\mathbf{x}_3\mathbf{x}_3 \quad . \end{aligned} \tag{4}$$

Equation (3) simply expresses the vector **F** as sum of unit vectors weighted by their vector components in equation (1), while equation (2) expresses the tensor **G** as a weighted sum of elementary tensors formed from the ordered pairs $(\mathbf{x}_i \mathbf{x}_j)$. These ordered pairs are seen to be the component place holders in the matrix schema of equation (1). Such 'dyads' are discussed in Pujo (2003) and Weatherburn (1960).

In contrast to equation (1), equations (3) and (4) show the explicit dependence on the co-ordinate axis vectors (or basis vectors) $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$. Their explicit presence allows any component in any reference system to be derived. For example, the components F_p, G_{pq} along arbitrary unit directions **p** and **q**, are given via equations (3) and (4) by

$$F_p = \mathbf{p}\cdot\mathbf{F} = \varphi_1(\mathbf{p}\cdot\mathbf{x}_1) + \varphi_2(\mathbf{p}\cdot\mathbf{x}_2) + \varphi_3(\mathbf{p}\cdot\mathbf{x}_3)\ , \tag{5}$$

$$\begin{aligned} G_{pq} &= \mathbf{p}\cdot\mathbf{G}\cdot\mathbf{q} \\ &= \mathbf{p}\cdot(\varphi_{11}\mathbf{x}_1\mathbf{x}_1 + \varphi_{12}\mathbf{x}_1\mathbf{x}_2 + \cdots)\cdot\mathbf{q} \\ &= \varphi_{11}(\mathbf{p}\cdot\mathbf{x}_1)(\mathbf{x}_1\cdot\mathbf{q}) + \varphi_{12}(\mathbf{p}\cdot\mathbf{x}_1)(\mathbf{x}_2\cdot\mathbf{q}) + \ldots \end{aligned} \tag{6}$$

Orthonormality of the unit basis vectors, namely

$$\mathbf{x}_i\cdot\mathbf{x}_j = \delta_{ij} = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ if } i \neq j \end{cases} \qquad i,j \in 1,2,3\ , \tag{7}$$

leads to the expected component results in the home basis. For example, with $\mathbf{p} = \mathbf{x}_1$, $\mathbf{q} = \mathbf{x}_2$, equations (5) and(6) give $\mathbf{x}_1\cdot\mathbf{F} = \varphi_1$ and $\mathbf{x}_1\cdot\mathbf{G}\cdot\mathbf{x}_2 = \varphi_{12}$ respectively.

We next use the basis representation of vectors and tensors to explore the relationships between three commonly used coordinate systems.

Comparison of representations in the *NED, END* and *ENU* systems

Consider three commonly used co-ordinate systems, with basis directions enumerated as a) *NED*, b) *END* and c) *ENU*. Let the unit vectors along the axes of these three systems be denoted by **N**, **E**, **D** and **U**. Since vectors **D** and **U** are oppositely directed, we have **D** = –**U**.

Let the components of a vector **F** along unit vector directions **N**, **E**, **D** and **U** be denoted by

$$\varphi_1 = \mathbf{N}\cdot\mathbf{F},\ \varphi_2 = \mathbf{E}\cdot\mathbf{F}, \varphi_3 = \mathbf{D}\cdot\mathbf{F}, -\varphi_3 = \mathbf{U}\cdot\mathbf{F}\ . \tag{8}$$

Expressions for the same vector **F** in the three systems are given by

$$\mathbf{F} = \varphi_1\mathbf{N} + \varphi_2\mathbf{E} + \varphi_3\mathbf{D} \qquad (NED, END) \tag{9}$$

$$= \varphi_2\mathbf{E} + \varphi_1\mathbf{N} + (-\varphi_3)\mathbf{U} \qquad (ENU) \tag{10}$$

Equation (9) holds for both *NED* and *END* systems. In the absence of explicit basis cues, the standard notation indicates the implied basis vectors via ordering of entries in the component arrays, thus

$$\mathbf{F} = \begin{bmatrix} \varphi_1 \\ \varphi_2 \\ \varphi_3 \end{bmatrix}_{NED} = \begin{bmatrix} \varphi_2 \\ \varphi_1 \\ \varphi_3 \end{bmatrix}_{END} = \begin{bmatrix} \varphi_2 \\ \varphi_1 \\ -\varphi_3 \end{bmatrix}_{ENU} \quad . \tag{11}$$

When a survey with multiple source data conventions is to be processed, and vectors are represented as component arrays, there is clearly scope for error by assuming unstated basis ordering conventions. The forms (9) and (10), on the other hand, make the assignment of components to their basis vectors visible, and draw attention to the assumptions made.

Particular care must be exercised when employing the vector cross (×) product operation. The same numerical operands can lead to different numerical results, as illustrated by

$$\underbrace{\begin{bmatrix}1\\0\\0\end{bmatrix}\times\begin{bmatrix}0\\1\\0\end{bmatrix} = \begin{bmatrix}0\\0\\1\end{bmatrix}}_{NED,\,ENU\,(right-handed)}\ ,\ \underbrace{\begin{bmatrix}1\\0\\0\end{bmatrix}\times\begin{bmatrix}0\\1\\0\end{bmatrix} = \begin{bmatrix}0\\0\\-1\end{bmatrix}}_{END\,(left-handed)}\ . \tag{12}$$

The meaning of the first product in the *NED* system is **N** × **E** = **D**, and in the *ENU* system its meaning is **E** × **N** = **U**, while the meaning of the second product in the *END* system is **E** × **N** = –**D**, all of which are correct. The apparent discrepancy in the numerical result is a consequence of the implicit ordering of the vector components.

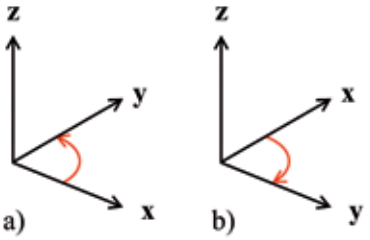

Figure 2 *Two reference systems with unit axis vectors shown. Following the right-hand rule for vector products as indicated by the curved arrow lines, the axis vectors satisfy a)* ***X*** *×* ***Y*** *=****Z*** *and b)* ***X*** *×* ***Y*** *=-****Z****. This makes system a) right handed and system b) left handed.*

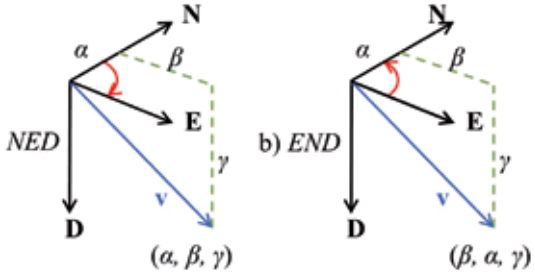

Figure 3 *Two reference systems, employing the same set of unit axis vectors, but enumerated as a)* ***N,E,D*** *and b)* ***E,N,D****. Properties* ***N*** *×* ***E*** *=* ***D****,* ***E*** *×* ***N*** *= -****D*** *indicate that system a) is right handed and system b) is left handed. In both systems, the displacement vector* $\alpha\mathbf{N} + \beta\mathbf{E} + \gamma\mathbf{D}$ *denotes the same position vector* ***v****, but its components are recorded as* $[\alpha, \beta, \gamma]$ *and* $[\beta, \alpha, \gamma]$ *in systems a) and b) respectively.*

The *NED* and *ENU* systems are said to be 'right-handed', their hallmark being that the cross product of any two basis vectors in a left to right cyclic sequence yields the third. Thus, for the *NED* system we have **N**×**E**=**D**, **E**×**D**=**N**, **D**×**N**=**E**, and for the *ENU* system we have **E** × **N** = **U**, and its cyclic permutations. In the case of the *END* system we have: **E**×**N** = –**D**, **N**×**D** = –**E**, **D**×**E** = –**N** (see Figures 2 and 3). The systematic sign reversal is a property of left-handed systems. Standard vector routines assume a right-handed reference system. Equation (12) shows that the standard result must be adjusted when performing vector cross product operations in left-handed systems.

For a tensor **G** with components in the *NED* system, we write

$$\begin{aligned}\mathbf{G} = \;& \varphi_{11}\mathbf{NN} + \varphi_{12}\mathbf{NE} + \varphi_{13}\mathbf{ND} \\ & + \varphi_{21}\mathbf{EN} + \varphi_{22}\mathbf{EE} + \varphi_{23}\mathbf{ED} \\ & + \varphi_{31}\mathbf{DN} + \varphi_{32}\mathbf{DE} + \varphi_{33}\mathbf{DD}\end{aligned} \quad . \tag{13}$$

We trivially rearrange the summed terms to reflect the orderings that would be used for the same tensor expressed in the *END* system,

$$\begin{aligned}\mathbf{G} = \;& \varphi_{22}\mathbf{EE} + \varphi_{21}\mathbf{EN} + \varphi_{23}\mathbf{ED} \\ & + \varphi_{12}\mathbf{NE} + \varphi_{11}\mathbf{NN} + \varphi_{13}\mathbf{ND} \\ & + \varphi_{32}\mathbf{DE} + \varphi_{31}\mathbf{DN} + \varphi_{33}\mathbf{DD}\end{aligned} \tag{14}$$

and, with **U** = –**D**, in the *ENU* system,

$$\begin{aligned}\mathbf{G} = \;& \varphi_{22}\mathbf{EE} + \varphi_{21}\mathbf{EN} - \varphi_{23}\mathbf{EU} \\ & + \varphi_{12}\mathbf{NE} + \varphi_{11}\mathbf{NN} - \varphi_{13}\mathbf{NU} \\ & - \varphi_{32}\mathbf{UE} - \varphi_{31}\mathbf{UN} + \varphi_{33}\mathbf{UU}\end{aligned} \quad . \tag{15}$$

From these rearrangements, we obtain the equivalent implicit representations of the tensor **G**,

$$\mathbf{G} = \underbrace{\begin{bmatrix}\varphi_{11} & \varphi_{12} & \varphi_{13} \\ \varphi_{21} & \varphi_{22} & \varphi_{23} \\ \varphi_{31} & \varphi_{32} & \varphi_{33}\end{bmatrix}}_{NED} = \underbrace{\begin{bmatrix}\varphi_{22} & \varphi_{21} & \varphi_{23} \\ \varphi_{12} & \varphi_{11} & \varphi_{13} \\ \varphi_{32} & \varphi_{31} & \varphi_{33}\end{bmatrix}}_{END} = \underbrace{\begin{bmatrix}\varphi_{22} & \varphi_{21} & -\varphi_{23} \\ \varphi_{12} & \varphi_{11} & -\varphi_{13} \\ -\varphi_{32} & -\varphi_{31} & \varphi_{33}\end{bmatrix}}_{ENU} \quad . \tag{16}$$

Once again, it is evident that tensor operations (e.g., differencing) will map to the corresponding matrix operations only if the matrix operands express tensors in the same reference system. In the next section we suggest a method of achieving such referential integrity.

Achieving a consistent internal representation

In order to achieve consistent vector and tensor processing, whatever the source data conventions, we propose to use an internally defined right-handed co-ordinate system into which all source data are mapped. The mapping is to occur during explicit initialisation, requiring the user to declare the co-ordinate system conventions of each external data source. Any result or display channel will invoke the inverse transformation, to convert the internal representation to that chosen by the user for the output channel. The user need not be aware of the specific internal representation – results are always displayed or output in a convention chosen by the user.

With this aim, let us define an internal right-handed reference system having unit vectors $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$ with an internal column matrix representation

$$\mathbf{e}_1 = \begin{bmatrix}1\\0\\0\end{bmatrix}, \mathbf{e}_2 = \begin{bmatrix}0\\1\\0\end{bmatrix}, \mathbf{e}_3 = \begin{bmatrix}0\\0\\1\end{bmatrix} \quad . \tag{17}$$

Knowing that the $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$ is a right-handed vector set, we can make the mapping

$$\mathbf{E} = \mathbf{e}_1 \;, \mathbf{N} = \mathbf{e}_2, \mathbf{D} = -\mathbf{e}_3 \;, \mathbf{U} = \mathbf{e}_3 \quad . \tag{18}$$

External input vectors, expressed as coordinate triples in possibly mixed conventions such as

$$\mathbf{a} = \begin{bmatrix}2\\-4\\-1\end{bmatrix}_{NED}, \mathbf{b} = \begin{bmatrix}1\\-3\\10\end{bmatrix}_{END}, \mathbf{c} = \begin{bmatrix}2\\0\\5\end{bmatrix}_{ENU} \quad , \tag{19}$$

will be redefined during the input protocol according to vector additions

$$\mathbf{a} = (2)\mathbf{N}+(-4)\mathbf{E}+(-1)\mathbf{D}\ , \tag{20}$$

$$\mathbf{b} = (1)\mathbf{E}+(-3)\mathbf{N}+(10)\mathbf{D}\ , \tag{21}$$

$$\mathbf{c} = (2)\mathbf{E}+(0)\mathbf{N}+(5)\mathbf{U}\ . \tag{22}$$

and under relations (17) and (18) lead to internal array representations (here suffixed by *int*)

$$\mathbf{a}=\begin{bmatrix}-4\\2\\1\end{bmatrix}_{int},\mathbf{b}=\begin{bmatrix}1\\-3\\-10\end{bmatrix}_{int},\mathbf{c}=\begin{bmatrix}2\\0\\5\end{bmatrix}_{int} \tag{23}$$

These allow the usual vector operations to be correctly executed, for example

$$\mathbf{a}-\mathbf{b}=\begin{bmatrix}-5\\1\\11\end{bmatrix}_{int},\mathbf{a}\cdot(\mathbf{b}\times\mathbf{c})=\det\begin{bmatrix}-4&1&2\\2&-3&0\\1&-10&5\end{bmatrix}_{int}=16\ . \tag{24}$$

To output the first result in the *NED* convention, for example, we compute

$$\mathbf{a}-\mathbf{b}=\begin{bmatrix}\mathbf{N}\cdot(\mathbf{a}-\mathbf{b})\\\mathbf{E}\cdot(\mathbf{a}-\mathbf{b})\\\mathbf{D}\cdot(\mathbf{a}-\mathbf{b})\end{bmatrix}_{NED}=\begin{bmatrix}1\\-5\\-11\end{bmatrix}_{NED}\ . \tag{25}$$

The result for a • (b × c), being scalar, is a coordinate system invariant. Conversion of tensors to and from internal form is similarly carried out, by appealing to external basis vectors expressed in the internal matrix representation. Moreover, elementary tensors $\mathbf{e}_i\mathbf{e}_j$ have a component matrix represented directly as the matrix product $\mathbf{e}_i\mathbf{e}_j^T$. The tensor **G**, formed originally from the *NED* components in equation (13), has an internal representation for which the component matrix is

$$\begin{aligned}[\mathbf{G}]=\ &\varphi_{11}\mathbf{e}_2\mathbf{e}_2^T+\varphi_{12}\mathbf{e}_2\mathbf{e}_1^T+\varphi_{13}\mathbf{e}_2(-\mathbf{e}_3)^T\\&+\varphi_{21}\mathbf{e}_1\mathbf{e}_2^T+\varphi_{22}\mathbf{e}_1\mathbf{e}_1^T+\varphi_{23}\mathbf{e}_1(-\mathbf{e}_3)^T\\&+\varphi_{31}(-\mathbf{e}_3)\mathbf{e}_2^T+\varphi_{32}(-\mathbf{e}_3)\mathbf{e}_1^T+\varphi_{33}(-\mathbf{e}_3)(-\mathbf{e}_3)^T\end{aligned}\ . \tag{26}$$

Terms $\mathbf{e}_i\mathbf{e}_j^T$ are matrices with a single unit element in position *ij*, for example

$$\mathbf{e}_2\mathbf{e}_3^T=\begin{bmatrix}0\\1\\0\end{bmatrix}\begin{bmatrix}0&0&1\end{bmatrix}=\begin{bmatrix}0&0&0\\0&0&1\\0&0&0\end{bmatrix}\ . \tag{27}$$

Direct evaluation of the matrix expression (26) then leads to the internal representation

$$\mathbf{G}=\begin{bmatrix}\varphi_{22}&\varphi_{21}&-\varphi_{23}\\\varphi_{12}&\varphi_{11}&-\varphi_{13}\\-\varphi_{32}&-\varphi_{31}&\varphi_{33}\end{bmatrix}_{int}\ . \tag{28}$$

To recover an external description of the tensor from the internal representation, we take the bi-projections of the tensor on the appropriate component directions, as described in equation (6). Thus, to find component 13 of tensor **G** in the *END* system, we calculate **E** • **G** • **D.** The dot products themselves are executable as row times column matrix operations, and this leads to the matrix expression

$$G_{13}=\mathbf{E}^T\mathbf{G}\mathbf{D}=\begin{bmatrix}1&0&0\end{bmatrix}\begin{bmatrix}\varphi_{22}&\varphi_{21}&-\varphi_{23}\\\varphi_{12}&\varphi_{11}&-\varphi_{13}\\-\varphi_{32}&-\varphi_{31}&\varphi_{33}\end{bmatrix}\begin{bmatrix}0\\0\\-1\end{bmatrix}=\varphi_{23} \tag{29}$$

where **E, G, D** are all expressed in the internal matrix form. The result is consistent with the *END* component matrix representation in equation (16).

The handling of *NED*, *END* and *ENU* data sources is a common practical requirement, but is rather special on account of axis reuse in these systems. Mappings between non-aligned data sources and the internal form follow readily from the above, and is summarised in the next section.

Matrix formulae for vector and tensor component transformation

Vector and tensor data in arbitrary reference systems can be handled similarly to the treatment above, by transforming the basis vectors into a standard internal form. This leads to the standard component transformations by matrix methods. Let an arbitrary external reference system have unit orthogonal basis vectors $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$. They are related to the internal reference vectors $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$ via the vector identities

$$\begin{aligned}\mathbf{e}_1&=\mathbf{x}_1(\mathbf{x}_1\cdot\mathbf{e}_1)+\mathbf{x}_2(\mathbf{x}_2\cdot\mathbf{e}_1)+\mathbf{x}_3(\mathbf{x}_3\cdot\mathbf{e}_1)\\\mathbf{e}_2&=\mathbf{x}_1(\mathbf{x}_1\cdot\mathbf{e}_2)+\mathbf{x}_2(\mathbf{x}_2\cdot\mathbf{e}_2)+\mathbf{x}_3(\mathbf{x}_3\cdot\mathbf{e}_2)\\\mathbf{e}_3&=\mathbf{x}_1(\mathbf{x}_1\cdot\mathbf{e}_3)+\mathbf{x}_2(\mathbf{x}_2\cdot\mathbf{e}_3)+\mathbf{x}_3(\mathbf{x}_3\cdot\mathbf{e}_3)\ .\end{aligned} \tag{30}$$

The coefficients form a 3 × 3 matrix **R**, where

$$\mathbf{R}=\begin{bmatrix}(\mathbf{x}_1\cdot\mathbf{e}_1)&(\mathbf{x}_1\cdot\mathbf{e}_2)&(\mathbf{x}_1\cdot\mathbf{e}_3)\\(\mathbf{x}_2\cdot\mathbf{e}_1)&(\mathbf{x}_2\cdot\mathbf{e}_2)&(\mathbf{x}_2\cdot\mathbf{e}_3)\\(\mathbf{x}_3\cdot\mathbf{e}_1)&(\mathbf{x}_3\cdot\mathbf{e}_2)&(\mathbf{x}_3\cdot\mathbf{e}_3)\end{bmatrix} \tag{31}$$

(see Figure 4). This matrix allows us to express the basis transformation (30) as

$$\begin{bmatrix}\mathbf{e}_1&\mathbf{e}_2&\mathbf{e}_3\end{bmatrix}=\begin{bmatrix}\mathbf{x}_1&\mathbf{x}_2&\mathbf{x}_3\end{bmatrix}\mathbf{R}\ . \tag{32}$$

The inverse relationship makes use of equations similar to (30) but with roles of symbols **e** and **x** reversed. This reversal

leads to the transpose of matrix **R**. The inverse relation is therefore

$$\begin{bmatrix}\mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3\end{bmatrix} = \begin{bmatrix}\mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3\end{bmatrix}\mathbf{R}^T \ . \tag{33}$$

Applying this change of basis to equation (3), we obtain the vector transformation formula

$$\mathbf{F} = \begin{bmatrix}\mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3\end{bmatrix}\begin{bmatrix}\varphi_1 \\ \varphi_2 \\ \varphi_3\end{bmatrix} = \begin{bmatrix}\mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3\end{bmatrix}\left(\mathbf{R}^T\begin{bmatrix}\varphi_1 \\ \varphi_2 \\ \varphi_3\end{bmatrix}\right) . \tag{34}$$

The transformation of tensor components in equation (4) is achieved in a similar way. We first reorganise the equation (4) into bi-quadratic form over the basis vector objects

$$\mathbf{G} = \begin{bmatrix}\mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3\end{bmatrix}\begin{bmatrix}\varphi_{11} & \varphi_{12} & \varphi_{13} \\ \varphi_{21} & \varphi_{22} & \varphi_{23} \\ \varphi_{31} & \varphi_{32} & \varphi_{33}\end{bmatrix}\begin{bmatrix}\mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3\end{bmatrix} \tag{35}$$

Substituting from equation (33) for the pre-multiplying row expression, and its transpose for the post-multiplying column expression, we obtain the tensor transformation formula

$$\mathbf{G} = \begin{bmatrix}\mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3\end{bmatrix}\left(\boldsymbol{R}^T\begin{bmatrix}\varphi_{11} & \varphi_{12} & \varphi_{13} \\ \varphi_{21} & \varphi_{22} & \varphi_{23} \\ \varphi_{31} & \varphi_{32} & \varphi_{33}\end{bmatrix}\boldsymbol{R}\right)\begin{bmatrix}\mathbf{e}_1 \\ \mathbf{e}_2 \\ \mathbf{e}_3\end{bmatrix} . \tag{36}$$

Equation (34) expresses the same vector **F** under different basis representations, and equations (35), (36) similarly express the same tensor **G**. The grouping of matrix operations in round brackets therefore represents the matrix operations to be carried out on the component arrays when there is a change of basis.

Arithmetic complexity issues

Our process model treats vectors and tensors as objects of an abstract class, with an internal representation used for calculation, but whose internal details are not of direct concern to the user. The class methods allow conversion from any user-defined reference system to the internal form, and vice versa. Usually the external view of a vector is an array of three numbers for the component values, together with information to identify the external reference system. For tensors, the array contains 9 numbers, but symmetry compaction may reduce this to a list of 6 numbers, further reduced to 5 if a zero trace condition is called upon.

Component conversion of vectors and tensors can always be carried out via matrix calculations described in equation (34) and equations (35), (36). The computational cost would be three inner products (row times column operations) for a vector transformation, and 18 inner products for a tensor transformation, although symmetry and trace conditions can

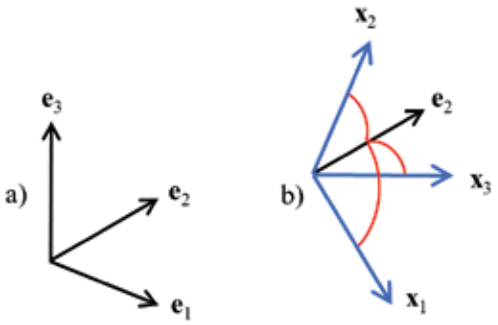

Figure 4 *The general case of two reference systems with non-aligned unit axes a) $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$, and a) $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$. The components of a vector or a tensor in the two systems are related by a 3 × 3 transformation matrix **R**, whose 9 elements $\mathbf{e}_i \bullet \mathbf{x}_j$ are the cosines of the angles each axis of one system makes with all the axes of the other. The angles made by vector $\mathbf{e}_2$ of system a) with each of the axes in system b) are shown in the diagram.*

reduce this count to 14 inner products. If the external system is right-handed and maps directly into the internal system, no arithmetic operations are involved. A left-handed system may require only one sign flip for vectors and two for symmetric tensors (compare the *END* and *ENU* systems in equation (16)) for external to internal conversion. This shows that the conversion methods may allow tuning to lower the computational complexity burden.

The formalism in equations (9), (10) and (13)-(15), that expresses the external-internal conversion of vectors and tensors as sums over basis vector expressions (e.g., via equations (16) and (17), may be highly non-optimal, particularly if patterns of zeros and ones in the internal basis representations are disregarded. When applied to high volumes of data, the methods should be considered for optimization. In some cases, a few component transpositions and sign flips may suffice to produce the internal forms. In general, though, the conversion overhead grows only linearly with data size. The advantage of the theoretical formalism is that it is highly intuitive and open to verification.

Consider, for example, expressing the magnetic field as a vector **H** when it is observed with total intensity H nT at declination δ and inclination ι degrees (see Blakely, 1995). In terms of the local *NED* reference frame, the magnetic field vector **H** can be written as

$$\mathbf{H} = H\left(\left(\mathbf{N}\cos\delta + \mathbf{E}\sin\delta\right)\cos\iota + \mathbf{D}\sin\iota\right) . \tag{37}$$

To manipulate such data over a range of latitude and longitude locations in a common reference frame, an Earth-centred system may be desirable. Employing internal right-handed unit basis vectors $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$, directed from the Earth's centre to (latitude, longitude) surface points at (0,0), (0,90), (90,-) degrees respectively, we may express the local basis vectors as functions of longitude φ and latitude λ by

$$\mathbf{E} = -\mathbf{e}_1\sin\phi + \mathbf{e}_2\cos\phi \ , \tag{38}$$

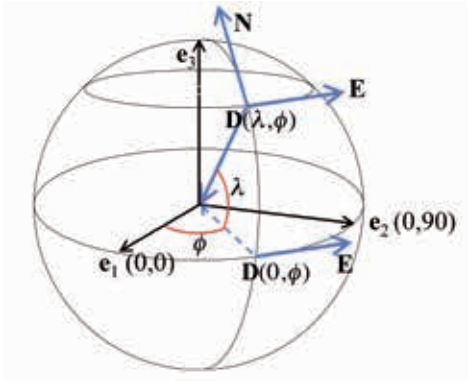

Figure 5 *Local **N**, **E**, **D** vectors at latitude λ and longitude ϕ, and an earth-centred reference system. Unit vectors $\mathbf{e}_1$, $\mathbf{e}_2$ are directed from the centre to points of longitude $\phi = 0$ and $\phi = 90$ degrees on the equatorial plane, and $\mathbf{e}_3 = \mathbf{e}_1 \times \mathbf{e}_2$. The **E** vector is constant along any meridian ϕ, and so equal to its equatorial value, where it is found by circular interpolation between $\mathbf{e}_2$ (its value at $\phi = 0$) and $-\mathbf{e}_1$ (its value at $\phi = 90$). The equatorial value of **D** at longitude ϕ can similarly be found, by circular interpolation between $-\mathbf{e}_1$ and $-\mathbf{e}_2$. Circular interpolation between this equatorial value and the polar value then gives the value of **D** for latitude λ. The third direction is found from $\mathbf{N} = \mathbf{E} \times \mathbf{D}$. These operations are expressed in equations (38)-(40).*

$$\mathbf{D} = -(\mathbf{e}_1 \cos\phi + \mathbf{e}_2 \sin\phi)\cos\lambda - \mathbf{e}_3 \sin\lambda \ , \qquad (39)$$

$$\mathbf{N} = \mathbf{E} \times \mathbf{D} \ . \qquad (40)$$

(See Figure 5). Practical calculation can now proceed using standard vector algebra routines, with the internal vectors mapped on to component arrays as given in equation (17). This will yield the numerical components of the magnetic field in the Earth-centred system, without explicit transformation matrices having been calculated or invoked.

Conclusions

Geomagnetic and gravity surveys may record vector and tensor fields components in any reference system preferred by the observer. The processing of such data, e.g., for filtering, gridding and geophysical model matching purposes, requires a consensus between the data reference systems and the assumed processing software reference system. By itself, specification of ordered Cartesian tuples (v_x, v_y, v_z) for vector components and $(T_{xx}, T_{xy}, \ldots T_{zz})$ for tensor components is insufficient, as the underlying (x, y, z) reference system may be right or left handed. The correct choice is essential for deriving the appropriate data semantics and for correct processing.

In response to the title's question, 'left or right handed potential data?' we state that both give legitimate ways of representing vectors and tensors. The onus is therefore on the processing software to provide the necessary flexibility of handling either kind of input data, and if need be, a mixture. We have suggested a protocol that treats vectors and tensors as objects, whose classes have conversion methods between the external representations and the internal one, and vice versa. The conversion methods specify how the external reference basis vectors are to map on to an internal set. The internal set is chosen to conform to a right-handed reference system. This ensures that internally applied standard vector and tensor software will give correct results.

Our formulation of the conversion methods is intuitive. It is stated in terms of vector operations, at a level where the internal structure of the basis vectors does not have to be revealed. The approach encourages high-level verification, an advantage over the usual implementation with component transformation matrices. However, we have shown the approaches are compatible. Indeed, a naive statement of a conversion method might optimize, in some cases towards component or sign flips, or even an identity operation, or tend towards full transformation matrices. It is hoped that awareness of the issues raised here will contribute towards improved reliability in vector and tensor geo-software to maintain integrity of potential data.

References

Blakeley, R.J. [1995] *Potential Theory in Gravity and Magnetic Applications*, Cambridge University Press.

Pujo, J. [2003] *Elastic Wave Propagation and Generation in Seismology*, Cambridge University Press.

Weatherburn, C.E. [1960] *Advanced Vector Analysis with Application to Mathematical Physics*. G Bell & Sons Ltd.