Geophysical Journal International

Geophys. J. Int. (2016) **207,** 414–421
Advance Access publication 2016 July 27
GJI Seismology

doi: 10.1093/gji/ggw273

Performance of continuous mass-lumped tetrahedral elements for elastic wave propagation with and without global assembly

W.A. Mulder[1,2] and R. Shamasundar[2]

[1]*Shell Global Solutions International B.V., Rijswijk, The Netherlands. E-mail: wim.mulder@shell.com*
[2]*Delft University of Technology, Delft, The Netherlands*

Accepted 2016 July 20. Received 2016 July 19; in original form 2016 May 4

SUMMARY

We consider isotropic elastic wave propagation with continuous mass-lumped finite elements on tetrahedra with explicit time stepping. These elements require higher-order polynomials in their interior to preserve accuracy after mass lumping and are only known up to degree 3. Global assembly of the symmetric stiffness matrix is a natural approach but requires large memory. Local assembly on the fly, in the form of matrix-vector products per element at each time step, has a much smaller memory footprint. With dedicated expressions for local assembly, our code ran about 1.3 times faster for degree 2 and 1.9 times for degree 3 on a simple homogeneous test problem, using 24 cores. This is similar to the acoustic case. For a more realistic problem, the gain in efficiency was a factor 2.5 for degree 2 and 3 for degree 3. For the lowest degree, the linear element, the expressions for both the global and local assembly can be further simplified. In that case, global assembly is more efficient than local assembly. Among the three degrees, the element of degree 3 is the most efficient in terms of accuracy at a given cost.

Key words: Numerical modelling; Computational seismology; Wave propagation.

1 INTRODUCTION

Finite-difference modelling of seismic wave propagation has become the workhorse of the industry for imaging hydrocarbon reservoirs. The spectral finite-element method plays a similar rôle in seismology. Higher-order finite-difference methods have problems with sharp material contrasts and topography, because they assume differentiability where it does not hold. Modifications can alleviate the decrease in accuracy, but at a cost in terms of complexity and compute time. Finite-element methods have an inherently larger computational cost, but do not suffer from a loss of accuracy if the mesh follows the interfaces between different materials and the topography. Because of their better accuracy, they may outperform the finite-difference method in some cases (e.g. Mulder 1996; Wang *et al.* 2010; Moczo *et al.* 2011; Zhebel *et al.* 2014). However, mesh generation can sometimes be difficult.

Spectral finite elements (Orszag 1980; Patera 1984; Maday & Rønquist 1990; Seriani *et al.* 1992; Komatitsch & Tromp 1999) require hexahedral meshes. Tetrahedral elements offer more flexibility in gridding, for instance, near pinch-outs. Suitable schemes are discontinuous Galerkin (DG) methods (e.g. Rivière & Wheeler 2003; Dumbser & Käser 2006; Käser & Dumbser 2006; Etienne *et al.* 2010; Wilcox *et al.* 2010), rectangular spectral elements mapped to triangles or tetrahedra (Sherwin & Karniadakis 1995; Mercerat *et al.* 2006), hybridized versions (Cockburn *et al.* 2009; Giorgiani *et al.* 2013), finite-volume methods (Dumbser *et al.* 2007; Brossier *et al.* 2008), mixed methods (Bécache *et al.* 2002; Cohen & Fauqueux 2005) or continuous mass-lumped finite elements, which we will consider here. DG methods offer the advantage that they can mix orders and types of elements on, for instance, hexahedra, tetrahedra and prisms, and also can work on non-conforming meshes. However, the fluxes required to impose continuity increase the computational cost. Since the mass matrix is block diagonal, its inversion is not costly.

Continuous mass-lumped triangular or tetrahedral finite elements avoid the cost of inverting a large sparse mass matrix by lumping the mass matrix into a diagonal one. Fried & Malkus (1975) noted, however, that with quadratic 2-D triangular elements, the lumping decreases the order of accuracy. They considered the heat equation, but the same holds for the acoustic and elastic wave equations in second-order form. Augmenting the element with polynomials of a higher degree in the interior can repair this deficiency (Fried & Malkus 1975). For the element of degree 2 in 2-D, a bubble function that vanishes on the edges suffices. Tordjman (1995) and Cohen *et al.* (1995) used this idea to construct a 2-D element of degree 3 on the edges and a bubble function times a polynomial of degree 1 in the interior, leading to an interior degree of 4. Cohen *et al.* (2001) provide error estimates. Mulder (1996) found an element of degree 4 and interior degree 5. Chin-Joe-Kong *et al.* (1999) found several elements of degree 5. The highest degree for mass-lumped triangular elements known so far is 6 (Mulder 2013).

Mulder (1996) made the generalization to tetrahedral elements with an element of degree 2 on the edges, 4 on the faces as product of a cubic bubble function and a polynomial of degree 1, and degree 4 in the interior as a product of a quartic bubble function and a constant polynomial. Lesage *et al.* (2010) and

Zhebel *et al.* (2011) applied that element to acoustic wave propagation modelling. Chin-Joe-Kong *et al.* (1999) constructed two elements of degree 3. The second one allows for a larger time step than the first (Zhebel *et al.* 2011, 2014) and will be used in the current paper. Elements of higher degree have not been found so far. Mulder *et al.* (2014) list stability estimates for the known tetrahedral lumped elements of degrees 1 to 3 as well as for the symmetric interior-penalty discontinuous Galerkin method up to degree 4.

Bao *et al.* (1998) worked with the classic linear tetrahedral mass-lumped elements for elastic wave propagation modelling. Here, we will also include elements of degrees 2 and 3.

With explicit time stepping, we can consider two approaches for assembling the stiffness and diagonal, lumped mass matrix: global assembly or local assembly on the fly. Global assembly is a standard approach with finite elements. The elements of the lumped mass matrix or its inverse can be represented by one value per node. For the symmetric global stiffness matrix, we store the symmetric block diagonal and the block upper triangular part separately, the latter in Block Compressed Sparse Row format. With local assembly on the fly, the contribution of each element to the solution update is treated independently. The displacement components on the nodes of one element are copied from a global vector and multiplied by pre-computed stiffness matrices on the reference element, nine in total. The results are then combined by geometrical factors that handle the map from the reference element to the actual element, multiplied by the inverse mass matrix, and used to increment the global solution vector for the new time level.

One might expect global assembly to produce results quicker than local assembly, at the expense of considerably larger storage, but as it turns out, this does not appear to be the case for the acoustic wave equation. The main question we address here is if a similar result also holds in the elastic case. To obtain performance figures within the same order of magnitude, we derive dedicated expressions for the matrix-vector multiplications that are part of the local assembly on the fly.

In Section 2, we describe the discretization and provide expressions for global assembly and local assembly for the general case. Simpler expressions are provided for linear elements. Section 3 presents results for global and local assembly on 24 cores. We start with the linear element. Then, we briefly consider the acoustic case, where local assembly outperforms global assembly for degree 3, before turning towards degrees 2 and 3 for the isotropic elastic case. The section ends with a slightly more realistic example. Section 4 summarizes the main conclusions.

2 METHOD

2.1 Discretization

The elastic system of wave equations for an isotropic medium in second-order form is

$$\rho \frac{\partial^2 u_m}{\partial t^2} = \sum_{j=1}^{3} \left[\frac{\partial}{\partial x_m} \left(\lambda \frac{\partial u_j}{\partial x_j} \right) + \frac{\partial}{\partial x_j} \left\{ \mu \left(\frac{\partial u_m}{\partial x_j} + \frac{\partial u_j}{\partial x_m} \right) \right\} \right] + s_m .$$

The displacement in coordinate direction x_m, $m = 1, 2, 3$, is $u_m(t, \mathbf{x})$ as a function of time, t, and position, $\mathbf{x}$. The material properties are density $\rho(\mathbf{x})$ and Lamé parameters $\mu(\mathbf{x}) = \rho v_s^2$ and $\lambda(\mathbf{x}) = \rho v_p^2 - 2\mu$, with P-wave velocity $v_p(\mathbf{x})$ and S-wave velocity $v_s(\mathbf{x})$. The forcing source function is typically of the form $s_m(t, \mathbf{x}) = f_m w(t) \delta(\mathbf{x} - \mathbf{x}_s)$, with wavelet $w(t)$ and force amplitude f_m at a source position $\mathbf{x}_s$. The domain consists of a subset of the Earth, bounded by a free surface. In exploration geophysics, absorbing boundaries are usually implemented on the sides where the domain is truncated.

The domain is meshed by tetrahedra, preferably such that the element size scales with the shear velocity, v_s (Kononov *et al.* 2012; Mulder *et al.* 2014). As wavelength scales with velocity, this provides a more or less uniform resolution over the entire mesh.

Here, the material parameters are assumed to be constant per element.

Next, we define the geometrical components (e.g. Zienkiewicz & Taylor 2000, chapter 9). Let the four vertices of the tetrahedron be denoted by $\mathbf{x}_k$, $k = 0, 1, 2, 3$. In terms of reference element, $\mathbf{x} = \sum_{k=0}^{3} \mathbf{x}_k \phi_k(\mathbf{x})$ with the basis functions, ϕ_k, of the linear element. The natural coordinates on the tetrahedron are $\xi_k = \phi_k$ for $k = 1, 2, 3$, augmented with $\phi_0 = \xi_0 = 1 - \xi_1 - \xi_2 - \xi_3$. The coordinate transformation is $\mathbf{x} = \mathbf{x}_0 + \xi_1(\mathbf{x}_1 - \mathbf{x}_0) + \xi_2(\mathbf{x}_2 - \mathbf{x}_0) + \xi_3(\mathbf{x}_3 - \mathbf{x}_0) = \sum_{k=0}^{3} \xi_k \mathbf{x}_k$ with Jacobian matrix $\mathbf{J} = \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\boldsymbol{\xi}} = (\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c)$.

It is convenient to define relative vertex positions

$$\mathbf{x}_a = \mathbf{x}_1 - \mathbf{x}_0, \quad \mathbf{x}_b = \mathbf{x}_2 - \mathbf{x}_0, \quad \mathbf{x}_c = \mathbf{x}_3 - \mathbf{x}_0,$$

and the cross products

$$\mathbf{g}_1 = \mathbf{x}_b \times \mathbf{x}_c, \quad \mathbf{g}_2 = \mathbf{x}_c \times \mathbf{x}_a, \quad \mathbf{g}_3 = \mathbf{x}_a \times \mathbf{x}_b .$$

Then, $\det \mathbf{J} = J_0 = \mathbf{x}_a \cdot \mathbf{g}_1 = 6V$, with V the volume of the tetrahedron. The matrix $\mathbf{F} = J_0 \mathbf{J}^{-\mathsf{T}}$ has $\mathbf{g}_k$, $k = 1, 2, 3$, as columns.

Note that

$$\mathbf{g}_1 \times \mathbf{g}_2 = J_0 \mathbf{x}_c, \quad \mathbf{g}_2 \times \mathbf{g}_3 = J_0 \mathbf{x}_a, \quad \mathbf{g}_3 \times \mathbf{g}_1 = J_0 \mathbf{x}_b .$$

The mass matrix $\mathbf{A}$ on the reference element has elements

$$A_{j,k} = \int_0^1 \xi_1 \int_0^{1-\xi_1} \mathrm{d}\xi_2 \int_0^{1-\xi_1-\xi_2} \mathrm{d}\xi_3 \ \phi_j(\boldsymbol{\xi}) \phi_k(\boldsymbol{\xi}),$$

for $j, k = 0, 1, 2, 3$. Mass lumping replaces this matrix by a diagonal one with the row sums as the diagonal elements: $A^{\mathrm{L}}_{j,k} = \delta_{j,k} \sum_{k=0}^{3} A_{j,k}$.

The nine stiffness matrices $\mathbf{B}^{m,n}$ on the reference element are

$$B^{m,n}_{j,k} = \int_0^1 \mathrm{d}\xi_1 \int_0^{1-\xi_1} \mathrm{d}\xi_2 \int_0^{1-\xi_1-\xi_2} \mathrm{d}\xi_3 \frac{\partial \phi_j}{\partial \xi_m} \frac{\partial \phi_k}{\partial \xi_n} .$$

They are symmetric: $\mathbf{B}^{n,m} = (\mathbf{B}^{m,n})^{\mathsf{T}}$.

For the higher order mass-lumped finite elements, the coordinate permutations listed in Appendix can simplify the implementation. Then, the code only has to define two arrays with pre-computed values on the reference element, for instance, $\mathbf{B}^{1,1}$ and $\mathbf{B}^{1,2}$, and the other seven follow from permutations and symmetries.

The stiffness matrix $\overline{\mathbf{B}}$ for the isotropic elastic system of equations per element can be constructed from the above $\mathbf{B}^{m,n}$. To obtain a matrix, the displacement components are taken as fastest index and the nodes as slowest. Then, the matrix elements are

$$J_0 \, \overline{B}_{m+3j,n+3k} = \sum_{p,q=1}^{3} F_{m,p} F_{n,q} \left(\lambda B^{p,q}_{j,k} + \mu B^{p,q}_{k,j} \right) + \mu \, \delta_{m,n} \sum_{p,q,r=1}^{3} F_{r,p} F_{r,q} B^{p,q}_{k,j} . \quad (1)$$

Here m and n run over the 3 components of the displacement, whereas j and k run over the nodes of the element: $m, n = 1, 2, 3$ and $j, k = 0, 1, \ldots, N_p - 1$. The number of nodes for the mass-lumped elements is $N_p = 4$ for degree 1, 23 for degree 2 and 50 for degree 3. The global stiffness matrix follows from the contributions of $\overline{\mathbf{B}}$ per element.

The upper triangular part of the sparse symmetric block matrix is stored in Block Compressed Sparse Row format, with 3×3 full blocks. The block diagonal is treated separately, as the small 3×3 blocks are symmetric and only 6 values need to be stored per element. Somewhat to our surprise, we found that our code, using OpenMP, outperformed the Intel Math Kernel Library routine `mkl_cspblas_scsrsymv()` that also uses OpenMP.

With local assembly, we can exploit the fact that the stiffness matrices $\mathbf{B}^{m,n}$ on the reference element have a zero row sum and, since they are symmetric, also a zero column sum. The zero row sum implies that the application of a stiffness matrix to a constant produces zero. We therefore define

$$v_k^m = u_k^m - u_0^m, \tag{2}$$

for nodes $k = 1, \ldots, N_p - 1$ and components $m = 1, 2, 3$, subtracting the values of the displacement components at the first vertex that corresponds to $k = 0$. Note that any node of the element can be selected here, with the first or last as a convenient choice.

Let $\mathbf{r} = \overline{\mathbf{B}}\mathbf{u} = \overline{\mathbf{B}}\mathbf{v}$ per element.

The zero column sum of the stiffness matrix implies

$$r_0^m = -\sum_{k=1}^{N_p-1} r_k^m, \quad m = 1, 2, 3. \tag{3}$$

This means that we can drop the first three rows and columns of the local elastic stiffness matrix $\overline{\mathbf{B}}$, work with v_k^m for $k = 1, \ldots, N_p - 1$ and $m = 1, 2, 3$, and reconstruct the first three entries of r_k^m by eq. (3). The result has to be multiplied by the pre-computed inverse of the diagonal global mass matrix and can then be used to increment the solution. Repeating this for all tetrahedra accomplishes the time step, together with the source term and interpolation to obtain the receiver traces at selected positions.

We can further simplify the evaluation of $\overline{\mathbf{B}}\mathbf{v}$.

Let $\mathbf{F}^\lambda = \frac{\lambda}{J_0}\mathbf{F}$, $\mathbf{F}^\mu = \frac{\mu}{J_0}\mathbf{F}$ and define the symmetric 3×3 matrix $\mathbf{C}^\mu = \frac{\mu}{J_0}\mathbf{F}^\mathsf{T}\mathbf{F} = \mathbf{F}^\mathsf{T}\mathbf{F}^\mu$. Define a set of 9 vectors for $p = 1, 2, 3$ and $q = 1, 2, 3$ with elements

$$\sigma_j^{p,q;n} = \sum_{k=1}^{N_p-1} B_{j,k}^{p,q}\left(u_k^n - u_0^n\right) = \sum_{k=1}^{N_p-1} B_{j,k}^{p,q} v_k^n,$$

for components $n = 1, 2, 3$ and nodes $j = 1, \ldots, N_p - 1$, ignoring node 0. Compute

$$\alpha_j^p = \sum_{n,q=1}^{3}\left(F_{n,q}^\lambda \sigma_j^{p,q;n} + F_{n,q}^\mu \sigma_j^{q,p;n}\right).$$

Then,

$$r_j^m = \sum_{p=1}^{3} F_{m,p}\alpha_j^p + \sum_{p,q=1}^{3} C_{p,q}^\mu \sigma_j^{q,p;m}.$$

for nodes $j = 1, \ldots, N_p - 1$ and components $m = 1, 2, 3$. Finally, use eq. (3) to obtain the values at node $j = 0$, multiply by the subset of the global inverse matrix on the element and update the solution. For degrees higher than one, the main computational effort consists in the nine matrix-vector products between the matrices $\mathbf{B}^{p,q}$ of the reference element and the vector $\mathbf{v}$.

Table 1. Pseudo-code in Matlab style for the evaluation of the stiffness matrix B per element for linear basis functions on a tetrahedron, with `glamba` as $\mathbf{g}^\lambda$ and `gmu` as $\mathbf{g}^\mu$, defined in the text. Unknowns are taken as triples of displacements on vertices 0 to 3.

```
glamba = (lambda/(6*J0))*g; gmu = (mu/(6*J0))*g;
B = zeros(12,12);
for k1=0:3:9,
  for k2=0:3:k1,
    s = 0;
    for m2=1:3,
      for m1=1:3,
        h1 = gla(k1+m1)*g(k2+m2);
        h2 = gmu(k1+m1)*g(k2+m2);
        B(k1+m1,k2+m2) = h1+h2;
        if(m1 == m2), s = s+h2; end
      end
    end
    for m=1:3,
      B(k1+m,k2+m) = B(k1+m,k2+m)+s;
    end
    if(k2 < k1),
       for m2=1:3,
         for m1=1:3,
           B(k2+m2,k1+m1) = B(k1+m1,k2+m2);
         end
       end
    end
  end
end
```

The standard second-order time stepping scheme reads

$$\mathbf{u}^{n+1} = 2\mathbf{u}^n - \mathbf{u}^{n-1} + (\Delta t)^2 \boldsymbol{\mathcal{M}}^{-1}(\mathbf{f} - \boldsymbol{\mathcal{K}}\mathbf{u}^n),$$

with global stiffness matrix $\boldsymbol{\mathcal{K}}$ and diagonal global mass matrix $\boldsymbol{\mathcal{M}}$. The inverse of mass matrix can in principle be avoided by considering the diagonal scaling

$$\boldsymbol{\mathcal{D}} = \Delta t \boldsymbol{\mathcal{M}}^{-1/2}, \quad \tilde{\mathbf{u}} = \boldsymbol{\mathcal{D}}^{-1}\mathbf{u}, \quad \tilde{\mathbf{f}} = \boldsymbol{\mathcal{D}}\mathbf{f},$$

and the symmetric matrix

$$\tilde{\boldsymbol{\mathcal{K}}} = \boldsymbol{\mathcal{D}}\,\boldsymbol{\mathcal{K}}\,\boldsymbol{\mathcal{D}},$$

leading to

$$\tilde{\mathbf{u}}^{n+1} = 2\tilde{\mathbf{u}}^n - \tilde{\mathbf{u}}^{n-1} + \tilde{\mathbf{f}} - \tilde{\boldsymbol{\mathcal{K}}}\tilde{\mathbf{u}}^n.$$

However, we have not used this approach for the numerical experiments reported further on as it complicates reading off receiver data. The required storage then consists in the solution at two time instances, which requires three times the number of nodes, the inverse mass matrix multiplied by $(\Delta t)^2$, also with a size equal to the number of nodes, and either the globally assembled sparse matrix or, with local assembly, the average of λ and of μ per element.

2.2 Linear element

The above expressions hold for any degree. For the linear element, we derive simpler expressions that will speed up the code.

Let $\mathbf{g}_0 = -\mathbf{g}_1 - \mathbf{g}_2 - \mathbf{g}_3$ and define a linear array $g_{m+3k} = g_k^m$, with nodes $k = 0, 1, 2, 3$ and components $m = 1, 2, 3$. Note that $g_k^m = F_{m,k}$ for $k = 1, 2, 3$. Let $\mathbf{g}^\lambda = \lambda/(6J_0)\mathbf{g}$ and $\mathbf{g}^\mu = \mu/(6J_0)\mathbf{g}$. Table 1 lists pseudo-code in Matlab style for the evaluation of the local stiffness matrix, $\overline{\mathbf{B}}$. When recoded in a language like C or C++, this code is more efficient than that of Alberty *et al.* (2002), which is geared towards use with Matlab.

For local assembly, let

$$s_m = \sum_{k=1}^{3} F_{m,k} v_k^m, \quad w_{m,m} = 2\frac{\mu}{6J_0} s_m,$$

and

$$w_{m,n} = w_{n,m} = \frac{\mu}{6J_0} \sum_{k=1}^{3} \left(F_{m,k} v_k^n + F_{n,k} v_k^m \right),$$

for $m < n$. Then, a simpler expression is

$$r_k^m = F_{m,k} \left(w_{m,m} + \frac{\lambda}{6J_0} \sum_{n=1}^{3} s_n \right) + \sum_{\substack{n=1 \\ n \neq m}}^{3} F_{n,k} w_{m,n},$$

for nodes $k = 1, 2, 3$ and components $m = 1, 2, 3$. Eq. (3) provides the values at node $k = 0$.

2.3 Acoustics

For the *acoustic* case, which we will briefly consider later on, it is convenient to define symmetric matrices

$$\overline{\mathbf{C}} = J_0 \mathbf{J}^{-1} \mathbf{J}^{-\mathsf{T}} = J_0^{-1} \mathbf{F}^{\mathsf{T}} \mathbf{F},$$

and

$$\tilde{\mathbf{B}}^{p,q} = \mathbf{B}^{p,q} + \mathbf{B}^{q,p} = \mathbf{B}^{p,q} + (\mathbf{B}^{p,q})^{\mathsf{T}}.$$

The contribution of an element to the stiffness matrix is

$$\overline{\mathbf{B}}^{\text{acou}} = \sum_{p=1}^{3} \left(\overline{C}_{p,p} \mathbf{B}^{p,p} + \sum_{q=p+1}^{3} \overline{C}_{p,q} \tilde{\mathbf{B}}^{p,q} \right), \tag{4}$$

where $\mathbf{B}^{p,p}$ and $\tilde{\mathbf{B}}^{p,q}$ are symmetric matrices on the reference element, containing pre-determined numerical values only, and $\overline{\mathbf{C}}$ deals with the geometry of the actual tetrahedron. For the linear element, the simplified expressions presented by Zhebel *et al.* (2014) are more efficient. For degrees 2 and 3 and with local assembly, the evaluation of $\overline{\mathbf{B}}^{\text{acou}}\mathbf{u}$ per element was implemented as six matrix-vector multiplications, namely $\mathbf{B}^{p,p}\mathbf{u}$ and $\tilde{\mathbf{B}}^{p,q}\mathbf{u}$. The vector $\mathbf{u}$ contains the pressure values on the nodes of the element. The matrices correspond to those in eq. (4) and were hardcoded from numerical values computed with Mathematica.

3 RESULTS

3.1 Linear element

As a test problem, we chose a homogeneous problem for which the exact solution is readily available. The constant material properties were a density $\rho = 2\,\text{g cm}^{-3}$, a *P*-wave velocity $v_p = 2\,\text{km s}^{-1}$ and an *S*-wave velocity $v_s = 1.2\,\text{km s}^{-1}$. The domain had a size $[-2, 2] \times [-1, 1] \times [0, 2]\,\text{km}^3$ and was divided into cubes with an edge length of 20 m. Each cube was partitioned into 6 tetrahedra, leading to 12 000 000 tetrahedra and 2 050 401 vertices. The cube has six possible tetrahedral decompositions. We used the periodic one, with matching diagonals on opposite faces and one diagonal to the cube's centre.

A vertical force source was placed at the centre of the domain. A line of receivers was located at a depth of 800 m with $y = 0$ m and x between -1925 and $+1925$ m, using a 50 m interval. The time steps started at -0.3875 s to let the 3.5 Hz Ricker wavelet peak at zero time. The time step, $\Delta t = 0.003125$ s, corresponded to 0.77 per cent of CFL-limit. Data were recorded up to 0.6 s at a 5 ms interval. We used the natural (free-surface) boundary conditions all around for simplicity.

Table 2. Performance on linear elements with global assembly of the stiffness matrix and with local assembly. For the latter, the wall-clock time with 24 threads is doubled in this particular example but less storage is needed.

Assembly	Threads	Assembly	Stepping	Total	Storage
Global	24	7.9 s	9.2 s	17.0 s	3.0 GB
	12	7.9 s	10.7 s	18.6 s	
	6	8.2 s	16.8 s	25.1 s	
Local	24		30.0 s	30.0 s	2.1 GB
	12		57.8 s	57.8 s	
	6		114 s	114 s	

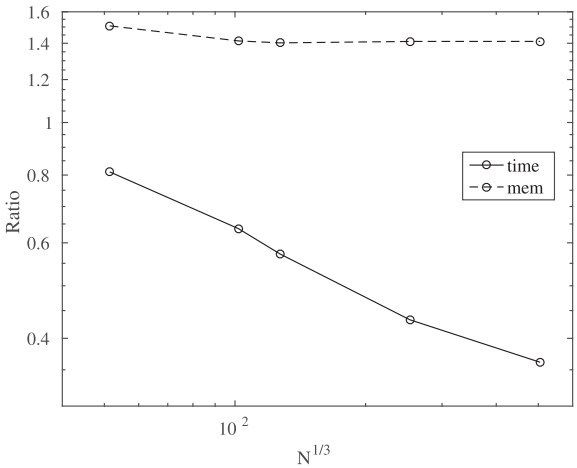


Figure 1. Ratios for compute time (drawn line) and storage (dashed) with linear elements as a function of $N^{1/3}$, where N is the number of nodes on the mesh. The results for the globally assembled case were divided by those for locally assembled stiffness matrices. The latter requires less storage, but is slower. The obtained reduction in storage does not seem to justify the larger compute times with linear elements.

Table 2 lists the timings and storage requirements using 24, 12 or 6 threads, all for the same mesh described earlier. Throughout this paper, reported timings are the average of 5 runs. The table shows that a smaller number of threads does not lead to a severe performance drop with global assembly, because memory access is the limiting factor. For local assembly on the fly, the performance is limited by the available compute power, at least up to the available 24 cores. OpenMP directives handled the multi-threading. The hardware consisted of a single board with two 12 core Intel Xeon CPU E5-2680 v3 processors running at 2.50 GHz and had hyper-threading disabled.

Fig. 1 shows the ratios between the runs with global assembly and those with local assembly, in terms of the required compute time and the maximum required storage, for a range of mesh sizes. Global assembly requires about 40 per cent more storage, but the gain in performance appears to amply justify that. Table 2 suggests that we could have used less than 24 threads for local assembly, as the computations are bound by memory access.

Note that the performance data should be taken as a rough indication, since the results strongly depend on code implementation, optimization and compiler. We did not put a lot of effort in code tuning for the specific compiler and hardware, but instead relied on the basic formulation of the method and the optimization ca-

Table 3. Ratio of compute time and memory with and without global assembly for the *acoustic* case on 24 cores.

M	Time	Storage
1	0.38	1.2
2	1.0	11
3	1.9	26

pabilities of the Intel compiler and OpenMP. The use of templated functions in terms of the number of nodes per element improved the performance of our C++ code.

3.2 The acoustic wave equation

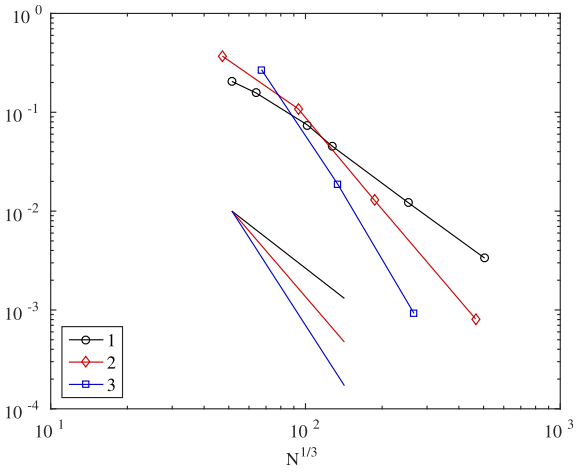
Before going to the higher-order elements for elastics, we briefly review the acoustic case, which can serve as a point of reference for the elastic problem. We consider the same test problem as before for degrees $M = 1$, 2 and 3. Table 3 lists the ratios of the compute time and of the required storage with and without global assembly, using 24 threads. The same tetrahedral mesh, derived from cubes with an edge length of 20 m, was used for each degree. For the linear element of degree 1, assembly of the global stiffness matrix reduces the required time significantly with only a 20 per cent increase of storage. For degree 2, there is no performance gain and the required storage is much larger. For degree 3, the scheme runs slower than with local assembly on the fly and requires a lot more memory. For that reason, Zhebel *et al.* (2011, 2014) only mentioned local assembly.

3.3 Higher orders

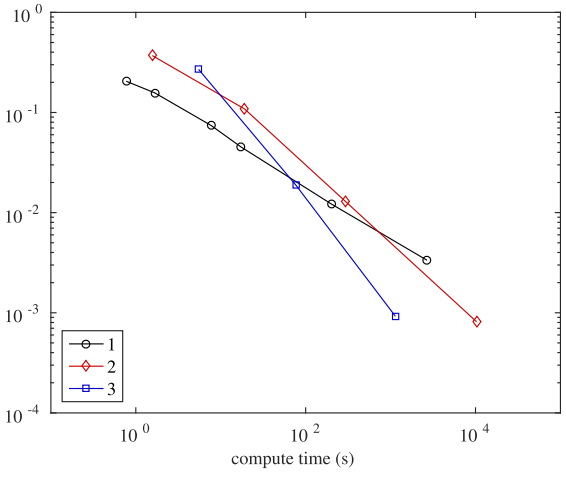
We now turn to the elastic case with discretizations of degrees 2 and 3, using the same homogeneous problem on meshes of different size.

Fig. 2(a) plots the maximum observed error in the receiver data for the vertical displacement component, scaled by the maximum amplitude over all traces, as a function of the number of scalar degrees of freedom or number of nodes. Fig. 2(b) depicts the same errors as a function of the required compute time with 24 cores. The actual number of degrees of freedom is $3N$ and equals the size of the numerical displacement vector $\mathbf{u}$. The element size scales with $N^{-1/3}$. The error is expected to behave as $N^{-(M+1)/3}$ for degree M. The numerical experiments more or less follow the expected trend. The compute time only includes the wall-clock time for sparse matrix assembly and time stepping, not the time spent on reading and checking the mesh, setting up the nodes, the local-to-global map, and locating source and receivers on the mesh. Because the scheme for degree $M = 1$ was treated in a different way, it performs quite well even with a large number of elements. If errors around 10 per cent are acceptable, it can be a viable alternative for the scheme of degree 3.

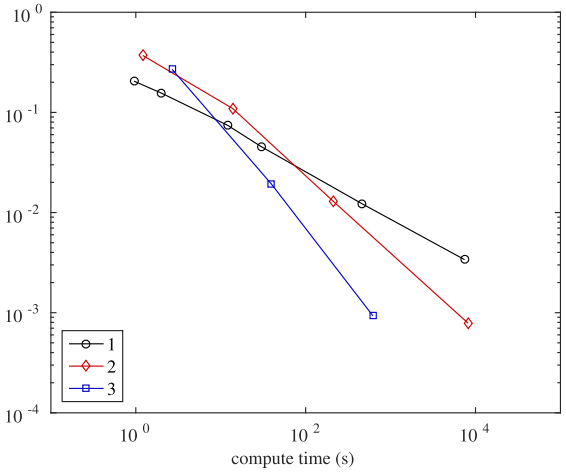
Fig. 2(c) is similar to Fig. 2(b), but with the product of the element stiffness matrix and element displacements evaluated on the fly during each time step. To better illustrate the differences in performance and memory usage, Fig. 3 plots the ratio in observed compute time as well as required storage between global assembly and local assembly for elements of degrees 1, 2 and 3. For degree 1, repeated from Fig. 1, the differences are not that large. Global assembly takes about 40 per cent more storage but runs more quickly. For degree 2, local assembly is faster by a factor of about 1.3 on 24 cores. For degree 3, it is about 1.9 times as fast. The savings in storage compared to global assembly are substantial. Therefore, global assembly may only be attractive for degree 1.



(a)



(b)



(c)

Figure 2. Maximum error in the vertical displacement, scaled by the maximum amplitude, for elements of degree 1, 2 or 3, as a function of (a) $N^{1/3}$, where N is the number of degrees of scalar degrees of freedom, (b) compute time with global assembly and (c) with local assembly. The extra set of three short lines in (a) depicts the theoretical asymptotic error behaviour.

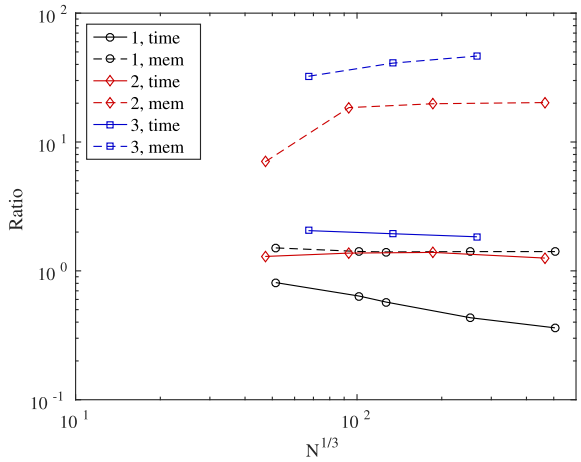


Figure 3. Ratios between compute time (drawn lines) and required storage (dashed lines) for global assembly and local assembly on the fly with elements of degree 1, 2 or 3. Global assembly is faster for degree 1 at 40 per cent more storage. For degrees 2 and 3, local assembly is faster and requires substantially less storage.

Table 4. Isotropic elastic properties: *P*- and *S*-wave velocities and densities are constant per layer.

v_p (km s^{-1})	v_s (km s^{-1})	ρ (g cm^{-3})
2.000	1.200	2.046
5.000	3.000	2.602
3.000	1.800	2.290
4.400	2.640	2.250
6.000	3.600	2.723
5.500	3.300	2.665

3.4 A more realistic example

We ran the code on the non-trivial model shown in fig. 10 of Zhebel *et al.* (2014), which is slightly more realistic than a homogeneous problem. The material properties are constant per layer and listed in Table 4 Fig. 4(a) displays a vertical cross section of the P-wave velocity. The source, indicated by a red star, is a vertical force at the surface, and has the signature of a Ricker wavelet with an 8 Hz peak frequency. The vertical displacement after 1 second in Fig. 4(b) shows strong Rayleigh waves. The tetrahedral mesh has 1 528 595 vertices and 8 826 636 elements of degree 3. The time step was about 75 per cent of the maximum value dictated by the CFL condition. Fig. 4(c) shows the vertical displacement, measured at the surface along a line corresponding to the earlier vertical cross section. The computation ran up till a time of 2 s..

Fig. 5 plots the observed ratios between compute time and memory requirements with global and with local assembly on different meshes using 24 cores. The behaviour is similar to that of Fig. 3. Again, global assembly is only faster for the linear elements, whereas local assembly on the fly wins for degree 2 and 3. For the latter, the performance gain now is about 2.5 and 3 times, respectively.

4 CONCLUSIONS

We have compared the performance of mass-lumped tetrahedral finite elements on isotropic elastic wave propagation without and with global assembly of the stiffness matrix. To preserve their

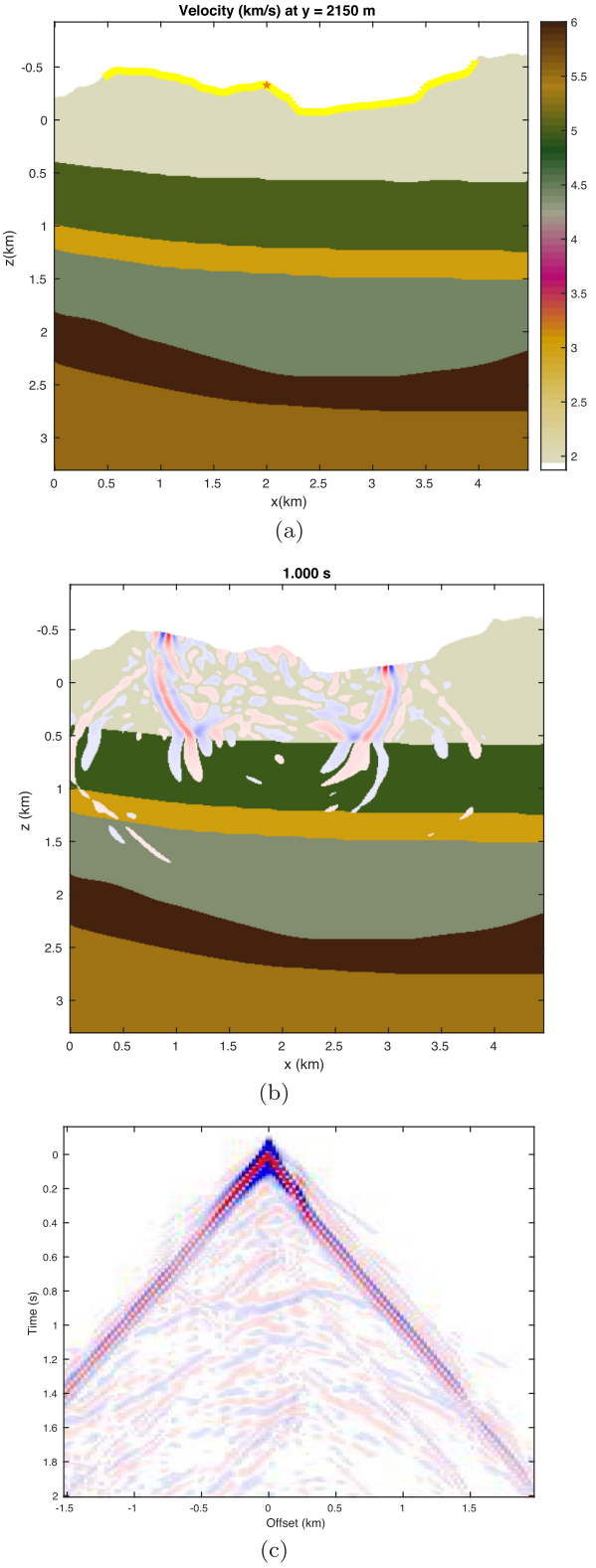


Figure 4. (a) *P*-wave velocities in km s^{-1}. The red star denotes the source positions and the yellow inverted triangles denote the receivers. (b) Vertical-displacement wavefield after 1 s. (c) Seismogram with vertical displacement.

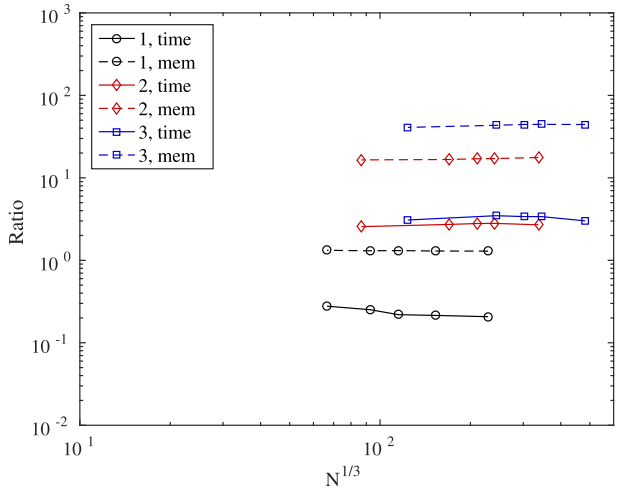


Figure 5. Ratios between compute time (drawn lines) and required storage (dashed lines) for global assembly and local assembly on the fly with elements of degree 1, 2 or 3. As in Fig. 3, global assembly is faster for degree 1 at 30–40 per cent more storage, whereas for degrees 2 and 3, local assembly is faster and requires substantially less storage.

accuracy after mass lumping, the higher-order elements are augmented with higher-degree polynomials in the interior of the faces and the tetrahedron. For the lowest degree, the linear elements, this is not necessary. For that case, we simplified the expression for the stiffness matrix.

We ran performance tests on a homogeneous problem. The parallelization of the most compute intensive loops was performed by OpenMP directives. With global assembly, this involved symmetric sparse matrix assembly and the matrix-vector product during the time stepping. With assembly on the fly, the local assembly and local matrix-vector multiplication per element were parallelized in a single OpenMP loop. Further code optimizations were left to the compiler.

In the acoustic case, local assembly is more efficient than global assembly, except for the lowest-order case with linear elements. In the elastic case, the same appears to be true. For degree 1, the code with global assembly ran faster and used about 40 per cent more storage than with local assembly. For degree 2, the numerical experiments with local assembly on the fly on 24 cores were about 1.4 times faster than with global assembly in one experiment and about two times in another. For degree 3, the gain was a factor 1.9 in one and 3 in the other. At the same time, the memory requirements were smaller by at least on order of magnitude for degrees 2 and 3.

We observed in a simple test problem that, for high accuracy, augmented cubic elements performed best in terms of compute time for a given accuracy. For low accuracy, the linear element may still be attractive. In that case, its efficiency compensates the need for a much finer mesh.

ACKNOWLEDGEMENTS

This work was partly funded by the Industrial Partnership Programme (IPP) ‘Computational sciences for energy research’ of the Foundation for Fundamental Research on Matter (FOM), which is part of the Netherlands Organisation for Scientific Research (NWO). This research programme is co-financed by Shell Global Solutions International B.V.

REFERENCES

Alberty, J., Carstensen, C., Funken, S.A. & Klose, R., 2002. Matlab implementation of the finite element method in elasticity, *Computing,* **69**(3), 239–263.

Bao, H., Bielak, J., Ghattas, O., Kallivokas, L.F., O’Hallaron, D.R., Shewchuk, J.R. & Xu, J., 1998. Large-scale simulation of elastic wave propagation in heterogeneous media on parallel computers, *Comput. Methods Appl. Mech. Eng.,* **152**(1–2), 85–102, containing papers presented at the Symposium on Advances in Computational Mechanics.

Bécache, E., Joly, P. & Tsogka, C., 2002. A new family of mixed finite elements for the linear elastodynamic problem, *SIAM J. Numer. Anal.,* **39**(6), 2109–2132.

Brossier, R., Virieux, J. & Operto, S., 2008. Parsimonious finite-volume frequency-domain method for 2-D *P-SV*-wave modelling, *Geophys. J. Int.,* **175**(2), 541–559.

Chin-Joe-Kong, M.J.S., Mulder, W.A. & van Veldhuizen, M., 1999. Higher-order triangular and tetrahedral finite elements with mass lumping for solving the wave equation, *J. Eng. Math.,* **35**(4), 405–426.

Cockburn, B., Gopalakrishnan, J. & Lazarov, R., 2009. Unified hybridization of discontinuous Galerkin, mixed, and continuous Galerkin methods for second order elliptic problems, *SIAM J. Numer. Anal.,* **47**(2), 1319–1365.

Cohen, G. & Fauqueux, S., 2005. Mixed spectral finite elements for the linear elasticity system in unbounded domains, *SIAM J. Sci. Comput.,* **26**(3), 864–884.

Cohen, G., Joly, P. & Tordjman, N., 1995. Higher order triangular finite elements with mass lumping for the wave equation, in *Proceedings of the Third International Conference on Mathematical and Numerical Aspects of Wave Propagation,* pp. 270–279, SIAM, Philadelphia.

Cohen, G., Joly, P., Roberts, J.E. & Tordjman, N., 2001. Higher order triangular finite elements with mass lumping for the wave equation, *SIAM J. Numer. Anal.,* **38**(6), 2047–2078.

Dumbser, M. & Käser, M., 2006. An arbitrary high-order Discontinuous Galerkin method for elastic waves on unstructured meshes – II. The three-dimensional isotropic case, *Geophys. J. Int.,* **167**(1), 319–336.

Dumbser, M., Käser, M. & de la Puente, J., 2007. Arbitrary high-order finite volume schemes for seismic wave propagation on unstructured meshes in 2D and 3D, *Geophys. J. Int.,* **171**(2), 665–694.

Etienne, V., Chaljub, J., Virieux, J. & Glinsky, N., 2010. An hp-adaptive discontinuous Galerkin finite elements method for 3-D elastic wave modelling, *Geophys. J. Int.,* **183**(2), 941–962.

Fried, I. & Malkus, D.S., 1975. Finite element mass matrix lumping by numerical integration with no convergence rate loss, *Int. J. Solids Struct.,* **11,** 461–466.

Giorgiani, G., Fernández-Méndezand, S. & Huerta, A., 2013. Hybridizable discontinuous Galerkin p-adaptivity for wave propagation problems, *Int. J. Numer. Methods Fluids,* **72**(12), 1244–1262.

Käser, M. & Dumbser, M., 2006. An arbitrary high-order Discontinuous Galerkin method for elastic waves on unstructured meshes – I. The two-dimensional isotropic case with external source terms, *Geophys. J. Int.,* **166**(2), 855–877.

Komatitsch, D. & Tromp, J., 1999. Introduction to the spectral-element method for 3-D seismic wave propagation, *Geophys. J. Int.,* **139**(3), 806–822.

Kononov, A., Minisini, S., Zhebel, E. & Mulder, W.A., 2012. A 3D tetrahedral mesh generator for seismic problems, in *Proceedings of the 74th EAGE Conference & Exhibition,* p. B006, EAGE.

Lesage, A.C., Aubry, R., Houzeaux, G., Polo, M.A. & Cela, J.M., 2010. 3D spectral element method combined with H-refinement, in *72nd EAGE Conference & Exhibition*, Barcelona, Spain, Extended Abstracts, doi:10.3997/2214-4609.201400702.

Maday, Y. & Rønquist, E.M., 1990. Optimal error analysis of spectral methods with emphasis on non-constant coefficients and deformed geometries, *Comput. Methods Appl. Mech. Eng.,* **80**(1–3), 91–115.

Mercerat, E.D., Vilotte, J.P. & Sánchez-Sesma, F.J., 2006. Triangular Spectral Element simulation of two-dimensional elastic wave propagation using unstructured triangular grids, *Geophys. J. Int.,* **166**(2), 679–698.

Moczo, P., Kristek, J., Galis, M., Chaljub, E. & Etienne, V., 2011. 3-D finite-difference, finite-element, discontinuous-Galerkin and spectral-element schemes analysed for their accuracy with respect to *P*-wave to *S*-wave speed ratio, *Geophys. J. Int.,* **187**(3), 1645–1667.

Mulder, W.A., 1996. A comparison between higher-order finite elements and finite differences for solving the wave equation, in *Proceedings of the Second ECCOMAS Conference on Numerical Methods in Engineering,* pp. 344–350, John Wiley & Sons, Chichester.

Mulder, W.A., 2013. New triangular mass-lumped finite elements of degree six for wave propagation, *Prog. Electromagn. Res.,* **141,** 671–692.

Mulder, W.A., Zhebel, E. & Minisini, S., 2014. Time-stepping stability of continuous and discontinuous finite-element methods for 3-D wave propagation, *Geophys. J. Int.,* **196**(2), 1123–1133.

Orszag, S.A., 1980. Spectral methods for problems in complex geometries, *J. Comput. Phys.,* **37**(1), 70–92.

Patera, A.T., 1984. A spectral element method for fluid dynamics: laminar flow in a channel expansion, *J. Comput. Phys.,* **54**(3), 468–488.

Rivière, B. & Wheeler, M.F., 2003. Discontinuous finite element methods for acoustic and elastic wave problems, in *Contemp. Math.,* **329,** 271–282.

Seriani, G., Priolo, E., Carcione, J. & Padovani, E., 1992. High-order spectral element method for elastic wave modeling, *SEG Technical Program Expanded Abstracts,* **11,** 1285–1288.

Sherwin, S.J. & Karniadakis, G.E., 1995. A new triangular and tetrahedral basis for high-order (hp) finite element methods, *Int. J. Numer. Methods Eng.,* **38**(22), 3775–3802.

Tordjman, N., 1995. Élements finis d'order élevé avec condensation de masse pour l'equation des ondes, *PhD thesis,* L'Université Paris IX Dauphine.

Wang, X., Symes, W.W. & Warburton, T., 2010. Comparison of discontinuous Galerkin and finite difference methods for time domain acoustics, *SEG Technical Program Expanded Abstracts,* **29**(1), 3060–3065.

Wilcox, L.C., Stadler, G., Burstedde, C. & Ghattas, O., 2010. A high-order discontinuous Galerkin method for wave propagation through coupled elastic-acoustic media, *J. Comput. Phys.,* **229**(24), 9373–9396.

Zhebel, E., Minisini, S., Kononov, A. & Mulder, W.A., 2011. Solving the 3D acoustic wave equation with higher-order mass-lumped tetrahedral finite elements, in *Proceedings of the 73rd Conference & Exhibition,* Vienna, Austria, p. A010.

Zhebel, E., Minisini, S., Kononov, A. & Mulder, W.A., 2014. A comparison of continuous mass-lumped finite elements with finite differences for 3-D wave propagation, *Geophys. Prospect.,* **62**(5), 1111–1125.

Zienkiewicz, O.C. & Taylor, R.L., 2000. *The Finite Element Method. Volume 1: The Basis,* 5th edn, Butterworth-Heinemann.

APPENDIX: PERMUTATIONS

Given the symmetries of the node positions, we can define various permutation arrays and their corresponding matrices. Let the N_p nodes of the element be $\mathbf{x}_k$, $k = 0, \ldots, N_p - 1$. The permutation array $\mathbf{p}^{2,1}$ swaps their x and y coordinates with $\mathbf{x}_{\mathbf{p}^{2,1}(k)}$ as result. Likewise, $\mathbf{p}^{3,1}$ swaps x and z and $\mathbf{p}^{3,2}$ interchanges y and z.

To these arrays correspond matrices $\mathbf{P}^{m,n}$ with elements $\mathbf{P}^{m,n}_{k,p^{m,n}_k} = 1$ and zero otherwise. The inverse and transpose of the permutation matrix equal the matrix itself:

$$(\mathbf{P}^{m,n})^{-1} = (\mathbf{P}^{m,n})^{\mathsf{T}} = \mathbf{P}^{m,n}.$$

With these matrices, the stiffness matrices obey

$$\mathbf{B}^{2,2} = \mathbf{P}^{2,1}\mathbf{B}^{1,1}\mathbf{P}^{2,1}, \quad \mathbf{B}^{3,3} = \mathbf{P}^{3,1}\mathbf{B}^{1,1}\mathbf{P}^{3,1},$$

$$\mathbf{B}^{1,3} = \mathbf{P}^{3,2}\mathbf{B}^{1,2}\mathbf{P}^{3,2}, \quad \mathbf{B}^{3,2} = \mathbf{P}^{3,1}\mathbf{B}^{1,2}\mathbf{P}^{3,1}.$$

Because $(\mathbf{B}^{m_1,m_2})^{\mathsf{T}} = \mathbf{B}^{m_2,m_1}$, we have

$$\mathbf{B}^{3,1} = \mathbf{P}^{3,2}\mathbf{B}^{2,1}\mathbf{P}^{3,2}, \quad \mathbf{B}^{2,3} = \mathbf{P}^{3,1}\mathbf{B}^{2,1}\mathbf{P}^{3,1}.$$

Also,

$$\mathbf{B}^{1,2} = \mathbf{P}^{2,1}\mathbf{B}^{2,1}\mathbf{P}^{2,1}, \quad \mathbf{B}^{2,1} = \mathbf{P}^{2,1}\mathbf{B}^{1,2}\mathbf{P}^{2,1},$$

$$\mathbf{B}^{1,3} = \mathbf{P}^{3,1}\mathbf{B}^{3,1}\mathbf{P}^{3,1}, \quad \mathbf{B}^{3,1} = \mathbf{P}^{3,1}\mathbf{B}^{1,3}\mathbf{P}^{3,1},$$

$$\mathbf{B}^{3,2} = \mathbf{P}^{3,2}\mathbf{B}^{2,3}\mathbf{P}^{3,2}, \quad \mathbf{B}^{2,3} = \mathbf{P}^{3,2}\mathbf{B}^{3,2}\mathbf{P}^{3,2}.$$

In summary: with two matrices $\mathbf{B}^{1,1}$ and $\mathbf{B}^{1,2}$, computed on the reference element, and three permutation vectors, $\mathbf{p}^{2,1}$, $\mathbf{p}^{3,1}$, and $\mathbf{p}^{3,2}$, all nine element stiffness matrices $\mathbf{B}^{p,q}$ can be determined.