<table>
<thead>
<tr>
<th>Title</th>
<th>On increasing the order and density of 3D finite element meshes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Lee, Chi King; Wong, S. M.; Lie, S. T.</td>
</tr>
<tr>
<td>Date</td>
<td>2001</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/10220/19244">http://hdl.handle.net/10220/19244</a></td>
</tr>
<tr>
<td>Rights</td>
<td>© 2001 John Wiley &amp; Sons, Ltd. This is the author created version of a work that has been peer reviewed and accepted for publication by Communications in Numerical Methods in Engineering, John Wiley &amp; Sons, Ltd. It incorporates referee’s comments but changes resulting from the publishing process, such as copyediting, structural formatting, may not be reflected in this document. The published version is available at: [DOI:<a href="http://dx.doi.org/10.1002/1099-0887(200101)17:1">http://dx.doi.org/10.1002/1099-0887(200101)17:1</a>&lt;55::AID-CNM389&gt;3.0.CO;2-O].</td>
</tr>
</tbody>
</table>
On increasing the order and density of 3D finite element meshes

C. K. Lee*, S. M. Wong and S. T. Lie

School of Civil and Structural Engineering
Nanyang Technological University
Nanyang Avenue, Singapore 639798

*Email: ccklee@ntu.edu.sg

Pre-printed version for the paper appeared in the journal

Summary
A simple procedure is suggested to increase the order of 3D finite element meshes. The method suggested can be applied to any 3D finite element mesh containing different commonly used 3D element types such as tetrahedron, pyramid, prism and hexahedron. Based on the element order increasing scheme, an element subdivision scheme is also proposed to carry out uniform refinement for general 3D meshes. Detailed timings indicate that the computational time needed to increase the order and density of the mesh is proportional to the number of elements generated.

Keywords: 3D solid elements, Higher order elements, Doubling and tripling of mesh density
1 Introduction

In recent years, the use of 3D elements in finite element analysis becomes more and more popular [1,2] due to the rapid increase in the computational and storage capacity of modern computers as well as the many advancements in 3D mesh generation [3]. Currently, 3D elements are now frequently used in the analysis of thin wall structures [4] for the advantage that they do not possess any rotational degree of freedom. Most of the previous works done related to 3D meshing can be classified into two main areas:

1) Automatic 3D mesh generation schemes using various techniques such as the advance front method, the Delaunay triangulation technique, the octree approach and other mapping procedures [3].

2) 3D Mesh manipulating tools for carry out some routine but important operations for 3D meshes such as local refinement [5], coarsening and dividing [6], Boolean operations [7] and shape quality measurements [8].

The objective of this paper is to introduce an efficient algorithm to raise the element order and density of general 3D meshes. It should be noted that the quadratic element is frequently the optimal element type [2] to be used in practice. Hence, to automatically increase the order of a 3D mesh will definitely be a useful tool in practice. Based on the element order increasing algorithm, element subdivision algorithm for some commonly used 3D elements will also be suggested. The division algorithm can be used for the uniform refinement of any (structured or unstructured) 3D finite element mesh.

2 Increasing the order of 3D element mesh

Even though it is always possible to fill up an indefinite space by using only the tetrahedral element, more than one element types are often employed to generate a suitably graded mesh. Besides the simplest tetrahedral (T4) element, the hexahedral (H8) element, the prism (P6) element and the pyramid (PR5) element are among the most commonly used 3D elements. The local node and face numberings of these elements are shown in Figs. 1 to 4.

In practical 3D finite element analysis, linear 3D elements are seldom used due to their poor accuracy. Indeed, the common practice is that once the linear mesh is formed, it will be converted to a higher order mesh. Higher order elements are formed by inserting edge, face or interior nodes to the linear elements. When increasing the order of a linear element, two options are normally available. The first option is to insert edge nodes together with face and interior nodes to generate higher order Lagrangian elements. For example, the linear H8 element (Fig. 2) can be converted to a quadratic H27 element (Fig. 5) or to a cubic H64...
element (Fig. 6). The second option is to insert edge nodes only and generate higher order serendipity elements. For example, the H8 element can be converted to the quadratic H20 or the cubic H32 elements. Similarly, other higher order Lagrangian and serendipity elements can be created and their properties are summarized in Figs. 7 to 12. Note that in order to facilitate the element order raising procedure, the nodes for the elements are numbering in a hierarchical (corner-edge-face-interior) manner. Furthermore, serendipity elements can also be easily formed by excluding all the faces and interior nodes of the corresponding Lagrangian elements.

The process of increasing the order of linear elements can be divided into three steps:

1) **Insertion of edge nodes**

If it is required to generate elements without any face and interior node, the edge nodes insertion process is the only step needed. Before the node insertion process is started, two sets of data structures, namely, the node-element connectivity graph and the unique line information of the mesh will be formed first. The node-element connectivity graph will record the elements connect to a given node and can be efficiently represented by two linear integer arrays XEADJ and EADJ. The array XEADJ of size NN+1, where NN is the number of nodes in the mesh, is the index array for the node-element connectivity list, EADJ, such that elements connected to node I can be found in EADJ(J), for J=XEADJ(I),....,XEADJ(I+1)-1. For example, for the 3D finite element mesh shown in Fig. 13a, the arrays XEADJ and EADJ will be defined as

\[
\begin{align*}
NN &= 13, \\
XEADJ &= \{1, 2, 3, 4, 5, 8, 11, 14, 17, 18, 21, 22, 23, 24\} \\
EADJ &= \{1, 1, 1, 1, 1, 2, 3, 1, 3, 4, 1, 3, 4, 1, 2, 3, 2, 2, 3, 4, 4, 4, 4\}
\end{align*}
\] (1a, 1b)

A unique line or edge in a 3D mesh can be identified by the node numbers of the two end nodes of the line. If one uses the convention that the end node with smaller node number is used as the first node in the unique line set (i.e. the unique line joining nodes 1 and 4 is represented as \{1,4\} rather than \{4,1\}), then the unique line information of the mesh can be represented by two integer arrays XUADJ and UADJ. The array XUADJ of size NN+1 is the index array for UADJ. The end node numbers of unique lines start from node I are stored in UADJ(J), for J=XUADJ(I),....,XUADJ(I+1)-1 such that XUADJ(J)>I. For example, for the 3D mesh shown in Fig. 13a, the arrays XUADJ and UADJ will equal to...
Note that the arrays XUADJ and UADJ can be constructed from the arrays XEADJ and EADJ. By using these two arrays to represent the unique line information, all unique line will only be stored once without repetition. Therefore, the edge node insertion process can be done by visiting the unique lines in the mesh one by one and inserting one or two edge nodes to it according to the order of the output mesh. For the coordinates of the edge nodes, they will be placed at the midpoint and at the one- and two-third points of the unique lines for quadratic and cubic elements respectively.

(2) Face nodes generation

If Lagrangian elements are to be generated, faces nodes will be added to the element. Before the face node insertion process is started, an element-face connectivity graph which defines the connectivity information of the elements through their faces will be created from the node-element connectivity list. Again, two linear integer arrays MPF and MF will be used. The array MPF of size NE+1, where NE is the number of elements in the mesh, is the index array for MF such that elements connected to element I through its faces are stored in MF(J), for J=MPF(I),.....,MPF(I+1)-1 while a zero value of MF(J) indicates that no element is connected to element I through that face. (The local face numberings for the elements are shown in Figs. 1 to 4.) For the mesh shown in Fig. 13a, if the local element face numberings are defined in Fig. 13b, the MF and MPF arrays will be given by

\[ \text{NE}=4, \text{MPF} = \{1, 7, 11, 16, 21\} \]  \hspace{1cm} (3a)

\[ \text{MF} = \{0, 0, 0, 3, 0, 0, 0, 3, 0, 0, 0, 2, 1, 0, 0, 0, 3\} \]  \hspace{1cm} (3b)

Once the arrays MPF and MF are formed, the face node insertion process can be carried out by visiting all the faces of the elements one by one and insert face nodes to the two elements connected to the face.

(3) Creation of interior nodes

Since interior nodes of an element do not connect to other elements in the mesh, the process of creation of interior nodes will only involve a single loop to visit all the elements in the mesh.
3 Doubling and tripling the density of 3D finite element meshes

Despite the recent advancements in automatic mesh generation, it is still not a trivial task to carry out adaptive refinement for 3D problems. As a result, uniform refinement, which does not require a complete re-generation of 3D mesh, remains one of an attractive techniques to assess the accuracy of the finite element solution. In general, the convergence rate of a given quality $\rho$ obtained from finite element analysis (e.g. the error norm or stress at a given point) can often be expressed in the form [9]

$$\rho - \rho_i = C(h_i)^\alpha, \text{ for } i=1,2,3,...$$  \hspace{1cm} (4)

where $\alpha$ is the convergence rate of $\rho$ and $\rho_i$ is the approximated solution obtained by using a mesh with element size $h_i$. $C$ is a constant independent of both $h_i$ and $\alpha$. Hence, if three uniform finite element analyses are carried out with element sizes equal to $h_1$, $h_1/2$ and $h_1/3$ respectively, one can write

$$\rho - \rho_1 = C(h_1)^\alpha, \rho - \rho_2 = C(h_1/2)^\alpha, \rho - \rho_3 = C(h_1/3)^\alpha$$  \hspace{1cm} (5)

From Eqn. 5, a more accurate estimated value of $\rho$ can be obtained by eliminating $\alpha$ and solving the equation [9]

$$\frac{\rho - \rho_3}{\rho - \rho_2} = \left(\frac{\rho - \rho_2}{\rho - \rho_1}\right)^{\text{ln}(h_1/3)/(h_1/2)} = \left(\frac{\rho - \rho_2}{\rho - \rho_1}\right)^{0.589625}$$  \hspace{1cm} (6)

Finite element meshes with double and triple mesh density can be obtained by first increasing the order of the elements to quadratic or cubic respectively and then subdividing the high order elements into linear elements. For example, a H8 element can be doubled (tripled) by first converted it to a H27 (H64) element and then subdivide the high order element to 8 (27) H8 elements (Fig. 14). Obviously, other 3D elements can also be divided in a similar manner. Note that during mesh doubling, the PR14 element will be divided to 6 PR5 and 4 T4 elements instead of 8 PR5 elements (Fig. 15). Similarly, during mesh tripling, a PR30 element will be divided to 19 PR5 elements and 16 T4 elements. Such division schemes will ensure the compatibility at the element interfaces when different elements are joined to a PR5 element.
4 Numerical examples

Example 1: Benchmark speed test

In the first example, a benchmark test will be carried out to assess the speed of the element order and mesh density increasing algorithms. Four single-element meshes (one from each linear element type described in Section 2) are used as the initial meshes for the test. The mesh doubling and tripling procedures are then repeatedly applied to generate meshes with different element density. An example for the T4 mesh is shown in Fig. 16 to illustrate the steps involved in the generation of the last two refined meshes. In order to generate a mesh with 96 elements along an edge (the 96×mesh), the initial mesh was doubled twice, tripled once and then doubled twice again (i.e. 2×2×3×2×2) while the 81×mesh can be obtained by tripling the initial mesh four times (i.e. 3×3×3×3). In general, any n×mesh can be generated by doubling and tripling provide that n is divisible by 2 or 3. The CPU times needed to generating a n×mesh for n=24 to 96 by doubling and tripling (using a PC equipped with a Pentium III 450MHz processor) are shown in Figs. 17 and 18 respectively. It can be seen that the CPU times used are proportional to the number of elements generated. In addition, the speed of the refinement is very fast. It only takes about 7.5s and 4.5s for a low-ended PC to generate nearly 0.9 million H8 elements by doubling and tripling respectively.

Example 2: Meshing generation for cracked tubular Y-joint

In the second example, the element order and density increasing schemes are applied to the mesh generation problem for cracked tubular joints. In offshore engineering, full 3D meshes are often used to study the detailed stress distribution near the joint and the crack mouth. However, for many thin-walled structures, the use of conventional 3D solid mesh generators will often result in a mesh with too many elements. Hence, tailor-made mesh generators which employ multiple mapping procedures are often used to generate element meshes for the joint model [10]. Since it is rather difficult to carry out an adaptive refinement procedure for this type of structures, uniform refinement, though could be more expensive in computational cost, reminds an attractive approach to study the convergence of the numerical solutions. A typical finite element mesh for a cracked tubular T-joint with welding details and surface crack at the crown is shown in Fig. 19. Two uniform refinements were carried out by doubling and tripling the mesh and the meshes generated near the joint and the crack are shown in Fig. 20. During the mesh generation procedure, the T-joint model is divided into different zones with different element density. In regions far from the joint intersection, only
one layer of 3D elements will be employed in the single density mesh (Fig. 19a). On the other hand, fine meshes with three layers of elements are generated near the joint intersection. One advantage of generating the mesh zone by zone is that only the region near the intersection (Fig. 19c) are modified when specially designed meshes are generated around the surface crack mouth (Fig. 20b to 20d).

5 Conclusions
In this paper, algorithms have been developed to increase the order and density of general 3D finite element meshes. The mesh order increasing algorithm can be applied to any 3D mesh consists of tetrahedral, pyramid, prism and hexahedral elements. It can convert linear finite element meshes to either quadratic or cubic meshes and both Lagrangian and serendipity elements can be generated. Based on this order increasing scheme, a simple subdivision procedure is developed to break down the higher order elements and to generate finite element meshes with double or triple element density. Furthermore, detail timings indicate that their operation complexities are proportional to the number of elements generated and are fast enough for almost any practical application.
Reference


Fig. 1 Local node and face numberings for the T4 element.
Note: Local coordinates of nodes are inside brackets.

Fig. 2 Local node and face numberings for the H8 element.

Fig. 3 Local node and face numberings for the P6 element.
Face 1, $\xi + \zeta = 0$

Face 2, $\eta + \zeta = 0$

Face 3, $-\xi + \zeta = 0$

Face 4, $-\eta + \zeta = 0$

Face 5, $\zeta = -1$

Fig. 4 Local node and face numberings for the PR5 element

Fig. 5 Local node numberings for the H27 and H20 elements
Note: The H20 element is formed by excluding of all face nodes (nodes 21-26) and the interior node 27.
Fig. 6 Local node numberings for the H64 and the H32 elements
Note: The H32 element is formed by excluding all face nodes (nodes 33-56) and interior nodes (nodes 57-64).

Fig. 7 Local node numbering for the T10 element
Fig. 8 Local node numberings for the T20 and the T16 elements
Note: The T16 element is formed by excluding all face nodes (nodes 17-20).

Fig. 9 Local node numberings for the P18 and the P15 elements
Note: The P15 element is formed by excluding all face nodes (nodes 16-18).
Fig. 10 Local node numberings for the P40 and the P24 elements
Note: The P24 element is formed by excluding all face nodes (nodes 25-38) and interior nodes (nodes 39 and 40).
Fig. 11 Local node numberings for the PR14 and the PR13 elements
Note: The PR13 element is formed by excluding the face node 14.

Fig. 12 Local node numberings for the PR30 and the PR21 elements
Note: The PR21 element is formed by excluding all face nodes (nodes 22-29) and interior node 30.

Fig. 13 A simple finite element mesh
Fig. 14 Doubling the mesh density for a H8 element

Fig. 15 Dividing a PR5 element to 6 PR5 elements and 4 T4 elements
Initial mesh with one element (1 mesh)

Doubling (2×)

Tripling (3×)

2× mesh

2×2×3

24×mesh

24×2×2

3× mesh

3×3

27×mesh

27×3

96× mesh

81× mesh

Fig. 16 Uniform refinements used in Example 1
Fig. 17 CPU time required for mesh doubling

Fig. 18 CPU time required for mesh tripling
Fig. 19 A full 3D finite element mesh for cracked tubular T-joint
Fig. 20 Double and tripled mesh near the joint and crack surface