ORIGINAL ARTICLE

Efficient tetrahedral remeshing of feature models for finite element analysis

M. Sypkens Smit · W. F. Bronsvoort

Received: 21 July 2008/Accepted: 22 April 2009/Published online: 27 June 2009 © The Author(s) 2009. This article is published with open access at Springerlink.com

Abstract Finite element analysis is nowadays widely used for product testing. At various moments during the design phase, aspects of the physical behaviour of the product are simulated by performing an analysis of the model. For each analysis, a mesh needs to be created that represents the geometry of the model at that point. In particular during the later stages of the development cycle, often only minor modifications are made to a model between design iterations. In that case it can be beneficial to reuse part of the previous mesh, especially if it was costly to construct. A new method is presented that efficiently constructs a tetrahedral mesh based on a tetrahedral mesh of a feature model at an earlier point of the design cycle. This is done by analysing the difference of the two feature models from the point of view of the individual features. By this means we can find a natural correspondence between the geometries of the feature models, and relate this to the mesh of the earlier model. We discuss the algorithm, gained improvements, quality of the results, and conditions for this method to be effective.

Keywords Finite elements analysis · Remeshing · Feature model · Cellular model · Feature difference

M. Sypkens Smit (⊠) · W. F. Bronsvoort
Computer Graphics and CAD/CAM Group, Faculty of Electrical
Engineering, Mathematics and Computer Science,
Delft University of Technology, Mekelweg 4,
2628 CD Delft, The Netherlands
e-mail: m.sypkenssmit@tudelft.nl

W. F. Bronsvoort e-mail: w.f.bronsvoort@tudelft.nl

1 Introduction

Finite element analysis (FEA) is nowadays widely used by industry to perform product tests. These tests reduce the number of real world test models that have to be built. This is beneficial as building prototypes is costly in terms of both time and money.

Although FEA saves time and money in comparison to traditional product testing, it is nonetheless a time-consuming operation by itself. For complex models, the analysis process can take multiple months from start to finish, with the actual numerical analysis taking far less time than the work in preparation of the analysis. This is partly due to lack of automation and tool integration, poor data conversion, and (manual) repetition of tasks. One of the pivotal steps that precedes the simulation is mesh generation, the decomposition of the virtual product model into a mesh of simple geometric elements.

The computation time and accuracy of the analysis depend, amongst many factors, on the mesh and the quality of its elements. In general, the use of higher quality meshes, decreases the time spent on analysis.

With more sophisticated algorithms for quality mesh generation coming at our disposal, more CPU time is being spent on meshing. Meshing algorithms that strive to optimise some quality measure on the mesh are often of variational nature, minimising an energy functional related to the quality measure. Some examples are [1-3]. This is a relatively costly operation. Alternatively, an extensive set of heuristics aiming to optimise mesh quality can also take significant time [4].

In this light, we look at the possibility of cutting the time spent on meshing by basing the construction of a new mesh on a previous mesh, which has been used in an earlier design iteration. Iterative improvements to a model, in particular during the latter stages, often have a local scope, i.e. change the geometry in a relatively limited way. If the meshing procedure is computationally expensive, we expect to save time by adapting the previous mesh, instead of meshing the modified model from scratch. Figure 1 gives an example of the evolution of a model. Figure 2 shows how this approach fits into the product design cycle in comparison with the common design cycle. We assume that the models are feature models, as feature modelling is now the prevalent way to develop product models.

This paper is structured as follows. We start with some background on FEA, mesh generation, and feature models in Sect. 2. In Sect. 3 we summarise variational tetrahedral meshing (VTM) for mechanical models, the method on which our remeshing procedure is based. In Sect. 4 we summarise the principal concept behind our remeshing approach, the *feature difference*, which gives the relation between the geometry of the original and the modified feature model. Then, in Sect. 5, we introduce our approach to remeshing, which leans on VTM and the feature difference. In Sect. 6 we describe how, based on the feature difference, node subsets of an earlier generated mesh are copied, and in Sect. 7 how we fill the areas of the model that have remained void of nodes with new nodes. In Sect. 8 we discuss how to complete the new mesh by combining the copied nodes and the new nodes, and efficiently constructing a quality mesh from all these nodes. This is followed by Sect. 9 with a presentation of various results that demonstrate the gain in efficiency and the quality of the meshes. Finally we conclude the paper in Sect. 10.

2 Background

Finite element analysis (FEA) is used to test a product by computationally simulating physical behaviour. It is performed from early on in the design cycle to keep a check on global compliance. Later on the analysis becomes more detailed, as the design gains detail. The results help to steer



Fig. 1 Model modification. a Original model, b modified model: four holes added



Fig. 2 The incorporation of remeshing into the product design cycle with analysis

the design. Not only must the product effectively fulfil its purpose, but also the cost of material and production must be kept low.

FEA can also be used to automatically optimise the shape. In such a case, the shape is determined directly by the outcome of analysis, instead of through an engineer/ designer who interprets the results of the analysis. During shape optimisation, commonly a whole range of calculations is performed, each for a slightly different model.

Two principal measures in analysis are *computation time* and *accuracy*. A certain minimal accuracy is required for most applications. The accuracy of an analysis is generally not known, but can be estimated. A posteriori estimates are more precise than a priori error estimates. The engineer chooses the parameters of the analysis such that he can be reasonably certain that the required accuracy is attained. With the analysis parameters set for a certain accuracy, computational time is effectively fixed. The primary ways to reduce it are better algorithms and more computing power.

In general, increasing the number of mesh elements lowers the discretisation error, which is often the main component of the error in the analysis. It is, however, also known that the shape of individual mesh elements is important for the accuracy, and that the global accuracy strongly depends on the accuracy of the worst element in the mesh [5]. It takes only a few bad elements, or sometimes just one, to spoil an analysis. It is thus essential that the quality of individual mesh elements is looked after.

In general, for a fixed number of nodes, the use of highquality meshes decreases the time spent on analysis. Therefore, a more expensive meshing method can still decrease the total time of analysis if it provides a higher quality mesh. The application of FEA can thus benefit from more efficient mesh creation, which we achieve by means of remeshing.

Another benefit of remeshing over meshing from scratch is that points and connectivity in certain areas can remain identical. This can aid in the comparison between analysis results. If points and elements can be mapped as identical entities between two models, then their values can be compared without interpolation.

The type of models we regard are feature models. This is currently the most common type of CAD model. A feature model is built through the combination of smaller objects (features) with a generic shape that often carry specific semantics within the context of the design. Examples of features are holes, ribs, rings, bumps, etc. See Fig. 1, in which the holes and the pins are features. They can, however, be more complex. For example, if the circular base plate, including holes and pins, would occur similarly in many designs, then this combination could be parameterised as a single feature.

A feature model is modified through the addition and removal of features, and by changing parameter values of features. These values affect the size, shape and location of the features in the model. This is a more high-level approach to creating a model than that of earlier days in design, when manipulation of geometry took more of the designer's time. The features that the designer has at his disposal in feature modelling, allow him to interact more intuitively with the model [6]. The features can carry more information than just geometry, such as semantics and material properties. In advanced feature modelling systems, the additional information is used to actively support the designer in his decisions and to warn him of unintended consequences, e.g. with respect to manufacturability of the product [7].

For our work it is important that the models are *complete* feature models. This means that the resulting model is completely determined by the aggregation of features, and that each feature has its individual geometry instead of implicitly being defined based on possibly non-persistent BRep (Boundary Representation) entities pertaining to other features. Blends in current commercial systems, for instance, are not implemented as features with a geometric definition of their own, but rather are BRep modifying features [8]. We, however, need to track the geometry of *each* feature throughout the design process. If a feature that has a blend attached, is moved together with the blend to a different location in the model, we should be able to relate the geometry of the blend at its new location, to the geometry on its previous location.

In our work, the analysis model is assumed to be a feature model. In current practice the design model and the analysis model are not always the same model. Research is currently being directed at improving the integration of the analysis model with the design model. We assume that the design model is the analysis model, or that changes to the design model can be automatically propagated to the analysis model, analogous to the integration of various views in multiple-view feature modelling [9].

Our aim is to improve the speed of automated quality mesh generation. Tetrahedral elements are popular largely because they are easy to deal with automatically. Our (re)meshing procedure is based on variational tetrahedral meshing [10, 11]. This method generates high-quality meshes by means of an optimisation process that simultaneously takes care of the boundary and the interior of the mesh (see Sect. 3). Our approach to remeshing is not strictly bound to this particular method though.

We look at model modification either in the context of a humanly controlled design cycle or an automatic shape optimisation process. In particular in the latter case, many meshes can be created. One way to avoid remeshing at each step of shape optimisation is to use mesh morphing / deformation, which in principle maintains mesh connectivity by only changing the node positions. Two recent expositions of this approach can be found in [12, 13]. Both work with surface meshes. The latter intends to extend the method to volumetric meshes for the purpose of analysis. This approach is most effective when the changes in the model are subtle. In fact, the range of shapes that can be dealt with by this approach is limited. In particular changes in topology pose a problem.

To our knowledge there are few published works on remeshing models after more general model modification. The best examples we know of are [14, 15]. The first work proposes two strategies of which the first is akin to mesh morphing. Changes in topology cannot be handled this way. The other strategy removes tetrahedrons around modified features and locally reconstructs the mesh for those features. With the first strategy, the quality of the elements decreases with the impact of the modification. The second strategy is only effective for changes with a local geometric scope. The work in [15] offers no single, generic approach, but discusses a range of techniques that can be applied in several different situations. The focus is specifically on parametric model modification. For tetrahedral meshes, the principal ideas are mesh morphing combined with selective quality improvement of bad elements that have appeared. Topological changes are not dealt with here either.

In our opinion it is natural, in the context of efficient remeshing of a feature model, to regard the geometry of the model and the changes therein from the point of view of the features. Instead of relating parts of two models in a global sense, such as by way of the boolean difference of the two models as a whole, we relate the two models on a feature basis. When, for example, a feature is relocated, it can still be identified as the same feature, and as such it relates the geometry pertaining to this feature between the two models [16]. Based on this observation, we have conceived a new, efficient remeshing approach that copies parts of the mesh, exploiting the relations of the features between the two models.

The method we describe can handle more complex shape modifications than the aforementioned approaches, including changes in topology caused by parameter modifications and feature addition/removal. It also tends to conserve larger portions of the mesh. Subtractive features and overlapping features are not an obstacle to our approach. Maintaining the quality of the existing elements and delivering a complete quality mesh as a result are our objectives. To this end we determine exactly how the geometry differs between the two models. This approach has, to our knowledge, not been explored before.

3 Variational tetrahedral meshing of mechanical models

We use variational tetrahedral meshing (VTM) [10] as a basis for our meshing procedure, as it generates meshes with highly regular elements, which increase accuracy and reduce the chance of problems with analysis. Since this algorithm was not intended for meshing mechanical models for finite element analysis, we have made several enhancements, mainly concerned with the accuracy of the representation of the boundary by the mesh, to make the procedure suitable for this task [11]. Figure 3 illustrates the evolution of a mesh during VTM.

We here summarise the working of the algorithm. For a more detailed treatment, we refer to [10, 11]. The VTM algorithm supports mesh grading, but in our remeshing procedure we work with uniformly graded meshes. We thus review the procedure that creates uniformly graded meshes. The algorithm consist roughly of four steps:

- 1. initialisation of data structures
- 2. node distribution
- 3. node optimisation
- 4. mesh extraction

3.1 Initialisation of data structures

An efficient point location test is needed, both for the distribution of nodes and for the extraction of the final mesh from the resulting Delaunay mesh that covers the convex hull of the nodes. For the latter we need to decide for tetrahedrons whether they fall inside or outside the model boundary. A constrained or conforming Delaunay



Fig. 3 Resulting mesh from VTM applied to a nut model. \bf{a} After initialisation, \bf{b} after one optimisation loop, \bf{c} after ten optimisation loops

mesh of the original model is used for this. We call this the control mesh. It accurately represents the boundary, which is important for the overall accuracy of the procedure. Boundary samples, which direct the formation of the boundary, are created and categorised into sets corresponding to the edges or the faces they belong to. These samples are points lying as a fine-mazed net over the boundary of the model, and collectively represent this boundary. They can be created as the nodes of a fine-sampled surface mesh. The samples are also referred to as *quadrature samples*.

3.2 Node distribution

The requested number of nodes is spread out roughly uniformly over the model. This is done by iterating over the cells of a grid that covers the bounding box of the model. In a first iteration, the number of cells that have their center inside the control mesh, is counted. Based on this number, in a second iteration, a fair proportion of the nodes are placed randomly inside those grid cells. After this process we end up with a cloud of nodes that covers approximately the volume of the original model, and has a more or less uniform density. Figure 3a shows an example of what the mesh is like at this stage.

3.3 Node optimisation

During the optimisation process, the nodes fall in two categories: boundary nodes and interior nodes. Every node starts as an interior node, but can become a boundary node when it is selected as such during the determination and repositioning of the boundary nodes. Each iteration of the optimisation loop starts with the identification and positioning of the boundary nodes. After that, the rest of the nodes, deemed part of the interior set, is optimised.

So at the start of each iteration it is determined which nodes are part of the boundary. These boundary nodes are then (re)positioned, aiming for balanced node spacing on the boundary. This is achieved by employing the quadrature samples.

For each quadrature sample, we locate its nearest node and have the sample exert a virtual pull on that node equal to the area that the sample covers. This is the weight of the pull or quadrature value of the sample. The nodes that have at least one sample pulling on them, are now considered part of the boundary set. They are moved to the average location of the pulling samples, weighted by the quadrature values. The rest of the nodes belongs to the interior.

To ensure that nodes end up at the edges and at the vertices, we treat their samples differently from the surface samples. The quadrature value for samples on edges is set to the length that the sample covers, and to vertex samples an infinite quadrature value is assigned, to ensure the assignment of nodes there. The procedure starts with the regular surface samples pulling in and repositioning nodes, then the edges and finally the vertices are taken care of.

If a node is being pulled on by boundary samples from different sets, we can decide to split this node, as this situation might indicate a local lack of nodes to accurately represent the boundary. Splitting such a node into two nodes, helps in ensuring that the final mesh represents the boundary correctly. For a more detailed motivation, see [11].

After dealing with the boundary, the Delaunay mesh is reconstructed as nodes have moved. Then each interior node, i.e. each node x_i that was not selected as a boundary node, is moved according to the formula:

$$\mathbf{x}_i^{\text{new}} = \frac{1}{|\Omega_i|} \sum_{T_j \in \Omega_i} |T_j| \mathbf{c}_j \tag{1}$$

Here Ω_i denotes the one-ring of tetrahedrons that share node x_i , $|\Omega_i|$ represents its volume, $|T_i|$ denotes the volume of tetrahedron T_j , and \mathbf{c}_j is the circumcenter of tetrahedron T_j . The effect of this relocation is similar in idea to the relocation of a node towards the center of its Voronoi cell. Instead of optimising the compactness of the Voronoi cell, this operation aims at improving the compactness of the tetrahedrons in the one-ring around the node. For a more detailed motivation, we refer to [3]. If the new location would invalidate the Delaunay property of the mesh, then the connectivity is changed to keep it a Delaunay mesh.

The optimisation loop alternates between these two phases of (1) determining and repositioning the boundary nodes, and (2) optimising the location of the interior nodes. Either a fixed number of iterations is performed, or a condition on the evolution of the quality improvement is used.

3.4 Mesh extraction

After the node optimisation, the mesh representing the model has to be extracted from the resulting Delaunay tetrahedrisation, which covers the convex hull of the nodes. As the model is usually not convex, it needs to be decided which tetrahedrons contribute to the model, i.e. are inside, and which tetrahedrons fall outside the model. This process is called *peeling*, as it can be envisioned as the removal of tetrahedrons that are outside the mesh that represents the real model boundary. It is worthwhile to ask why this is possible in the first place. Normally, the Delaunay tetrahedrisation of a node set of a model does not contain the complete boundary; some faces and edges have to be recovered. Why can we expect the model boundary to be present in the tetrahedrisation after the optimisation procedure in VTM?

There are no theoretical guarantees that the boundary will be present, but with the quadrature samples being finer, the chance of success increases; see Fig. 4. If a node encroaches upon the minimal circumsphere defined by two adjacent nodes on the same boundary edge, that node is likely to be drawn to the boundary, since a sample near the center of the edge between the two nodes will most likely have the encroaching node as its nearest node. This procedure of pulling the node closest to a boundary sample to the boundary, aims for a distribution of nodes over the edges, such that for each point on an edge, the closest node lies on that edge. This, in turn, means that the minimal circumsphere of any two adjacent nodes on an edge, will be empty of other nodes. A segment between two such nodes is called Gabriel, and it is guaranteed to be present in the Delaunay mesh [17]. We expect that the edges can be (almost) completely covered by Gabriel segments, and thus that all edges from the model are found in the Delaunay mesh. Segments that are not Gabriel, are not guaranteed to be in the mesh, but it is still highly likely that they are. The



Fig. 4 Encroaching of part of edge

reasoning holds similarly for the presence of triangles that can represent the faces of the model. The procedure might, however, fail near small angles.

With the expectation that an accurate representation of the boundary is present, the final mesh can easily be extracted from the Delaunay mesh. The nodes are already divided into boundary and interior nodes. Any tetrahedron that has at least one interior node at its vertices, is clearly inside. The remaining tetrahedrons all have four boundary nodes. If the centroid of such a tetrahedron falls inside the control mesh, then we consider it part of the final mesh. All other tetrahedrons are outside and thus removed. A more concise description and motivation of this procedure is given in [11].

4 The difference between two feature models

For the remesh procedure, a description is needed of the difference between two models, such as the models in Fig. 1. The model that was made first is being referred to as the original model. When the original model has been adapted, the resulting model is being referred to as the modified model. Our goal is to determine for each section of the geometry of the modified model, both of BRep elements and the volume, whether it relates to some section of the previous model or not. The sections that can be related might carry the same mesh, whereas for the remaining geometry new mesh elements have to be constructed. Of course, if we reuse a mesh subset for a section of the model, it must be connected to other subsets of the mesh, either also reused or newly constructed. We must take specific care that the mesh quality in those regions is on par with the overall quality.

Our approach to remeshing builds on the concept of feature difference [16]. The approach distinguishes itself from other methods for model comparison. The majority deals exclusively with the BRep, whereas the feature difference explicitly maps sections of volume between the original model and the modified model as well. We need this for the purpose of copying mesh elements between corresponding sections of volume. Also many of the existing methods lack the geometric precision and completeness that we need.

The feature difference essentially describes how two feature models differ from each other from the point of view of the individual features. This is another significant difference with the conventional approaches to model comparison, where there is only a single point of view for the comparison, namely the complete model. With a single point of view, every geometric element only has a single interpretation, i.e. related to the other model or not related. In our approach, a geometric element can be related to the other model from the point of view of one feature, whereas from the point of view of another feature there is no relation. Multiple points of view only occur in the feature difference where features overlap each other.

We will now discuss the concepts and data models that are used to accomplish our goal. We start with a review of the cellular model, which is the data structure that is used for storing and maintaining the feature models, in Sect. 4.1. Next, in Sect. 4.2, comes a review of the feature difference concept. This essentially describes for each individual feature how the model has changed from its particular point of view. We also motivate the relevance of this concept for the application of remeshing. Lastly, in Sect. 4.3, we show how the set of all feature differences, which is referred to as the *difference model*, relates to both the original and the modified model, as it can be used to describe either of them and to deduce geometrical relations between them. Such deductions supply the information needed for our remeshing procedure.

4.1 The cellular model

Basically, a cellular model, in contrast to BRep models, stores the complete geometry of all the features that compose the model. The information stored in the cellular model can be used for validity maintenance, visualisation of functional information, and multiple-view modelling [9]. Features that intersect with other features, are divided into cells. A feature that covers a cell, is said to *own* that cell. Each cell is owned by at least one feature; where features overlap, the cell has multiple owners. The cells that contribute material to the model are said to have *positive* nature, whereas the other cells have *negative* nature. Figure 5 shows a simple cellular model. In this figure, there are three features: a base block, and two slots. For each cell with negative nature it is indicated which feature owns it.

The cellular model allows features to be added and removed, while keeping track of the feature ownership for all cells and faces. To keep track of the cell geometry, and ownership, we partly rely on the *cellular topology* component of ACIS [18]. A thorough description and discussion of the cellular model can be found in [19].



Fig. 5 Example of a cellular model

We use cellular models to store and query the original and the modified feature model. Also, we copy subsets of these cellular models to other cellular models for the construction of the feature difference.

4.2 The feature difference

The analysis mesh conforms to the geometry of the model, which in turn depends on the features. Changes in the geometry result from manipulation of features. The mesh thus depends indirectly on the geometry of the features.

If, after a model modification, the geometry of an additive feature, including its interactions with other features, has not changed, then the mesh corresponding to that feature does not need to change either, and can thus be reused in the mesh for the modified model. See, for example, Figure 6, which shows the meshes of two variants of a model with a base block feature and a rib feature, including two cylindrical protrusions, on top: the mesh section of the rib feature on top could be identical in both meshes, as the geometry of this feature is identical in both models. The geometry of the base block, on the other hand, is subtly different between the two models, as its top surface connects in different locations to the rib feature (see Figs. 6, 7). Therefore, the mesh corresponding to the base block feature cannot be identical for both models. However, as the difference is subtle, the meshes could for the larger part still be the same. For this we need a description of how the geometry of the base blocks differs between the two models.

If we know for each feature whether its geometry has remained the same, or, alternatively, how it is different from its original geometry, we have enough information to reuse sections of the original mesh in the mesh for the modified model. All details of the remeshing procedure are discussed in Sect. 5 and onward. We proceed here with the description of how the geometry of a feature relates between two models. This description can be constructed for any feature of the two models, and is called the *feature difference*.



Fig. 6 Original and modified model and their meshes. **a** Original model: base block with a rib feature on top, **b** mesh of original model, **c** modified model: base block with rib feature translated, **d** mesh of modified model



Fig. 7 Subtle difference in geometry of base block. \mathbf{a} Geometry of base block in original model, \mathbf{b} geometry of base block in modified model

An aspect important to the concept of feature difference is that a feature, placed into a model, is affected by its interaction with other features. One of the clearest examples hereof is the interaction of features having different nature (adding material or removing material): when a slot feature is added to a block, then the geometry of the block changes, as does the volume the block occupies. This is illustrated in Fig. 8. More subtle is the change when an additive feature is attached to another additive feature. This was already illustrated in Fig. 7. When looking at the geometry of a feature in a particular model, we also take this additional geometry into account.

We now explain in some more detail what the feature difference is. For a particular feature, it is the comparison of the feature as it was in the original model, with the



Fig. 8 Block feature interacting with a slot feature. **a** Cellular model of block feature (single cell), **b** block feature with a slot feature, **c** cellular model of block feature resulting from interaction with slot feature (two cells)

feature as it is in the modified model. This comparison includes the whole of the geometry of the feature, as it is stored in the cellular models, which is more than what appears in a BRep. In a BRep only those elements appear that are part of the boundary of the model, whereas in the cellular model the complete boundary of each individual feature is stored. We refer to those elements that are part of the cellular model, but do not appear in the BRep, as nonboundary geometry. The feature difference is thus defined for each feature, and consists of both BRep and nonboundary geometry of the feature as part of the original model, combined with the corresponding geometric information of the feature as part of the modified model. Additionally, the structure consists of cells that represent regions of volume. The feature difference relates the complete geometric information of a single feature, including geometry emerged from interaction with other features, between the two models. The collection of all feature differences is called the *difference model*. Since features overlap-if not with their volumes, then with their faces-some geometric elements are part of the feature difference for more than one feature. Dependent on the point of view of an involved feature, it can vary how that element is interpreted (as unchanged geometry, new or old). The following example will illustrate the feature difference.

In Fig. 9a there are two models. On the left is the original model and on the right is the modified model. The original model consists of two features and the modified model of three. In Fig. 9b we see the feature difference for each of the features. For the purpose of remeshing, we



Fig. 9 Difference model. **a** Original model (*left*) and modified model (*right*), **b** difference model, consisting of the feature difference for all three features

discern four classifications of geometry in the feature difference: (1) persistent-identical, (2) persistent-different, (3) new, and (4) old. Persistent-identical geometry is, from the point of view of a particular feature, identical in every aspect in both models. The complete geometry of feature F_2 is an example of this, as both its shape and its interactions with other features are the same in the original and the modified model. Feature F_3 is an example of new geometry, as it has been added to the model. Feature F_1 carries an example of persistent-different geometry: the separation between feature F_1 and feature F_3 already existed as part of the BRep in the original model. In the modified model, however, it is not part of the BRep, but it does exist in the cellular model, since it is part of the geometry of the individual features. We classify this in the feature difference as persistent-different geometry, which thus indicates that the element was part of the BRep in one of the two models, but not in the other. The vertices at the ends of the persistent-different geometry are new to feature F_1 , as they were not present in the original model. Table 1

 Table 1
 Classification of the feature difference for faces, edges and vertices

orig.	modif.	Classification
b	b	Persistent-identical: boundary
n	n	Persistent-identical: non-boundary
n	b	Persistent-different: modified to boundary
b	n	Persistent-different: modified to non-boundary
	b	New: boundary
	n	New: non-boundary
b		Old: boundary
n		Old: non-boundary

b = boundary geometry, n = non-boundary geometry, ' ' = not in model

Table 2 Classification of the feature difference for cells

orig.	modif.	Classification
+	+	Persistent-identical: positive nature
_	-	Persistent-identical: negative nature
_	+	Persistent-different: modified to pos. nature
+	-	Persistent-different: modified to neg. nature
	+	New: positive nature
	-	New: negative nature
+		Old: positive nature
-		Old: negative nature

'+' = positive nature, '-' = negative nature, ' ' = not in model

lists all possible classifications of geometry that can be encountered in the feature difference for faces, edges and vertices, whereas Table 2 does this for cells. Except for the terminology, the two classification schemes are very similar, as the 'boundary/non-boundary' and 'positive/negative nature' classifications fulfil analogous roles. The distinction in the names helps to reason more intuitively with the concepts.

The feature differences from Fig. 9 are easy to construct: take all the geometry from the cellular model that is owned by a particular feature for both the original and the modified model, and then overlay and merge these geometries with a non-regular union operation, to find the classification of the elements. See [16] for more details.

In Fig. 9, none of the individual features change shape. The case that features do change shape is essentially handled in the same way, only now with the geometries of the differently shaped features being overlaid. This is illustrated by feature F_2 in the difference model of Fig. 10. The feature difference for feature F_2 may seem surprising. In overlaying the geometries, the feature center has been used as a point of reference to align on. The reasons for this are



Fig. 10 Difference model including reshaped feature. **a** Original model (*left*) and modified model (*right*), **b** difference model, consisting of the feature difference for both features

detailed in [16]. In short: the preference for a particular point of reference is subjective, and tends to vary with the context. This is particularly the case when there are dependent and interacting features.

We have here explained only the essence of the feature difference; it is discussed in detail in [16].

4.3 Relation between the feature differences and the two models

The feature difference for a particular feature contains the combined geometrical description of the feature for the two models that are being compared. All the geometric elements carry a feature difference classification (see Table 1), and a list of the features that own the element. From the classification it can be inferred to which model(s) an element belongs. This is enough information to reconstruct the geometry of a feature in either the original or the modified model from the feature difference. For example, from the feature difference F_2 in Fig. 10b, we can recover the geometry of the feature in the original model, by removing all geometric elements classified as new; the geometry of the feature in the modified model can be obtained by removing the old elements.

Each feature in a model is associated with a transformation, which relates to a global point of reference by which the positions of all features are linked in the model's cellular model. We regard the transformation as part of the feature parameterisation. When the feature parameterisation is known for all features in a model, then its cellular model can be (re)constructed.

It follows from the previous two paragraphs that from the feature difference, both the original and the modified model might be obtained. Of course we do not actually do this, as we already have those models, but in a similar vein we can also combine the feature differences applying the feature transformations of either the original model or the modified model. Figure 11 shows a simple model modification (a and b) and its corresponding features differences (c-e). The model consists of three features: a base block, a through hole and a rib. Figure 12 shows how the three feature differences can be combined into two different structures: (a) the *combined original model* (applying the positioning of the features in the original model), and (b) the *combined modified model* (applying the positioning of the features in the modified model). Notice how the two structures differ. For the combined original model, the feature difference for the through hole and the rib are positioned w.r.t. the base block such that the original feature geometry lines up. For the combined modified model, the feature differences are positioned such that the modified feature geometry lines up.

simple model, for which the

and relocated, and the rib enlarged. a Original model, **b** modified model, **c** feature difference for base block.





Fig. 12 Combining feature differences according to the feature configuration of either the original or the modified model. a Combined original model, b combined modified model

A combined model captures the interaction of feature differences, and by this means enables us to relate the modified geometry to the original geometry, or vice versa. By selecting the right subset of cells from the model in Fig. 12b, we can construct the model of Fig. 11b. To each of these cells, at least one difference classification is associated, e.g. the feature differences for the through hole and the base block both carry classifications that end up at cells that overlap in the combined model. From the difference classifications, we know whether the geometry is persistent (or not) and thus can be related to geometry of the original model (or not). Through these relations we find the mesh subsets to copy from the mesh of the original model, the areas where new mesh needs to be constructed, and all other areas where attention to the mesh is required, e.g. places where mesh sections of different origin meet.

The geometry of a combined model can get fairly complex as the feature differences, which already contain the geometry of two different versions of a feature, can intersect once more amongst each other. By inspecting this structure, we can find the information that the remeshing procedure needs.

Before delving into the details of using the information in the combined model, we take a step back, and discuss the overall remeshing procedure.

5 The remeshing procedure

The remeshing procedure is conceptually simple. We wish to generate a mesh for a model that is a modified version of an earlier design model (the modified and the original model respectively). Part of this mesh is going to be supplied by the mesh of the original model. The procedure for this is:

- analyse the difference between the two models 1.
- 2. initialise the new mesh with all mesh sections that can be copied
- 3. construct new mesh elements in remaining void areas
- 4. perform VTM with an appropriate subset of the nodes and boundary samples.

This procedure is schematically illustrated in Fig. 13.

The first step consists of the construction of the feature difference and the combined modified model, which have been discussed in Sect. 4.

In the second step, we map sections of the previous mesh to the new model. The idea is that we determine from the combined modified model which (parts of) features from the original model can serve as a mesh source. Here





we aim to copy large and continuous mesh sections, such that extra work to improve connections between separately copied mesh sections is minimised. Any section of the new model that is not assigned a mesh by copying, remains without mesh.

In the third step, we fill up these voids. We now have the basis for the new mesh.

In the fourth step, the VTM algorithm is executed, with the result of the previous two steps as initialisation of the mesh. Since copied sections of the mesh do not need optimisation, we adapt the VTM algorithm to work only with a subset of the nodes and the boundary samples. The attention is focussed on the new sections and the other places were the mesh needs improvement, such as between adjacent mesh sections where a good connection is lacking.

Until now we have consistently talked about copying "sections of the mesh". Since the VTM algorithm generates a Delaunay mesh, we can suffice with copying just the mesh nodes, as the connectivity is handled in the VTM algorithm by the Delaunay criterium. This makes the procedure considerably easier to implement. The input for the VTM algorithm will be the set of initial nodes to start with, accompanied by a flag that indicates which nodes are *free* and thus are in need of optimisation. The new nodes and the nodes in areas of transition between copied mesh subsets, are the free nodes. The other nodes are *fixed*. Their relative positioning can remain untouched. We now continue to discuss the procedure for copying nodes.

6 Copying mesh nodes

The idea behind the copying procedure is that the modified model can receive nodes through the relation of persistent feature volume with the original model. Each feature whose feature difference contains some persistent volume of positive nature, can copy nodes that belong to that volume in the original model, to the modified model.

Since features can overlap, we cannot simply copy the nodes for each cell of persistent volume in each feature. Where features overlap in volume, too many nodes would be assigned. Also, we want the copied nodes to form large and continuous sections, such that the need for optimisation between copied sections is minimised. To this end, we sort the features by the size of their persistent-identical volume of positive nature, and start with copying nodes from the feature for which this volume is largest. Next comes the feature with the second largest persistent volume, etc. The complete procedure is as follows:

1. assign nodes to features in the original model

- 2. find cells with persistent volume of positive nature (*copycells*) and cells with new volume (*newcells*) in the combined model for the modified model
- determine which features correspond to which copycells and calculate for each feature the volume of copycells that it covers
- 4. copy nodes of features to empty copycells, starting with the feature with the largest volume to copy.

The last step is finished when all copycells have nodes copied to them. Due to overlap of features, it is not required that all features have contributed nodes at the end of this step.

6.1 Assign nodes to features in the original model

This step is independent of the modified model, and as such it is not strictly part of the remeshing procedure. It can be performed long in advance.

The original model is stored as a cellular model. All nodes of the corresponding mesh are assigned to cells of the cellular model. To speed up the operation, the nodes are first tested for inclusion with the bounding boxes of the cells. Then, for each node on or inside a bounding box, an accurate inclusion test (an internal function of ACIS) with the corresponding cell is performed. A few nodes might not be assigned to any cells at all, due to tiny differences between the coordinates of the mesh and the geometry of the model. In such a case the projection distance of the node to the cells is calculated and the node is assigned to at least one cell.

For each cell it is known to which features it corresponds. After the assignment of the nodes to the cells, we can thus retrieve all nodes inside or on the boundary of each feature.

6.2 Determine copycells and newcells

In the combined modified model (cf. Fig. 12b), all cells have classifications attached, indicating for each feature that (co)owns the cell how its geometry relates to the original model (persistent-identical, persistent-different, new or old). This classification can differ between the owning features. We limit our attention to the cells of positive nature that contribute to the representation of the modified model, since these are the cells that need to be covered by mesh elements. Of those cells, we mark each cell that can be regarded as a persistent volume, according to at least one classification, as a copycell. The volume of such a cell can be related, in at least one way, to the original model, as the persistent classification indicates that a similar counterpart exists in the original model. All remaining cells of positive nature belong to the set of newcells. These also need to be covered by mesh elements, but they cannot be provided by the mesh of the original model.

6.3 Create map from features to copycells

At this point, we know which cells in the combined modified model are copycells and which nodes are contained in each cell of the original model. Unfortunately, there is no one-to-one map between the cells in the modified combined model and those in the original model. A cell from the original model can relate to multiple cells in the modified combined model. It should be possible to identify those relations, but instead we have opted for another approach.

Since we want to copy sets of cells that are as much as possible adjacent, to avoid having to perform mesh improvements between nodes copied from different origins, we instead copy the nodes on a feature by feature basis. All nodes copied from a single feature, obviously have the same, consistent origin, meaning that the relative positions of these nodes is preserved. An additional benefit of this approach is that we avoid the need to explicitly establish relations between the copycells and the cells of the original model.

We want to start copying nodes from the feature that occupies the largest volume of copycells. We thus need to calculate the total copy volume for each feature. This calculation is combined with the creation of a map that relates the features to their copycells.

6.4 Copy nodes from features to copycells

Starting with the feature that has the largest persistent volume, the nodes are copied to the corresponding copycells. For efficient copying, the copycells are joined into a single body (which can consist of multiple lumps). Then for each node in the feature, an inclusion test is performed against the copycell body. The nodes that belong to the body are included in the mesh of the new model. Before performing the inclusion test, the coordinates of the nodes have to be transformed from their location in the original model to their location in the modified model. This transformation is the compound of the inverse transformation of the feature's position in the original model and the transformation that gives the position in the modified model. We avoid copying multiple times to the same copycell, by keeping a list of copycells that have been taken care of.

Once nodes have been copied to each copycell, the model still lacks nodes in those regions of the model that could not be mapped to the original model, i.e. the new-cells. See for an example Fig. 15a, where the copied nodes are shown for remeshing the model of Fig. 14b based on the mesh of Fig. 14a. At the original location of the four hole features, there were no nodes to be copied to the mesh of the modified model. These regions are covered by newcells, and these need to be filled by nodes as well (see Fig. 15b). This is elaborated in the next section.

7 Adding new nodes, and the free/fixed distinction

The previous section dealt with filling the set of copycells, coming from the combined model, with nodes from the mesh that corresponds to the original model. The newcells set comprises those cells that have positive nature, but cannot receive nodes from the original model. This is either due to a newly added feature, an enlarged feature or the removal/translation of a feature with subtractive nature.



Fig. 14 Original and modified model (tool1-a and tool1-b). a Original model, including *top view*, b modified model, including *top view*: four holes translated



Fig. 15 Top view of copied and new nodes for the original and modified model of Fig. 14. a Copied nodes, b new nodes

The procedure for filling the set of newcells with nodes is the same as the node initialisation procedure of standard VTM [10]. Before the nodes are spread out, the average node density in the earlier mesh is measured. Then the cells of a grid that covers the newcells volume are traversed, to calculate the average number of nodes that each gridcell should receive. Only gridcells of which the center lies inside the newcell volume are counted. Finally the gridcells are traversed once more in serptentine order, this time to spread out the nodes. Nodes are placed randomly in the gridcell. If a gridcell requires a non-integer number of nodes, then the non-integer part is added to the number of nodes for the next gridcell. This way, the average node density in the newcells matches well with the required node density. Figure 15b shows an example of a set a new nodes, with in the background the copied nodes.

We now have all the nodes that are to be used for the initialisation of the new mesh. However, to efficiently process these nodes, the nodes need to be divided into a *fixed* and a *free* set. The free nodes will be actively involved in the optimisation process, whereas the fixed nodes will be left untouched. It is clear that the new nodes belong to the free set. The majority of the copied nodes should belong to the fixed set—otherwise there would be little gain in remeshing—but not all of them.

There are two cases that warrant attention to copied nodes:

- 1. adjacent cells that have nodes copied to them from different origins
- 2. persistent-different faces on copycells.

Both cases are illustrated by the example of Figs. 6 and 7: (1) in the mesh of the modified model, the nodes of the base block do not match with the nodes of the rib feature on top, as the node sets connect at a place different from before; (2) the nodes of the base block near the place where the rib feature was previously connected, do not properly represent the surface; previously those nodes were in the vicinity of a non-boundary face, whereas in the modified model that face does represent boundary.

The faces from the cellular model affected by the first case are determined during the node copy operation. If the copied nodes in two adjacent copycells are transfered from their original coordinates by the same coordinate transformation, then they have the same origin; otherwise, attention is required to assure the quality of the mesh around the separating face.

The second case can be inferred from the combined modified model: for each face of a copycell it is determined (1) whether it has nodes lying in that face, i.e. the face was part of the BRep of the original model, and (2) whether nodes need to lie in that face, i.e. the face is part of the BRep of the modified model. If these two results are different, then the nodes in and near that cell face need to be in the set of free nodes. This can be inferred by looking at the classifications of the faces, from the point of view of the features that supplied the nodes on either side of the face.

We call the faces from the combined modified model that warrant attention for the nodes in their vicinity, *active faces*. For each active face, it is determined which nodes in its vicinity have to be transferred to the set of free nodes. This is done by means of regularly spaced sample points on the surface of the face. Each sample locates its nearest node. All nodes that are nearest to a sample, become part of the free set.



Fig. 16 Top view of new, active and expanded active nodes, for the models of Fig. 14. **a** New nodes, **b** free nodes, **c** expanded free nodes, **d** expanded free nodes; rotated view

Fig. 17 Six cases of model modification. a simple-a and simple-b, \blacktriangleright b tool1-a and tool1-b, c tool1-a and tool1-c, d tool2-a and tool2-b, e tool2-b and tool2-c, f tool3-a and tool3-b

8 Efficiently constructing the quality mesh

The meshing procedure is basically the same as the one described in [11], but adapted to avoid unnecessary computations. Instead of following the procedure for creating the initial node distribution, we use the nodes as determined by the procedures described in Sects. 6 and 7.

These nodes have been divided in two sets, fixed and free. In our optimisation procedure we only handle the free nodes and leave the fixed nodes untouched. However, nodes from the fixed set can be transfered to the free set. This happens twice during the optimisation, as we transfer the layer of fixed nodes adjacent to the current free set, to the free set. The underlying idea of this expansion of the set of free nodes, is to give the optimisation procedure more freedom to achieve a quality connection between the free and fixed nodes. It is done once directly after the first iteration of the optimisation loop, and three iterations later a second time. The delay in adding the second layer is to first give the smaller set of free nodes an opportunity to settle a bit, as the largest variations in node locations occur during the first couple of iterations. Once the extent of the changes has diminished, we expand the set of free nodes one more time. Figure 16 compares the new nodes, the free nodes at initialisation, and the free nodes after the first expansion of the free set. The final set of free nodes covers a sizeable region, compared to the initial set of free nodes. This is necessary to uphold the quality of the final mesh. When expanding the free node set, we must take care to keep the expansion local, as the Delaunay mesh covering the convex hull also connects nodes that are not adjacent in the final geometry. The risk of expanding the free set to another section of the model occurs nearly always when expanding from a boundary node to a boundary node. For that case we have added the precondition that the two nodes share a connection to an internal node.

The other reduction in computational cost is achieved by adapting the boundary procedure. Normally all boundary samples look for their closest node, but since a large part of the mesh does not change at all, this would be unnecessary work. Instead we only do this once for all boundary samples at the start. Those boundary samples that have a free node as their closest node are added to the set of *active* samples, the rest is *non-active*. The samples adjacent to an active sample are placed in the set of *border-active* samples. Only the active and the border-active samples are used in the boundary procedure to pull on nodes, to possibly change their positions. Most of the time the borderactive samples will have a fixed node as their closest node,



Table 3 Runtime measurements comparing the regular meshing procedure with the remeshing procedure for the six cases of model modification

Model	#samples	#nodes	#iter	t_1	t_2	<i>t</i> ₃	total ₁	t_4	<i>t</i> ₅	<i>t</i> ₆	<i>t</i> ₇	<i>t</i> ₈	total ₂	% time
simple	56,232	15,058-15,062	5	24.85	835.32	3.12	863.29	0.10	5.07	28.26	54.88	3.44	91.74	10.62
simple	56,232	15,058-15,062	10	25.01	1606.25	3.11	1634.37	0.10	5.03	29.82	125.47	3.38	163.80	10.02
simple	111,532	30,087-30,087	5	87.54	3051.84	6.47	3145.85	0.10	9.65	77.10	142.30	7.18	236.34	7.51
simple	111,532	30,088-30,088	10	87.37	5936.73	6.32	6030.42	0.10	9.75	79.93	336.41	7.01	433.20	7.18
tool1-ab	89,290	15,195–15,199	5	42.90	422.75	6.80	472.46	2.01	12.51	45.19	23.94	7.32	90.97	19.25
tool1-ab	89,290	15,195–15,199	10	43.07	882.48	6.81	932.37	1.97	12.46	45.39	46.55	7.24	113.61	12.18
tool1-ab	165,814	30,341-30,345	5	84.99	1437.65	12.69	1535.33	1.99	24.11	88.03	49.74	13.67	177.54	11.56
tool1-ab	165,814	30,341-30,345	10	85.74	2772.13	12.50	2870.37	2.00	24.08	89.09	111.83	13.57	240.57	8.38
tool1-ac	87,352	14,997–15,005	5	41.46	422.27	5.67	469.40	2.34	12.91	44.30	42.42	6.10	108.08	23.02
tool1-ac	87,352	14,999–15,007	10	41.50	875.08	5.71	922.29	2.33	12.89	44.49	100.02	6.10	165.83	17.98
tool1-ac	163,136	30,089-30,097	5	82.97	1434.56	10.88	1528.41	2.37	25.68	89.24	115.09	11.55	243.93	15.95
tool1-ac	163,136	30,075-30,083	10	83.59	2754.29	10.75	2848.63	2.28	25.66	87.24	252.19	11.62	378.99	13.30
tool2-ab	121,788	15,637–15,649	5	28.65	161.59	6.65	196.89	1.40	9.01	38.88	22.18	6.39	77.86	39.54
tool2-ab	121,788	15,637–15,649	10	28.83	297.40	6.53	332.76	1.35	8.98	38.97	42.45	6.35	98.10	29.48
tool2-ab	242,711	31,275–31,287	5	65.89	427.40	12.48	505.77	1.39	16.26	86.78	45.64	11.57	161.64	31.95
tool2-ab	242,711	31,275–31,287	10	67.10	795.11	12.41	874.61	1.38	15.99	87.88	85.55	11.39	202.19	23.11
tool2-bc	117,216	15,047-15,055	5	35.81	149.42	10.12	195.35	2.32	9.42	40.74	21.51	9.59	83.57	42.77
tool2-bc	117,216	15,038-15,046	10	35.91	281.37	10.00	327.27	2.30	9.23	40.48	42.23	9.66	103.91	31.75
tool2-bc	233,187	30,100-30,108	5	70.66	427.90	16.59	515.15	2.32	16.59	81.27	37.69	16.33	154.20	29.93
tool2-bc	232,111	30,087-30,095	10	70.84	801.57	16.76	889.17	2.36	16.81	81.12	73.78	16.01	190.08	21.37
tool3	51,818	10,248–10,272	5	12.31	178.98	2.53	193.81	1.77	8.62	17.88	52.57	3.03	83.86	43.26
tool3	51,580	10,194–10,218	10	12.40	315.54	2.46	330.39	1.79	8.56	18.12	113.31	2.95	144.72	43.80
tool3	101,220	20,480-20,504	5	27.70	499.90	4.99	532.60	1.76	16.67	37.34	101.07	5.87	162.72	30.55
tool3	101,216	20,420-20,444	10	28.28	951.69	4.94	984.91	1.76	16.84	37.26	210.09	5.82	271.79	27.59

#samples: number of boundary samples, #nodes: number of nodes in the mesh for the modified model after meshing and after remeshing respectively, #iter: number of iterations of the optimisation loop. All times are in seconds. Columns $t_1 - \text{total}_1$ correspond to the regular meshing procedure, $t_4 - \text{total}_2$ to the remeshing. t_1 and t_6 : setup, t_2 and t_7 : optimisation loop, t_3 and t_8 : mesh extraction and other post-processing, t_4 : setup and construction of feature difference, t_5 : construction of combined model, copying and creation of nodes, and analysis indicating free/fixed nodes. total $_1 = t_1 + t_2 + t_3$, total $_2 = t_4 + t_5 + t_6 + t_7 + t_8$, % time = 100.0 total/total $_1$

but when it does have a free node closest, then this borderactive sample is transferred to the set of active samples and its non-active adjacent samples turn to border-active samples. This way we remain confident that the boundary samples are doing their work where needed, but no more than that. When the set of free nodes is expanded, the set of active samples is expanded accordingly.

Summarising, the new meshing procedure is as follows:

- 1. initialise data structures
- 2. initialise the mesh with the free and fixed nodes
- 3. optimisation loop:
 - use adjusted boundary procedure based on only the active samples
 - optimise node positions of free nodes
 - if iteration-step = 1 or iteration-step = 4: expand set of free nodes and set of active samples
- 4. extract mesh.

We will now show some of the results that are achieved by the complete remeshing procedure.

9 Results and discussion

To study and compare the effectiveness of our remeshing procedure, we present six cases of model modification, all shown in Fig. 17:

- (a) *simple* translation of block on top
- (b) tool1-ab enlargement of pins
- (c) *tool1-ac* translation of holes
- (d) tool2-ab addition of stiffener
- (e) *tool2-bc* translation of pipe
- (f) tool3 reparameterisation of base block

These cases cover a diverse range of situations encountered in model modification. The first case (a) is a Fig. 18 Quality comparisons between result of regular meshing (*darker*, *blue*) and remeshing (*lighter*, *yellow*). The histograms overlap almost completely; the small differences are visible on the top of the bars. a tool1-ac, 10 iterations with ~15,000 nodes; b tool3, 10 iterations with ~15,000 nodes; c tool2-ab, 5 iterations with ~30,000 nodes

simple example that serves as a reference. The second (b) case demonstrates a change where only reparameterised features are involved. The third case (c) involves the creation of new holes and filling the remaining voids. The fourth case (d) demonstrates the addition of a new feature and its interaction with existing features. The fifth case (e) deals with changes in feature interaction and topology. Finally, case (f) demonstrates the capability of the approach to even handle changes in the shape of the base feature. We do not consider this last case to be typical for the application of remeshing.

The first aspect we consider is runtime. The results are shown in Table 3. We compare the time that it takes to completely mesh the modified model, with the time needed for the remeshing procedure. Because the final number of nodes of a remeshed model is hard to control, we first performed the remeshing in our experiments. After this, the regular meshing procedure was performed, aiming for the same number of nodes. Since occasionally nodes can be removed or added during the algorithm, this match is not perfect. The column #nodes first lists the number of nodes in the final mesh for the regular meshing procedure and then the number of nodes in the remesing result. We have roughly aimed at 15,000 or 30,000 nodes. The number of boundary samples is, by means of a heuristic formula, roughly aimed to be eight times the number of nodes that ends up on the boundary. This is a reasonable lower bound for this ratio, since in most cases one would prefer to use more boundary samples per boundary node. The ACIS faceter component is used to generate the samples. Each experiment is performed with either five or ten iterations. The final column list the runtime of the remeshing procedure as a percentage of the runtime of the regular meshing procedure.

We observe that in general:

- By remeshing, the runtime is brought down to between 10 and 45% of the time for the regular meshing procedure.
- For a larger number of nodes, the efficiency gain is higher. In such cases, the percentage of internal nodes is higher, and since the optimisation of the internal nodes takes the bulk of the cpu-time, the realised savings by remeshing are higher. A secondary factor is that for a higher number of nodes, the expansion of the free node set by two layers affects a smaller percentage of the nodes.
- The optimisation loop scales roughly with the number of iterations.



• The time spent analysing the models and copying the nodes (*t*₄ and *t*₅) is smaller than the setup time of the standard meshing procedure (*t*₁).

We measure the quality of the meshes by calculating the volume-length ratio of each tetrahedron, and display these values in a histogram. The measure is defined as $V/l_{\rm rms}^3$, with V the volume of the tetrahedron and $l_{\rm rms}$ the root mean square of the edge lengths [4, 20]. The value 1 corresponds to a perfectly regular tetrahedron. The mesh quality of the regular and the remeshing results is very similar. The most salient differences are shown in Fig. 18. Here Fig. 18a and b show that the quality of the regular meshing is slightly better than the remeshing results, as the blue bars stick out on the highest end of the quality spectrum and the yellow bars at the left of that. Figure 18c shows an example of the opposite case, where the remeshing result is slightly better. The difference in quality for all the other test cases is either similar or less pronounced.

The thin 'tails' of lowest quality elements in the volume-length histograms, can often be further improved without much effort. By application of simple operations, such as flipping, most, if not all, of the remaining lowquality tetrahedrons can be eliminated.

Figure 19 shows the result of remeshing for model tool2, including a close-up and a cut of the close-up, showing part of the interior of the mesh. There are virtually no visible signs that the stiffener in the middle was added later by means of remeshing.



Fig. 19 Remesh result for tool2-ab. a Full view of tool2ab, b close-up, c close-up with interior exposed by slice 343

10 Conclusions

We have presented a viable technique for tetrahedral remeshing of feature models for finite element analysis. The approach is based on the feature difference, and approaches the issue from the point of view of individual features. This is sensible since incremental changes to feature models are made in terms of addition, removal, or reparameterisation of features.

A reduction of meshing time between 55 and 90% is achieved for our test models. The efficiency gain depends on several factors. Models that have a large volume relative to the surface area, generally result in the most substantial improvements. They have relatively many internal nodes, which are more costly to optimise than boundary nodes, and hence yield considerable gains because generally these internal nodes can largely be kept fixed. Similarly, for a particular model, a larger node set tends to show bigger gains in efficiency, as the percentage of internal nodes is higher. The time to construct the combined model can increase quickly when many features are overlapping or interacting. In most feature models, however, we expect the number of simultaneous interactions to be low, as many overlapping features are an indication of a bad feature model. Furthermore, for the analysis of the difference between the original and the modified models, resulting in the output of the free and fixed nodes set, we have implemented one particular approach. We feel that this approach is a good compromise between complexity and efficiency. Other approaches might improve the efficiency of this step, but this would have to be researched.

The quality of the remeshing result is consistently on par with the high-quality of the normal meshing approach. A key factor in upholding the quality is the expansion of the free node set, which moves fixed nodes from the immediate neighbourhood of the areas where the attention is directed to, into the set that is actively optimised. The optimisation procedure can achieve a higher-quality result when it can move more nodes. Not expanding the node set leads to a visible disparity between the fixed and free nodes in the final result, and it might even result in failure to properly represent the boundary.

The node density is currently assumed to be uniform over the model. Using graded meshes would require the calculation of a new or adjusted sizing field. The sizing field could possibly be adjusted incrementally. How this impacts the efficiency gain of remeshing, is open for investigation.

Acknowledgments This research has been financed by The Netherlands Organisation for Scientific Research (NWO).

Open Access This article is distributed under the terms of the Creative Commons Attribution Noncommercial License which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

References

- Acikgoz N, Bottasso CL (2007) Metric-driven mesh optimization using a local simulated annealing algorithm. Int J Numer Methods Eng 71(2):201–223
- 2. Branets L, Carey GF (2005) A local cell quality metric and variational grid smoothing algorithm. Eng Comput 21(1):19–28
- Chen L (2004) Mesh smoothing schemes based on optimal Delaunay triangulations. In: Proceedings of the 13th international meshing roundtable. Williamsburg, VA, pp 109–120
- Klingner BM, Shewchuk JR (2007) Aggressive tetrahedral mesh improvement. In: Proceedings of the 16th international meshing roundtable. Seattle, WA, pp 3–23
- Shewchuk JR (2002) What is a good linear element? Interpolation, conditioning, and quality measures. In: Proceedings of the 11th international meshing roundtable. Ithaca, NY, pp 115–126
- 6. Bronsvoort WF, Bidarra R, Nyirenda PJ (2006) Developments in feature modelling. Comput Aided Des Appl 3(5):655–664
- Bidarra R, Bronsvoort WF (2000) Semantic feature modelling. Comput Aided Des 32(3):201–225
- Nyirenda PJ, Bidarra R, Bronsvoort WF (2007) A semantic blend feature definition. Comput Aided Des Appl 4(6):795–806
- Bronsvoort WF, Noort A (2004) Multiple-view feature modelling for integral product development. Comput Aided Des 36(10): 929–946
- Alliez P, Cohen-Steiner D, Yvinec M, Desbrun M (2005) Variational tetrahedral meshing. ACM Trans Graph 24(3):617–625
- Sypkens Smit M, Bronsvoort WF (2008) Variational tetrahedral meshing of mechanical models for finite element analysis. Comput Aided Des Appl 5(1–4):228–240
- Liu W, Yang Y (2007) Multi-objective optimization of an auto panel drawing die face design by mesh morphing. Comput Aided Des 39(10):863–869
- Date H, Onosato M (2008) Triangular mesh deformation based on dimensions. Comput Aided Des Appl 5(1–4):287–295
- François V, Cuillière J-C (2000) 3D automatic remeshing applied to model modification. Comput Aided Des 32(7):433–444
- Sheffer A, Ungor A (2001) Efficient adaptive meshing of parametric models. J Comput Inf Sci Eng 1(4):366–375
- Sypkens Smit M, Bronsvoort WF (2007) The difference between two feature models. Comput Aided Des Appl 4(6):843–851
- Matula DW, Sokal RR (1980) Properties of Gabriel graphs relevant to geographic variation research and the clustering of points in the plane. Geograph Anal 12:205–222
- Spatial Corp. (2006) 3D ACIS modeler. http://www.spatial. com/products/acis.html
- Bidarra R, de Kraker KJ, Bronsvoort WF (1998) Representation and management of feature information in a cellular model. Comput Aided Des 30(4):301–313
- Parthasarathy VN, Graichen CM, Hathaway AF (1994) A comparison of tetrahedron quality measures. Finite Elem Anal Des 15(3):255–261