Virtual Reality Based Interactive Conceptual Simulations

Combining Post-processing and Linear Static Simulations

Holger Graf and André Stork

Fraunhofer Institute for Computer Graphics, Fraunhoferstr. 5, 64283 Darmstadt, Germany
{holger.graf,andre.stork}@igd.fraunhofer.de

Abstract. This paper presents a new approach for the design and realization of a Virtual Reality (VR) based engineering front end that enables engineers to combine post processing tasks and finite element methods for linear static analyses at interactive rates. "What-if-scenarios" have become a widespread methodology in the CAE domain. Here, designers and engineers interact with the virtual mock-up, change boundary conditions (BC), variate geometry or BCs and simulate and analyze its impact on the CAE mock-up. The potential of VR for post-processing engineering data enlightened ideas to deploy it for interactive investigations at conceptual stage. While it is a valid hypothesis, still many challenges and problems remain due to the nature of the "change'n play" paradigm imposed by conceptual simulations as well as the non-availability of accurate, interactive FEM procedures. Interactive conceptual simulations (ICS) require new FEM approaches in order to expose the benefit of VR based front ends.

Keywords: Computer Aided Engineering, Interactive Conceptual Simulations, VR environments for engineering.

1 Introduction

"What-if-scenarios" (conceptual simulations) have become a widespread methodology within the computer aided engineering (CAE) domain. Here, designers and engineers interact with the virtual mock-up, change boundary conditions (BC), variate geometry or BCs and simulate and analyze its impact on the CAE mock-up. The potential of VR for post-processing engineering data enlightened ideas to deploy it for interactive investigations at conceptual stage (interactive conceptual simulations - ICS). It is still a valid hypothesis, while many challenges and problems remain. The conceptual stage during a design is inherently driven by the nature of the "change'n play" paradigm. Coupling these with Finite Element Methods (FEM) imply new solutions and optimizations in view of the current non-availability of accurate, interactive FEM procedures for interactive processing. VR is predominately used for data visualization of scientific raw data. Therefore, classical solutions still use scientific visualization techniques for large data visualization [1] or use interpolated pre-computed result data sets, e.g. [2,3] for interactive investigations. Both approaches imply a bottleneck of

R. Shumaker (Ed.): VAMR/HCII 2013, Part I, LNCS 8021, pp. 13–22, 2013.

data set processing, filtering and mapping and impose restrictions to the processing capability of the underlying system thus influence the turn-around loop of simulation and visualization. Thus, conducting a CAE analysis, steered from a VR environment is a different story and only few research work exist, e.g. [4,5,6]. Classical CG methods are too limited due to the simplified, underlying mathematical models for real-time analysis [7]. In fact several approaches are driven by visual appearance rather than physical accuracy needed within engineering environments. Major attention has been given to the area of *deformable object simulations* in the past [8]. Here, the challenge is to solve the underlying system of differential equations imposed by the physical phenomena modeled by Newton's second law of motion. The approaches make typically use of explicit time integration schemes, fast in evaluation and small in computational overhead, e.g. [9]. Implicit time integration schemes which usually lead to a more stable calculation of the results for solid deformation are based on complete assemblies into large systems of algebraic equations, which might be solved using pre-processing techniques (such as matrix pre-inversion) [10,11], or the conjugate gradient method eliminating corotational artifacts, e.g. [12,13]. A combination of several "best" practices for physical simulations has been published recently in [14] with a dedicated focus on fast solutions being robust to inconsistent input, amenable to parallelization, and capable of producing the dynamic visual appearance of deformation and fracture in gaming environments. However, the scope of all mentioned methods cannot handle more than very few thousand elements or are too imprecise for engineering analyses. So the main question to be answered remains: how can interactive FEM methods be designed and realized at conceptual stage that are efficient with respect to time consumption, computer resources and algorithmic complexity but at the same time result in an accurate and robust simulation?

2 Concept

Aim of our approach is to provide the possibility to couple post processing tasks with a simulation engine, that allows for any interaction performed by the end user to update the simulation results in real-time at the same time to perform an analysis. Here, we are focussing on a direct link of typical post-processing metaphors such as cross sectioning which should provide an insight into the object while simulating the model.

Typically the user wants to move a load case from one node position to another one yet being unrestricted to the number of nodes within the load case. For cross sectioning the plane that cuts through the object will be orthogonal to the device of the end user: In our case a flying pen. Moving the device results in an update of the position of the cross section. A re-simulation of the mock-up should then be performed instantly. However, the use of VR environments implies an intervention with the simulation engine at update rates for (re-)simulation-visualization loops at 20-30 fps, i.e. 0.05 secs. This in turn requires direct access to the underlying mathematical procedures respectively finite element methods.

We headed for a concept based on a classical CAE/VR process chain using a distributed software architecture [15]. Typically, we hold model presentation in a VR client as a surface model being pre-processed by scientific visualization techniques (i.e. extracting the outer domain for visualization, filtering and mapping of results to color scales, etc.). The overall volumetric CAE mock-up is kept on a dedicated simulation service that accounts for linear static analysis.

Thus, the coupling of post processing tasks with the simulation engine requires operations/interactions being performed in the VR client (e.g. moving a load vector/user force) being mirrored to the simulation services. Ideally the simulation engine might be based on optimized FE-methods that could comply with the requirements of the post-processing tasks. I.e., if an engineer uses cross sectioning through the model, the simulation engine would only need to calculate for the "visible" elements (this means the element in the current view frustum). This leads conceptually to a reduction of the solution space, thus a reduction of the system of linear equations that needs to be solved. Another challenge will be imposed by aiming at changing geometrical features in the mock-up (e.g. through holes). Moving features provide an insight to Any changes done at surface level need to be reflected within the volumetrical mock-up in the simulation service.

3 Realisation

3.1 ICS at Boundary Condition Level

The realisation for conceptual simulations at BC level is based on a methodology introduced in earlier work using a pre-processing step [11]. This method uses an inversion of the underlying stiffness matrix $\boldsymbol{A}$ via a *preconditioned minimal residual method* for the overall linear static equation $\boldsymbol{u} = \boldsymbol{A}^{-1} \cdot \boldsymbol{l}$, with $\boldsymbol{u}$ being displacement and $\boldsymbol{l}$ load vector.

The iterative scheme is given by minimizing a Frobenius norm, i.e. it minimizes the functional $F(B) = \|I - BA\|_F^2$ with B being the sought inverse to A. Splitting the functional into components, the scheme seeks a solution $\widehat{f}_j\left(\underline{b}_j\right) = \min_j \left\|\underline{e}_j - A\underline{b}_j\right\|_2^2$, $j=1,\ldots,n$ deploying a CG-iterative method [16]. Completing the inversion of the matrix for a given error threshold, a dedicated stop criterion provides the envisaged precision. This can be adjusted by the engineer himself. Due to the nature of iterative schemes, the precision significantly influences the computation time of the matrix inversion, thus the availability of the model to be inspected. Therefore, this step is done within an offline preparation mode. However, once the model is available, the engineer has several degrees of freedom to investigate conceptual changes at boundary condition level. Here, we have managed to reduce the solution to a simple matrix-vector multiplication.

During the real-time calculation step within the VR client by interactively moving the load case, $\boldsymbol{l}$ is dynamically filled with values according to force and direction by

the position/orientation of the user's interaction device (fig. 2). The simulation filters all unnecessary elements and related rows in the matrix $(a_{i_j}) := 0\,, \forall\, j \in \{s | l_s = 0\}$ (marked black – fig. 1, left), thus, takes only those elements into account that contribute to the results. A second optimisation uses a view dependent element masking technique by neglecting the affected rows within the inverted matrix. As a consequence, a further acceleration and turn-around loop speed-up of the matrix-vector multiplication is feasible. In order to include only those elements visible to the user into the computation, an additional occlusion evaluation step as to which elements are within the view direction of the viewer and which are occluded has to be performed (marked grey – fig. 1, right).

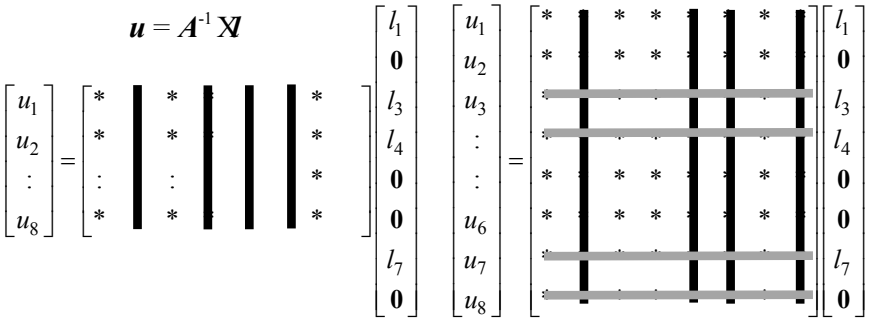

Fig. 1. The used simulation scheme for solving the system of linear equations: throwing away useless values and reducing the matrix-vector calculation load (left); Results of element masking using an occlusion evaluation and taking into account only visible elements (right) [11]

The major advantage of the viewdependent masking due to an occlusion evaluation results in a direct exploitation for post processing tasks.

3.2 Implementation

With respect to the UI concept of a direct interaction method within the *VR client* (`VSC::VR::SG := pVRservice`[1]) simple user interactions based on selection boxes provide a mechanism to assign loads on a surface or a group of elements. The system supports different kinds of loads (BCs) that can be attached or might even be deleted to/from nodes. In order to reflect the changes of the VR client within the instance of the *simulation service* (`VSC::RT := pCAEService`), the methods can be accessed by asynchronous calls to the remote service instance in order not to block the current visualization process. Adequate methods are provided by each service instance through their interface.

[1] Within the implementation the different services are represented by a service instance (`p{X}Service`, $X \in$ `{VR;CAE}`) providing adequate interfaces to other services of the distributed system. `VSC::VR::SG` itself is the client service, `pVRService` an instance of it.

As mentioned above, the user interactions performed on the mock-up are not restricted to a pure visualisation of the results. The user is able during post processing tasks to define cross sections in which the user is able to newly mask the volume taking into account only the visible elements on the surface. Therefore, the `pCAEService` instance allows extracting the outer surface of the mock-up taking into account the position of the pen and the plane being orthogonal to the pen (`pPosition`, `pNormal`). Having fixed the cross section position, the user is able to trigger instantly a re-simulation through a change of magnitude and direction of the user force vector (`pUser_Force_Vector`), allowing a view insight the volume's newly simulated stress field. The magnitude of the user defined force can still be varying.

```
// Operations during the second operational phase (online simulation)
// Within the VR client capture position, orientation and normal of the
// interaction device during post processing as well as the magnitude of
// device movement and establish pUser_Force_Vector
// Extract the cutting plane and mask the resulting elements (BC_MASK)
// for updating the simulation and visualisation (updateRTCalculation)

while(pPosition, pNormal) //- moving the interaction device
{
  pCuttingPlane = pVRService -> updateCrossSection
     (pCAEService, pFlag, pPosition, pNormal)
  pVRService -> updateElementMasks(pCAEService, ELEMENTS, pCutting
      Plane->getNodes());
  pVRService -> updateElementMasks(pCAEService, BC_MASK pCutting
      Plane->getNodes());
  pMesh = pVRService -> updateRTCalculation(pCAEService, pFlag,
      pUser_Force_Vector);
  pVrService -> visualise(pMesh);
}
```

3.3 ICS at Geometrical Level

This section covers interactive modifications of the engineering domain, i.e. geometrical respectively topological level. The implemented techniques for mesh manipulation of the surface/volume mesh are classified as *feature dragging* or surface/volume re-meshing techniques. The process consists in moving vertices of selected simplicial elements causing mesh collapse/split operations in a way that allows the consistency of the mesh being kept using local topological operations. Those operations deal with adapting ill-shaped areas of the given simplicial mesh to a well-shaped area of it.

Of course adequate metrics for quality evaluation of the affected elements during a user defined movement of a group of vertices have to be used in order to perform topological operations for those elements with a low quality (i.e., a quality lower than a given threshold), e.g. [17]. The ill-shaped simplexes are then modified by operations such as edge collapse, edge split, tetrahedron collapse-face swap which are applied appropriately, depending on the damage or degeneracy of the elements in question. The smallest and biggest edges (as well as sliver tetrahedron, i.e. flat tetrahedrons) are adapted to maintain a mesh with a good quality and preserving the consistency for the newly triggered simulation run. This is necessary due to the fact that slight changes of the underlying mesh and thus the associated interpolation schemes might have significant impact on the results and the quality of the simulation.

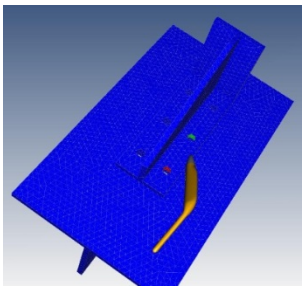

Fig. 2. Interactive feature dragging of through holes (according to pen movement)

In order to be able to move selected features within a mesh, an update mechanism of the manipulations performed on VR client side and its propagation back into CAE service has to be established. As our VR environment processes surface meshes, e.g. the outer surfaces of a CAE mock-up, we need to be consistent with the different object instantiations in the system, several interactions and manipulations done within the client have to be propagated to the `pVRService` and/or `pCAEservice`. The methodology used for the conceptual simulation process with respect to feature movement is shown in fig 3. This mechanism allows the feature movement being extended for real conceptual simulations. The feature history propagation is dived into different routines which are synchronised using the asynchronous method `updateHistory()`.

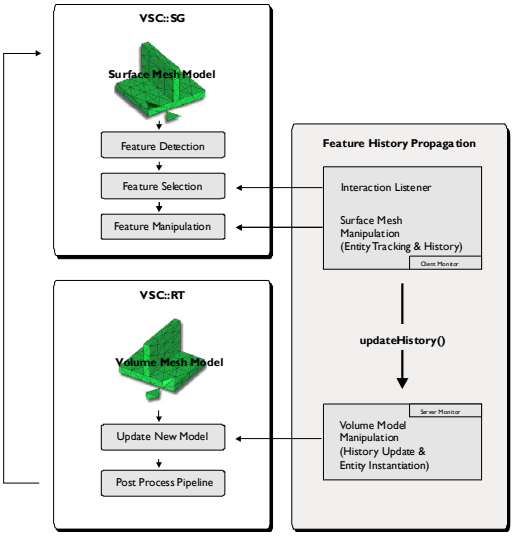

Fig. 3. Feature synchronisation between VR client (`pVRService`) and simulation service (`pCAEService`)

On client side a monitoring mechanism, collecting the modifications done on entity level controls the surface mesh manipulations and records user interactions, i.e. change of entities, displacements of nodes, edges, connectivity information, etc. On service side the monitoring mechanism is responsible for a volume mesh instantiation of the performed actions recorded on client side. As the instantiation does not need to be performed in real-time, it is done after the interaction terminates triggering the `updateHistory()` method.

3.4 Implementation

In general, a typical dragging operation is performed based on a change of selected vertice positions belonging to a feature. The displacements of those vertices entering a new position within the higher level compound of elements (face) trigger a remeshing according to the decisions taken by the quality measure. Several vertices can be classified and marked as those belonging to a feature ("`SELECTED`") and those belonging to a fixed part of an area ("`FIXED`"). As the positions of feature vertices change during dragging operations the vertices undergo a penalty criteria as to which a further collapse, swap or insert operation will be conducted. A special area of interest around the feature is the one containing vertices or edges of the feature as well as vertices or edges that belong to a fixed part of the compound face. Therefore, a further flag for element vertices and edges as "`SHARED`" indicate that several dragging operations are only performed on those ("`SHARED`", "`SELECTED`") and lead to the envisaged remeshing. After a principle topological modification of elements their vertices are marked as "`KILLED`", "`SELECTED`", "`SHARED`" or "`FIXED`" depending on whether vertices are deleted (i.e. during a collapse operation) or further used for operation (i.e. during a split process). The realization sequence in pseudo code looks like:

```
// Selection of entities
      defineFeatureBoundary(in pPosition, in pOrientation, in
                  p{element_heuristics}, out Q p{entitiy_selection});

// label the boundary elements
      labelBoundary(in Q p{entitiy_selection}, out Q p{marked_entities});

// inquire for "SHARED" and "SELECTED" elements and define buffer of
// elements that listen to modifications triggered by VSC::EVT events
      getSharedBoundaries(in Q p{marked_entities}, out
                                          Q p{buffer_entities});

// translate pPosition and pOrientation of the pen into displacements of
// the vertices for the buffer_entities and record movements

  while(pPosition, pOrientation) // -- Start movement of the pen
  {
      calculateMovement(in pPosition, in pOrientation, out pDirection,
                                      out pStart, out pDistance);

// Topological operations are applied to every boundary resp. buffer
// vertex.
      applyMovementsToSurface(in Q p{buffer_entities}, in pDirection ,
                                                  in pDistance)
```

```
        {
             ∀ pEntities ∈ p{buffer_entities}
             {
             // edge_collapse, edge_swap, edge_split process
                  adaptiveRemesh(in pEntities, in pDirection, in
                                                          pDistance);
                  labelEntities(in pEntities, out Q
                                              p{new_buffer_entities});
             }
        }
   }
// Update the remote service with modified entities and instantiate
// changes on the surface into the volume mesh
   pVRService::SG -> sendc_updateHistory(pCAEService, pDirection,
                pDistance, pStart, Q p{new_buffer_entities});
```

4 Results

The presented system enables us to link typical post-processing tasks (e.g. cross sectioning, etc.) directly to the simulation engine evaluating the viewpoint and current force vector of the engineer. He is then able to define cross sections enabling to newly mask the volume taking into account only the visible elements on the surface (see fig. 4). A re-simulation due to a change of the magnitude and direction of the user force vector allows a view insight the volume's stress field. The magnitude of the user defined force can still be varying.

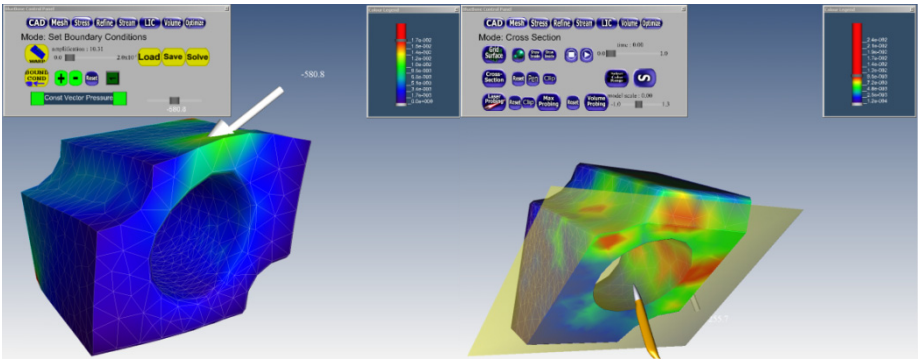

Fig. 4. Integrated post-processing and simulation; left: resulting deformations (scalable post-processing of displacement field); right: results of a cross-section simulation with an update of element masks

The integration of a conceptual change of a feature position within a given design domain into the framework follows the mechanism described in above using the feature history propagation. As a result, it enables the user to mark and select certain features in the domain on client side and "drag" them in 3D space to another position. As the manipulations within the `VSC::SG` client are performed on the surface

representation, the volume mesh kept in the VSC::RT service backbone has to be updated accordingly. The selection of mesh entities is based on face identification. Several compound elements belonging to a compound of faces with heuristically similar characteristics or a CAD face can be selected (see fig. 5). During a spatial change of selected features, i.e. through holes, on client side (VSC::SG) a monitoring mechanism collecting the modifications done at entity level controls the surface mesh manipulations and records user interactions, i.e. change of entities, displacements of nodes, edges, connectivity information, etc. They are then propagated back to the VSC::RT service.

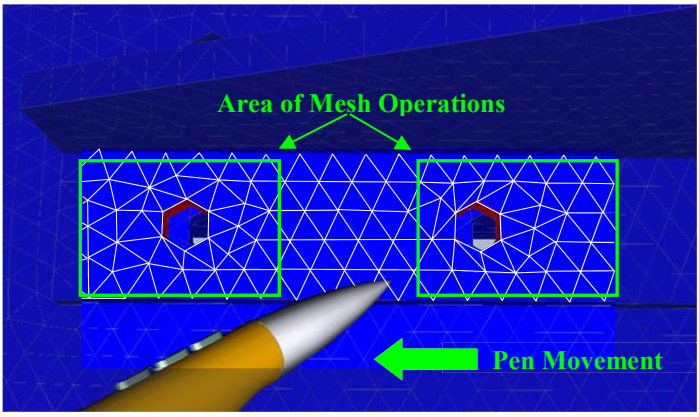

Fig. 5. Interactive feature dragging: two through holes (marked as red) and resulting mesh operations at surface level being propagated to the volume simulation engine

5 Conclusion

This paper presents a new approach for the design and realization of a Virtual Reality (VR) based engineering front end that enables engineers to combine post processing tasks and finite element methods for linear static analyses at interactive rates. The engineer is able to steer post-processing analysis and re-simulation "at his fingertip". The implementation has been done within a distributed set-up in order to comply with the limitations of CAE simulations and their mock-ups. Several operations can be performed in real-time for selected domains. However, the model size cannot be arbitrary as shown in [11]. This might impose a critical limitation to the use of the system for larger models. We therefore are working on subdomaining mechanism that might reduce the overall domain in order to provide also a certain scalability of the system. Yet, we are optimistic that the presented ICS might enable engineers to use this paradigm for "what-if-analysis" in order to be capable of answering the question: *"where do I have to spend my analysis time?"*. As further future work we head towards a closer interlink between mesh and simulation. Thus, exploiting the neighborhood relationships between nodes might lead to an optimization of the stiffness matrix entries that might eventually lead to a significant reduction in computational time.

References

1. Scherer, S., Wabner, M.: Advanced visualization for finite elements analysis in virtual reality environments. International Journal on Interactive Design and Manufacturing 2, 169–173 (2008)
2. Stork, A., Thole, C.A., Klimenko, S., Nikitin, I., Nikitina, L., Astakhov, Y.: Simulated Reality in Automotive Design. In: Proc. of IEEE International Conference on Cyberworlds 2007, pp. 23–27 (2007)
3. Lee, E.J., El-Tawill, S.: FEMVrml: FEMvrml: An Interactive Virtual Environment for Visualization of Finite Element Simulation Results. Advances in Engineering Software 39, 737–742 (2008)
4. Vance, J.M., Ryken, M.J.: Applying virtual reality techniques to the interactive stress analysis of a tractor lift arm. Finite Element Analysis and its Design 35(2), 141–155 (2000)
5. Connell, M., Tullberg, O.: A Framework for the Interactive Investigation of Finite Element Simulations within Virtual Environments. In: Proc. of the 2nd International Conference on Engineering Computational Technology, Leuven, Belgium (2000)
6. Connell, M., Tullberg, O.: A Framework for Immersive FEM Simulations using Transparent Object Communication in Distributed Network Environments. Advances in Engineering Software 33(7-10), 453–459 (2002)
7. Georgii, J.: Real-Time Simulation and Visualisation of Deformable Objects. Dissertation, Institut für Informatik, Technische Universität München (TUM) (2008)
8. Nealen, A., Mueller, M., Keiser, R., Boxerman, E., Carlson, M.: Physically Based Deformable Models in Computer Graphics. Computer Graphics Forum 25(4), 809–836 (2006)
9. Debunne, G., Desbrun, M., Barr, A., Cani, M.-P.: Dynamic real time deformations using space & time adaptive sampling. In: Proceedings of SIGGRAPH 2001, pp. 31–36 (2001)
10. Bro-Nielsen, M., Cotin, S.: Real-time volumetric deformable models for surgery simulation using finite elements and condensation. In: Proceedings of Eurographics 1996, pp. 57–66 (1996)
11. Graf, H., Stork, A.: Linear static real-time finite element computations based on element masks. In: Proc. of the ASME 2011 World Conference on Innovative Virtual Reality, WINVR 2011, Milan, Italy (2011)
12. Müller, M., Dorsey, J., McMillan, L., Jagnow, R., Cutler, B.: Stable real-time deformations. In: Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 49–54 (2002)
13. Müller, M., Gross, M.: Interactive virtual materials. In: Proceedings of Graphics Interface, pp. 239–246 (2004)
14. Parker, E.G., O'Brien, J.F.: Real-time deformation and fracture in a game environment. In: Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Computer Animation, New Orleans (2009)
15. Graf, H.: A "Change'n Play" Software Architecture Integrating CAD, CAE and Immersive Real-Time Environments. In: Proc. of the 12th International Conference on CAD/Graphics, Jinan, China (2011)
16. Chow, E., Saad, Y.: Approximate Inverse Preconditioners via Sparse-sparse Iterations. SIAM J. Sci. Comput. 19(3), 995–1023 (1998)
17. Shewchuk, J.: What is a Good Linear Element? Interpolation, Conditioning and Quality Measures. In: Proc. of the 11th International Meshing Roundtable, pp. 115–126 (2002)