Visualizing Results of Transient Flow Simulations

Harald F. Mayer Behnam Tabatabai
JOANNEUM RESEARCH
Institute for Information Systems
A-8010 Graz, Austria
{mayer,tabatabai}@pbox.joanneum.ac.at

Abstract

This work briefly describes our approach to visualize results of transient flow simulations in the application areas of groundwater flow and pollutant transport as well as compressible fluid flow in engine parts. The simulations use finite element data structures and can have geometries which change over time. We designed a client-server model to handle the huge amount of data that can be obtained either directly from the simulation process or from files on disk. As standard visualization packages are not able to cope with transient unstructured data, we implemented streamlines, stream surfaces and particle systems as our main visualization methods. Our experiences and results with these techniques will be discussed in this paper.

1. Introduction

We are a small working group which deals with numerical simulations, computer graphics and visualization for scientific computing at Joanneum Research, an Austrian research organization owned by the Province of Styria. Our task in the visualization group mainly comprises the areas of environment and flow simulations [4,8].

Regarding flow simulations, we deal with two different application fields which we will briefly describe in the following. One of them is the numerical simulation of groundwater flow and pollutant transport. This simulation is carried out by using our own newly-developed proprietary finite element program called FEJUX [5]. This program runs on every workstation or supercomputer supporting OSF/Motif (Unix or VMS). The typical applications of this package are investigations of the impact on the groundwater situation caused by building projects. One example is the simulation of driving a road tunnel through an environmentally sensitive area with mineral springs. The main part of interest is the influence of different construction methods (e.g. shotcrete lining and/or advance grouting) on the groundwater lowering in the area of observation.

The time-dependent calculation of the lowering of the groundwater-table is based on a 2D horizontal model. The water inflow into the tunnel, used as a boundary condition for this investigation, is calculated in a pre-processing step using two horizontal models. Primary results from this simulation are the height of the groundwater-table and the velocity vectors stored per node for all time steps computed.

The other field of application is a problem from the classical computational fluid dynamics (CFD) domain. It deals with the simulation of flow, spray mixture preparation and combustion in engines. The numerical simulation is realized by another Austrian company, AVL, with its proprietary CFD program called FIRE [1]. FIRE is a powerful tool for the prediction of flow, temperature and concentration distribution inside arbitrary flow domains. A broad variety of two- or three-dimensional turbulent compressible fluid flow problems can be solved based on the finite volume method, using a two equation k-ε model. Our task is to visualize the results of these simulations.

An important issue of the application domains mentioned is the structure of the data which is principally the same. Both simulations operate on a data structure which is a mesh of finite elements. This means that the spatial domain of the simulations is of arbitrary shape, usually defined by B-spline surfaces, subdivided into small (finite) elements. The results for each time step are calculated either on a per element- or a per node-basis. One characteristic feature which can not be found in other simulation packages very often is the possibility of moving coordinates and also changing topologies. Moving coordinates mean that the coordinates of the nodes are changed with the time whereas the topology of the mesh remains the same, in contrast to changing topologies where everything is allowed to be changed.

2. Results

At the very beginning of every visualization there is the problem of data acquisition. This is particularly true for transient data sets. Usually, all geometry information and results are stored in huge data files which are saved on the computer performing the computation. The geometry information consists of 3D coordinates for every node and arrays describing the connections between elements and nodes. As results we get at least one velocity vector and usually also some other scalar values per node. The problem size we have to deal with typically lies between 5000 and 200000 elements and at about 2000 simulated time steps implying data sizes up to 5 Mbyte per time step for the velocity results only. If the geometry of the objects changes over time, also this information has to be obtained more often than once.

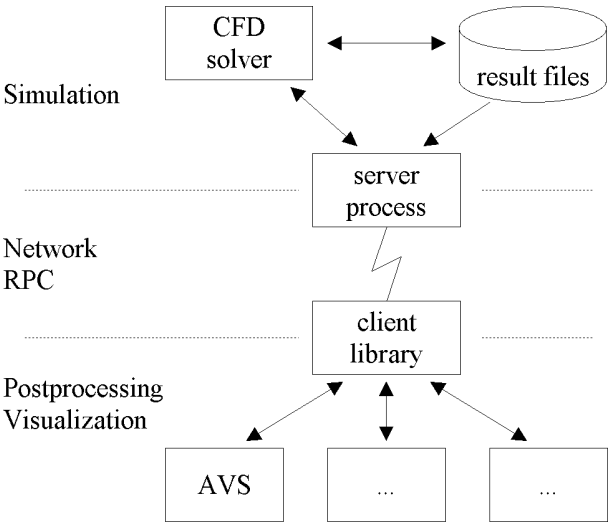

Figure 1: Client-server model for data acquisition.

To have an easy and network-transparent access to all this data, we developed a client-server model application interface based on the standard RPC mechanism. The visualization, running on a high-speed graphics workstation, represents the client that requests data from the server (usually a number-crunching workstation or supercomputer). The server either gets its data directly from a running simulation or from a pre-calculated set of data on a fileserver (see Figure 1). As visualization and simulation are independent processes and can run simultaneously we designed an asynchronous protocol, so no process has to wait for the other. This functionality is achieved by providing two steps for getting data. In a first call the client tells the server which data it wants and returns control immediately. After that, both the client and the server processes can work independently. With a utility function the client can ask the server if it is ready to send the data which then can be obtained by using a final call. This approach is very powerful for visualizing transient data since the client usually knows what data (time step) it will need after preparing one visualization.

As our next step we evaluated the standard visualization tools for our data. We soon discovered that a lot of visualization packages available are not suitable for unstructured data like ours and that the support for transient data is very limited. Finally, we decided to use AVS [10] from AVS Inc. as our primary visualization and development tool and to make use of its extendibility features to implement visualization methods suitable for our applications. However, we carefully designed all interfaces for our visualization methods to be independent from AVS and to have the possibility to integrate these methods into other visualization packages or postprocessors easily. AVS itself proved to be very powerful for visualizing scalar results as shown in Figure 2 which presents a set of isosurfaces of pressure inside a manifold intake system.

Both simulations produce flow data (in our cases 2D or 3D velocity vectors per node) as one of their results. Therefore, we developed and implemented methods to visualize such vector data sets. The most straightforward method - also directly available by most visualization systems - is to draw vectors on every position with an existing result. This method proved to be useful for small simulations but failed for simulation sizes we normally have to deal with. A large number of vectors is very confusing for the scientist who analyses the data.

The next method, which is very common in flow visualization [3], is to calculate the way a massless particle would take in the flow. Such ways are called streamlines for stationary flows and particle paths for transient flow fields. We soon realized that the available programs or modules are not able to meet our requirements. These requirements include:

- the ability to work with unstructured data;
- to have mechanisms to calculate particle paths (transient models);
- a good support to define starting or ending points;
- methods to deal with the boundaries of the object the flow goes through or around.

It is especially the last item that is very important when calculating streamlines in a real object. Considering a tube where flow goes through, particles are only allowed to enter at one end and to leave at the other end. It is not possible for a streamline or particle path to end inside the tube, just because it hits the boundary surface. Due to the reasons mentioned above, we had to develop our own implementation of streamlines and particle paths which

are able to cope with these requirements. The problem with the streamlines ending at the boundary surfaces - where the velocities approach zero - is solved by assuming a reflection of the particles about the surface normal. For the integration of streamlines we implemented an improved Euler formula and the classical Runge-Kutta algorithms of orders 3, 4, 5, and 8. For the orders 3 and 5 we used algorithms to calculate the stepsize in the integration process adaptively. It turned out that the Runge-Kutta algorithm of order 5 with adaptive stepsize yields the best results. Higher order of integration only causes higher computational effort but does not produce more accurate results. The adaptive stepsize is very important since the size of the elements and the associated velocity fields may vary enormously in different parts of the investigated object.

Another important issue is the interpolation of results while preserving C^0-continuity at element boundaries. This is achieved by using shape functions exactly in the same way as used in finite element applications. For this approach, however, the discrete function values must be known at element nodes. This is true only in the case of primary results of a finite element simulation. The secondary results are calculated indirectly, i.e. out of primary results at the Gauss integration points, yielding higher accuracy than other locations inside the element [2]. Before we use the element intrinsic shape functions these results must be extrapolated to the nodes. Among many different and well-investigated methods, we choose the most simple and obvious approach of linear extrapolation towards element nodes. For the nodes shared by more than one element, this generally entails discrepancies in the extrapolated function value which we resolve by a non-weighted averaging.

Next, we implemented different approaches for selecting the starting or ending points of the streamlines. These approaches include an interactive point and click interface which turned out to be uncomfortable to use due to the difficulties of pointing into a 3D object on a 2D screen. The selection by defining a cut plane through the object has proved to be a useful alternative. Of course, it is also possible to take just the inlet or outlet endings of the object. Another approach was to apply the idea of Helman´s work [6] on critical points for our 3D unstructured data sets. These critical points, i.e. points where the magnitude of the vector vanishes like turbulences and other significant features of the flow topology, can be used as starting points of streamlines in both directions. The streamlines starting at the inlet or near critical points offer a good impression of the main flow directions as well as of the turbulent remaining flow.

Streamlines are visualized as simple lines or tubes, though shaded tubes give a better idea of the streamline´s position in 3D space. The lines or tubes can also be rendered color-coded with another scalar value such as magnitude of velocity. Figure 3 shows an example of streamlines rendered as tubes. This example figures the flow inside a manifold intake system from the inlet (air filter) to the valve just opened.

Next, we moved to the concepts of stream surfaces to visualize the flow topology. Stream surfaces can be generated by the triangular tiling of adjacent pairs of streamlines. Figure 4 shows an example of stream surfaces inside our test object. These surfaces show one side color-coded according to the magnitude of velocity and the other side rendered in a single color. We choose this form of representation, since it would be very difficult to distinguish between both sides if they were rendered in a similar way. The advantage of stream surfaces is the small number of streamlines that need to be calculated to have a superior impression of the flow topology. However, there are problems in generating stream surfaces if the gap between adjacent streamlines becomes too wide. In this case, the surface has to be divided and new streamlines have to be calculated [7].

Particle paths should not be drawn as simple lines or tubes because it is very difficult to interpret the time incorporated in the paths. A better solution for particle paths is to use the idea of particle systems [9]. In such systems only a set of particles moving along their calculated paths is drawn as spheres, producing a form of computer animation. As for complex simulations, our hardware (SGI Crimson VGX, DECstation 5000/240 PXGT) is not fast enough to generate such animations in realtime. To cope with the problem, we produce animations on video utilizing single frame recording on an Abekas A66 digital disk recorder. A disadvantage of this technique is that in a single frame the observer does not get any visual clues about speed and direction of the particles. This problem is solved by using a kind of "motion blur" technique where each particle gets a tail consisting of a small number of previous positions rendered as transparent darker spheres.

With the help of these techniques we visualized the results of the groundwater simulation mentioned in the introduction (Figure 5). The water table is rendered as a layer color-coded by its height, with particles running along their paths into the tunnel as its construction advances. In this animation we included an advancing tunnel geometry to give the observer visual clues about the progress of the construction phases.

3. Future and Conclusions

As stated above, today´s standard visualization tools are not able to handle transient data satisfactorily for our

applications. However, some tools are easy to extend, thus making it possible to achieve visualization based on existing software. It is very convenient that one has not to reinvent the wheel and can make use of the existing powerful rendering features of software packages like AVS. A desirable enhancement for future versions of professional visualization systems would be some kind of support for transient data sets.

Our future work will particularly be oriented towards direct volume rendering for scalar results of transient flow simulations. Again, it will be important to handle unstructured data and to find methods that save computation time when visualizing a time series of data, since volume rendering is very CPU-intensive. At present, we investigate the methods for breaking down the integration process into result-dependent and result-independent parts, in order to calculate as much as possible in a first pre-integration phase and subsequently apply different results (e.g. from different time steps) to the constant pre-calculated data.

Animations on video have already been presented to different groups, from students to managers as well as technical and scientific experts. In all cases, the animations have proved to be very helpful, either to introduce to the problem in short time or to help grasp the problems and their possible solutions.

Acknowledgements

This work was partially supported by funds from the Austrian Industrial Research Promotion Fund (FFF) under contract number 6/674.

All data for the visualization of flow simulations in engine parts were provided by the company of AVL.

The authors would like to thank Georg Thallinger for his efforts in designing and implementing the streamline and particle path code and Doris Plank for her valuable support.

References

[1] BACHLER G. ET AL, *A General Purpose Fluid Mechanics Software Package,* Leavelet, 1990.

[2] BARLOW J., *Optimal stress locations in finite element models*, Int. J. Num. Meth. Eng., Vol. 10, pp. 243-251, 1976.

[3] BRYSON S., LEVIT C., *The Virtual Windtunnel: An Environment for the Exploration of Three-Dimensional Unsteady Flow*, Visualization ´91 Conference Proceedings, pp. 17-24, 1991.

[4] GRUBER-GEYMAYER B., MAYER H.F., HAAS W., *Using a Standard Tool for Visualization of Environmental Data,* in German, presented at the 2nd Workshop on Visualization of Environmental Data, Schloß Dagstuhl, Germany, 1991.

[5] HAAS W., SCHEWIG D., RESCH M.M., *Numerical Simulation of Ground Water Flow and Ground Water Pollution in a Graphical Software Environment*, Computer Techniques in Environmental Studies IV, P. Zannetti (ed.), Computational Mechanics Publications, Southampton Boston, 1992.

[6] HELMAN J., HESSELINK L., *Visualizing Vector Field Topology in Fluid Flows,* IEEE Computer Graphics & Applications, Vol. 11, No. 3, pp. 36-46, 1991.

[7] HULTQUIST J.P.M., *Constructing Stream Surfaces in Steady 3D Vector Fields*, Visualization ´92 Conference Proceedings, pp. 171-178, 1992.

[8] MAYER H.F., FANK J., HAAS W., *Visualizing complex relationships in the soil area*, in German, presented at the 3rd Workshop on Visualization of Environmental Data, Zell an der Pram, Austria, 1992.

[9] REEVES W.T., *Particle Systems - A Technique for Modeling a Class of Fuzzy Objects,* SIGGRAPH'83 Conference Proceedings, Vol. 17, pp. 359-376, 1983.

[10] UPSON C., FAULHABER T., KAMINS D., LAIDLAW D., SCHLEGEL D., VROOM J., GURWITZ R., VAN DAM A., *The Application Visualization System: A Computational Environment for Scientific Visualization,* IEEE Computer Graphics & Applications, Vol. 9, No. 4, pp. 30-42, 1989.

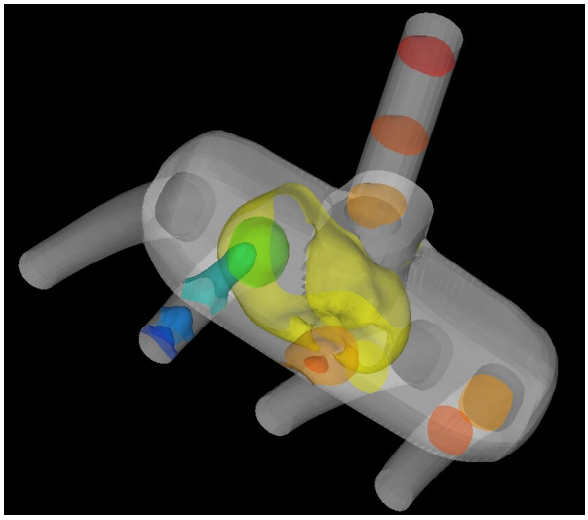

Figure 2: Isosurfaces of pressure inside a manifold intake system.

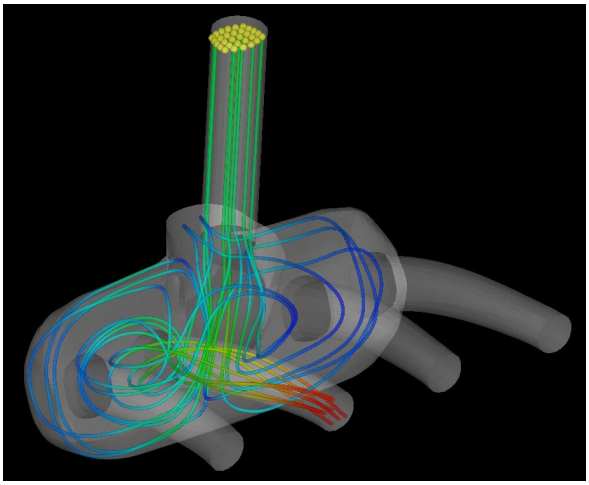

Figure 3: Streamlines rendered as tubes.

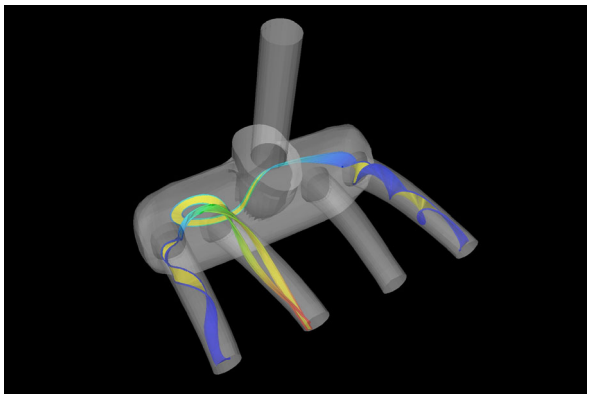

Figure 4: Two stream surfaces.

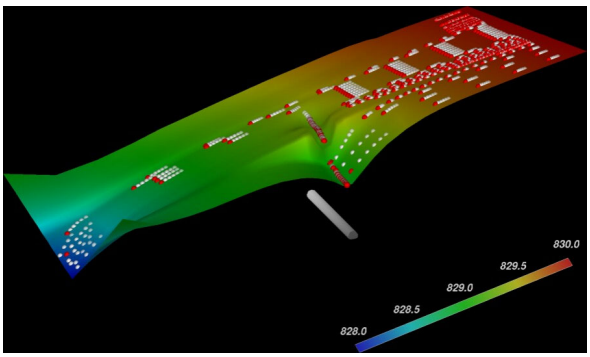

Figure 5: Groundwater flow visualized by particles with tails.