

The Dataflow Visualization Pipeline as a Problem Solving Environment

Helen Wright[†], Ken Brodlie[†] and Martin Brown[‡]

[†]School of Computer Studies, University of Leeds, Leeds LS2 9JT, UK
Email: helenw, kwb@scs.leeds.ac.uk

[‡]British Gas plc, Gas Research Centre, Ashby Road, Loughborough LE11 3QU, UK
Email: martin.brown@bggrc.co.uk

Abstract. Visualization systems based on the dataflow paradigm are enjoying increasing popularity in the field of scientific computation. Not only do they permit rapid construction of a display application, but they also allow the simulation to be incorporated, giving the scientist the opportunity to interact with the calculation as well. However, if these systems are to realise their full potential for problem solving, additional support must be given for the iterative investigation which characterises this activity. This paper will review these systems, identify some of their shortcomings as problem solving environments and describe current work which addresses these deficiencies. An implementation of our ideas for the IRIS Explorer system will demonstrate their effectiveness in a study of gas turbine exhaust emissions.

1 Introduction

Visualization has been a tool of the computational scientist from the time of the first mainframes, through the rise and fall of the minicomputer and nowadays in conjunction with powerful desktop workstations. In the early days programs would run overnight, outputting a file of results which in turn was processed by a specially-written graphics program. But with the change in computing environment has also come a change in our way of working; the trend has been to move away from application-specific visualization in favour of re-usable software packages and, most recently, towards the so-called Modular Visualization Environments (MVEs).

This paper will give an overview of these systems – what they offer in visualization and, moreover, as Problem Solving Environments (PSEs). We then identify some of their shortcomings in this respect and describe current work to extend one such system. The tools developed will be demonstrated in a case study taken from reaction chemistry and the paper concludes with some possibilities for future work.

1.1 Modular visualization environments

Cameron [1] gives a useful summary of what constitutes an MVE – all tend to consist of building blocks, or modules, performing separate functions. Blocks are connected by links, typically within a visual programming environment, so

that together they describe the series of transformations the data will undergo. These systems allow visualization program development without programming knowledge, though there is still a learning curve involved in their use. Four visualization systems in common usage can be classed as MVEs; these are AVS [2], IRIS Explorer [3], IBM Data Explorer [4] and Khoros [5].

A key feature of MVEs which contributes to their growing popularity in scientific computing is their extensibility, allowing the simulation process to be incorporated into the environment and the results delivered directly into the visualization pipeline. Mathematical parameters can be implemented as widgets on the simulation module, giving the opportunity to steer [6, 7] the calculation and perhaps halt part-way if the visualized results so indicate.

MVEs make use of the dataflow paradigm [8, 9], where processes are categorised as performing filter, map or render functions. Input data can be read in or generated within the environment if steering a calculation. The original dataflow model is an excellent paradigm for visualization but has limitations as an environment for problem solving. One is the inherently uni-directional flow which makes it difficult to query data held at, say, the filter stage, by interacting with a render process. Another is the difficulty in preserving previous states of the pipeline during the iterative solution of a problem – for example, when comparing results with those from an earlier run employing different parameters.

As MVEs develop there have been improvements made which can be seen as an extension of the dataflow model. For example, envisioning data flowing upstream as well as downstream is the basis of current systems' image probing capabilities; that is, display functions which return a geometric position to an earlier visualization process in order to retrieve data values. Likewise, facilities to perform animations by varying process parameters in sequence can be thought of as adding loop constructs to the dataflow model.

Both of these extensions go some way towards supporting the iterative process which is inherent in problem solving, but even modern MVEs fall short of the comprehensive facilities which are needed for this activity. Steering applications of the type described remain a rarity and MVEs tend not to be used as PSEs – hence the motivation for our current work.

1.2 GRASPARC

GRASPARC (GRaphical Support for PARallel Computing [10]) looked at the way an investigation progressed and put forward the history tree as a model for computational problem solving. The objective was to record information (input parameters and output results) as the simulation progressed, so that the calculation could be stopped at any stage and rolled-back to some previous point. Here a modified set of parameters could be specified based on the recorded set, and the simulation restarted. This created a branch point in the tree. An important feature was that the *tree as well as the data* was recorded, providing an audit of how the problem had been tackled.

GRASPARC treated the simulation and visualization processes as external to a central, invariant core of software which provided the tree-oriented database and user interface. Interaction by the scientist was by means of the tree – for example, “start a new branch here”, “halt this branch”, “start a new tree” and so on (Figure 1). Given the chosen architecture, it followed that GRASPARC

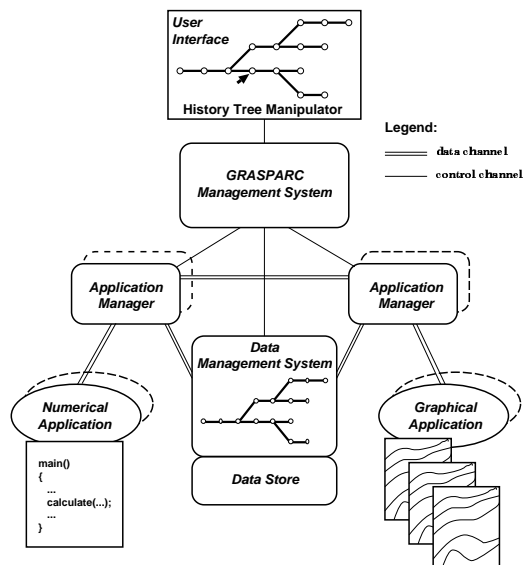


Fig. 1. The GRASPARC Architecture

adopted its own data model, so Application Managers were needed to convert the external numerical and visualization formats into GRASPARC format.

A number of demonstrators were constructed during the GRASPARC project to demonstrate the validity of the approach and some of these used MVEs as their visualization component. The problems tackled ranged from computational physics, computational fluid dynamics and planetary motion [10], to reaction chemistry [11]. Whilst successful in terms of their problem solving approach, a considerable development effort was needed to construct the Application Managers and configure these demonstrators. The thrust of our current work, therefore, has been to find a more accessible means of delivering this type of support to potential users.

2 HyperScribe - problem solving support for dataflow

Brodlie and Wright [12] describe a number of ways of delivering this support. One suggestion is to turn the GRASPARC architecture ‘inside out’, that is, instead of having an MVE attached to the GRASPARC framework, we could put GRASPARC functionality inside the MVE. We have investigated what this entails within the dataflow paradigm and have implemented a first version for IRIS Explorer in the form of the HyperScribe module. Wright and Walton [13] furnish a detailed description of the implementation – here we give just a brief overview in order to present our case study.

2.1 Architecture

GRASPARC functioned by passing messages between the various components so that the central management system could direct the attached applications

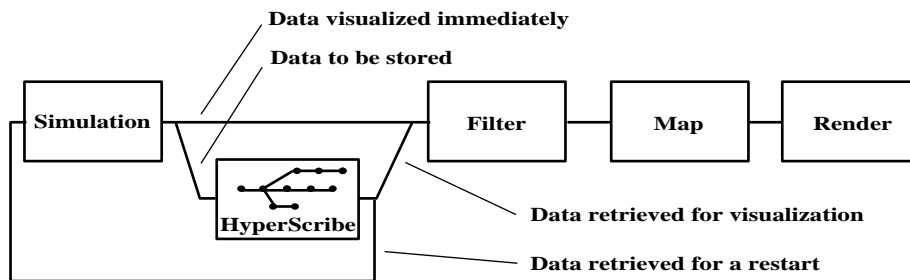


Fig. 2. The HyperScribe architecture

to perform tree-building operations. For example, if the user wanted to compute a new branch, the data store would be requested to provide the restart data and pass it to the computational part, which in turn would generate new data to be passed back to the data store. The processes in a dataflow system, however, are activated only when new data arrives; the potential for directing the flow of data is thus very limited by comparison with a GRASPARC system.

As we might expect in a dataflow environment, the key to resolving this difficulty is to concentrate on the data rather than the tree operations. HyperScribe has therefore been designed around the concept of data inputs and data outputs, as the architecture in Figure 2 shows.

The price paid for this simplification is that the user must decide which data sets to store and how to place them to form the tree – functions that were formerly carried out by the GRASPARC management system. Similarly the user must decide which data sets to retrieve and how to route them to other modules in order to effect a computation restart, or to perform visualization of previously computed data.

2.2 Implementation

The IRIS Explorer implementation of this architecture consists of two parts: the first part is the HyperScribe module itself which deals with the storage of the data and the physical representation of the tree. The second part is a dedicated Render module which draws the tree and handles the user's interaction with it when they specify which data to store or retrieve. Communication between the two is by means of the Geometry and Pick datatypes. Data inputs to the HyperScribe part are implemented as text slots on the left of the user interface panel, whilst the corresponding data outputs appear as text slots on the right hand side. The two modules can be grouped together if desired, as shown in Figure 3.

Data and problem independence. Using text slots within HyperScribe rather than integers, floats or doubles ensures that *any* type of scalar parameter can be stored since all are first converted to a character string. A slot may also be used to record some comment typed in directly by the scientist to describe the current state of the experiment. For non-scalars such as lattices, pyramids, images or geometry the data must first be passed through a WriteLat, WritePyr, WriteImg or WriteGeom module to generate a file – only the file name need then be stored by HyperScribe. Furthermore, when HyperScribe is first launched in

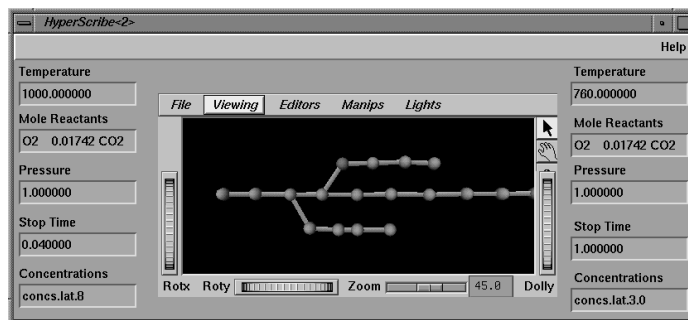


Fig. 3. The HyperScribe/Render module combination

the map the user is given the opportunity to re-label the text slots however they wish, so the module is completely problem-independent.

Data capture and retrieval. HyperScribe's set of input text slots capture parameter values and the file names of any associated data sets delivered from the upstream simulation. The dataflow model ensures that each new occurrence upstream is delivered to HyperScribe immediately it is generated, but the final decision on whether to record the information rests with the experimenter. If it is to be recorded, an *add* event is signalled at an appropriate point on the tree. This is received by HyperScribe, which enters the input information into its database and generates a sphere as the physical representation of this stored data set. The location of the sphere is determined by the coordinates of the mouse cursor when the event was generated. Pairs of spheres can be linked together by cylinders in order to develop the tree structure.

Subsequent retrieval of the information is by interaction with its representative sphere. When a *retrieve* event is specified for a sphere, the stored information is found in the database and delivered to the set of output text slots, from where it can either flow down the map for visualization, or upstream to restart some computation process. Since retrieved data flows immediately from the text slots as soon as it is delivered, we have also found it useful to implement an *inform* event. This behaves just like *retrieve* but delivers values to the input side of HyperScribe's interface. Thus it is possible to browse the tree to find a particular data set before committing the information to flow into the map.

The database exists in shared memory whilst the module is running but HyperScribe also allows for it to be written to file between sessions. When the module is restarted the user is given the option to continue with a previous experiment, or to start afresh.

3 Case Study

Our case study is taken from the field of chemical kinetics, where the computational task is to determine the time-varying concentrations of a number of chemical species taking part in a reaction. The system is modelled as a set of ordinary differential equations, one for each species plus possibly temperature and pressure.

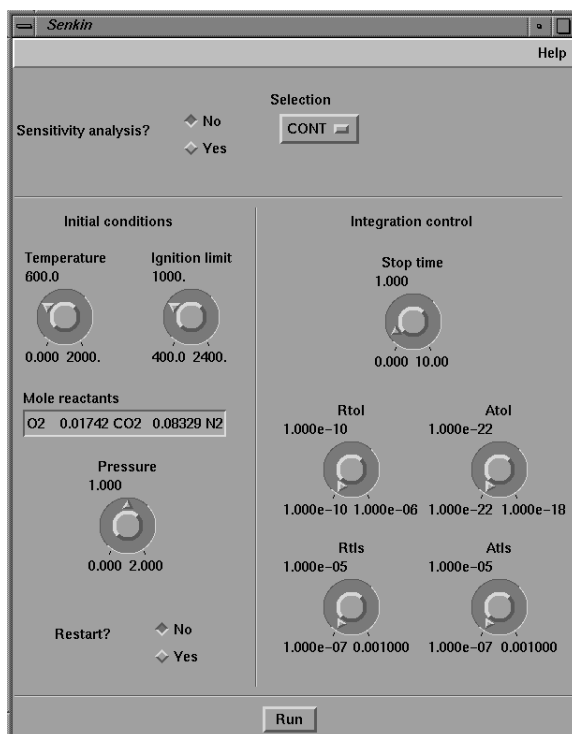


Fig. 4. The SENKIN module user interface panel

3.1 Computation component

The physical process being modelled is the combustion of fuel in gas turbine based Combined Heat and Power systems, with particular reference to the impact on different exhaust gas concentrations of changes in operating conditions [14]. In this paper we will pay particular attention to the conversion of NO to NO_2 at different initial temperatures of the gas mixture, since the relative concentrations of these are significant when considering emission limits for such systems.

The computation component has been developed from the SENKIN program distributed as part of the CHEMKIN system [15] by Sandia National Laboratories, California. For this study we have implemented SENKIN as an IRIS Explorer module which can be used in the Map Editor along with any of the system-provided modules. SENKIN's original keyword input file has been replaced by a user interface panel with interactive widgets (Figure 4), whilst the output results comprising species concentrations and sensitivities are generated in the form of the IRIS Explorer lattice datatype. The user specifies their problem by altering widget values and starts the computation by pressing the *Run* button. Once the integration is completed the results flow down the map and are visualized immediately. Experimentation with the input parameters is very easy – for instance, a sequence of time traces (plots of species concentrations *vs.* time) at various temperatures can be created and visualized within just a few minutes.

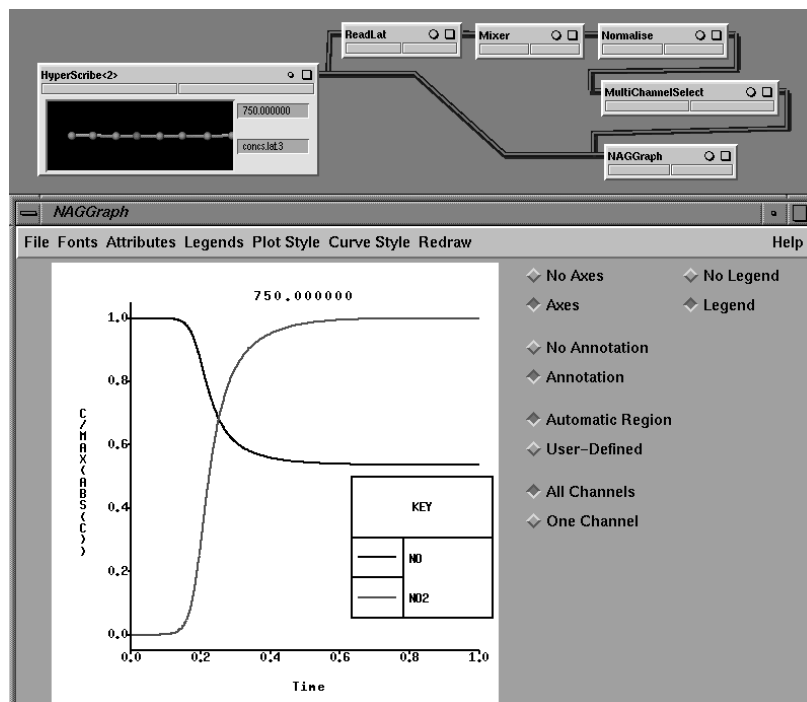


Fig. 5. Offline visualization of time series plots

3.2 Problem solving support

Having made a quick pass through the time traces for a number of temperature values, the experiments are re-run in a systematic way using HyperScribe to record the initial gas composition, temperature increasing by 50K intervals, pressure and the file name in which the corresponding lattice of results is recorded.

In figure 5 HyperScribe has been combined with its Render module to give a simplified composite interface, where only the retrieved temperature and file name parameters are exported to the map. The upper connection carries the file name to the standard ReadLat module, whose lattice output then flows into the visualization pipeline. The lower connection carries the corresponding temperature value used to title the plot. Using HyperScribe's data retrieval facility we can thus traverse the tree to bring back the results and view them offline in *flipbook* style.

Individual SENKIN outputs represent species concentrations as a function of time, whilst the axis of the tree in Figure 5 can be thought of as denoting the changing temperature parameter. It follows that if we now combine a number of datasets from the tree, the composite result will represent concentrations as a function of time *and* temperature. We achieve this using a module called DaisyChain, which has been written to concatenate recalled lattices. The user moves along the tree specifying *retrieve* events for the required datasets, and once the traverse is completed a Switch is opened to pass the data into a contour module (Figure 6).

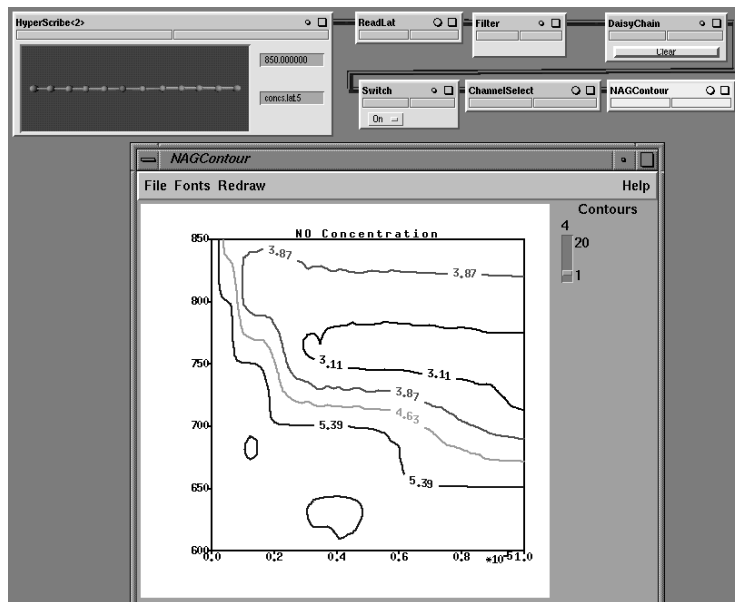


Fig. 6. Concatenating data sets to make a 2D plot (NB. the ‘Filter’ module in this map is just a collection of three standard modules grouped together for greater clarity)

3.3 Results

Figure 6 shows the contour plot of NO concentration resulting from the original experiment specifying temperatures at 50K intervals. The area of greatest interest is a region of low NO concentration (and correspondingly high NO_2 concentration) lying between 700 and 800K, which is worthy of investigation at a greater resolution of the temperature parameter. However, no data is available so HyperScribe is used to reset the SENKIN module parameters for the 700K experiment. The temperature is increased to 710K, new data is generated at 10K intervals and is recorded by HyperScribe alongside the original data. To reinforce the idea of having re-run the simulation in order to insert data, the new data sets are recorded as a branch rooted at the original 700K set. The process is repeated from 800K and once the requisite area of data has been ‘filled in’, the tree is traversed again to create the composite data for NAGContour, but this time including the branched data. Figure 7 shows the new tree and set of contours, with the minimum of NO concentration now clearly pinpointed.

4 Conclusions and further work

An architecture for problem solving support within a dataflow environment has been proposed and an implementation demonstrated in the IRIS Explorer MVE. Results from the case study show that the resulting module is readily incorporated into the visualization pipeline and that complex sequences of parameters and results can be managed using it.

Future work will include developments to HyperScribe, such as a facility to

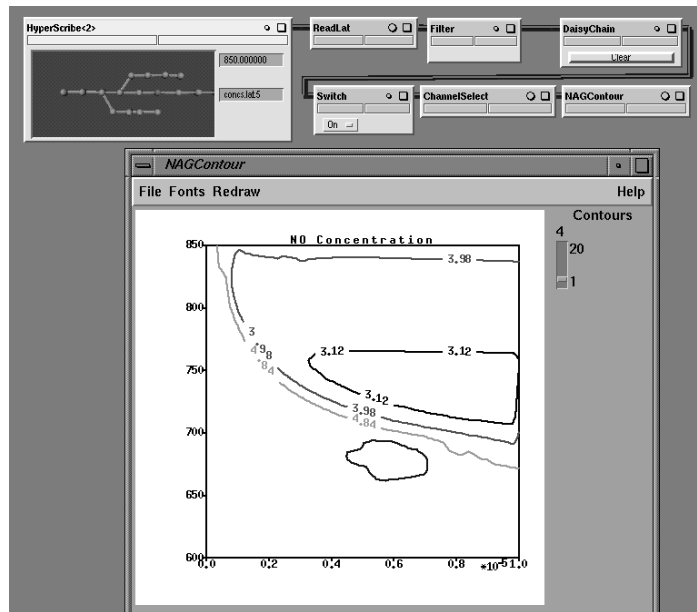


Fig. 7. A 2D plot with new data between 700 and 800K

delete stored information, and the addition of further tools like the DaisyChain module to collate data sets retrieved from the tree. We are also working with the COVISA project [16] at the University of Leeds, which is developing collaborative visualization facilities, and are investigating the potential of paradigms other than the history tree for problem solving.

Acknowledgements

Helen Wright gratefully acknowledges the support of NAG Ltd in carrying out this work. We are also indebted to colleagues in the School of Chemistry at the University of Leeds, in particular Professor Mike Pilling for his interest and encouragement; to the IRIS Explorer team at NAG Ltd, especially Jeremy Walton for assistance with technical aspects; and to British Gas plc for permission to use the case study. Finally, many people contributed to the GRASPARC project, without which the present work would have had no proper foundation.

References

1. Gordon Cameron. Modular visualization environments: Past, present and future. *Computer Graphics*, 29(2):3–4, 1995.
2. Hambleton D. Lord. Improving the application development process with modular visualization environments. *Computer Graphics*, 29(2):10–12, 1995.
3. David Foulser. IRIS explorer: A framework for investigation. *Computer Graphics*, 29(2):13–16, 1995.

4. Greg Abram and Lloyd Treinish. An extended data-flow architecture for data analysis and visualization. *Computer Graphics*, 29(2):17–21, 1995.
5. Mark Young, Danielle Argiro, and Steven Kubica. Cantata: Visual programming environment for the Khoros system. *Computer Graphics*, 29(2):22–24, 1995.
6. R. Marshall, J. Kempf, S. Dyer, and C. Yen. Visualization methods and simulation steering for a 3D turbulence model for Lake Erie. *ACM SIGGRAPH Computer Graphics*, 24(2):89–97, 1990.
7. G.D. Kerlick and E. Kirby. Towards interactive steering, visualization and animation of unsteady finite element simulations. In G.M. Nielson and D. Bergeron, editors, *Proceedings of IEEE Visualization 1993 Conference*, pages 374–377, Los Alamitos, CA, 1993. IEEE Computer Society Press.
8. C. Upson, T. Faulhaber, D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz, and A. van Dam. The Application Visualization System : A computational environment for scientific visualization. *IEEE Computer Graphics and Applications*, 9(4):30–42, 1989.
9. R.B. Haber and D.A. McNabb. Visualization idioms : A conceptual model for scientific visualization systems. In B. Shriver G.M. Nielson and L.J. Rosenblum, editors, *Visualization in Scientific Computing*, pages 74–93. IEEE, 1990.
10. Ken Brodlie, Lesley Brankin, Greg Banecki, Alan Gay, Andrew Poon, and Helen Wright. GRASPARC: A problem solving environment integrating computation and visualization. In G.M. Nielson and D. Bergeron, editors, *Proceedings of IEEE Visualization 1993 Conference*, pages 102–109, Los Alamitos, CA, 1993. IEEE Computer Society Press.
11. H. Wright, G.A. Stead, and K. W. Brodlie. Interactive exploration of chemical reaction mechanisms using novel visualization and integration techniques. In M. Göbel, H. Müller, and B. Urban, editors, *Visualization in Scientific Computing*, Eurographics, pages 166–173. Springer-Verlag, 1995.
12. Ken Brodlie and Helen Wright. From a Modular Visualization Environment to an environment for computational problem solving. *Computer Graphics*, 29(2):29–32, 1995.
13. H. Wright and J.P.R.B. Walton. HyperScribe: A data management facility for the dataflow visualization pipeline. IRIS Explorer Technical Report IETR/4, NAG Ltd, 1996.
14. M.J. Brown, D.P. Graham, B. Strugnell, and R.M. Davies. Environmental impact of gas turbine CHP systems. In *Proceedings of 1995 International Gas Research Conference*, pages 491–500. Vol. V Industrial Utilisation, 1995.
15. R.J. Kee and J.A. Miller. A structured approach to the computational modelling of chemical kinetics and molecular transport in flowing systems. Technical Report SAND86-8841, Sandia National Laboratories, Livermore, California, Reprinted 1993.
16. Jason Wood, Helen Wright, and Ken Brodlie. CSCV - Computer Supported Collaborative Visualization. In *Proceedings of BCS Displays Group International Conference on Visualization and Modelling*, 1995.