

Computational Steering by Direct Image Manipulation

F. Chatzinikos and H. Wright

University of Hull, Department of Computer Science
Simulation and Visualization Research Group
Cottingham Road, HU6 7RX, Hull, UK
Email: F.Hadginikos@dcs.hull.ac.uk
H.Wright@dcs.hull.ac.uk

Abstract

Computational steering requires the coupling of simulation and visualization elements, but if the latter is targetted at general requirements, little or no information about the calculation itself may survive in the final image. Consequently, changes made there cannot, in general, propagate back to the source. For example, in a study of a chemical reaction, a line on a graph simply consists of linked pairs of x and y coordinates, with no indication that these denote the concentrations of, say, oxygen or hydrogen at certain times. This paper will introduce a new visualization taxonomy and data structure which allow changes in the simulation to be accomplished by direct image manipulation, allowing more intuitive steering of a range of scientific applications.

1 Introduction

Advances in computer processing power over the past few years have brought a significant change to the modelling and simulation of complex phenomena. Problems that formerly could only be tackled in batch mode, with their results visualized afterwards, can now be monitored whilst in progress using graphical means – in certain cases it is even possible to alter parameters of the computation whilst it is running, depending on what the scientist sees in the current visual output. This ability to monitor and change parameters of the computational process at any time is called computational steering.

Johnson et al [1] describe three broad approaches to computational steering, which they term instrumentation, directed computation, and dedicated implementation. The last of these is only available when a system is being built from scratch, where the effort of doing so is repaid by the improved cou-

pling of computational and graphical components that can then be achieved. Commonly, though, the requirement is to re-use some existing code in conjunction with a separate visualization element. Sometimes the code can be broken down into components and the whole arrangement directed using a scripting language, or it may be possible to insert function calls at strategic points which deliver parameter values or collect results. In contrast to a dedicated implementation, the systems integration effort involved in these approaches is minimal. However, if the visualization element in such a steering arrangement has been provided with general requirements in mind, then the interaction with the computational component is mostly limited to altering a set of dials or sliders arranged on its Graphical User Interface (GUI), quite separate from the image showing the results. The goal of this project is to address this deficiency by enabling computational steering via direct image manipulation within an existing, general purpose visualization system, thereby combining some of the advantages of a dedicated implementation with the low cost approach of re-using code.

A summary of current computational steering and visualization models follows, as well as some previous work on improving these models. Our own proposal is then described, together with a case study to illustrate the salient points of our approach. We conclude with some possible directions for future work.

1.1 Models for Visualization and Steering

The lack of coupling between computational and graphical elements described above is not difficult to diagnose, since the process of visualization involves progressive manipulation of data to

produce an image. If we adopt the classic Haber and McNabb [2] description of visualization, then the transformations that the data undergoes can be classified as filter, map and render processes (Figure 1). During filtering, the data is sampled to

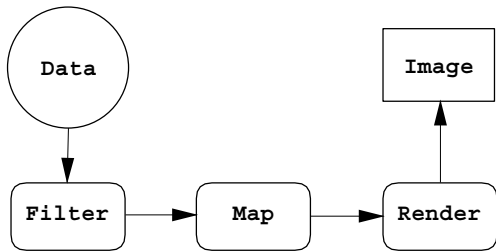


Figure 1: Haber & McNabb visualization pipeline

reduce its quantity, or interpolated to fill in between discrete points. In the mapping phase, an abstract geometrical object representing the data is generated, which in turn is rendered to produce an image. Commercial dataflow visualization systems implement the Haber-McNabb visualization pipeline explicitly as part of their user interface, but even in turnkey visualizers, object oriented systems, augmented dataflow and component based systems such as [3, 4, 5, 6], the process of visualization still involves the successive transformation of data in order to create the final rendered image. Not only can valuable information be lost at each of these stages but the initial simulation data first entering the visualization process may already have been stripped of much of its semantic content, if systems are aimed at satisfying general requirements.

Figure 1 depicts visualization being used to post-process data, but Figure 2 shows the interaction available to the scientist using computational steering, where now the simulation code has been incorporated as described in section 1. As can be seen there, simulation, visualization and image manipulation are now all available but the scientist needs to change context in order to accomplish each one. Imagine the case where a scientist is examining the visualization results by looking at and rotating the rendered image. If they need to change a visualization parameter they have to switch attention to the visualization process, alter some dial or slider and then move back to the image to see the new results. Likewise, if a simulation parameter has to be steered, the scientist must switch to the simula-

tion process and then return to the render process to watch the effect. How much more compelling this interaction would become, if it could be accomplished entirely by working within the context of the image showing the results.

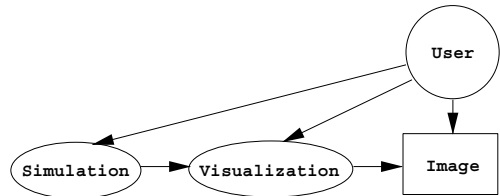


Figure 2: Current model of simulation and visualization interaction

1.2 Prior Work

Figure 3 shows the type of interaction envisaged in direct image manipulation, but it also shows a major obstacle to achieving this, namely the need somehow to reverse the transformation of the data as it passes from the simulation to the image. Overcoming this difficulty has prompted a variety

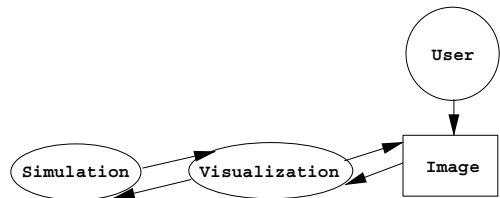


Figure 3: Proposed model for image based interaction

of approaches. AVS [7] and IRIS Explorer [8], for example, both use an augmented dataflow model supporting image probing facilities based on feedback loops – functions that accept a geometrical position, query the input data and return a value interpolated at the required point.

Certain visualization techniques are especially amenable to direct image manipulation to specify their parameters. Streamlines and particle tracks, for example, require seed points which are readily generated by placing a rake or bar in the image. Iso-surfaces and contour lines likewise can be drawn by placing a data probe in the image and generating the surface or line joining all points of the same

value. Collecting and using the necessary geometrical information to specify parameters in this way has led to the construction of the 3D widget software library described in [9].

Another example of this kind of functionality can also be found in [10] and in Amira [4], one of the most recent additions to the visualization armoury. Amira supports a set of visualization components such as cutting planes, streamline seed point initialisation and volume segmentation, all accessible directly via the image.

Felger and Schröder [11] considered the problem of image interaction in dataflow systems and suggested the separation of the visualization pipeline into two components: the visualization input pipeline (VIP) and the visualization output pipeline (VOP). The goal of the VIP was to reverse the effect of the VOP, the VOP being equivalent to the original, output-oriented visualization pipeline of Haber and McNabb.

This principle of a reversible pipeline was based on three techniques: the inverse function method, the look-up method, or the listening method. In the first, provided there is a unique, one-to-one mapping between an element of the input set and an element of the output set, an inverse module can be created which recreates the input from the output. In the second method the VIP module 'remembers' all the incoming and outgoing data of the VOP module and performs a look-up operation to find the required data. In the last method the VIP module forces the corresponding VOP module to re-execute, picking up the input data when it detects output data matching that needed. The authors used their methods to demonstrate image probing similar to that now implemented using shared memory and dataflow feedback loops in AVS and IRIS Explorer, but in principle their approach could be used to reverse each and every stage of the visualization pipeline.

Mulder and van Wijk [12] took a different route, adopting a modular, as opposed to a dataflow, architecture, targetted specifically at computational steering. They suggested the use of 3D objects, parameterised to data variables coming from the simulation. In their design, PGOs (Parameterised Graphical Objects) are used both to visualize the output of the simulation and to steer it by manipulating the objects' characteristics. In contrast with dataflow systems, their data binding mechanism ensures di-

rect image manipulation, but the 3D objects chosen need not be an intrinsic part of the graphical representation, and each steerable graphic must be specified beforehand.

2 A Generic Approach

The goal of the present project, therefore, is to allow the scientist to interact with the simulation directly by manipulating the rendered image. Here we meet our next difficulty, since the question arises as to how to support a wide variety of computational problems with a generic visualization system. This is the same problem as already occurs when post-processing simulation data, but here its effect is magnified because of the need to supply input parameters as well.

Fortunately, many such problems can be visualized using only a modest number of well-used techniques. Our solution to this difficulty has therefore been to consider the problem primarily from a visualization standpoint, and to develop a taxonomy that classifies established graphical representation techniques in terms of the interactions they allow. Having enumerated all possible interactions for each of these techniques we can then, by means of a suitable data model, give these interactions meaning in the computational context.

2.1 Developing the Taxonomy

For example, consider the problem of visualizing flow within or around an object in two or three dimensions, such as the movement of oil past a ball bearing. Figure 4 shows the combination of streamlines and domain geometry in the same display, which allows us to steer both the initial flow of the oil as well as the bearing's geometry.

Looking at the figure, the following interactions are immediately obvious. Firstly, dragging the corners of the domain could signify a change in the allowable range of the independent variable, whereas changing the eccentricity of the bearing's profile is accomplished by moving boundary points in region A. Furthermore, the specification of a new flow direction is possible by dragging the initial direction of the streamline. Finally, since the colour of the streamline denotes the magnitude of the flow, its speed can be varied using a spray can.

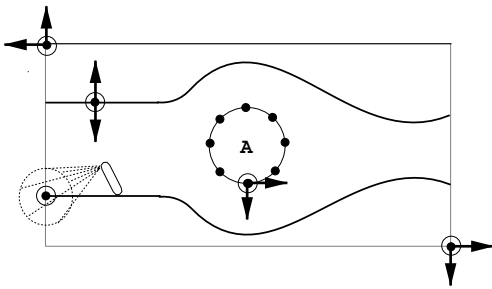


Figure 4: Combination of interactive streamlines and domain geometry, incorporating manipulators and colour spray can

Work such as that in [4] and [10] leads us to suggest that steering values in this way could be far more compelling than inputting numbers or dragging a dial. Just as an ordinary streamline seeded directly into the image can give an immediate understanding of the direction and speed of the incident flow, so our augmented streamline could be used to “turn the initial flow a few degrees left”, or to “make the oil flow a bit quicker”.

2.2 An Extended Brodlie Taxonomy

Our approach has therefore been to extend Brodlie’s taxonomy of visualization techniques [13], by identifying the possible interactions each technique can support (Table 1). If, then, we were interested in visualizing our data using a technique other than streamlines, we could use Table 1 to find one suitable for this type of problem. Such a technique would be the arrow plot, the augmented version of which would offer equivalent computational interaction to that afforded by the streamline technique. The only difference between the two techniques is the different interaction style when altering the speed of the flow. In the case of streamlines this is done by changing their initial colour, whereas in the case of the arrow plot, the length of the perimeter arrows is used instead.

2.3 Data Model for Semantic Information

Of course, the interactions in Table 1 are generic to the specific visualization technique chosen, so, as our example demonstrates, the activity only assumes significance for a particular computation

when it is presented in context. Also needed, therefore, is a data model with sufficient semantic content to describe a wide range of simulation data. This data model should provide for scalar, vector and tensor data types; upper and lower bounds; tolerances within which they were computed; informational fields, and an indication of their rôle in the computation be it as a dependent, independent or parameter variables.

Our preliminary data structure, which caters for interaction with single or multiple scalars defined over an independent variable space of arbitrary dimensions, is discussed here. This structure is composed of two main components: the data and the coordinates portion. Looking first at the coordinates portion, there is provision for describing the independent variable domain by means of its corners (minimal specification), by specifying points along its boundaries (perimeter specification), or by giving the coordinates of each data point explicitly (explicit specification); Figure 5 (a), (b), and (c) respectively show the resulting interaction points for a 2D domain.

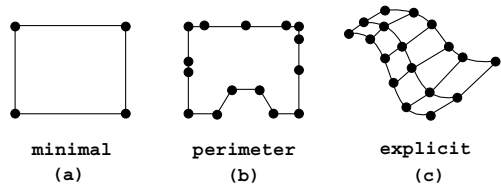


Figure 5: Different coordinate types and their active points

Interaction with each of Figure 5’s active points gives the opportunity to change the extents of the problem domain and possibly the mesh over which the problem is solved. Each of these types has different storage needs. For example, eight values would be needed to describe a 1D problem using the minimal method – these would be the minimum, maximum and current values for both the start and the end point of the x-axis, together with a flag indicating whether or not they can be steered. This highly compressed description would not, however, allow access to the individual values of the independent variable.

Next we consider the dependent variables occupying the data portion, which may consist of 1-, 2- or 4-byte integers, floats or doubles. Visualization

Table 1: Visualization taxonomy incorporating interaction (after Brodlie et al [13])

Problem Type	Visualization Technique	Dependent Variable Interaction	Independent Variable Interaction
1D, Scalar	Line Graph	Drag intersection with Y axis (position and slope).	Drag points in 1D
2D, Scalar	Surface View	Drag intersection with Z axis (position and gradient)	Drag points in 2D
	Image Plot	Edit perimeter colour	
2D(3D), Vector	Arrows	Drag length and direction of perimeter arrow	Drag points in 2D (3D)
	Streamlines coloured by magnitude	Drag initial direction and edit colour	
3D, Scalar	Volume plot	Edit visible surface colour	Drag points in 3D

ordinarily requires just the data values across the problem domain to be provided, however, our data model also includes derivative information to enable support of a variety of initial value and boundary value problems. In common with the coordinates structure, a vector type is used to identify which of the variables are interactive and what are their current values and bounds. The two portions of the description are then combined using a container device.

It should be noted how the data structure makes no reference to the graphical technique to be used. This dissociation is an important feature of our design, as whichever technique is finally employed simply reads this structure and brings its available interactions to bear on the relevant parts. For example, data which can be visualized using either an image plot or a surface view, and which carries the requirement to steer perimeter values, will result in the first case in a spray can appearing to change the perimeter colour, and in the second a drag manipulator. Both will have the same effect when the new initial values are returned to the computational element in order to re-run the simulation.

3 Case Study

A demonstrator based on our data structure and an augmented line graph technique has been constructed using IRIS Explorer, in order to test whether our design is feasible. Using IRIS Explorer

allows us to re-use software wherever possible, but at the same time to implement our own modules and data types when necessary. Additionally, the visual output of this environment is based on Open Inventor [14], a widely available, well-documented and extensible graphical toolkit.

3.1 Line Graph Interaction

Our demonstrator is based on a time-varying chemical simulation which is described in the next section. This type of problem is usually visualized using a line graph technique. In this case the line graph (Figure 6) will be used to steer an initial value problem in one dimension, with the independent variable domain specified using the minimal type described earlier. Dragging on the x-axis end points could therefore signify a change in the start or stop time of the integration. Furthermore, if we revisit Table 1 we can see that there are two possible dependent variable interactions. Dragging the plotted values up and down the y-axis allows the specification of a new initial value for one of the chemical reactants. In the case where a reactant has a specified initial rate of change, the user would be able to alter the direction of the line's tangent vector, but our computational component does not support this interaction.

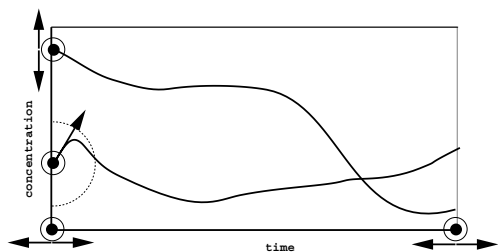


Figure 6: Line graph showing interactors for domain steering and dependent variable position and slope

3.2 Computational Component

The chosen study is an investigation of the spontaneous ignition of a hydrogen-air mixture (H_2 and O_2 diluted with N_2) at constant pressure. The computational component used is based on the SENKIN program for gas phase kinetics [15], developed by Sandia National Laboratories as part of their CHEMKIN system. Originally, SENKIN received its input from a file containing several different keywords and their values. In this case study, SENKIN implemented as an IRIS Explorer module is used instead [16], with an interface consisting of text fields and dials replacing the original keyword file. This interface enables the scientist to choose and alter several different parameters of the simulation such as the integration stop time, ambient temperature, pressure, and the composition of the initial gas mixture.

The following values were chosen for SENKIN's initial execution:

- Constant pressure equal to 1 atmosphere
- Initial temperature of 1000 Kelvin
- Integration time of 0.0004 seconds
- Concentrations of H_2 and O_2 (in mole fractions) respectively 0.27 and 0.14, with N_2 added to make up a single mole of substance, i.e. 0.57 initially

Of these parameters, the H_2 and O_2 concentrations were to be steered whilst the remainder were fixed at their initial values or, in the case of nitrogen, calculated as described above once the other inputs were known.

3.3 Steering Support

The output of the SENKIN module is an array of data containing reactant concentrations for every time point computed from the start of the simulation until the end. This output, in the form of a standard IRIS Explorer lattice, is connected in turn to a graph module which converts the data to geometry and passes it to a render module to draw the graph on the screen.

To test our proposed support for image based steering, SENKIN's output was converted to the augmented data structure before being passed to a custom-built graph module. The purpose of this module is to read the augmented structure, identify the interactive elements and insert special information nodes in the Open Inventor geometry description, or scene graph. This enables the render module further down the pipeline to identify those elements that are interactive and display the appropriate manipulators.

After the first run of the simulation, the render window displays the initial output (Figure 7). This image looks like a normal line graph but in fact contains the semantic data that will enable the user to interact with it. Note that for clarity we have chosen to display the concentrations of only three reactants in the graph. These are the O_2 , H_2 and H_2O concentrations, represented respectively by the dotted, unbroken, and dashed line. As we note above, of these three reactants only the first two can be steered, since H_2O is a by-product of the combustion and is not part of the initial gas mix. At this point the scientist can start to interact with the rendered image to set new conditions for the simulation. Referring back to Figure 6 we see that a line graph allows four kinds of interaction. All of these are supported by our graph module but only three are needed for this particular type of chemical problem. With respect to the problem's independent variables, the integration range can be changed by clicking on the x-axis and moving its end point, and the simulation can also be restarted at non-zero time by clicking on the x-axis starting point and moving it to the right; the computed concentrations at the particular time point chosen are then used as initial conditions for a new simulation. For the dependent variables, the scientist can move the starting position of a line in the graph up or down the y-axis to input a new initial value for that particular reactant, provided it is a component of the initial gas mix.

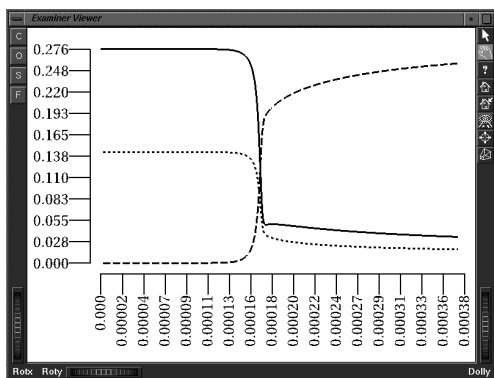


Figure 7: Initial graph showing three reactant concentrations over time. O_2 is represented by the dotted line, H_2 by the unbroken line and H_2O by the dashed line

However, they cannot specify a line's initial slope, since this type of information is not accepted by the simulation.

All of this information is conveyed by the augmented data structure, such that only the allowed interactions are made available to the scientist. Figure 8 (a) shows how, with image steering support, the scientist simply clicks on the O_2 line in the graph to display a drag manipulator, showing that this line can be moved both up and down. The slope manipulator is not shown, however, since this interaction is not allowed. Dragging the manipulator to a new position on the y-axis causes the system to compute the relative change in the position of the manipulator and re-execute the simulation using the new values specified. Clicking on the H_2 line likewise will bring up a manipulator, but the H_2O line has no such interaction specified in the data structure for the reasons described above. Accordingly, this and any other by-product of the reaction can be plotted, but none will react to the scientist's attempts to select them for steering. Figure 8 (b) shows the interactor responsible for domain steering. Using this interactor the scientist can increase or decrease the integration range of the simulation.

4 Conclusions and Further Work

We have presented a scheme that allows computational steering by direct image manipulation, supported by a collection of appropriate visualization

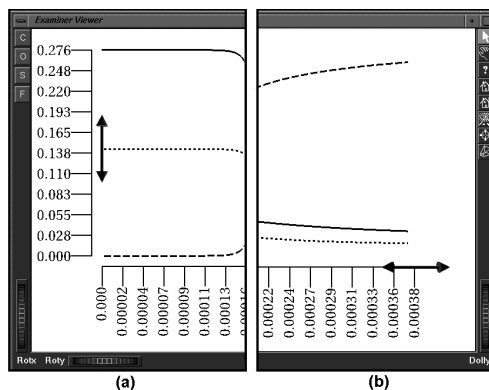


Figure 8: O_2 is represented by the dotted line, H_2 by the unbroken line and H_2O by the dashed line. (a) O_2 selected for interaction. (b) domain selected for interaction

techniques and a comprehensive, yet extensible, data structure which captures the computational context. To demonstrate our ideas, a case study has been described from the field of chemical kinetics. The success of this has shown that our initial design is feasible but it has also raised some suggestions for further work.

Firstly, a potential problem exists with the precision of our technique, as there is no means for the scientist to fine-tune their adjustments. A solution to this problem would be to use an interactor that contains a text field that shows the current value of the manipulated variable, and allows keyboard input when required.

Another issue is how to provide steering support for variables that are not visible in the final image. These can be parameter variables, such as the temperature and pressure in our chemical simulation, or other data that have undergone so many transformations that they are difficult to visualize. Displaying dials or sliders as part of the rendered image can solve this problem, and would be preferable to having these on a separate GUI. In addition, since all the interactive elements would be specified in the data structure, the widgets required in the rendered image would be automatically determined.

Possible interdependence of the computational variables should also be considered. This situation arose in the case study, where the amount of nitrogen in the gas mix was determined by the computational code after the hydrogen and oxygen in-

puts were known. Had this information been available to the line graph, the nitrogen plot could have been shown to go up or down as the scientist varied the input oxygen concentration. We intend to address this requirement by allowing the data structure to describe a functional specification of some variables in terms of others.

Finally, the chemistry problem presented here is just one of several we intend to study during the remainder of the project. This will enable us to validate both our taxonomy and data model, ensuring that they will support a range of computational problems arising from a variety of fields.

5 Acknowledgements

This work has been carried out at the University of Hull as part of the project 'Computational Interaction – Realising the Potential of Visualization', supported by the UK Engineering and Physical Sciences Research Council, grant number GR/M96094. We would also like to thank Sandia National Laboratories for provision of their CHEMKIN suite of programs and associated test data, and we are most grateful to Prof R Phillips and Mr D P M Wills of the Department of Computer Science at the University of Hull for helpful discussions. Finally, we would like to thank the reviewers of this article for their valuable comments.

References

- [1] C. Johnson, S. G. Parker, C. Hansen, G. L. Kindlmann, and Y. Livnat, IEEE, December 1999, "Interactive Simulation and Visualization".
- [2] R. B. Haber and D. A. McNabb, Visualization and Scientific Computing, IEEE Computer Society Press, 1990, pp 74-93, "Visualization Idioms: A Conceptual Model for Visualization Systems".
- [3] PV-WAVE Development Environment, 2001, <http://www.vni.com/products/wave/specs.html>
- [4] Amira - an Advanced 3D Visualization and Volume Modeling System, 2001, <http://www.amiravis.com>
- [5] J. Davison de St. Germain, J. McCorquodale, S. G. Parker, C. R. Johnson, Ninth IEEE International Symposium on High Performance Distributed Computing, August 2000, "Uintah: A Massively Parallel Problem Solving Environment".
- [6] Los Alamos National Laboratory, California, USA, 2001, "Efficient coupling of parallel applications using Parallel Application WorkSpace", <http://www.acl.lanl.gov/PAWS>
- [7] H. D. Lord, Computer Graphics, 29(2), 1995, pp. 10-12, "Improving the Application Development Process with Modular Visualization Environments".
- [8] D. Foulser, Computer Graphics, 29 (2), 1995, pp 13-16, "IRIS Explorer: A Framework for Investigation".
- [9] D. B. Conner, S. S. Snibbe, K. P. Herndon, D. C. Robbins, R. C. Zeleznik and A. van Dam, Computer Graphics, 25(2), ACM SIGGRAPH, March 1992, pp. 183-188, "Three-Dimensional Widgets".
- [10] P. Hastreiter and T. Ertl, Proc. of Img. and Multidim. Dig. Sig. Process. (IMDSP), 1998, pp 41-44, "Fast and Interactive 3D-Segmentation of Medical Volume Data".
- [11] W. Felger and F. Schröder, Proceedings of Eurographics 92, Eurographics Association, 1992, pp 139 - 151, "The Visualization Input Pipeline - Enabling Semantic Interaction in Scientific Visualization".
- [12] J. D. Mulder and J. J. van Wijk, Proceedings of IEEE Visualization 95, IEEE Computer Society Press, 1995, pp 304 - 311, "3D Computational Steering with Parametrized Graphical Objects".
- [13] K. W. Brodli, L. A. Carpenter, R. A. Earnshaw, J. R. Gallop, R. Hubbolt, A. M. Mumford, C. D. Osland, P. Quarendon (eds), Springer-Verlag, 1992, pp 37-85, "Scientific Visualization: Techniques and Applications".
- [14] J. Wernecke, Addison-Wesley, 1993, "The Inventor Mentor".
- [15] A. E. Lutz, R. J. Kee and J. A. Miller, Sandia Report: SAND87-8248, 1987, "SENKIN: A Fortran Program for Predicting Homogeneous Gas Phase Chemical Kinetics With Sensitivity Analysis".
- [16] IRIS Explorer Centre of Excellence, <http://www.comp.leeds.ac.uk/iecoe/Products/kinex/kinex.html>