SCIRun: Application to Atmospheric Dispersion Problems Using Unstructured Meshes.

C.R. Johnson[1], **M. Berzins**[2], **L. Zhukov**[1], **and R. Coffey**[1]

[1]*Dept. of Computer Science, University of Utah, Salt Lake City, Utah, USA*
[2]*CPDE Unit, Computer Studies, University of Leeds, Leeds LS2 9JT,UK*

Abstract

We present an overview of SCIRun, a scientific problem solving environment that allows the interactive construction, debugging, and steering of large-scale scientific computations. Using this "computational workbench," a scientist can design and modify simulations interactively via a dataflow programming model. SCIRun enables scientists to design and modify model geometry, interactively change simulation parameters and boundary conditions, and interactively visualize geometric models and simulation results. We show an integrated prototype problem solving environment for computational fluid dynamics may be constructed by combining SCIRun with the Tetrad/CSPRINT codes. The success of the approach is illustrated by using the resulting code to solve an atmospheric diffusion example consisting of time dependent p.d.e.s on unstructured tetrahedral spatial meshes.

1 Introduction.

In recent years, the scientific computing community has experienced an explosive growth in both the possible size and the possible complexity of numeric computations. One of the significant benefits of this increased computing power is the ability to perform complex three-dimensional simulations. However, such simulations present new challenges for computational scientists. How does one effectively analyze and visualize complex 3D data? How does one solve the problems of working with very large datasets often consisting of tens to hundreds of gigabytes? How does one provide tools that address these computational problems while serving the needs of scientific users? These problems are particularly acute in the area of transient three-dimensional reacting flows in which it is possible to generate very large data sets purely because of the numbers of chemical species involved and the time-dependent nature of the problem.

Scientific visualization clearly plays a central role in the analysis of data generated by scientific simulations. Unfortunately visualization is often performed only as a mystical post-processing step after a large-scale computational batch job is run. For this reason, errors invalidating the results of the entire simulation may be discovered only during post-processing. What is more, the decoupling of simulation and visualization presents serious scientific obstacles to the researcher. This issue was addressed by the 1987, Visualization in Scientific Computing (ViSC) workshop which concluded that scientists not only want to analyze data, they also want to interpret what is happening to data and want to *steer* calculations in real-time by changing parameters, resolution or representation. This *interactive visual computing* is a process whereby scientists communicate with data by manipulating its visual representation during processing. The more sophisticated process of *navigation* allows scientists to *steer*, or dynamically modify computations while they are occurring. These processes are invaluable tools for scientific discovery.

Although these thoughts were reported more than ten years ago, they express a simple and still current idea: scientists want more interaction than is currently present in most simulation codes. While the scientific computing community is still trying to find better ways to address these needs, we feel that the problems encountered by computational scientists encompass a wider range of issues, including but not restricted to scientific visualization. In this paper, we give an overview of the SCIRun[1][11] problem solving environment and focus on incorporating an unstructured tetrahedral mesh solver, called Tetrad, [3], and use it with the CSPRINT time integration software [1] within SCIRun to solve a transient atmospheric diffusion problems with multiple species on an unstructured adaptive tetrahedral mesh. In doing so we will demonstrate the applicability of SCIRun to complex multi-species multidimensional transient flows.

2 SCIRun - a Computational Steering System

SCIRun is a scientific programming environment that allows the interactive construction, debugging and steering of large-scale scientific computations [8, 7, 6]. SCIRun can be envisioned as a "computational workbench," in which a scientist can design and modify simulations interactively via a dataflow programming model. SCIRun enables scientists to modify geometric models and interactively change numerical parameters and boundary conditions, as well as to modify the level of mesh adaptation needed for an accurate numerical solution. As opposed to the typical "off-line" simulation mode - in which the scientist manually sets input parameters, computes results, visualizes the results via a separate visualization package, then starts again at the beginning - SCIRun "closes the loop" and allows interactive steering of the design, computation, and visualization phases of a simulation.

The dataflow programming paradigm has proven useful in many applications. In the scientific community, it has been successfully applied in several scientific visualization packages, including AVS from Advanced Visual Systems Inc. and Iris Explorer from NAG. We have extended the use of the dataflow programming model into the computational pieces of the simulation. To make the dataflow programming paradigm applicable to large scientific problems, we have identified ways to avoid the excessive memory use inherent in standard dataflow

[1]SCIRun is pronounced "ski-run" and derives its name from the Scientific Computing and Imaging (SCI) research group which is pronounced "ski" as in "Ski Utah."

implementations, and we have implemented fine-grained dataflow in order to further promote computational efficiency.

2.1 An Iterative Environment for Scientific Computing

Currently, the typical process of constructing a computational model consists of creating (modifying) a discretized geometric model and its associated mathematical model. Once the solution is computed results are analyzed using a separate visualization package and the cycle repeated.

The "art" of obtaining valuable results from a model has up until now required a scientist to execute this process time and time again. Changes made to the model, input parameters, or computational processes are typically made using rudimentary inefficient tools (text editors being the most common). Ideally, a system should link all these computational components so that all aspects of the modeling and simulation process could be controlled graphically within the context of a single application program. This is not the current standard of scientific computing because of the difficulties in creating such a unified program. These difficulties arise from the need to integrate a wide range of disparate computing disciplines (such as user interface technology, 3D graphics, parallel computing, programming languages, and numerical analysis) with a wide range of equally disparate application disciplines (such as medicine, meteorology, fluid dynamics, geology, physics, and chemistry). Our approach to overcoming these difficulties is to separate the components of the problem. SCIRun's dataflow model employs "modules" that can be tailored for each application or computing discipline in a single unified framework. In this paper we will show how modules in time integration and unstructured mesh computational fluid dynamics have been linked with SCIRun.

2.2 Steering

SCIRun is an integrated problems solving environment in which allows the user to interactively control scientific simulations while the computation is in progress [10, 9]. This control allows the user to vary model or computational parameters during simulation. SCIRun is designed to provide high-level control over parameters in an efficient and intuitive way, through graphical user interfaces and scientific visualization. These methods permit the scientist or engineer to "close the loop" and use the visualization to steer phases of the computation in an advantageous way. As changes in parameters become more instantaneous, the cause-effect relationships within the simulation become more evident, allowing the scientist to develop more intuition about the effect of problem parameters, to detect program bugs, to develop insight into the operation of an algorithm, or to deepen an understanding of the physics of the problem(s) being studied.

2.3 Requirements of SCIRun as a Computational Steering System

SCIRun was designed to solve specific problems in Computational Medicine, but has been since extended to other computational science and engineering problem domains. In attacking the specific problems, we found that there were a wide range of disparate demands placed on such a system. Each of these demands reveals a different facet of what we call SCIRun.

2.4 SCIRun the Operating System

In a sophisticated simulation, each of the individual components (modeling, mesh generation, nonlinear/linear solvers, visualization, etc.) typically consumes a large amount of memory and CPU resources. When all of these pieces are connected into a single program, the potential computational load is enormous. In order to use the resources effectively, SCIRun adopts a role similar to an operating system in managing these resources. SCIRun manages scheduling and prioritization of threads, mapping of threads to processors, inter-thread communication, thread stack growth, memory allocation policies, and memory exception signals (such as segmentation violations).

2.5 SCIRun the Scientific Library

SCIRun uses a visual programming interface to allow the scientist to construct simulations through powerful computational components. While the visual programming environment is the central focus of SCIRun, it requires a powerful set of computational tools. In the first stage of SCIRun, we have concentrated on integrating the computational components that we have used to solve our own computational problems. Such tools Delaunay 3D tetrahedral mesh generators and mesh adaptation routines, direct and iterative linear and nonlinear equations solvers and finite element space discretisation routines, see [8]. We have recently expanded focus and are now in the process of integrating popular libraries and tools, see [8] into the SCIRun environment, much as will be shown here with TETRAD in Section 2.7 below.

2.6 SCIRun the Development Environment

Perhaps the most powerful facet of SCIRun is the ability to use it in the development phases of a simulation. SCIRun augments the development environment by providing convenient access to a powerful set of computational components. However, these components could never be comprehensive, so SCIRun also provides an environment whereby new modules can be developed efficiently. If a module fails, SCIRun stops the module at the point of error, thus allowing the developer to attach a debugger to the program at the point of failure. This avoids the frustrating experience of trying to reproduce these errors in the debugger. In addition, SCIRun provides simple instrumentation of module performance (CPU times printed out interactively), feedback execution states (waiting for data, percent completed, etc.), and visualization of memory usage. SCIRun employs dynamic shared libraries to allow the user to recompile only a specific module without the expense of a complete re-link. Another SCIRun window contains an interactive prompt which gives the user access to a Tcl shell that can be used to interactively query and change parameters in the simulation.

2.7 Application Requirements–An Atmospheric Dispersion Example

SCIRun is not magic – it is simply a powerful, expressive problem solving environment for constructing steerable applications, either from existing applications or starting from the ground-up. The application programmer must assume the responsibility of breaking up an application into suitable components. More importantly, it is the responsibility of the application programmer to ensure that parameter changes make sense with regard to the underlying problem physics.

The application we consider here is taken from a model of atmospheric dispersion from a power station plume - a concentrated source of NO_x emissions, [2]. The photo-chemical reaction of this NO_x with polluted air leads to the generation of ozone at large distances downwind from the source. An accurate description of the distribution of pollutant concentrations is needed over large spatial regions in order to compare with field measurement calculations. The present trend is to use models incorporating an ever larger number of reactions and chemical species in the atmospheric chemistry model. The complex chemical kinetics in the atmospheric model gives rise to abrupt and sudden changes in both space and time in the concentration of the chemical species in both space and time. These changes must be matched by changes in the spatial mesh and the timesteps if high resolution is required, [4]. The difference in time-scale between the reaction of these species leads to stiff systems of equations which require implicit numerical solvers and special linear equations solvers [1]. The requirements of such a problem are that it is necessary to combine:

- Unstructured tetrahedral mesh generation and adaptation.
- Physically realistic spatial discretisation methods.
- Stiff ode integrators tailored to the application.
- Fast interactive visualization for multi-species flows
- Computational steering facilities for transient problems.

These requirements were met by combining the SCIRun software with the spatial discretisation, mesh adaptation and time integration codes CSPRINT and TETRAD [1, 3] and by wrapping these codes in SCIRun modules, with converters to map to the SCIRun data structures, see Section 5 below.

3 SCIRun Dataflow System

SCIRun composes computational and visualization algorithms with these data elements using a dataflow style "boxes and wires" approach. An example of the atmospheric diffusion dataflow network using Tetrad is shown in Figure 1. The basic features of the SCIRun dataflow system are summarized below.

- A module represents an algorithm or operation, is drawn as a box in the network, with input and output ports to define its external parameters.

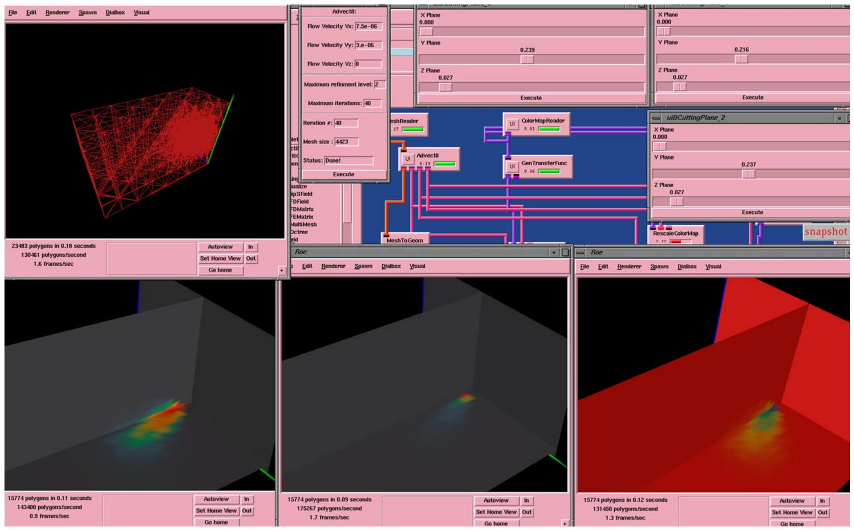
Advect8:
Flow Velocity Vx: 7.5e-06
Flow Velocity Vy: 3.e-06
Flow Velocity Vz: 0
Maximum refinment level: 2
Maximum iterations: 40
Iteration #: 40
Mesh size : 4423
Status: Done!

Figure 1. A dataflow network, showing modules (boxes), connections (wires) and the i/o ports on the modules that the wires connect). The plume and mesh shown are in the early stages of development.

- A port provides a connecting point for routing data to different modules. Ports are typed: each datatype has a different color, and datatypes cannot be mixed. In SCIRun, ports can be added to and removed from a module dynamically. Input ports are represented on the top of the module icon, and output ports are on the bottom. Output ports can cache (user-defined) datasets to avoid recomputation.
- A datatype represents a concept behind the numbers. Datatypes are quantities such as scalar fields or matrices, but are not the specific representations, such as unstructured grids, or sparse matrices, etc.
- A connection connects two modules together and thus controls where data is sent within SCIRun. The output port of one module is connected to the input port of another module. Output ports can be connected to multiple input ports, but input ports accept only a single connection.
- A network consists of a set of modules and the connections between them. This represents a complete dataflow "program."

The dataflow library is responsible for deciding which modules need to be executed and when. A module is typically re-executed when the user changes a module parameter, when new data is available at an input port, or when data is required at its output port. The SCIRun scheduler analyzes the dataflow graph to determine what other modules will also need to be executed. The modules actually communicate directly with each other using a threadsafe FIFO for the dataset handoffs. A centralized scheduler is used in order to avoid redundant module re-execution in a branching situation. Since we leave the central scheduler out of the loop for individual dataset handoffs, it does not become a bottleneck in the execution of the program.

The Dataflow library also contains a base class from which all modules are derived. This class contains the data structures that are required to implement the dataflow structures; it also contains various utility functions, such as `update_progress`, a function that the module writer can call periodically to update the graph on the module icon that indicates the approximate percentage of work the module has completed.

4 Steering in a Dataflow System

All of the pieces described above have been designed to support steering of large scale scientific simulations. SCIRun uses three different methods to implement steering in this dataflow oriented system:

1. Direct lightweight parameter changes. For example the TETRAD interface module allows the user to change the tolerance used for remeshing and and the number of refinement levels used. even while the module is executing.
2. Cancellation. When parameters are changed, the module can choose to cancel the current operation. For example, if boundary conditions are changed, it may make sense to cancel the computation in order to focus on the new solution. This makes the most sense when solving elliptic problems, since the solution does not depend on any previous solution.
3. Feedback loops in the dataflow program. For a time varying problem, the program usually goes through a time stepping loop with several major operations inside. The boundary conditions are integrated in one or more of these operations. If this loop is implemented in the dataflow system, then the user can make changes in those operators which will be integrated on the next trip through the loop.

These three methods provide the mechanisms whereby computational parameters can be changed during the execution of the program.

5 Atmospheric Diffusion Equation Example

The power plant plume application outlined in Section 2.7 is modelled by the atmospheric diffusion equation in three space dimensions given by:

$$\begin{aligned}\frac{\partial c_s}{\partial t} = -\frac{\partial u c_s}{\partial x} - \frac{\partial v c_s}{\partial y} - \frac{\partial w c_s}{\partial z} + D(\frac{\partial c_s}{\partial x}, \frac{\partial c_s}{\partial y}, \frac{\partial c_s}{\partial z}) \\ + R_s(c_1, c_2, ..., c_q) + E_s - (\kappa_{1s} + \kappa_{2s}) c_s, \qquad (5.1)\end{aligned}$$

where c_s is the concentration of the s'th compound, u,v and w, are wind velocities, K_x and K_y are diffusivity coefficients and k_{1s} and k_{2s} are dry and wet deposition velocities respectively. E_s describes the distribution of emission sources for s'th compound and R_s is the chemical reaction term which may contain nonlinear terms in c_s. $D()$ is the diffusion term, which is set to zero here. For n chemical species an n-dimensional set of partial differential equations (p.d.e's) is formed where each is coupled through the nonlinear chemical reaction terms.

The test case model covers a region of 300 x 500 km. and is a three-dimensional form of that used by [2] and although far from a detailed, does represent the main features which would commonly be found in an atmospheric model including slow and fast nonlinear chemistry, concentrated source terms and advection The chemical mechanism contains only 7 species but still represents the main features of a tropospheric mechanism, namely the competition of the fast inorganic reactions $NO_2 \overset{O_2}{\rightarrow} O_3 + NO$ and $NO + O_3 \rightarrow NO_2 + O_2$ with the chemistry of volatile organic compounds (VOC's), which occurs on a much slower time-scale. This separation in time-scales generates stiffness in the resulting equations. The reaction rate constants have been chosen as in Tomlin et al. [4]. and the photolysis rates were parametrised as a function of the solar zenith angle, see [4]. The background concentrations listed by [2] form the initial conditions for the model. These concentrations will then change diurnally as the chemical transformations take place. The power station is taken to be the only source of NOx and this source is treated by setting the concentration in the chimney set as an internal boundary condition. In terms of the mesh generation this ensures that the initial grid will contain more elements close to the concentrated emission source. The concentration in the chimney corresponds to an emission rate of NOx of 400kghr^{-1} and only 10% of the NOx to be emitted as NO_2. We have assumed a constant wind speed of 5ms^{-1} in the x-direction with y and z components of one tenth of this value.

5.1 Tetrahedral Finite Volume Spatial Discretisation Method.

Spatial discretisation of the model atmospheric diffusion equation on unstructured tetrahedral meshes reduces the set of p.d.e's in four independent variables to a system of ordinary differential equations (o.d.e's) in one independent variable -time. This system of o.d.e's can then be solved as an initial value problem, using the software tools that exist for this purpose[1]. For advection dominated problems it is important to choose a discretisation scheme which preserves the physical range of the solution [3]. The method used here is a , cell centered, finite volume discretisation scheme of [3] which enables accurate solutions to be determined for both smooth and discontinuous flows by making use of the upwind techniques for the advective parts of the fluxes.

5.2 Time Integration.

The time integration method used is the theta method module of the CSPRINT software which is designed for the moderate accuracy solution of stiff systems using local error control in time, [1]. Once the p.d.e's have been discretized in space we are left with a large system of coupled o.d.e's of dimension m $\times$ n where m is the number of mesh points, and n the number of species. These equations may now be written for a single species as

$$\underline{\dot{U}} \;=\; \underline{F}_N \;\;(\; t,\; \underline{U}(t)\;)\;\;,\; \underline{U}(0)\;\;\text{given}\;, \tag{5.2}$$

where $\underline{U}(t) \;=\; [U(x_1, y_1, z_1, t), ..., U(x_N, y_N, z_N, t)\;]^T$. The point x_i, y_i, z_i is the centroid of the i th cell and $U_i(t)$ is a numerical approximation to the exact solution to the p.d.e. evaluated at the centroid i.e. $u(x_i, y_i, z_i, t)$. The time integrator computes an approximation, $\underline{V}(t)$, to the vector of exact p.d.e. solution values at the mesh points. This numerical solution at $t_{n+1} = t_n + k$, where k is the time step size, as denoted by $\underline{V}(t_{n+1})$, is computed from

$$\underline{V}(t_{n+1}) \;=\; \underline{V}(t_n) \;+\; (1-\theta)k\;\underline{\dot{V}}(t_n) + \theta\;k\;\underline{F}_N(t_{n+1}, \underline{V}(t_{n+1})), \tag{5.3}$$

in which $\underline{V}(t_n)$ and $\underline{\dot{V}}(t_n)$ are the numerical solution and its time derivative at the previous time t_n and $\theta = 0.55$. The equations to be solved for the correction to the solution $\underline{\Delta V}$ for the $p+1$ th iteration of the modified Newton iteration used with the Theta method are:

$$[I - k\theta J]\;\underline{\Delta V} \;=\; \underline{r}\left(t^p_{n+1}\right) \tag{5.4}$$

where $J = \frac{\partial\;\underline{F}_N}{\partial \underline{U}}$, $\underline{\Delta V} \;=\; \left[\underline{V}(t^{p+1}_{n+1}) - \underline{V}(t^p_{n+1})\right]$ and

$$\underline{r}\left(t^p_{n+1}\right) \;=\; -\,\underline{V}(t^p_{n+1}) + \underline{V}(t_n) + (1-\theta)k\underline{\dot{V}}(t_n) - \theta k\underline{F}_N(t_{n+1}, \underline{V}(t^p_{n+1})),$$

The solution of this system of equations constitutes the major computational task of the calculation. The cpu times are excessive unless special solution techniques such as splitting the nonlinear equations [1] into a set of flow terms and a reactive source term are employed. Consider the o.d.e. function $\underline{F}_N\;(t,\;\underline{U}(t)\;)$ defined by equation (5.2) and decompose it into two parts:

$$\underline{F}_N\;(t,\;\underline{U}(t)\;) \;=\; \underline{F}^f_N\;(t,\;\underline{U}(t)\;) \;+\; \underline{F}^s_N\;(t,\;\underline{U}(t)\;) \tag{5.5}$$

where $\underline{F}^f_N\;(t,\;\underline{U}(t)\;)$ represents the discretization of the convective flux terms f and g in equation (1) and $\underline{F}^s_N\;(t,\;\underline{U}(t)\;)$ represents the discretization of the of the source term h in the same equation. The splitting approach used, [1], is to employ the following approximation to the Jacobian matrix used by the Theta method within a Newton iteration:

$$I - k\theta J \;\approx\; [I - k\theta\;J_f]\quad[I - k\theta\;J_s\;]\;)\;\;+\;\;O(k^2). \tag{5.6}$$

where $J_f \;=\; \frac{\partial\;\underline{F}^f_N}{\partial \underline{U}},\quad J_s \;=\; \frac{\partial \underline{F}^s_N}{\partial \underline{U}}$. The new iteration may thus be written as

$$[I - k\theta J_s]\;\;\underline{\Delta V^*} \;\;=\;\; \underline{r}\left(t^p_{n+1}\right) \tag{5.7}$$

where $\underline{\Delta V^*}$ is the operator splitting approximation to $\underline{\Delta V}$. The advantage of this is that each block of equations corresponding to a tetrahedral element may be solved separately using the Gauss Seidel method of Verwer [5]. The Jacobian matrix $[I - k\gamma J_s]$ is split into L, the strictly Lower triangular, D, the Diagonal and U the strictly Upper triangular matrices. and the equation rearranged to get

$$(I - \gamma k D - \gamma k L)\Delta \underline{V}^*_{m+1} = \gamma k U \Delta \underline{V}^*_m + \underline{r}(t^p_{n+1}). \qquad (5.8)$$

This approximation introduces a splitting error which fortunately only alters the rate of convergence of the iteration as the residual being reduced is still that of the full o.d.e. system. A lack of convergence of this iteration is dealt with by reducing the timestep k. The matrix $I - k\theta J_s$ is the Jacobian of the discretization of the time derivatives and the chemistry source terms. This matrix is thus composed of independent diagonal blocks with as many block as there are tetrahedra. Each block has as many rows and columns as there are p.d.e.s. and each block's equations may be solved independently . The choice of a time step is a difficult issue in reacting flow problems however in this case the chemistry reacts quickly compared to wind speed. The approach here is thus to use a standard local error control, though it is often the case that the convergence of the iteration that limits the timestep.

5.3 Mesh Generation and Adaptation

The initial unstructured meshes used are created from a geometry description using the SCIRun [8] mesh generator. The initial mesh inside a rectangular bounding box was generated with approximately 5000 elements. This resulted in a largest element with a side length 50km It is difficult to directly relate the size of unstructured meshes to regular rectangular ones, but our original mesh was comparable to the size of mesh generally used in regional scale atmospheric models. The fine scale grids used in present regional scale models are of the order of 10-20km. For a power plant plume with a width of approximately 20km, it is impossible to resolve the fine structure within the plume using grids of this size,[4], hence our use of adaptive grids. Close to the chimney the mesh was refined to elements of length 5km or 500m depending on the mesh level used. This ensured that the mesh would be refined to a reasonable resolution in this region of steep gradients.

These meshes are then refined and coarsened by the TETRAD [3] mesh adaptation module which is based on the refinement of tetrahedra into 8 tetrahedra with appropriate adjustments to ensure that the mesh is conforming at the edges. The criterion for the application of the adaptivity used in this work is based on refining or derefining the mesh based on the magnitude of solution gradients of the key chemical species NO and NO_2 across the faces of the tetrahedron, see [3]. For applications such as atmospheric modeling it is important that a maximum level of refinement can be set, to prevent the code from adapting to too high a level in regions with concentrated emissions. This is especially important if sources which are close to point sources in nature exist. For the test problem here the maximum level of refinement was a user defined parameter that was often limited to level 2 or 3. At the same time a sufficiently small refinement tolerance must be set so as to ensure enough refinement to determine the detail of the plume.

5.4 Integration with SCIRun

The integration of the above routines with SCIRun required writing a few SCI-link functions that could be called both from SCIRun and the program to provide control parameters for the program and get the feedback during the execution. No other changes where made to SCIRun. The transient aspect of the problem was dealt with by the integration module TETRAD/SPRINT sending out new mesh on every time step (rarely) or more usually immediately after to each remesh. Then, while the module would continue time integration, the rest of SCIRun network would process and visualize the current mesh.

The major advantage of SCIRun is provided by computational steering, i.e. the possibility not only to set up initial conditions and parameters, but also to have control over the execution. In the case of TETRAD the user interface allows the user to set up initial parameters of the problem: the position of the pollution source, initial velocity of the flow and level of refinement. If in the process of execution user decides that the refinement level or the refinement tolerance is to high or too low, then it can be changed for the next refinement. Similarly the species used as the basis for refinement can be altered dynamically. without quitting and losing any part of the data. An important "What -if" question is to do with the effect of the changes in the wind velocity on the existing solution. Accordingly we allowed the user to change the wind velocity during the execution. At the same time the visualization module provided 3D visualization of the flow, so without stopping computation one can look at the plume from different directions and angles, explore its cross-section as the plume develops. This example shows, that SCIRun can be used not only with its "native" modules, but can also successfully play a role of a framework to support all different kind of scientific computations.

6 Visualization Modules

6.1 Salmon Module

One of SCIRun's key modules is a graphical viewer named **Salmon**, so named for its ability to spawn multiple views (Roe) and its ability to send messages upstream in the dataflow network. Salmon collects the geometric primitives from any number of modules and presents them in a single 3D view. The user can rotate scale and then translate the objects as well as manipulating lighting camera parameters and rendering method, in order to obtain the desired view. Other views can be spawned to separate windows to simultaneously display the objects from other viewpoints. In the case of multi-species flows the requirement to see different species at the same time is obvious.

Geometric primitives are passed from the modules to Salmon as a subset of a scenegraph. These scenegraphs are a tree-based display list that defines geometric primitives, colors, and other rendering parameters. Drawing the scene involves traversing the graph and emitting OpenGL commands at each node.

Scenegraphs can be composed stored to disc via a persistent object mechanism and then read back for later display.

In addition for using the Salmon module for visual output we can also use it for 3D input by allowing the user to interact with specific objects in a scene. These objects, called Widgets, allow the user to augment parameters directly in the 3D scene.

6.2 Data-structures

In implementing SCIRun's datatypes, we considered what type of operations SCIRun modules would require. One example of such a consideration was the implementation of the vector and scalar field datatypes. Each of these fields comes in a variety of flavors, corresponding to the internal representation of the data - implicit, explicit, parametric - and the topology of the field - structured (implicit) or unstructured (explicit). But in designing the field datatype, we chose several generic operators to allow module writers to access information from the field without needing to know how the field is internally represented. As discussed above, these operators can query the field for minimum and maximum scalar values or geometric bounds, or for the field's value at an arbitrary point in space. This last operation, retrieving the value of the field at any location, is widely used in SCIRun and is implemented by an interface called `interpolate`.

6.3 Interpolation

The SCIRun field interface operator `interpolate` takes a point as an argument and calculates the value of the field at that specific location by linear interpolation. One example of a module that calls the `interpolate` method is the **Streamline** module. The **Streamline** module is used for vector field visualization: by tracing particles advecting through the vector field, the user can examine local flow phenomena around critical points (such as vortices and turbulence) while also gaining a global sense of the field's flow.

7 Atmospheric Diffusion Simulation Results

Each run was been carried out over a period of 48 hours so that the diurnal variations could be observed. We present here only a selection of the results that illustrate the main features relating to the adaptivity and to the use of SCIRun. The main area of mesh refinement is along the plume edges close to the chimney, indicating that there is a high level of structure in the plume. Using the adaptive mesh, we can clearly see the plume edges and can easily identify areas of high concentrations. The effects of the plume on ozone concentrations also provides some interesting results. Close to the plume the concentration of O_3 is much lower than that in the background. Due to the high NOx concentrations the inorganic chemistry is dominant in this region and the ozone is consumed by the second reaction in Section 5. As the plume travels downwind and the NOx levels decrease, the plume gradually picks up emissions of VOC's, as shown in Figure 2. The OVC chemistry leads to the production of NO_2 which pushes the above reaction in the reverse route. The levels of ozone can therefore rise above the

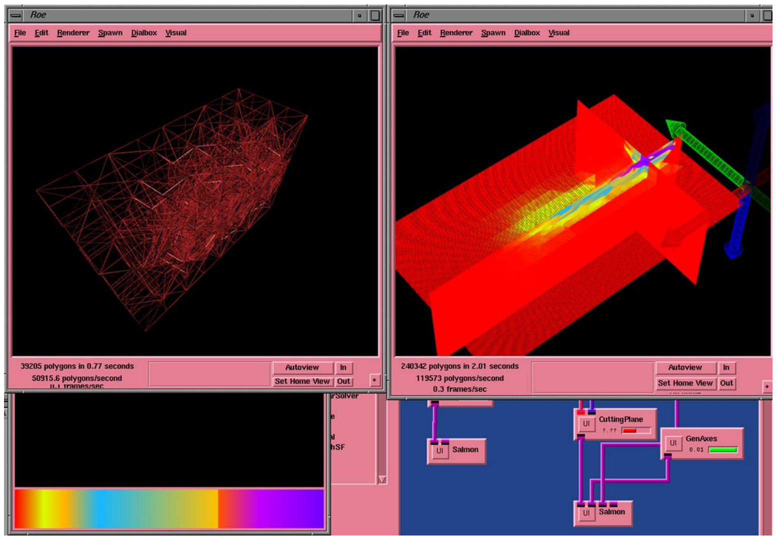

Figure 2. This picture shows one component of the plume in greater detail in three perpendicular cross-sections.

background levels at quite large distances downwind from the source of NOx.

8 Conclusions

We have presented in this work a simple example of the applicability of using SCIRun and 3D tetrahedral mesh adaptation for solving atmospheric models. SCIRun allowed for the easy incorporation of a 3D, adaptive, unstructured, finite element simulation codes (Tetrad/CSPRINT) to be easily incorporated, connected to an existing unstructured 3D mesh generator and 3D visualization modules. Using a case study of a power station plume we have illustrated how the adaptive solution reveals features such as peak levels of NO_2 and O_3 which, as in the two dimensional case could not be detected using a coarse mesh.

Future work will include incorporating a fully parallel version of Tetrad into SCIRun in order to simulate much larger scale models. Furthermore, more interactive links will be integrated to take better advantage of the SCIRun steering capabilities.

Acknowledgment. The authors would like to thank their many collaborators for their help, notably A.Tomlin G.Hart, and W. Speares at Leeds and Steve Parker and David Weinstein at Utah.

References

[1] I.Ahmad I. and M.Berzins. "An Algorithm for ODEs from Atmospheric Dispersion Problems." Appl. Num. Math. (25) 137-149 1997

[2] G.Hart, A.Tomlin, J.Smith and M.Berzins. " Multi-scale Atmospheric Dis-

persion Modelling by the Use of Adaptive Grid Techniques." Environmental Monitoring and Assessment, 1997.

[3] W.Speares and M.Berzins."A 3D Unstructured Mesh Adaptation Algorithm for Time Dependent Shock Dominated Problems." Int. Jour Num. Meths. in Fluids, 1997, 25, 81-104.

[4] A.Tomlin, M.Berzins J.M.Ware, J.Smith and M.Pilling. "On the use of adaptive gridding methods for modelling chemical transport from multi-scale sources." Atmospheric Env. Vol. 31 (18) 2945-2959.

[5] J.G. Verwer. "Gauss Seidel Iteration for Stiff o.d.es from Chemical Kinetics." SIAM J. Sci. Comp, 15:1243–1250.

[6] S.G. Parker and C.R. Johnson. "SCIRun: A scientific programming environment for computational steering." Supercomputing '95, IEEE Press, 1995.

[7] S.G. Parker, D.M. Beazley, and C.R. Johnson. "Computational steering software systems and strategies." IEEE Computational Science and Engineering, Vol. 4 (4), 50-59, 1997.

[8] S.G. Parker, and D.M. Weinstein, and C.R. Johnson. "The SCIRun computational steering software system." Modern Software Tools in Scientific Computing, Arge, E. and Bruaset, A.M. and Langtangen, H.P. editors, Birkhauser Press, 1-44, 1997.

[9] J. Vetter, G. Eisenhauer, W. Gu, T. Kindler, K. Schwan, D. and Silva. "Opportunities and tools for highly interactive distributed and parallel computing." Proceedings of the Workshop On Debugging and Tuning for Parallel Computing Systems, 139-142, 1994.

[10] W. Gu, J. Vetter, and K. Schwan. "An Annotated Bibliography of Interactive Program Steering." Georgia Institute of Technology, Tech. Report, 1994.

[11] SCI web page, http://www.cs.utah.edu/∼sci.