1998

SciAgents tool: User's Guide

John R. Rice
Purdue University, jrr@cs.purdue.edu

P. Tsompanopoulou

E. Vavalis

Report Number:
98-043

SCIAGENTS TOOL: USER'S GUIDE

John R. Rice
P. Tsompanopoulou
E. Vavalis

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

CSD-TR #98-043
November 1998

SciAgents tool: User's Guide

J.R.Rice, P. Tsompanopoulou and E. Vavalis

November 24, 1998

1 SciAgents GUI(SAtool)

Graphical user interfaces are applied to facilitate the use of large programs with a great amount and/or complication of input/output data. The main reasons are to help the users of these programs to reduce the time to provide the input data, to execute the program and to interpret the output.
Therefore, there is a necessity to built a GUI for SciAgents. Because of the complexity of composite PDE problems, it takes almost the same time to (1) define the input data of the problem, (2) specify the local network of workstations that will collaborate to solve the problem and (3) to actually compute the solution. Up to now the **SciAgents** is operated *manually* which takes a lot of time and effort. The problems that **SciAgents** handles are 2-D composite PDE problems, and by now it uses 3 interface relaxation methods on the interfaces. Questions like *How is the convergence to the solution of the PDE affected by the particular relaxation method, how to select optimum/good parameters of each particular method, how to determine initial guesses on the interfaces?* are very important and, since there no theoretical results, we need to experiment a lot to empirically understand them. Also we can use **SciAgents** to verify the results that come from a theoretical analysis of a relaxation method for model problems. So it is clear that it is critical to implement a user interface for SciAgents. Let us name this GUI SAtool. **SciAgents** is based on PELLPACK, it is natural that the **SAtool** design philosophy is similar to **Pelltool**(the GUI of PELLPACK), and in many cases it uses some of the individual tools either as they appear in **Pelltool** or extended/modified.

The design of **SAtool** is presented in the form of the a brief user's guide and its implementation is not presented here.

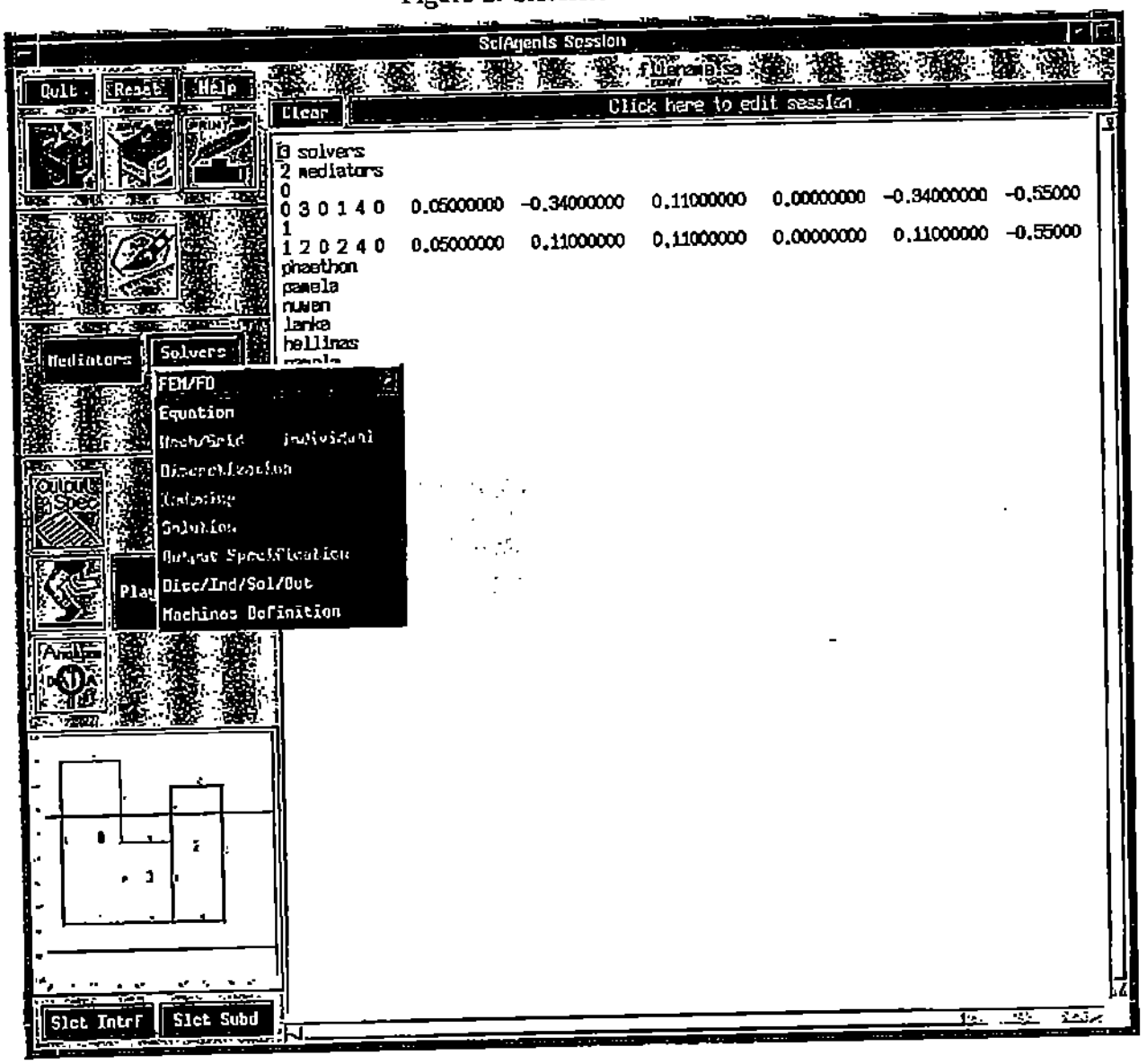
Figure 1: The initial window that appears typing SAtool

The introductory window (Figure 1) pops up when the user types **SAtool**. This command panel allows the user to start defining a new multi-PDE problem, load an existing one, or `Quit`. The `About SA` button provides information for the SciAgents, while the `Help` provides information about **SAtool** using hypertext files. Clicking on the `Load` or `New File` buttons starts the SAsession (Figure 2). This window is fully functional.

The SAsession window (Figure 2) organizes the effort that one needs to go through in order to specify the multi-PDE problem for the SA infrastructure. The `Quit`, `Save`, `Save as`, `Print` are essentially the same as in Pelltool (with only few extensions added) and are already fully functional (except from `Print`). The `Reset` button clears up the interconnectivity of the SciAgents network of machines. Also starts up the `httpd` daemon and the SciAgents server of default or user defined machines. The first part is already functional but there are some problems on starting the SciAgents-server remotely that hopefully will not take long to be fixed. The `Help` button provide information about the **The SAsession window** and the functionalities of each button. The `Clear` button clears the editor, on the right part of the window and the whole definition of an existing problem.

The session editor shown on the right is an emacs-type editor and can be used to manually generate the SciAgents input file that is currently needed. Using this file requires the user to declare the number of solvers, i.e. subdomains of the general domain of the PDE and the number of the mediators which is the number of the interfaces. Then for each of the mediators one provides the following information: its *id number*, the *number of solvers* that it has to collaborate and exchange data the *id number* of that interface (related with the specific mediator) for each subdomain, and the particular *relaxation scheme*

Figure 2: SAsession window



each mediator uses. The tolerance needed for the convergence criteria is given next, and then the coordinates of the starting and the ending point of the interface are given with the value of the solutions on those taken from either boundary conditions (for interface points on the original boundary) or initial guess otherwise. Then machine names are declared. First of all is the machine for displaying the various outputs, then the one for the global control. Next are the machines where solvers will run (each one might run on a different machine), and, finally, are the machines where the mediators will run. A first version of the input file is generated by the **Boundary, Interface and BC Editor** (see below), and contains information on how interfaces and solvers are matched together. The rest information can be added either manually by the user or using the corresponding buttons.

Clicking the geometry button, (Figure 3), raises **Boundary, Interface and BC Editor**. This tool (described below) assists the user in drawing the domain of the composite problem.

The canvas on the lower left is used to display a sketch of the composite articraft as shown in (Figure 5). This might be used for a successive view of the global domain which makes the selection of one or a set of subdomains or interfaces much easier. The selection mechanism is provided by the buttons `Slct Subd` and `Slct Intrf` and applies to any of the `Mediator`, `Solvers`, `Output`, `PlayBack` and Analyze data buttons whose actions are described below. Right now clicking on the canvas one can view an enlarged image of the global domain.

2 The Boundary and Interface Specification Editor.

This editor (Figure 4), is an interactive, graphical editor and it is raised by clicking on the icon shown in Figure 3, from the **SciAgents Session** window. It is used to define the outer boundary of the domain of the PDE problem, the interfaces (which may come from the physics of the problem or not) and to specify the conditions that apply to the outer boundary and to the interfaces. In addition it is used to define the subdomains that the PDE problem decomposition. It also produces textual descriptions, one for of each subdomain, and stores them in separate files, that can be used later when **Pelltool** is used to solve each subproblem. Finally, the editor creates

Figure 3: Icon that invokes the Boundary, Interface and BC Editor

an image (Figure 5) of the decomposed and saved domain and passes it to the SciAgents Session where it is displayed in the canvas area at the lowest left.

The editor itself consists of a command panel and a drawing window below it. If the SciAgents Session editor contains the specification of a previously defined problem (which the whole domain and the subdomains already defined) then the drawing window is used to display and modify the domain of a previously defined problem. (This facility is under construction). If no domain is defined then the drawing window contains only an empty grid. In the command panel there are buttons that allow the user to define and manipulate the domain and the subdomains of the PDE problem and to set conditions on each piece of both the boundary and the interfaces. The buttons on the command panel can be activated by clicking on them with the left mouse button, while the middle button has no affect and the right one provides help messages for each operation. In the drawing window there are two working modes. Some of the operations, operate in the edit mode and the rest operate in the command mode. Operations like Add CP operates on the control points that can be selected by clicking on them with the left mouse button. A boundary component is selected by clicking on one of its pieces or on one of its control points with any of the mouse buttons. If an operation switches the mode into the edit mode, clicking on the right mouse button switches to the command mode.

Buttons descriptions [functionality status indicated at the end]:

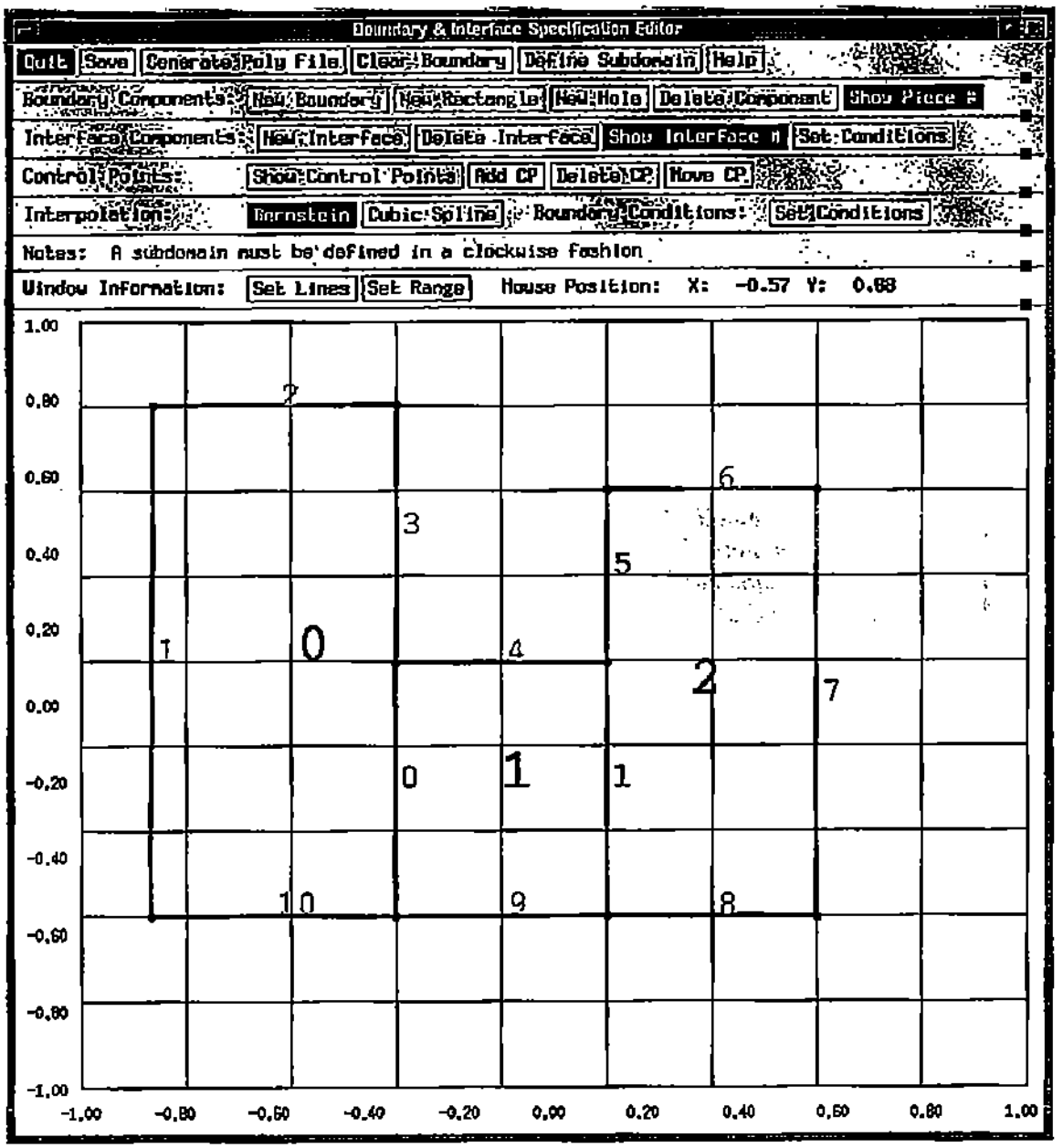
Quit: Quit the editor. If the domain created is not saved, it asks the user to save it or not. (functional)

Save: Save the textual representation of each subdomain displayed in the editor. Each subdomain is stored in a file named 'subdomainN.e', where N is the order number of the subdomain. If the subdomains were already defined and/or stored in files, they are overwritten. It also writes the contents of the drawing area to a file, named input.sa (the input to SA file). (functional)

Generate Poly File: (Under construction).

Clear Boundary: Clear the drawing window, by deleting both the outer

Figure 4: Domain, Interface and BC Editor



boundary and the interfaces if there are any. (functional)

Define Subdomain: Define the subdomains of the problem. It uses the already drawn pieces of the outer boundary and/or the interface pieces. In the simplest case where there is only one subdomain (i.e., the PDE domain) this is defined automatically. A subdomain is defined by selecting sequentially boundary and/or interface pieces in a *clockwise* order. In the edit mode of `Define Subdomain` button the left mouse button determines the first piece of the subdomain, the right mouse button determines the last piece and the middle mouse button is used to specify the rest pieces of the subdomain. When the last piece is selected, the definition of the subdomain is completed and the current mode is switched into command mode. This button is used each time a new subdomain is defined. (functional)

Help: Provide information about the buttons of the editor and their functionality using hypertext files. (functional)

New Boundary: Define the outer boundary component.The control points must be defined sequentially and in clockwise order. In the edit mode of 'New Boundary' the left mouse button must be used to define the first control point of the boundary component. In addition it defines control points of the current boundary piece. The middle mouse button complete the current boundary piece by defining its endpoint. The shape of the new piece is completely defined by its control points and the selected interpolation scheme. The right mouse button completes the boundary component and exits the edit mode. It completes the last piece of the boundary by closing it but does not define a control point. After that a default boundary condition $U = true(x,y)$ is assigned to every edge of the boundary. (functional)

New Rectangle: An easy way to define a rectangular outer domain, whose range is set using the `Set Range`. (functional)

New Hole: Used to create holes inside the outer domain. (not functional)

Delete Component: Delete the selected boundary component. Now it delete the outer boundary and the interfaces if there are any. (functional)

Show Piece #: Display the piece numbers of the outer boundary. (functional)

New Interface: Define a new interface piece. It must begin and end on beginning/ending points of existing pieces (either boundary pieces or previously defined interfaces pieces). An interface component may consist of many pieces. The edit mode of `New Interface` is very similar to the edit mode of `New Boundary`. The left mouse button must be used to specify among all the existing control points, the beginning point of the current interface. Also it can be used to define control points of the current piece

of the interface. The middle mouse button completes the current interface piece by defining its endpoint (i.e., define one more control point in the interior of the outer domain). The shape of the new piece is completely defined by its control points and the selected interpolation scheme. The right mouse button completes the interface component and exits the edit mode. It completes the last piece of the current interface by selecting an existing control point. Note that the first and the last points of the interface component do not define more control points since they are already defined. After completing the current component, a default interface condition $U =$ *rinterface(x,y)* is assigned to every edge of the interface. This operation must be selected each time a new interface is defined. (functional)

Delete Interface: Delete the selected interface component. All the interface components that are built based on points of the selected interface, are also deleted. (functional)

Show Interface #: Display the piece numbers of the interface components. The number is globally determined over all interfaces. (functional)

Set Conditions: Specify the interface conditions (that come from the physics of the PDE problem) associated with each piece of the selected interface component. To specify that a boundary piece is an interface component use the **Set Conditions** button 2 rows below this one. (not functional)

Show Control Point: Display the control points of the boundary/interface pieces in the drawing window. This is helpful when adding, deleting, moving control points. (functional)

Add CP: Add control point to the drawn pieces. To add a new control point after an existing control point, select the existing one (i.e., click on it with the left/middle mouse button) and without releasing the button, drag the cursor to the new point's location. Release the mouse button. The affected pieces are redrawn. If the selected point is a beginning/ending point and it is selected by the left mouse button, then the new point is added to the left boundary/interface piece of the selected point and is the new beginning/ending point. If the selection is made by the middle mouse button, then the new point is added to the right piece of the selected point and the beginning/ending point remains at the same location. Many control points can be added to various pieces before exiting the edit mode. To complete the operation and switch to command mode, click on the right mouse button. Currently the selected point can not be a beginning/ending point since interface pieces might need to be moved. (functional)

Delete CP: Delete control points of the drawn pieces. To delete a control point click on it with the left/middle mouse button. If the selected control

point is a beginning/ending point and if the point selection is done with the left mouse button, then the the new begin/end point is the previous (but not begin point) of the left piece. If the selection is done with the middle mouse button then the new begin/end point is the next (but not endpoint) of the right piece. To complete the operation and switch to command mode, click on the right mouse button. Currently this operation should not be used to control points *near* begin/end points. (functional)

Move CP: Move control points of drawn pieces. To move a control point select it (using left/middle mouse button) and, without releasing the button, drag the cursor to the point's new location. To complete the operation and switch to command mode, click on the right mouse button. Currently the selected point can not be a beginning/ending point since interface pieces might need to be moved. (functional)

Bernstein: Use Bernstein polynomial interpolation of the control points to define curved pieces. (functional)

Cubic Spline: Use the cubic Spline interpolation of the control points to define curved pieces. (functional)

Set Conditions: Specify the boundary and interface conditions associated with each piece of the selected boundary/interface component. When this operation is selected, the Boundary/Interface Conditions Specification Editor displays the current number and condition of each piece of the selected component. The pieces are numbered in the order they are created. To see this numbering select the `Show Piece/Interface #` before selecting `Set Conditions`. To modify the condition for one of the pieces, erase the current one and enter the new expression in its place. To modify all the conditions, select clear to delete the current conditions and enter the new. To save the new conditions, select `Continue` and select `Cancel` to quit the Boundary/Interface Conditions Specification Editor without saving the new conditions. (functional)

Set Lines: Set the number of the grid lines in the X and Y directions. The lines of the grid are simply a guide for placing control points at correct locations and not part of the discretization of the domain. To change the number of the lines, place the cursor in the rectangle that contains the current numbers, erase them and enter the number of lines. Select `Continue` to save and quit, click on `Cancel` to quit without saving. (functional)

Set Range: Set the range of the drawing area for the X and Y directions. The four numbers, separated by spaces, should be erased before the new values are entered. The first and the second numbers specify the minimum and maximum in X direction and the third and the forth numbers specify

the minimum and maximum in the Y direction. Select Continue to save and quit and click on Cancel to quit without saving. (functional)
Mouse Position: Display the coordinates of the mouse cursor. Functional only when the cursor is in the drawing area.
To define a domain and decompose it to several subdomains, follow the following procedure:

- *Clear* the drawing window, if an old domain definition exists.
- Specify the range of the drawing area, using Set Range.
- Specify the number of grid lines in each direction, using Set Lines.
- Select Show Control Points operation, if any of the outer boundary or the interfaces are curved pieces.
- Define the outer boundary of the PDE problem using the New Boundary or the New Rectangular operations.
- Define the interfaces using New Interface.
- Define the conditions for each component using the Set Condition operation.
- Use Define Subdomain to specify the subdomains of the decomposed PDE problem.
- Save the subdomains in their files and post an image of the drawing area in the canvas on the lower left part of SciAgents Session tool.

We might want to investigate the possibility to provide a second way to create the main domain of the problem. One could create the subdomains separately and then move them, by clicking with the left button of the mouse inside them, to the desired position. They would have to match and create the main domain.

3 The Mediator Editor.

This editor (Figure 6) allows the user to specify (1) the machine that will handle the selected (by Slct Intrf button) mediators, (2) the relaxation method that will be used, and (3) the value of the parameter(s) of the selected method. The user can edit the inputs of all dialog boxes and can

either select one of the existing relaxation schemes or select CUSTOM to pop-up an editor for specifying a new scheme. (functional) Two relaxations methods are already ready to be specified through the editor. The AVE and the three variations of Default defined by T. Drashansky in his Ph.D. The Help button again provide information on the utilities of this Editor, using hypertext files.

4 The Solver Editor.

For each subdomain we need to specify the PDE equation, the type of discretization (grid/mesh), the PDE discretization scheme and the method to solve the linear algebraic system. This specification is made by clicking on the Solvers button in the SAsession window, after the *id number* of the solver(s) has been selected through the Slct Subd button. Clicking on the Solvers raises a small pull down menu and the user selects how to proceed. First of all has to select the type of discretization, either *FEM* or *FDM*, and then click on Equation to define the PDE equation. This will be done through the SciAgent enhanced version of Pelltool that will be started for the selected solvers. Having defined the equation and the boundary for a subdomain, the corresponding .e file will contain default specifications for discretization, indexing, solution and output segments. If the user wish to modify one or more of these specifications, she/he has to select each subdomain separately (using the Slct Subd button) and the click on the Disc/Index/Sol/Out button on the Solvers pull down menu. The Pelltool session will come up with the selected subdomain's .e file so one can proceed with the required modifications. To save the modified .e file the user has to click onth Save button of the Pelltool's Session. To define the mesh discretization scheme, the user has to work each .e file individually since the mesh/grid depends strongly on the geometry of each subdomain. Therefore first click on Slct Subd button to select a subdomain and then click on Disc/Index/Sol/Out button. The Pelltool will come up with the corresponding .e file and now the user can go on as usual. At this state some output files must be defined for the boundary points and the values of the function on these points, and this can be done automatically be clicking on the Agent button on the command panel of the Pelltool's Session. The machines can be specified when on clicks on the Machines Definition button. An editor comes up like in Figure 7 and the user has to fill in the boxes with the machine names chosen from the Local Access Network. (functional)

5 The Run Tool.

The global execution of **SciAgents** starts when all definitions and specifications are completed, and the user clicks on the `Run` icon. This raises a window similar to Pellpack's ExecuteTool. When the actual execution starts an Execution Trace window pops-up which selectively displays the execution procedure graphically (Figure 8). Appropriate colors are used to indicate what the agents are actually doing (computing or waiting for data). This window can also provide the values of the convergence criteria and iteration numbers, list the messages being sent and allow the user to pause or stop the execution and modify certain parameters. (not functional)

6 The PlayBK Tool.

The **PlayBk** button raises the Execution and Communication Info window. It contains a graph (similar to the one described in the Run tool above) of the main domain with its subdomains and two time diagrams, one for the mediators and the other for the solvers. The graph, illustrated in (Figure 8), shows if a solver is working or waiting to receive new data for its interfaces. Colors are used to provide this information to the user, for example, a subdomain is colored green when its solver is working and red when its solver is waiting. Other time diagrams like the ones in Figure 9 might be added. There is a great possibility to use **ParaGraph** (a graphical display system for visualizing the behavior and the performance of parallel programs on message-passing multicomputer architectures) to process and visualize the execution trace information. (not functional)

7 The Output Display.

The `Output Spec` and the `Analyze DATA` buttons raise Pellpack's Output Specification Editor and Analyze data editors with which one can specify views the various data after the computation is complete. (not functional) Right now the user can plot an image of the function (or the first derivatives) on the global domain, running the script (written in Perl) `preplot` (the whole path for that script is `/p/pses/sciagents/src/front_end/scripts/preplot`) to process and manipulate the `.out` and `.mesh` files of each solver and the files that contains the boundary points on the interfaces.. This script also copies from the script directory 3 MATLAB files (`plotu.m plotux.m`

plotuy.m) that can be used to make the plot of either the function or its derivatives.

8 The Selection Buttons.

The two buttons `Slct Subd` and `Slct Intrf` at the bottom of th e SAsession Window have similar functionality. If one clicks on `Slct Subd` a dialog box comes up, like in Figure 10, and the user needs to define the number of the subdomains that will apply other functionalities of the GUI. For the case of subdomains, these functionalities can be the *Solver Editor*, the *Output Specification Editor*, the *Analyze Data Editor* and the *PlayBack tool* The numbers of the selected interfaces is specified in the dialog box raised by clicking on the `Slct Intrf` button, where the user then can apply the *Mediator Editor* and the *PlayBack tool.* (functional)

Figure 5: Drawing Area image that is saved in the canvas of SciAgents Session window.

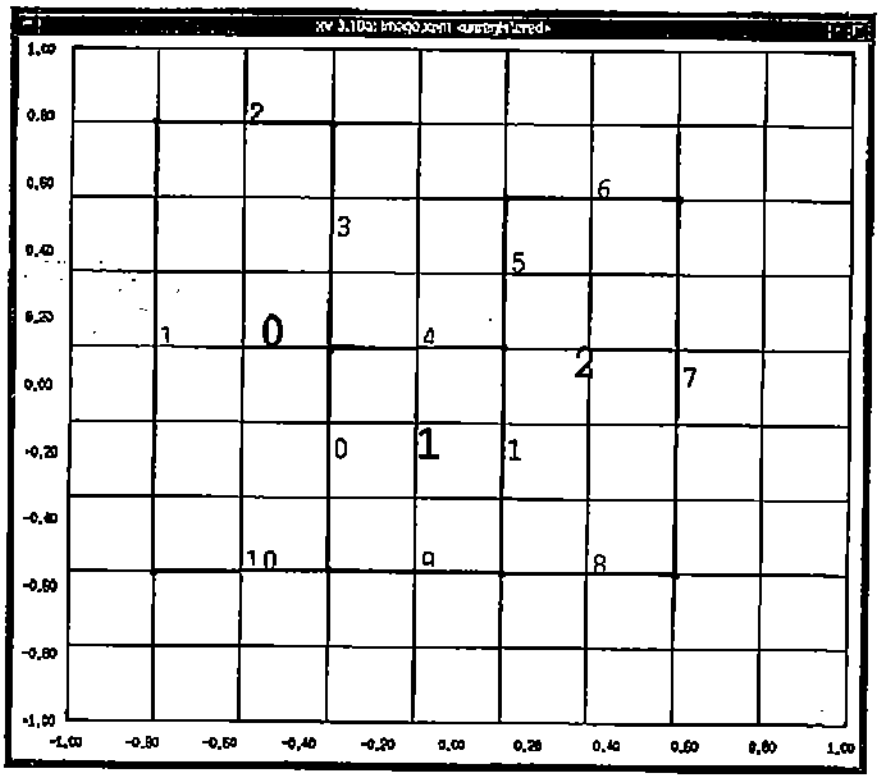


Figure 6: Mediator Editor

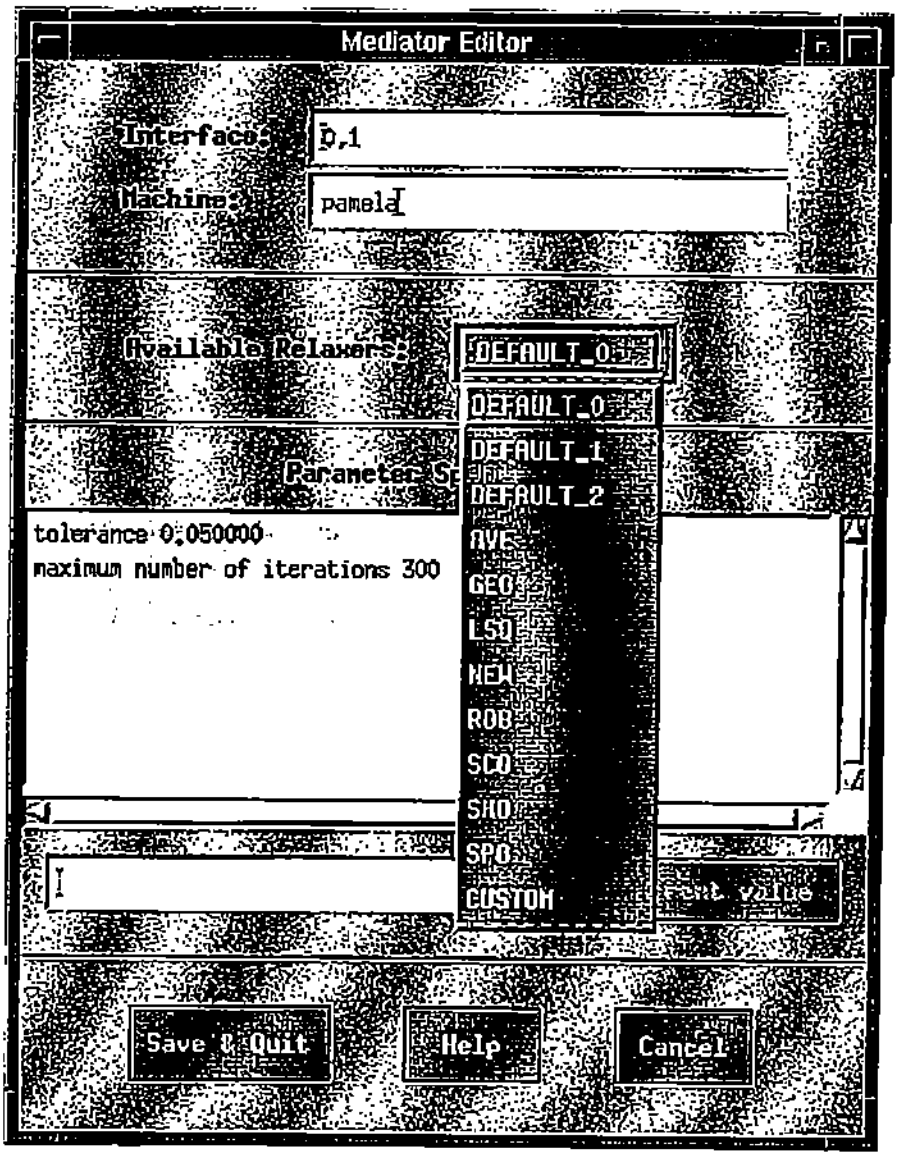


Figure 7: Machines Specification Editor

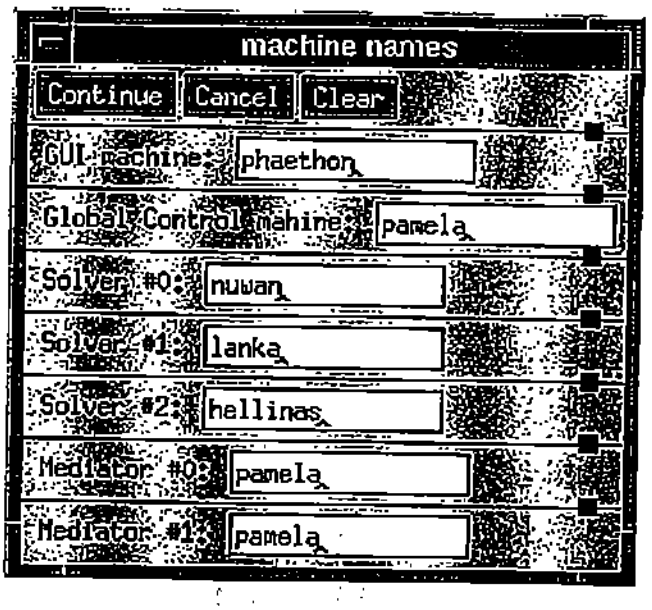


Figure 8: Supervising Execution Graph

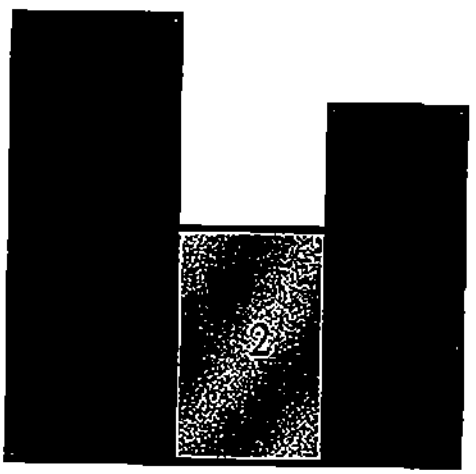


Figure 9: Execution Trace Diagrams

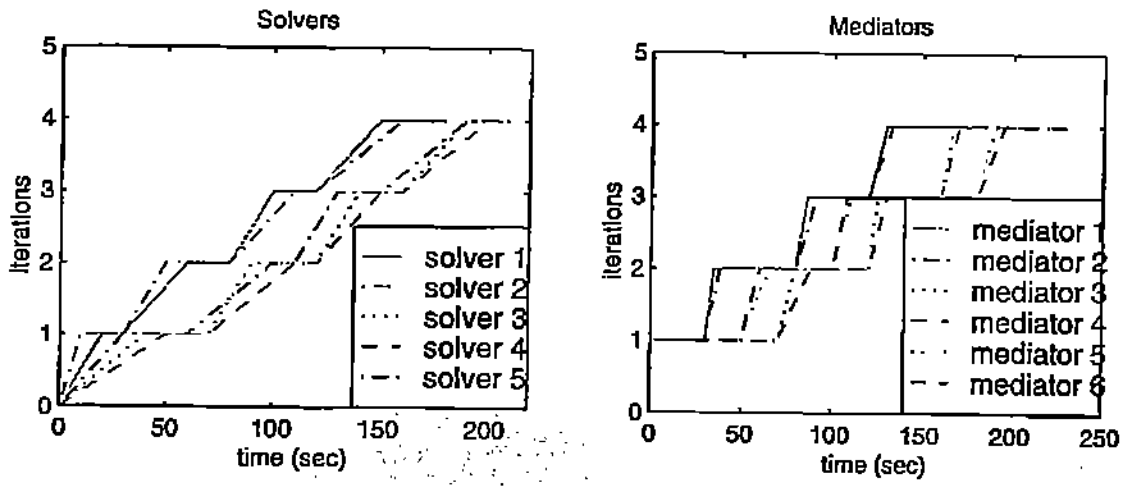


Figure 10: Select Interface/Subdomain Dialog Box