Macro Data-Flow Simulator Display Interface

BY

Olivier Tardieu

Technical Report 90-23

Electrical Engineering Systems

University of Southern California

Los Angeles, CA. 90089-0781

MACRO DATA-FLOW SIMULATOR

DISPLAY INTERFACE

Olivier Tardieu

April - August 1990

Department of Computer Science
Ecole Nationale Superieure de l'Aeronautique et de l'Espace
10 Av. E. Belin 31400 Toulouse
France

Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, California 90089-0782
U.S.A.

(for the degree of Ingenieur diplome de l'Ecole Nationale Superieure de l'Aeronautique et de l'Espace)

This material is based upon work supported mainly by personal funds and in part by the U.S. Department of Energy, Office of Energy Research under Grant no. DE-FG03-87ER25043

INTRODUCTION

A macro data-flow simulator has been developed at the University of Southern California in the Electrical Engineering Department under the guidance of professor J.L. Gaudiot.

But the main purpose of this simulator was to obtain a large quantity of information to allow progress in partitioning problem.
However to achieve this goal, the results obtained from the simulator have to be fully exploited. A friendly user tool should be developed to allow an automatic exploitation of the results, so that the amount of data processed can be largely increased.
In this way the research on partitioning problems can be considerably accelerated.

A software meeting these requirements is presented here. It is called Simulator Display Interface, because it provides the user with an interface whose major goal for the moment is to display the results of the simulations run by the U.S.C. macro dataflow simulator.
As will be discussed later it is the starting point of a whole environment that will allow complex issues based on the macro dataflow simulator.

ACKNOWLEDGMENTS

I would like first to express all my thanks to Dr. Jean-Luc Gaudiot Associate Professor at the University of Southern California in the Electrical Engineering department and who was my advisor all along this five months project. His very friendly welcome, his constant help and invaluable assistance have allowed to solve many problems and to conduct this project properly.

I also want especially to thank Mr. Bernard Lecussan , Professor and Chairman of the Computer Science department at the Ecole Nationale Superieure de l'Aeronautique et de l'Espace (Toulouse-France). Again his invaluable assistance and advises have been very helpful for the preparation and coordination of the project.

I don't want to forget to thank the very friendly and donuts eating staff of Dr. J.L. Gaudiot for their help and everlasting good mood. Among them special thanks to Moez Ayed, Chih-Ming Lin, Chinhyun Kim, Andrew Sohn, Dae-kyun Yoon,and Nahmoon Yoo.

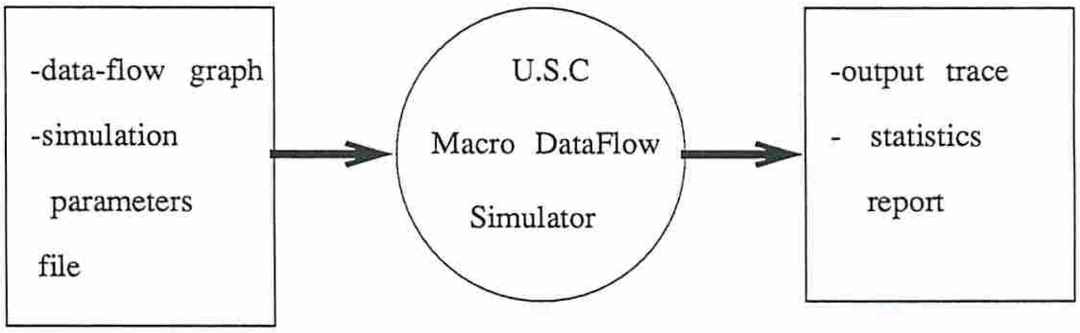
Last I want to thank all the staff of the University of Southern California and especially Mary Zitterco for her extraordinary friendliness.

U.S.C. MACRO DATAFLOW SIMULATOR

Although the goal of the project is to exploit the results obtained in running the DataFlow Simulator and create a user friendly environment, the U.S.C. DataFlow Simulator is shortly described here.

The u.s.c. macro DataFlow model is a scheme having a multilevel of model of execution (higher level is a tagged data-flow and lower level is von Neumann)
At low level the simplicity of control loci of von Neumann model is exploited and at high level, the automatic parallel sequencing of data-flow.
In order to obtain some ideas about partitioning, a simple simulator has been developed executing a macro data-flow graph.
In addition to the above hardware simulator, a graph simulator has been developed for simple execution of data-flow graph without resource limit of hardware details.
This simulator uses the same program format of instructions. However, simulation time and parameter setting for simulation are different.

Thus the existing software can be summarized by the following diagram:



The goal of the project is thus to create an environment allowing to exploit in a more easily way the existing software and featuring especially a tool aimed at displaying automatically the results of the simulations run by the simulator.

SPECIFICATIONS

The problem :

A dataflow simulator has been developed at the University of Southern California which is currently used by all people working on this subject in the staff of J.L. Gaudiot . However the results provided by the simulator are difficult to manage. The information has to be retrieved by the user himself in various files and its analysis and representation in a format that can be more easily understood than a succession of boolean characters is also the job of the user . Because all this work is repetitive and extremely time consuming an automatic tool aimed at displaying the simulations results has to be created. In the following paragraph all the specifications for this tool are presented .

The basic specifications :

The software should be able to automatically retrieve the information selected by the user and display the results. The basics functions that should be implemented are the trace of the simulations with sampling intervals chosen by the user and other functions related to the analysis of more than one simulation after the simulations have been run (post processing) . Of special interest among these post processing functions is the Speed Up and the Sensitivity of the simulations to specified parameters .

Other specifications concern the way the software should interact with the user (man machine interface approach). The software should prove actually to be user friendly and minimize the task of the user. As such it should be designed to be interactive. Another point is that this software (namely simulator display interface) should be easily expanded , that is it should be designed with structures allowing an implementation of future issues with the minimum of modifications .

Resulting specifications :

To achieve the above results, a file system should be designed to store all the simulations in an easy retrievable way , along with the parameters that were used to run these simulations.

Directions :

Basically the functions that should be implemented are :

-storage of simulation parameters
-storage of simulation results (trace)
-storage of simulation statistics

(For the three above points an ordered system of files has to be created)

-Display functions:

* show statistics of a simulation
* show histograms of a simulation
  (the user should be able to specify sampling time intervals and data selected for display)
* show Speed Up of a set of simulations
  (same dataflow graph with number of processing elements changing)
* show Sensitivity to parameters specified by user

-Interface with user:

*interactive interface
* system of menus with selection operated by mouse
* large graphic possibilities
* implementation of the interface with a tool widely spread in the industry in a window system environment.

Other requirements :

To complete specifications, a simulator display interface user's manual has been edited and strong guidelines on the functions and aspects of the software can be found there.
Therefore we recommend that it should be read for more complete specifications.

THE TOOLS

I MATLAB :

Matlab was chosen for two major reasons : first it has been especially built to process large amounts of numerical data stored in matrices type arrays . Many functions have been designed to process these matrices which can be easily manipulated with usual operations (addition , multiplication, powers). Second Matlab has got a large library of graphing functions which are especially useful for display purpose . The results that are obtained using these graphing functions can be stored in special graphing matlab files and printed later (or even post-processed) at the user convenience .

II C Language :

The C language was chosen mainly for its compatibility with the dataflow simulator program and the various X windows libraries . Because close connections exist between the simulator (entirely written in C) and the display environment it seemed logical to write this last part using this same language. This will prove to be especially useful when the environment is expanded and new functions depending on the simulator program are added (such as display on time which will have to work closely with the simulator program in order to achieve the required efficiency) . More importantly, the X windows libraries chosen to design the interactive environment are all C libraries.

III X Windows System :

The Windows system was chosen primarily because it provides adequate ways to implement an interactive environment. Another reason for it selection is that it has become a standard and as such allows any user to easily integrate the display environment in his own system . The fact that the windows can be manipulated as autonomic entities within the X windows environment is another advantage . Last, the X toolkit athena widgets which was also extensively used to build the simulator display interface proves useful to avoid design interactive windows from scratch : this is because the programmer has at his disposal a set of preconfigurated interactive windows which he can organize and customize at his convenience by using the toolkit library (other facilities that are not provided with the toolkit can be found in the large Xlib C language interface library) .

Description of the widgets used :

Command Widget :

The command widget is a rectangular button that contains a text or pixmap label . When the pointer cursor is on the button, the button border is highlighted to indicate that the button is available for selection. Then, when a pointer button is pressed and released the button is selected, and the application's callback routine is invoked.
When a Command widget instance is created, resources are retrieved from the resource database (cf Command.h [5])

Label Widget :

A label is an noneditable text string or pixmap that is displayed within a window The string is limited to one line. A label can neither be selected nor directly edited by the user. (cf Label.h for the list of resources retrieved at creation from the database).

Text Widget :

(This widget has not been used in the 1.0 version of the simulator display interface but shall be utilized in later issues . Therefore a description is provided here).
A text widget is a window that provides a way for an application to display one or more lines of text. An option also lets an application display a vertical scrollbar in the text window letting the user scroll through the displayed text. Other options allow an application to let the user modify the text of the window.
The text widget is divided into three parts : source , sink and text widget. Thus, the storage of the text (source) is separated from the painting of the text (sink). The text widget coordinates source and sink.

Box Widget :

The box widget provides geometry management of arbitrary widgets in a box of a specified dimension. The children are rearranged when resizing events occur or when children are added or deleted. The box widget always attempts to pack its children as closely as possible within the geometry allowed by its parent. (cf box.h for the list of resources retrieved at creation from the database).

Form Widget :

The form widget can contain an arbitrary number of children or subwidgets. The form provides geometry management for its children, which allow an individual control of the position of each child. Any combination of children can be added to the form. The initial positions of the children may be computed relative to the positions of other children. When the form is resized, it computes new positions and sizes for its children . This computation is based upon information provided when a children is added to the form. (cf form.h)

Dialog Widget :

The dialog widget is used whenever an application requires a small piece of information , such as a file name, from the user. A dialog widget is a special case of the form widget. A dialog widget can contain up to three areas : the first line contains a description of the function of the dialog widget (label). The second line contains an area into which the user types input. The third line can contain buttons that let the user confirm or cancel the dialog input. (cf dialog.h)

SIMULATOR DISPLAY INTERFACE USER MANUAL

I INTRODUCTION

You now hold in your hands the Simulator Display User Manual which describes the whole user friendly environment aimed at displaying the results of the simulations run by the USC dataflow simulator .

As this is the first version of the Simulator Display Interface (1.0) you might find that some parts of the user's manual are not clear or that the Simulator Display Interface can be improved. Do not hesitate to communicate us your remarks which will be helpful to enhance our product.
(you can leave messages or talk to the author on tardieu@pollux.usc.edu)

And now, go on to the reading and enjoy !

II SETTING THE ENVIRONMENT

This software was originally developed on sun-3/50 and sun-3/80 computers with the X Windows System . Although it is possible to run this program on other computers, best results will be achieved using these computers.

First install the X Windows System environment, and be sure to have at least a shell or a command window at your disposal . Then run the IconManager which will prove helpful in managing all the windows that constitute the Simulator Display Interface . To use this software you must also be able to use the Matlab language .

Run the Xparam program in your shell or command window. Some of the selections you can make in Xparam lead to the display of text in this window (this will eventually disappear in later versions of the Simulator Display Interface where a special text window will be created for such display purposes).
Run the Xdisplay program.
Open a matlab command window and a matlab graphing window.
Arrange all the windows at your convenience on your screen (be sure not to shrink the Xdisplay and Xparam windows). Your display environment is now ready.

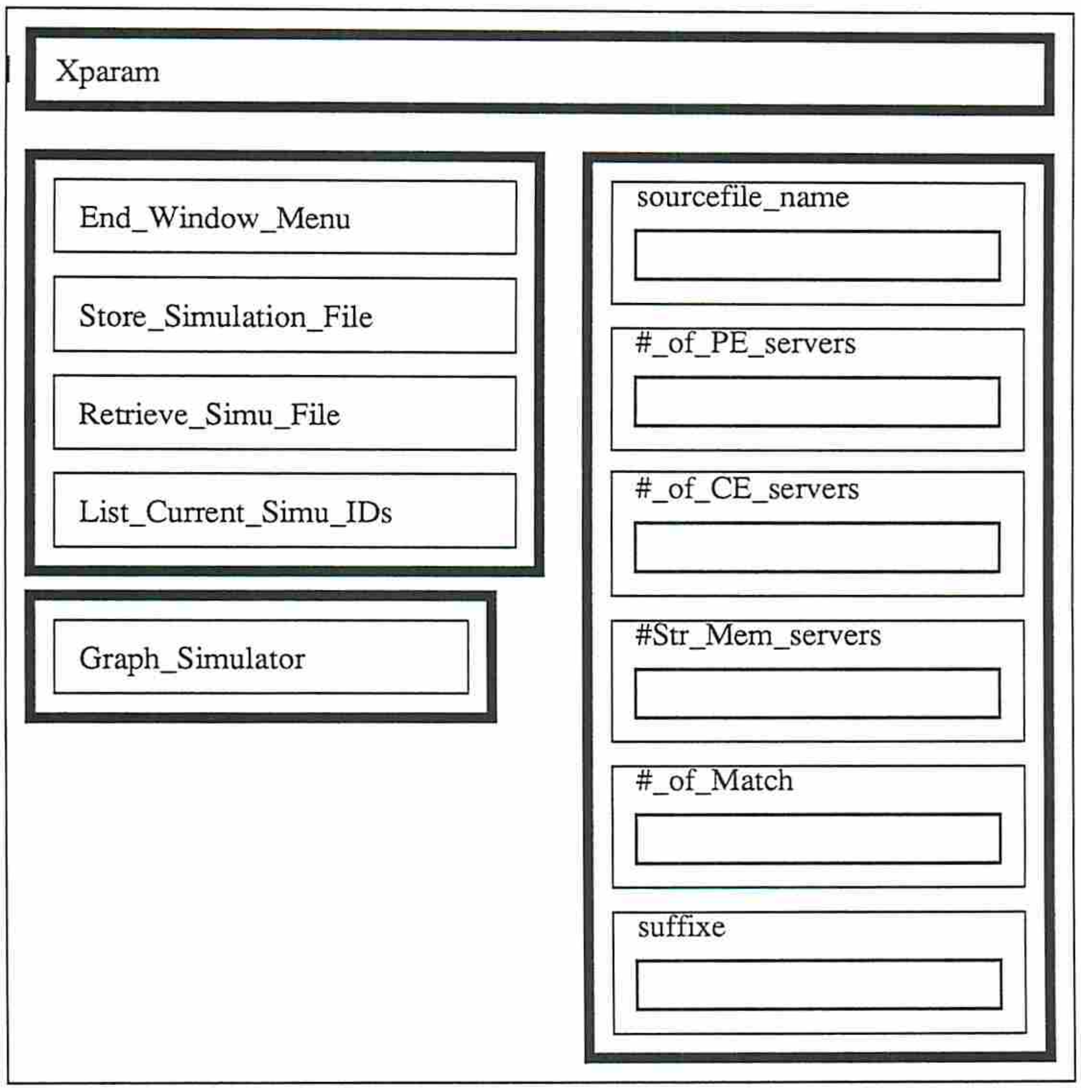
about the windows : as all windows used here are part of the X Window System, you can operate on them all the standard windows commands :

open/close:	if the window is shown on the screen closing it will reduce it to an icon in the icon manger. Click again on the icon in the icon manager to let it reappear.
quit	: will permanently exit the window. Make sure you do not need the window anymore if you select this option
move	: to move the window! this will be useful for you to · set a display environment that please you.
resize	: you are free to resize the matlab windows and the shell/command window. You can do the same with the Xparam and Xdisplay windows but do not attempt to shrink them beyond their initial size.Their is optimized to be the minimum needed to display the information inside. To shrink them more will loose this information on the screen (if you enlarge the window, the contents will reappear).
front/back:	brings the window in front/back of the others

III the Xparam Window

Before starting to display the simulations coming from the data-flow simulator you have to become familiar with the functions of the Xparam window.

When you first run Xparam, the window that appear looks like this:



The left-upper buttons are commands buttons. When you move the cursor over one of these buttons, it is highlighted which means that it is now available for selection. Click on the mouse to execute the corresponding command.
The left-lower button acts as a selection button. Click once on it and it changes its label. Click once again and it comes back to the original label. The information displayed on the screen (the label) is the current mode.
The right column is constituted of Dialog Widgets. Each Dialog Widget is composed of a label explaining its function (on the top of the widget) and a box where you can edit a string . To edit the string, you have to point the cursor in the box. It will change its form, indicating that you can now edit in the window. You will see that a smaller cursor is present in the window. You can add text at the location the small cursor is pointing to. To delete a character use delete to delete the previous character or Ctl-d to delete the forgoing character. To position the smaller cursor you can position the cursor at the position you want to bring the smaller cursor and click on the mouse to confirm the position. You can also use Ctl-b to move the mall cursor backwards or Ctl-f to move it forwards. Do not use the Return button.

And now we will explain the Xparam functions.

End_Window_Menu

Click on this button will exit the Xparam Window. Be sure that you do not need it anymore when you do so, because if you exit the window, all the information within will be lost.

Store_Simulation_File

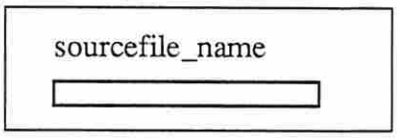
Before you click on this button you must be sure that the simulation_result file you want to store exists and that it is present in your current directory (the one from which you run Xparam) under the WWW name.
You must also be sure that you have correctly given the ID of the simulation you want to store. Before going any further we will explain it now.

First select the simulation mode. You have the choice between

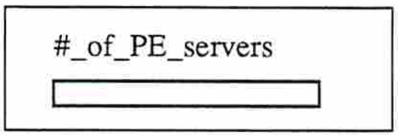
Graph_Simulator — which is the default mode (automatically) selected at the beginning of the display session) and

Simulator — (which stands for DataFlow Simulator) if you click once on the button.

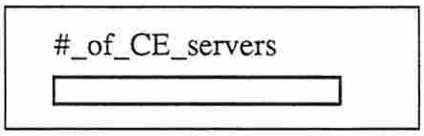
Then enter the abbreviation of the source file used as input of the dataflow/graph simulator. Use three letters for this. The abbreviation is entered in the dialog window.



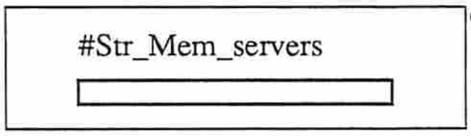
Enter the number of processing elements servers (two characters for the moment) in the dialog window. You can select any number from 00 to 99 or ' in' for an infinite number of servers.



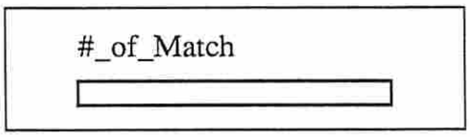
Enter the number of communicating elements servers (two characters for the moment) in the dialog window. You can select any number from 00 to 99 or ' in' for an infinite number of servers.



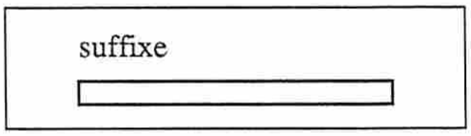
Enter the number of Structure Memory servers (two characters for the moment) in the dialog window. You can select any number from 00 to 99 or ' in' for an infinite number of servers.



Enter the number of Matching units (two characters for the moment) in the dialog window. You can select any number from 00 to 99 or ' in' for an infinite number of units .



Then using the window, give a suffixe to the simulation ID (identification). This suffixe can be another parameter of



the simulation or anything you want that will allow you to distinguish two simulation_results files whose first part of ID (cf preceding page) are the same. Use three characters (preferably letters) for the suffix.

Now let us suppose that you have recorded the following simulation ID:

Simulator type :	graph simulator
sourcefile_name :	src
#_of_PE_servers :	64
#_of_CE_servers:	16
#Str_Mem_servers:	28
#_of_Match	: in
suffixe	: mic

The use of the Store_Simulation_File command (triggered by clicking on the button) is the following:

- check to see if the src directory exists in your current directory.
 If this directory does not exist , the command creates a directory called src in your current directory. Then it updates the IDhome file which contains all the directories names that have been created using this command. The IDhome file is present in your directory and is automatically created the first time you store a simulation_result file.

-once the src directory exists, the command stores in this directory the simulation_results file currently stored under the WWW name. The file is renamed in the case of our example G641628inmic.m . The G stands

for graph simulator, but you can also find simulation_results files whose name starts with an S standing for dataflow simulator. The .m suffixe at the end of the file is a convention for the Matlab language and you should not have to worry about this.
Then the parameters of the simulation are stored in the ID Ascii file existing in the src directory (this file is automatically created when the directory is itself created). The ID file contains the parameters of all the simulation_results files present in the src directory.

-Last, you will notice that a refload.mat file is also present in the src directory. This file is used by the display program and is too a special Matlab file. You should not worry about it either.

Retrieve_Simulation_File

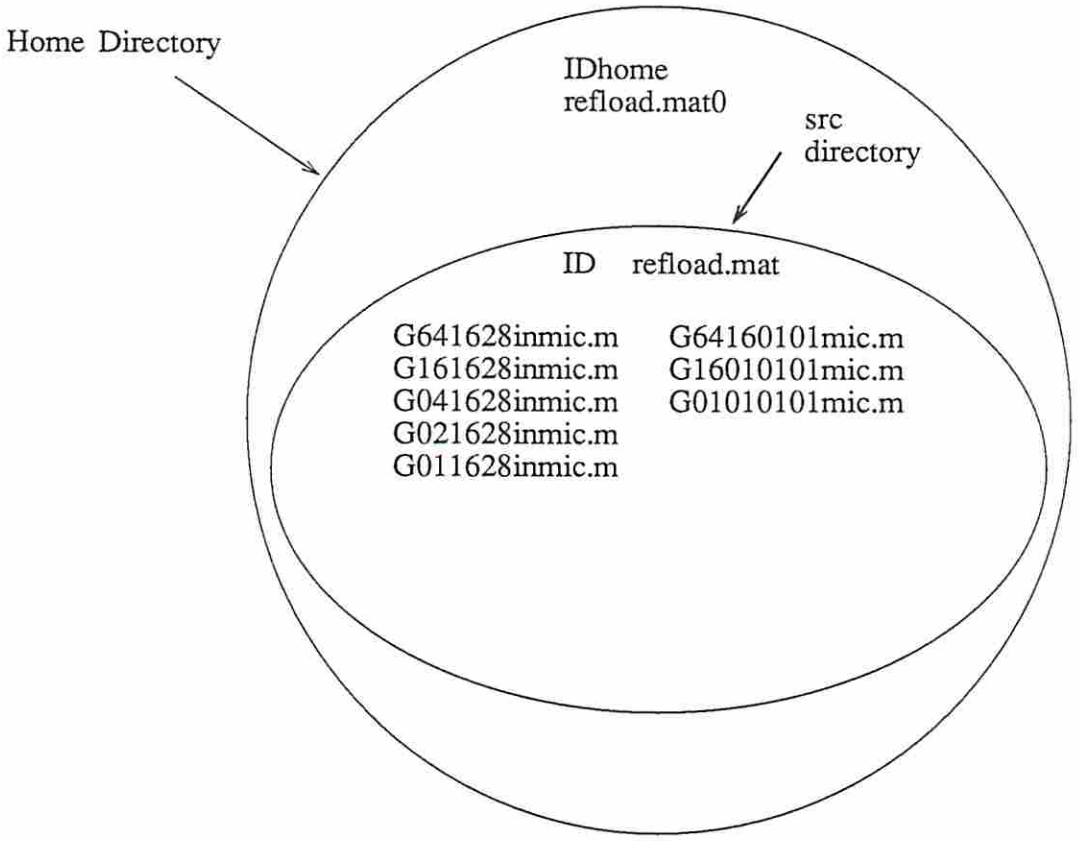
This function is for the moment (in the 1.0 version) mainly used by the simulator display program to handle the display of SpeedUp and Sensitivity of simulations and therefore will be explained later within the 'Xdisplay Window' part of this manual.

List_Current_Simu_IDs

this command will list in your shell/command window all the names (including directory) of the simulations_result files whose IDs match the informations you have indicated on the Xparam window. Within the dialog windows you can enter an existing (or not) parameter or put a '?' character as first character of the string to mean that you do not take care of the value of this parameter.

To make the things clear let us handle the following example:

Let us suppose that the files that follow have already been stored in your src directory.



Now if the parameters displayed on the Xparam window are:

Simulator type: graph simulator	
sourcefile_name :	src
#_of_PE_servers :	01
#_of_CE_servers :	01
#Str_Mem_servers :	01
#_of_Match :	01
suffixe :	mic

selecting the `List_Current_Simu_IDs` button will result in

```
***************************
*                         *
*   Xparam selected ID's  *
*                         *
***************************

src/G01010101mic
```

appearing in the shell/ command window. A 'listID' file is now also present in your current/Home directory. The listID contains the list of all the simulation_results files you have selected with the List_Current_Simu _IDs command.

If you now enter the following information in the Xparam window:

```
Simulator type : graph simulator

sourcefile_name  :   src

#_of_PE_servers :    ?

#_of_CE_servers:    16

#Str_Mem_servers  :  28

#_of_Match           :?

suffixe                : ?
```

you will obtain:

```
***********************
*                     *
*  Xparam selected ID's  *
*                     *
***********************

src/G641628inmic.m
src/G161628inmic.m
src/G041628inmic.m
src/G021628inmic.m
src/G011628inmic.m
```

You might also want to have the list of all the simulation_result files present in a given directory.
To obtain this result, just put '?' in all Xparam dialog widgets (except the name of the directory in the sourcefile_name box) and press the list_current_simu_IDs button.

IV the Xdisplay Window

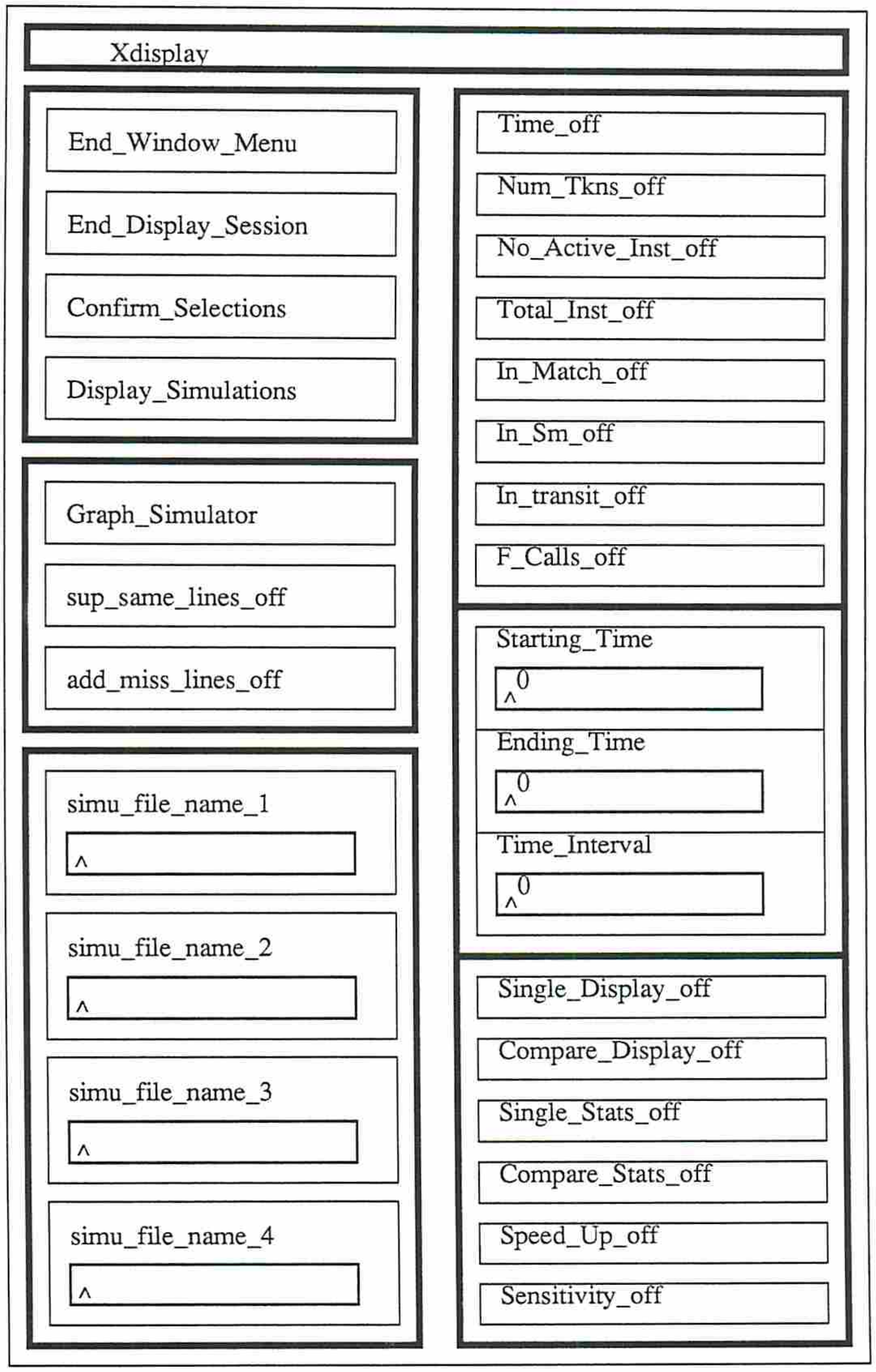
running the Matlab display program

Before running the Matlab display program , you must have made at least one display selection with the Xdisplay program (how to do this will be explained a bit later) . Then you can run the Matlab display program by typing 'displ' in the matlab command window. For the program to function correctly you must verify that the following files are stored in your current/Home directory. (These files are normally included in the package provided with the whole display environment) :

- saisinit.m
- loadname.m
- muldisp.m
- compln.m
- select.m
- supred.m
- traita.m
- selecdata.m
- lstatmat.m
- savstatmat.m
- efr.m
- efw.m

working with the Xdisplay window

On the next page is shown the 'Xdisplay' window as it should appear the first time you open the window in a simulations display session.


Xdisplay
End_Window_Menu
End_Display_Session
Confirm_Selections
Display_Simulations
Graph_Simulator
sup_same_lines_off
add_miss_lines_off
simu_file_name_1
^
simu_file_name_2
^
simu_file_name_3
^
simu_file_name_4
^
Time_off
Num_Tkns_off
No_Active_Inst_off
Total_Inst_off
In_Match_off
In_Sm_off
In_transit_off
F_Calls_off
Starting_Time
^0
Ending_Time
^0
Time_Interval
^0
Single_Display_off
Compare_Display_off
Single_Stats_off
Compare_Stats_off
Speed_Up_off
Sensitivity_off

The Xdisplay main functions

There are basically four main functions you can utilize (with this version of the simulator display interface) using the Xdisplay window. These are:

- ○ Display the trace of a simulation
- ○ Show statistics of a simulation
- ○ SpeedUp of simulations
- ○ Sensitivity

To begin with we will see how to obtain the statistics of a simulation (results reported by the simulator after simulation).

How to display simulations statistics

First of all you have to give the ID of the simulation_results file. In the appropriate dialog windows of the Xdisplay window enter this ID. You can see that there are four widgets that are used for this purpose: namely those labelled simu_file_name_1 to simu_file_name4. Supposing that you want to know the statistics of the simulation_result file you have just stored (src/G641628inmic) put her ID in the first dialog widget and put stars '*' in the three remaining windows. A star as first character of a string displayed in one of the simu_file_name dialog windows means that the corresponding simulation is not taken in account for any display purpose. Do not forget to put the stars whenever needed. If you do not funny results could appear (we personally doubt they would make you laugh).
When you enter the ID of the simulation do not forget to give the directory in which it is located and the '/' symbol (same as in the UNIX system) after, as part of the ID.

Then use the Confirm_Selections command button to confirm that the strings that are located in the dialog widgets are those really intended to display. If not, change them and when you are sure of your selection click on the command button.

As a rule of thumb always click on the confirm_selections button when you make a change in any one of the Xdisplay dialog widgets. Otherwise the results you would obtain could not be those you expected.

The next step is to click on the $\boxed{\text{Single\_Stats\_off}}$ button which is one of the command buttons in the south eastern part of the Xdisplay window (these buttons are used to specify the activation of the main functions that are implemented here). $\boxed{\text{Single\_Stats\_on}}$ will appear in place of the button you clicked upon meaning that the Single_Stats function is now activated.

Specify then the simulator that has been used to obtain the simulation by clicking on the correct button in the left-middle subwindow of the Xdisplay window. You have the choice between Graph_Simulator and Simulator and the selection is made in the same way as this is explained in page five of this manual.

You can now display the statistics of the simulation by clicking on the $\boxed{\text{Display\_Simulations}}$ command button. After a while you will see the statistics displayed in the Matlab command window.

In the 1.0 Version of the simulator display interface the following statistics are available:

○ Data_Flow Simulator:

- A_TKN the average number of tokens.
- M_TKN the maximum number of tokens.
- G_TKN the number of garbage tokens after completion of simulation.
- S_TIME the simulation time unit
- N_LOAD the accumulated load of the communication network.

○ Graph Simulator

the same statistics except N_LOAD

You can proceed in the same way if you want to check the statistics of several simulations and compare them. You just have to put all the IDs of the simulations_result files you want to know in the simu_file_name widgets of the Xdisplay window (maximum is four simulations, we will see later how to do if you need the statistics of more than four at the same time) and click on the 'Compare_Statistics ' Compare_Stats_off button to activate the function.

How to display the trace of a simulation

Before showing you how to handle this function let us remind you of the output trace that is issued by the simulator during a simulation.

☐ If you use the Graph Simulator, the following statistics are issued at specified intervals during simulation (in further parts of the manual we will rather use the term of data when we refer to these statistics to avoid the confusion with the statistics after simulation mentioned in the previous paragraph).

-Time: current simulating time.

-Numtkns: number of tokens issued at current time.

-No_of_Active_Inst: number of busy servers of processing elements (PE).

-Total_Inst: number of total instructions executed.

-In_Match: number of tokens at 'match' facility.

-In_Transit: number of tokens at communication element(CE)

-In_SM: number of tokens at Structure memory.

-F_Call: number of active function calls.

☐ If you use the DataFlow simulator the trace of output is

-Time
-Numtkns
-F_Call

The following steps are required to display the trace of a simulation:

☐ enter the ID of the simulation in a "simu_file_name" dialog widget.

☐ indicate the simulator type using the Graph_Simulator/Simulator command button.

☐ select the data you want to display using the buttons set in the top-right window. The data you have selected will be displayed as a function of the current simulating time.
To select any data just click on the corresponding button until "data_on" appears instead of "data_off".

☐ give the time specifications (use the "Staring_Time","Ending_Time", and "Time_Interval" dialog widgets). The trace that will be displayed in the Matlab graphing window will begin at the time you have entered in "Starting_Time" and will end at the Time you have entered in "Ending_Time" with the data displayed only every Time_Interval.
The only constraint you have here is that the time you select in "Ending_Time" should be equal or less to the total simulation time of the simulation_result file you have chosen to display.
So, if you are not sure of this specific time be sure to check it first using the function that allows you to display the statistics of a simulation.

☐ click on the "Single_Display" button until "Single_Display_on" appears to activate the trace function.

☐ confirm your selections using the "Confirm_Selections" command button.

☐ Click at last on the "Display_Simulations" command button and wait until the trace is displayed on the Matlab graphing window.

How to display the trace of several simulations at a time

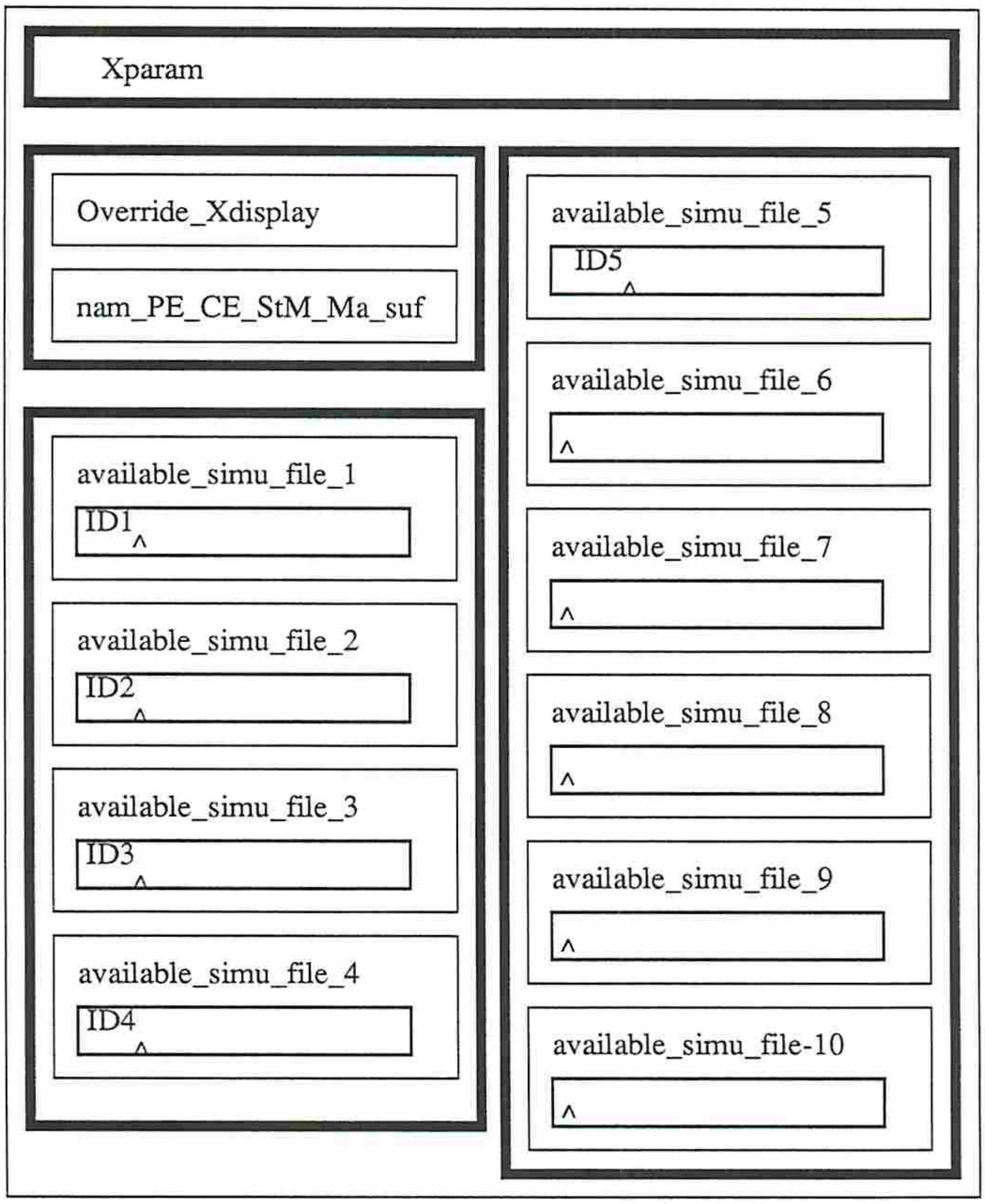
If you want to display the trace of more than one simulation, just put the IDs of all the simulations_file whose traces you intend to display in the "simu_file_name" dialog widgets. (do not forget to put stars in the unused widgets)
Select "Compare_Display_on" in the way we just saw in the previous paragraph and wait until the traces are displayed in the Matlab graphing window. You will notice that in the version 1.0 of the simulator display interface the Matlab graphing window is splitt into as many subwindows as the number of traces you have selected and that each subwindow is holding one trace. As the maximum number of windows is four we suggest that you try to display no more than four traces at a time. This will eventually be improved in later releases of the simulator display interface if many users think that a maximum of four is to restrictive.

Sensitivity

It is often useful to check the sensitivity of a simulation towards the change of one (or more?) parameters.
We suppose here as an example that you wish to study the impact of the number of communicating elements servers on the results of a simulation.
Let us show you how you can do this using our simulator display interface .

- first step is to come back to the Xparam window to select and specify the values of all the parameters except the one you have chosen to study Sensitivity. In our case you would give the values of parameters #_of_PE_servers,#Str_Mem_servers,#_of_Match and of course the sourcefile_name and suffix. The #_of_CE_servers window would be filled with a '?' as you do not know what are the simulations IDs that match the values you have given for the other parameters.
- use the List_Current_Simu_IDs command button to verify that the number of simulations IDs that you have selected in this way does not exceed ten (in the 1.0 version of the simulator display interface). If not you won't be able to proceed towards the following steps.
- click on the **Retrieve_Simu_File** button.

After you have done this the Xparam window will swap its contents to display a window that has the outlook shown bellow.



You can see that there are ten dialog widgets labelled "available_simu_filei" and that some of them have their dialog box filled with an ID name. All the IDs displayed in the dialog widgets are those you have precedently selected. If there are more than five of them select four of them by putting a star '*' on those you wish to discard (again in later releases of the simulator display interface this number might be increased).

- ○ At this time if you are not satisfied with the IDs you have selected or if you want to change the sensitivity parameter or if you do not finally intend to display the Sensitivity you have the option to come back to the Xparam origin window without taking in account the IDS you have selected.
  To do this click on the $\boxed{\text{nam\_PE\_CE\_StM\_Ma\_suf}}$ button. The label of this button has for purpose to remind you of the order in which the #PE,#CE ... are arranged to create the simulation ID.

- ○ then click on the $\boxed{\text{Override\_Xdisplay}}$ button to confirm your selection.
  You must carefully notice that whenever you do this the IDs that are displayed in the "available_simu_filei" boxes will only be taken in account for display purpose when you execute the next main command with Xdisplay (such as Sensitivity or SpeedUP). The IDs name you could have given in the Xdisplay "simu_file_name" boxes will not be taken in account (actually they will be overwritten).

  (you can see now another way to enter simulations IDs when you have to use them : you just have to write or modify the Xparam "available_simu_filei" according to your purposes and click on the Xdisplay Override_Xmenu button)

- ○ Last step is to come back to the Xdisplay window, select the data you want to display (if you want to study the sensitivity of the trace to a parameter) and give the time specifications, or select the statistics function if you merely intend to see the sensitivity of the statistics towards the change of the sensitivity parameter .
  Click on the $\boxed{\text{Sensitivity}}$ button until "Sensitivity_on" appears to activate the Sensitivity function.

- ○ click on "Confirm_Selections" and then on "Display_Simulations" to display the Sensitivity results on the Matlab graphing and/or command window.

Speed_Up

The last main function the simulator display interface 1.0 version accomplishes is to compute and display the SpeedUP .
This is achieved by taking the following steps:

- using the Xparam window specify all the parameters of the simulation whose SpeedUP you expect to compute and display, except for the #_of_CE_servers parameters in which you put a '?'.
  (use the List_Current_simu_IDs button to verify that their number is less or equal than ten).

- click on the retrieve_simu_files Xparam command button. Again a window will replace the Xparam window and this window will contain all the IDs that are selected for SpeedUP. Discard the IDs you do not need by placing a '*' in the corresponding "available_simu_file_i" dialog widgets (you can keep all the IDs if you wish).

- click on the "Override_Xdisplay" Xparam button and come back to the Xdisplay window

- click on the Speed_Up command button until "Speed_Up_on" appears to activate the Speed_Up function.

- click on the "Display_Simulations" Xdisplay command button to display the Speed_Up.

- The curve of the SpeedUp as a function of the number of processing elements servers (PE) appears now on the Matlab graphing window.

"minor" Simulator Display Interface functions

sup_same_lines

When the simulator stores its results in an ascii file it can happen that there are two or more lines with the same simulating time and different corresponding datas. The purpose of this function is to keep only the first of such lines whenever this occurs and to store the resulting file (after this processing) in a way easily manageable by Matlab (the ascii file is not modified).
In the 1.0 version of the simulator display interface, this function is working only on the simulation_result file whose ID is specified in the simu_file_name_1 Xdisplay dialog widget.

add_miss_lines

When the simulator stores its results in the ascii file, if it occurs that the value of the "data" did not change during a period of time including more than one simulating time, only the first simulating time and its associated data is recorded in the ascii file. The purpose of this function is to "restore" all these missing lines and store the resulting file in a way easily manageable by Matlab (again, the ascii file is not modified).
As for the "sup_same_lines" function, "add_miss_lines" is working upon one file only (whose ID is written in the "simu_file_name" widget).

There are two things we would like to mention here:

- first, when you use these functions always use the first one before the second or use both at the same time.

-second you should not try to work on a simulation_result_file if you are not sure that it has been processed with these two functions or that the associated ascii file is of the form that you should obtain after processing by these two functions. (failure to do so could result in misfunctionning of the simulator display interface).

So we suggest you to always use these functions the first time you store a simulation_result_file (with Xparam). This has two advantages: you will be sure that the display you plan to do later will work properly and your simulation_file will be stored in a way easily readable by Matlab which will save time on later sessions.

Ending a session

If you want to end a simulator display session, you should do the following:

-click on the End_Display_Session button. This will end
the "displ" Matlab program.

-click on the End_Window_Menu button will then exit the Xdisplay window. (you can do the same with Xparam).

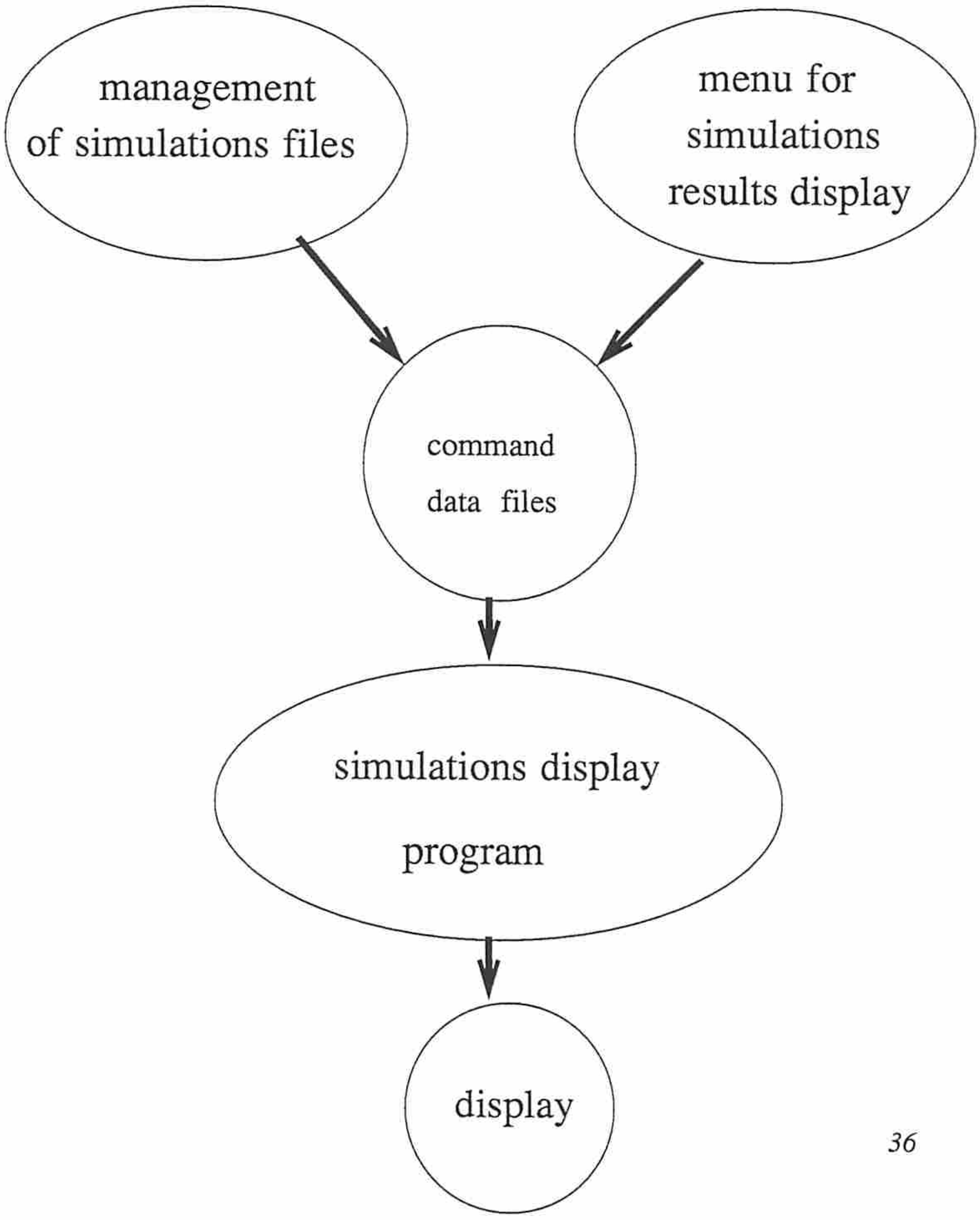
-quit the Matlab command window.

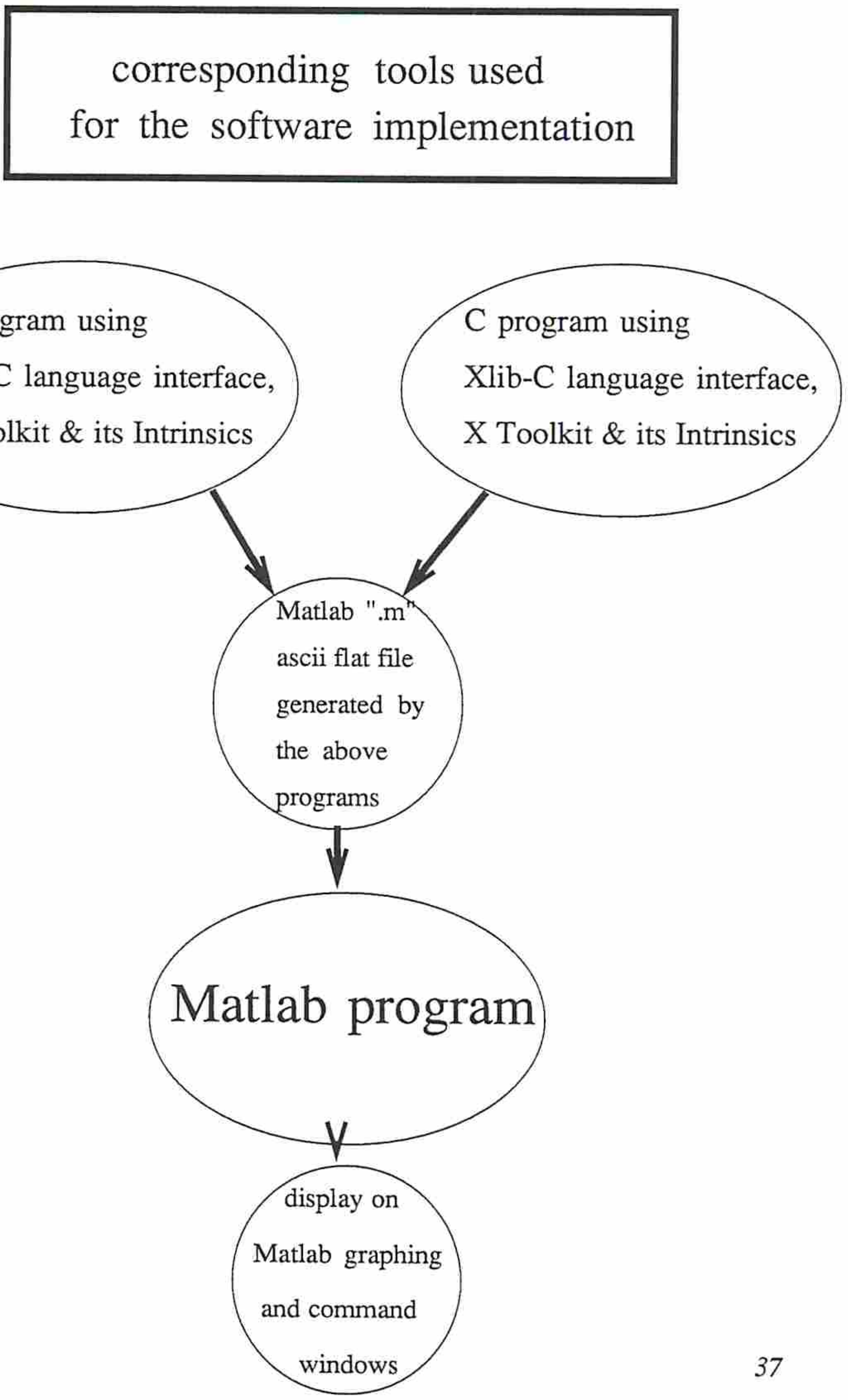
REFERENCES

ALGOS

This is an overall description
of the algorithms aimed
at making understand the main
structure of the program and the
implementation of the major
algorithms used. For more
information please refer to the
listings of the program. Take also
a look at the user's manual

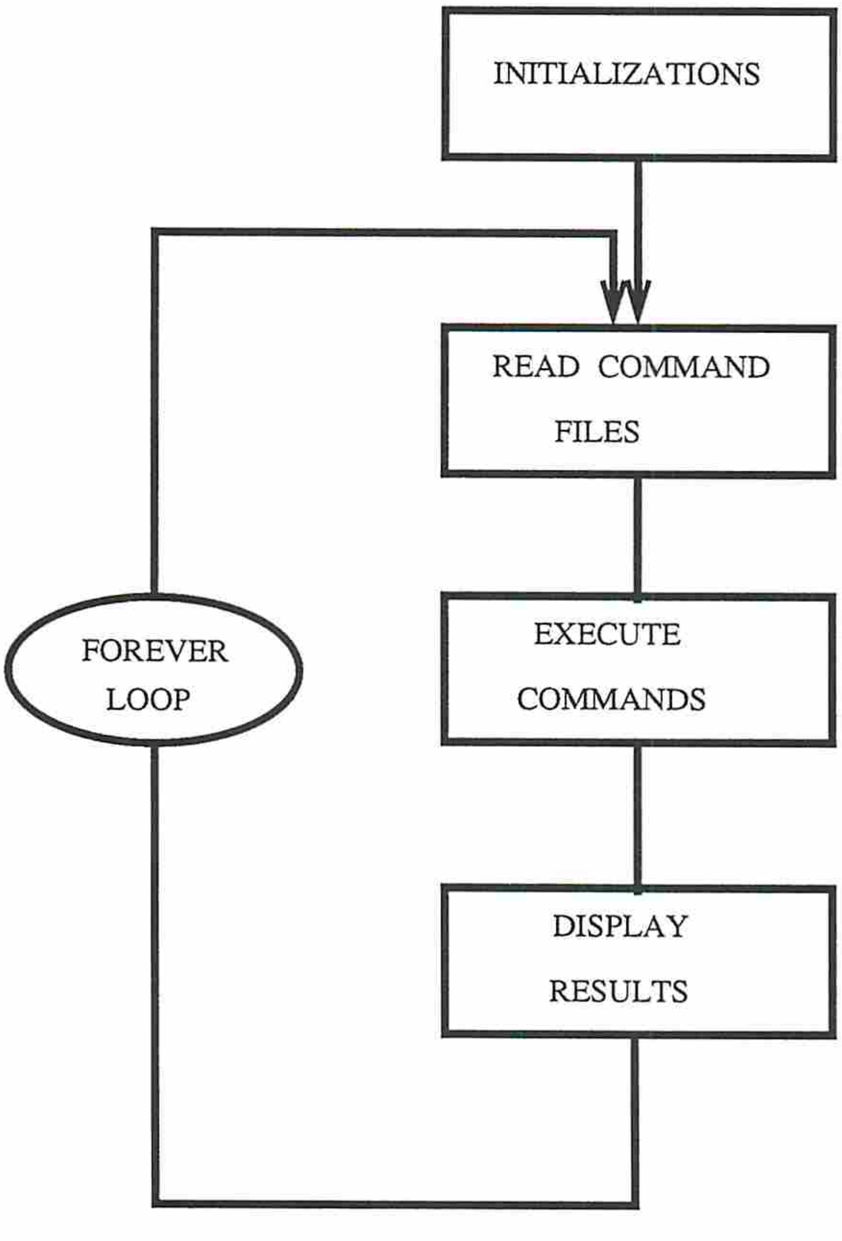
General Organization


management of simulations files
menu for simulations results display
command data files
simulations display program
display


corresponding tools used for the software implementation
C program using Xlib-C language interface, X Toolkit & its Intrinsics
C program using Xlib-C language interface, X Toolkit & its Intrinsics
Matlab ".m" ascii flat file generated by the above programs
Matlab program
display on Matlab graphing and command windows

ARCHITECTURE of the MATLAB

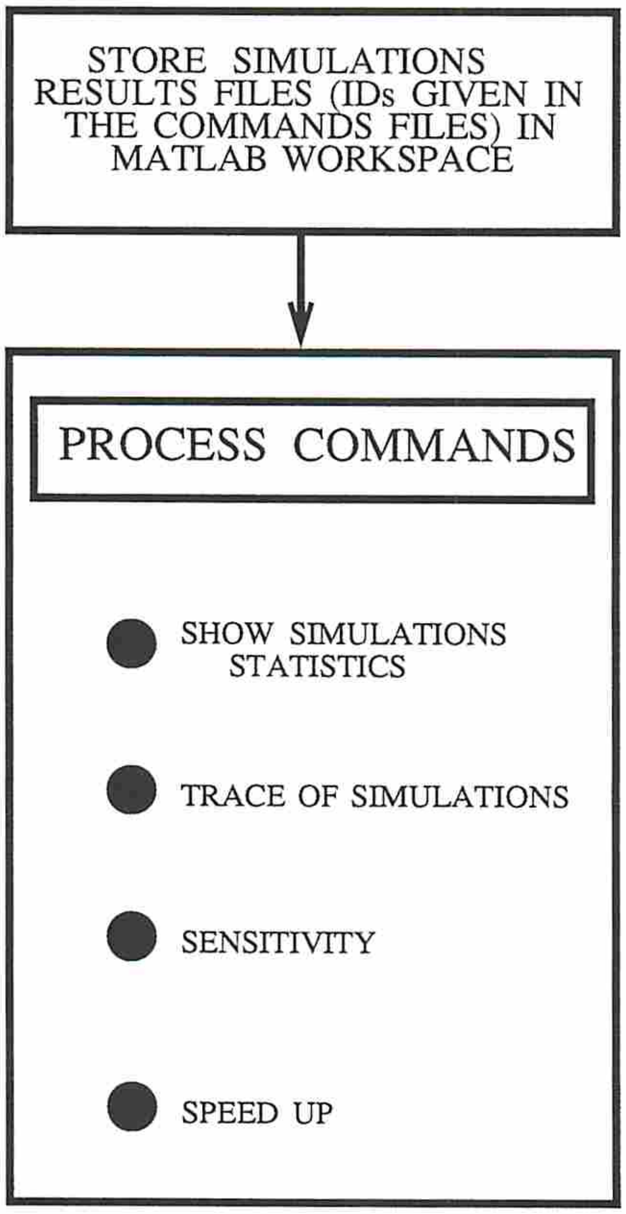
DISPL.M PROGRAM



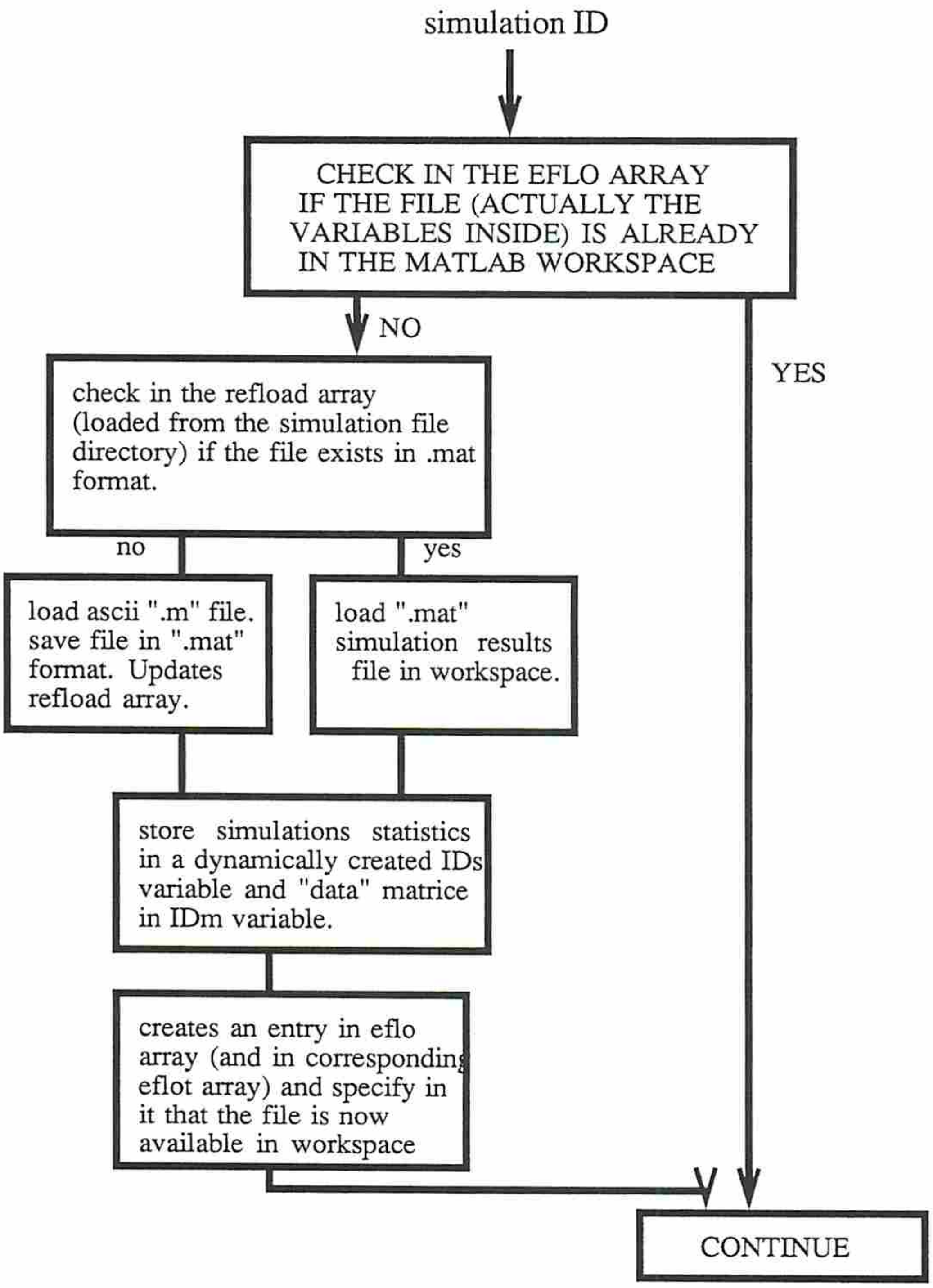
COMMAND FILES ORGANIZATION

- ■ SIMULATIONS IDs
- ■ TIME SPECIFICATIONS
- ■ SELECTED DATA
- ■ DISPLAY COMMANDS
  (SPEED UP, SENSITIVITY . . .)

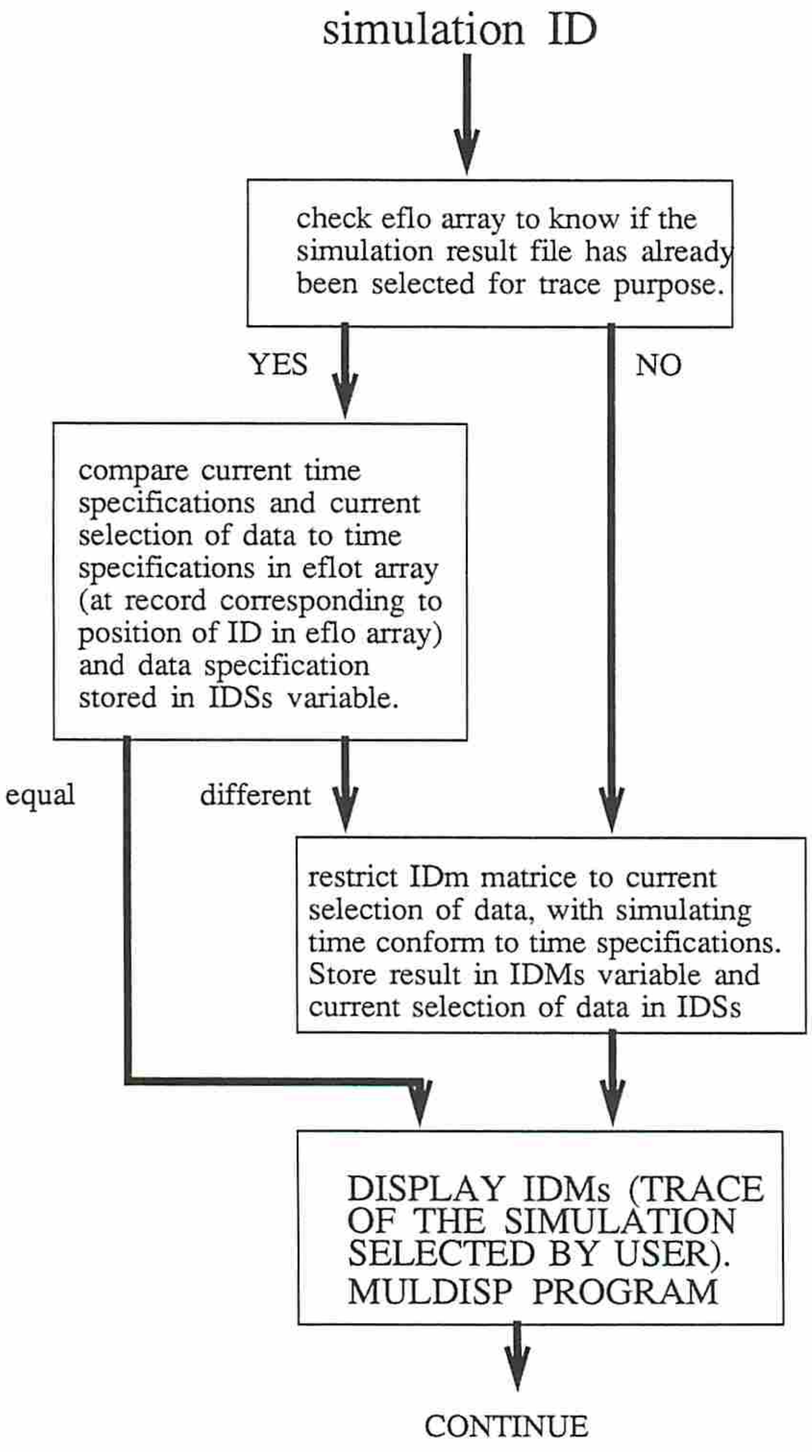
MAIN LOOP OF MATLAB DISPL.M PROGRAM


STORE SIMULATIONS RESULTS FILES (IDs GIVEN IN THE COMMANDS FILES) IN MATLAB WORKSPACE
PROCESS COMMANDS
SHOW SIMULATIONS STATISTICS
TRACE OF SIMULATIONS
SENSITIVITY
SPEED UP

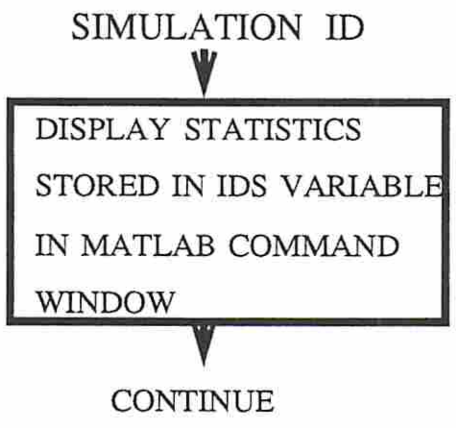
STORE SIMULATIONS RESULTS FILES IN MATLAB WORKSPACE


simulation ID
CHECK IN THE EFLO ARRAY IF THE FILE (ACTUALLY THE VARIABLES INSIDE) IS ALREADY IN THE MATLAB WORKSPACE
NO
YES
check in the refload array (loaded from the simulation file directory) if the file exists in .mat format.
no
yes
load ascii ".m" file. save file in ".mat" format. Updates refload array.
load ".mat" simulation results file in workspace.
store simulations statistics in a dynamically created IDs variable and "data" matrice in IDm variable.
creates an entry in eflo array (and in corresponding eflot array) and specify in it that the file is now available in workspace
CONTINUE

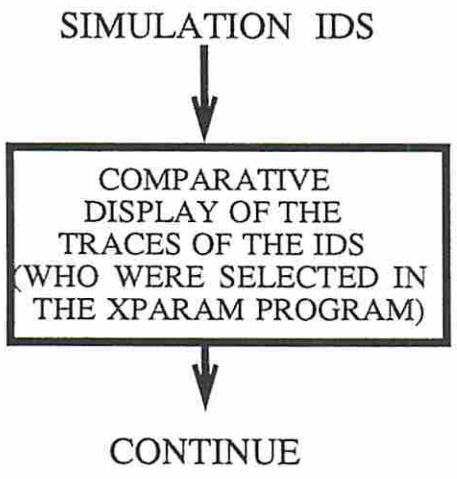
TRACE OF SIMULATIONS


simulation ID
check eflo array to know if the simulation result file has already been selected for trace purpose.
YES
NO
compare current time specifications and current selection of data to time specifications in eflot array (at record corresponding to position of ID in eflo array) and data specification stored in IDSs variable.
equal
different
restrict IDm matrice to current selection of data, with simulating time conform to time specifications. Store result in IDMs variable and current selection of data in IDSs
DISPLAY IDMs (TRACE OF THE SIMULATION SELECTED BY USER). MULDISP PROGRAM
CONTINUE

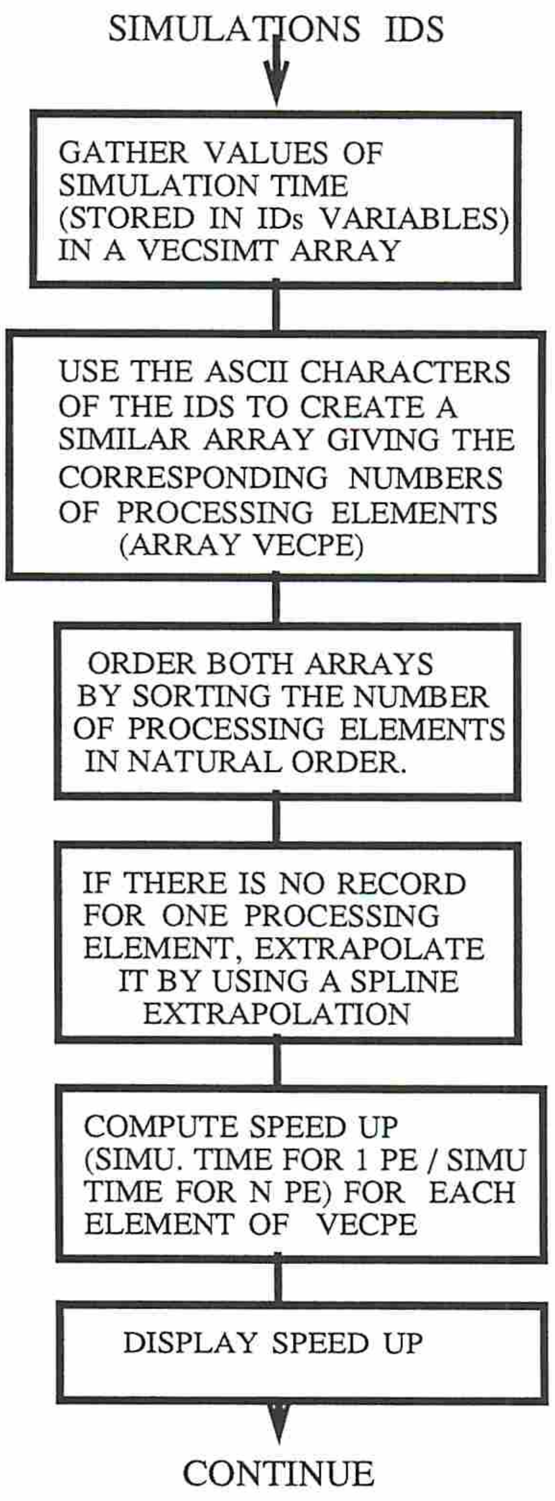
SHOW SIMULATION STATISTICS


SIMULATION ID
DISPLAY STATISTICS STORED IN IDS VARIABLE IN MATLAB COMMAND WINDOW
CONTINUE

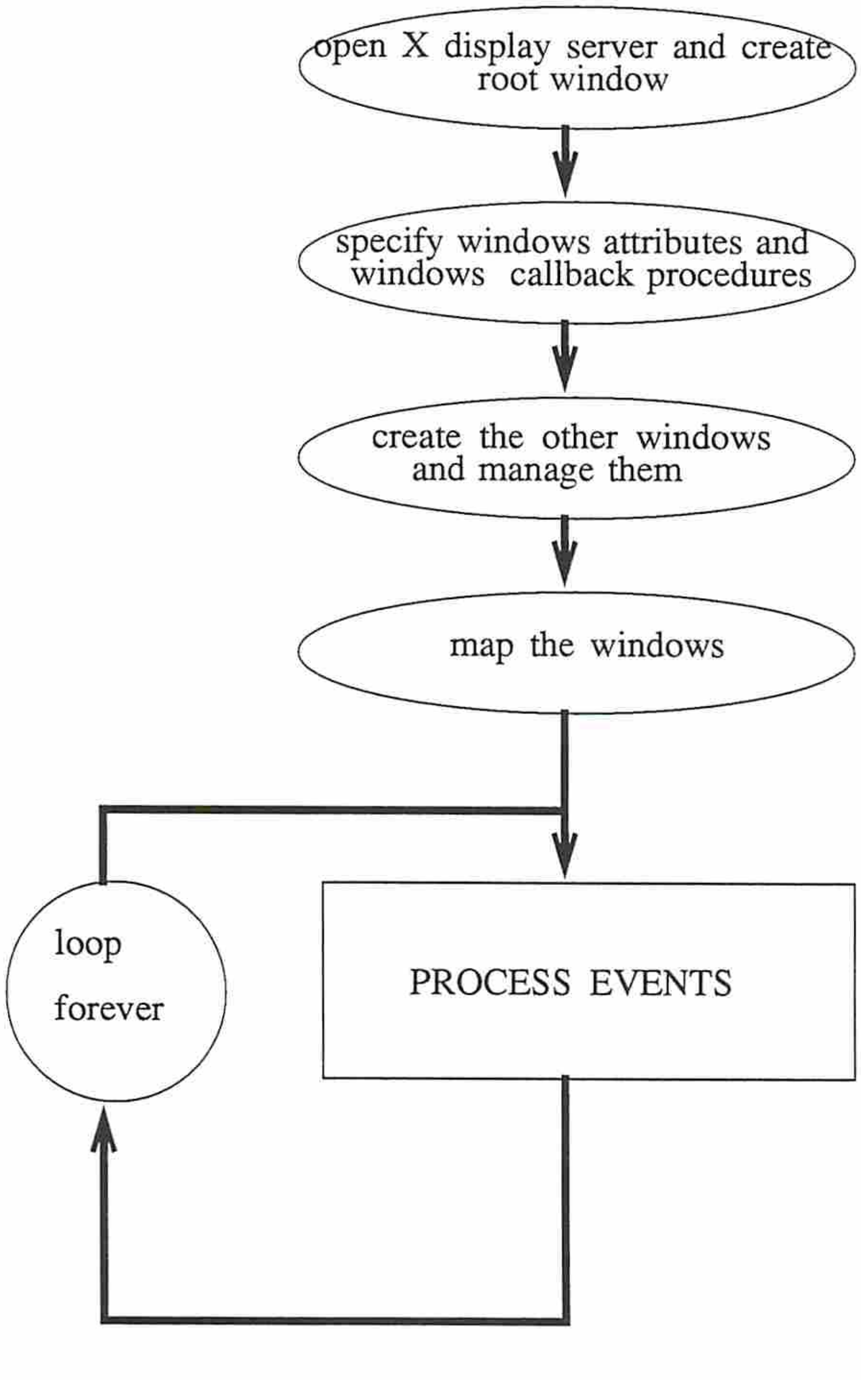
SENSITIVITY


SIMULATION IDS
COMPARATIVE DISPLAY OF THE TRACES OF THE IDS (WHO WERE SELECTED IN THE XPARAM PROGRAM)
CONTINUE

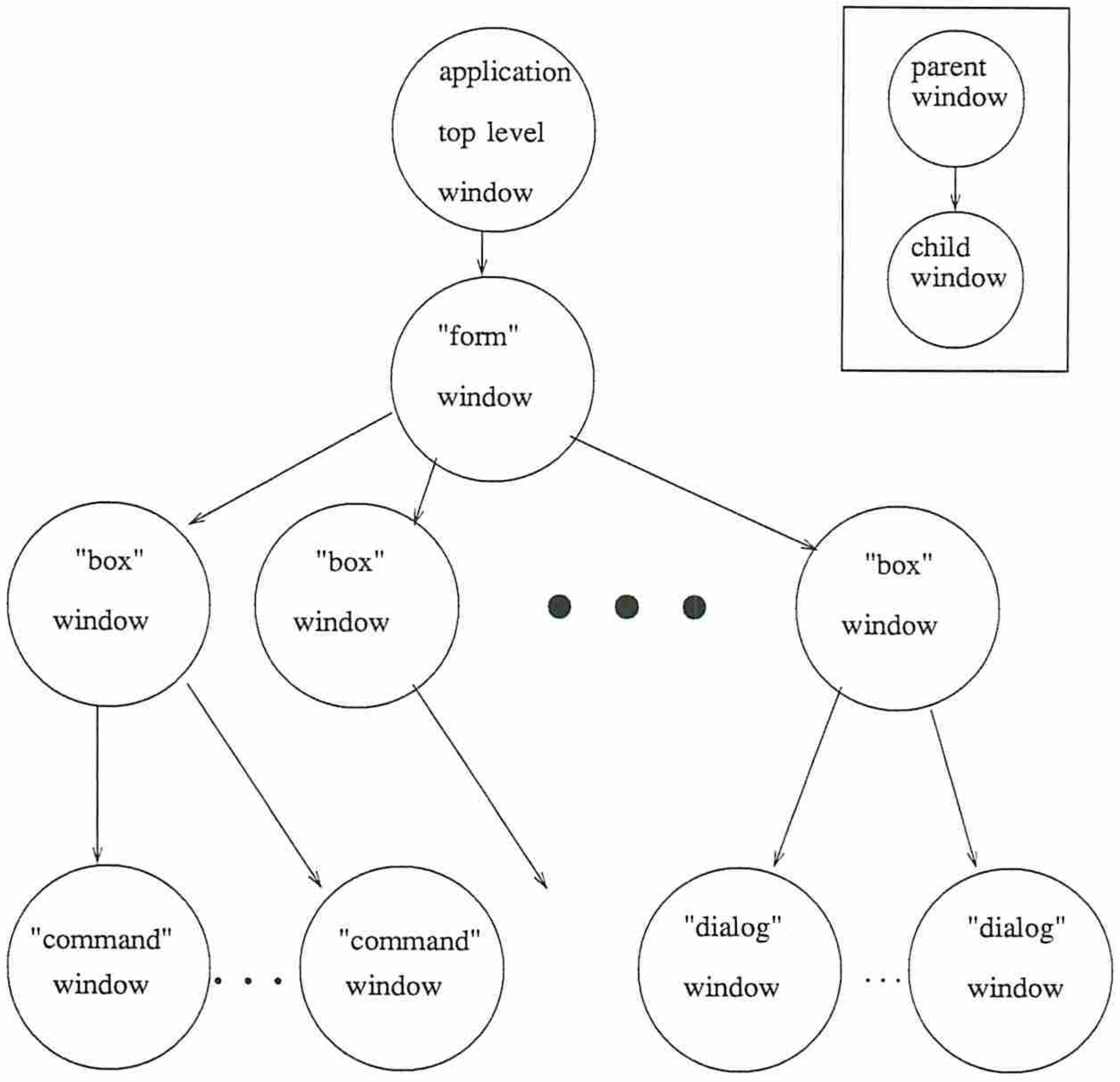
SPEED UP

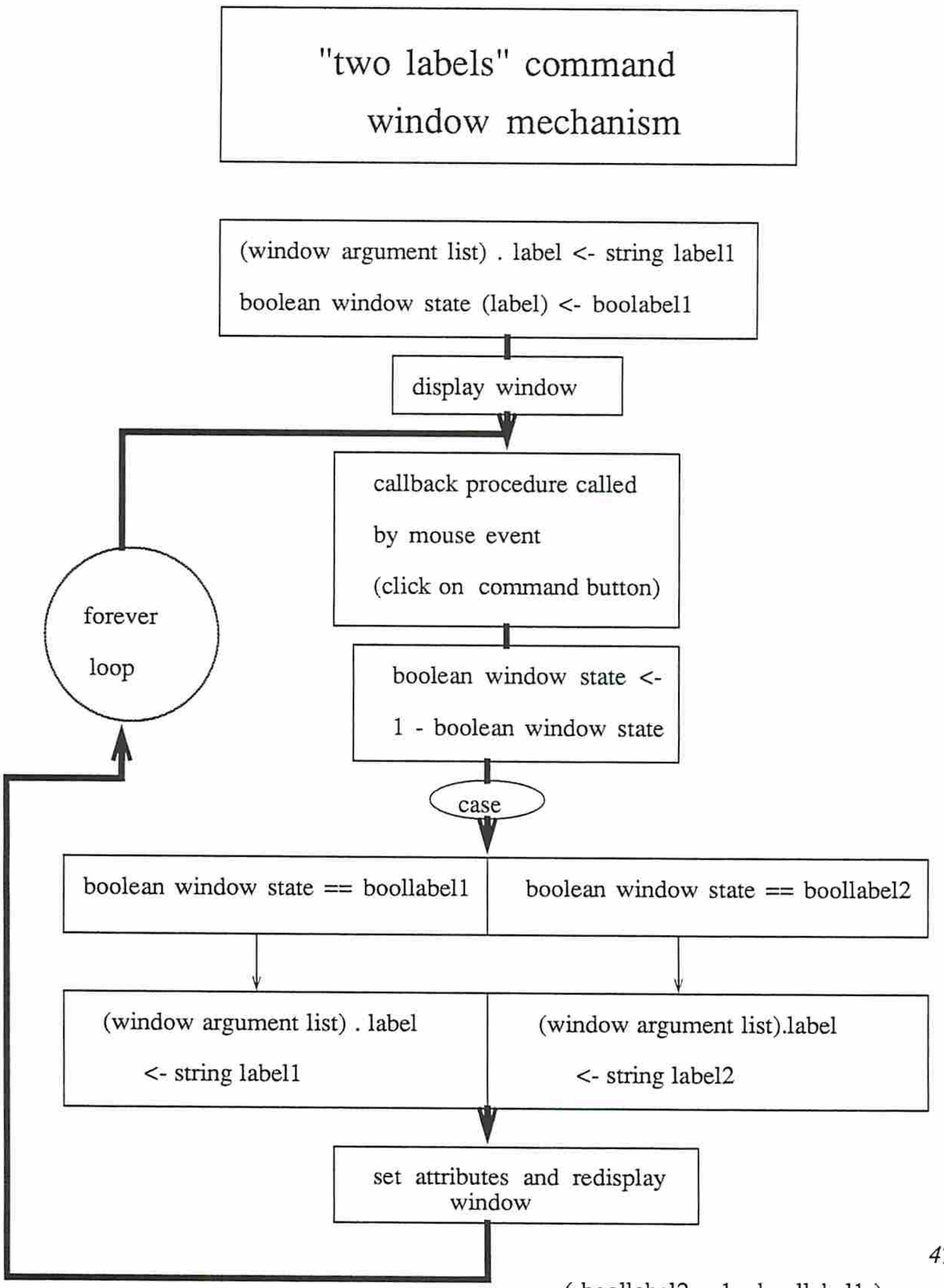

SIMULATIONS IDS
GATHER VALUES OF SIMULATION TIME (STORED IN IDs VARIABLES) IN A VECSIMT ARRAY
USE THE ASCII CHARACTERS OF THE IDS TO CREATE A SIMILAR ARRAY GIVING THE CORRESPONDING NUMBERS OF PROCESSING ELEMENTS (ARRAY VECPE)
ORDER BOTH ARRAYS BY SORTING THE NUMBER OF PROCESSING ELEMENTS IN NATURAL ORDER.
IF THERE IS NO RECORD FOR ONE PROCESSING ELEMENT, EXTRAPOLATE IT BY USING A SPLINE EXTRAPOLATION
COMPUTE SPEED UP (SIMU. TIME FOR 1 PE / SIMU TIME FOR N PE) FOR EACH ELEMENT OF VECPE
DISPLAY SPEED UP
CONTINUE

CREATE A MENU IN X ENVIRONMENT


open X display server and create root window
specify windows attributes and windows callback procedures
create the other windows and manage them
map the windows
PROCESS EVENTS
loop forever

hierarchy of the windows used
in X display and Xparam menus




"two labels" command window mechanism
(window argument list) . label <- string label1
boolean window state (label) <- boolabel1
display window
callback procedure called by mouse event (click on command button)
forever loop
boolean window state <- 1 - boolean window state
case
boolean window state == boollabel1
boolean window state == boollabel2
(window argument list) . label <- string label1
(window argument list).label <- string label2
set attributes and redisplay window

(boollabel2 = 1 - boollabel1)

ALGORITHMS OF CALLBACK PROCEDURES USED IN XDISPLAY

End Window_Menu command button :

- exit XDISPLAY PROGRAM

End_Display_Session command button :

-open "saistab.m" command datafile

-write command to assign end to "end of session" matlab variable

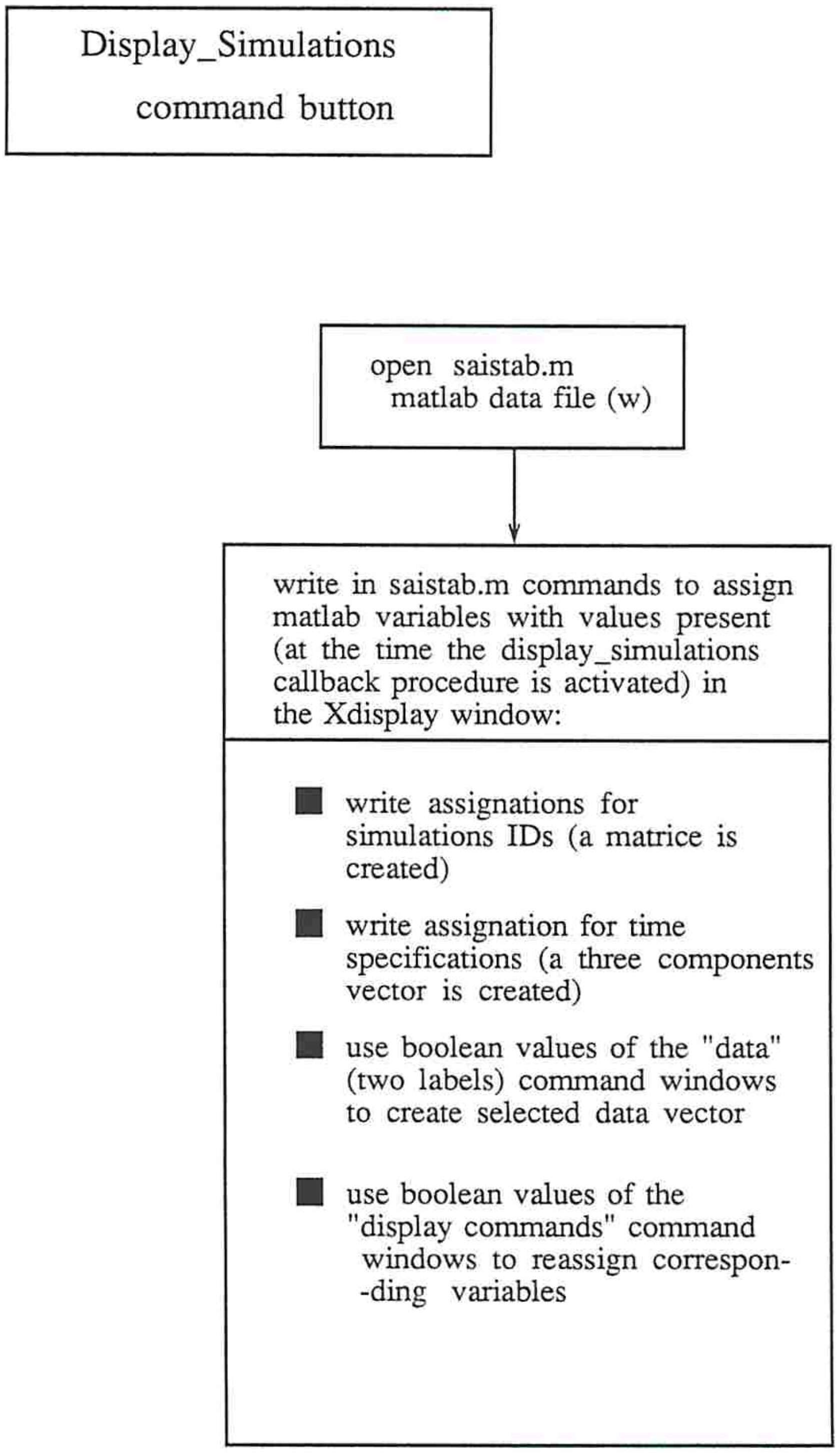
Confirm_Selections command button

-read simulation IDs strings present in simulation IDs dialog widgets

- assign the strings in memory with these read values (the program displays always in the dialog part of the dialog widgets the value of the corresponding string in memory)

- read time specifications strings present in time specifications dialog widgets.

-assign the corresponding strings in memory with these values



Display_Simulations command button

open saistab.m matlab data file (w)

write in saistab.m commands to assign matlab variables with values present (at the time the display_simulations callback procedure is activated) in the Xdisplay window:

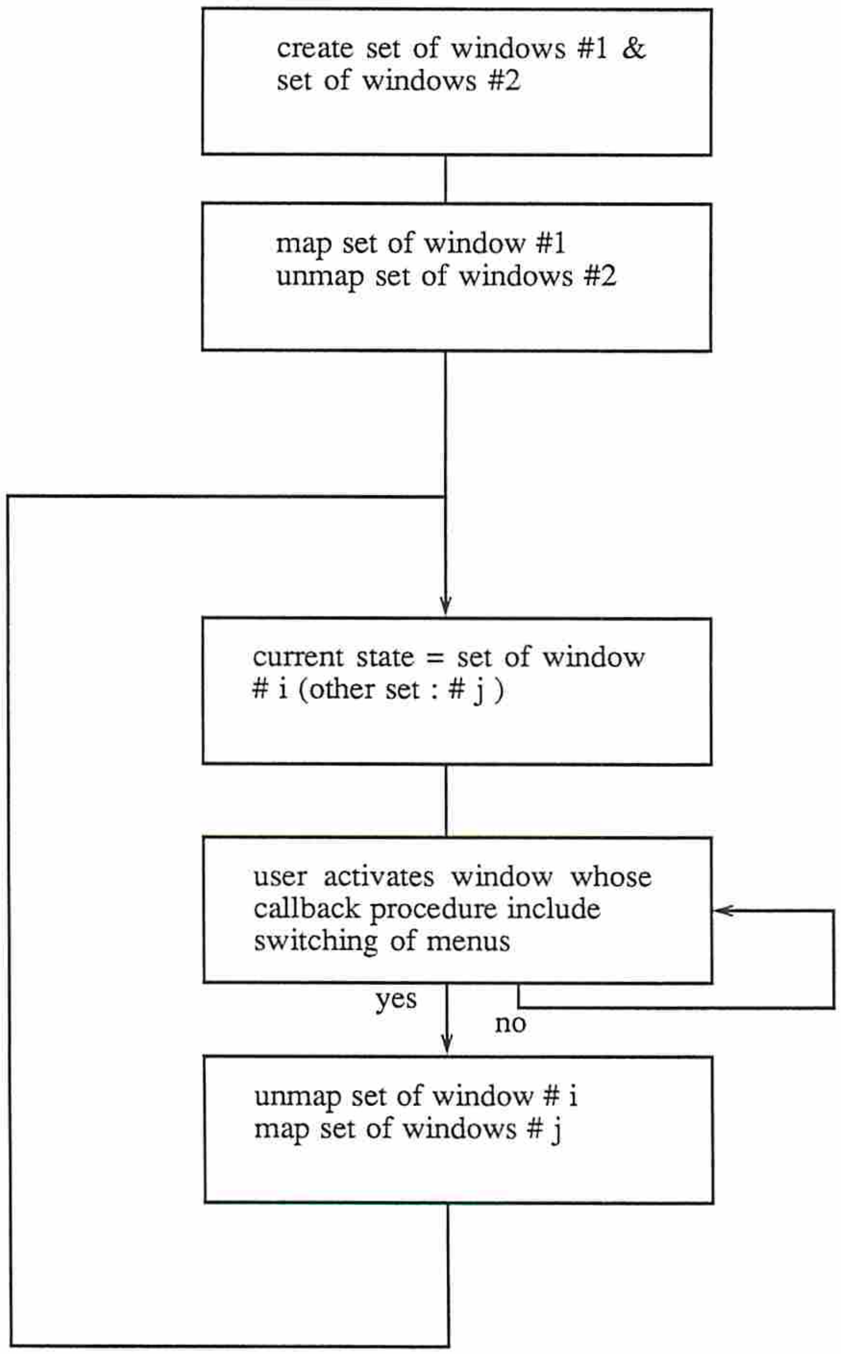
■ write assignations for simulations IDs (a matrice is created)

■ write assignation for time specifications (a three components vector is created)

■ use boolean values of the "data" (two labels) command windows to create selected data vector

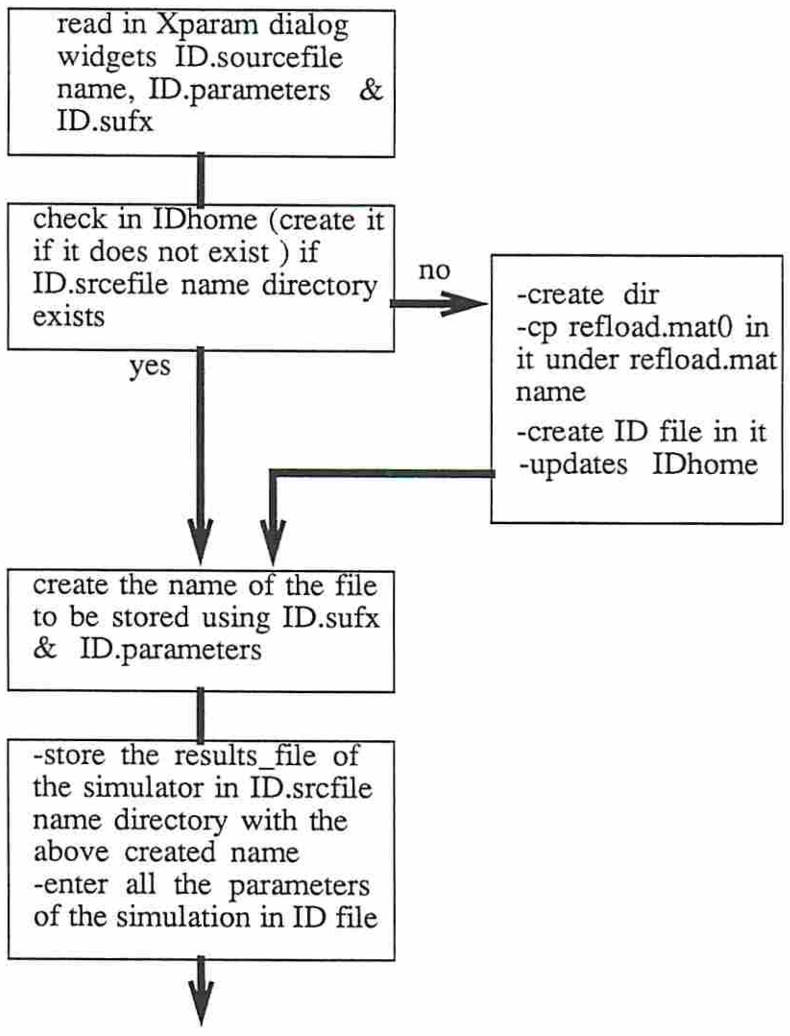
■ use boolean values of the "display commands" command windows to reassign correspon- -ding variables

XPARAM "TWO MENUS" SWITCHING MECHANISM

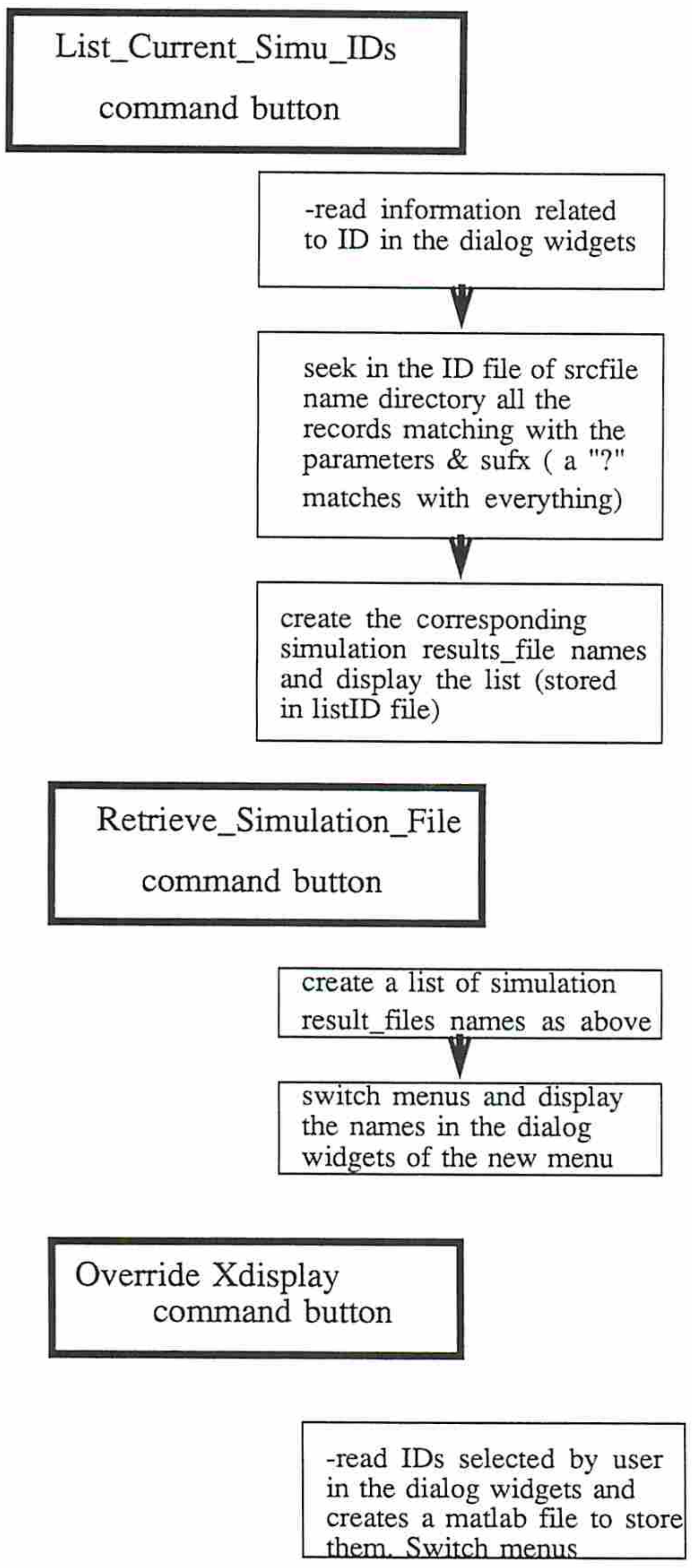

create set of windows #1 & set of windows #2
map set of window #1 unmap set of windows #2
current state = set of window # i (other set : # j)
user activates window whose callback procedure include switching of menus
yes
no
unmap set of window # i map set of windows # j

ALGORITHMS OF CALLBACK PROCEDURES USED IN XPARAM

Store_Simulation_File command button


read in Xparam dialog widgets ID.sourcefile name, ID.parameters & ID.sufx
check in IDhome (create it if it does not exist) if ID.srcefile name directory exists
no
yes
-create dir
-cp refload.mat0 in it under refload.mat name
-create ID file in it
-updates IDhome
create the name of the file to be stored using ID.sufx & ID.parameters
-store the results_file of the simulator in ID.srcfile name directory with the above created name
-enter all the parameters of the simulation in ID file

continue



List_Current_Simu_IDs
command button

-read information related to ID in the dialog widgets

seek in the ID file of srcfile name directory all the records matching with the parameters & sufx (a "?" matches with everything)

create the corresponding simulation results_file names and display the list (stored in listID file)

Retrieve_Simulation_File
command button

create a list of simulation result_files names as above

switch menus and display the names in the dialog widgets of the new menu

Override Xdisplay
command button

-read IDs selected by user in the dialog widgets and creates a matlab file to store them. Switch menus

FURTHER ISSUES

I Product final outlook :

The software will be delivered as a package that will allow the user to access the following functions when first run : run a simulation , store simulation and parameters, display simulation results. This basic menu will allow him to access the other simulator display interface window.

RUN SIMULATION	STORE SIMULATION & PARAMETERS	DISPLAY RESULTS

II Selection of Simulations files :

In the 1.0 version of the simulator display interface, the simulations can be retrieved by the user if he enters the correct ID in the dialog window. In the next version he should be able to do this with the mouse without typing anything. A program using special text widgets will be implemented allowing the user to "scroll" in the directories reserved for simulations storage and select the simulations by a mere clicking. This will be another step towards freeing the user of constraining (and sometimes error resulting) tasks.

III Display of results in the course of the simulation :

It can be interesting to display the trace of a simulation while it is running . This can be especially useful if one plans to control the stopping of the simulation. For example if the results of the simulation go out of bound or if they seem totally uninteresting, to stop the simulation can save a lot of (CPU and user) time. The simulation could also be automatically stopped on triggering specified conditions. This function could add a supplementary comfort to the simulator display interface user.

Basically, the steps required for the algorithm should be :

- transfer simulation result each time it is issued by the simulator to the display program .
  If the matlab display program is used, care should be taken to the fact that it takes some time for matlab to extract results from an ascii flat file. In this case a solution could be that the simulator stores a certain amount of results before releasing them to the display program (this amount could be specified and tests made to see what it should be for the display to be efficient).
  Another solution could be expand the simulator program to allow the display on time (for example using graphics C libraries) . Of course one would lose part of the benefit of the existing software.

- verify the coherency of the simulation time with last issued result and eventually make corrections (this step can be connected to the add_miss_lines and sup_same_lines commands of the simulator display interface Xdisplay window). That is if (actual simulation time - last simulation time) / simulation time unit doesn't equal 1 changes are needed (intermediary results should be supplied or actual result invalidated).

- display newly available results (the results should be either stored by the simulator program, and reprocessed after or by the Matlab display program. In the last case the simulator should not write itself the results because this would cause a waste of time)

- The last step supposes that the display window scale has been computed to fit the results. Therefore, the following algorithm is proposed:

  * before a simulation starts an horizontal scale is allocated corresponding to p times the simulation time unit (let us call this parameter hoscal). The vertical scaling is not decided yet. Let us suppose it is 0 .
  We initiate two parameters hor and ver that are the values of the vertical and horizontal spaces reserved for display in our window (actually half the value of these spaces). Each time a simulation result is issued we compare it with the value of ver and we do the same between hor and the simulation time unit.
  * If simu_res > ver then ver <- 2*simu_res and we redisplay all the results using the new scaled window.
  In a same way, if simu_time > hor we extend the vertical scaling with hoscal*simulation time unit and redisplay the results

The purpose of this is to not have to refresh the window (waste of time and unpleasant "swapping" effect) every time a new result is issued by the simulator. Old plots can be frozen in the window and new plots added after.

IV display improvement :

The graphing possibilities of matlab are not for the moment fully exploited. Actually, all the structures are now at hand to display results in a more complex way than a simple data vs time representation. For example the mesh matlab function allowing to display 3D curves can be easily used. Other solutions to display the information will eventually be achieved depending on the needs of the users.

V parameters :

The present version of the simulator display interface allows the user to give an ID to a simulation as a function of the parameters used to run the simulation. However other parameters have been discarded (actually a suffix is provide to keep a track of them). A special window should be created to store all the other parameters. The information will be used to complete the parameter record associated with every simulation result file.

VI Other Functions :

The version of the simulator display interface discussed here is the first one. As such it not intended to be complete , but rather to provide the basement on which further issues will be added. The users may feel the need to improve the actual version with new functions or many other modifications. The way it was designed should allow this. So if you have any other ideas don't hesitate to share them, they might be sooner or later implemented.

CONCLUSION

The software that has been presented here and whose programs listings follow is the "first stone" in the design of a large environment centered around the U.S.C. macro dataflow simulator.

With the structures used in implementing the software expanding the environment should be easy (actually far easier than designing a such environment and tool from scratch).

But before someone starts to expand the environment, the Simulator Display Interface provides the users with a crucial tool that will allow them to concentrate on their research and avoid wasting time in doing the job in a non automatic ways.

I sincerely hope that it will help them greatly.

BIBLIOGRAPHY

[1] Sun Microsystem
" C programmers Guide"

[2] B.W Kernighan and D.M. Ritchie
" The C programming language"

[3] S.R. Bourne
" The Unix System"

[4] Berkeley Software Distribution
"Unix User's Reference Manual"

[5] R.R. Swich and T. Weissman
"X Toolkit Athena Widgets- C Language Interface"
X window System

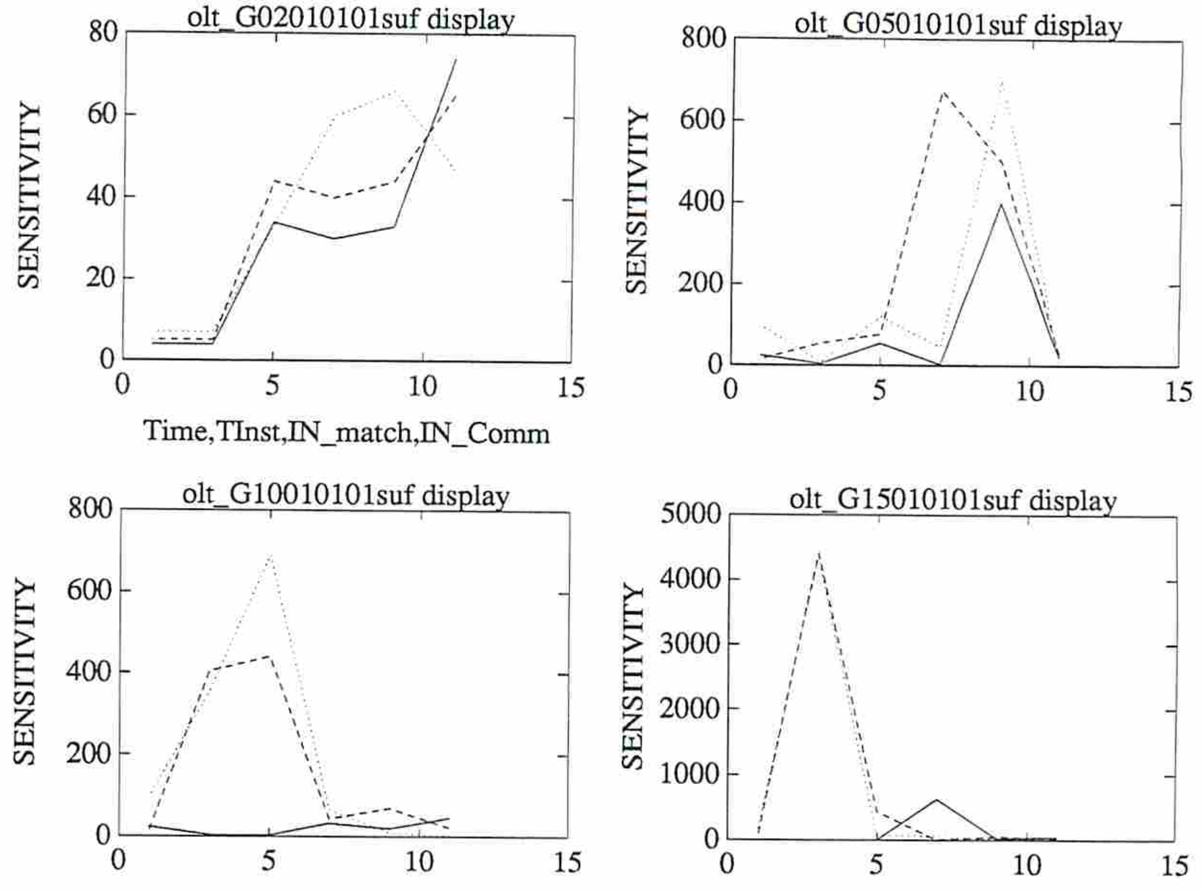
[6] J. Mc Cormack, P. Asente and R.R. Swich
"X Toolkit Intrinsics - C Language Interface"
X window System

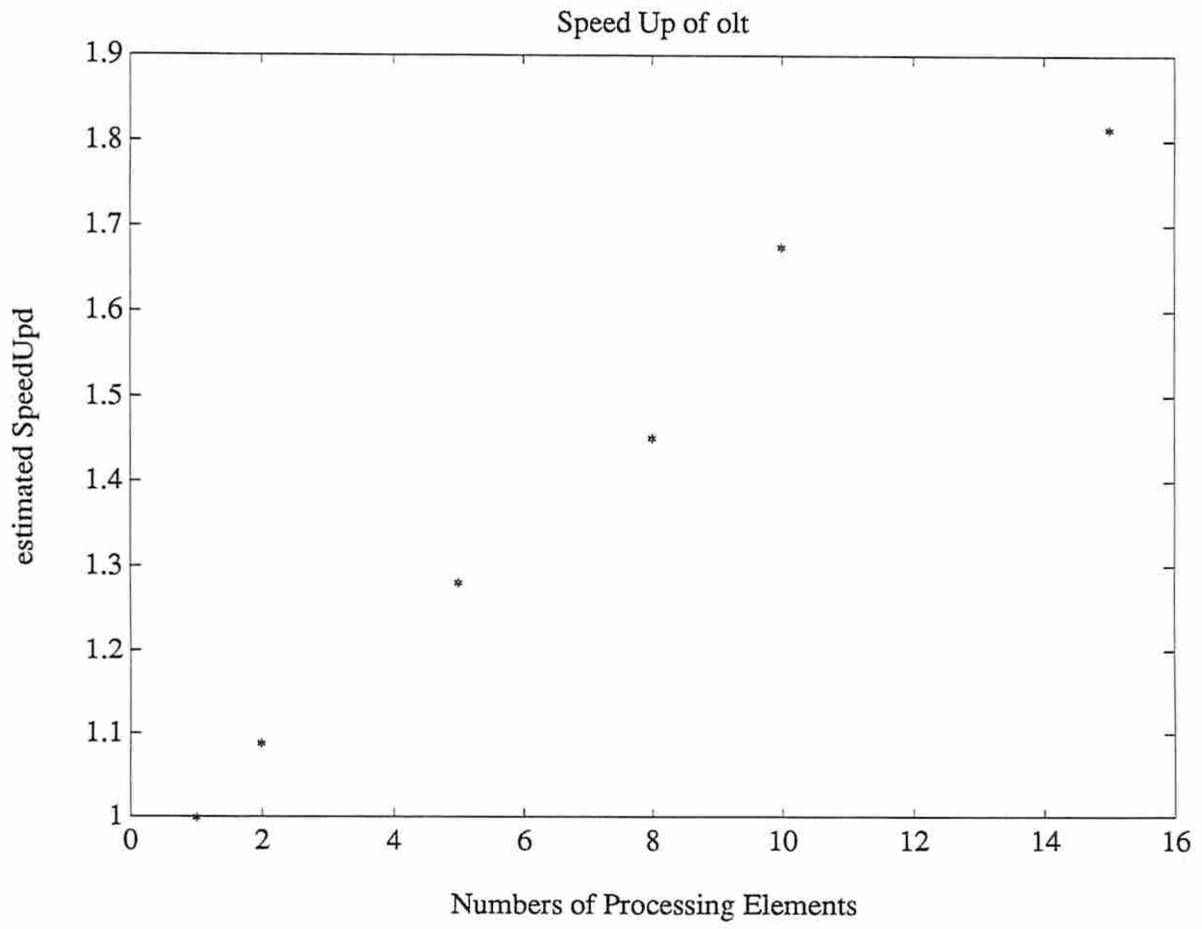
[7] J. Gettys, R.W. Sceiffer and R. Newmann
"X lib - C Language X Interface"
X window System

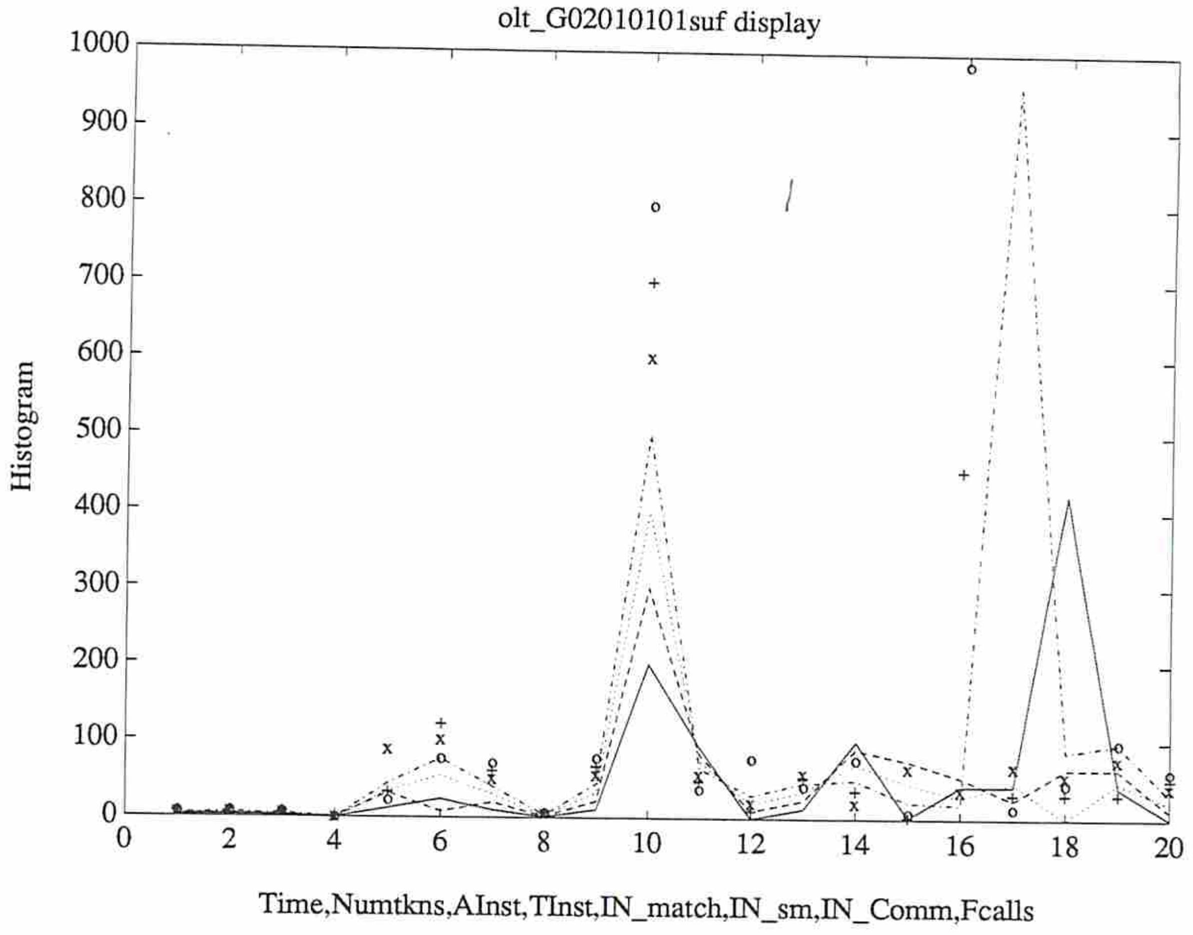
[8] R.W. Scheiffer
"X Window System Protocol"

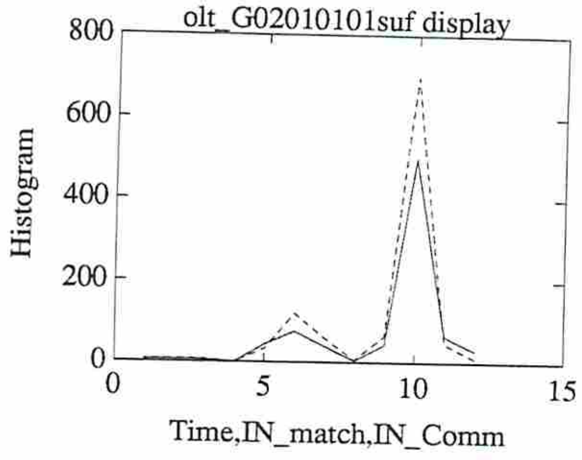
[9] C. Moler J. Little and S. Bangert
" PRO-MATLAB User's Guide for Sun Workstations"

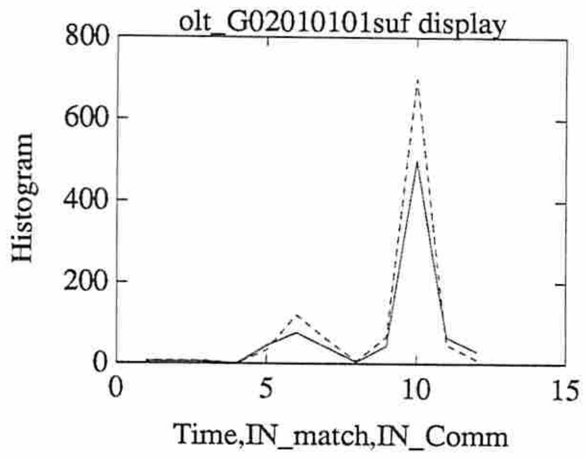
[10] Namhoon Yoo and Jean-Luc Gaudiot
"Macro Data-Flow Simulator"

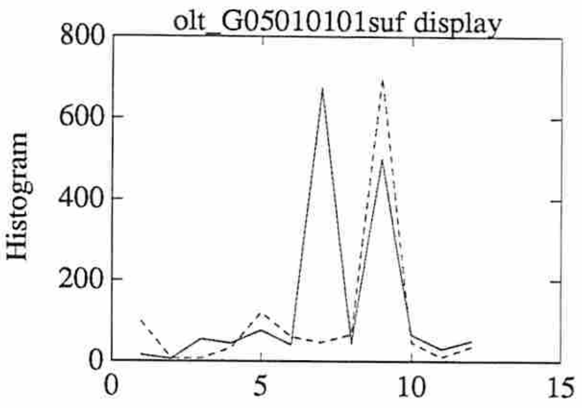

olt_G02010101suf display
SENSITIVITY
Time,TInst,IN_match,IN_Comm
olt_G05010101suf display
SENSITIVITY
olt_G10010101suf display
SENSITIVITY
olt_G15010101suf display
SENSITIVITY

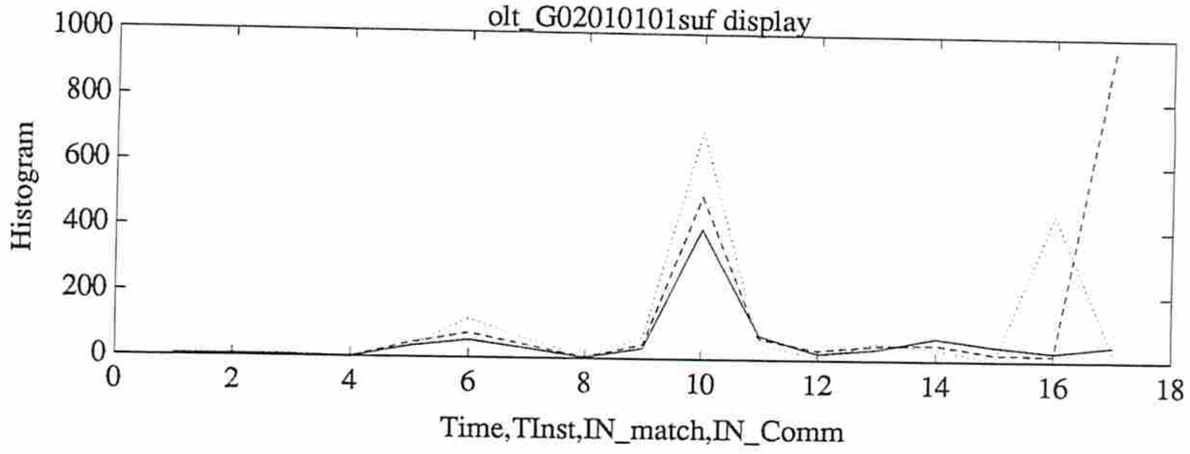

Speed Up of olt
estimated SpeedUpd
Numbers of Processing Elements

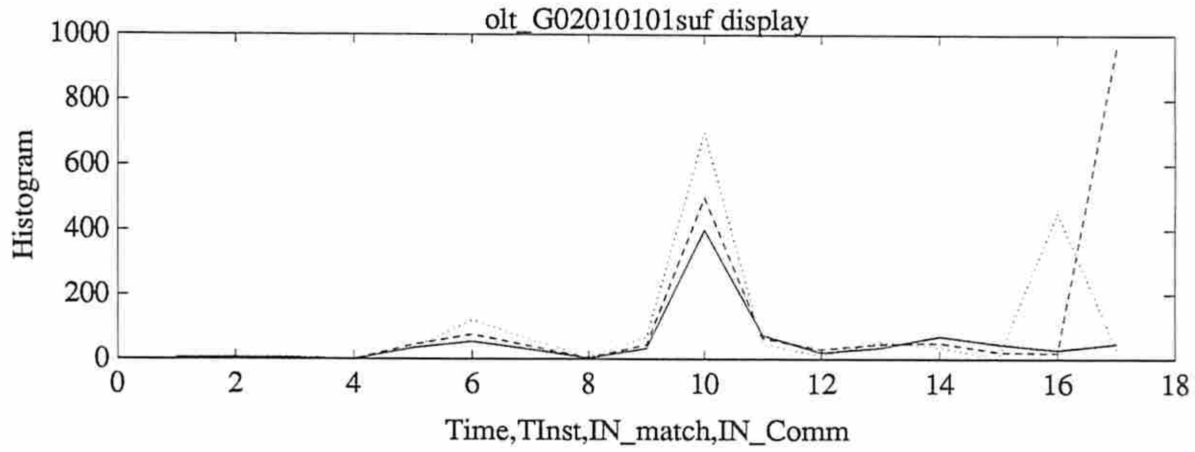

olt_G02010101suf display
Histogram
Time,Numtkns,AInst,TInst,IN_match,IN_sm,IN_Comm,Fcalls
0
100
200
300
400
500
600
700
800
900
1000
0
2
4
6
8
10
12
14
16
18
20

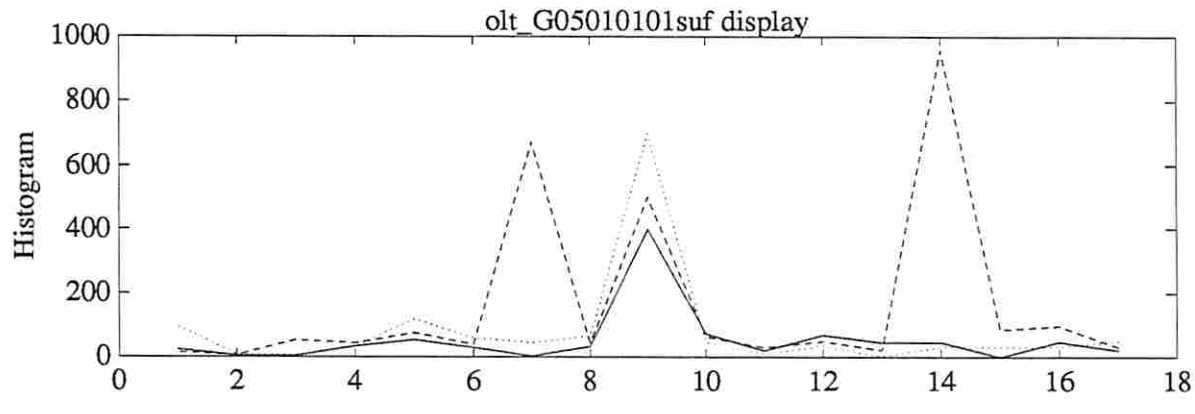

olt_G02010101suf display
Histogram
Time,IN_match,IN_Comm
800
600
400
200
0
0
5
10
15

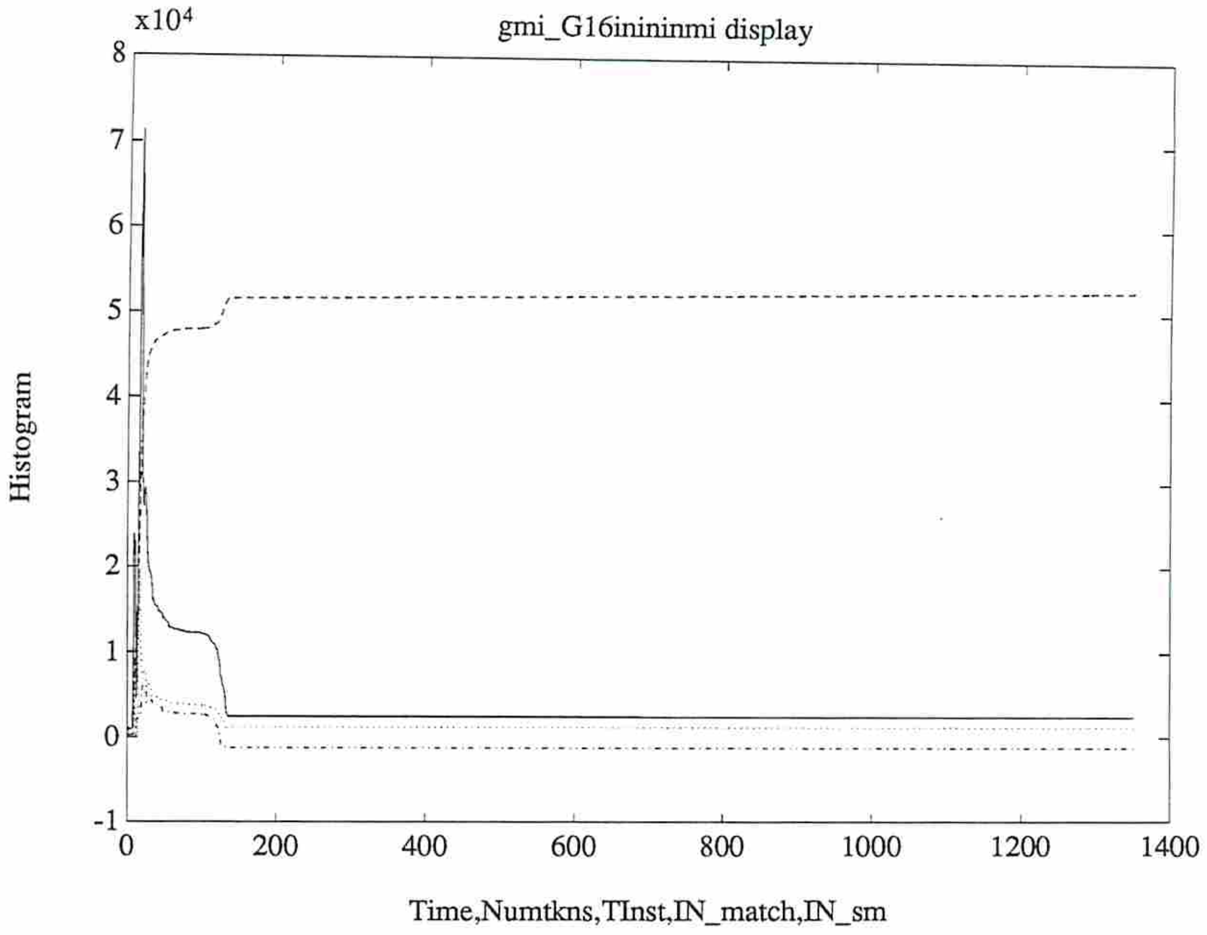

olt_G02010101suf display
Histogram
Time,IN_match,IN_Comm
800
600
400
200
0
0
5
10
15


olt_G05010101suf display
Histogram
800
600
400
200
0
0
5
10
15

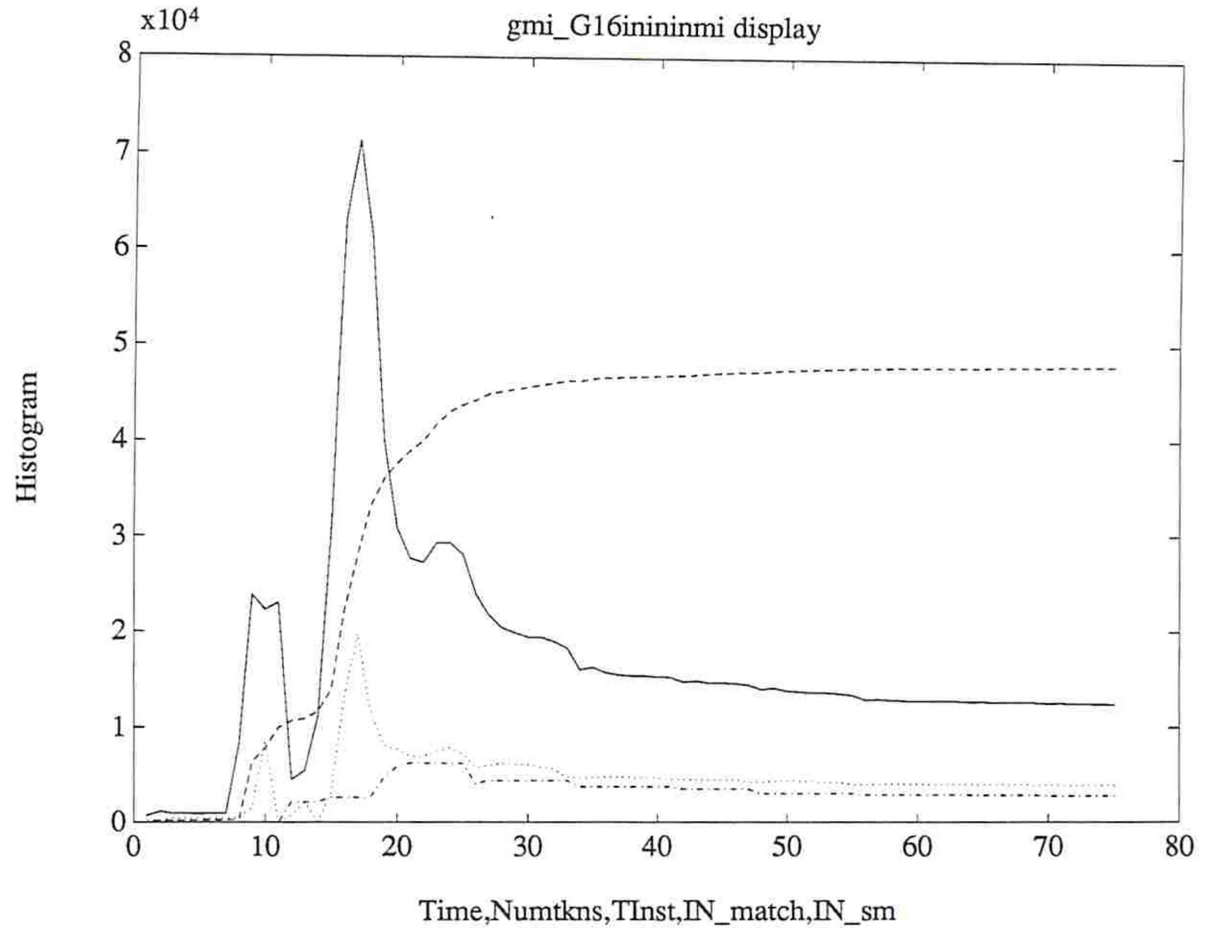

olt_G02010101suf display
Histogram
Time,TInst,IN_match,IN_Comm
0
200
400
600
800
1000
2
4
6
8
10
12
14
16
18


olt_G02010101suf display
Histogram
Time,TInst,IN_match,IN_Comm
0
200
400
600
800
1000
2
4
6
8
10
12
14
16
18


olt_G05010101suf display
Histogram
0
200
400
600
800
1000
2
4
6
8
10
12
14
16
18



x10^4
gmi_G16inininmi display
Histogram
Time,Numtkns,TInst,IN_match,IN_sm
8
7
6
5
4
3
2
1
0
-1
0
200
400
600
800
1000
1200
1400


x10^4
gmi_G16inininmi display
Histogram
Time,Numtkns,TInst,IN_match,IN_sm
0
1
2
3
4
5
6
7
8
10
20
30
40
50
60
70
80

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}$$

MATLAB PROGRAMS

MATLAB PROGRAMS

MATLAB PROGRAMS

MATLAB PROGRAMS

MATLAB PROGRAMS

MATLAB PROGRAMS

MATLAB PROGRAMS

MATLAB PROGRAMS

MATLAB PROGRAMS

MATLAB PROGRAMS

MATLAB PROGRAMS

MATLAB PROGRAMS

MATLAB PROGRAMS

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}$$

```
% this program will execute all  requested commands from the Xmenu
% program and display the results on the matlab graphic window.
% the necessary informations are supposed to be stored in the
% saistab.m and matname2.m files with the format specified in the
% Xmenu program.
%
%
%
%
% DECLARATIONS
%
%although the matlab language doesn't need the declaration
% of a variable before its usage, such declarations are
%provided here for maintenability convenience
%
%
%
%sub routines included
%
% #include "saisinit.m" (data file)
% #include "saistab.m" (data file)
% #include "matname2.m" (data file)
% #include "loadname.m" (used in lstatmat and savstatmat)
% #include "muldisp.m"
%
%
%functions used that do not appear in the standart libraries
%
% function compln
% function select
% function lstatmat
% function supred
% function traita
% function efr
% function efw
% function savstamat
% function selecdata
%
%
%variables declarations
%
% array of strings eflo
% array of strings matnames
% array of strings matames
% array of strings v2
%
% string taking 'oui' or 'non' values
%
%  string endsession
%  string sup
%  string admis1
%  string simpdis
%  string stat0
%  string comstat
%  string comdis
%  string speed
%  string sens
%
% string used as input of the eval function
%
```

```
%  string o1
%  string o2
%  string o3
%  string o10
%
% other strings
%
%  string dn
%  string eflodref
%  string dns
%  string e1
%  string e2
%  string s1
%  string s2
%  string ea
%  string plo
%
%
% integers
%
%  integer strln
%  integer lm
%  integer i
%  integer k
%  integer e4
%  integer rank
%  integer j
%  integer nPE
%
%
% reals
%
%  real SimT
%  real v3
%
% Vectors (one_dim array)
%
%  vector of integer nam
%  vector of integer slash
%  vector of integer St
%  vector of integer PE
%  vector of integer CE
%  vector of integer SM
%  vector of integer Ma
%  vector of integer suf
%  vector of integer selcol
%
%  vector of real delta
%  vector of real eflot
%  vector of real eflotref
%  vector of real e3
%  vector of real eb
%  vector of real vecPE
%  vector of real vecSimT
%
%
% Arrays
%
%  array of reals v1
%
```

```
%


%
%
%
%
%
%
%
%          DISPL PROGRAM
%
%
%
%



% clear workspace
clear;



% INITIALIZATIONS

saisinit;
%(this is a initialization version of saistab.m)
% the following is loaded in the workspace
%
%       endsession
%       matnames
%       sup
%       admis1
%       delta
%       simpdis
%       stat0
%       comstat
%       comdis
%       speed
%       sens
%       selcol
% saisinit is only intended to be loaded at the begining of a session



strln=16;
eflo=[compln('reflo',strln),',1lor,slor'];
% eflo keeps track of the state of the simulations files
% whose contents are managed by this program
% eflo has the following format
%  [refloXXXXXXXXXXX 1lor slor]
%  [name1XXXXXXXXXXX str1 str2]
%  [................ .... ....]
%  [................ .... ....]
%  [namenXXXXXXXXXXX str1 str2]
%
%  str1=l1oy means that the simulation file whose
%  name appear in first raw has been loaded in the worspace
%  that is 'namei'm (simulation file results matrice) is present
%  in the worspace and nameis(simulation file statistics) too
% str2=s1on means that the matrice contained in the simulation
% file has not been used as input to the selecdata function
% that is 'namei'Sm and 'namei'Ss don't exist yet
% 'namei'Sm is the result of the restriction of 'namei'm to the 'delta'
% time specification (which is stored in eflot) and the selcol
% specification (which is stored in string form in 'namei'Ss)
%
eflot=[0 0 0];
%eflot : this array has the same dimension as eflo and it contains
% the delta specifications that have been used to create 'namei'Sm
%

% matnames lines format initialization
%
%
% matnames has the following format
%       matnames=[matname1;..;matnamep];
%        where matnamesi=nam / S PE CE SM Ma suf
%                nam=sourcefile_name
%                S=St=simulation type (DataFlow simulator or Graph simulator)
%                PE=number of Processing Elements servers
%                CE=number of Communication Elements servers
%                SM=Structure Memory servers
%                Ma=number of match
%                suf=suffix
%
nam=[1:3]
slash=4;
St=5;
PE=[6:7]
CE=[7:9]
SM=[10:11]
Ma=[12:13]
suf=[13:16]



%
%
%    START MAINLOOP
%
%
while (exist('saistab.m')==0)
        disp('pas commence');
end;

while (endsession=='non')
      %if endsession=oui end of DISPL




 %store the modifications done in Xmenu.
 %the new values of the following data are stored:
 %      sup
 %      admis
 %      matnames
 %      data
 %      delta
 %      simpdis
 %      comdis
%       stat
%       comstat
%       speed
%       sens
while (exist('saistab.m')==0)
        disp('olt');
end;
%keyboard;
!sleep 1
disp('abon')
saistab;
disp(delta)
if (exist('matname2.m')~=0)
        matname2;
        !rm matname2.m
end;
!rm saistab.m
signal='non';
disp(matnames);

%store the value of matnames in case the matnames used in Xmenu
% is to be overwritten (this is the case if the display of
% SpeeUp or Sensitivity is intended
%this new value of matnames is only taken in account


%create and initialize matames
%matames is the same as matnames except that is slash column
%is replaced by an underscore column
%
matames=matnames;
lm=length(matnames(:,1));
%          lm=number of matname records in matnames
for i=1:lm
        matames(i,slash)='_';
end;


%select the good value of dn depending of the simulator used
%    'G' : Graph Simulator
%    'S' : DataFlow Simulator
if matnames(1,St)=='G'
        dn='Time,Numtkns,AInst,TInst,IN_match,IN_sm,IN_Comm,Fcalls';
else
        dn='Time,NTinSys,Fcall';
end;




%
%processing of sup & admis1 commands
%
%
%  (the reason why sup and admis are processed first is
%   that they must process a simulation_file results matrice
%   before the simulation results are stored in a '.mat' file)
if ((sup=='oui')|(admis1=='oui'))
        %
        %v1 <- simulation_file results matrice
        %v2 <- simulation statistics
        [v1,v2]=lstatmat(matnames(1,:));
                %the first line element of matnames is selected
                %the sup and admis1 command apply only on this element

        %processing of sup and admis1 commands
        %   the goal of sup is to suppress redondant lines
        %   in v1 (such lines may appear in the simulation results).
        %   the goal of admis1 is to add missing lines to the results
        %   matrice.
        %   sup  must be processed before admis1
        %
        if sup=='oui'
                v1=supred(v1);
                %cf supred.m file fore more details
        end;
        %
        if admis1=='oui'
                v1=traita(v1);
                %cf traita.m file fore more details
        end;

        %store the resulting natrice (v1) in the workspace
        %under the 'matname'm name
        %and store the coresponding simulation statistics
        %in the workspace under the 'matname's name
        o1=[matames(1,:),'m=v1;'];
        o2=[matames(1,:),'s=v2;'];
        eval(o1);
        eval(o2);

        %update eflo and eflot arrays and store the simulation_file
        %in '.mat' form which can be then retrieved in the user
        %directories far more efficiently than the ASCII flat
        %simulation_file.m
        %
        s1=matames(1,:);
        %       s1 <- first element of matnames
        %
        %   check eflo and eflot for the position of the matname
        %   record (the result expected is 0 : non existent. However
        %   if the user has selected by error a sup or admis1 command
        %   on an already stored simulation file, 'matname'm is already
        %   recorded in the eflo table and shall be updated at the place
        %   of the record
        [e1,e2,e3,e4]=efr(eflo,eflot,s1);
        %
        %  fills the postion line of eflo with ea and the
        %  'postion' line of eflot with eb. 'matname'Sm and 'matname'Ss
        %  are considered inexistant ('slon')
        ea=[compln(s1,strln),',l1oy',',slon'];
        eb=[1 1 1];
        [eflo,eflot]=efw(eflo,eflot,ea,eb,e4);
        %
        %  store the processed simulation file in the appropriate directory
        savstatmat(matnames(1,:),v1,v2);
end;


%
```

```
%
%loading of the simulation_files selected in Xmenu
%(for a given simulation_file, if the result matrice and the statistics
% are already in the workspace they are not loaded again)
%
%
for i=1:lm
    %all the requested simulations files are loaded
    %
    %check the position of the simulation_file number i
    % in the eflo array
    s1=matames(i,:);
    [e1,e2,e3,e4]=efr(eflo,eflot,s1);
    %
    %if e4=0 the simulation matrice and statistics are
    %   not in the workspace
    if e4==0
       %v1<-simulation_file number i matrice
       %v2<-simulation_file number i results
       [v1,v2]=lstatmat(matnames(i,:));
       %
       %v1 is stored in 'matnamei'm
       %v2 is stored in 'matnamei's
       o2=[matames(i,:),'m=v1;'];
       o3=[matames(i,:),'s=v2;'];
       eval(o2);
       eval(o3);
       %
       %updates eflo and eflot
       ea=[compln(s1,strln),',l1oy',',slon'];
       eb=[1 1 1];
       [eflo,eflot]=efw(eflo,eflot,ea,eb,e4);
    end;
end;

%
%
%
% simulations statistics
%
%
%
if ((statc=='oui')|(comstat=='oui'))
       clc;
       disp('statistics');
       disp(' ');
       disp(' ');
       disp('simulation time')
       for i=1:lm
               disp(' ');
               disp(matnames(i,:));
               disp('value: ');
               o1=['v3=',matames(i,:),'s;'];
               eval(o1);
               disp(v3);
       end;
end;


%
```

```
%
%
% trace of the simulations
%
%
%
if ((simpdis=='oui')|(comdis=='oui'))
   %
   %if simpdis==oui give only the first simulation trace
   if simpdis=='oui'
       k=1;
   end;
   %
   %
   %check to see that all the data needed  for
   % simulations trace display is at hand, that is
   % for each simulation required that the coresponding
   % 'matnamei'Ms and 'matamei'Ss variables are present
   % and are assigned with proper values
   %
   %('matnamei'Ms contains the simulation_file matrice with
   %  columns coresponding to the value of selcol at the moment
   %   when 'matnamei'Ms is initialized or reassigned,
   %  'matnamei'Ss contains 'dn' with columns coresponding
   %  to the value of selcol at the moment when 'matnmaei'SS
   %  is initialized or reassigned )
   %
   %
   k=lm;
   for i=1:k
       %
       %check position and status of matnamei in
       % the eflo array
       s2=matames(i,:);
       [e1,e2,e3,e4]=efr(eflo,eflot,s2);
       %modif derniere minute keyboard;
       %
       j=0;
       %
       % if e2=='sloy' 'matnamei'Ms and 'matnamei'Ss exists
       % already, but we have to check that they are
       % assigned with the proper values
       % if this is the case j==0, if it is not j<-1
       if all(e2=='sloy')
               %
               %dns<-'matnamei'Ss
               o3=['dns=',matames(i,:),'Ss;'];
               eval(o3);
               %
               % if 'matnamei'Ss<>dn or
               % old delta <> current delta give the order to
               % reload 'matnamei'Ms and 'matnamei'Ss (j<-1)
               if any(delta~=e3)
                       j=1;
               end;
               if (lengths(dns))~=(lengths(dn))
                       j=1;
               else
                       if any(dns~=dn)
                               j=1;
```

```
                        end;
                end;
        end;
        %
        %
        %load 'matnamei'Ms and 'matnamei'Ss if necessary
        %  (if  e2=='slon' such variables do not exist yet)
        if (all(e2=='slon')|(j==1))
           %
           disp('on est entre')
           % v1<-'matnamei'm
           plo=matnames(i,:);
           [v1,v2]=lstatmat(plo);
           disp(v1);
           disp(v2);
           %
           % dns <- dn coresponding to dn
           % v1 <- v1 with columns coresponding to selcol
           selcol=data;
           [dns,v1]=selecdata(dn,selcol,delta,v1);
           %
           %load 'matnamei'Ms and 'matnamei'Ss
           o1=[matames(i,:),'Ms=v1;'];
           o2=[matames(i,:),'Ss=dns;'];
           % (M=mat S=stat s=select)
           eval(o1);
           eval(o2);
           %
           %updates eflo and eflot
           plo=matames(i,:);
           ea=[compln(plo,strln),',l1oy',',sloy'];
           eb=delta;
           [eflo,eflot]=efw(eflo,eflot,ea,eb,e4);
         end;
    end;
    %
    %
    % display the requested simulations trace on MATLAB
    % graphics window
    muldisp;
    clear i;
end;




%
%
%
%   processing of Speed UP function
%
%
%
if speed=='oui'
        %
        %vecPE is a vector that will contain all the
        % #PE of the requested simulations_files
        %simT will contain all the coresponding
        % simulation times
        vecPE=[0];
```

```
vecSimT=[0];
%
% vecPE and vecSimt receive #PE and Simulation_Time for each
% simulation_file and are sorted like shown in the
% following example
%
%  simfil: simfil1 simfil2 simfil3 simfil4
%  #PE   :      64       1      16       4
%  SimuT :      30     500      50     100
%
%         (warning: these numbers come from the imagination of
%          the autor and any matching with existing simulation
%           results would be a pure coincidence)
%
% would result in
%
% vecPE  =[   1   4  16  64]
% vecSimT=[ 500 100  50  30]
for i=1:lm
        %
        %nPE<-#PE of simulation matnamei
        if (matnames(i,PE)=='in')
                nPE=9999;
        else
                nPE=eval(matnames(i,PE));
        end;
        %
        %load simT coresponding to matnamei. the information
        % is loaded from 'matnmanei's and for the moment
        % we suppose that this is the only data available
        % from 'matnamei's ie: the statistics of the simulation
        % file include only the simulation time
        o1=['SimT=eval(',matames(i,:),'s);'];
        eval(o1);
        %
        %rank is used to find the exact position to insert nPE in vecPE
        rank=0;
        for j=1:length(vecPE)
                if vecPE(j)<nPE
                        rank=rank+1;
                else
                        ;
                end;
        end;
        %
        % insert nPE and SimT in vecPE and vecSimT in the
        % rank position
        vecPE=[vecPE(1:rank);nPE;vecPE(rank+1:length(vecPE))];
                %if length(vecPE)=rank
                %vecPE(rank+1,ln(vecPE))=[]
        vecSimT=[vecSimT(1:rank);SimT;vecSimT(rank+1:length(vecSimT))];
                %idem
end;
%
%vecPE and vecSimT are now two vectors containing respectively
%the numbers of PE of the different simulations and the coresponding
%simulation execution Times
vecPE=vecPE(2:length(vecPE));
vecSimT=vecSimT(2:length(vecSimT));
if vecPE(1)~= 1
```

```
                %means that the simulation time for 1 single processor
                %is not recorded in vecPE, we will try to extrapolate it
                % that is we will compute sim_extra using a spline
                % extrapolation
                sim_extra=spline(vecPE,vecSimT,1);
                vecPE=[1;vecPE];
                vecSimT=[sim_extra;vecSimT];
        end;
        %We now compute the speedUp and store the result in vecSimT
        vecsim1=vecSimT(1);
        for i=1:(length(vecSimT))
                vecSimT(i)=vecsim1/vecSimT(i);
        end;
        %We now display the results in the Matlab graphics window
        %
        clg;
        % display Speed Up graph
        plot(vecPE,vecSimT,'*');
                %each plot of the speed up graph is a star
        %
        % display data relative to graph
        o1=['Speed Up of ',matames(1,nam)];
        title(o1);
        xlabel('Numbers of Processing Elements');
        ylabel('estimated SpeedUpd');
 end;
 %modif derniere mn keyboard
end;
```

```
%
%
%                MULDISP.M
%
%       ********************************
%       *                              *
%       *                              *
%       *   copyright Olivier TARDIEU  *
%       *                              *
%       *         ( 1990 )             *
%       *                              *
%       *                              *
%       ********************************
%
% program mudisp(matnames)
% the call of this program assumes
% that the matrices and stats corresponding
% to matnames have already been stored in
% 'matnamei'Ms & 'matnamei'Ss
% it is also supposed that there aren't more than
% 4 matrices to display (one per window)
%
%




%
% DECLARATIONS
%



%
% #include displ.m
%
% functions used that do not appear in the standart libraries
%
%   function lengths
%
% vector of characters coloration
% vector of characters poinlin
% vector of characters poinlin2
%
%
% strings
%
%   string smi
%   string yl
%   string tl
%
%  strings used as input of eval function
%
%   string tt
%   string dd
%
% integer l1
% integer m
% integer n
% integer sp
% integer lsyl
%
```

```
% vector of real T
%
% array of real D
%



%
%
%
%      'MAIN' of DISPL.M
%
%
%



%
% if there are more than 4 simulations traces to
% display , display only the first 4 (for the moment)
l1=k;
if (l1>4)
        l1=4;
end;
%


% create table of the available colors
coloration=['r';'g';'b';'w'];
% create table of the available curves display options
% for discrete plots display: + * o or x
% for continuous curves - -- : or -.
poinlin=['+';'*';'o';'x';'-';'-';':';'-'];
poinlin2=['*';'*';'*';'*';'*';'-';'*';'.'];
%



% clear graphing window
clg;
% show graphing window
shg;



% store data necessary to split graphing window
% in l1 graphing subwindows
if ((l1==3)|(l1==4))
   m=2;
   n=2;
else
   m=l1;
   n=1;
end;
% m,n are the parameters used to specify
% the requested number of display windows
% (m*n windows)
%
```

```
for i=1:l1
    %
    %access window i for display
    sp=(100*m+10*n+i);
    subplot(sp);
    %
    % load 'matnamei'Ss in yl
    smi=matames(i,:);
    o10=['yl=',smi,'Ss;'];
    eval(o10);
    %
    %load T and D vectors which are used
    % to display the trace of simulation matnamei
    lsyl=lengths(yl);
    tt=['T=',smi,'Ms(:,1);'];
    dd=['D=',smi,'Ms(:,2:lsyl);'];
    eval(tt);
    eval(dd);
    %
    %specify the graphing display mode for the
    % trace of the simulation
    o2='';
    % specify the graphing mode for
    % each curve to be displayed (lsyl-1 curves)
    for k=2:lsyl
        o1='';
        if (poinlin2(k-1)=='*')
           o1=['''',poinlin(k-1)];
        else
           o1=['''',poinlin(k-1),poinlin2(k-1)];
        end;
        o1=[o1,coloration(i),''''];
        o2=[o2,',T,D(:,',num2str(k-1),'),',o1];
    end;
    o2(1)='';
    %
    % display trace of simulation matnamei
    o3=['plot(',o2,');'];
    eval(o3);
    %
    % write comments on graphing screen related
    % to the displayed curves
    tl=[smi,' display'];
    title(tl);
    if i==1
       xlabel(yl);
    end;
    if sens=='oui'
        ylabel('SENSITIVITY');
    else
        ylabel('Histogram');
    end;
end;
```

```
%
%
%
%
%
%               LOADNAME.M
%
%
%
%
% the purpose of this program is to load the file 'name'
% ('name' is stored in the matname variable ) in the matlab
%  workspace.
% matname='name'
% name.matrice -> matr & name.stats -> stat
%
% the program is first checking if the file has already
% been saved by matlab in the appropriate refload.mat file
% if this is the case, the program operates a 'load name.mat' command
% if it is not, the program operates a 'name' command then
% saves the file 'name' and indicates in a table that name is now
% available under the 'name.mat' reference. the 'name.m' file
% is not deleted
%
%
%       DECLARATIONS
%
% #include displ.m
% #include lstatmat.m
%
% string o8
% string o9
%
% integer flagldna
% integer iiio
%
%


%
% load matname in workspace
load tralala
%
% load refload related to matname(nam) in workspace
% in the refload array
o10=['!cp ',matname(nam),'/refload.mat refload.mat;'];
eval(o10);
load refload
% refload is now an array that contains the names of the already
% existing 'names.mat'

flagldna=0;
o9='';
o8='';
%
% check in refload to see if the simulation file matname
% exists already in .mat form.
% if this is the case flagldna <-1 else flagldna <- 0
o8=compln(matname,strln);
for iiio=1:length(refload(:,1))
```

```
    if refload(iiio,:)==o8
       flagldna=1;
    end;
end;

%
%if the .mat form does not exist yet
if flagldna==0
   %
   % load matr and stat related to matname in the workspace.
   %
   %    because only flat files present in the current directory
   %    can be loaded easily,matname.m is copied in the current
   %    directory.
   o10=['!cp ',matname,'.m olive.m;'];
   eval(o10);
   eval('olive');
   o10=['!rm olive.m'];
   eval(o10);
   %
   % matr and stat are saved in the matnameXX.mat file
   o9=['save ',compln(matname,strln),' matr stat'];
   eval(o9);
   %
   % matr and stat are stored in matrstat to be retrieved
   % by the lstatmat.m function
   save matrstat matr stat;
   %
   % refload is updated and saved under the refload.mat name
   % in the directory nam and refload present in the
   % current directory is removed
   o9=compln(matname,strln);
   refload=[refload;o9];
   o10=['save ',matname(nam),'/refload refload;'];
   eval(o10);
   !rm refload.mat;

else

   % the .mat form exists
   %
   % load matr and stat related to matname
   o9=['load ',compln(matname,strln)];
   eval(o9);
   %store matr and stat for lstatmat
   save matrstat matr stat;
   % refload does not need to be updated and is removed
   % from the current directory
   o10=['!mv refload.mat ',matname(nam),'/refload.mat;'];
   eval(o10);
end;

% clear workspace variables declared in this file
clear refload;
clear iiio;
clear matr stat;
% the lines of the refload array are composed of strln characters
clear o9 flagldna o8;
```

```
endsession='non';
matnames='';
sup='non';
admis1='non';
delta=[9 9 9];
simpdis='non';
stat='non';
comstat='non';
comdis='non';
speed='oui';
sens='non';
selcol=[0 0 0 0 0 0 0 0];
signal='non';
```

```
%
%
%
%                LSTATMAT.M
%
%
%
%



function [matri,stati]=lstatmat(matname)
%function [matri,stati]=lstatmat(matname)
%this function uses the loadname program
%matri and stati are the matrice and statistics
%of the matname file
%
%
%save matname in tralala (this variable is read then
%  by loadname)
save tralala matname;
loadname;
% load matr and stat in the function workspace
load matrstat;
matri=matr;
stati=stat;
```

```
%
%
%
%
%                 SAVSTAMAT.M
%
%
%


function savstatmat(nommat,mat,sta)
%function savstamat(nommat,mat,sta)
%this function saves sta and mat in the 'nommatXXXXX.mat'
%file
stat=sta;
matr=mat;
olt=['save ',compln(nommat,16),' matr stat'];
eval(olt);
```

```
%
%
%                EFR.M
%
%


function [e1,e2,e3,e4]=efr(elf,elft,name)
%function [e1,e2,e3,e4]=ef(elf,eflt,name)
%elf<->eflo
%eflt<->eflot
%status:
% r if compln('name',16) exists in elf(1,:)
%   then ef returns the corresponding following informations
%   e1='1loy':y
%   e2='sloy':y/n (depends on what was recorded)
%   e3 previous time specifications (idem)
%   e4=position in elf
%   else e4=0
%        e1='1lor'
%        e2='slor'
%        e3=[0 0 0]

%
% uses function compln.m
%

e4=0;
r=0;
name=compln(name,16);
for i=1:length(elf(:,1))
    r=r+1;
    if name==syntaxe(elf(i,:),1)
       % the name exists in elf
       % r is his position
       e1=syntaxe(elf(i,:),2);
       e2=syntaxe(elf(i,:),3);
       e3=elft(i,:);
       e4=r;
    end;
end;
%
% if name does not exist
if r==0
   e1='1lor';
   e2='slor';
   e3=[0 0 0];
end;
```

```
%
%
%                 EFW.M
%
%



function [e1,e2]=efw(elf,elft,ea,eb,pos)
%function [e1,e2]=efw(elf,elft,ea,eb,pos)
% replace the pos line of elf & elft by the corresponding
% datas ea & eb
% if pos=0 ea and eb are added at the end of elf & elft
%ea=compln(name),',1lor,','slor'
%eb=[n1 n2 n3]
if pos==0
   elf=[elf;ea];
   elft=[elft;eb];
   e1=elf;
   e2=elft;
else
   elf(pos,:)=ea;
   elft(pos,:)=eb;
   e1=elf;
   e2=elft;
end;
```

```
%
%
%                SELECDATA.M
%
%



function [dns,ass]=selecdata(dn,dnb,tsi,a)
%INPUT
% a: data to process
%tsi : time specifications 0,1000,10
%dn e.g. dn='name1,name2'
%dnb [1 0]
%the above example selects raw name1 of a
%OUTPUT
%dns = dn  after selection
%ass = a after selection


as=a(:,dnb);
% as= a with selected raws
dns=select(dn,dnb);
%dns: dn with selected raws

% calculus of 'a' time interval (which may be
% different from the interval specified in tsi)
ati=a(2,1)-a(1,1);

%normalisation of tsi to put it in a format (tsisn)
% that will allow to select the proper lines of as
tsisn=floor(tsi/ati);
%creates fe1 and fe2
%
%fe1=[1 0 0] fe2 =[0 1 0]
fe1=ones(tsisn);
fe2=ones(tsisn);
fes=zeros(tsisn);
fe1(:,3)=fes(:,3);
fe2(:,3)=fes(:,3);
fe1(:,2)=fes(:,2);
fe2(:,1)=fes(:,1);
% modify fe1 and fe2 to be sure that the first and
% end lines of as that we want to select are not forgotten
fe1=fe1*(floor(as(1)/ati)-1);
fe2=fe2*(floor(as(1)/ati)-1);
tsisn=tsisn-fe1-fe2;

% selects pertinent lines of as
ass=as(tsisn(:,1):tsisn(:,3):tsisn(:,2),:);
```

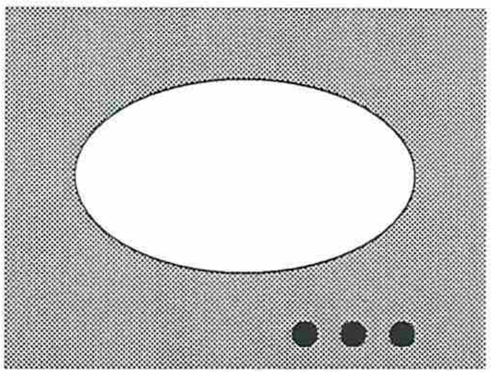
```
%
%
%
%                  SUPRED.M
%
%
%




function y=supred(x)
%
%function deletes the second of 2 successive
%lines with same first raw value
%
y=x(1,:);
for i=2:length(x(:,1))
    if (x(i,1)-x(i-1,1))~=0
       y=[y;x(i,:)];
    end
end
```

```
%
%
%                  TRAITA.M
%
%



function y=traita(a)
%a is a matrice whose first column are naturally sorted integers
%y returns a with all "missing" lines of a.  E.g. if
%a=[1 . . .
%   2 . . .
%   4 . . .]
% the function returns
%a=[1 . . .
%   2 . . .
%   3 . . .
%   4 . . .]
%
%( the rest of the line (the line without first column) of
% an added line is the same as the rest of the line
%  preceding it. E.g. if line 3 is missing and the preceding
% line is 2 f d s g then line 3 is 3 f d s g)
%
b=length(a(:,1));
c=a(2,1)-a(1,1);
y=a(1,:);
for p=2:b
    d=a(p,1)-a(p-1,1);
    e=d/c;
    if e==1
       y=[y;a(p,:)];
    else
       for h=1:e-1
           y=[y;a(p-1,:)];
           f=size(y);
           g=(f(1)-1)*c+a(1,1);
           y(f(1),1)=g;
       end
       y=[y;a(p,:)];
    end
end
```

```
function y=select(a,v)
% s is of the format s='w1,w2,..wn'
%v =[ d1 d2 .. dn] where di=[0 or 1]
%y returns all the words of s which correspond to a 1 of v
i=1;
while i~=length(v)
      if v(i)==1
         y=[y,syntaxe(a,i),','];
      end
      i=i+1;
end
if v(i)==1
   y=[y,syntaxe(a,i)];
end
if y(length(y))==',';
   y(length(y))='';
end
```

```
function y=compln(s1,n)
% function y=compln(s1,n)
% this function completes a string s with less than n elements
% (length m) with n-m X
m=length(s1);
y=s1;
for i=1:(n-m)
    y=[y,'X'];
end;
```

X



DISPLAY

```
/*
        **********************************
        *                                *
        *                                *
        *         XDISPLAY               *
        *                                *
        *  copyright Olivier TARDIEU     *
        *                                *
        *        (1990)                  *
        *                                *
        *                                *
        **********************************
*/




#include <stdio.h>
#include "X11/Xlib.h"
#include "X11/Intrinsic.h"
#include "X11/StringDefs.h"
#include "X11/Xaw/Label.h"
#include "X11/Xaw/Form.h"
#include "X11/Xaw/Box.h"
#include "X11/Xaw/Command.h"
#include "X11/Xaw/Dialog.h"


FILE *fpMatlab,*fopen();

/*
 *variables used to keep track of menu content"
 */
int bcom1,bcom2,bcom3,bcom4;
int bmat1_1=1,bmat1_2=1,bmat1_3=1 ;
int bdat1=1,bdat2=1,bdat3=1,bdat4=1;
int bdat5=1,bdat6=1,bdat7=1,bdat8=1;
int bdisop1=1,bdisop2=1,bdisop3=1;
int bdisop4=1,bdisop5=1,bdisop6=1;

char *matname1="",*matname2="";
char *matname3="",*matname4="";

char *delta1="0",*delta2="0",*delta3="0";

/*
 * application rootwindow
 */
Widget toplevel;
```

```
/*
 *Display_Interface form window with display_interface subwindows
 */
/*form widget*/
Widget form_menu;
/*boxwidgets*/
Widget wcom,wmat1,wmat2,wdat,wtim,wdisop;

/*
 *Data types selection
 */
/*command widgets*/
Widget wdat1,wdat2,wdat3,wdat4,wdat5,wdat6,wdat7,wdat8;

/*
 *Session commands
 */
/*command widgets*/
Widget wcom1,wcom2,wcom3,wcom4;


/*
 *simulation file transformations options
 */
/*commandwidgets*/
Widget wmat1_1,wmat1_2,wmat1_3;

/*
 *simulation file identification names
 */
/*dialogwidgets*/
Widget wmat2_1,wmat2_2,wmat2_3,wmat2_4;

/*
 *time specifications
 */
/*dialogwidgets*/
Widget wtim1,wtim2,wtim3;

/*
 *Display options
 */
/*command widgets*/
Widget wdisop1, wdisop2, wdisop3, wdisop4, wdisop5, wdisop6;

/*
 *CallBack_Lists for wcom subwidgets
 */

XtCallbackRec cbwcom1[]={
{NULL,NULL},
{NULL,NULL},
};

XtCallbackRec cbwcom2[]={
{NULL,NULL},
{NULL,NULL},
};

XtCallbackRec cbwcom3[]={
```

```
{NULL,NULL},
{NULL,NULL},
};

XtCallbackRec cbwcom4[]={
{NULL,NULL},
{NULL,NULL},
};


/*
 * CallBack_Lists  for wmat1 subwidgets
 */


XtCallbackRec cbwmat1_1[]={
{NULL,NULL},
{NULL,NULL},
};

XtCallbackRec cbwmat1_2[]={
{NULL,NULL},
{NULL,NULL},
};

XtCallbackRec cbwmat1_3[]={
{NULL,NULL},
{NULL,NULL},
};

/*
 * CallBack_Lists  for wdat
 */


XtCallbackRec cbwdat1[]={
{NULL,NULL},
{NULL,NULL},
};

XtCallbackRec cbwdat2[]={
{NULL,NULL},
{NULL,NULL},
};

XtCallbackRec cbwdat3[]={
{NULL,NULL},
{NULL,NULL},
};

XtCallbackRec cbwdat4[]={
{NULL,NULL},
{NULL,NULL},
};

XtCallbackRec cbwdat5[]={
{NULL,NULL},
{NULL,NULL},
};
```

```
XtCallbackRec cbwdat6[]={
{NULL,NULL},
{NULL,NULL},
};

XtCallbackRec cbwdat7[]={
{NULL,NULL},
{NULL,NULL},
};

XtCallbackRec cbwdat8[]={
{NULL,NULL},
{NULL,NULL},
};

/*
 * CallBack_Lists  for wdisop
 */


XtCallbackRec cbwdisop1[]={
{NULL,NULL},
{NULL,NULL},
};

XtCallbackRec cbwdisop2[]={
{NULL,NULL},
{NULL,NULL},
};

XtCallbackRec cbwdisop3[]={
{NULL,NULL},
{NULL,NULL},
};

XtCallbackRec cbwdisop4[]={
{NULL,NULL},
{NULL,NULL},
};

XtCallbackRec cbwdisop5[]={
{NULL,NULL},
{NULL,NULL},
};

XtCallbackRec cbwdisop6[]={
{NULL,NULL},
{NULL,NULL},
};


/*
 *form_menu Widget argument list
 */

Arg argform[]={
{XtNheight,(XtArgVal) 60},
{XtNwidth ,(XtArgVal) 50},
};
```

```
/*
 *display_interface subwidgets argument lists
 */

/*
 *arguments to position wcom on left-top
 *corner of form
 */
Arg argwcom[]={
{XtNleft,(XtArgVal) XtChainLeft},
{XtNtop,(XtArgVal) XtChainTop},
};

/*
 * give arguments to chain wmat1 on left side
 * of form and specify that wcom1 will be chained
 * (vertically) to another window whose ID can
 * not be specified yet because it does not exist.
 * (this window willbe wcom)
 */
Arg argwmat1[]={
{XtNfromVert,(XtArgVal) NULL},
{XtNleft,(XtArgVal) XtChainLeft},
{XtNvertDistance,(XtArgVal) 10},
};

/*
 * give arguments to chain wmat2 on left side
 * of form and specify that wcom1 will be chained
 * (vertically) to another window whose ID can
 * not be specified yet because it does not exist.
 * (this window will be wmat1)
 */
Arg argwmat2[]={
{XtNfromVert,(XtArgVal) NULL},
{XtNleft,(XtArgVal) XtChainLeft},
{XtNvertDistance,(XtArgVal) 10},
};

/*
 * give arguments to chain wdat on right-top corner
 * of form and specify that wdat will be chained
 * (horizontally) to another window whose ID can
 * not be specified yet because it does not exist.
 * (this window will be wcom)
 */
Arg argwdat[]={
{XtNright,(XtArgVal) XtChainRight},
{XtNtop,(XtArgVal) XtChainTop},
{XtNfromHoriz,(XtArgVal) NULL},
{XtNhorizDistance,(XtArgVal) 10},
};

/*
 * give arguments to chain wtim on right side
 * of form and specify that wtim will be chained
 * (vertically/horizontally) to another window whose ID can
 * not be specified yet because it does not exist.
 * (this window will be wdat/wmat1)
 */
```

```
Arg argwtim[]={
{XtNfromVert,(XtArgVal)  NULL},
{XtNfromHoriz,(XtArgVal) NULL},
{XtNright,(XtArgVal) XtChainRight},
{XtNvertDistance,(XtArgVal) 10},
{XtNhorizDistance,(XtArgVal) 10},
};

/*
 * give arguments to chain wdisop on right side
 * of form and specify that wdisop will be chained
 * (vertically/horizontally) to another window whose ID can
 * not be specified yet because it does not exist.
 * (this window will be wtim/wmat2)
 */
Arg argwdisop[]={
{XtNfromVert,(XtArgVal) NULL},
{XtNfromHoriz,(XtArgVal) NULL},
{XtNright,(XtArgVal) XtChainRight},
{XtNvertDistance,(XtArgVal) 10},
{XtNhorizDistance,(XtArgVal) 10},
};

/*
 *display_interface subwidgets 's children argument lists
 */

/*
 *COM children argument lists
 */

/*
 * give arguments to specify the label to be displayed
 * in wcom1 window at the start of Xdisplay and the procedure
 * that is called whenever a selection is made with the
 * mouse on this window
 */
Arg argwcom1[]={
{XtNlabel, (XtArgVal) "End_Window_Menu"},
{XtNcallback,(XtArgVal) cbwcom1},
};

/*
 * idem for wcom2
 */
Arg argwcom2[]={
{XtNlabel, (XtArgVal) "End_Display_Session"},
{XtNcallback,(XtArgVal) cbwcom2},
};

/*
 * idem for wcom3
 */
Arg argwcom3[]={
{XtNlabel, (XtArgVal) "Confirm_Selections"},
{XtNcallback,(XtArgVal) cbwcom3},
};

/*
 * idem for wcom4
 */
Arg argwcom4[]={
{XtNlabel, (XtArgVal) "Display_Simulations"},
{XtNcallback,(XtArgVal) cbwcom4},
};

/*
 *MAT1 children argument lists
 */

/*
 * give arguments to specify the label to be displayed
 * in wmat1_1 window at the start of Xdisplay and the procedure
 * that is called whenever a selection is made with the
 * mouse on this window
 */
Arg argwmat1_1[]={
{XtNlabel, (XtArgVal) "Graph_Simulator"},
{XtNcallback,(XtArgVal) cbwmat1_1},
};

/*
 * idem for wmat1_2
 */
Arg argwmat1_2[]={
{XtNlabel, (XtArgVal) "sup_same_lines_off"},
{XtNcallback,(XtArgVal) cbwmat1_2},
};

/*
 * idem for wmat1_3
 */
Arg argwmat1_3[]={
{XtNlabel,(XtArgVal) "add_miss_lines_off"},
{XtNcallback,(XtArgVal) cbwmat1_3},
};

/*
 *MAT2 children argument lists
 */

/*
 * give arguments to specify the label to be displayed
 * in wmat1_1 window at the start of Xdisplay and the
 * string to be shown on the editor part of the
 * dialog widget. (this string is actually specified
 * at run time).
 */
Arg argwmat2_1[]={
{XtNlabel,(XtArgVal) "simu_file_name_1"},
{XtNvalue,(XtArgVal) NULL},
};

/*
 * idem for wmat2_2
 */
Arg argwmat2_2[]={
{XtNlabel,(XtArgVal) "simu_file_name_2"},
{XtNvalue,(XtArgVal) NULL},
};
```

```
/*
 * idem for wmat2_3
 */
Arg argwmat2_3[]={
{XtNlabel,(XtArgVal) "simu_file_name_3"},
{XtNvalue,(XtArgVal) NULL},
};

/*
 * idem for wmat2_4
 */
Arg argwmat2_4[]={
{XtNlabel,(XtArgVal) "simu_file_name_4"},
{XtNvalue,(XtArgVal) NULL},
};

/*
 *DAT children argument lists
 */

/*
 * give arguments to specify the label to be displayed
 * in wdat1 window at the start of Xdisplay and the procedure
 * that is called whenever a selection is made with the
 * mouse on this window
 */
Arg argwdat1[]={
{XtNlabel, (XtArgVal) "Time_off"},
{XtNcallback,(XtArgVal) cbwdat1},
};

/*
 * idem for wdat2
 */
Arg argwdat2[]={
{XtNlabel, (XtArgVal) "Num_Tkns_off"},
{XtNcallback,(XtArgVal) cbwdat2},
};

/*
 * idem for wdat3
 */
Arg argwdat3[]={
{XtNlabel, (XtArgVal) "No_Active_Inst_off"},
{XtNcallback,(XtArgVal) cbwdat3},
};

/*
 * idem for wdat4
 */
Arg argwdat4[]={
{XtNlabel, (XtArgVal) "Total_Inst_off"},
{XtNcallback,(XtArgVal) cbwdat4},
};

/*
 * idem for wdat5
 */
Arg argwdat5[]={
```

```
{XtNlabel, (XtArgVal) "In_Match_off"},
{XtNcallback,(XtArgVal) cbwdat5},
};

/*
 * idem for wdat6
 */
Arg argwdat6[ ]={
{XtNlabel, (XtArgVal) "In_Sm_off"},
{XtNcallback,(XtArgVal) cbwdat6},
};

/*
 * idem for wdat7
 */
Arg argwdat7[ ]={
{XtNlabel, (XtArgVal) "In_transit_off"},
{XtNcallback,(XtArgVal) cbwdat7},
};

/*
 * idem for wdat8
 */
Arg argwdat8[ ]={
{XtNlabel, (XtArgVal) "F_Calls_off"},
{XtNcallback,(XtArgVal) cbwdat8},
};

/*
 *TIM children argument lists
 */

/*
 * give arguments to specify the label to be displayed
 * in wtim1 window at the start of Xdisplay and the procedure
 * that is called whenever a selection is made with the
 * mouse on this window
 */
Arg argwtim1[ ]={
{XtNlabel,(XtArgVal) "Starting_Time"},
{XtNvalue,(XtArgVal) NULL},
};

/*
 * idem for wtim2
 */
Arg argwtim2[ ]={
{XtNlabel,(XtArgVal) "Ending_Time"},
{XtNvalue,(XtArgVal) NULL},
};

/*
 * idem for wtim2
 */
Arg argwtim3[ ]={
{XtNlabel,(XtArgVal) "Time_Interval"},
{XtNvalue,(XtArgVal) NULL},
};

/*
```

```
 *DISOP children argument lists
 */

/*
 * give arguments to specify the label to be displayed
 * in wdisop1 window at the start of Xdisplay and the procedure
 * that is called whenever a selection is made with the
 * mouse on this window
 */
Arg argwdisop1[]={
{XtNlabel, (XtArgVal) "Single_Display_off"},
{XtNcallback,(XtArgVal) cbwdisop1},
};

/*
 * idem for wdisop2
 */
Arg argwdisop2[]={
{XtNlabel, (XtArgVal) "Compare_Display_off"},
{XtNcallback,(XtArgVal) cbwdisop2},
};

/*
 * idem for wdisop3
 */
Arg argwdisop3[]={
{XtNlabel, (XtArgVal) "Single_Stats_off"},
{XtNcallback,(XtArgVal) cbwdisop3},
};

/*
 * idem for wdisop4
 */
Arg argwdisop4[]={
{XtNlabel, (XtArgVal) "Compare_Stats_off"},
{XtNcallback,(XtArgVal) cbwdisop4},
};

/*
 * idem for wdisop5
 */
Arg argwdisop5[]={
{XtNlabel, (XtArgVal) "Speed_Up_off"},
{XtNcallback,(XtArgVal) cbwdisop5},
};

/*
 * idem for wdisop6
 */
Arg argwdisop6[]={
{XtNlabel, (XtArgVal) "Sensitivity_off"},
{XtNcallback,(XtArgVal) cbwdisop6},
};

/*
 *procedure used by store_data to
 * store data relative to MAT2
 */
void sto_mat2()
```

```
{
     fprintf(fpMatlab,"matnames=[\n");
     /*
      * stores all names whose
      * first character is not a star
      * in saistab.m
      */
     if (matname1[0]=='*')
          ;
     else {
          fprintf(fpMatlab,"'");
          fprintf(fpMatlab,"%s",matname1);
          fprintf(fpMatlab,"';\n");
     }
     if (matname2[0]=='*')
          ;
     else {
          fprintf(fpMatlab,"'");
          fprintf(fpMatlab,"%s",matname2);
          fprintf(fpMatlab,"';\n");
     }
     if (matname3[0]=='*')
          ;
     else {
          fprintf(fpMatlab,"'");
          fprintf(fpMatlab,"%s",matname3);
          fprintf(fpMatlab,"';\n");
     }
     if (matname4[0]=='*')
          ;
     else {
          fprintf(fpMatlab,"'");
          fprintf(fpMatlab,"%s",matname4);
          fprintf(fpMatlab,"';\n");
     }
     fprintf(fpMatlab,"]\n");
     /*
      *this results in the following
      * appended to saistab.m
      *
      * matnames=[
      * 'name1'
      * 'name2'
      * 'name3'
      * 'name4'
      * ];
      */
}

/*
 *procedure sto_dat used by store_data to store
 *information related to DAT
 *uses ouinon function
 */
void ouinon(pp)

  int pp;

{
     if (pp==1)
```

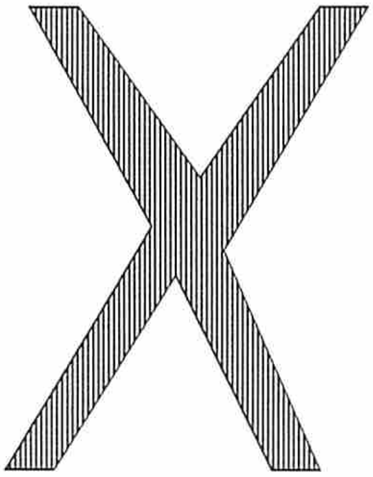
```
          fprintf(fpMatlab,"0 ");
     else
          fprintf(fpMatlab,"1 ");
}
/**/
void sto_dat()

{
     fprintf(fpMatlab,"data=[");
     ouinon(bdat1);
     ouinon(bdat2);
     ouinon(bdat3);
     ouinon(bdat4);
     ouinon(bdat5);
     ouinon(bdat6);
     ouinon(bdat7);
     ouinon(bdat8);
     fprintf(fpMatlab,"]\n");
     /*
      * this results in
      * data=[b1 . bi .. bn];
      * written in saistab.m where
      * bi=1 or 0 depending on
      * the contents of the wdati
      * widgets
      */
}

/*
 *this procedure stores all the useful information in
 *the saistab.m file which is then used by the
 *matlab displ.m program to retrieve all display information
 */
void store_data()

{

  fpMatlab=fopen("saistab.m","w");

  /*data related to MAT1*/
  if (bmat1_2==1)
       fprintf(fpMatlab,"sup='non'\n");
  else
       fprintf(fpMatlab,"sup='oui'\n");
  if (bmat1_3==1)
       fprintf(fpMatlab,"admis1='non'\n");
  else
       fprintf(fpMatlab,"admis1='oui'\n");

  /*data related to MAT2*/
  sto_mat2();

  /*data related to DAT*/
  sto_dat();

  /*data related to TIM*/
  fprintf(fpMatlab,"delta=[");
  fprintf(fpMatlab,"%s",delta1);
  fprintf(fpMatlab,"%c",' ');
  fprintf(fpMatlab,"%s",delta2);
```

```
   fprintf(fpMatlab,"%c",' ');
   fprintf(fpMatlab,"%s",delta3);
   fprintf(fpMatlab,"]\n");

   /*data related to DISOP*/
   if (bdisop1==1)
        fprintf(fpMatlab,"simpdis='non'\n");
   else
        fprintf(fpMatlab,"simpdis='oui'\n");
   if (bdisop2==1)
        fprintf(fpMatlab,"comdis='non'\n");
   else
        fprintf(fpMatlab,"comdis='oui'\n");
   if (bdisop3==1)
        fprintf(fpMatlab,"statc='non'\n");
   else
        fprintf(fpMatlab,"statc='oui'\n");
   if (bdisop4==1)
        fprintf(fpMatlab,"comstat='non'\n");
   else
        fprintf(fpMatlab,"comstat='oui'\n");
   if (bdisop5==1)
        fprintf(fpMatlab,"speed='non'\n");
   else
        fprintf(fpMatlab,"speed='oui'\n");
   if (bdisop6==1)
        fprintf(fpMatlab,"sens='non'\n");
   else
        fprintf(fpMatlab,"sens='oui'\n");
   /*signal*/
   fprintf(fpMatlab,"signal='oui'\n");
   fclose(fpMatlab);
}

/*
 *this procedure creates and manages the display_interface form window
 *and her subwidgets
 */
void create_form()


{


   Widget formlist[6];

   /*
    * create form_menu form window
    */
   form_menu=XtCreateWidget("Wform_menu",formWidgetClass,toplevel,argform,XtNumber(argform
   /*
    * create wcom box window
    */
   wcom=XtCreateWidget("Wcom",boxWidgetClass,form_menu,argwcom,XtNumber(argwcom));
   /*
    * create wmat1 box window
    */
   argwmat1[0].value=(XtArgVal) wcom;
   wmat1=XtCreateWidget("Wmat1",boxWidgetClass,form_menu,argwmat1,XtNumber(argwmat1));
   /*
   * create wmat2 box window
   */
  argwmat2[0].value=(XtArgVal) wmat1;
  wmat2=XtCreateWidget("Wmat2",boxWidgetClass,form_menu,argwmat2,XtNumber(argwmat2));
  /*
   * create wdat box window
   */
  argwdat[2].value=(XtArgVal) wcom;
  wdat=XtCreateWidget("Wdat",boxWidgetClass,form_menu,argwdat,XtNumber(argwdat));
  /*
   * create wtim box window
   */
  argwtim[0].value=(XtArgVal) wdat;
  argwtim[1].value=(XtArgVal) wmat1;
  wtim=XtCreateWidget("Wtim",boxWidgetClass,form_menu,argwtim,XtNumber(argwtim));
  /*
   * create wdisop box window
   */
  argwdisop[0].value=(XtArgVal) wtim;
  argwdisop[1].value=(XtArgVal) wmat2;
  wdisop=XtCreateWidget("Wdisop",boxWidgetClass,form_menu,argwdisop,XtNumber(argwdisop));
  /*
   * manage form_menu and his children
   */
  XtManageChild(form_menu);
  formlist[0]=wcom;
  formlist[1]=wmat1;
  formlist[2]=wmat2;
  formlist[3]=wdat;
  formlist[4]=wtim;
  formlist[5]=wdisop;
  XtManageChildren(formlist,XtNumber(formlist));


                  }

/*
 *callback procedures for COM children
 */

/*
 * this procedure allows the user
 * to quit Xdisplay
 */
void pcbwcom1(w,client_data,call_data)

  Widget w;
  caddr_t client_data; /*unused*/
  caddr_t call_data; /*unused*/
{
    exit(0);
}

/*
 * this procedure allows the user
 * to end a matlab display session
 */
void pcbwcom2(w,client_data,call_data)

  Widget w;
  caddr_t client_data;/*unused*/
  caddr_t call_data;/*unused*/
{
     fpMatlab=fopen("saistab.m","w");
     fprintf(fpMatlab,"endsession='oui'\n");
     fclose(fpMatlab);
}

/*
 * this procedure allows the user to
 * confirm all the modifications
 * done in the dialog widgets
 */
void pcbwcom3(w,client_data,call_data)

  Widget w;
  caddr_t client_data;/*unused*/
  caddr_t call_data;/*unused*/
{
     /*
      * please take not that the new version
      * of XtDialogGetValueString is
      * used here (i.e. XawDialogGetValueString)
      */
     matname1=XawDialogGetValueString(wmat2_1);
     matname2=XawDialogGetValueString(wmat2_2);
     matname3=XawDialogGetValueString(wmat2_3);
     matname4=XawDialogGetValueString(wmat2_4);
     printf("%s\n",matname1);
     printf("%s\n",matname2);
     printf("%s\n",matname3);
     printf("%s\n",matname4);
     delta1=XawDialogGetValueString(wtim1);
     delta2=XawDialogGetValueString(wtim2);
     delta3=XawDialogGetValueString(wtim3);
     printf("%s\n",delta1);
     printf("%s\n",delta2);
     printf("%s\n",delta3);


}

/*
 * this procedure stores all data
 * selected in the Xdisplay menu
 * in saistab.m in a format
 * specific to matlab
 */
void pcbwcom4(w,client_data,call_data)

  Widget w;
  caddr_t client_data;/*unused*/
  caddr_t call_data;/*unused*/
{
     store_data();
}

/*
 * callback procedures for MAT1 children
 */
```

```
/*
 * this procedure allows the user to
 * change the label of the wmat1_1
 * widget on which he clicked.
 * Also, the variable used to keep
 * track of this label is accordingly
 * modified
 */
void  pcbwmat1_1(w,client_data,call_data)

  Widget w;
  caddr_t client_data; /*unused*/
  caddr_t call_data; /*unused*/

{
     bmat1_1=1-bmat1_1;
     if (bmat1_1==1)
          argwmat1_1[0].value="Graph_Simulator";
     else
          argwmat1_1[0].value="Simulator";
     XtSetValues(wmat1_1,argwmat1_1,XtNumber(argwmat1_1));

}

/*
 * idem for wmat1_2
 */
void pcbwmat1_2(w,client_data,call_data)

  Widget w;
  caddr_t client_data; /*unused*/
  caddr_t call_data; /*unused*/

{
     bmat1_2=1-bmat1_2;
     if (bmat1_2==1)
          argwmat1_2[0].value="sup_same_lines_off";
     else
          argwmat1_2[0].value="sup_same_lines_on";
     XtSetValues(wmat1_2,argwmat1_2,XtNumber(argwmat1_2));

}

/*
 * idem for wmat1_3
 */
void pcbwmat1_3(w,client_data,call_data)

  Widget w;
  caddr_t client_data; /*unused*/
  caddr_t call_data; /*unused*/

{
     bmat1_3=1-bmat1_3;
     if (bmat1_3==1)
          argwmat1_3[0].value="add_miss_lines_off";
     else
          argwmat1_3[0].value="add_miss_lines_on";
     XtSetValues(wmat1_3,argwmat1_3,XtNumber(argwmat1_3));
}

/*
 * callback procedures for DAT children
 */

/*
 * this procedure allows the user to
 * change the label of the wdat1
 * widget on which he clicked.
 * Also, the variable used to keep
 * track of this label is accordingly
 * modified
 */
void  pcbwdat1(w,client_data,call_data)

  Widget w;
  caddr_t client_data; /*unused*/
  caddr_t call_data; /*unused*/

{
     bdat1=1-bdat1;
     if (bdat1==1)
          argwdat1[0].value="Time_off";
     else
          argwdat1[0].value="Time_on";
     XtSetValues(wdat1,argwdat1,XtNumber(argwdat1));

}

/*
 * idem for wdat2
 */
void  pcbwdat2(w,client_data,call_data)

  Widget w;
  caddr_t client_data; /*unused*/
  caddr_t call_data; /*unused*/

{
     bdat2=1-bdat2;
     if (bdat2==1)
          argwdat2[0].value="Num_Tkns_off";
     else
          argwdat2[0].value="Num_Tkns_on";
     XtSetValues(wdat2,argwdat2,XtNumber(argwdat2));

}


/*
 * idem for wdat3
 */
void  pcbwdat3(w,client_data,call_data)

  Widget w;
  caddr_t client_data; /*unused*/
  caddr_t call_data; /*unused*/
```

```
{
     bdat3=1-bdat3;
     if (bdat3==1)
          argwdat3[0].value="No_Active_Inst__off";
     else
          argwdat3[0].value="No_Active_Inst_on";
     XtSetValues(wdat3,argwdat3,XtNumber(argwdat3));

}

/*
 * idem for wdat4
 */
void  pcbwdat4(w,client_data,call_data)

  Widget w;
  caddr_t client_data; /*unused*/
  caddr_t call_data; /*unused*/

{
     bdat4=1-bdat4;
     if (bdat4==1)
          argwdat4[0].value="Total_Inst_off";
     else
          argwdat4[0].value="Total_Inst_on";
     XtSetValues(wdat4,argwdat4,XtNumber(argwdat4));

}


/*
 * idem for wdat5
 */
void  pcbwdat5(w,client_data,call_data)

  Widget w;
  caddr_t client_data; /*unused*/
  caddr_t call_data; /*unused*/

{
     bdat5=1-bdat5;
     if (bdat5==1)
          argwdat5[0].value="In_Match_off";
     else
          argwdat5[0].value="In_Match_on";
     XtSetValues(wdat5,argwdat5,XtNumber(argwdat5));

}

/*
 * idem for wdat6
 */
void  pcbwdat6(w,client_data,call_data)

  Widget w;
  caddr_t client_data; /*unused*/
  caddr_t call_data; /*unused*/

{
     bdat6=1-bdat6;
     if (bdat6==1)
          argwdat6[0].value="In_Sm_off";
     else
          argwdat6[0].value="In_Sm_on";
     XtSetValues(wdat6,argwdat6,XtNumber(argwdat6));

}


/*
 * idem for wdat7
 */
void  pcbwdat7(w,client_data,call_data)

  Widget w;
  caddr_t client_data; /*unused*/
  caddr_t call_data; /*unused*/

{
     bdat7=1-bdat7;
     if (bdat7==1)
          argwdat7[0].value="In_Transit_off";
     else
          argwdat7[0].value="In_Transit_on";
     XtSetValues(wdat7,argwdat7,XtNumber(argwdat7));

}


/*
 * idem for wdat8
 */
void  pcbwdat8(w,client_data,call_data)

  Widget w;
  caddr_t client_data; /*unused*/
  caddr_t call_data; /*unused*/

{
     bdat8=1-bdat8;
     if (bdat8==1)
          argwdat8[0].value="F_Calls_off";
     else
          argwdat8[0].value="F_Calls_on";
     XtSetValues(wdat8,argwdat8,XtNumber(argwdat8));

}

/*
 * callback procedures for DISOP children
 */

/*
 * this procedure allows the user to
 * change the label of the wdisop1
 * widget on which he clicked.
 * Also, the variable used to keep
 * track of this label is accordingly
 * modified
 */
```

```
void  pcbwdisop1(w,client_data,call_data)

  Widget w;
  caddr_t client_data; /*unused*/
  caddr_t call_data; /*unused*/

{
     bdisop1=1-bdisop1;
     if (bdisop1==1)
          argwdisop1[0].value="Single_Display_off";
     else
          argwdisop1[0].value="Single_Display_on";
     XtSetValues(wdisop1,argwdisop1,XtNumber(argwdisop1));

}

/*
 * idem for wdisop2
 */
void  pcbwdisop2(w,client_data,call_data)

  Widget w;
  caddr_t client_data; /*unused*/
  caddr_t call_data; /*unused*/

{
     bdisop2=1-bdisop2;
     if (bdisop2==1)
          argwdisop2[0].value="Compare_Display_off";
     else
          argwdisop2[0].value="Compare_Display_on";
     XtSetValues(wdisop2,argwdisop2,XtNumber(argwdisop2));

}

/*
 * idem for wdisop3
 */
void  pcbwdisop3(w,client_data,call_data)

  Widget w;
  caddr_t client_data; /*unused*/
  caddr_t call_data; /*unused*/

{
     bdisop3=1-bdisop3;
     if (bdisop3==1)
          argwdisop3[0].value="Single_Stats_off";
     else
          argwdisop3[0].value="Single_Stats_on";
     XtSetValues(wdisop3,argwdisop3,XtNumber(argwdisop3));

}

/*
 * idem for wdisop4
 */
void  pcbwdisop4(w,client_data,call_data)

  Widget w;
  caddr_t client_data; /*unused*/
  caddr_t call_data; /*unused*/

{
     bdisop4=1-bdisop4;
     if (bdisop4==1)
          argwdisop4[0].value="Compare_Stats_off";
     else
          argwdisop4[0].value="Compare_Stats_on";
     XtSetValues(wdisop4,argwdisop4,XtNumber(argwdisop4));

}

/*
 * idem for wdisop5
 */
void  pcbwdisop5(w,client_data,call_data)

  Widget w;
  caddr_t client_data; /*unused*/
  caddr_t call_data; /*unused*/

{
     bdisop5=1-bdisop5;
     if (bdisop5==1)
          argwdisop5[0].value="Speed_Up_off";
     else
          argwdisop5[0].value="Speed_Up_on";
     XtSetValues(wdisop5,argwdisop5,XtNumber(argwdisop5));

}

/*
 * idem for wdisop6
 */
void  pcbwdisop6(w,client_data,call_data)

  Widget w;
  caddr_t client_data; /*unused*/
  caddr_t call_data; /*unused*/

{
     bdisop6=1-bdisop6;
     if (bdisop6==1)
          argwdisop6[0].value="Sensitivity_off";
     else
          argwdisop6[0].value="Sensitivity_on";
     XtSetValues(wdisop6,argwdisop6,XtNumber(argwdisop6));

}


/*
 *this procedure fills the COM window
 *with all her managed children
 */

void create_com()

{
    Widget comlist[4];

    /*
     * create wcom1 command widget
     */
    cbwcom1[0].callback=(XtArgVal) pcbwcom1;
    wcom1=XtCreateWidget("Wcom1",commandWidgetClass,wcom,argwcom1,XtNumber(argwcom1));
    /*
     * create wcom2 command widget
     */
    cbwcom2[0].callback=(XtArgVal) pcbwcom2;
    wcom2=XtCreateWidget("Wcom2",commandWidgetClass,wcom,argwcom2,XtNumber(argwcom2));
    /*
     * create wcom3 command widget
     */
    cbwcom3[0].callback=(XtArgVal) pcbwcom3;
    wcom3=XtCreateWidget("Wcom3",commandWidgetClass,wcom,argwcom3,XtNumber(argwcom3));
    /*
     * create wcom4 command widget
     */
    cbwcom4[0].callback=(XtArgVal) pcbwcom4;
    wcom4=XtCreateWidget("Wcom4",commandWidgetClass,wcom,argwcom4,XtNumber(argwcom4));
    /*
     * manage wcom children
     */
    comlist[0]=wcom1;
    comlist[1]=wcom2;
    comlist[2]=wcom3;
    comlist[3]=wcom4;
    XtManageChildren(comlist,XtNumber(comlist));
}

/*
 *this procedure  fills the MAT1 display subwindow
 * with all her managed children
 */

void create_mat1()

{
    Widget mat1list[3];

    /*
     * create wmat1_1 command widget
     */
    cbwmat1_1[0].callback=(XtArgVal) pcbwmat1_1;
    wmat1_1=XtCreateWidget("Wmat1_1",commandWidgetClass,wmat1,argwmat1_1,XtNumber(argwma
    /*
     * create wmat1_2 command widget
     */
    cbwmat1_2[0].callback=(XtArgVal) pcbwmat1_2;
    wmat1_2=XtCreateWidget("Wmat1_2",commandWidgetClass,wmat1,argwmat1_2,XtNumber(argwma
    /*
     * create wmat1_3 command widget
     */
    cbwmat1_3[0].callback=(XtArgVal) pcbwmat1_3;
    wmat1_3=XtCreateWidget("Wmat1_3",commandWidgetClass,wmat1,argwmat1_3,XtNumber(argwma
    /*
     * manage wmat1 children
     */
     mat1list[0]=wmat1_1;
     mat1list[1]=wmat1_2;
     mat1list[2]=wmat1_3;
     XtManageChildren(mat1list,XtNumber(mat1list));
}

/*
 *this procedure  fills the MAT2 window
 *with all her managed children
 */

void create_mat2()

{
     Widget mat2list[4];

     /*
      * create wmat2_1 dialog widget
      */
     argwmat2_1[1].value=(XtArgVal) matname1;
     wmat2_1=XtCreateWidget("Wmat2_1",dialogWidgetClass,wmat2,argwmat2_1,XtNumber(argwma
     /*
      * create wmat2_2 dialog widget
      */
     argwmat2_2[1].value=(XtArgVal) matname2;
     wmat2_2=XtCreateWidget("Wmat2_2",dialogWidgetClass,wmat2,argwmat2_2,XtNumber(argwmat
     /*
      * create wmat2_3 dialog widget
      */
     argwmat2_3[1].value=(XtArgVal) matname3;
     wmat2_3=XtCreateWidget("Wmat2_3",dialogWidgetClass,wmat2,argwmat2_3,XtNumber(argwmat
     /*
      * create wmat2_4 dialog widget
      */
     argwmat2_4[1].value=(XtArgVal) matname4;
     wmat2_4=XtCreateWidget("Wmat2_4",dialogWidgetClass,wmat2,argwmat2_4,XtNumber(argwmat
     /*
      * manage wmat2 children
      */
     mat2list[0]=wmat2_1;
     mat2list[1]=wmat2_2;
     mat2list[2]=wmat2_3;
     mat2list[3]=wmat2_4;
     XtManageChildren(mat2list,XtNumber(mat2list));

}

/*
 *this procedure  fills the DAT  subwindow
 * with all her managed children
 */

void create_dat()

{
     Widget datlist[8];

     /*
      * create wdat1 command widget
      */
     cbwdat1[0].callback=(XtArgVal) pcbwdat1;
     wdat1=XtCreateWidget("Wdat1",commandWidgetClass,wdat,argwdat1,XtNumber(argwdat1));
     /*
      * create wdat2 command widget
      */
     cbwdat2[0].callback=(XtArgVal) pcbwdat2;
     wdat2=XtCreateWidget("Wdat2",commandWidgetClass,wdat,argwdat2,XtNumber(argwdat2));
     /*
      * create wdat3 command widget
      */
     cbwdat3[0].callback=(XtArgVal) pcbwdat3;
     wdat3=XtCreateWidget("Wdat3",commandWidgetClass,wdat,argwdat3,XtNumber(argwdat3));
     /*
      * create wdat4 command widget
      */
     cbwdat4[0].callback=(XtArgVal) pcbwdat4;
     wdat4=XtCreateWidget("Wdat4",commandWidgetClass,wdat,argwdat4,XtNumber(argwdat4));
     /*
      * create wdat5 command widget
      */
     cbwdat5[0].callback=(XtArgVal) pcbwdat5;
     wdat5=XtCreateWidget("Wdat5",commandWidgetClass,wdat,argwdat5,XtNumber(argwdat5));
     /*
      * create wdat6 command widget
      */
     cbwdat6[0].callback=(XtArgVal) pcbwdat6;
     wdat6=XtCreateWidget("Wdat6",commandWidgetClass,wdat,argwdat6,XtNumber(argwdat6));
     /*
      * create wdat7 command widget
      */
     cbwdat7[0].callback=(XtArgVal) pcbwdat7;
     wdat7=XtCreateWidget("Wdat7",commandWidgetClass,wdat,argwdat7,XtNumber(argwdat7));
     /*
      * create wdat8 command widget
      */
     cbwdat8[0].callback=(XtArgVal) pcbwdat8;
     wdat8=XtCreateWidget("Wdat8",commandWidgetClass,wdat,argwdat8,XtNumber(argwdat8));
     /*
      * manage wdat children
      */
     datlist[0]=wdat1;
     datlist[1]=wdat2;
     datlist[2]=wdat3;
     datlist[3]=wdat4;
     datlist[4]=wdat5;
     datlist[5]=wdat6;
     datlist[6]=wdat7;
     datlist[7]=wdat8;
     XtManageChildren(datlist,XtNumber(datlist));
}

/*
 *this procedure  fills the TIM window
 *with all her managed children
 */

void create_tim()

{
     Widget timlist[3];
     /*
      * create wtim1 dialog widget
      */
     argwtim1[1].value=(XtArgVal) delta1;
     wtim1=XtCreateWidget("Wtim1",dialogWidgetClass,wtim,argwtim1,XtNumber(argwtim2));
     /*
      * create wtim2 dialog widget
      */
     argwtim2[1].value=(XtArgVal) delta2;
     wtim2=XtCreateWidget("Wtim2",dialogWidgetClass,wtim,argwtim2,XtNumber(argwtim2));
     /*
      * create wtim3 dialog widget
      */
     argwtim3[1].value=(XtArgVal) delta3;
     wtim3=XtCreateWidget("Wtim3",dialogWidgetClass,wtim,argwtim3,XtNumber(argwtim3));
     /*
      * manage wtim children
      */
     timlist[0]=wtim1;
     timlist[1]=wtim2;
     timlist[2]=wtim3;
     XtManageChildren(timlist,XtNumber(timlist));

}

/*
 *this procedure  fills the DISOP  subwindow
 * with all her managed children
 */

void create_disop()

{
     Widget disoplist[6];

     /*
      * create wdisop1 command widget
      */
     cbwdisop1[0].callback=(XtArgVal) pcbwdisop1;
     wdisop1=XtCreateWidget("Wdisop1",commandWidgetClass,wdisop,argwdisop1,XtNumber(argwd
     /*
      * create wdisop2 command widget
      */
     cbwdisop2[0].callback=(XtArgVal) pcbwdisop2;
     wdisop2=XtCreateWidget("Wdisop2",commandWidgetClass,wdisop,argwdisop2,XtNumber(argwd
     /*
      * create wdisop3 command widget
      */
     cbwdisop3[0].callback=(XtArgVal) pcbwdisop3;
     wdisop3=XtCreateWidget("Wdisop3",commandWidgetClass,wdisop,argwdisop3,XtNumber(argwd
     /*
      * create wdisop4 command widget
      */
     cbwdisop4[0].callback=(XtArgVal) pcbwdisop4;
     wdisop4=XtCreateWidget("Wdisop4",commandWidgetClass,wdisop,argwdisop4,XtNumber(argwd
     /*
      * create wdisop5 command widget
      */
     cbwdisop5[0].callback=(XtArgVal) pcbwdisop5;
     wdisop5=XtCreateWidget("Wdisop5",commandWidgetClass,wdisop,argwdisop5,XtNumber(argw
     /*
      * create wdisop6 command widget
      */
     cbwdisop6[0].callback=(XtArgVal) pcbwdisop6;
     wdisop6=XtCreateWidget("Wdisop6",commandWidgetClass,wdisop,argwdisop6,XtNumber(argw
     /*
      * manage wdisop children
      */
     disoplist[0]=wdisop1;
     disoplist[1]=wdisop2;
     disoplist[2]=wdisop3;
     disoplist[3]=wdisop4;
     disoplist[4]=wdisop5;
     disoplist[5]=wdisop6;
     XtManageChildren(disoplist,XtNumber(disoplist));
}

/*
 *open the display server
 *and maps the  display_interface selection menu Xmenu
 */

main(argc,argv)
  int argc;
  char **argv;
{

   /*
    *Create the Widget  that supports all the application
    */
   toplevel=XtInitialize(argv[0],"XLabel",NULL,0,&argc,argv);

   create_form();

   create_com();

   create_mat1();

   create_mat2();

   create_dat();

   create_tim();

   create_disop();

   /*
    *Create the windows and set their attributes
    *according to the Widget data
    */
   XtRealizeWidget(toplevel);

   /*
    *Process the events
    */
   XtMainLoop();
}
```



PARAM ***PARAM***

PARAM

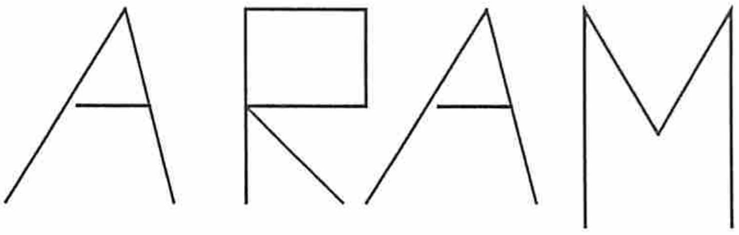
PARAM ***PARAM***

PARAM

PARAM

PARAM

PARAM



PARAM

```
/*       *************************************
         *                                   *
         *                                   *
         *          XPARAM                   *
         *                                   *
         *     copyright Olivier TARDIEU     *
         *                                   *
         *          ( 1990 )                 *
         *                                   *
         *                                   *
         *************************************
 */
```

```
#include <stdio.h>
#include "X11/Xlib.h"
#include "X11/Intrinsic.h"
#include "X11/StringDefs.h"
#include "X11/Xaw/Label.h"
#include "X11/Xaw/Form.h"
#include "X11/Xaw/Box.h"
#include "X11/Xaw/Command.h"
#include "X11/Xaw/Dialog.h"


FILE *fpDirSimu,*fpsim1,*fpsim2,*fopen();

typedef char STRING[80];

struct SimuID {
     char sname[80];
     char simtype[80];
     char PE[80];
     char CE[80];
     char StM[80];
     char Match[80];
     char sufx[80];
};

struct SimuID workID1,workID2;

/*
 *variables used to keep track of Xparam_selected contents"
 */

int bcomp1,bcomp2,bcomp3,bcomp4;
int bpar1_1=1 ;

int bhidc;
```

```
char *parameter1="",*parameter2="";
char *parameter3="",*parameter4="";
char *parameter5="",*parameter6="";

STRING hidname[10];


/*
 * application rootwindow
 */
Widget topparam;


/*
 *main form window with included subwindows
 */
/*form widget*/
Widget form_param;
/*boxwidgets*/
Widget wcomp,wpar1,wpar2,whid1,whid2;

/*
 *Session commands (first form window)
 */
/*command widgets*/
Widget wcomp1,wcomp2,wcomp3,wcomp4;


/*
 *simulator type descriptor
 */
/*commandwidgets*/
Widget wpar1_1;

/*
 *simulation file identification (names and parameters)
 */
/*dialogwidgets*/
Widget wpar2_1,wpar2_2,wpar2_3,wpar2_4,wpar2_5,wpar2_6;

/*
 *simulation_file names
 */
/*commandwidget*/
Widget whidc;
/*labelwidget*/
Widget whidl;
/*dialogwidget*/
Widget whidn[10];

/*
 *CallBack_Lists for wcomp subwidgets
 */

XtCallbackRec cbwcomp1[]={
{NULL,NULL},
{NULL,NULL},
};
```

```
XtCallbackRec cbwcomp2[ ]={
{NULL,NULL},
{NULL,NULL},
};

XtCallbackRec cbwcomp3[ ]={
{NULL,NULL},
{NULL,NULL},
};

XtCallbackRec cbwcomp4[ ]={
{NULL,NULL},
{NULL,NULL},
};


/*
 * CallBack_Lists  for wpar1 subwidgets
 */


XtCallbackRec cbwpar1_1[ ]={
{NULL,NULL},
{NULL,NULL},
};

/*
 *CallBack_Lists for whid1 subwidgets
 */

XtCallbackRec cbwhidc[ ]={
{NULL,NULL},
{NULL,NULL},
};

XtCallbackRec cbwhidl[ ]={
{NULL,NULL},
{NULL,NULL},
};


/*
 *form_param Widget argument list
 */

Arg argform[ ]={
{XtNheight,(XtArgVal) 60},
{XtNwidth ,(XtArgVal) 50},
};

/*
 * form window subwidgets argument lists
 */

/*
 * give arguments to position wcomp on left-top
 * corner of form
 */
```

```
Arg argwcomp[]={
{XtNleft,(XtArgVal) XtChainLeft},
{XtNtop,(XtArgVal) XtChainTop},
};

/*
 * give arguments to chain wpar1 on leftside of
 * form and specify that wpar1 will be chained (vertically)
 * to another window whose ID can not be specified yet
 * because it does not exist
 *  (this window will be wcomp)
 */
Arg argwpar1[]={
{XtNfromVert,(XtArgVal) NULL},
{XtNleft,(XtArgVal) XtChainLeft},
{XtNvertDistance,(XtArgVal) 10},
};

/*
 * give arguments to chain wpar2 on top-right corner of form
 * and specify that wpar2 will be chained (horizontally) to
 * another window  whose ID can not  be specified yet because
 * it does not exist.
 *  (this window will be wcomp)
 */
Arg argwpar2[]={
{XtNfromHoriz,(XtArgVal) NULL},
{XtNtop,(XtArgVal) XtChainTop},
{XtNhorizDistance,(XtArgVal) 10},
{XtNright,(XtArgVal) XtChainRight},
};

/*
 * give arguments to position whid1 on left-top
 * corner of form
 */
Arg argwhid1[]={
{XtNleft,(XtArgVal) XtChainLeft},
{XtNtop,(XtArgVal) XtChainTop},
};

/*
 * give arguments to chain whid2 on top-right corner of form
 * and specify that whid2 will be chained (horizontally) to
 * another window  whose ID can not  be specified yet because
 * it does not exist.
 *  (this window will be whid1)
 */
Arg argwhid2[]={
{XtNfromHoriz,(XtArgVal) NULL},
{XtNtop,(XtArgVal) XtChainTop},
{XtNhorizDistance,(XtArgVal) 10},
{XtNright,(XtArgVal) XtChainRight},
};

/*
 * form window subwidgets 's children argument lists
 */

/*
```

```
 *COMP children argument lists
 */

/*
 * give arguments to specify the label to be displayed
 * in wcomp1 window at the start of Xparam and the procedure
 * that is called whenever a selection is made on this
 * window
 */
Arg argwcomp1[]={
{XtNlabel, (XtArgVal) "End_Window_Menu"},
{XtNcallback,(XtArgVal) cbwcomp1},
};

/*
 *idem for wcomp2
 */
Arg argwcomp2[]={
{XtNlabel, (XtArgVal) "Store_Simulation_File"},
{XtNcallback,(XtArgVal) cbwcomp2},
};

/*
 *idem for wcomp3
 */
Arg argwcomp3[]={
{XtNlabel, (XtArgVal) "Retrieve_Simu_Files"},
{XtNcallback,(XtArgVal) cbwcomp3},
};

/*
 *idem for wcomp4
 */
Arg argwcomp4[]={
{XtNlabel, (XtArgVal) "List_Current_Simu_IDs"},
{XtNcallback,(XtArgVal) cbwcomp4},
};

/*
 *PAR1 children argument lists
 */

/*
 * give arguments to specify the label to be displayed
 * in wcpar1_1 window at the start of Xparam and the procedure
 * that is called whenever a selection is made on this
 * window
 */
Arg argwpar1_1[]={
{XtNlabel, (XtArgVal) "Graph_Simulator"},
{XtNcallback,(XtArgVal) cbwpar1_1},
};

/*
 *PAR2 children argument lists
 */

/*
 * give arguments to specify the label to be displayed
 * in wcpar2_1 window at the start of Xparam and the
 * string to be shown on the editor part of the
 * dialog widget
 * (this string is not specified here because its
 *  value is assigned at run-time. This is not
 *  useful yet but will certainly prove to be
 *  when the simulator environment will be expanded)
 */
Arg argwpar2_1[]={
{XtNlabel,(XtArgVal) "sourcefile_name"},
{XtNvalue,(XtArgVal) NULL},
};

/*
 *idem for wpar2_2
 */
Arg argwpar2_2[]={
{XtNlabel,(XtArgVal) "#_of_PE_servers"},
{XtNvalue,(XtArgVal) NULL},
};

/*
 *idem for wpar2_3
 */
Arg argwpar2_3[]={
{XtNlabel,(XtArgVal) "#_of_CE_servers"},
{XtNvalue,(XtArgVal) NULL},
};

/*
 *idem for wpar2_4
 */
Arg argwpar2_4[]={
{XtNlabel,(XtArgVal) "#Str_Mem_servers"},
{XtNvalue,(XtArgVal) NULL},
};

/*
 *idem for wpar2_5
 */
Arg argwpar2_5[]={
{XtNlabel,(XtArgVal) "#_of_Match"},
{XtNvalue,(XtArgVal) NULL},
};

/*
 *idem for wpar2_6
 */
Arg argwpar2_6[]={
{XtNlabel,(XtArgVal) "suffixe"},
{XtNvalue,(XtArgVal) NULL},
};

/*
 *HID1 children argument lists
 */

/*
 * give arguments to specify the label to be displayed
 * in whidc window at the start of Xparam and the procedure
 * that is called whenever a selection is made on this
 * window
 */
Arg argwhidc[]={
{XtNlabel, (XtArgVal) "Override_Xmenu"},
{XtNcallback,(XtArgVal) cbwhidc},
};

/*
 * idem for whidl
 */
Arg argwhidl[]={
{XtNlabel, (XtArgVal) "nam_PE_CE_StM_Ma_suf"},
{XtNcallback,(XtArgVal) cbwhidl},
};

/*
 * give arguments to specify the label to be displayed
 * in whidn1 window at the start of Xparam and the
 * string to be shown on the editor part of the
 * dialog widget
 * (this string is not specified here because its
 *  value is assigned at run-time. This is not
 *  useful yet but will certainly prove to be
 *  when the simulator environment will be expanded)
 */
Arg argwhidn1[]={
{XtNlabel,(XtArgVal) "available_simu_file1"},
{XtNvalue,(XtArgVal) NULL},
};

/*
 * idem for whidn2
 */
Arg argwhidn2[]={
{XtNlabel,(XtArgVal) "available_simu_file2"},
{XtNvalue,(XtArgVal) NULL},
};

/*
 * idem for whidn3
 */
Arg argwhidn3[]={
{XtNlabel,(XtArgVal) "available_simu_file3"},
{XtNvalue,(XtArgVal) NULL},
};

/*
 * idem for whidn4
 */
Arg argwhidn4[]={
{XtNlabel,(XtArgVal) "available_simu_file4"},
{XtNvalue,(XtArgVal) NULL},
};

/*
 *HID2 children argument lists
 */

/*
 * idem for whidn5
 */
Arg argwhidn5[]={
{XtNlabel,(XtArgVal) "available_simu_file5"},
{XtNvalue,(XtArgVal) NULL},
};

/*
 * idem for whidn6
 */
Arg argwhidn6[]={
{XtNlabel,(XtArgVal) "available_simu_file6"},
{XtNvalue,(XtArgVal) NULL},
};

/*
 * idem for whidn7
 */
Arg argwhidn7[]={
{XtNlabel,(XtArgVal) "available_simu_file7"},
{XtNvalue,(XtArgVal) NULL},
};

/*
 * idem for whidn8
 */
Arg argwhidn8[]={
{XtNlabel,(XtArgVal) "available_simu_file8"},
{XtNvalue,(XtArgVal) NULL},
};

/*
 * idem for whidn9
 */
Arg argwhidn9[]={
{XtNlabel,(XtArgVal) "available_simu_file9"},
{XtNvalue,(XtArgVal) NULL},
};

/*
 * idem for whidn10
 */
Arg argwhidn10[]={
{XtNlabel,(XtArgVal) "available_simu_file10"},
{XtNvalue,(XtArgVal) NULL},
};

/*
 *BUG proof version of strcpy
 */
strcopy(s,t)

char s[],t[];

{
     int i;

     i=0;
     while((s[i]=t[i]) != '\0')
          i++;
}
```

```
strcomp(s,t)

char s[],t[];

{
     int i;

     i=0;
     while(s[i]==t[i])
          if (s[i++] =='\0')
               return(0);
     return(s[i]-t[i]);
}

/*
 *procedures used by callback procedures
 */

/*
 * this procedure stores all the strings displayed
 * in the wpar2_i windows in the coresponding
 * parameteri variables
 */
void GVSparam()

{
     /*
      * XawDialogGetValueString is used instead
      * of XtDialogGetValueString which remains only
      * for compatibility facility and does not
      * seem to work properly
      */
     parameter1=XawDialogGetValueString(wpar2_1);
     parameter2=XawDialogGetValueString(wpar2_2);
     parameter3=XawDialogGetValueString(wpar2_3);
     parameter4=XawDialogGetValueString(wpar2_4);
     parameter5=XawDialogGetValueString(wpar2_5);
     parameter6=XawDialogGetValueString(wpar2_6);

}

/*
 * this procedure stores all the strings displayed
 * in the whidn[i] windows in the coresponding
 * hidname[i] variables
 */
void GVShidnam()

{
     int i;

     for(i=0;i<10;i++)
          strcpy(hidname[i],XawDialogGetValueString(whidn[i]));

}

/*
 *this procedure reads the contents of the par1 commandwidget and par2
 *dialogwidgets and puts the results in the variable rec
 */
void store_ID(rec)

struct SimuID *rec;

{
     if (bpar1_1==1)
          strcpy((*rec).simtype,"G");
     else
          strcpy((*rec).simtype,"S");
     strcpy((*rec).sname,parameter1);
     strcpy((*rec).PE,parameter2);
     strcpy((*rec).CE,parameter3);
     strcpy((*rec).StM,parameter4);
     strcpy((*rec).Match,parameter5);
     strcpy((*rec).sufx,parameter6);

}

/*
 *this procedure uses the information stored
 *in the SimuID rec and creates a path_name towards
 *the concerned simulation file
 */
void pathID(rec,s)

struct SimuID rec;
char s[80];

{
     strcpy(s,"");
     strcat(s,rec.sname);
     strcat(s,"/");
     strcat(s,rec.simtype);
     strcat(s,rec.PE);
     strcat(s,rec.CE);
     strcat(s,rec.StM);
     strcat(s,rec.Match);
     strcat(s,rec.sufx);
}

/*
 *this procedure checks in the IDhome file
 *(which is created if it doesn't exist)
 *containing simuID.sname s strings to see
 *if the workID1.sname directory exists.
 *if it is not the case,this directory is created
 *and refload.mat0 is copied in this new directory under
 *the name refload.mat
 */
void cre_dir_name()

{
     char st[80],st1[80],st2[80];
     int n,flag;
     char b;

     /*
      *open the file IDhome in append to make
      *sure that it is created if it does
      *not exist and will not be modified if
      *it does exist
      */
     fpsim1=fopen("IDhome","a");
     fclose(fpsim1);
     /*
      *check to see if workID1.sname is
      *recorded in the IDhome file
      *in this case flag <-1
      */
     fpsim1=fopen("IDhome","r");
     rewind(fpsim1);
     flag=0;
     n=1;
     strcpy(st,"");
     while (fscanf(fpsim1,"%s",st) !=EOF) {
          if (strcmp(st,workID1.sname)==0) {
               flag=1;
          }
     }
     fclose(fpsim1);
     /*
      *workID1.sname directory does not exist
      */
     if (flag==0) {
          /*
           *add name of directory to be
           *created in IDhome
           */
          fpsim1=fopen("IDhome","a");
          fprintf(fpsim1,"%s\n",workID1.sname);
          fclose(fpsim1);
          /*
           *create directory
           */
          strcpy(st1,"mkdir ");
          strcat(st1,workID1.sname);
          system(st1);
          /*
           *copy refload.mat0 in this new
           *directory under refload.mat name
           */
          strcpy(st2,"cp refload.mat0 ");
          strcat(st2,workID1.sname);
          strcat(st2,"/refload.mat");
          system(st2);
     }

}

/*
 *this procedure appends the contents of the simuID workID1
 *in the file fpsim1
 *the procedure adds rec to the end of the file
 */
void write_ID()

{
     char st[80];
     store_ID(&workID1);
     /*
      *store the simulation ID in ID which is a
      *file of the workID.sname directory
      */
     strcpy(st,workID1.sname);
     strcat(st,"/ID");
     fpsim1=fopen(st,"a");
     fprintf(fpsim1,"%s%s",workID1.sname," ");
     fprintf(fpsim1,"%s%s",workID1.simtype," ");
     fprintf(fpsim1,"%s%s",workID1.PE," ");
     fprintf(fpsim1,"%s%s",workID1.CE," ");
     fprintf(fpsim1,"%s%s",workID1.StM," ");
     fprintf(fpsim1,"%s%s",workID1.Match," ");
     fprintf(fpsim1,"%s\n",workID1.sufx);
     fclose(fpsim1);
}

/*
 *this procedure stores the current ID in workID1
 *then executes cre_dir_name and calls write_ID
 *This is done to update IDhome and various sname/ID
 *files before storing the WWW simulation file
 *which is lastly done in this procedure
 */
void store_sim()

{
     char st[80],stcp[80];

     store_ID(&workID1);
     cre_dir_name();
     write_ID();
     pathID(workID1,st);
     /*
      *copy WWW file created by the simulator
      *under the pathID(workID).m name
      */
     strcpy(stcp,"cp WWW ");
     strcat(stcp,st);
     strcat(stcp,".m");
     system(stcp);
}

/*this procedure creates a file listID which
 *contains all the simulation file pathIDs whose
 *description corresponds to the information
 *stored in the current Menu parameters
 *parameter[0]='?' means that the corresponding
 *value of pathID can be anything
 */
void cre_list()

{
     int flag;
     char st[80],st1[80];

     /*
      *check and store menu contents
      *related to parameters
 */
GVSparam();
/*
 *open simuID.sname/ID file (coresponding
 *to first parameter selected by user) in
 *read. In the current implementation of this
 *program, we suppose that the user gives a
 *correct name.
 */
store_ID(&workID1);
strcpy(st,workID1.sname);
strcat(st,"/ID");
fpsim1=fopen(st,"r");
/*
 *open listID (listID is eventually
 *overwritten if it existed already)
 */
fpsim2=fopen("listID","w");
/*
 *write in listID all the names of the
 *simulations_result_files stored in
 *workID1.sname directory whose ID
 *matches the parameters selected by the
 *user
 */
while (fscanf(fpsim1,"%s",st) != EOF) {
     strcpy(workID2.sname,st);
     fscanf(fpsim1,"%s",workID2.simtype);
     fscanf(fpsim1,"%s",workID2.PE);
     fscanf(fpsim1,"%s",workID2.CE);
     fscanf(fpsim1,"%s",workID2.StM);
     fscanf(fpsim1,"%s",workID2.Match);
     fscanf(fpsim1,"%s",workID2.sufx);
     flag=0;
     /*
      * a '?' in a parameter means that the user
      * does not care of the value of the parameter
      * for the selection
      */
     if ((strcmp(workID1.simtype,workID2.simtype)==0)|(workID1.simtype[0]=='?'))
          flag++;
     if ((strcmp(workID1.PE,workID2.PE)==0)|(workID1.PE[0]=='?'))
          flag++;
     if ((strcmp(workID1.CE,workID2.CE)==0)|(workID1.CE[0]=='?'))
          flag++;
     if ((strcmp(workID1.StM,workID2.StM)==0)|(workID1.StM[0]=='?'))
          flag++;
     if ((strcmp(workID1.Match,workID2.Match)==0)|(workID1.Match[0]=='?'))
          flag++;
     if ((strcmp(workID1.sufx,workID2.sufx)==0)|(workID1.sufx[0]=='?'))
          flag++;
     /*
      * the ID of the simulation matches
      * the parameters selected by the user.
      * store simuID in listID
      */
     if (flag==6) {
          pathID(workID2,st1);
          fprintf(fpsim2,"%s\n",st1);}
}
```

```
	fclose(fpsim1);
	fclose(fpsim2);
}

/*
 *this procedure stores the n
 *pathIDs of listID in the hidname
 *array. if n<length(hidname)
 *"" strings are stored.
 */
void wr_lID_hidn()

{
	int flag;
	char st[80];

	flag=0;
	fpsim1=fopen("listID","r");
	/*
	 *store all the IDs of listID in
	 *the hidname array in the order
	 *where they appear in listID
	 */
	while (fscanf(fpsim1,"%s",st) != EOF) {
		strcpy(hidname[flag],st);
		flag++;
	}
	/*
	 *if number IDs in list ID < 10
	 *fill the remaining lines of
	 *hidname with ""
	 */
	while (flag != 10) {
		strcpy(hidname[flag],"");
		flag++;
	}
	fclose(fpsim1);
	/*
	 * show the IDs selected by the user to
	 * the user in the whidn[i] dialog widgets
	 */
	XtSetValues(whidn[0],argwhidn1,XtNumber(argwhidn1));
	XtSetValues(whidn[1],argwhidn2,XtNumber(argwhidn2));
	XtSetValues(whidn[2],argwhidn3,XtNumber(argwhidn3));
	XtSetValues(whidn[3],argwhidn4,XtNumber(argwhidn4));
	XtSetValues(whidn[4],argwhidn5,XtNumber(argwhidn5));
	XtSetValues(whidn[5],argwhidn6,XtNumber(argwhidn6));
	XtSetValues(whidn[6],argwhidn7,XtNumber(argwhidn7));
	XtSetValues(whidn[7],argwhidn8,XtNumber(argwhidn8));
	XtSetValues(whidn[8],argwhidn9,XtNumber(argwhidn9));
	XtSetValues(whidn[9],argwhidn10,XtNumber(argwhidn10));
}

/*
 *this procedure creates the matname2.m file which
 *contains the list of all the elements
 *of the hidname array at the moment where
 *override display is selected
 * a '*' preceding a string means that the corresponding
 *string is not recorded in matname2.m .
 *as matname2.m is read after Xtsaistab.m, the value of
 *matnames in Xtsaistab.m is overwritten
 */
void rec_mtnm2()

{
     int i;

     fpsim1=fopen("matname2.m","w");
     GVShidnam();
     fprintf(fpsim1,"matnames=[\n");
     /*
      * store all IDs whose first
      * character is not a star
      * in matname2.m
      */
     for (i=0;i<10;i++) {
          if (hidname[i][0]=='*')
               ;
          else {
               if (strcmp(hidname[i],"")==0)
                    ;
               else {
                    fprintf(fpsim1,"'");
                    fprintf(fpsim1,"%s",hidname[i]);
                    fprintf(fpsim1,"';\n");
               }
          }
     }
     fprintf(fpsim1,"]\n");
     fclose(fpsim1);
     /*
      *this results in the following
      *stored in matname2.m
      * matnames=[
      * 'string1'
      * 'string2'
      * '.......'
      * 'stringp'
      * ];
      * with p <= 10
      */
}


/*
 *this procedure creates and manages the main form window
 *and her subwidgets
 */
void create_formp()


{


  Widget formlist[5];

  /*
   *create form_param
   */
  form_param=XtCreateWidget("Wform_param",formWidgetClass,topparam,argform,XtNumber(argf
  /*
   *create wcomp
   */
  wcomp=XtCreateWidget("Wcomp",boxWidgetClass,form_param,argwcomp,XtNumber(argwcomp));
  /*
   *create wpar1
   */
  argwpar1[0].value=(XtArgVal) wcomp;
  wpar1=XtCreateWidget("Wpar1",boxWidgetClass,form_param,argwpar1,XtNumber(argwpar1));
  /*
   *create wpar2
   */
  argwpar2[0].value=(XtArgVal) wcomp;
  wpar2=XtCreateWidget("Wpar2",boxWidgetClass,form_param,argwpar2,XtNumber(argwpar2));
  /*
   *create whid1
   */
  whid1=XtCreateWidget("Whid1",boxWidgetClass,form_param,argwhid1,XtNumber(argwhid1));
  /*
   *create whid2
   */
  argwhid2[0].value=(XtArgVal) whid1;
  whid2=XtCreateWidget("Whid2",boxWidgetClass,form_param,argwhid2,XtNumber(argwhid2));
  /*
   * manage all created windows
   */
  XtManageChild(form_param);
  formlist[0]=wcomp;
  formlist[1]=wpar1;
  formlist[2]=wpar2;
  formlist[3]=whid1;
  formlist[4]=whid2;
  XtManageChildren(formlist,XtNumber(formlist));
  /*
   * Unmap whid1 and whid2 windows
   * (which seem now to be hidden behind the
   *  other subwindows of form_param)
   */
  XtSetMappedWhenManaged(whid1,False);
  XtSetMappedWhenManaged(whid2,False);



                }


/*
 *callback procedures for COMP children
 */

/*
 * this procedure allows the
 * user to quit Xparam
 */
void pcbwcomp1(w,client_data,call_data)

  Widget w;
  caddr_t client_data; /*unused*/
  caddr_t call_data; /*unused*/
```

```
{
     exit(0);
}

/*
 * this procedure allows the user to
 * store a simulation_result file
 * created by the simulator
 */
void pcbwcomp2(w,client_data,call_data)

  Widget w;
  caddr_t client_data;/*unused*/
  caddr_t call_data;/*unused*/
{
     GVSparam();
     store_sim();
}

/*
 * this procedure allows the user to
 * "retrieve" simulation_result files
 * whose ID matches the selected
 * parameters
 */
void pcbwcomp3(w,client_data,call_data)

  Widget w;
  caddr_t client_data;/*unused*/
  caddr_t call_data;/*unused*/
{
     cre_list();
     wr_lID_hidn();
     /*
      * let the hidden windows come
      * in front of the Xparam menu
      * window
      */
     XtSetMappedWhenManaged(wcomp,False);
     XtSetMappedWhenManaged(wpar1,False);
     XtSetMappedWhenManaged(wpar2,False);
     XtSetMappedWhenManaged(whid1,True);
     XtSetMappedWhenManaged(whid2,True);
}

/*
 * this procedure allows the user to
 * get the list of all simulations_result
 * files whose ID matches the selected parameters
 */
void pcbwcomp4(w,client_data,call_data)

  Widget w;
  caddr_t client_data;/*unused*/
  caddr_t call_data;/*unused*/
{
     cre_list();
     printf("*****************************\n");
     printf("*                           *\n");
     printf("*   Xparam selected ID's    *\n");
     printf("*                              *\n");
     printf("********************************\n");
     printf("\n");
     printf("\n");
     system("more listID");
}

/*
 * callback procedures for PAR1 children
 */

/*
 * this procedure allows the user
 * to change the label of the wpar1_1
 * widget on which he clicked.
 * also, the variable used to keep track
 * of this label is accordingly modified.
 */
void  pcbwpar1_1(w,client_data,call_data)

  Widget w;
  caddr_t client_data; /*unused*/
  caddr_t call_data; /*unused*/

{
     bpar1_1=1-bpar1_1;
     if (bpar1_1==1) {
          argwpar1_1[0].value="Graph_Simulator";
     }
     else {
          argwpar1_1[0].value="Simulator";
     }
     XtSetValues(wpar1_1,argwpar1_1,XtNumber(argwpar1_1));
}

/*
 *callback procedures for HID1 children
 */

/*
 * this procedure allows the user
 * to generate the matname2.m file
 * coresponding to the data displayed
 * in the whidn[i] dialog widgets and
 * to come back to the main Xparam menu
 */
void pcbwhidc(w,client_data,call_data)

  Widget w;
  caddr_t client_data; /*unused*/
  caddr_t call_data; /*unused*/
{
     rec_mtnm2();
     /*
      * 'hide' whid1 and whid2 and let
      * the other windows of form_param
      * appear.
      */
     XtSetMappedWhenManaged(whid1,False);
     XtSetMappedWhenManaged(whid2,False);
     XtSetMappedWhenManaged(wcomp,True);
     XtSetMappedWhenManaged(wpar1,True);
     XtSetMappedWhenManaged(wpar2,True);
}

/*
 * this procedure allows the user to
 * come back to the main Xparam menu
 */
void pcbwhidl(w,client_data,call_data)

  Widget w;
  caddr_t client_data; /*unused*/
  caddr_t call_data; /*unused*/
{
     XtSetMappedWhenManaged(whid1,False);
     XtSetMappedWhenManaged(whid2,False);
     XtSetMappedWhenManaged(wcomp,True);
     XtSetMappedWhenManaged(wpar1,True);
     XtSetMappedWhenManaged(wpar2,True);
}

/*
 *this procedure fills the COMP window
 *with all her managed children
 */

void create_comp()

{
     Widget complist[4];

     /*
      * create wcomp1 commandwidget
      */
     cbwcomp1[0].callback=(XtArgVal) pcbwcomp1;
     wcomp1=XtCreateWidget("Wcomp1",commandWidgetClass,wcomp,argwcomp1,XtNumber(argwcomp
     /*
      * create wcomp2 commandwidget
      */
     cbwcomp2[0].callback=(XtArgVal) pcbwcomp2;
     wcomp2=XtCreateWidget("Wcomp2",commandWidgetClass,wcomp,argwcomp2,XtNumber(argwcomp
     /*
      * create wcomp3 commandwidget
      */
     cbwcomp3[0].callback=(XtArgVal) pcbwcomp3;
     wcomp3=XtCreateWidget("Wcomp3",commandWidgetClass,wcomp,argwcomp3,XtNumber(argwcomp
     /*
      * create wcomp4 commandwidget
      */
     cbwcomp4[0].callback=(XtArgVal) pcbwcomp4;
     wcomp4=XtCreateWidget("Wcomp4",commandWidgetClass,wcomp,argwcomp4,XtNumber(argwcomp
     /*
      *manage created widgets
      */
     complist[0]=wcomp1;
     complist[1]=wcomp2;
     complist[2]=wcomp3;
     complist[3]=wcomp4;
     XtManageChildren(complist,XtNumber(complist));
```

```
}

/*
 *this procedure  fills the PAR1  subwindow
 * with all her managed children
 */

void create_par1()

{
     Widget par1list[1];

     /*
      * create wpar1_1 commandwidget
      */
     cbwpar1_1[0].callback=(XtArgVal) pcbwpar1_1;
     wpar1_1=XtCreateWidget("Wpar1_1",commandWidgetClass,wpar1,argwpar1_1,XtNumber(argwp
     /*
      * manage created widgets
      */
     par1list[0]=wpar1_1;
     XtManageChildren(par1list,XtNumber(par1list));
}

/*
 *this procedure  fills the PAR2 window
 *with all her managed children
 */

void create_par2()

{
     Widget par2list[6];

     /*
      * create wpar1_1 dialog widget
      */
     argwpar2_1[1].value=(XtArgVal) parameter1;
     wpar2_1=XtCreateWidget("Wpar2_1",dialogWidgetClass,wpar2,argwpar2_1,XtNumber(argwpar
     /*
      * create wpar1_2 dialog widget
      */
     argwpar2_2[1].value=(XtArgVal) parameter2;
     wpar2_2=XtCreateWidget("Wpar2_2",dialogWidgetClass,wpar2,argwpar2_2,XtNumber(argwpar
     /*
      * create wpar1_3 dialog widget
      */
     argwpar2_3[1].value=(XtArgVal) parameter3;
     wpar2_3=XtCreateWidget("Wpar2_3",dialogWidgetClass,wpar2,argwpar2_3,XtNumber(argwpar
     /*
      * create wpar1_4 dialog widget
      */
     argwpar2_4[1].value=(XtArgVal) parameter4;
     wpar2_4=XtCreateWidget("Wpar2_4",dialogWidgetClass,wpar2,argwpar2_4,XtNumber(argwpar
     /*
      * create wpar1_5 dialog widget
      */
     argwpar2_5[1].value=(XtArgVal) parameter5;
     wpar2_5=XtCreateWidget("Wpar2_5",dialogWidgetClass,wpar2,argwpar2_5,XtNumber(argwpar
     /*
```

```
         * create wpar1_6 dialog widget
         */
        argwpar2_6[1].value=(XtArgVal) parameter6;
        wpar2_6=XtCreateWidget("Wpar2_6",dialogWidgetClass,wpar2,argwpar2_6,XtNumber(argwpa
        /*
         * manage created windows
         */
        par2list[0]=wpar2_1;
        par2list[1]=wpar2_2;
        par2list[2]=wpar2_3;
        par2list[3]=wpar2_4;
        par2list[4]=wpar2_5;
        par2list[5]=wpar2_6;
        XtManageChildren(par2list,XtNumber(par2list));

}

/*
 *this procedure fills the HID1 window
 *with all her managed children
 */

void create_hid1()

{
        int i;
        Widget hid1list[6];

        /*
         *initialize hidname array
         */
        for(i=0;i<10;i++)
                strcpy(hidname[i],"");
        /*
         * create whidc command widget
         */
        cbwhidc[0].callback=(XtArgVal) pcbwhidc;
        whidc=XtCreateWidget("Whidc",commandWidgetClass,whid1,argwhidc,XtNumber(argwhidc));
        /*
         * create whidl command widget
         */
        cbwhidl[0].callback=(XtArgVal) pcbwhidl;
        whidl=XtCreateWidget("Whidl",commandWidgetClass,whid1,argwhidl,XtNumber(argwhidl));
        /*
         * create whidn[0] dialog widget
         */
        argwhidn1[1].value=(XtArgVal) hidname[0];
        whidn[0]=XtCreateWidget("Whidn[0]",dialogWidgetClass,whid1,argwhidn1,XtNumber(argwhi
        /*
         * create whidn[1] dialog widget
         */
        argwhidn2[1].value=(XtArgVal) hidname[1];
        whidn[1]=XtCreateWidget("Whidn[1]",dialogWidgetClass,whid1,argwhidn2,XtNumber(argwhi
        /*
         * create whidn[2] dialog widget
         */
        argwhidn3[1].value=(XtArgVal) hidname[2];
        whidn[2]=XtCreateWidget("Whidn[2]",dialogWidgetClass,whid1,argwhidn3,XtNumber(argwhi
        /*
         * create whidn[3] dialog widget
```

```
      */
     argwhidn4[1].value=(XtArgVal) hidname[3];
     whidn[3]=XtCreateWidget("Whidn[3]",dialogWidgetClass,whid1,argwhidn4,XtNumber(argwh
     /*
      * manage whid1 children
      */
     hid1list[0]=whidc;
     hid1list[1]=whidl;
     hid1list[2]=whidn[0];
     hid1list[3]=whidn[1];
     hid1list[4]=whidn[2];
     hid1list[5]=whidn[3];
     XtManageChildren(hid1list,XtNumber(hid1list));
}

/*
 *this procedure fills the HID2 window
 *with all her managed children
 */

void create_hid2()

{
     Widget hid2list[6];

     /*
      * create whidn[4] dialog widget
      */
     argwhidn5[1].value=(XtArgVal) hidname[4];
     whidn[4]=XtCreateWidget("Whidn[4]",dialogWidgetClass,whid2,argwhidn5,XtNumber(argwhi
     /*
      * create whidn[5] dialog widget
      */
     argwhidn6[1].value=(XtArgVal) hidname[5];
     whidn[5]=XtCreateWidget("Whidn[5]",dialogWidgetClass,whid2,argwhidn6,XtNumber(argwhi
     /*
      * create whidn[6] dialog widget
      */
     argwhidn7[1].value=(XtArgVal) hidname[6];
     whidn[6]=XtCreateWidget("Whidn[6]",dialogWidgetClass,whid2,argwhidn7,XtNumber(argwhi
     /*
      * create whidn[7] dialog widget
      */
     argwhidn8[1].value=(XtArgVal) hidname[7];
     whidn[7]=XtCreateWidget("Whidn[7]",dialogWidgetClass,whid2,argwhidn8,XtNumber(argwhi
     /*
      * create whidn[8] dialog widget
      */
     argwhidn9[1].value=(XtArgVal) hidname[8];
     whidn[8]=XtCreateWidget("Whidn[8]",dialogWidgetClass,whid2,argwhidn9,XtNumber(argwhi
     /*
      * create whidn[9] dialog widget
      */
     argwhidn10[1].value=(XtArgVal) hidname[9];
     whidn[9]=XtCreateWidget("Whidn[9]",dialogWidgetClass,whid2,argwhidn10,XtNumber(argwh
     /*
      * manage whid2 children
      */
     hid2list[0]=whidn[4];
     hid2list[1]=whidn[5];
```

```
      hid2list[2]=whidn[6];
      hid2list[3]=whidn[7];
      hid2list[4]=whidn[8];
      hid2list[5]=whidn[9];
      XtManageChildren(hid2list,XtNumber(hid2list));
}

/*
 *open the display server
 *and maps the parameters selection menu Xparam
 */

main(argc,argv)
  int argc;
  char **argv;
{


   /*
    *Create the Widget  that supports all the application
    */
   topparam=XtInitialize(argv[0],"XLabel",NULL,0,&argc,argv);

   create_formp();

   create_comp();

   create_par1();

   create_par2();

   create_hid1();

   create_hid2();


   /*
    *Create the windows and set their attributes
    *according to the Widget data
    */
   XtRealizeWidget(topparam);

   /*
    *Process the events
    */
   XtMainLoop();
}
```