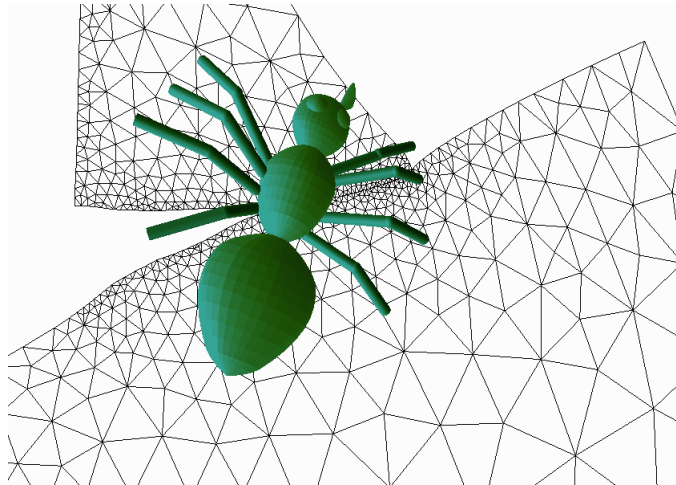

SPIDER

USER'S MANUAL

Stand Alone, Window Based, 3D Finite Element Postprocessor

Version 2.0



Prepared by:

Prof. Victor Saouma
Department of Civil Engineering,
University of Colorado, Boulder
Boulder, CO 80309-0428

Under Contract from:

Tokyo Electric Power Service Company
3-3-3 Higashiueno, Taito-ku, Tokyo 110-0015

Electric Power Research Institute
3412 Hillview Avenue
Palo Alto, California 94304

June 30, 2020

DISCLAIMER OF WARRANTIES AND LIMITATION OF LIABILITIES

THIS REPORT WAS PREPARED BY VICTOR SAOUMA AS AN ACCOUNT OF WORK SPONSORED OR COSPONSORED BY THE ELECTRIC POWER RESEARCH INSTITUTE, INC. (EPRI) AND THE TOKYO ELECTRIC POWER SERVICE COMPANY (TEPSO). NEITHER EPRI, TEPSO, OR SAOUMA NOR ANY PERSON ACTING ON BEHALF OF ANY OF THEM:

(A) MAKES ANY WARRANTY OR REPRESENTATION WHATSOEVER, EXPRESS OR IMPLIED, (I) WITH RESPECT TO THE USE OF ANY INFORMATION, APPARATUS, METHOD, PROCESS OR SIMILAR ITEM DISCLOSED IN THIS REPORT, INCLUDING MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, OR (II) THAT SUCH USE DOES NOT INFRINGE ON OR INTERFERE WITH PRIVATELY OWNED RIGHTS, INCLUDING ANY PARTY'S INTELLECTUAL PROPERTY, OR (III) THAT THIS REPORT IS SUITABLE TO ANY PARTICULAR USER'S CIRCUMSTANCES; OR

(B) ASSUMES RESPONSIBILITY FOR ANY DAMAGES OR OTHER LIABILITIES WHATSOEVER (INCLUDING ANY CONSEQUENTIAL DAMAGES, EVEN IF EPRI, TEPSO OR THEIR REPRESENTATIVES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES) RESULTING FROM YOUR SELECTION OR USE OF THIS REPORT OR ANY INFORMATION, APPARATUS, METHOD, PROCESS OR SIMILAR ITEM DISCLOSED IN THIS REPORT.

Contents

1	Post-Processor .pst Files	11
1.1	Toolbar	11
1.2	File Operation	12
1.2.1	Open File	12
1.2.2	Copy to Clipboard	12
1.2.3	Export	12
1.3	View	13
1.3.1	Regular Plot	13
1.3.2	Vector Plot	17
1.3.3	Contour Plot	18
1.3.4	Principal Plot	21
1.3.5	Carpet Plot	22
1.3.6	Surface Plot	23
1.3.7	Shrink Plot	26
1.3.8	Smeared Crack Opening	26
1.3.9	Reinforcing Steel Stresses/Strains	27
1.3.10	Vertex Info/Mesh Plot	27
1.3.10.1	Nodal Values	27
1.3.10.1.1	Full Nodal Information	29
1.3.10.1.2	Value vs Increments	29
1.3.10.2	Values Along a Line	29
1.3.10.3	Values On Contour Plots	32
1.3.11	Dynamics	32
1.3.12	Deconvolution	33
1.3.13	X-Y Plot	34
1.3.14	XYZ-V Plot; 3D Cracks-Joints	35
1.3.15	Show Title	37
1.3.16	Focus View	38
1.3.17	Reset Camera	38
1.3.18	Clear Points of Interest	38
1.3.19	Toolbar	38
1.3.20	Statusbar	38
1.4	Options	38
1.4.1	Increments	38
1.4.2	Groups	39
1.4.3	Settings	39
1.4.4	Cut Mesh	40

1.4.5	Split Mesh	40
1.4.6	Lighting	42
1.4.7	Separate Group	43
2	Eigenvalue .eig Visualizer	45
3	Real Time .rtv Viewer	47
A	Safety Factors	53
A.1	Mohr-Coulomb	53
A.2	Von-Mises	53
B	FFT, Transfer Functions, and Deconvolution	55
B.1	Fourrier Transform	55
B.2	Transfer Function	56
B.3	Deconvolution	57
B.3.1	1-D	57
B.3.2	3-D	58
B.3.2.1	Simplification	58
C	System Implementation	59
C.1	Program structure	59
C.1.1	The mesh variables	59
C.1.2	The winged and radial edge structures	59
C.1.3	Other structures	60
C.2	General program structure	60
D	.pst Post Data file Format	61
D.1	Writing Merlin Subroutine	61
D.2	Introduction	64
D.2.1	Post File Structure	65
D.3	File Header Block	65
D.3.1	Title	66
D.3.2	Stamp Definition	66
D.3.3	Post Variable List	66
D.3.4	Block Separator	67
D.4	Incremental Data	67
D.4.1	Solution Status Parameters	68
D.4.2	Finite Element Data	68
D.4.2.1	Mesh Size Parameters	68
D.4.2.2	Nodal Coordinates	69
D.4.2.3	Element Connectivity	69
D.4.2.4	Finite Element Application Data	70
D.4.3	Nodal Post Variables	74
D.4.4	Block Separator	74
E	.eig Post Data file Format	79
F	.rtv Post Data file Format	81

List of Figures

1.1	Spider's Toolbar	11
1.2	Open File	12
1.3	Exporting Files	13
1.4	Control for Regular Mesh	13
1.5	Regular Mesh	14
1.6	Regular Mesh; Hidden Line Removed	15
1.7	Regular Mesh; Mesh Outline	15
1.8	Filled with Mesh	16
1.9	Regular Mesh; Node & Element Numbering	16
1.10	Display of Deformed Mesh for Multiple Increments	17
1.11	Control for Vector Plots	17
1.12	Vector Plot	18
1.13	Control for Contour Plots	19
1.14	Contour Plot	20
1.15	Contour Plot; Separate Groups	20
1.16	Control for Principal Values Plots	21
1.17	Principal Stresses Plot	22
1.18	Control for Carpet Plots [2D]	22
1.19	Carpet Plot	24
1.20	Control for Surface Plots [2D]	24
1.21	Surface mesh	26
1.22	Control for Shrink Plots	26
1.23	Shrink mesh	27
1.24	Display of Smeared Crack	28
1.25	Vertex information	28
1.26	Sample of Nodal Values Displayed inside Notepad	30
1.27	Plot Options	31
1.28	Sample of Gnuplot Generated Stacked Plot	31
1.29	Plotted Data Saved in an Excel-alike Grid	31
1.30	Plot of Selected Scalar Between Two User-Selected Nodes.	32
1.31	Nodal Value Displayed on top of Contour Line.	33
1.32	Deconvolution of Seismic Records	34
1.33	XY Plot From Finite Element Analysis Program	35
1.34	Control Panel for the Display of Surface Plots Associated with Cracks/Joints . .	36
1.35	Spider Display When user Selects $xyz - v$ data Set	36
1.36	Example of $xyz - v$ 3D Plot of two Data Sets	37
1.37	Plot Title, without and with Labels	37

1.38	ToolBar	38
1.39	Statusbar	38
1.40	Option: Increments	38
1.41	Option: Groups	39
1.42	Option: Setting	39
1.43	Factor of Safety Parameter Setting	40
1.44	Option: Cut	41
1.45	Example of Cut Mesh	41
1.46	Option: Split	42
1.47	Example of Split Mesh	42
1.48	Option: Lighting	43
1.49	Regular Mesh; Separate Groups (GUI and Effect)	43
1.50	Regular Mesh; Selected Groups	44
2.1	Eigenvalue Control	45
2.2	Example of eigenmode Viewing	46
3.1	Real Time Control	48
3.2	Example of Real Time Viewing; Full Display	49
3.3	Example of Real Time Viewing; Partial Display	50
3.4	Example of Real Time Viewing Accelerogram Plots	51
A.1	Safety Factor for Cohesive Materials	54
B.1	Deconvolution Graphical User Interface	56
B.2	Time Frequency Domains	56
B.3	Deconvolution Definition	57
D.1	Elements Supported by Spider	71

List of Tables

D.1	Element Types Supported by Spider	70
D.2	File Header Block	74
D.3	Post Variable List	75
D.4	Solution Status Parameter	76
D.5	Mesh Size Parameter	76
D.6	Nodal Coordinates	76
D.7	Element Connectivity	76
D.8	Nodal Post Variables	77

SUMMARY

Spider is a general purpose 3D post-processor for static and dynamic nonlinear finite element analysis results¹.

Spider is an OpenGL implementation under Windows. There is no Unix implementation yet.

Spider can read post-data of any (properly written) finite element analysis program, as long as it includes: nodal coordinates, element connectivities, and nodal characteristics (defined as scalar, vector or tensors of order two). In addition the Spider can display x-y or x-y-z plots either coming from the finite element analysis (through GnuPlot), or internally generated. In addition Spider can compute the FFT of a data set, resultant force and moment if stresses along a line are being plotted.

Spider display regular meshes, shrink plots, vector and principal values plots (it will internally compute the eigenvalues/eigenmodes of the order two tensors), contour, carpet, and surface plots.

Three dimensional meshes can be sliced and provide two dimensional displays of the interior. Finally, and in the context of a nonlinear analysis of concrete structures, disks can display the smeared cracks.

Spider can also handle eigenvalue analysis results through the display of animated eigenmodes, and the display of their corresponding eigenfrequencies.

Finally, Spider can also display in real time (i.e. while an analysis is running) results of a dynamic or nonlinear analysis. For dynamic analysis, accelerograms of selected nodes can be monitored along with the corresponding deformed shapes. For nonlinear static analysis, deformation in real time can be monitored. This feature of Spider is particularly useful for monitoring dynamic analysis which is computationally intensive.

Spider has a mouse oriented, graphical user interface, which makes the program easy and intuitively to use. Hence, there is not a command prompt and no directives to memorize.

The Spider input files are relatively straightforward to define, and can be read either as binary or ASCII files. Since Spider understands various data types and reads the labels used in the menus along with the post data from the post file, the type of analysis a finite element code performs does not affect Spider. The menus will be displayed with proper labels, and the the plots will visualize the data in the formats described below. Hence Spider is not limited (or tied) to stress analysis.

This document is broken in three chapters:

.pst File definition for regular finite element analysis.

.rtv File definition for Real Time View of a lengthy dynamic analysis. In this case display is limited to deformation versus time, and accelerograms.

.eig File definition for the display of results of an eigenvalue analysis.

The format of each of those files is separately described in the appendix, in which an overview of the system implementation is also described.

About 20% of Spider's development can be traced back to an initial research grant from the Electric Power Research Institute (EPRI), and 80% to a research grant from the Tokyo Electric

¹Initially, Spider was developed as the post processor for the analysis program MERLIN, but has since been expanded to be usable with any finite element analysis program.

Power Service Company (TEPSCO). In both of them, Prof. Victor Saouma was the sole and principal investigator.

Spider was coded by Dr. G. Haussmann (based on an initial development by Mr. J. Hermanrud) and extensively tested by the research team of Prof. Saouma.

Chapter 1

Post-Processor .pst Files


























This chapter describes Spider's functionality in viewing and displaying results of a finite element analysis. It is assumed that the user has loaded a .pst file (which format is described in Appendix D).

1.1 Toolbar

Spider's main tool bar is shown in Fig. 1.1 The toolbar is composed of five parts



Figure 1.1: Spider's Toolbar

File	View
 Open File.	 Display regular mesh.
 Copy Screen display.	 Display vectors.
 Export display in .pdf or .emf format.	 Display contour lines.
Settings	 Display principal values.
 Reset view to original display.	 Display carpet plot.
 Pick a node.	 Display surface plot.
 Display nodal data.	 Display shrink plot.
 Clear markers from the main display.	Control
 Adjust center of zoom.	 Control regular display.
 Dynamic control.	 Control vectors display.
 Program setting.	 Control contour lines display.
Help	 Control principal values display.
 Help	 Control carpet plot display.
	 Control surface plot display.
	 Control shrink plot display.

1.2 File Operation

1.2.1 Open File

This allows the user to load a spider files (one with an `.pst`, `.eig` or `.rtv` file extension), or to export a graphical file.

Opening a file  will allow the user to select the file to be loaded by Spider Note, that

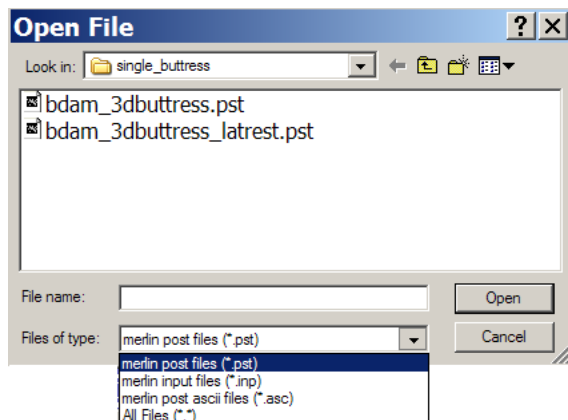



Figure 1.2: Open File

whereas by default Spider loads a binary `.pst`, `eig`, `rtv` file, it can also read their ASCII equivalent. In addition, Merlin can recognize Merlin input data file and display the mesh.

When 3D `.pst` file are loaded for the first time, Spider builds the winged and radial edge structures (Section C.1.2). This computationally intensive task eliminates internal nodes, edges, and surfaces, before the mesh can be properly displayed.

Hence, to accelerate subsequent viewing of the `.pst` file, Spider stores this data structure as a `.pst-radial` file.

1.2.2 Copy to Clipboard

 enables the user to copy the current screen display into the clipboard as an extended metafile file `.emf`. This (rather large) file, can then be easily imported by applications such as Word, Power-Point, or Adobe-Illustrator.

Note, this copy mode is much better than simply “printing the screen” content, as not only does it automatically take care of background color, but it also create a *scalable* image inside the intended target document.

For 3D displays, it is advisable to switch the display mode to **Mesh** rather than the (defaulted) setting of **Mesh, with hidden line**.

1.2.3 Export

 Exports current displays into either one of the following formats, as Fig.:

1. Encapsulated postscript file, `.pdf` (best for \LaTeX).
2. Window Enhanced Metafile, `.emf`, (best for Word).

3. Bitmap .bmp.
4. Graphics Interchange Format, Gif
5. Joint Photographic Experts Group, .jpg, with quality control

1.3. User can select path and file name.

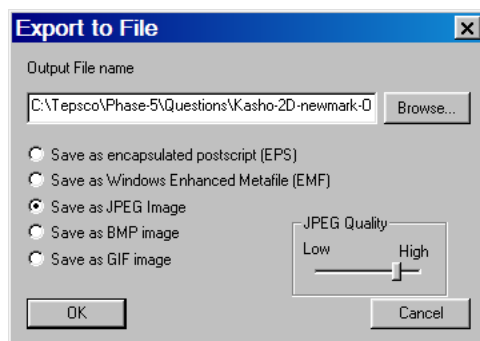


Figure 1.3: Exporting Files

1.3 View

View enables the user to describe one or more different type of displays. Hence, a vector plot can be superimposed on a contour plot. Note that through the slice/cut feature described below, part of the mesh can be viewed with one type of display and one or more other partitions with another type.

1.3.1 Regular Plot

Control of the regular plot is shown in Fig. 1.4.

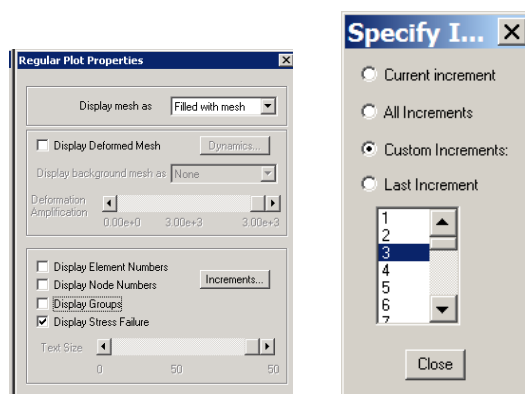


Figure 1.4: Control for Regular Mesh



provides control for the display of the regular plots.

Display mesh as in any one of the following format

Mesh displays the entire mesh, Fig. 1.5.

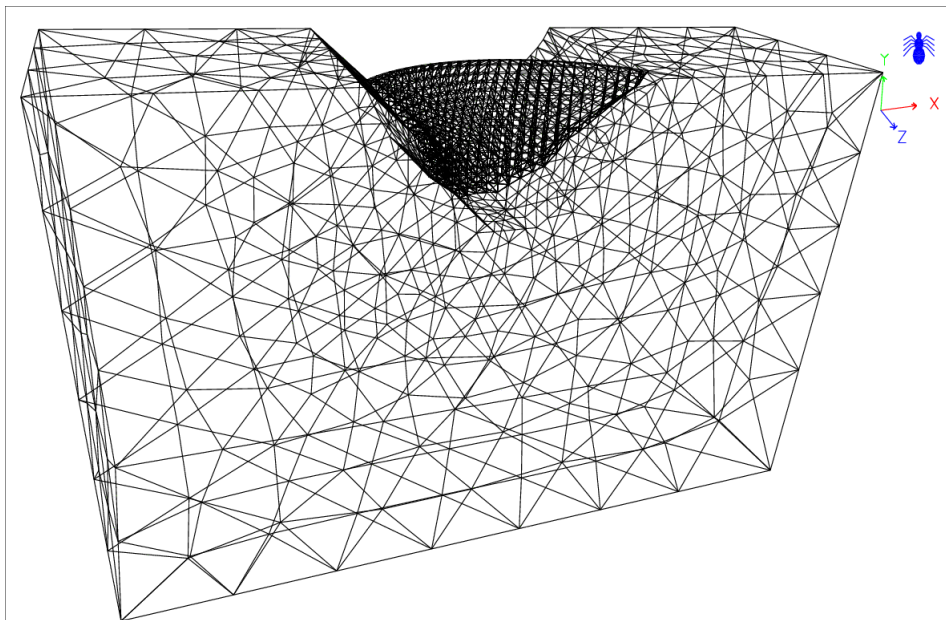


Figure 1.5: Regular Mesh

Mesh hidden line Displays 3D meshes with hidden lines removed, Fig. 1.6.

Mesh Outline displays only those lines separating different material groups, or those defining the boundary of the mesh, Fig. 1.7.

Filled Displays a color filled mesh. Color is either a single one (useful in 3D to view with proper lighting set-up), or multiple if the user asks for each material group to have its own color, Fig. 1.8.

Filled with mesh Same as above, but with the finite element mesh superimposed. shows the mesh as a wire frame,

Display Deformed Mesh allows the user to display and control the deformation of the deformed mesh. One may superimpose to the display the background mesh as None, Mesh outline, Mesh, Filled, Filled with Mesh.

Display Element, Nodes, Groups allow the user to display those numbers. If Display Group is checked, then the element number will be color coded to reflect the group to which it belongs. Note: User may have to adjust the font size (through Settings) and properly zoom in to properly visualize text display, Fig. 1.9. Note that if there are more than one node sharing the same coordinates (Master/Slaves), then all node number are displayed separated by a /.

Display Stress Failure will display the failure mode (shear, traction or combined) for the non-linear rock elements only.

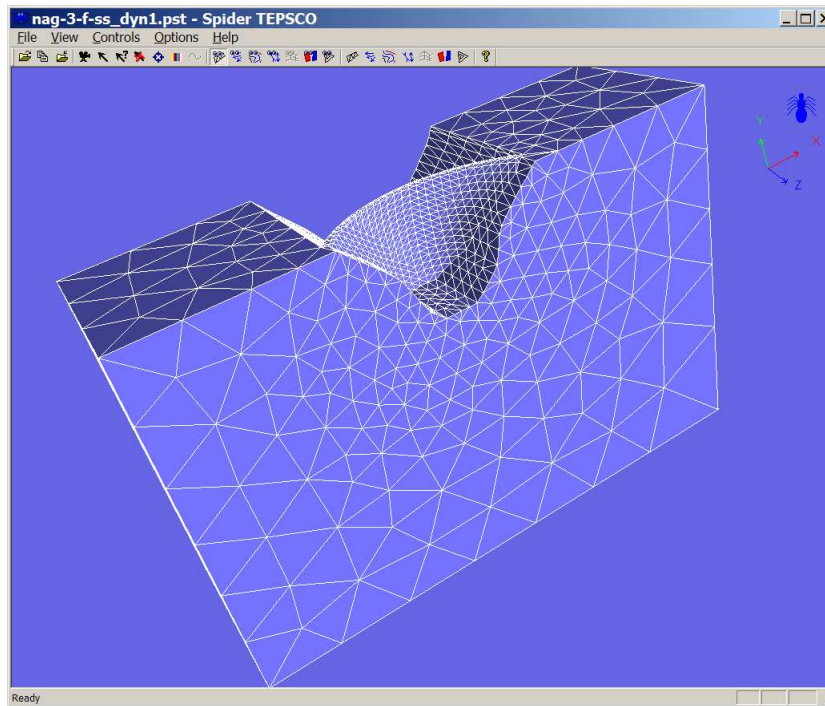


Figure 1.6: Regular Mesh; Hidden Line Removed

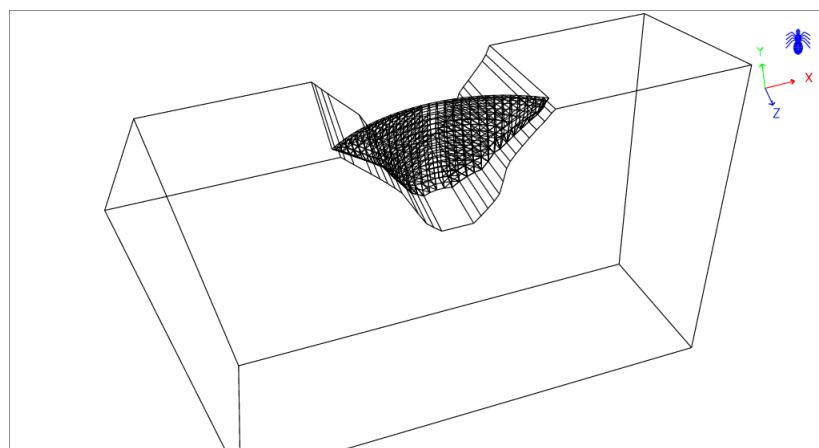


Figure 1.7: Regular Mesh; Mesh Outline

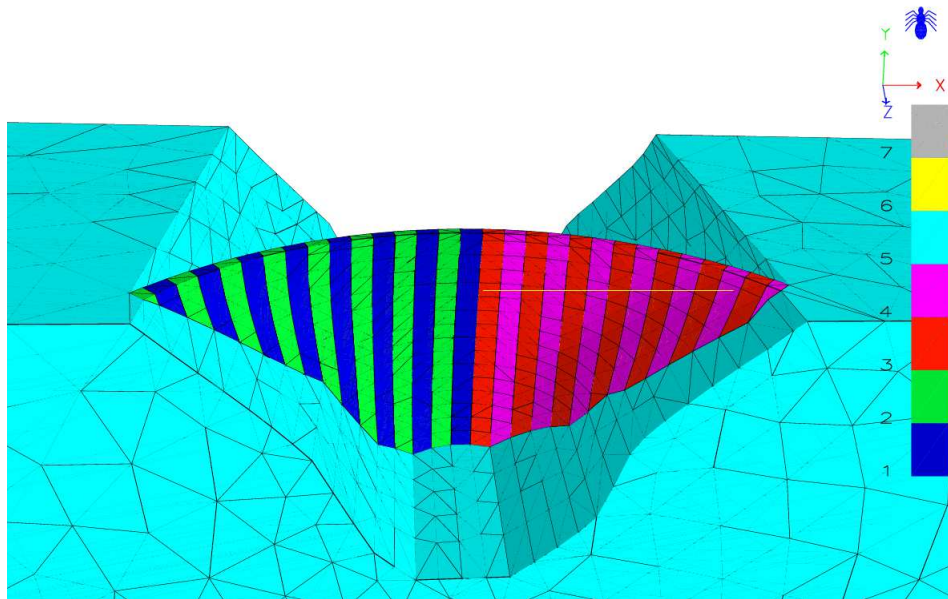


Figure 1.8: Filled with Mesh

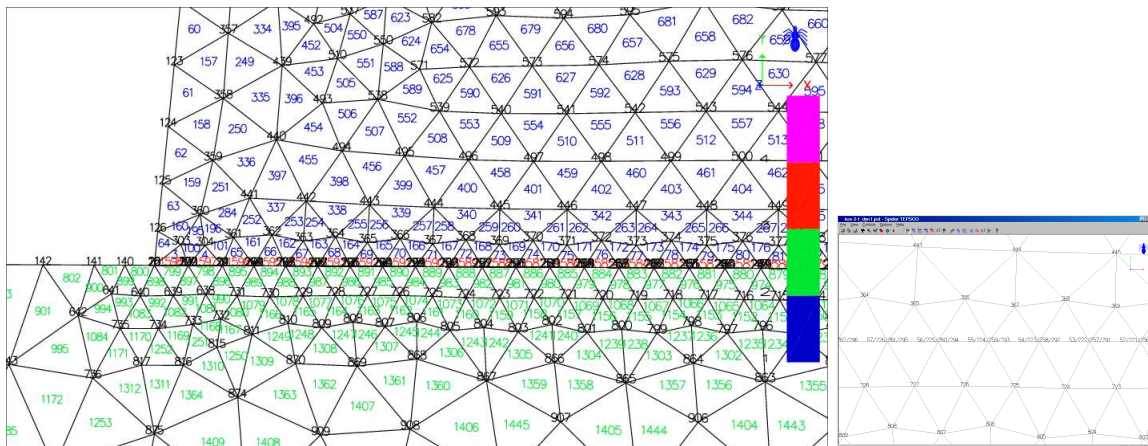


Figure 1.9: Regular Mesh; Node & Element Numbering

Increments User may want to monitor the evolution of the deformation by viewing the deformed mesh at selected or all increments, this is possible by specifying the increments to be viewed, Fig. 1.10. Note that in this case, only the mesh outline is displayed.

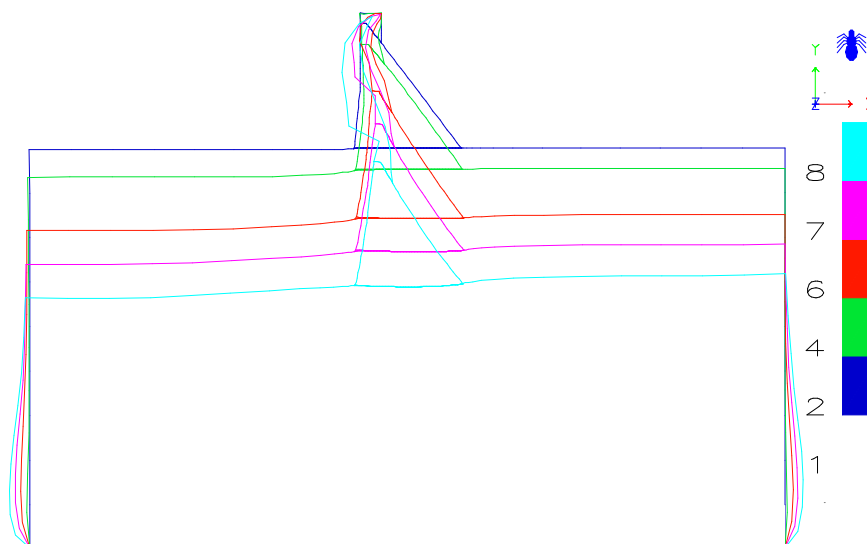



Figure 1.10: Display of Deformed Mesh for Multiple Increments

1.3.2 Vector Plot

 The vector plot shows vector format post data using vectors extending from each node in the mesh, Fig. 1.11.

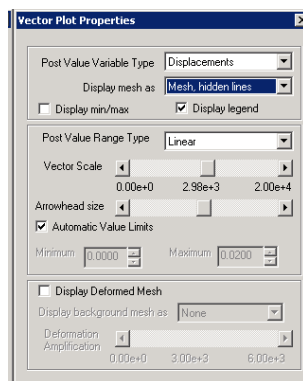


Figure 1.11: Control for Vector Plots

Post Value variable Type; Vector Plot will display all entities specified in the .pst file as tensors of order 1 or vectors. For Merlin, this includes: Velocities, Accelerations, Displacements, Applied Forces, Reactions, and Residuals. Spider will determine and plot the resultant of the 2 or 3 components.

Display mesh as Mesh outline, Mesh, Mesh hidden lines or solid filled.

Display min/max Spider will place two marker at the min/max locations, and the numerical values will be displayed in the lower right corner.

Display legend to toggle the display of the “thermometer” which color codes the magnitude of the vector length.

Post Value Range Type is by default set to linear, but for problems with strong discontinuity, the user may select a logarithmic distribution.

Vector Scale slider allows the user to set the length of the vector. Note that if the scale is set to a value too low, small vectors may not be displayed.

Arrowhead size provides control of the arrowhead to be in proportion with the rest of the graphical display.

Automatic Value limits is by default set to on. However user can overwrite the lower and upper limits to better focus on a range of values.

Display Deformed Mesh allows the user to display and control the deformation of the deformed mesh. One may superimpose to the display the background mesh as None, Mesh or Mesh Outline.

Separate Groups allows the user to “pull out” all, one, or more groups through the slider for better view.

Fig. 1.12 is an example of the generated display.

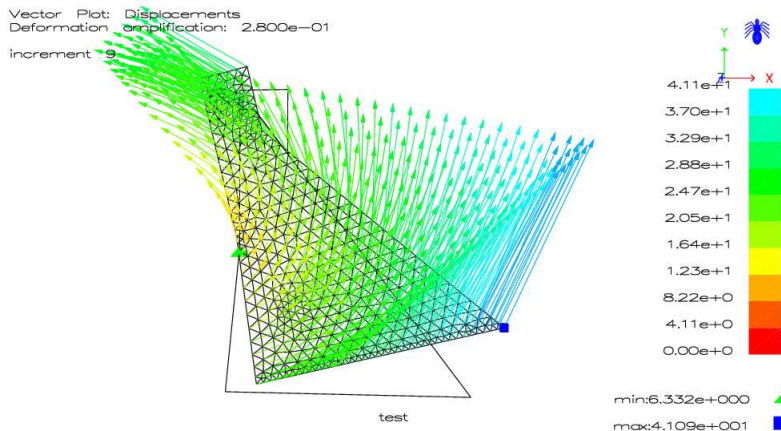


Figure 1.12: Vector Plot

1.3.3 Contour Plot



The contour plot can show any post data type, using contour lines, solid filled contour areas, or shaded contour areas, Fig. 1.13.

Post Value variable Type; Contour Plot will display any of the followings: scalars, components of vectors and of tensors, eigenvalues of tensors of order 2 (max, minimum, and intermediate principal values). In addition, Spider will internally compute the hydrostatic, volumetric, and von Mises values associated with a tensor of order 2.

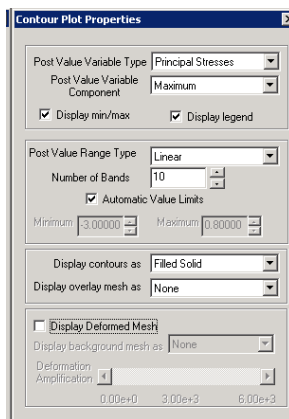


Figure 1.13: Control for Contour Plots

It should be noted that internally (through a flag in the .pst file, Spider distinguishes between strain and stress tensors).

For a Merlin input file, the following scalar quantities are displayed

Tensors of Order 1 or vectors

Velocities : $v_x, v_y, [v_z]$

Accelerations : $a_x, a_y, [a_z]$

Displacements : $u, v, [w]$

Applied Forces : $F_x, F_y, [F_z]$

Reactions : $R_x, R_y, [R_z]$

Residual : $R_x, R_y, [R_z]$

Tensors of Order 2 Strains : $\varepsilon_{xx}, \varepsilon_{yy}, [\varepsilon_{zz}], \varepsilon_{xy}, [\varepsilon_{xz}, \varepsilon_{yz}]$

Stresses : $\sigma_{xx}, \sigma_{yy}, [\sigma_{zz}], \sigma_{xy}, [\sigma_{xz}, \sigma_{yz}]$

Scalar Vector length without components:

Displacement vector length

Applied forces vector length

Reactions vector length

Residuals vector length

Eigenvalues of Tensors of Order 2 which are internally determined:

Principal strains : Maximum, minimum, [intermediary]

Principal stresses : Maximum, minimum, [intermediary]

Volumetric strain $\varepsilon_{Vol} = \frac{\varepsilon_{11} + \varepsilon_{22} + \varepsilon_{33}}{3}$

Hydrostatic stress $\sigma_{Hyl} = \frac{\sigma_{11} + \sigma_{22} + \sigma_{33}}{3}$

Von Mises Stress $\sqrt{\frac{(\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_3)^2 + (\sigma_3 - \sigma_1)^2}{2}}$

Display min/max Spider will place two marker at the min/max locations, and the numerical values will be displayed in the lower right corner.

Display legend to toggle the display of the “thermometer” which color codes for the contour lines.

Post Value Range Type is Wire frame, solid filled or shaded. It is recommended to use shaded. the user may select a logarithmic distribution.

Number of Bands Defaulted to 10, maximum is 29.

Automatic Value limits is by default set to on. However user can overwrite the lower and upper limits to better focus on a range of values.

Display Deformed Mesh allows the user to display and control the deformation of the deformed mesh. One may superimpose to the display the background mesh as None, Mesh or Mesh Outline.

Separate Groups allows the user to “pull out” all, one, or more groups through the slider for better view.

Examples of contour plots are shown in Fig. 1.14 and 1.15.

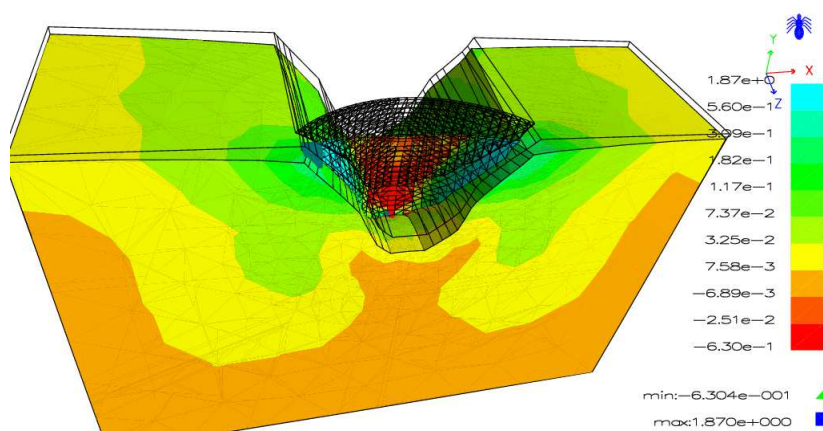


Figure 1.14: Contour Plot

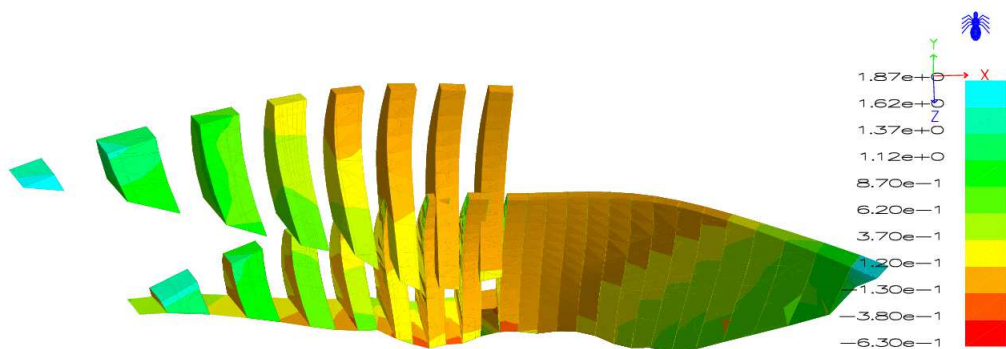



Figure 1.15: Contour Plot; Separate Groups

1.3.4 Principal Plot

 The principal plot displays the eigenvectors of tensors 2, defined in the `.pst` file, and internally computed by Spider (which differentiates, through a flag in the `.pst` file, between engineering and tensorial stains), Fig. 1.16.

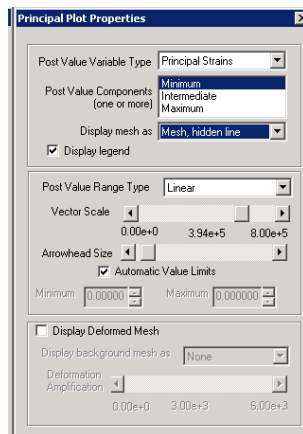


Figure 1.16: Control for Principal Values Plots

Post Value variable Type; is restricted to the eigenvectors associated with the tensors of order 2 given to Spider. For Merlin files, this corresponds to Principal Strains, principal Stress, and Principal AAR Strains.

Component User may select one, two or three components to be displayed simultaneously.

Display mesh as Mesh outline, Mesh, Mesh hidden lines or solid filled.

Display legend to toggle the display of the “thermometer” which color codes the magnitude of the vector length.

Post Value Range Type is by default set to linear, but for problems with strong discontinuity, the user may select a logarithmic distribution.

Vector Scale slider allows the user to set the length of the vector. Note that if the scale is set to a value too low, small vectors may not be displayed.

Arrowhead size provides control of the arrowhead to be in proportion with the rest of the graphical display.

Automatic Value limits is by default set to on. However user can overwrite the lower and upper limits to better focus on a range of values.

Display Deformed Mesh allows the user to display and control the deformation of the deformed mesh. One may superimpose to the display the background mesh as None, Mesh or Mesh Outline.

Separate Groups allows the user to “pull out” all, one, or more groups through the slider for better view.

Fig. 1.17 is an example of the generated display.

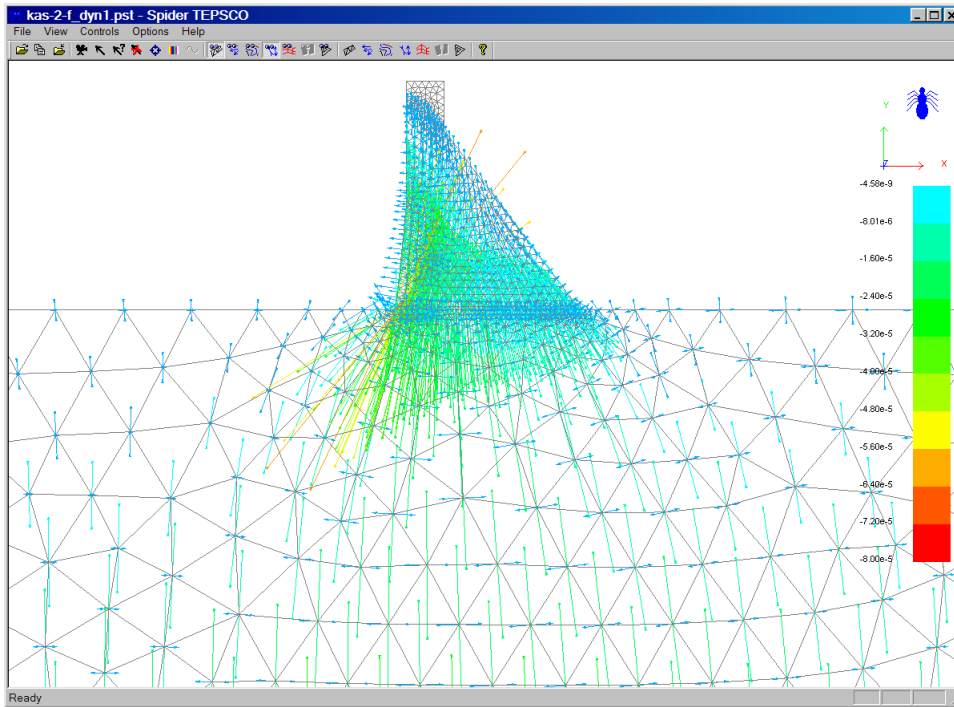



Figure 1.17: Principal Stresses Plot

1.3.5 Carpet Plot

 The carpet plot is available for two dimensional meshes only. In a carpet plot, each node is extended in the third dimension proportional to the post data value at the node. Thus the general trends and also spikes in the data is easily visible. The carpet plot can be shown as a wire frame, solid filled or a shaded object, Fig. 1.18. The carpet plot displays essentially the

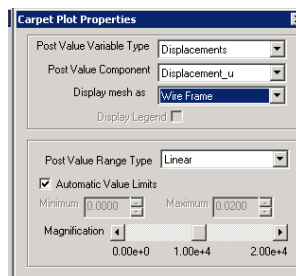


Figure 1.18: Control for Carpet Plots [2D]

same data as the contour plot:

Post Value variable Type; Contour Plot will display any of the followings: scalars, components of vectors and of tensors, eigenvalues of tensors of order 2 (max, minimum, and intermediate principal values). In addition, Spider will internally compute the hydrostatic, volumetric, and von Mises values associated with a tensor of order 2.

It should be noted that internally (through a flag in the .pst file, Spider distinguishes between strain and stress tensors).

For a Merlin input file, the following scalar quantities are displayed

Tensors of Order 1 or vectors

Velocities : $v_x, v_y, [v_z]$

Accelerations : $a_x, a_y, [a_z]$

Displacements : $u, v, [w]$

Applied Forces : $F_x, F_y, [F_z]$

Reactions : $R_x, R_y, [R_z]$

Residual : $R_x, R_y, [R_z]$

Tensors of Order 2 Strains : $\varepsilon_{xx}, \varepsilon_{yy}, [\varepsilon_{zz}], \varepsilon_{xy}, [\varepsilon_{xz}, \varepsilon_{yz}]$

Stresses : $\sigma_{xx}, \sigma_{yy}, [\sigma_{zz}], \sigma_{xy}, [\sigma_{xz}, \sigma_{yz}]$

Scalar Vector length without components:

Displacement vector length

Applied forces vector length

Reactions vector length

Residuals vector length

Eigenvalues of Tensors of Order 2 which are internally determined:

Principal strains : Maximum, minimum, [intermediary]

Principal stresses : Maximum, minimum, [intermediary]

Volumetric strain $\varepsilon_{Vol} = \frac{\varepsilon_{11} + \varepsilon_{22} + \varepsilon_{33}}{3}$

Hydrostatic stress $\sigma_{Hvd} = \frac{\sigma_{11} + \sigma_{22} + \sigma_{33}}{3}$

Von Mises Stress $\sqrt{\frac{(\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_3)^2 + (\sigma_3 - \sigma_1)^2}{2}}$

Display legend to toggle the display of the “thermometer” which color codes for the contour lines.

Post Value Range Type is Wire frame, solid filled or shaded. It is recommended to use shaded. the user may select a logarithmic distribution.

Automatic Value limits is by default set to on. However user can overwrite the lower and upper limits to better focus on a range of values.

Magnification Through this slider, the user can control the magnitude of the (artificially imposed) third dimension of the mesh.

Fig. 1.19 illustrates Carpet Plot.

1.3.6 Surface Plot



Surface plot displays contour surfaces *inside* a 3D structure, whereas contour lines are displayed on the surface of the 3D structure.

Surface plots has essentially the same capabilities as contour plots, Fig. 1.20:

Post Value variable Type; Contour Plot will display any of the followings: scalars, components of vectors and of tensors, eigenvalues of tensors of order 2 (max, minimum, and

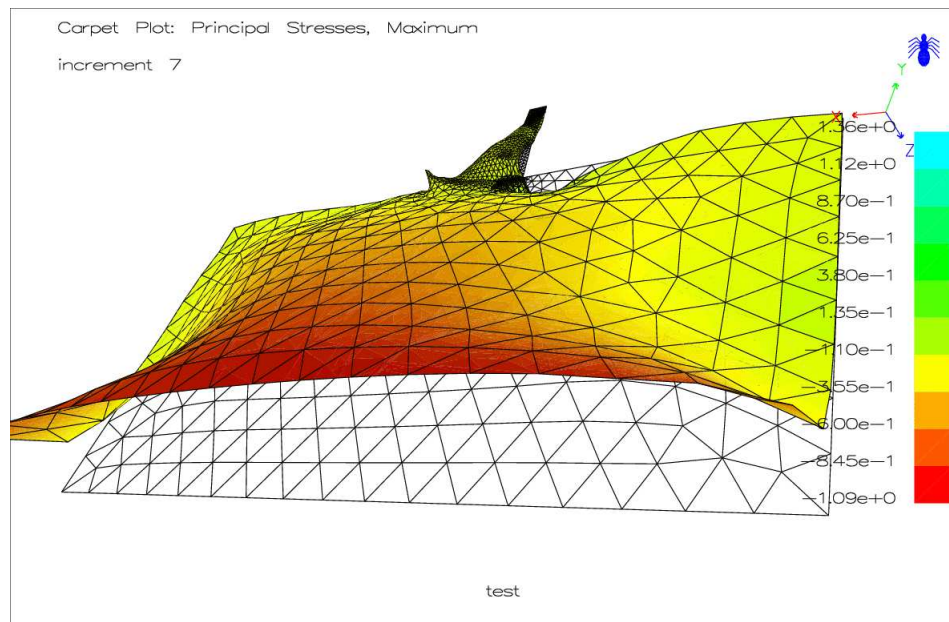


Figure 1.19: Carpet Plot

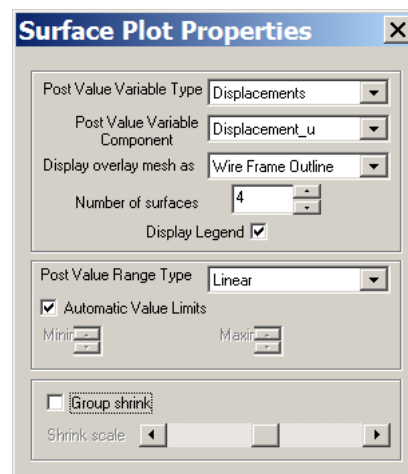


Figure 1.20: Control for Surface Plots [2D]

intermediate principal values). In addition, Spider will internally compute the hydrostatic, volumetric, and von Mises values associated with a tensor of order 2.

It should be noted that internally (through a flag in the `.pst` file, Spider distinguishes between strain and stress tensors).

For a Merlin input file, the following scalar quantities are displayed

Tensors of Order 1 or vectors

Velocities : $v_x, v_y, [v_z]$

Accelerations : $a_x, a_y, [a_z]$

Displacements : $u, v, [w]$

Applied Forces : $F_x, F_y, [F_z]$

Reactions : $R_x, R_y, [R_z]$

Residual : $R_x, R_y, [R_z]$

Tensors of Order 2 Strains : $\varepsilon_{xx}, \varepsilon_{yy}, [\varepsilon_{zz}], \varepsilon_{xy}, [\varepsilon_{xz}, \varepsilon_{yz}]$

Stresses : $\sigma_{xx}, \sigma_{yy}, [\sigma_{zz}], \sigma_{xy}, [\sigma_{xz}, \sigma_{yz}]$

Scalar Vector length without components:

Displacement vector length

Applied forces vector length

Reactions vector length

Residuals vector length

Eigenvalues of Tensors of Order 2 which are internally determined:

Principal strains : Maximum, minimum, [intermediary]

Principal stresses : Maximum, minimum, [intermediary]

Volumetric strain $\varepsilon_{Vol} = \frac{\varepsilon_{11} + \varepsilon_{22} + \varepsilon_{33}}{3}$

Hydrostatic stress $\sigma_{Hyd} = \frac{\sigma_{11} + \sigma_{22} + \sigma_{33}}{3}$

Von Mises Stress $\sqrt{\frac{(\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_3)^2 + (\sigma_3 - \sigma_1)^2}{2}}$

Number of surfaces Controls the number of internal contour surfaces.

Display overlay mesh as None, wire frame outline or wire frame.

Display legend to toggle the display of the “thermometer” which color codes for the contour lines.

Post Value Range Type Linear or logarithmic.

Automatic Value limits is by default set to on. However user can overwrite the lower and upper limits to better focus on a range of values.

An examples of surface plot is shown in Fig. 1.21.

Note: Continuity of the individual patches into a smooth surface is not always satisfied unless a very fine mesh is used.

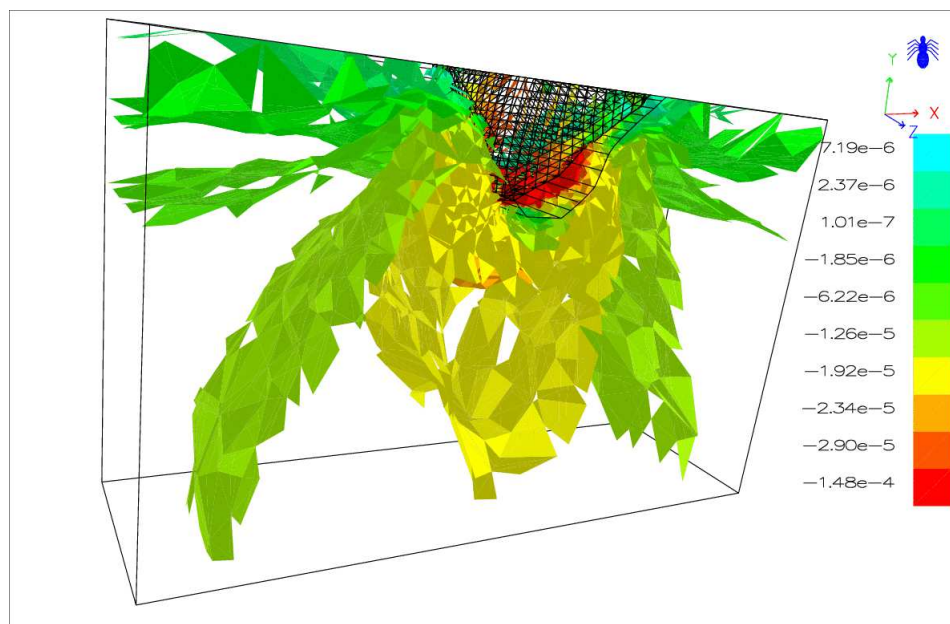



Figure 1.21: Surface mesh

1.3.7 Shrink Plot

 The shrink plot shows the mesh with each element shrunk by a factor. This is useful to see if there are holes in a mesh, or to check connectivity in the presence of interface elements.

Shrink plots, Fig. 1.22 enables the user to control the shrink factor (through a slider), whether elements are to be displayed as wireframe or as solid, and whether node and element sizes should be displayed (size controlled by a slider), Fig. 1.22:

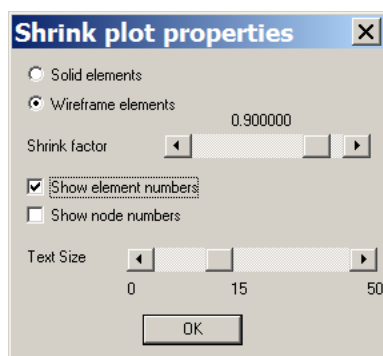


Figure 1.22: Control for Shrink Plots

An examples of shrink plot is shown in Fig. 1.23.

1.3.8 Smeared Crack Opening

Spider can also display location, orientation and opening of smeared cracks, Fig. 1.24. The data is supplied through the .pst file as a scalar quantity (crack opening), $x - y - [z]$ location,

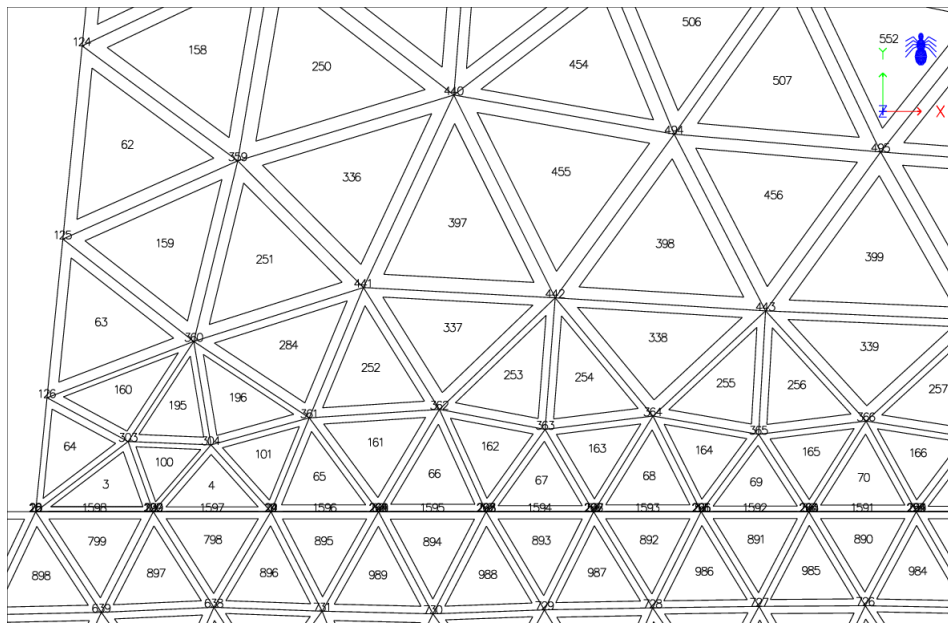


Figure 1.23: Shrink mesh

and direction of the normal to the crack.

1.3.9 Reinforcing Steel Stresses/Strains

Whereas there is no explicit commands in Spider to display reinforcement stresses, those can be generated by the finite element analysis program and passed to spider as an $x - y$ plot.


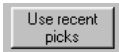
1.3.10 Vertex Info/Mesh Plot

This powerful feature enables the user to

1. Extract all known values associated with a node.
2. Plot variation of a scalar quantity in terms of increments.

1.3.10.1 Nodal Values

Spider loads the `.pst` file containing all the scalar, vectorial and tensorial (order 2) associated with each and every node. It internally computes the eigenvalues (principal) associated with the tensors of order 2, and all this data is accessible for display. If the user wants a textual display of all those values associated with a node, then:

1. Select . This will enable the pointer.
2. Click on one or (at most) two separate nodes.
3. Select **View**, and then **Vertex Info/Mesh Plot**, Fig. 1.25.
4. Select  **Use Recent Picks** (alternatively, user can directly enter the node number in the appropriate box).

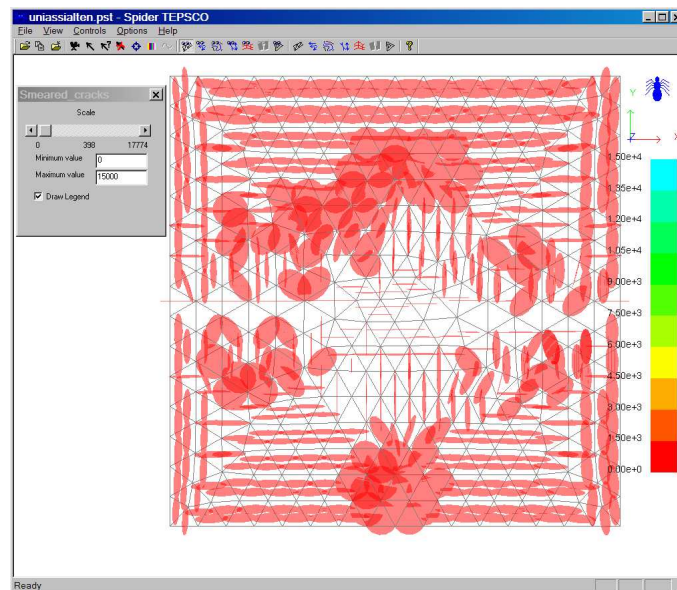


Figure 1.24: Display of Smeared Crack

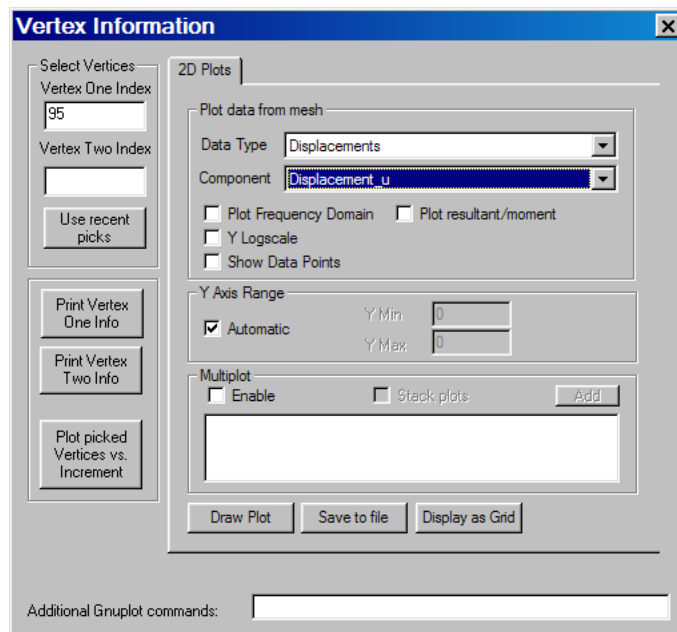
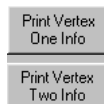


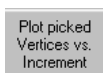
Figure 1.25: Vertex information



1.3.10.1.1 Full Nodal Information

Select **Print Vertex One Info** and/or **Print Vertex Two Info**, and then Spider will display inside an (editable) Notepad all known data associated with the respective node, Fig. 1.26.

Note if more than one node is selected during one session, new nodal information are not appended at the end of the existing file, but rather placed on the top of the file. Furthermore, the notepad file can be edited and saved on disk.



1.3.10.1.2 Value vs Increments


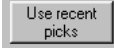
Allows the user to generate a gnuplot (x-y) for the selected scalar associated with the selected node in terms of all the increments. This is most useful in nonlinear analyses.

Furthermore, user can, Fig. 1.27.

1. Select multiple scalar values to be plotted by Enabling the Multiplot option.
2. Stack those plots, Fig. 1.28. Note that x-y coordinates can be read at the bottom left corner.
3. Save the data associated with those files into a disk ascii file.
4. Display the data as Grid (Excel format), Fig. ?? such that data can be easily exported to an .xls file (through CTRL-C).
5. Modify the Y Axis range
6. Select a Log scale for the Y axis
7. Plot the Resultant and first moment (useful to convert stresses into force)
8. Plot the FFT of the selected data
9. Request additional gnuplot options

1.3.10.2 Values Along a Line

If the nodal value distribution of a selected scalar along two arbitrary nodes is desired, then the user should

1. Select  two distinct nodes.
2. Select  **Use Recent Picks** (alternatively, user can directly enter the node number in the appropriate box)
3. Select the scalar values to be displayed along the two selected nodes.
4. Plot the curve, Fig. 1.30.

Note that user can also:

1. Select multiple scalar values to be plotted by Enabling the Multiplot option.

```

Vertex number 1 of 1: Pick node data - Mon Jul 25 10:39:29 2005

Pick data Node id: 95 Increment number: 1 X-coordinate: 0.186105
Y-coordinate: 0.817043 Z-coordinate: 0
===== Scalars =====
===== Vectors =====
Displacements:
Displacement_u:      -0.000180527 Displacement_v:
-0.0006695153 Displacements vector length      0.0006934268
-----
Applied Forces:
Force_x:              0 Force_y:
0 Applied Forces vector length      0
-----
Reactions:
Reaction_x:           -1.607872e-016 Reaction_y:
-8.777701e-016 Reactions vector length      8.923748e-016
-----
Residuals:
Residual_x:           -1.607872e-016 Residual_y:
-8.777701e-016 Residuals vector length      8.923748e-016
-----
Principal Strains Minimum vector:
vector_u:              2.993032e-005 vector_v:
-0.0002328966
-----
Principal Strains Maximum vector:
vector_u:              7.360876e-005 vector_v:
9.45971e-006
-----
Principal Stresses Minimum vector:
vector_u:              2.620028e-005 vector_v:
-7.837194e-005
-----
Principal Stresses Maximum vector:
vector_u:              0.0005629875 vector_v:
0.0001882106
-----
===== Tensors =====
Strains:
Epsilon_xx:           6.919327e-005 Epsilon_yy:
-0.0002297911 Gamma_xy:           7.813746e-005
Principal Strains:
Minimum:              -0.0002348119 Maximum:
7.421412e-005
Principal Strains Minimum vector:
vector_u:              2.993032e-005 vector_v:
-0.0002328966
Principal Strains Maximum vector:
vector_u:              7.360876e-005 vector_v:
9.45971e-006
Volumetric Strains:
Volumetric Strains:   -0.0001605978
-----
Stresses:
Sigma_xx:             0.0005256337 Sigma_yy:
-1.465462e-005 Tau_xy:             0.0002033485
Principal Stresses:
Minimum:              -8.263544e-005 Maximum:
0.0005936145
Principal Stresses Minimum vector:
vector_u:              2.620028e-005 vector_v:
-7.837194e-005
Principal Stresses Maximum vector:
vector_u:              0.0005629875 vector_v:
0.0001882106
Hydrostatic Stresses:
Hydrostatic Stresses: 0.0001703264
Von Mises Stresses:
Von Mises Stresses:   0.0006389526

```

Figure 1.26: Sample of Nodal Values Displayed inside Notepad

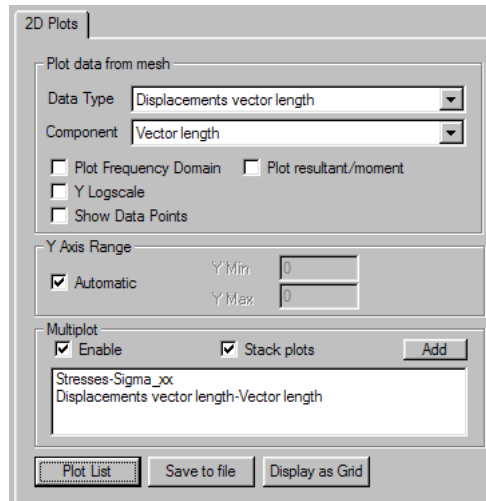


Figure 1.27: Plot Options

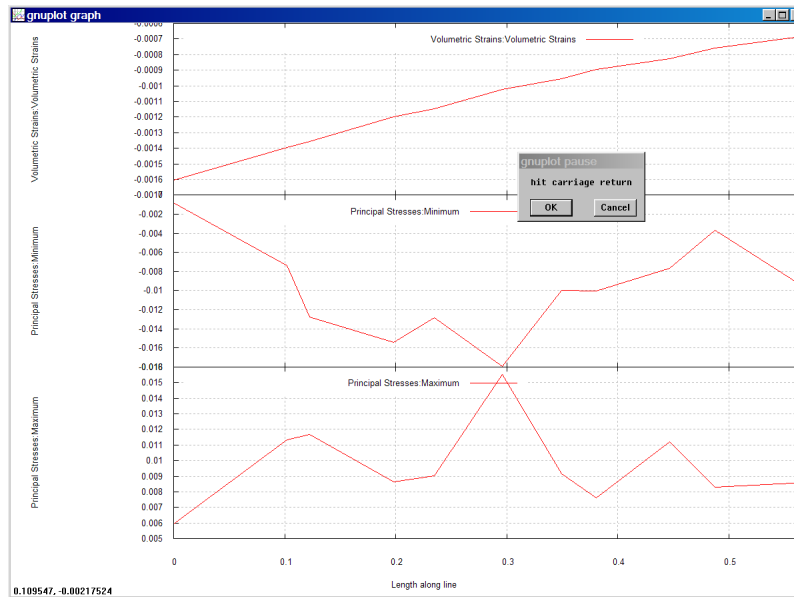


Figure 1.28: Sample of Gnuplot Generated Stacked Plot

Grid Data Display					
	d 0	d 1	d 2	d 3	d 4
v 0	0.000000	0.005936	0.000000	-0.000826	0.000000
v 1	0.000000	0.005936	0.000000	-0.000826	0.000000
v 2	0.101795	0.011329	0.101795	-0.007383	0.101795
v 3	0.122670	0.011659	0.122670	-0.012763	0.122670
v 4	0.197971	0.008643	0.197971	-0.015458	0.197971
v 5	0.234486	0.009025	0.234486	-0.012874	0.234486
v 6	0.295673	0.015500	0.295673	-0.017912	0.295673
v 7	0.349163	0.009171	0.349163	-0.009943	0.349163
v 8	0.380400	0.007597	0.380400	-0.010059	0.380400
v 9	0.446323	0.011187	0.446323	-0.007657	0.446323

Figure 1.29: Plotted Data Saved in an Excel-alike Grid

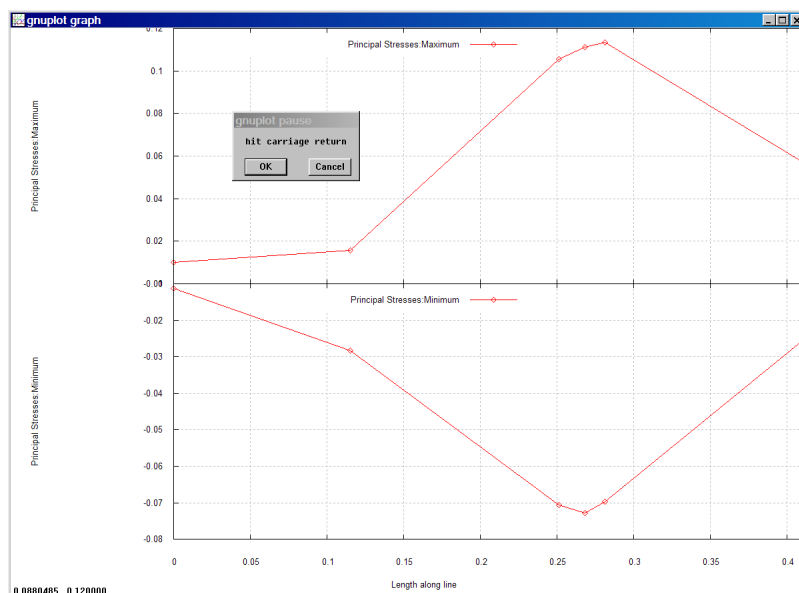




Figure 1.30: Plot of Selected Scalar Between Two User-Selected Nodes.

2. Save the data associated with those files into a disk ascii file.
3. Display the data as Grid (Excel format), Fig. ?? such that data can be easily exported to an .xls file (through CTRL-C).
4. Modify the Y Axis range.
5. Select a Log scale for the Y axis.
6. Plot the Resultant and first moment (useful to convert stresses into force).
7. Plot the FFT of the selected data
8. Request additional gnuplot options.

1.3.10.3 Values On Contour Plots

User can display on top of the contour line the node number and scalar value of a selected node through  Note:

1. If user changes the scalar value to be displayed (such as maximum instead of minimum principal stress) those values are automatically updated.
2. Selected values are displayed along the thermometer on the right.
3. Values will be cleared from the screen through selection of 

1.3.11 Dynamics

This section is associated with the display of real time data .rtv files. Please consult Chapter 3.

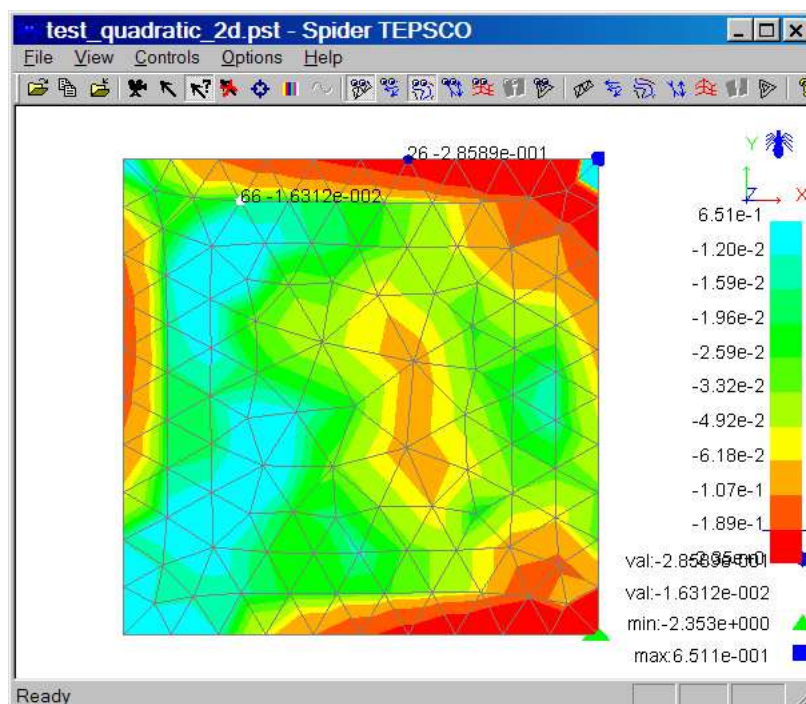


Figure 1.31: Nodal Value Displayed on top of Contour Line.

1.3.12 Deconvolution

Deconvolution of seismic record is an essential part of a “good” dynamic analysis where the rock is massless. The underpinning theory for deconvolution is described in Appendix B.

Deconvolution can be performed either for data files (.rtv) coming from Merlin, or from other external ascii files (**not yet implemented.**) To perform a deconvolution of a seismic record (with Merlin files), the user should:

1. Select the original surface seismic record.
2. Perform three (two in 2D) separate dynamic analysis. In each one of them only one component of the surface seismic record is applied at the base of the foundation. In a Merlin Analysis, user should specify **RealTimeView** to monitor the accelerations of (at least) two nodes:
 - (a) Node *I* at the base of the foundation where the input seismic record is applied.
 - (b) Node *O* (output) at the top of the foundation where the original seismic record was measured.
3. Select Deconvolution, **Data Format** Merlin, and **Dimension** 2D or 3D.
4. Select the type of data filter
 - (a) Low Pass
 - (b) High Pass
 - (c) Band Pass

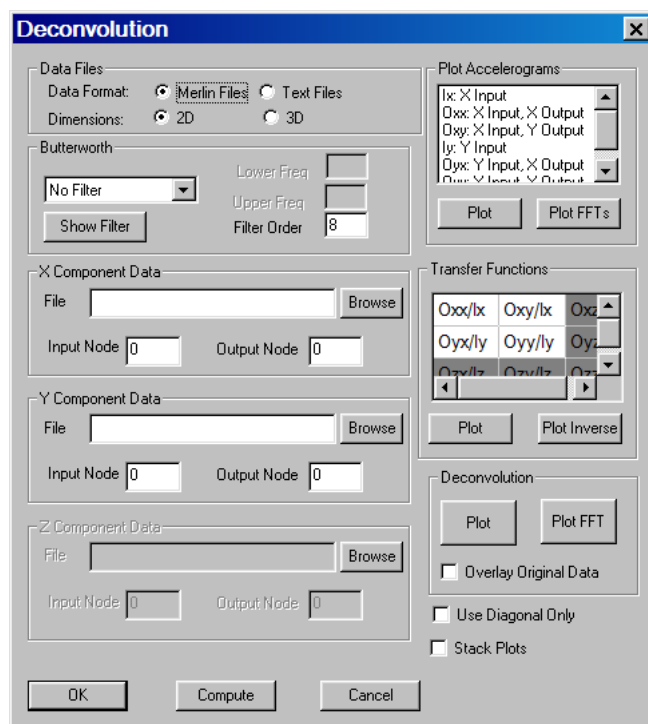


Figure 1.32: Deconvolution of Seismic Records

- (d) Band Stop
- along with the filter order.
- 5. For each component I_K (where $K = X, Y, \text{ or } Z$)
 - (a) Use **Browse** and select the **.rtv** file corresponding to the analysis in which I_K was applied.
 - (b) Select the input Node and the Output Node
- 6. Plot
 - (a) Any of the acceleration history, or their FFT.
 - (b) Any of the transfer function or their inverse.
- 7. Perform a Deconvolution.

1.3.13 X-Y Plot

The .pst file may contain Finite Element Application Data (X-Y) data to be plotted (as defined in Sect. D.4.2.4). These labelled x-y data set can be plotted in Spider, Fig. 1.33 through Gnuplot. If no Application Data are defined in the .pst file, this entry is greyed out, if not, the user can

1. Select the **Data** (label is defined by the user who wrote the .pst file), and the data set (whcih usually corresponds to the increment).

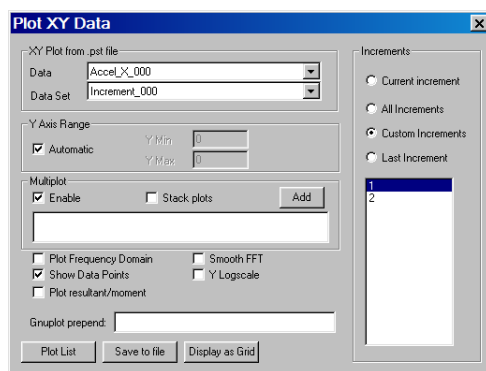


Figure 1.33: XY Plot From Finite Element Analysis Program

2. Y axis can be reset by user (as opposed to automatic determination of min. and max.).
3. Plot $X - Y$ for
 - (a) Current increment (loaded by Spider).
 - (b) All increments (one plot for each increment) superimposed or stacked.
 - (c) Custom Increment (such as increments 1, 5 and 18).
 - (d) last increment.
4. Y axis may be plotted on a Log scale,
5. FFT or smoothed FFT of the selected data set may be plotted.
6. Resultant/moment (for stresses versus length) may also be determined.
7. Display the data as Grid (Excel format), Fig. ?? such that data can be easily exported to an .xls file (through CTRL-C).
8. Additional Gnuplot commands can also be manually specified.

1.3.14 XYZ-V Plot; 3D Cracks-Joints

In 3D structures when cracks or joints are present, it is often desirable to visualize relative data such as opening, sliding, or internal pressure. Those data can be supplied by the finite element analysis code in the form of $x - y - z$ coordinate and a corresponding scalar value v . By extension, user may submit to Spider such data which are not necessarily associated with a crack.

Spider provides the capability of accessing this data set through the graphical user interface shown in Fig. 1.34.

When this option is activated, Spider will display the mesh outline with all data sets, and the data set selected by the user (10 in this case) will be highlighted, Fig. 1.35. Spider will use the distinct (and unstructured) data points and fit them into a grid with a user specified resolution (10 in this case). Actual data points used may be displayed or not. The shaded surface can be displayed with or without hidden lines. User may also specify a flat reference plane at a given v value (such as a critical crack opening). This will enable the user to easily

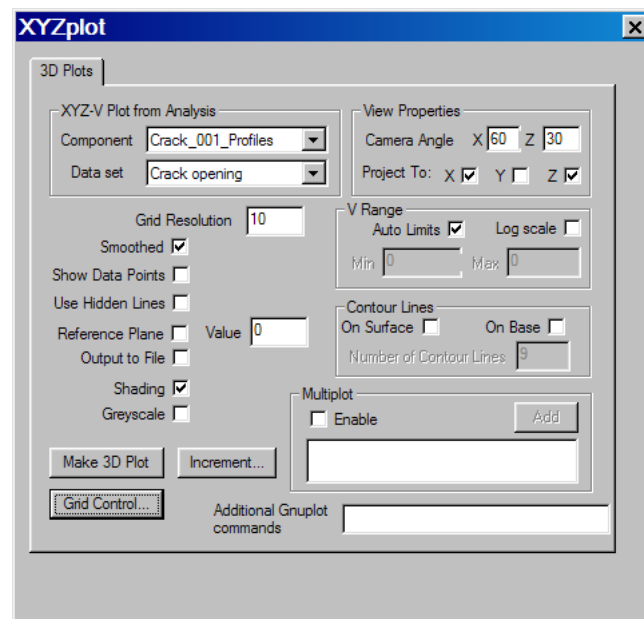


Figure 1.34: Control Panel for the Display of Surface Plots Associated with Cracks/Joints

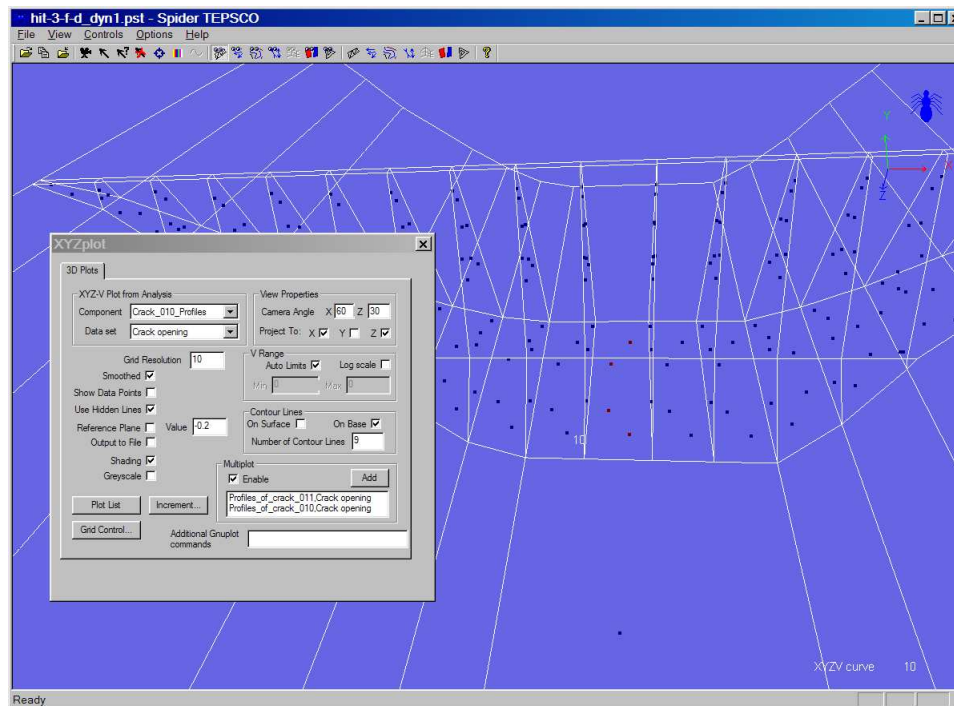


Figure 1.35: Spider Display When user Selects $xyz - v$ data Set

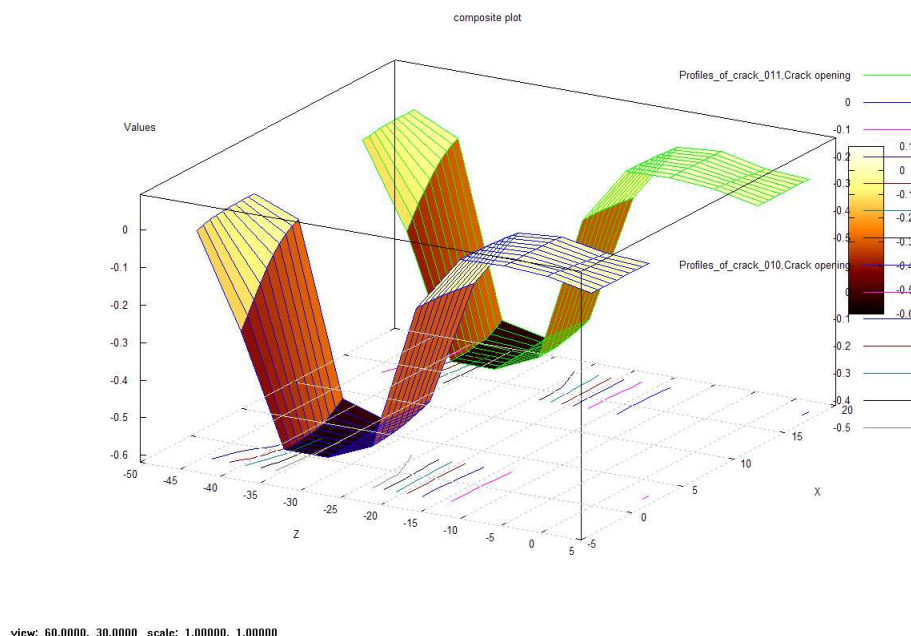
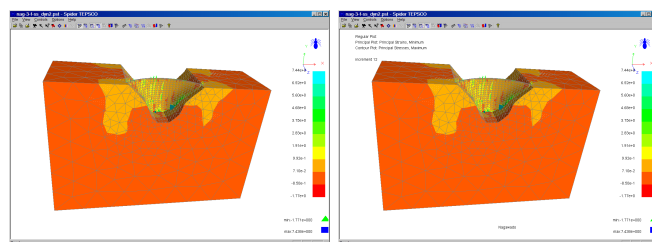
Figure 1.36: Example of $xyz - v$ 3D Plot of two Data Sets

Figure 1.37: Plot Title, without and with Labels

determine if part of the plots violates a certain criteria. As in other plots, data can be either saved into a file, or displayed in a grid (excel alike) format. Shading and greyscale are also possible.

User can specify the camera angle, set the limits of the v values and specify if a log scale (to avoid distortions caused by singularities) is required. Contour lines may be superimposed on the surface or projected on the base. A key aspect of this feature is Spider ability to project the user supplied data points into one of the major planes ($x - y$, $y - z$ or $x - z$). It will do its best to determine the optimal one, however the user can overwrite Spider's guess. Again, multiplot is possible, and the user can provide additional Gnuplot commands, Fig. 1.36.

1.3.15 Show Title

This will simply display the title, the Increment No., and for each type of plot (contour, vector, carpet, principal), the corresponding scalar being plotted, Fig. 1.37.



Figure 1.38: ToolBar

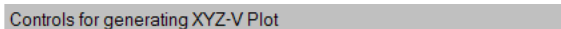


Figure 1.39: Statusbar

1.3.16 Focus View

Rotate and scale with respect to a point selected by , can also be activated from .

1.3.17 Reset Camera

 will recenter the display inside the window.

1.3.18 Clear Points of Interest

Will clear the picked points from the mesh, similar to .

1.3.19 Toolbar

Display the toolbar, Fig. 1.38

1.3.20 Statusbar

Display the statusbar, Fig. 1.39

1.4 Options

1.4.1 Increments

Increments, Fig. 1.40, allows the user to select the load increment to be viewed. Once selected, the graphical display is automatically updated.

The Animate/Stop button allows the user to view a continuous simulation of the mesh as the load is being applied. Hence, all plots (contour, vectors, and others) will be updated as Spider loops through the increments.

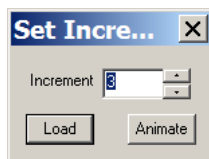


Figure 1.40: Option: Increments

1.4.2 Groups

In the mesh definition provided to Spider, each element is assigned a group number. The group typically include all those elements with the same element type and same material model. Hence, the Group Option, Fig. 1.41 allows the user to select those group identifiers to be displayed. When a group is selected, the min/max and “thermometers” are automatically

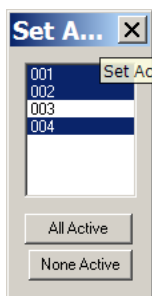


Figure 1.41: Option: Groups

updated to consider only those groups currently displayed.

1.4.3 Settings

The Settings Option, Fig. 1.42 allows the user to control a number of control parameters in Spider. In particular

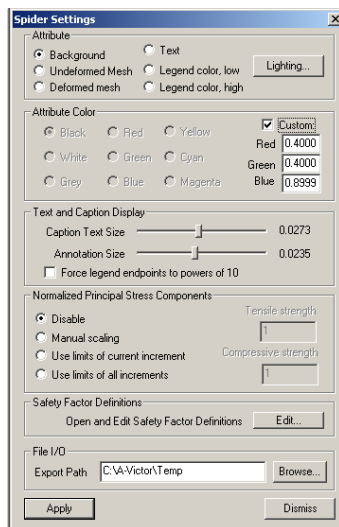


Figure 1.42: Option: Setting

Colors for background, mesh, deformed mesh, low and high of the “thermometer”.

Caption Text Size controls the font size (all of them).

Annotation Size controls the size of the marker identifying user selected nodes.

Force legend endpoints to power of 10 rounds the legend end point to power of 10 which may facilitate data interpretation.

Normalized Principal Stress Components allows the user to normalize the maximum (positive) and minimum (negative) principal stresses with respect to a user defined value:

Disable No normalization (default).

Manual scaling allows the user to specify the maximum and minimum normalizing values (typically the tensile and compressive strength).

Use limits of current increment normalizes with respect to the maximum and minimum of the current increment (hence the legend upper and lower bounds will be +/- 1).

Use limits of all increments normalizes with respect to the maximum and minimum of all the increments.

Factor of Safety Settings Mohr-Coulomb or von Mises, see Appendix A, Fig. 1.43.

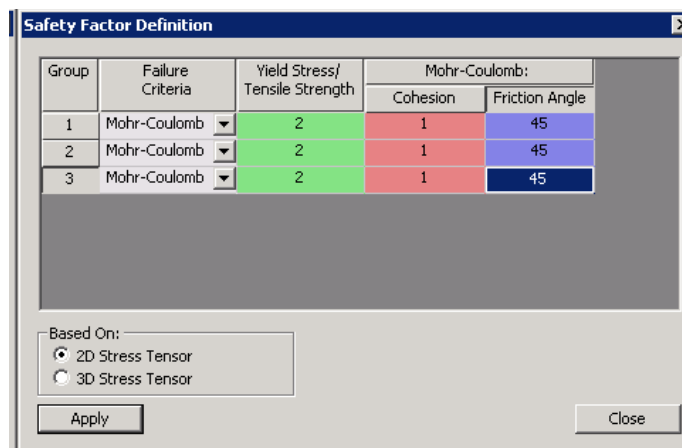


Figure 1.43: Factor of Safety Parameter Setting

1.4.4 Cut Mesh

Cut Mesh, Fig. 1.44 (restricted to 3D meshes) allows the user to slice the mesh into many layers, and then display contour lines on the surface of those cut planes. When first invoked, a grey disk is displayed (its size is irrelevant, it can be adjusted to properly identify the cutting plane) and its location orientation can be adjusted with the appropriate sliders. The user can then select the number of cuts, and the spacing between plots. Then, the **Apply** button will instruct Spider to perform the desired operation (for complex 3D meshes this operation is computationally intensive). Fig. 1.45 illustrates this capability of Spider. Note that in this figure the “cutting disc” is displayed.

1.4.5 Split Mesh

Split Mesh, Fig. 1.46 allows the user to split the mesh into two or more subgroups, each one with different view characteristics. When first invoked, a grey disk is displayed (its size is

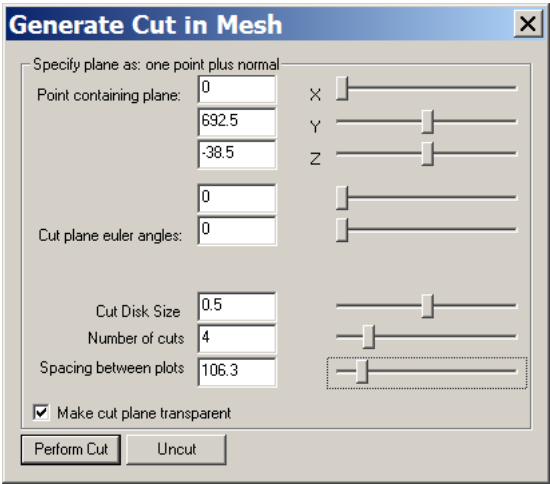


Figure 1.44: Option: Cut

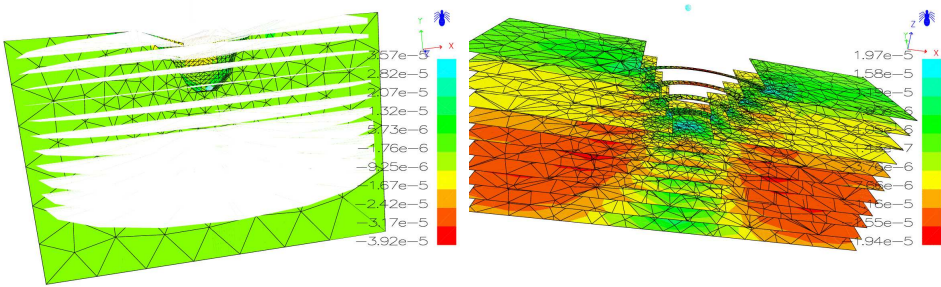


Figure 1.45: Example of Cut Mesh

irrelevant, it can be adjusted to properly identify the cutting plane) and its location orientation can be adjusted with the appropriate sliders. Once the disk location corresponds to the desired boundary between two different plot regions, user must select the **Add plot layer**. Then, a

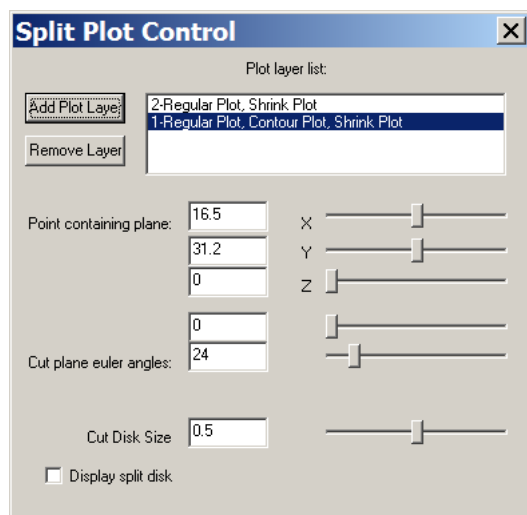


Figure 1.46: Option: Split

new entry is added to the display list. At that point, the user can select any entry from the list, and adjust its display characteristics, Fig. 1.47. Note that in this figure the “cutting disc” is displayed, and we first have two groups (maximum principal stress shown on the left, and minimum principal shown on the right). In the second case we have added a third layer showing principal plots.

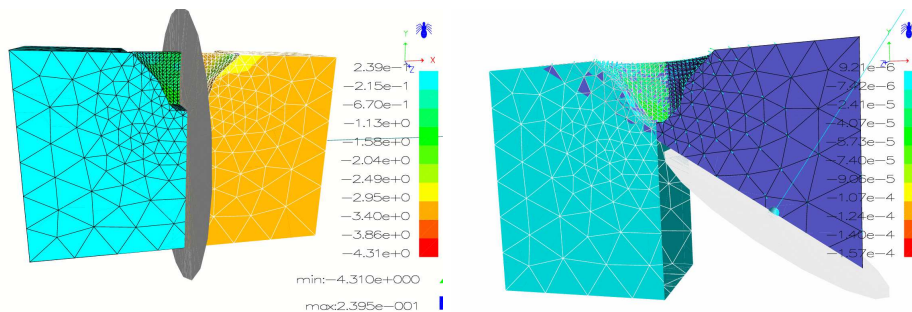


Figure 1.47: Example of Split Mesh

1.4.6 Lighting

Lighting can also be controlled, Fig. 1.48. This feature is critical for proper 3D viewing. Lighting can be enabled or disabled, and the lighting position can also be shown. The user has control on both the ambient and diffuse lighting, as well as on the orientation (but not location) of the light via its euclidian angles.

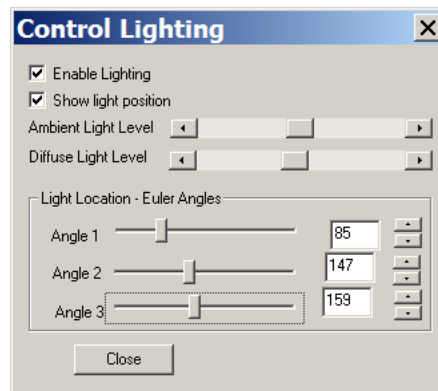


Figure 1.48: Option: Lighting

1.4.7 Separate Group

This option enables the user to “pull out” all, one, or more groups through the slider for better view, Fig. 1.49.

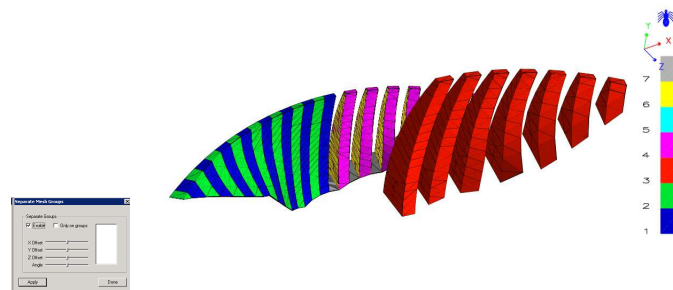


Figure 1.49: Regular Mesh; Separate Groups (GUI and Effect)

User may want to select only selected group of elements, Fig. 1.50.

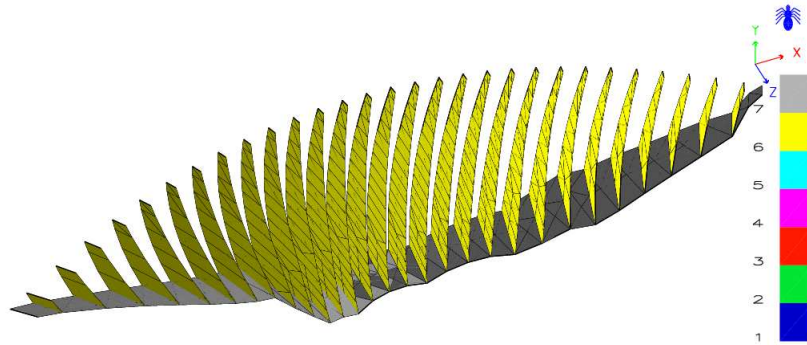


Figure 1.50: Regular Mesh; Selected Groups

Chapter 2

Eigenvalue .eig Visualizer

Some finite element analysis are limited to an eigenvalue one (such as in Stability or modal dynamic analysis). In those analysis, the only relevant quantity to be displayed is the eigenvector.

Spider, recognizes .eig files as one resulting from an eigenvalue analysis, and allow the display of the eigenvalue, Fig. 2.1. Once the file has been loaded, the user must select the

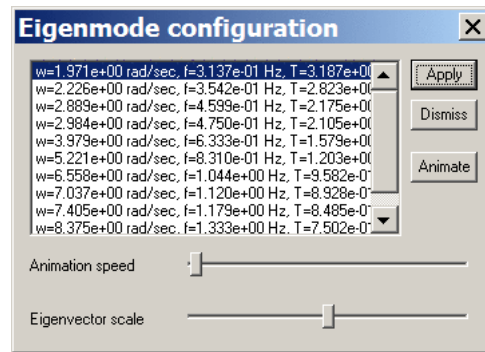



Figure 2.1: Eigenvalue Control

dynamic icon () and then select the desired eigenmode. The mode shape is then displayed, and the user has control on the deformation factor and can simulate the vibration of the structure. User can also control the animation speed.

Most of the regular features of Regular plots can be activated for eigenmode display, in particular group selection and lighting, Fig. 2.2.

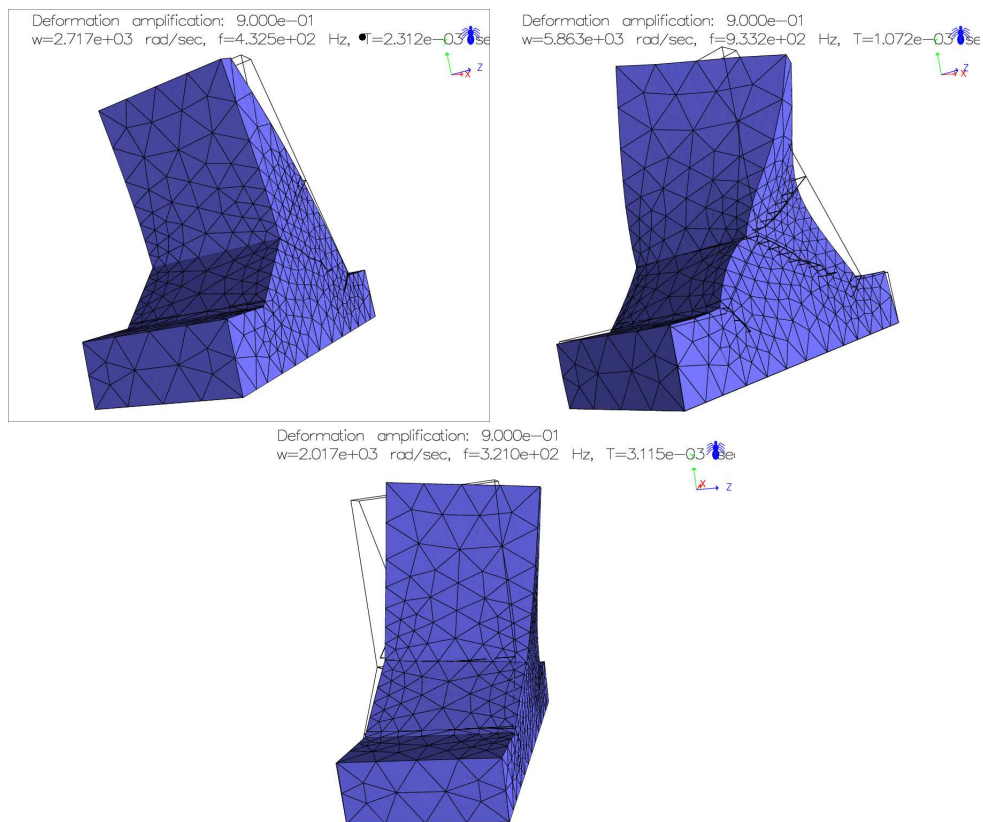


Figure 2.2: Example of eigenmode Viewing

Chapter 3

Real Time .rtv Viewer

3D dynamic nonlinear finite element analysis can necessitate many hours of computation. The Real Time viewer allows the user to monitor displacements and accelrograms in **real time** while the analysis is taking place. Hence, through this immediate feedback, one can have the assurance that the analysis is proceeding correctly, and that that there are no signs of divergence.

Similarly, upon completion of the analysis, the user can play-back the structural response and closely monitor key accelerations.

An **.rtv** file contains: nodal coordinates, element connectivities, and for each time step displacements and accelerations at selected nodes.

Hence, Spider, recognizes **.rtv** files as one resulting from a time dependent analysis, and allow the display of deformed shapes and accelrograms, Fig. 3.1. and then the user can control the graphical display.

Increment allows the user to select the increment number. Once entered, the deformed mesh is automatically updated.

Animate will initialize the dynamic animation of the mesh. Speed and mesh deformation can be controlled by the corresponding sliders. During animation, the accelrogram is being updated. If the analysis is running in the background, then the graph is continuously being updated. If the analysis is completed, then a vertical line scrolls along the time axis of the accelrogram to indicate current accelerations corresponding to the deformed mesh.

Components to Display allows the user to select one or more of the cartesian components of the accelerations, and/or the root mean square of the acceleration.

Show Nodes permits the user to select the nodes which accelerations are to be monitored.

Gnuplot will send current accelrogram to a gnuplot window for better viewing, and possible copying into the clipboard (as an **.emf** file).

Time Limits are by default set to t_{min} and t_{max} , but they can be overwritten by the user.

Use global min/max allows the user to set the min and max accelerations in the accelrogram (y axis) to correspond to those of the current display, or of the entire data range. This greatly facilitate data interpretation, and puts them “in perspective”.

Display accel. Tex will display textual information on the screen.

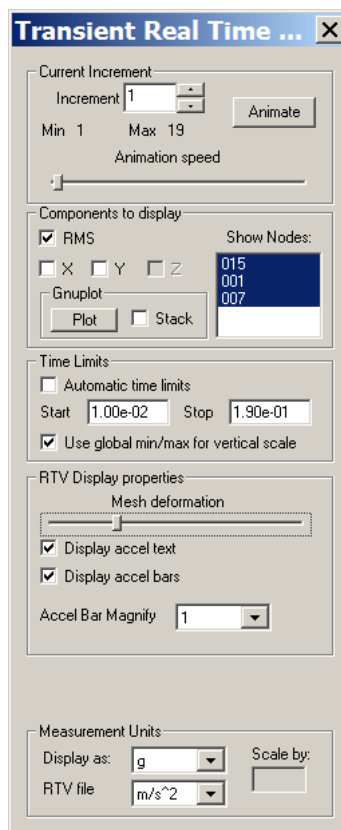


Figure 3.1: Real Time Control

Display accel. bars will display bars on the screen corresponding to each of the acceleration components. The bar is normalized to one unit as defined below. It provides a visual feedback on the acceleration of various degrees of freedom.

Accel Bar magnify is a magnifying factor for the bar accelerations. By default it is set to 1, but can be increased to 10, 20, 50 and 100.

Display as will include the acceleration units. By default it is none, but can be set to g , gal , m/s^2 , ft/s^2 , and custom.

RT file allows the user to specify the acceleration units in the `.rtv` file. Spider will then internally perform the appropriate conversion of the input acceleration to be consistent with the display unit. For custom units, the user can manually set the scaling factor.

Fig. 3.2 shows the full display of `.rtv` files, we note the textual information, and the varying

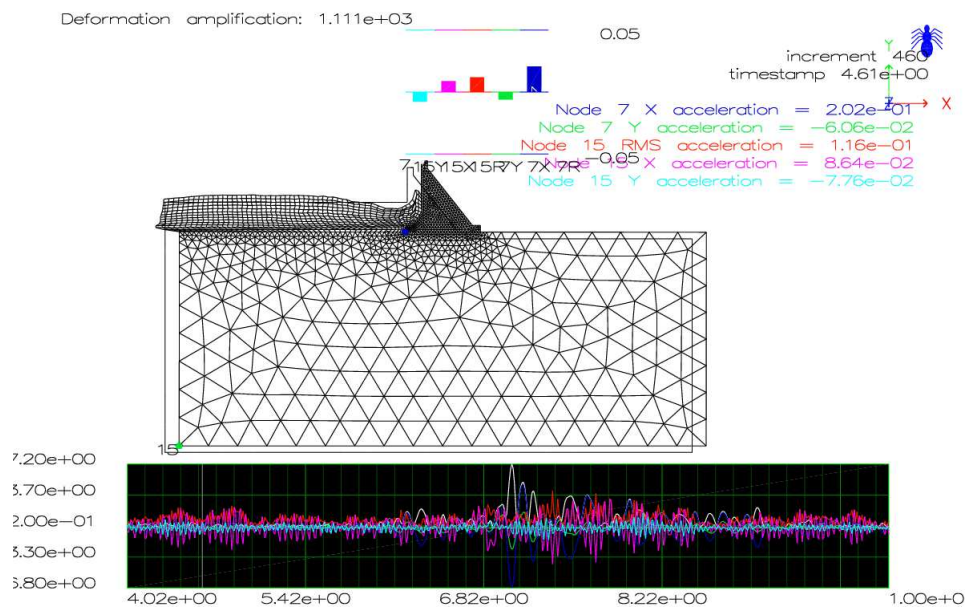


Figure 3.2: Example of Real Time Viewing; Full Display

thermometer associated with each node degree of freedom acceleration. In Fig. 3.3 textual display was removed.

Finally, the two type of (GnuPlot generated) plots associated with an `rtv` file are show in Fig. 3.4.

Note, that this feature can also be used in static nonlinear problems to simply monitor the deformation of the mesh in real time.

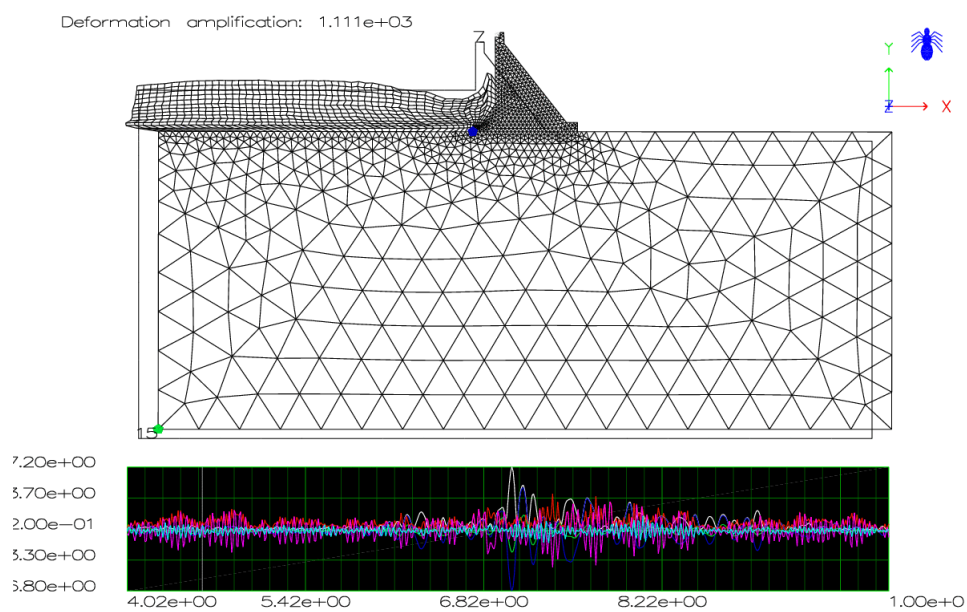


Figure 3.3: Example of Real Time Viewing; Partial Display



Figure 3.4: Example of Real Time Viewing Accelerogram Plots

Appendix A

Safety Factors

A.1 Mohr-Coulomb

With respect to Fig. A.1, Spider uses the following equations:

```
OA=(sig3+sig1)/2
AB=(sig1-sig3)/2
AD=-OA*sin(phi)
DC=c*cos(phi)
AC=AD+DC
sf1=AC/AB
sf2=(ft-OA)/AB
sf=min(sf1,sf2)
sf=max(sf,0)
```

A.2 Von-Mises

For Metals, Spider computes the safety factor as the ratio of the von-Mises stress divided by the yield stress.

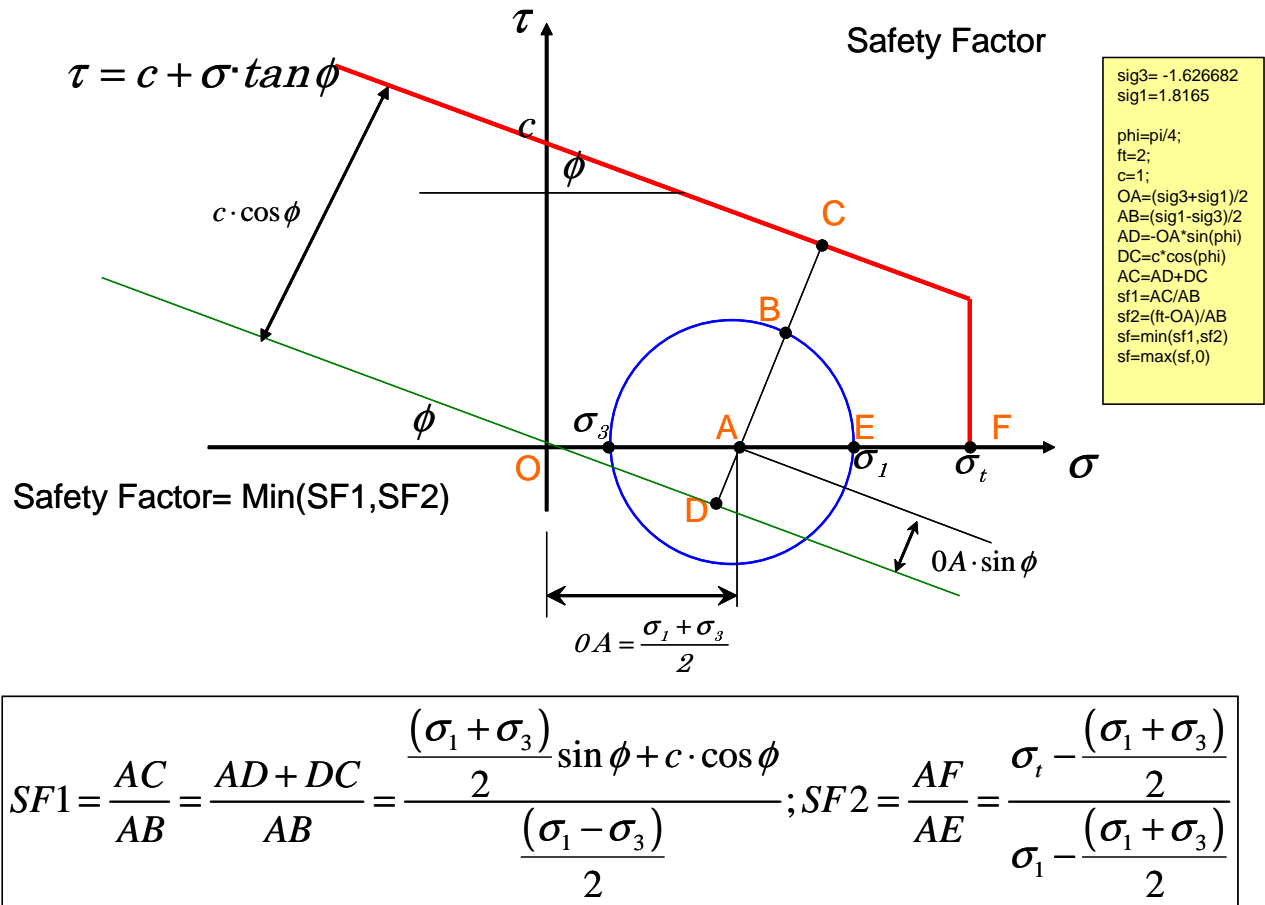


Figure A.1: Safety Factor for Cohesive Materials

Appendix B

FFT, Transfer Functions, and Deconvolution

Seismic events originate through tectonic slips and elastic waves (p and s) traveling through rock/soil foundation up to the surface. Hence, the seismographs (usually installed at the foot of the dam) record only the manifestation of the event.

On the other hand, modelling the foundation is essential for proper and comprehensive analysis of the dam, and as such the seismic excitation will have to be applied at the base of the foundation.

However, Fig. B.1, if we were to apply at the base the accelerogram recorded on the surface $I(t)$, the output signal $A(t)$ at the surface will be different than the one originally recorded (unless we have rigid foundation).

Hence, the accelerogram recorded on the surface must be deconvoluted into a new one $I'(t)$, such that when the new signal is applied at the base of the foundation, the computed signal at the dam base matches the one recorded by the accelerogram.

B.1 Fourier Transform

Fourier transforms enables us to transfer a signal from the time domain to the frequency domain.

Hence, the FFT takes us from the time domain to the frequency domain through the following equation:

$$X(\omega) = \int_{-\infty}^{\infty} x(t)e^{-i2\pi\omega t}dt \quad (\text{B.1})$$

$$x(t) \xrightarrow{\text{FFT}} X(\omega) \quad (\text{B.2})$$

while the inverse FFT takes us back from the frequency domain to the time domain through:

$$x(t) = \int_{-\infty}^{\infty} X(\omega)e^{i2\pi\omega t}d\omega \quad (\text{B.3})$$

$$X(\omega) \xrightarrow{\text{FFT}^{-1}} x(t) \quad (\text{B.4})$$

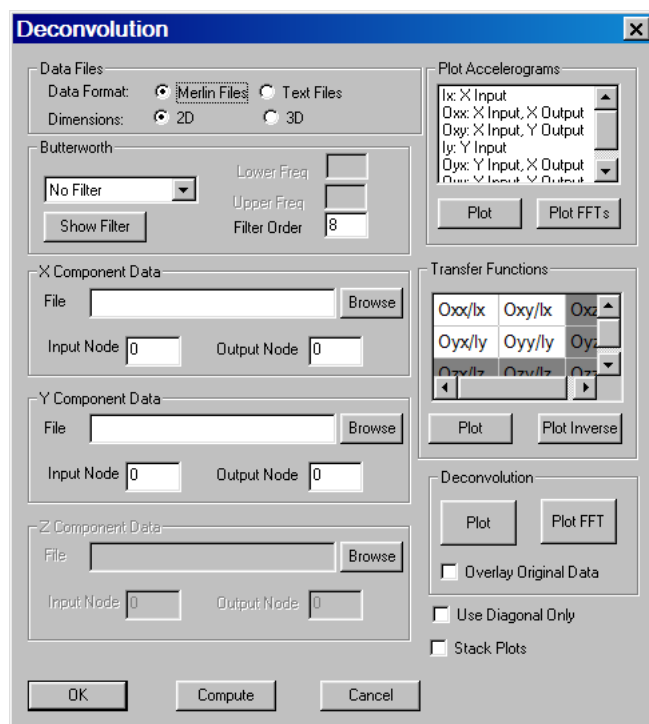


Figure B.1: Deconvolution Graphical User Interface

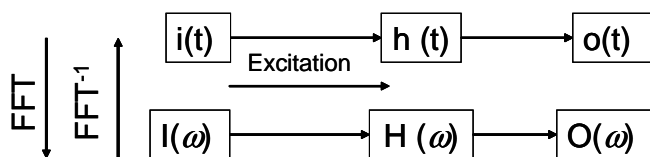


Figure B.2: Time Frequency Domains

B.2 Transfer Function

In dynamic event, we can define an input record $i(t)$ which is amplified by $h(t)$ resulting in an output signal $o(t)$, Fig. B.2. Similarly, the operation can be defined in the frequency domain. This output to input relationship is of major importance in many disciplines.

The transfer function is the Laplace transform of the output divided by the Laplace transform of the input.

Hence, in 1D, we can determine the transfer function as follows:

1. $i(t) \xrightarrow{\text{FFT}} I(\omega)$
2. $o(t) \xrightarrow{\text{FFT}} O(\omega)$
3. Transfer Function is $TF_{I-O} = O(\omega)/I(\omega)$

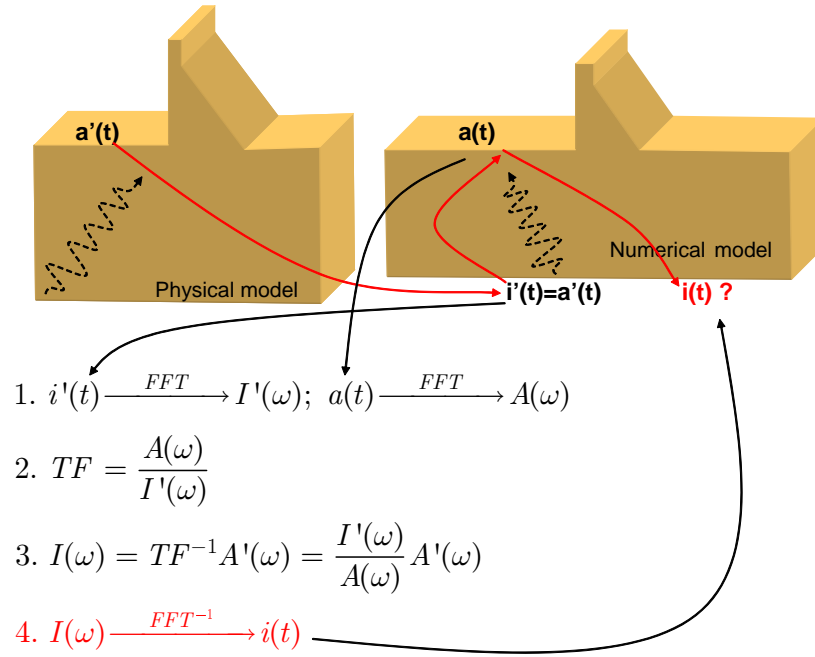


Figure B.3: Deconvolution Definition

B.3 Deconvolution

B.3.1 1-D

Extending our discussion one step further, we introduce the concept of deconvolution which addresses the dilemma posed above, and will now require one (or more) finite element analyses.

With reference to Fig. B.3

1. We record the earthquake induced acceleration on the surface $a'(t)$. and apply it as $i'(t)$ at the base of the foundation.
2. Perform a transient finite element analysis.
3. Determine the surface acceleration $a(t)$ (which is obviously different from $i(t)$).
4. Compute:

$$i'(t) \xrightarrow{FFT} I'(\omega) = A'(\omega) \quad (2.5-a)$$

$$a(t) \xrightarrow{FFT} A(\omega) \quad (2.5-b)$$

5. Compute transfer function from base to surface as $TF_{I'-A} = A(\omega)/I'(\omega)$.
6. Compute the inverse transfer function $TF_{I'-A}^{-1}$.
7. Determine the updated excitation record in the frequency domain

$$I(\omega) = TF_{I'-A}^{-1} A'(\omega) = \frac{I'(\omega)}{A(\omega)} A'(\omega) \quad (2.5-c)$$

8. Determine the updated excitation in the time domain

$$i(t) \xrightarrow{\text{FFT}^{-1}} I(\omega) \quad (2.5-d)$$

B.3.2 3-D

In 3-D applications, the transfer function is a 3x3 matrix, each row corresponds to the response to an excitation in a given direction, and each column corresponds to the response in a given direction. Hence, three separate analysis must be performed $[I'_x \ I'_y \ I'_z]$ and for each excitation, we must determine the three components of the surface acceleration. Then we will compute the 3D transfer function:

$$[TF] = \underbrace{\begin{bmatrix} TF_{xx} & TF_{xy} & TF_{xz} \\ TF_{yx} & TF_{yy} & TF_{yz} \\ TF_{zx} & TF_{zy} & TF_{zz} \end{bmatrix}}_{TF_{I'-A}} = \begin{bmatrix} \frac{A_{xx}(\omega)}{I'_x(\omega)} & \frac{A_{xy}(\omega)}{I'_x(\omega)} & \frac{A_{xz}(\omega)}{I'_x(\omega)} \\ \frac{A_{yx}(\omega)}{I'_y(\omega)} & \frac{A_{yy}(\omega)}{I'_y(\omega)} & \frac{A_{yz}(\omega)}{I'_y(\omega)} \\ \frac{A_{zx}(\omega)}{I'_z(\omega)} & \frac{A_{zy}(\omega)}{I'_z(\omega)} & \frac{A_{zz}(\omega)}{I'_z(\omega)} \end{bmatrix} \quad (B.5)$$

Hence, the excitation to be applied in the frequency domain is given by:

$$\begin{Bmatrix} I_x(\omega) \\ I_y(\omega) \\ I_z(\omega) \end{Bmatrix} = [TF]^{-1} \begin{Bmatrix} A'_x(\omega) \\ A'_y(\omega) \\ A'_z(\omega) \end{Bmatrix} \quad (B.6)$$

while in the time domain it is

$$\begin{Bmatrix} I_x(\omega) \\ I_y(\omega) \\ I_z(\omega) \end{Bmatrix} \xrightarrow{\text{FFT}^{-1}} \begin{Bmatrix} I_x(t) \\ I_y(t) \\ I_z(t) \end{Bmatrix} \quad (B.7)$$

B.3.2.1 Simplification

The preceding 3D generalized procedure can be simplified if we were to ignore the off diagonal terms

$$[TF] = \begin{bmatrix} TF_{xx} & 0 & 0 \\ 0 & TF_{yy} & 0 \\ 0 & 0 & TF_{zz} \end{bmatrix} = \begin{bmatrix} \frac{A_{xx}(\omega)}{I'_x(\omega)} & 0 & 0 \\ 0 & \frac{A_{yy}(\omega)}{I'_y(\omega)} & 0 \\ 0 & 0 & \frac{A_{zz}(\omega)}{I'_z(\omega)} \end{bmatrix} \quad (B.8)$$

which will greatly simplify the inversion of the transfer function.

$$\begin{Bmatrix} I_x(\omega) \\ I_y(\omega) \\ I_z(\omega) \end{Bmatrix} = [TF_{I'-A}]^{-1} \begin{Bmatrix} A'_x(\omega) \\ A'_y(\omega) \\ A'_z(\omega) \end{Bmatrix} \quad (B.9)$$

$$\begin{Bmatrix} I_x(\omega) \\ I_y(\omega) \\ I_z(\omega) \end{Bmatrix} \xrightarrow{\text{FFT}^{-1}} \begin{Bmatrix} I_x(t) \\ I_y(t) \\ I_z(t) \end{Bmatrix} \quad (B.10)$$

Appendix C

System Implementation

Spider has been under (almost) continuous development for over 12 years. It was first developed on a Sun/Unix workstation using PHIGS. It was then ported to a Window environment into Open-Inventor, and finally rewritten in Open-GL. Its graphical user interface has followed a similar path; first written in Open Look, then in Motif, and finally in Microsoft MFC.

C.1 Program structure

Spider is built around three central data structures. The lowest level is the mesh variables, a structure containing the mesh definition and associated data. For two dimensional meshes the winged edge data structure (Reference to Winged Edge paper) is used to represent the mesh's topology, and for three dimensional meshes the radial edge data structure (Reference to Radial Edge paper) is used to represent the mesh's topology.

There are also other important structures, to hold information about the plot to draw and the properties of the various possible plots.

Spider is written in a modular fashion, and parts of the code that do not interact are independent of each other.

C.1.1 The mesh variables

The mesh variables structure contains the data found in the input and post files. This includes the node and element definitions, the post data type information and the post data, as well as the application specific data such as the XY plot and fracture parameter data. Computed post data such as the principal values and vector lengths are also stored in the mesh variables structure.

C.1.2 The winged and radial edge structures

The winged and radial edge structures contain a representation of the mesh topology for two and three dimensional meshes, respectively. Each is a hierarchical structure where the basic elements are vertices, edges and faces, and the topology of the mesh are represented by the positional relationship of these basic elements. The radial edge structure is a more complex structure, with more elements than the winged edge structure, but the two are used in a similar fashion in Spider.

Both the winged and radial edge structures are not as complete and general as described in (Reference to Winged Edge paper) and (Reference to Radial Edge paper), as the problem of representing a finite element mesh's topology is fairly straightforward.

C.1.3 Other structures

There are two additional structures, the plot variables and the phigs variables, that are important in Spider.

The plot variables is a structure containing the data specified by the user in the various plot properties popup windows, as well as the various computed and temporary data used when creating a plot.

C.2 General program structure

Spider is event driven, i.e. it does not have a main loop, but relies on user interface callback functions to call the necessary functions to perform the actions desired by the user. The program is built as a number of functions with a user interface layer taking care of the interface between the user and the functionality offered.

Spider can broadly be split into five parts, the user interface functions, the drawing functions, the graphics functions, the data structure functions and the utility functions.

- The user interface functions handles the visible user interface, and fills in the plot variables and calls the drawing functions to create the plot desired by the user.
- The drawing functions takes the mesh, wing/radial, plot and Open-GL variables as arguments and draws the plot according to the data present in those structures.
- The graphics functions takes care of the interface between the drawing functions and Open-GL.
- The data structure functions offers the data structures and operations on the data structures to the rest of the code in the program.
- The utility functions handle odds and ends, such as file checking.

Each of the various data structures are implemented as a separate module, only dependent and relying on lower level data structure modules. This makes the data structures easy to maintain and easy to change if there is a need.

For each of the various plots or other functionality in Spider, there are a number of functions that implements the plot's property popup window and drawing routines, that are independent of other plots' functionality. This modularity makes Spider easy to maintain and extend, as a new type of plot or other functionality can be added without affecting other plots or other functionality.

Appendix D

.pst Post Data file Format

D.1 Writing Merlin Subroutine

```
1 C      character*80 xxxxxx , xxxxxx , xxxxxx , xxxxxx , xxxxxx , xxxxxx ,
2 integer
3      *      function wrtpst ( nodatr , id , flxnod , tmptot ,
4      *      epsnod , signod , dsptot , frctot , frccup ,
5      *      react , resid , epspla , eqepla , errnod ,
6      *      veltot , acctot , tottml , aarnto , rvalue )
7 c
8 c*****
9 c
10 c      This subroutine writes values of post variables from current
11 c      increment to the post file. Post variables are nodal data,
12 c      that will be used mostly by the postprocessor.
13 c
14 c      Variables Required:
15 c      nodatr: Nodal attributes
16 c      id : Equation id's for nodal DOF's
17 c      tmptot: Total nodal temperatures/heads
18 c      epsnod: Total nodal strains
19 c      signod: Total nodal stresses
20 c      dsptot: Total nodal displacements
21 c      react : Nodal reaction vector
22 c      resid : Residual force vector
23 c      epspla: Nodal plastic strains
24 c      eqepla: Nodal equivalent plastic strains
25 c      veltot: Total nodal velocity
26 c      acctot: Total nodal acceleration
27 c      tottml: Total temperature loading
28 c      aareps: Total aar strain
29 c
30 c*****
31 c
32 c
33 c      include 'dim4db.cmn'
34 c      include 'groups.cmn'
35 c      include 'iodata.cmn'
36 c      include 'pmmkey.par'
37 c      include 'status.cmn'
38 c      include 'postfl.cmn'
39 c
40 c      Argument Type Declarations:
41 c
42 c      integer
43 c      *      nodatr (Nndatr , Nnode), id (Mnddof , Nnode)
44 c      double precision
45 c      *      flxnod (Melstr , Nnode), tmptot ( Neq),
46 c      *      epsnod (Melstr , Nnode), signod (Melstr , Nnode),
47 c      *      dsptot ( Neq), frctot ( Neq), frccup ( Neq),
48 c      *      react ( Neq), resid ( Neq),
49 c      *      epspla (*), eqepla (*),
50 c      *      errnod (Nelerr , Nnode), veltot ( Neq),
51 c      *      acctot ( Neq), tottml ( Nnode),
52 c      *      aarnto (Melstr , Nnode),
53 c      *      rvalue (Nrvals , Nnode)
54 c
55 c      Local Variable Type Declarations:
56 c
57 c      integer
58 c      *      strcmp ( 6), nbytes , nwrite , ipstvr , node ,
59 c      *      inod , ndof , postid , icmpnt , ieq
60 c      double precision
61 c      *      vars (MAXVAR)
62 c
```

```

63 c Function Type Declarations:
64 c
65 c     integer iepspla , ieqepla ,
66 c     *           pmsize , wrtbin
67 c
68 c Initialize I/O error flag to indicate an error occurred
69 c
70 c     wrtpst = 0
71 c
72 c Define correspondence table for strain/stress components
73 c
74 c     strcmp(1) = 1
75 c     strcmp(2) = 2
76 c     strcmp(3) = 3
77 c     strcmp(4) = 4
78 c     strcmp(5) = 5
79 c     strcmp(6) = 6
80 c
81 c     if (Melstr .eq. 3) strcmp(4) = 3
82 c
83 c Loop over nodes and post variables to write nodal post data
84 c
85 c     nbytes = Npstvr * pmsize( REAL )
86 c
87 c     do 20 node = 1, Nnode
88 c         inod = nodatr(1,node)
89 c         ndof = nodatr(2,node)
90 c
91 c         do 10 ipstvr = 1, Npstvr
92 c             postid = (Pstkey(ipstvr) / 100) * 100
93 c             icmpnt = Pstkey(ipstvr) - postid
94 c
95 c             if (postid .eq. 100) then
96 c
97 c Post variable is a temperature
98 c
99 c                 ieq = id(icmpnt,inod)
100 c                 vars(ipstvr) = tmptot(ieq)
101 c
102 c             else if (postid .eq. 200) then
103 c
104 c Post variable is a head
105 c
106 c                 ieq = id(icmpnt,inod)
107 c                 vars(ipstvr) = tmptot(ieq)
108 c
109 c             else if (postid .eq. 300) then
110 c
111 c Post variable is a flux component
112 c
113 c                 vars(ipstvr) = flxnod(icmpnt,inod)
114 c
115 c             else if (postid .eq. 400) then
116 c
117 c Post variable is a displacement component
118 c
119 c                 if (icmpnt .le. ndof) then
120 c                     ieq = id(icmpnt,inod)
121 c                     vars(ipstvr) = dsptot(ieq)
122 c                 else
123 c                     vars(ipstvr) = 0.0
124 c                 end if
125 c
126 c             else if (postid .eq. 500) then
127 c
128 c Post variable is an applied force component
129 c
130 c                 if (icmpnt .le. ndof) then
131 c                     ieq = id(icmpnt,inod)
132 c                     vars(ipstvr) = frctot(ieq) + frccup(ieq)
133 c                     vars(ipstvr) = frctot(ieq)
134 c                 else
135 c                     vars(ipstvr) = 0.0
136 c                 end if
137 c
138 c             else if (postid .eq. 600) then
139 c
140 c Post variable is a reaction component
141 c
142 c                 if (icmpnt .le. ndof) then
143 c                     ieq = id(icmpnt,inod)
144 c                     vars(ipstvr) = react(ieq)
145 c                 else
146 c                     vars(ipstvr) = 0.0
147 c                 end if
148 c
149 c             else if (postid .eq. 700) then
150 c
151 c Post variable is a residual force component
152 c
153 c                 if (icmpnt .le. ndof) then
154 c                     ieq = id(icmpnt,inod)

```

```

155      vars(ipstvr) = resid(ieq)
156      else
157      vars(ipstvr) = 0.0
158      end if
159 c
160      else if (postid .eq. 800) then
161 c
162 c      Post variable is a total strain component
163 c
164      icmpnt      = strcmp icmpnt)
165      vars(ipstvr) = epsnod(icmpnt,inod)
166 c
167      else if (postid .eq. 900) then
168 c
169 c      Post variable is a total stress component
170 c
171      icmpnt      = strcmp(icmpnt)
172      vars(ipstvr) = signod(icmpnt,inod)
173 c
174      else if (postid .eq. 1000) then
175 c
176 c      Post variable is a plastic strain component
177 c
178      icmpnt      = strcmp(icmpnt)
179      vars(ipstvr) = epspla(iepspla(icmpnt,inod))
180 c
181      else if (postid .eq. 1100) then
182 c
183 c      Post variable is a equivalent plastic strain
184 c
185      vars(ipstvr) = eqepla(ieqepla(1,inod))
186 c
187      else if (postid .eq. 1200) then
188 c
189 c      Post variable is a strain error
190 c
191      vars(ipstvr) = errnod(1,inod)
192 c
193      else if (postid .eq. 1300) then
194 c
195 c      Post variable is an energy error
196 c
197      vars(ipstvr) = errnod(2,inod)
198 c
199      else if (postid .eq. 1400) then
200 c
201 c      Post variable is a optimal element size
202 c
203      vars(ipstvr) = errnod(3,inod)
204 c
205      else if (postid .eq. 2000) then
206 c
207 c      Post variable is a velocity
208 c
209      if (icmpnt .le. ndof) then
210      ieq      = id(icmpnt,inod)
211      vars(ipstvr) = veltot(ieq)
212      else
213      vars(ipstvr) = 0.0
214      end if
215 c
216      else if (postid .eq. 2100) then
217 c
218 c      Post variable is an acceleration
219 c
220      if (icmpnt .le. ndof) then
221      ieq      = id(icmpnt,inod)
222      vars(ipstvr) = acctot(ieq)
223      else
224      vars(ipstvr) = 0.0
225      end if
226 c
227      else if (postid .eq. 3000) then
228 c
229 c      Post variable is a temperature loading
230 c
231      vars(ipstvr) = tottml(inod)
232 c
233      else if (postid .eq. 3100) then
234 c
235 c      Post variable is an AAR strain component
236 c
237      icmpnt      = strcmp(icmpnt)
238      vars(ipstvr) = aarnto(icmpnt,inod)
239 c
240      else if (postid .eq. 3200) then
241 c
242 c      Post variable is an R-value for the rock model loading
243 c
244      vars(ipstvr) = rvalue(1 , inod)
245 c
246      end if

```

```

247 10  continue
248 c
249 c      write(*,8000) node,(vars(ipstvr),ipstvr=1,Npstvr)
250 c 8000  format('Node ',i1,':',12(1pe15.5))
251 c
252 c      nwrite = wrtbin( Pstfid , nbytes , vars )
253 c      if (nwrite .ne. nbytes) goto 999
254 c
255 c 20  continue
256 c
257 c      wrtpst = 1
258 c
259 c 999 return
260 c      end

```

D.2 Introduction

Spider was originally developed as a 3D postprocessor for the finite element code MERLIN (which is primarily concerned with fracture mechanics analysis).

However, since its inception Spider was made as general as possible (i.e. no labels/information specific to MERLIN were “hardwired”), and hence it can easily be used a powerful and flexible finite element post-processor for virtually any finite element code and is not restricted to stress analysis. This is achieved by transferring all relevant nodal information (including labels) via the generic Spider input data file. It should be noted that in all cases only nodal values can be given (it remains the responsibility of the FE code to determine nodal stresses from Gauss Points).

Spider’s input data file is composed of various blocks of information, some of them listed only once while others may have to be either repeated or not listed at all.

Spider accept its input in either binary or as ASCII data files. In those files, unless otherwise stated,

INTEGER is a 32-bits integer number

REAL is a 64-bits floating point number (i.e., double precision),

STRING is a variable length character string

The structure of the file is the same in both cases, however there are differences mostly in the representation of the various data types. The following table summarizes the various data types in the file and their representation in the ASCII and binary format.

Data Type	Description	Format in ASCII File	Format in Binary File
INTEGER	Any integer type data	Integer value without decimal point, delimited by any whitespace character.	4 bytes
REAL	Any real type data	Real value, delimited by any whitespace character. Don't use D format in in FORTRAN.	8 bytes
STRING	Any character based data	String of characters, delimited by new line characters	String of characters preceded by an INTEGER indicating it's length.

Because binary data files can not be read from a different machine than the one which wrote it, and since binary data files are much more compact than their ASCII counterparts, tools are available to translate binary to ASCII and ASCII to binary Spider files (hence one can run a FE code on a Unix workstation, and post-process results on a Pentium based machine using compact binary data files).

D.2.1 Post File Structure

The structure of the unformatted output file is described in the following diagram. The first block is always the file header followed by the incremental data blocks. The number of incremental data blocks is unlimited, however there can be only one header block in the file.

FILE HEADER

INCREMENTAL DATA - increment 0

INCREMENTAL DATA - increment 1

INCREMENTAL DATA - increment 2

...

D.3 File Header Block

The first block in the unformatted output file is the file header block. It contains the title of the problem, the definition of the stamp used as a separator between blocks, and a list of the post variables that appear in the file as nodal post variables. The list of post variables cannot change for the duration of an analysis, so an unformatted output file has only one file header block.

D.3.1 Title

The title is written to the unformatted output file as a variable length character string of the type **STRING**. The maximum number of characters that can be expected for the title is eighty (80). If the title is a null string (i.e., contains only blanks) the length of the string is zero and no string or an empty line appears in the file.

D.3.2 Stamp Definition

After the title an **INTEGER** value called a **stamp** must be given. The stamp is a bit-wise coded integer that will be used later in the file as a separator between blocks. When used as a separator, the stamp appears in pairs. The value for the stamp probably should not be zero as a pair of integer zeros or a double precision zero might commonly occur in the file. In MERLIN, the stamp has all bits “turned on”, which is -1 on most machines. If another application were to write an unformatted output file for post-processing, the choice of the stamp is left to the programmer.

D.3.3 Post Variable List

The list of nodal post variables in the unformatted output file contains a combination of **INTEGER**s and variable length character **STRING**s. Each unique post variable appearing in the file is called a **post variable type**. For example, displacements and stresses are post variable types for a stress analysis. The **INTEGER**s identify the rank (i.e., scalar, vector, or tensor) and the order (i.e., the number of components) for each post variable type. An additional **INTEGER**, which is typically a bit-wise flag, is provided to allow for the specification of characteristic of a post variable type beyond its order and rank. In order to allow for deformed structure plots when post-processing, the additional **INTEGER** is used to identify displacements.

Immediately following the stamp definition in the unformatted output file is an integer that indicates the number of nodal post variable types in the file. This number must be positive and non-zero or post-processing cannot be performed. A value of zero indicates that the file contains no nodal post variables and a value less than zero is considered to be an error. If there are no nodal post variables, post-processing cannot be performed because the post-processor does not recognize integration point information and, therefore, does not perform an extrapolation of integration point values to the nodes. If the unformatted output file is written by a program other than MERLIN, nodal extrapolation must be performed by that program for all post variable types that are not already nodal values.

For each post variable type there is a **post variable record** containing the information that defines it. The information defining a post variable type is, in the order of appearance, as follows:

1. **Rank:** An **INTEGER** indicating the rank:
 - 0 Scalar
 - 1 Vector
 - 2 Second order tensor
2. **Order:** An **INTEGER** indicating the order:
 - 0 For a single scalar value the order would be 1 and for scalar list (i.e., a set of unrelated scalar variables) the order would be the number of scalar list items.

- 1 For a 2-D vector quantity the order would be 2 and for a 3-D vector the order would be 3.
- 2 For a 2 by 2 symmetric tensor the order would be 3; for a 2 by 2 unsymmetric tensor the order would be 4; for a 3 by 3 symmetric tensor the order would be 6; and for a 3 by 3 unsymmetric tensor the order would be 9.
3. **Flag:** An INTEGER containing the bit-wise flags used to identify “special case” data for the post-processor. Special cases would include directly and indirectly scaled vectors (e.g., displacements are directly scaled and forces are indirectly scaled) or a tensor type (e.g., a tensor of engineering strains must be differentiated from one of true strains). The flag indicating that a vector post variable is a displacement vector has a value of 2 (i.e., 10 binary), the flag indicating that a tensor post value is a stress tensor has a value of 8 (i.e., 1000 binary) and the flag indicating that a tensor post value is an engineering strain tensor has a value of 16 (i.e., 10000 binary)
4. **Keyword:** A STRING variable is used to identify the post variable type in the finite element application program when performing a restart. This character string is called the **keyword**. It is not appropriate to use a null string as a keyword.
5. **Label:** A STRING variable is used as a menu label to identify the post variable type in the post-processor. This character string is called the **post variable type label**. If the post variable type label is a null string, this post variable type will be ignored by the post-processor.
6. **Component:** A list of STRINGS that serve as **post variable component labels** (i.e., the number of strings in the list corresponds to the order of the post variable type). The post variable component labels are used by the post-processor in the menus that “pull down” from the menu item from the post variable type label. A null string is an acceptable label, but the post-processor ignores all post variables that have a null string as their label. Labels are generally not used by the finite element application program.

D.3.4 Block Separator

The file header block terminates with a pair of stamps which act as an error checking mechanism. If both of these integers are not equal to the predefined stamp, an alignment error has been encountered and reading of the unformatted output file will be terminated immediately.

D.4 Incremental Data

After the file header block comes the incremental data block, which is made up of several sub-blocks. These sub-blocks include solution status parameters, finite element data, and the nodal post variables. The incremental data block can appear in the unformatted output file an arbitrary number of times, but it is not necessary to include an incremental data block for every increment of an analysis. This section describes the sub-blocks comprising the incremental data block, with the order of presentation for the sub-blocks corresponding to their order of appearance on the file.

D.4.1 Solution Status Parameters

The first sub-block in the incremental data block contains the solution status parameters. Three parameters indicate the status of the solution for the current increment:

1. The increment number (**INTEGER**).
2. The load factor (**REAL**).
3. The time (**REAL**).

The increment number is an **INTEGER** and the load factor and time are **REAL** numbers. The presence of the increment number in this information makes it possible to write only those increments to the file that are of interest to the user. The load factor is used in conjunction with automatic load scaling algorithms, such as the arc-length method, to indicate the current level of loading with respect to some arbitrary system of applied loads. It is included in the file to facilitate restarting of analyses that use these algorithms. The time is the total elapsed time for transient analyses. It is also included in the file to facilitate restarts.

D.4.2 Finite Element Data

Certain information related to the finite element method can be regarded as generic, such as nodal coordinates and element connectivity. Because the topology and geometry of a mesh can potentially change during the course of an incremental analysis, this information appears in the incremental data block of the unformatted output file. However, program dependent finite element data, such as a crack surface definition, might also change during the course of an analysis. Program dependent data can also be included in the file for use by the finite element application program, but this data will be ignored by the post-processor.

D.4.2.1 Mesh Size Parameters

The second sub-block in the incremental data block contains the mesh size parameters. Six parameters indicate the size of the mesh for the current increment:

1. The number of nodes in the mesh (**INTEGER**).
2. The number of corner nodes in the mesh (**INTEGER**).
3. The number of coordinates per node (**INTEGER**).
4. The number of elements in the mesh (**INTEGER**).
5. The maximum number of nodes per element (**INTEGER**).
6. The number of finite element application arrays (**INTEGER**).

All of the mesh size parameters are **INTEGER**s. Including the number of nodes and elements in the mesh in this sub-block allows for consistency checking between the mesh in program memory and the mesh defined in the file. If the number of nodes and elements in program memory do not correspond to the number of nodes and elements in the file, the mesh definition in the file must be read into program memory. If there is no mesh definition data in the file, reading of the file must be terminated prematurely. The number of corner nodes in the mesh

indicates to the post-processor how many of the nodes define element corners. The information for the corner nodes must appear before the mid-side nodes in the nodal coordinates and nodal post variables sub-blocks of the incremental block. The number of coordinates per node is non-zero only if nodal coordinates are included in the file. The maximum number of nodes per element is non-zero only if element connectivity is included in the file. If the number of coordinates per node is non-zero, the maximum number of nodes per element must also be non-zero, and vice versa. If the number of coordinates per node is zero, the maximum number of nodes per element must also be zero. The number of finite element application arrays can be zero, indicating that there are no finite element application arrays included in the incremental data.

D.4.2.2 Nodal Coordinates

Nodal coordinates are optional, appearing in the unformatted output file only when the number of coordinates per node is non-zero. MERLIN writes the nodal coordinates to the file for increment zero, and then for any other increment where the number of nodes change or the coordinates themselves change. By writing the nodal coordinates to the unformatted output file other finite element programs can avoid having to convert their input to MERLIN's input file format for post-processing. The nodal coordinates for each node are as follows:

1. The x -coordinate.
2. The y -coordinate.
3. The z -coordinate.

Only the x and y -coordinates are required for 2-D geometries and all three coordinates are required for 3-D geometries. The x -coordinate is the x -coordinate for plane stress, plane strain, and generalized 2- and 3-D continuum idealizations and the r -coordinate for axisymmetric idealizations. The y -coordinate is the y -coordinate for plane stress, plane strain, and generalized 2- and 3-D continuum idealizations and the z -coordinate for axisymmetric idealizations. The z -coordinate is the z -coordinate in generalized 3-D continuum idealizations. All coordinates are of type **REAL**.

D.4.2.3 Element Connectivity

Element connectivity is optional, appearing in the unformatted output file only when the maximum number of nodes per element is non-zero. MERLIN writes the element connectivity to the file for increment zero, and then for any other increment where the number of elements change or the element connectivity itself changes. By writing the element connectivity to the unformatted output file other finite element programs can avoid having to convert their input to MERLIN's input file format for post-processing. When writing element connectivity to file, only the nodes defining the element are actually written. In order to facilitate this approach, additional information is written to the file along with the connectivity. The information defining the connectivity for each element is as follows:

- The element type (**INTEGER**).
- The element group ID (**INTEGER**).
- First node in connectivity of element (**INTEGER**).

- Second node in connectivity of element (INTEGER).
- ...
- Last node in connectivity of element (INTEGER).

n is the number of nodes defining the element, including mid-side nodes, plus two. This information constitutes an **element connectivity record** and all n values are INTEGERS.

The nodal numbering conventions for the various combinations of nodes and geometric configurations are given in Figures ??-??.

Recognizable elements by Spider are shown in Fig. D.1¹.

Tables ??,

Element Type	Description		
	Number of Nodes	Geometry	Formulation
1	2	2-D Reinf-rod	Linear
8-10	3	Triangle	Linear
11-16	6	Triangle	Quadratic
2-7	4	Quadrilateral	Linear
17-19	8	Quadrilateral	Quadratic
20-22	9	Quadrilateral	Quadratic
23	4	2-D Interface	Linear
24	6	2-D Interface	Quadratic
25	4	Tetrahedron	Linear
26	10	Tetrahedron	Quadratic
27	6	Wedge	Linear
30-31	15	Wedge	Quadratic
45	5	Pyramid	Linear
46	13	Pyramid	Quadratic
28-29	8	Brick	Linear
32	20	Brick	Quadratic
33	6	Interface Triangle	Linear
35	12	Interface Triangle	Quadratic
34	8	Interface Quadrilateral	Linear
36	16	Interface Quadrilateral	Quadratic

Table D.1: Element Types Supported by Spider

The element group (defined to be any subset of elements within a mesh that are of the same element type and have the same geometric attributes and material properties) ID corresponds to a set of elements of the same type with the same material properties.

D.4.2.4 Finite Element Application Data

To accommodate finite element data which is program dependent, the finite element application data sub-block has been included in the incremental data block of the unformatted output

¹The “odd” numbering system is meant to accomodate elements defined in the Merlin library.

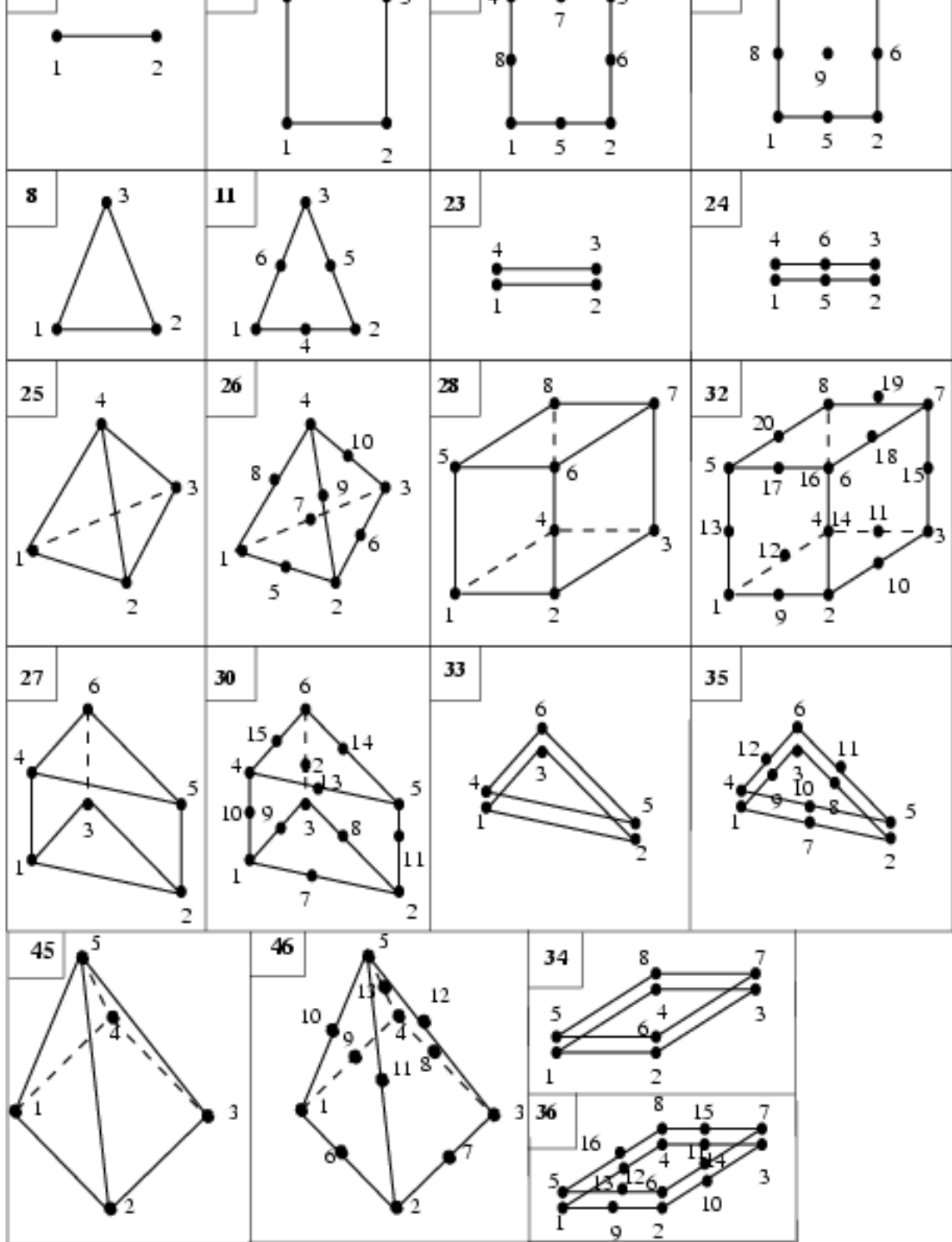


Figure D.1: Elements Supported by Spider

file. Finite element application data is optional, appearing in the file only if the number of finite element application data arrays is non-zero. An arbitrary number of finite element data application sub-blocks can appear in the incremental data block. Each finite element application data sub-block has three records: the **header record**, the **array record** and the **label record**.

The finite element application data header record identifies and describes the array. Seven parameters are used to identify and describe the array:

1. The array name.*j*
2. The flag
3. The data type:
 - 1 16-bit integer
 - 2 32-bit integer
 - 3 32-bit floating point
 - 4 64-bit floating point
4. Number of rows (NR), (INTEGER)

XY plottable	$NR = 2$ (x,y)
XYZ plottable	$NR = 3$ (x,y,z)
fracture parameters	$NR =$ (number of fracture parameters)

5. Number of columns. (NC), (INTEGER)

XY plottable	$NC \geq 1$ (number of data sets)
XYZ plottable	$NC \geq 1$ (number of data sets)
fracture parameters	$NC = 1$

6. Number of pages. (NP), (INTEGER)

XY plottable	$NP \geq 1$ (number of data points)
XYZ plottable	$NP \geq 1$ (number of data points)
fracture parameters	$NP = 1$

7. Number of labels. (NL), (INTEGER)

XY plottable $NR = 2$	$NL = NC + 4$
XYZ plottable $NR = 3$	$NL = NC + 5$
fracture parameters	$NL = NR$

The array name is a **STRING** and all other parameters are **INTEGERs**. The array name should not be a null string. The flag denotes some special kind of application data that can be recognized by the postprocessor. The following table summarizes the special types of data currently available in Spider.

Special Data Type	Flag Value
XY plotable	2
XYZ plotable	4
fracture parameters	8

The “XY plotable” data is a table of **REAL** values that can be plotted in Spider using XY plots. If this is the case the labels for the plot must be provided in the **label record**.

The “XYZ plotable” data is a table of **REAL** values that can be plotted in Spider using XYZ plots. If this is the case the labels for the plot must be provided in the **label record**. **This option is not available at present version of Spider and will be ignored until this notice is removed.**

The “fracture parameters” data are used to transfer the results from the fracture mechanics analysis to Spider or PreMERLIN.

The data type must be either 1, 2, or 3; no other values are recognized. The number of rows, columns, and pages must all be greater than or equal to one and their product should be equal to the total number of elements in the array. In the case of special type of application data as for example “XY plotable” or “fracture parameters” data the rows, columns and pages have the following meaning:

Application Data	Rows	Columns	Pages
XY plotable	NR=2 (x,y)	(number of data sets)	(number of data points)
XYZ plotable	NR=3 (x,y,z)	(number of data sets)	(number of data points)
fracture parameters	(number of parameters)	NC=1	NP=1

The finite element application data array record is the contents of the array. Since the header record establishes the type and size of the array, it is possible to read the entire array record with one read statement or to skip over it completely, as is done in the post-processor.

The finite element application data label record is the list of labels for the application data array. The labels are of type **STRING**. Since in most cases the application data block is ignored by the post-processor, the number of labels in the application data header record can be zero and the label record can be empty. Only in the case of special types of data like “XY plotable”, or “fracture parameters” the labels has to be provided and the number of labels is determined as follows:

XY plotable NR=2	NL=NC+4
XYZ plotable NR=3	NL=NC+5
fracture parameters	NL=NR

where the individual labels should be as follows:

- XY plotable**
1. menu label
 2. graph title
 3. label for x axis
 4. label for y axis
 5. label for data set 1

Test Problem	Title
6	Number of Post Variable Types
-1	Definition of the stamp

Table D.2: File Header Block

- 6. ...
- 7. label for data set NC

XYZ plotable 1. menu label

- 2. graph title
- 3. label for x axis
- 4. label for y axis
- 5. label for z axis
- 6. label for data set 1
- 7. ...
- 8. label for data set NC

fracture parameters 1. label for fracture parameter 1

- 2. ...
- 3. label for fracture parameter NR

D.4.3 Nodal Post Variables

The last sub-block in the incremental data block of the unformatted output file is the nodal post variables sub-block. The nodal post variables sub-block is optional, appearing only when the number of post variable types is non-zero. If there are no nodal post variables, post-processing cannot be performed because the post-processor does not recognize integration point information and, therefore, does not perform an extrapolation of integration point values to the nodes. If the unformatted output file is written by a program other than MERLIN, nodal extrapolation must be performed by that program for all post variable types that are not already nodal values.

Nodal post variables are written to the file in **post variable records**. Post variable records contain only **REAL** numbers. There is one post variable record for each node and each record is of the same fixed length. The length of the post variable record is the sum of the orders for the post variable types. Post variables appear in the same order in the post variable record as the post variable types and their components appear in the file header.

D.4.4 Block Separator

The incremental data block terminates with a pair of stamps, which act as an error checking mechanism. If both of these integers are not equal to the predefined stamp, an alignment error has been encountered and reading of the unformatted output file will terminate immediately.

1 3 2	vector order 3; scalable(displacement)
DSPTOT	keyword
Displacements	post variable type label
Displacement_u	post variable component labels
Displacement_v	post variable component labels
Displacement_w	post variable component labels
1 3 0	vector order 3; non scalable
FRCTOT	keyword
Applied Forces	post variable type label
Force_x	post variable component labels
Force_y	post variable component labels
Force_z	post variable component labels
1 3 0	vector order 3; non scalable
REACT	keyword
Reactions	post variable type label
Reaction_x	post variable component labels
Reaction_y	post variable component labels
Reaction_z	post variable component labels
1 3 0	vector order 3; non scalable
RESID	keyword
Residuals	post variable type label
Residual_x	post variable component labels
Residual_y	post variable component labels
Residual_z	post variable component labels
2 6 16	Tensor, 3x3 symmetric; Engineering strain
EPSNOD	keyword
Strains	post variable type label
Epsilon_xx	post variable component labels
Epsilon_yy	post variable component labels
Epsilon_zz	post variable component labels
Gamma_xy	post variable component labels
Gamma_yz	post variable component labels
Gamma_xz	post variable component labels
2 6 8	Tensor, 3x3 symmetric; Stress
SIGNOD	keyword
Stresses	post variable type label
Sigma_xx	post variable component labels
Sigma_yy	post variable component labels
Sigma_zz	post variable component labels
Tau_xy	post variable component labels
Tau_yz	post variable component labels
Tau_xz	post variable component labels
-1	stamp
-1	stamp

Table D.3: Post Variable List

Solution Status Parameter		
Increment	load factor	time
0	0.00	0.00

Table D.4: Solution Status Parameter

Mesh Size Parameter; No. of					
Nodes	Corner	Coord.	Elem.	Max. No.	FE application
Nodes	Nodes	Node	Elem.	nodes/elem.	arrays
8126	8126	3	3300	6	0

Table D.5: Mesh Size Parameter

Nodal Coordinates		
X	Y	Z
3.61	17.00	431.00
16.45	38.94	479.00
15.27	40.00	481.50
15.27	40.00	486.70
23.10	33.00	479.00
...

Table D.6: Nodal Coordinates

Element Connectivity (one for each element)							
Type	Group ID	Nodes					
25	1	311	24	309	3938	0	0
25	1	310	311	308	3939	0	0
25	1	8	310	307	3940	0	0
...
33	2	71	870	2885	72	923	3225
33	2	870	871	2886	923	924	3226
33	2	871	872	2887	924	925	3227
33	2	872	873	2888	925	926	3228
...

Table D.7: Element Connectivity

Displacements					
Displacement_u	Displacement_v	Displacement_w			
134.658	19.01	52.20			
138.585	75.48	5.05			
...			
Applied Forces					
Force_x	Force_y	Force_z			
7992.04	-601.90	-25.17			
0.00	0.00	-0.0287			
...			
Reactions					
Reaction_x	Reaction_y	Reaction_z			
7992.04	-6901.90	-25.17			
0.000000	-0.00	-0.028			
...			
Residuals					
Residual_x	Residual_y	Residual_z			
0.00	-0.000	-0.00			
0.000	-0.000	-0.00			
...			
Strains					
Epsilon_xx	Epsilon_yy	Epsilon_zz	Gamma_xy	Gamma_yz	Gamma_xz
7.48294×10^{-6}	2.53439×10^{-6}	-6.42840×10^{-6}	1.34533×10^{-6}	-3.568×10^{-6}	-9.4764×10^{-6}
-3.98371×10^{-6}	9.47294×10^{-6}	-9.32848×10^{-6}	9.35271×10^{-6}	-2.28563×10^{-6}	2.37651×10^{-6}
...
Stresses					
Sigma_xx	Sigma_yy	Sigma_zz	Tau_xy	Tau_yz	Tau_xz
-1848.5	-9549.66	1978.7	3823.2	-1299.803	3920.30
-1490.52	-1770.0	-588.2	1562.9	83801.585	5327.77
...

Table D.8: Nodal Post Variables

Appendix E

.eig Post Data file Format

.eig files contain the result of an eigenvalue analysis (typically dynamic, but possibly stability) and they can also be viewed by Spider which will animate the mode shapes.

eig files can be ascii or binary.

The file format is pretty straightforward:

1. 0
2. Number of Elements
3. Element connectivity
 - (a) Element_id, Element_Type, Node_1, Node_2,...
 - (b) ...
4. "Mode ", Mode_id, "Eigenvalue = ", ω^2
5. Eigenvector
 - (a) Node_id, x-coord_id, y-coord_id, [z_id], mode_shape_x, mode_shape_y, [mode_shape_z]
 - (b) ...

Note that there is no counter for the number of eigenvalues/eigenvectors.

Appendix F

.rtv Post Data file Format

.rtv file, or Real Time View file, store the nodal displacements, and some accelerations during a time-history dynamic analysis. The file can be viewed by Spider both during and after analysis. This feature enables the user to monitor a lengthy dynamic analysis (such as a 3D nonlinear one which may take a couple of days), and visualize both the displacements and the accelerogram.

1. Number of Nodes, Number of Elements, Number of time increments, Number of Accelerations being monitored.
2. Nodal coordinates
 - (a) Node_id, x-coord_id, y-coord_id, [z_id]
 - (b) ...
3. Element connectivity
 - (a) number_of_nodes, group_number, Node_1, Node_2,...
 - (b) ...
4. For each time step
 - (a) Time stamp (can be an ascii character such as T=1.03 sec.)
 - (b) Accelerations at each of the points (as fractions of g)
 - (c) Displacements at each of the nodal points.