

User instructions for Post3D beta release

(Draft version, under development)

Pete Olley and Rob Spares 25th May 1999
IRC in Polymer Technology/
Department of Mechanical Engineering
University of Bradford

General

Post3D is a 3D viewer designed for both fluid and solid finite element visualisation. It uses Mesa graphics, and can currently run under Linux or Windows 9x /NT operating systems. Development has been under Linux; the Windows port does not appear to have introduced any new bugs. Input file format is ASCII based.

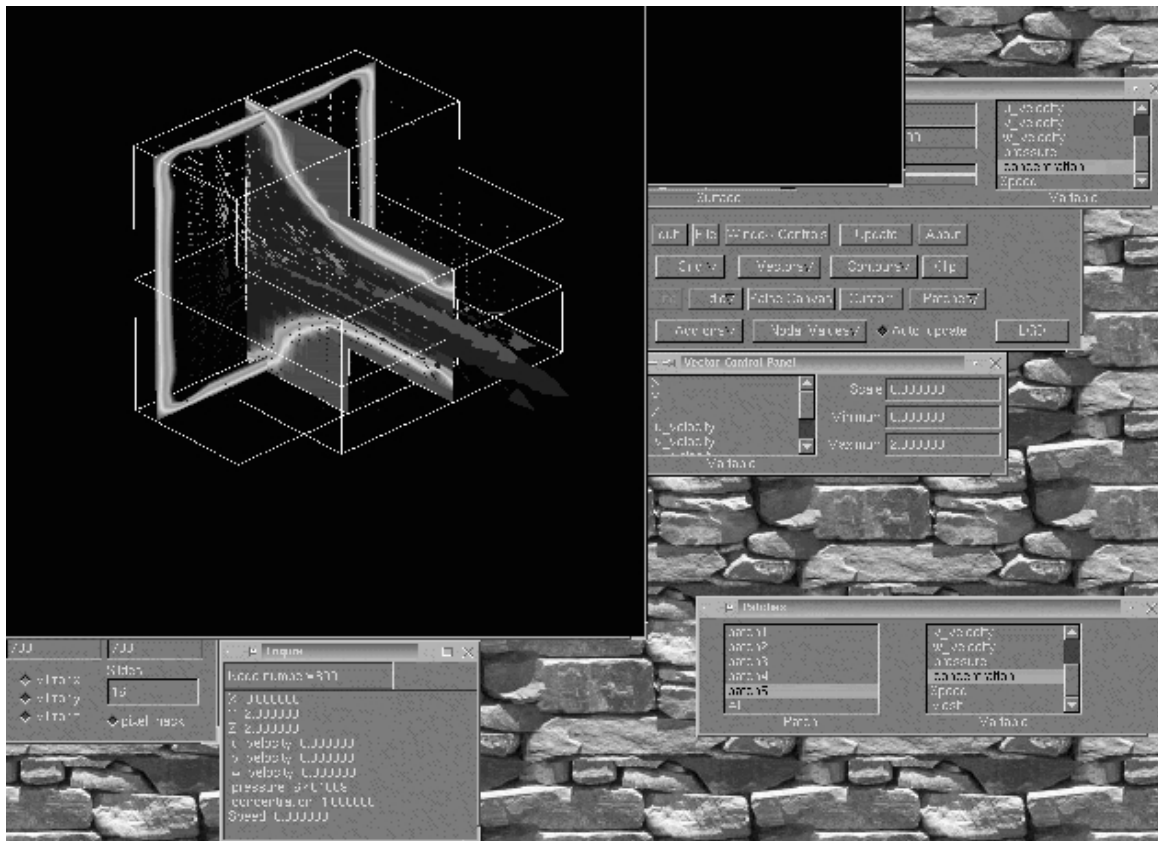


Fig. 1 Screen shot of viewer and control panels

Requirements

Intel Linux or Windows9x/NT operating system, 486 or above cpu, 16+ MB RAM. Open GL video card drivers must be installed for Windows version.

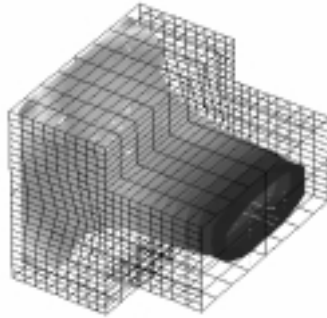


Fig.2 Arbitrary variable surface contours

Features

Fluid Solutions

Contours

Contour any variable on a surface defined by a constant value of any variable. (the latter will generally be x, y , or z but other variables can give a useful plot – see Fig.2 where pressure is plotted versus gas concentration)

Vector Plots

Vectors can be plotted, with colour dependant on any variable. Lengths, and features of the arrow heads can be varied (see under Custom, and Vectors on Main Panel)

Mesh

Full mesh, or just outline are available

Mirroring

Mirroring of solution about any ,or all of $x=0$, $y=0$, $z=0$

Patches

Contours, or mesh can be plotted on patches.

Detail viewing

Node numbers, element numbers , and nodal values of any solution variable can be viewed on screen.

Navigation

Image translation – drag with left mouse button

Image rotation – drag with right mouse button

Zoom in – press ↓ key

Zoom out – press ↑ key

Solid Solutions

Deformed meshes can be viewed, and contouring can be performed on the deformed, or undeformed mesh. The deformed mesh is deduced by the program by adding the first 3 solution variable (which are assumed to be x–displacement, y–displacement, and z–displacement respectively, to the nodal positions (which are assumed to be the initial nodal positions)

Miscellaneous – keyboard

'h' – output image to file (.png format)

'k' – show key to display colouring

'r' – raise the control panel

'e' – raise an enquiry window, normally followed by ...

'p' – ... which gives details of nodal values at the 'picked' node
(this feature works only on the non–deformed mesh)

The graphics window must have the mouse 'focus' for the above to be activated

Animation

Under 'IDLE' is a selection of contour animation, auto rotation , and (if a sequence of results files is available) animation from sequential files. A file name sequence that has been tested to work is 'res3D001.out' , res3D002.out', res3D003.out etc. Any sequence works, but it is its last continuous run of alphanumeric characters that is used.

The commands 'n' (next), and 'f' (first) automatically load the next or first file in a sequence.

.....

Input file format – (look at the example files before reading this)

Currently only 8–node 'brick' elements are supported – other element types can often be rearranged to fit this type. Node ordering is as follows:

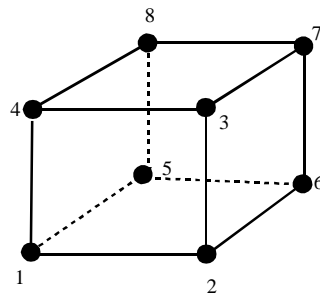


Fig. 3 Local node numbering for 8–node brick element

Two example files are given: *solid.example* and *fluid.example*

The file format is in ASCII : first some basic parameters are entered – number of nodes etc.

Then the nodal x,y,z co–ordinates of each node are entered in turn.

Then the element nodes are entered in turn, each preceded by 'npe', or nodes per element – currently only npe=8 is supported.

Title (anything up to 40 characters)

Element Type (only 'BRICK' supported at present)

Problem Type ('Fluid' or 'Solid')

Co–ordinate type (only 'Cart3d' supported at present)

number of nodes (nnum)

number of elements (nel)

number of variables	(nvar)
x1 y1 z1	(undeformed values always –see note at bottom)
x2 y2 z2	
.	
.	
.	
X _{nnum} Y _{nnum} Z _{nnum}	
npe node1 node 2	node _{npe} (start of element list)
.	
.	
.	
.	
.	(nel such entries)
'variable name1'	(eg x-velocity)
[symmetry vector]	(eg -1.0 1.0 1.0)
value at node 1	
value at node 2	
...	
.	
... value at node <i>nnum</i>	
'variable name2'	(eg y-velocity)
[symmetry vector]	(eg 1.0 -1.0 1.0)
value at node 1	
value at node 2	
...	
.	
... value at node nnum	
.	
.	
.	
.	
.	
'variable name <i>nvar</i> '	(eg pressure)
[symmetry vector]	(eg 1.0 1.0 1.0)
value at node 1	
value at node 2	
...	
.	
... value at node <i>nnum</i>	(anything)
number_of_patches	(an integer)
patchname1	(string)

patch_type	(set to 2 for element , face definition)
no_of_faces_in_patch	
element, face	(integer, integer)
.	
.	
.element, face	(repeat for no_of_faces_in_patch)
patchname2	(string)
patch_type	
no_of_faces_in_patch	
element, face	(integer, integer)
.	
.	
.element, face	(repeat for no_of_faces_in_patch)
.	
.	
.patchname _{no_of_patches}	(string)
patch_type	
no_of_faces_in_patch	
element, face	(integer, integer)
.	
.	
.element, face	(repeat for no_of_faces_in_patch)

General notes on format : the format will become clearer by studying the example files. The first 3 nodal variables are assumed to be U–velocity, V–velocity and W–velocity for fluids (usual notation), and are used as such for plotting velocity vectors. For solids, the first 3 nodal variables are assumed to be U–displacement, V–displacement, and W–displacement, and are used to calculate the deformed grid. The nodal positions which are given are assumed to be the original positions before any deformation took place . The 'symmetry vector' allows control over the colouring scheme when the mirroring options are used under CUSTOM button – eg u–velocity is symmetrical about the y and z–axes, but anti–symmetrical about the x–axis. If in doubt , or not interested, enter:

1.0 1.0 1.0

for the symmetry vector. Patch are useful for displaying mesh details, or contours of values on surfaces. Currently only 'element, face' format is supported which is denoted by 'patch_type'=2 .(patch_type=1 is reserved for a node list, and patch_type=3 is reserved for an element list – neither

yet implemented. This part is often the most fiddly part of the data file to output; it can be neglected by entering 0 for number_of_patches, and no more of the file will be read.

An element number and a face number define a segment of the patch. For 'brick' elements, faces go from 1 to 6. With reference to Fig.3, face numbers are defined as follows:

<i>Face number</i>	<i>Position in Fig.3</i>	<i>Nodes defined</i>
1	"bottom"	1-5-6-2
2	"right"	2-6-7-3
3	"top"	4-3-7-8
4	"left"	5-1-4-8
5	"front"	1-2-3-4
6	"back"	5-6-7-8