

Parameter Steering in Geomechanics

J Leng¹, L Margetts²

¹ *Manchester Visualization Centre, University of Manchester*

² *Manchester Computing, University of Manchester
(currently at the Universidad Politecnica de Madrid, Spain)*

December 2002

Synopsis

Here we use the term parameter steering to mean the process of interactively post processing and visualizing simulation data to allow a user to investigate the parameter space of the simulation.

Often a computational scientist will run a simulation under a variety of conditions and produce a number of data sets. It can be difficult to understand one data set but to understand how they compare to each other and fit into both parameter and solution space is challenging. Here we explore one possible method to allow scientists to explore the parameter space of a geomechanics problem. It is hoped that this method may reduce the number of data sets and hence simulations that are needed for the scientist to appreciate how the parameters interact to produce particular solutions.

1 Introduction

The idea of parameter steering comes from the culmination of 2 different technologies, computational steering and the visualization of time dependant data. Although very different techniques both are used to explore how the results of a simulation evolve over the time that the simulation runs.

1.1 Computational Steering

Typically simulations (large scale computations) run in batch mode. The scientist has little or no ability to intervene or guide the simulation. At best the scientist can monitor the simulation by writing out information as it progresses. With this approach

the most interaction a scientist can achieve is to prematurely terminate a computation if for example it reaches a steady state earlier than expected.

A computational scientist would prefer greater interaction for two reasons:

- In simulations where the physical model relies on either the convergence or divergence of the underlying mathematics, obtaining the appropriate convergence or divergence is difficult. For example a gyroscopic model has periods of chaotic behavior which the scientist may wish to steer away from.
- Interactively altering the parameters that control the behavior of a physical model and viewing the results allows the scientist a greater understanding of model.

Computational steering has been defined [2] as the capacity to control the execution of long-running, resource-intensive programs. This may include modifying program state, managing data output, starting and stalling program execution, altering resource allocations etc. Dynamic steering requires the user to monitor program or system state and have the ability to make changes. This could be through subroutine calls added as "instrumentation" or by interacting with the data structures in the code.

These systems are limited as they only allow the scientist to explore the data as they come out of the simulation. Some systems for example Grasparc [3], [4] have stop pointing and produce a tree that follows the path across parameter space that the user picked, stop pointed data forms the nodes of the tree. The scientist can compare results of different nodes on the tree. More advanced systems will allow the scientist to select data at a node and recommence the simulation down another branch and cover new areas of parameter space.

Parameter steering is one part of a scientific problem solving environment (PSE). SCIRun [4] is a good example of a PSE with good visualization capabilities. SCIRun is a scientific programming environment that allows the interactive construction, debugging and steering of large scale scientific computations. Using this "computational workbench," a scientist can design and modify simulations interactively via a dataflow programming model. SCIRun enables scientists to design and modify models and automatically change parameters and boundary conditions as well as the mesh discretisation level needed for an accurate numerical solution. As opposed to the typical "off-line" simulation mode - in which the scientist manually sets input parameters, computes results, visualizes the results via a separate visualization package, then starts again at the beginning - SCIRun "closes the loop" and allows interactive steering of the design and computation phases of the simulation. SCIRun has been optimised for efficient use with large data.

1.2 Time Dependant Data in Visualization Systems

In visualization systems data is read in as one single block of data, referred to as a field by AVS/Express [1]. A field contains all the information needed to process and render a 2D or 3D block of data, for example its coordinates and data values.

Often simulations write out data at various time steps of the computation. By using a visualization system to read in sequentially each data step the scientist can see how the data has evolved during the simulation. Time dependant data fields allow the visualization system to read in multi-step data and linear interpolation can be used to calculate interstep data values.

This method gives a better understanding of parameter space than reading a single step of data but still has limitations.

- Time steps act as a single path across parameter space. If more than one parameter affects the solution space then the ability to investigate this more complex problem is lost.
- The time step reader will read all the time steps and put them all in memory at the initial read. If the data sets are large this can slow down the execution of AVS/Express.
- If some components of the field like the coordinates are the same for each step then there will be multiple copies of the same piece of information held in memory.

1.3 Parameter Steering

Parameter steering allows the scientist to explore the parameter space of their simulations in a branching way like computational steering but also allows them to explore areas of parameter space not covered by the simulation by using linear interpolation.

Often computational scientists run a number of simulations as batch jobs. The results of these many batch jobs cover various parts of parameter space. It can be difficult for the scientist to remember precisely what areas of parameter space have been covered and understand the relationships between these various data sets. Parameter steering can be considered a data management tool or in a more general way part of a scientific problem solving environment like SCIRun.

2 Finite Element Analysis In Engineering

Generally Engineering applies scientific principles to the design and construction of engines, cars, buildings or machines. As computers have become more prevalent they have been used for to run computational simulations where the conditions of a particular situation are imitated. The Finite Elements Method is a computational method of solving mathematical equations on a mesh designed to fit the physical structure you have designed. This computational method is commonly used for engineering simulations.

The Finite Element mesh is generated to fit the shape of the system being investigated. Typically the mesh is made of:

- NODES a location in 2D or 3D space (termed coordinates in many visualization systems)
- ELEMENTS a geometric shape which has apexes and/or edges that lie on the NODES (termed cells in many visualization systems). An element also has a material type (termed cell data and cell group in many visualization systems)
- DISPLACEMENTS vector data (2D or 3D the same as the NODES) that shows how the geometry has deformed during the simulation, over a series of load steps (node data with a vector length of 2 or 3)
- SCALARS single value data like temperature or pressure (node data with a vector length of 1)

Visualization of FE data at The University of Manchester's Engineering Department (now moved to UMIST) has been adequately accomplished for the last 10 years on software produced by a one of their researchers Dan Kidger. He was also a key developer of FE codes so he included functionality that was important to him as a FE developer. The special features included:

- Uses the same data that is produced by the FE codes
- Special functionality allowed users to create and manipulate coloured contours on the object. The contours followed the same path as an isoline.
- Uses a lighting model that gives no shading
- Deals with displacements, mesh, reduction of vectors and node numbering in a particular way

This "home grown" product has been central to FE analysis at Manchester but recently original developer left. This researcher was solely responsible for all development, bug fixing and training. The system was not fully documented.

The engineers needed a new system that they could use immediately. Any new system would have to come with a full complement of visualization libraries and would have to read their data in directly.

AVS/Express was selected for this job. AVS/Express is a modular visualization system with large libraries. It has an extensive GUI which allows the user to develop networks of visualization functions. The results of the networks are displayed on an associated viewer. Advanced users can develop their own modules that are added to the original AVS/Express libraries, this functionality was used to develop the Manchester FE data reader. Finally AVS/Express can be run in a variety of virtual environments and with large data sets.

3 A Reader for FE Engineering Data

Fields are AVS/Express's internal 'data type' [1]. Fields hold all the arrays and information necessary to process and render 2D or 3D blocks of data. Most visualization systems have an internal 'data type' similar to this containing roughly the same information.

3.1 Field Data Hierarchy

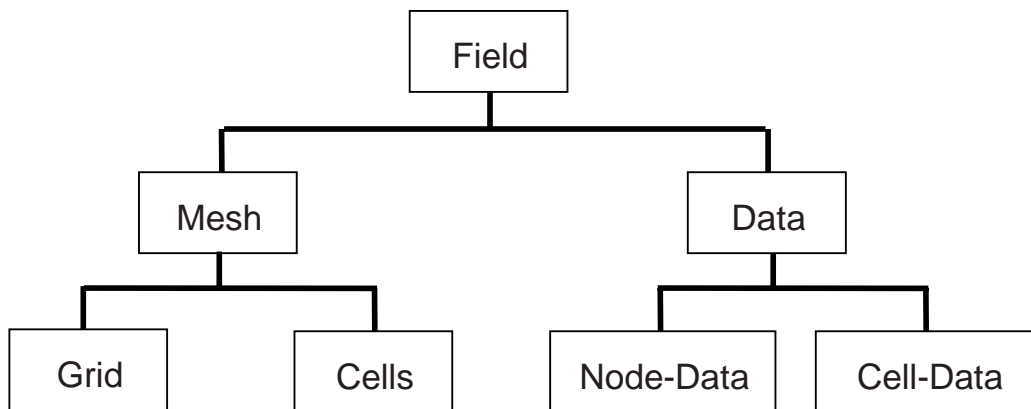


Figure 1: The structural components of a field.

Generally speaking a field has 2 major components:

- **Data** which can be either scalar or vector
- **Mesh** which defines the location of the data

The simplest type of field is a **uniform** field. A uniform field has a simple array of scalar or vector data values, node data. The mesh is implied from the extents of the data array and is of a grid type.

A **rectilinear** field is very similar to the uniform field. The data is array type node-data and the mesh is implied from the extents of the node-data array. However the node-data array can be stretched in space along each of the principal axes of the data.

A **Structured** or **Irregular** field is one that again has node-data but coordinate information is given for each axis or for each data point. The coordinates are given but the connectivity between the coordinates and the data are implied.

An **Unstructured** field is one where the data is not arranged in any particular pattern so that none of the spatial or connectivity information can be implied. The mesh is made up of **cells** which have a particular geometry like a triangle or a pyramid. The apexes of the cells are given in a coordinate array and the connectivity between

apexes must also be given. Second order cells are ones which have extra nodes half way between apexes, these points must also be given in the coordinates array. The data for an unstructured field can be given for each cell or for each coordinate node, i.e., node-data.

Finite element data can be mapped directly into cell data, an unstructured field. Often the FEM will use second order cell data which is not available in all visualization systems. Data could be interpolated into other field types but information would be lost. It is most accurate to keep the data in its native form.

3.2 Reading Fields

The easiest way for users to get their own data into AVS/Express is to use a field file descriptor. The field file descriptor is a simple text file that gives AVS/Express all the information it needs to read and map the data into a field. Simple information about the field is at the top of the text file. Pointers to the files and position of the data in the file are given below, at the end of the file. The **Read_Field** module is used within AVS/Express to process the field file descriptor and place the data in AVS/Express. Most visualization systems allow the users to read in data via a text descriptor file.

```
# AVS field file
#
ndim = ?
dim1 = ???
dim2 = ???
nspace = ???
veclen = ???
data = (byte, integer, float, double)
field = (uniform, rectilinear, irregular)
label = ???
unit = ???
variable n file = ??? filetype = ??? skip = ??? offset = ??? stride = ???
coord n file = ??? filetype = ??? skip = ??? offset = ??? stride = ???
```

Figure 2: The format of a field file descriptor.

Field file descriptors will work with uniform, rectilinear, structured or irregular fields. They do not work with unstructured fields, the type that can hold native FE data. Instead a UCD (Unstructured Cell Data) text file can be created which can be read in by AVS/Express.

The uses of UCD Files are of limited benefit for several reasons. Most importantly the UCD files originate from AVS 5 and have only been implement in AVS/Express for

backward compatibility. Documentation is included in the AVS 5 installation and is not complete. This makes the UCD format difficult to work with.

The UCD file has a text format. To create one all the data must be written directly as text into the file (no pointers to data files are used) and two copies of each data set one produced, one by the simulation and one in the UCD file. Another result of not using pointers is that if several data sets used the same coordinates, the coordinates must be written into each UCD file. Commonly in engineering FE codes one set of coordinates are used in a series of simulations. Using UCD files would mean duplicate copies of many data files (especially coordinate data) would be inevitable. The UCD format is a closed format and can not be extended to read binary data that is necessary when reading large files.

The ideal reader for the Engineering departments FE data would be similar to that of a field file descriptor. However this format would point to the data files already produced by the engineers' FE analysis. A module has been written that runs within AVS/Express and can read data from this format and map it into a field. The format and reader have been developed to deal with displacement data and node data in as simple a way as possible. The reader and the format are open to the user community so that in future adaptations can be made to meet new needs, like the use of binary data.

There are 2 formats for FE simulation data, .out or .key (which are the same format with but different file extensions) and .pl. The descriptor file is similar for each of these it is the reader in AVS/Express that must use a different mapping function for each type. The text file descriptor format is given below.

```
# Manchester FE Viz format
#
format = (.pl or .key)
dimensions = (2 or 3)
coord_nodes = ???
node_data = ?
labels =
displacements = ?
cell_groups = ?

coord n file=??? file_type=? skip=??? offset=??? stride=???
node_data n file=??? file_type=? n_vec=? data_type=? skip=??? offset=???
stride=???
displace n file=??? file_type=? skip=??? offset=??? stride=???
cell_group n label=? cell_type=? n_cells=??? file=??? file_type=? skip=?
```

Figure 3: The format of a field file descriptor for Manchester Engineers' FE data.

4 A Reader to Allow Parameter Steering

The field file descriptor has been extended to read more than just steps of data. The descriptor is a text file that holds information on each ‘block’ of data that can be read. All this information is then read into AVS/Express and stored as an internal database. The user can then select parameter values and the data around these values are read in and are used to calculate a linear interpolated data set to match the selected parameters. The linear interpolation used in this application has been defined in the appendix.

Information is stored about which values can be interpolated, which can not be interpolated and which components do not change.

- ❑ In geomechanics statistical models are generated for many different soil properties. These values can not be interpolated.
- ❑ A 3D block of data which results from a fluid flow simulation may have temperature and pressure as input parameters to the simulation. In this case there can be interpolation for both temperature and pressure.
- ❑ A scientist may use the same mesh for a number of simulations that cover different parts of parameter space. Shared components should be marked so they are only read when necessary.

The more parameters there are the more computationally expensive it is to calculate the values of the interpolated data, see the appendix.

```

# Manchester FE Viz format
#
format= (.pl or .out)
dimensions=???
coord_nodes=???
node_data=???
labels=???
displacements=???
cell_groups=???
steps=???
parameters=???
param_type ???
param_interp ???
param_name ???

step=???
params ???
coord ??? file=??? file_type=??? skip=??? offset=??? stride=???
node_data n file=??? file_type=? n_vec=? data_type=? skip=??? offset=???
stride=???
displace 0 file=??? file_type=ascii skip=??? offset=??? stride=???
cell_group ??? label=??? cell_type=??? n_cells=??? file=??? file_type=???
skip=???

```

Figure 4: The format of a field file descriptor for parameter steering Manchester Engineers' FE data.

5 Parameter Steering of a Geomechanics Using Finite Element Methods Code

A group of modules were written in AVS/Express to allow simple test data to be explored with parameter steering. This is proof of concept for the idea of parameter steering.

5.1 The Example

A cubic block of soil is subjected to a foundation load (at one corner of the mesh). The soil deforms elastically and the foundation undergoes a settlement that depends on the properties of the soil.

The elastic behaviour is governed by two properties, the modulus of elasticity, E , and the Poisson's ratio, ν . A series of analyses were run where the two parameters were altered in the way shown in the table below.

E	v
10, 000	0.2
10, 000	0.3
10, 000	0.4
1000	0.2
1000	0.3
1000	0.4
100	0.2
100	0.3
100	0.4

Table 1: The parameters used in the simulation.

5.2 The Field File Descriptor for the Results

Nine data sets result from this analysis. The basic mesh components (coordinates and cell groups) were the same for all analyses and were stored in a single separate file. This file was used directly by the analysis code and was read into AVS/Express through the field file descriptor. Each point in parameter space was denoted by a step in the field file descriptor file and the components that changed i.e. the displacements were read in for each step. The components that stayed the same i.e. the mesh were given by reference to the step where they were first used. This is shown in the exert from the field file descriptor given below.

```

# Manchester FE Viz format
#
format= .pl
dimensions=3
coord_nodes=4961
node_data=0
labels=
displacements=1
cell_groups=1
steps=9
parameters=2
param_type int float
param_interp yes yes
param_name E v

step=0
params 100 0.2
coord 0 c:\man_fe_work\data_param_steer\cube_coord.pl file_type=ascii
skip=2 offset=1 stride=3
coord 0 c:\man_fe_work\data_param_steer\cube_coord.pl file_type=ascii
skip=2 offset=2 stride=3
coord 0 c:\man_fe_work\data_param_steer\cube_coord.pl file_type=ascii
skip=2 offset=3 stride=3
displace 0 file=c:\man_fe_work\data_param_steer\disp_e3v2.pl file_type=ascii
skip=0 offset=1 stride=1
cell_group 0 label=stats cell_type=Hex2 n_cells=1000
file=c:\man_fe_work\data_param_steer\cube_coord.pl file_type=ascii
skip=4964

step=1
params 1000 0.2
coord step 0
displace 0 file=c:\man_fe_work\data_param_steer\disp_e4v2.pl file_type=ascii
skip=0 offset=1 stride=1
cell_group step 0

step=2
params 10000 0.2
coord step 0
displace 0 file=c:\man_fe_work\data_param_steer\disp_e5v2.pl file_type=ascii
skip=0 offset=1 stride=1
cell_group step 0

```

Figure 5: Extract from the example parameter steering field file descriptor.

5.3 Visualization of the Simulation Results

The results of the simulation were visualized to show the physical effects. The data formed a cubic block and was vector data showing the displacements for each data cell. The external surface of the data block was rendered semi-transparently and coloured to show the magnitude of the displacement. An arrow has been rendered to show where the load was placed. The colour on the slice plane confirms that the centre of the data block has similar data features to the block's surfaces. Three isosurfaces of displacement magnitude have been produced.

The variation of vector magnitude was not great enough for the isosurface module to calculate isosurfaces so the magnitude data was multiplied by a factor of 100000. This allowed the isosurfaces to be calculated for all 9 data sets. However the data range for the data with parameters $E=100$ and $\nu=0.2$ was 0-370 and with parameters $E=10000$ and $\nu=0.4$ was 0.0-3.2. The ranges were too different for one consistent isovalue to be used for all the data.

Figure 6 shows the visualization of the data set with parameters $E=100$ and $\nu=0.2$. The selected isovalues were 6.14, 53.2 and 108.

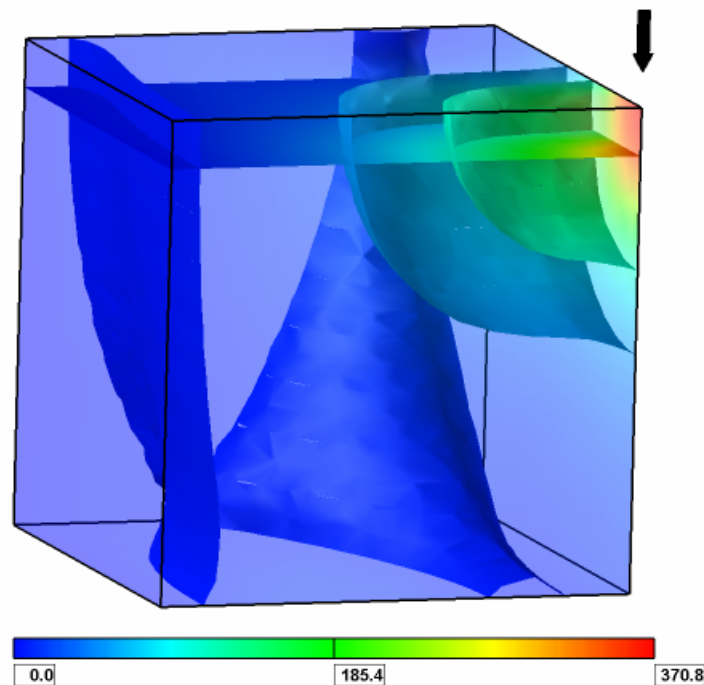


Figure 6: Visualization of the results of the simulation with parameters $E=100$ and $\nu=0.2$.

The results of the simulations varied in magnitude by a factor of 100. Although across each data set a range of similar shaped isosurfaces were produced. At high values a surface was created near the load that would have been roughly the shape of a hemisphere if the load had been central and not on the corner. Low isovalues, in the

lowest 5% of the data range, create a series of complex surfaces that move across the data block away from the load.

5.4 Comparative Visualization of the Simulation Results

Visualizing each data set separately gave the user little idea of how the parameters had altered the results. To show this a comparative visualization method would be most appropriate. A comparative method uses techniques that highlight the difference or similarity of the data values. For example a difference data set (produced by subtracting the data values in one block of data from those in the same location in another) gives high values at locations where there is much change. With this data a difference data set would be of little importance because the data ranges varied in magnitude by 100 and this difference would dominate any difference data.

Alternatively each data set could be normalised to the same data range but with this all data would “look” the same. Instead 3 isosurfaces were created at isovalues 110, 30 and 2.5. These would produce at least one isosurface for each data set but the similarity in data features would not be lost.

Visualization results are shown below across the whole of the parameter space. Here each cell of the table shows individual results of a simulation. The visualization is similar to the original method. The extents of the data are shown as a semi-transparent block coloured to show the difference of the magnitude of displacements. Three isosurfaces are produced again of the difference of the magnitude of displacements. For each image the three isovalues are the same but the shape of the isosurfaces alters across parameter space. Finally an arrow shows the location of the load in each case.

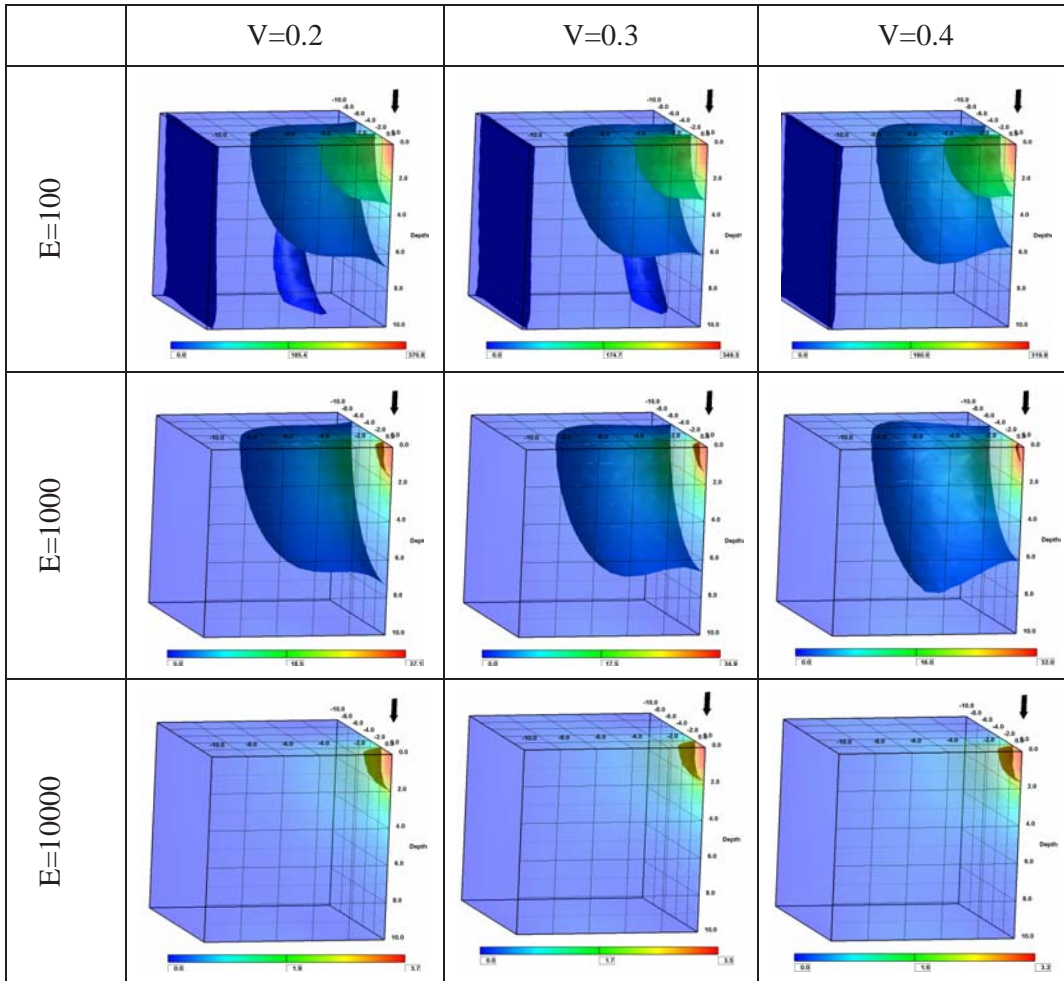


Table 2: Visualizations of the results across parameter space

These results give us some useful information about the parameters E and ν . The biggest differences in visualization results are seen between the rows of the table which indicates in this block of soil with these ranges of parameters it is the E parameter that has the largest effect on the results.

It would be interesting to know how the features in the top row developed into the features in the row below and we can use interactive interpolation to do this, we do not have to run new simulations. In this example we have picked the top left corner of the table (parameter space) to explore.

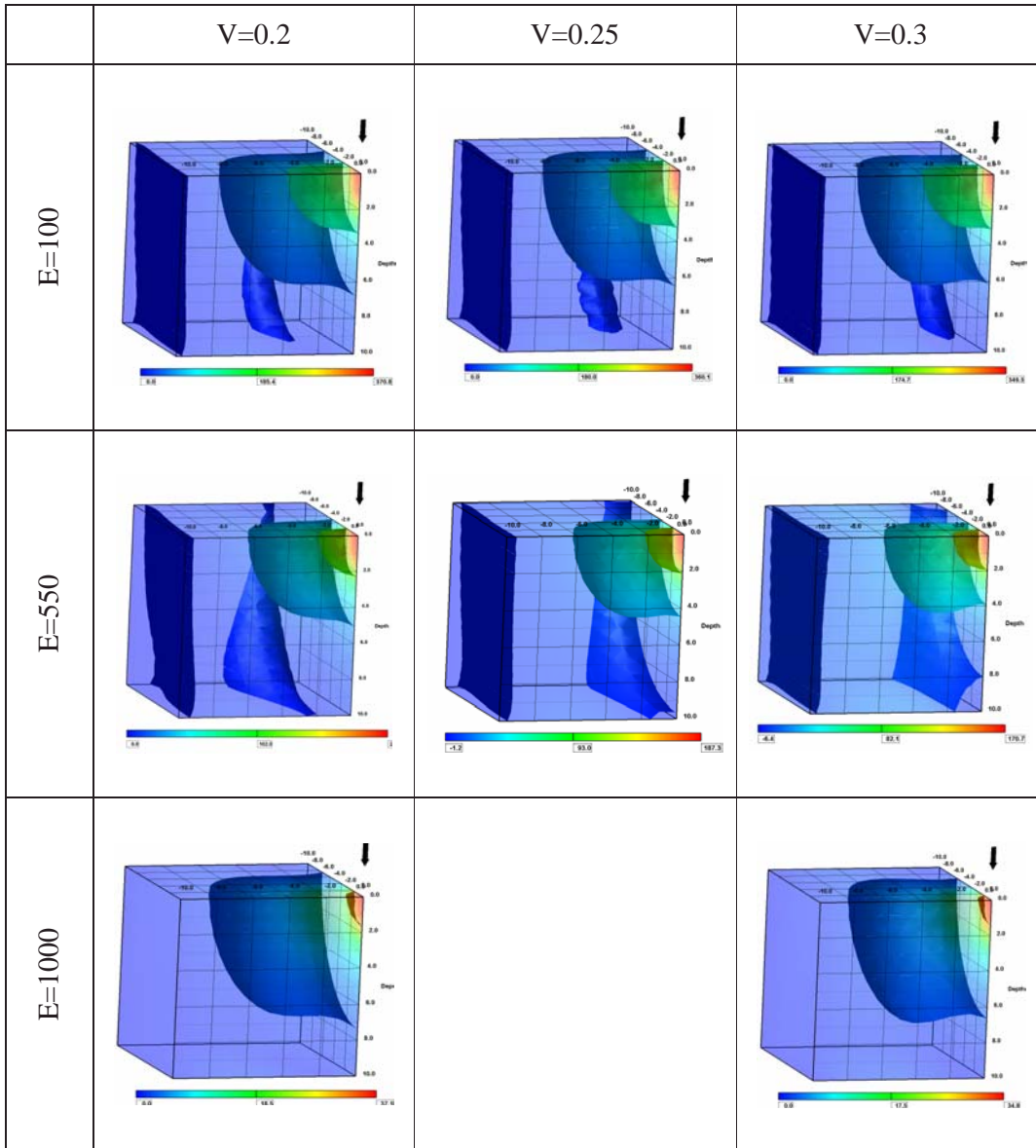


Table 3: Visualizations of the data shows how the results change across parameter space. Results in the central column and central row have been generated by interactive interpolation.

Results in the central column and central row have been generated by interactive interpolation. The cells of the table holding interpolated results show how the features seen in the original results alter in other parts of parameter space, something that the scientist would previously had to imagine.

To test the concept of parameter steering we have picked a feature that we can trace across parameter space. The feature selected is the isosurface of magnitude of displacement created at isovalue 30 when it passes through the 5.0 depth value below the load. Alternatively if we remember that each data set has been scaled by a factor of 100000 then we can say we are interested in parameter values that create data that

has a magnitude of displacement of 0.0003, at the middle depth value directly below the load.

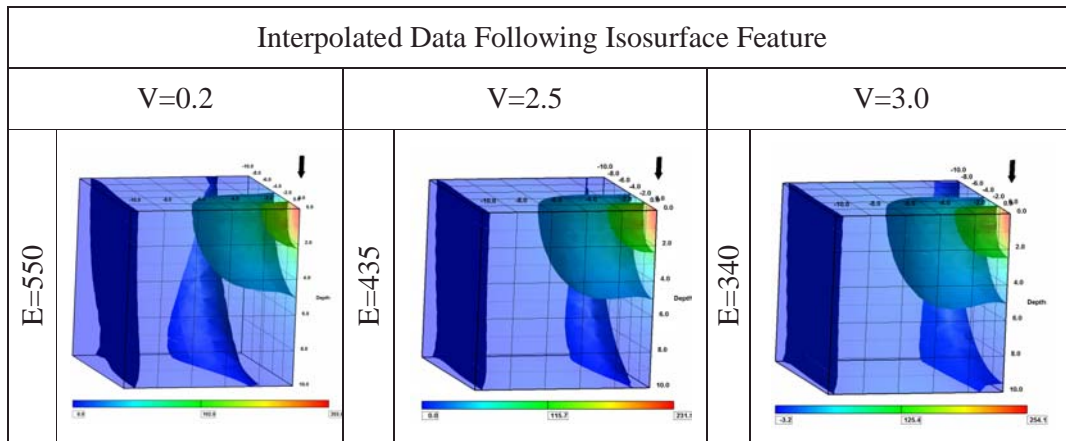
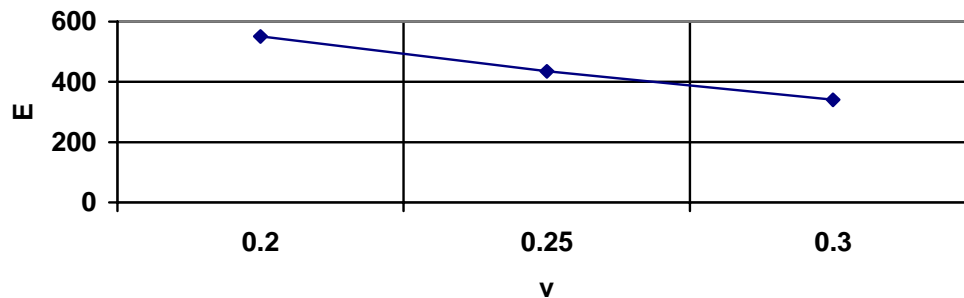


Table 4: Interactively interpolation of the data allows the user to find the values of E and v that create an isosurface of isovalue 30 which passes through the middle (5.0) depth under the load (below the arrow).

Graph Showing Values of E and v That Give the Desired Feature



Parameter steering allows scientists to explore parameter space without having to run a simulation for each point of solution space. This can help them:

- Understand the importance/dominance of each parameter within the range used
- Spot and trace the generation of importance features across different parameters
- Pick new parameters for new simulations to confirm results from parameter steering
- Manage their data

However it must be noted that important features may be missed if not enough points in parameter space are calculated or the features are weak.

The interpolation scheme used effects the results seen and is the part of the system that would give the best improvement. The interpolation function implemented now gives linear interpolation for 1 or 2 parameters. This could be improved to deal with more parameters relatively easily. However it may be useful to use quadratic interpolation that uses more than just the closest data for interpolation, for example in the case presented here there are 2 parameters interpolated and 4 data sets used for each interpolation but with quadratic interpolation the nearest 9 data sets in parameter space would be used. This would be however far more computationally expensive.

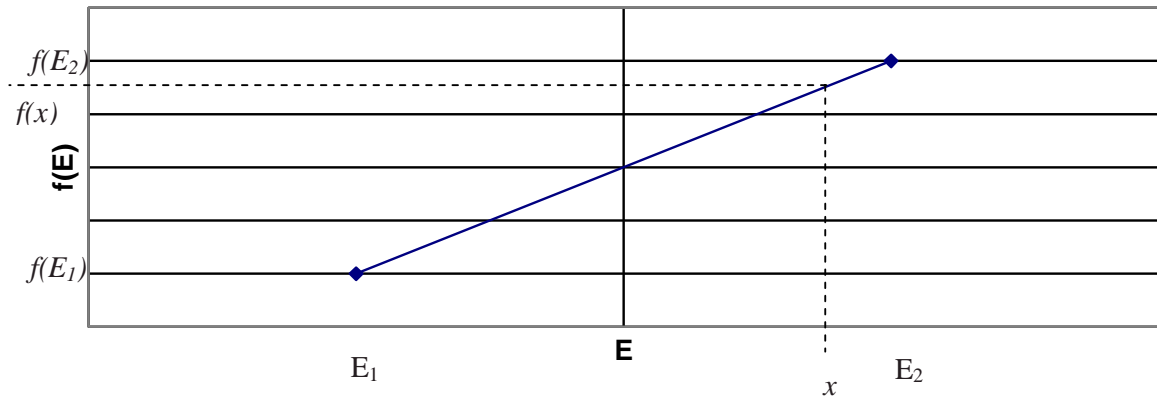
In this example parameter space is covered regularly. This simplifies the interpolation because 4 data sets can always be used for the interpolation. If irregular parameter space is to be covered then interpolation over 3 data sets needs to be included. For more complex parameter spaces better user interfaces are needed.

At the moment the system only interpolates node data for datasets of the same resolution and grid structure. Without interpolation the comparative functionality is still useful but in the future the interpolation function will need to be improved to deal with adaptive meshes or meshes of different resolution. There are other components of the field that could also be interpolated for example the coordinates or cell data, these interpolation elements could be added if/when appropriate.

6 Appendix

6.1 Linear Interpolation

Linear Interpolation in 1D



$$f(x) = f(E_1) + (x - E_1) * \frac{f(E_2) - f(E_1)}{E_2 - E_1}$$

For linear interpolation in 2D:

$$\begin{aligned} f(x,y) = & f(E_1, v_1) + (x - E_1) * \frac{f(E_2, v_1) - f(E_1, v_1)}{(E_2 - E_1)} \\ & + (y - v_1) * \frac{f(E_1, v_2) - f(E_1, v_1)}{(v_2 - v_1)} \\ & + (x - E_1)(y - v_1) * \frac{\{ f(E_2, v_2) + f(E_1, v_1) - f(E_2, v_1) - f(E_1, v_2) \}}{(E_2 - E_1)(v_2 - v_1)} \end{aligned}$$

References

- [1] Advanced Visual Systems Inc., “AVS/Express Developer’s Reference, Release 3.0”
- [2] R Allen, “Parallel Application Software on High Performance Computers. Survey of Computational Grid, Meta-computing and Network Information Tools”, technical report available see URL: <http://www.cse.clrc.ac.uk/Activity/HPCI>
- [3] K Brodlie, L Brankin, A Poon, G Banecki, H Wright, Alan Gay, “GRASPARC - A Problem Solving Environment Integrating Computation and Visualization”, Proceedings of IEEE Visualization 1993 Conference
- [4] S Parker, C Johnson, “SCIRun: A Scientific Programming Environment for Computational Steering”, Proceedings of Supercomputing 95, ACM
- [5] H Wright, K Brodlie, Tim David, “Navigating High-dimensional Spaces to Support Design Steering”, Proceedings of IEEE Visualization 2000 Conference