Sandia National Laboratories
Operated for the U.S. Department of Energy by
Sandia Corporation

Albuquerque, New Mexico 87185-0557

date: January 12, 2006

to: J.M. Redmond, 1524
Distribution

from: D. Todd Griffith, 1524

subject: User's Guide for mat2exo: A program for writing Matlab mat-file data to Exodus II format

Enclosure(1) details the user's guide for a program which translates mesh data from Matlab mat-file format to Exodus II format. This tool, mat2exo, is the inverse of the commonly used tool exo2mat which translates Exodus II data to the Matlab mat-file format. These tools provide a means for pre-processing an Exodus II model file or post-processing an Exodus II results file using Matlab.

A large number of SNL analysts use the Matlab software package for post-processing of data because it provides for easy access and clear understanding of the data, and contains an abundance of data processing commands as well as a high-level programming capability. Typically, an output Exodus II database from a code such as Salinas or Presto is translated to Matlab format using the exo2mat translation tool. This permits the information in the Exodus II database to be loaded into the Matlab environment for data interrogation, manipulation, comparison, or many other post-processing needs. Sometimes, it is desired that the data manipulated in Matlab be written back to the Exodus II format so that post-processed calculations of the full mesh results can be visualized using tools such as Ensight. One example of this is the subtraction of two sets of nodal variable solutions in Matlab, and subsequent writing of this new result back to Exodus II so that the difference between the two nodal solutions can be visualized. The subject of this note is a tool for translating from Matlab format to Exodus II format, mat2exo.

(page intentionally left blank)

Enclosure(1)

User's Guide for mat2exo: A program for writing Matlab mat-file data to Exodus II format

D. Todd Griffith
Structural Dynamics Research Department
Sandia National Laboratories
Albuquerque, New Mexico 87185-0557

January 12, 2006

USAGE

mat2exo *file.mat*

where *file.mat* is a Matlab data file which contains the mesh information and results data to be converted to Exodus II format. The output is a file named *file.exo* saved in the working directory. The Exodus II format is well documented in Reference 1. It is expected that mat2exo will be added to the SEACAS system of tools on the LAN.

INTRODUCTION

A large number of SNL analysts use the Matlab software package for post-processing of data because it provides for easy access and clear understanding of the data, and contains an abundance of data processing commands as well as a high-level programming capability. Typically, an output Exodus II database from a code such as Salinas or Presto is translated to Matlab format using the exo2mat translation tool. This permits the information in the Exodus II database to be loaded into the Matlab environment for data interrogation, manipulation, comparison, or many other post-processing needs. Sometimes, it is desired that the data manipulated in Matlab be written back to the Exodus II format so that the full post-processed calculations of the mesh results can be visualized using tools such as Ensight. One example of this is the subtraction of two sets of nodal variable solutions in Matlab, and subsequent writing of this new result back to Exodus II so that the difference between the two nodal solutions can be visualized. The subject of this note is a tool for translating from Matlab format to Exodus II format, mat2exo.

The original versions of exo2mat and mat2exo were written by Mazen Tabarra and Richard Naething, respectively, though much modification has been made since.

SPECIAL USAGE NOTES

It is expected that mat2exo will serve as the inverse translation tool for exo2mat. In a typical mat2exo conversion, only a limited number of the variables from the original Exodus II file will be modified with the vast majority of the variables such as those related to the mesh geometry (nodal locations, number of elements, element types, etc) left unchanged. Therefore, the typical bi-directional conversion path should involve:

1) generating an Exodus II database from a meshing code such as Cubit or a simulation code such as Salinas or Presto,
2) converting it to Matlab mat-file format using exo2mat,
3) loading the mat-file into the Matlab software environment,
4) manipulation of the variables of interest in the Matlab workspace,
5) clearing the workspace of variables not part of the Exodus environment and saving all the remaining variables in the workspace to a new Matlab mat-file,
6) converting the new Matlab mat-file to Exodus II using mat2exo.

This path preserves the static information in the database with relative ease. For example, typically it is desired that the geometry or mesh information remain unchanged and available for the inverse translation back to Exodus II. It is preferred that no extraneous variables to the original Exodus file be loaded into the Matlab workspace; therefore, the workspace should be cleared of all variables

prior to loading the mat-file. There is no limitation on the type of manipulation of the workspace variables of interest; however, when the new mat-file is saved the workspace should contain variables whose names, dimensions, and data types are unchanged from the original mat-file – a set of conditions which mat2exo assumes in order to execute properly (a listing of these variables is given in Tables 1-3). Typically, the only changes to take place are the numerical values for the workspace variables of interest; however, in some special cases such as database size reduction, which will be discussed later, do involve removing some workspace variables.

On the other hand, there is no requirement that this bi-directional path be followed. As long as all of the variables expected by mat2exo are located in the mat-file to be converted, mat2exo can be executed successfully. An example of this could be the generation of Exodus II database information originating within Matlab using a specialized code.

Additionally we note that saving the database information to a new file in Matlab is quite simple. All workspace variables can simply be stored in the binary file *newfile.mat* using the following command:

>> save *newfile.mat*

UPDATES MADE TO EXO2MAT

In this development, some additional coding was performed to update the exo2mat database converter. The prior version did not fully translate side set and node set information, thus these updates were made to preserve this information in the exo2mat step so that it would exist at the mat2exo stage. Of course the sideset information is now also available for manipulation while the database is resident in a Matlab session. As mentioned in the previous section, it is imperative that variables contained in the Matlab mat-file have the proper variable name, dimension and datatype for the mat2exo converter to execute properly. This is not a significant limitation because in a typical transformation only a few variables will be modified. Furthermore, updates made to exo2mat and mat2exo must be consistent in order for the bi-directional translation to execute properly. The most recent versions of exo2mat and mat2exo, which are listed below in the revision history section, should be used together. However, mat-files created using previous versions of exo2mat can be translated to Exodus II using mat2exo as long as side set and node set information is ignored. In Matlab, simply set the the number of side sets (nssets) and the number of node sets (nnsets) to zero and save the mat-file. As an aside to this point, any group of data can be ignored in the mat2exo step by setting the group count to zero; as another example, set the variable “nevars” to zero so that no element variables will be written back to Exodus II.

Before proceeding to some of the uses of mat2exo, we look at the outputs of the updated exo2mat program. In the following set of tables, we list the variables which are written by exo2mat and provide the Matlab variable name, a description of the variable, and its data type. Essentially these tables provide a list of variables available for processing and/or manipulation. We group these variables according to the following categories: 1) Mesh and geometry information, 2) Side set and node set information, and 3) Results information. The mat-file variables are presented in this fashion in order to better detail the types of manipulations possible which generally will fall into the categories of mesh/geometry-related, loading characteristics or boundary conditions, or results. The mesh/geometry related information and the side set/node set information listed in Tables 1 and 2

respectively are contents of a mesh file such as those generated by Cubit. The results information of Table 3 is added to the database once this mesh file is used for an analysis run.

Table 1. Description of Mesh/Geometry Variables Contained in Mat-file

Matlab variable name	Description of Variable	Variable Type
Title	Database title	Character array
blk01, blk02, ...blkN...	Lists nodes for each element in Nth element block	Double array
blkids	Lists identification numbers associated with each block	Double array
blknames	String containing block names (typically this is the type of element in each block)	Character array
elem_num_map	List of numbers associated with the elements	Double array
naxes	Number of coordinate axes, problem dimension	Double array
nblks	Number of element blocks	Double array
nelems	Number of elements	Double array
nnodes	Number of nodes	Double array
node_num_map	List of numbers associated with the nodes	Double array
x0, y0, z0	Nodal coordinate values	Double arrays

Table 2. Node Set and Side Set Variables Information Contained in Mat-file

Matlab variable name	Description of variable	Variable type
nsids	Node set ID list	Double array
nnsets	Number of node sets	Double array
nnsnodes	Number of nodes in each node set	Double array
nsfac01, nsfac02, ….nsfacN…	Distribution factors for Nth node set	Double array
nsnod01, nsnod02, ….nsnodN…	List of nodes for Nth node set	Double array
nnsdfac	Number of distribution factors in each node set	Double array
ssids	Identification numbers associated with the side sets	Double array
nssets	Number of side sets	Double array
ssfac01, ssfac02, ….ssfacN…	Side set distribution factors for the Nth side set	Double array
ssnod01, ssnod02, …ssnodeN…	Node list for Nth side set	Double array
ssnum01, ssnum02, …ssnumN…	List of number of nodes in each element of the Nth side set	Double array
sselem01, sselem02, ….sselemN…	List of elements in Nth side set	Double array
ssside01, ssside02, ….sssideN…	Side numbers of each element in Nth side set (specifies sides corresponding to sselemN)	Double array
nssdfac	Number of distribution factors in each side set	Double array
nsssides	Number of sides in each side set	Double array

Table 3. Nodal, Elemental, and Global Variable Results Contained in Mat-file

Matlab variable name	Description of variable	Variable type
enames	String containing the names of the elemental results variables	Character array
evar01, evar02, evarN…	Elemental variable solution values for the Nth elemental variable	Double array
gnames	String containing the names of the global results variables	Character array
gvar01, gvar02, …gvarN…	Global variable solution values for the Nth global variable	Double array
info	String containing information records pertaining to the model or analysis code used	Character array
nelems	Number of elements in model	Double array
nevars	Number of element variables	Double array
ngvars	Number of global variables	Double array
nnames	String containing the names of the nodal results variables	Character array
nnvars	Number of nodal variables	Double array
nvar01, nvar02,….nvarN…	Nodal variable solution values for the Nth nodal variable	Double array
nsteps	Number of time steps (number of modes for EigenAnalysis)	Double array
time	Time step values (mode numbers for EigenAnalysis)	Double array

EXAMPLE USES OF MAT2EXO

The primary intention for writing mat2exo is to provide a means of storing some numerical results values from a mesh file modified within Matlab into an Exodus II database for the purpose of visualization; however, there are many uses when considering the numerous core capabilities of Matlab and specialized analyst-written Matlab codes for use in pre-processing or post-processing of a mesh file. Some examples include:

1) Algebraic manipulation of results data (e.g. scaling)
2) Comparison of multiple mesh data sets (e.g. differencing)
3) Digital filtering of results (e.g. low-pass filtering)
4) Database size reduction (e.g. isolating a particular set of results)
5) Special geometry modifications/mesh distortion (e.g. changing the nodal locations)
6) Modifications for spatially dependent loadings (e.g. modifying side set distribution factors)
7) Processing and visualization of experimental data (e.g. writing test mode shapes for visualization)

In the following sections, two examples are presented to demonstrate some useful modifications to the Exodus II database within Matlab. The first example demonstrates a *pre-processing* modification for defining loads which vary spatially, and in the second example a *post-processing* example is demonstrated whereby a transient solution solved using an analyst written Matlab code is written to Exodus II for visualization.

EXAMPLE #1. PRE-PROCESSING: SPECIFYING SPATIALLY DEPENDENT LOAD FACTORS ON A SINGLE SIDE SET

A common requirement on a static or dynamic loading is that it vary in some fashion with position. For example, a hydrostatic pressure loading on a submerged structure will vary linearly in the coordinate describing the depth below the surface. More generally, the loading may require that its distribution vary with an arbitrary function of the spatial coordinates, $f(x, y, z)$. The following example demonstrates how to define an arbitrary loading distribution in Matlab over a single side set or node set.

When a mesh is created using a program such as Cubit, only the information in Tables 1 and 2 related to mesh/geometry information and side set/node set information is created. It is at this point that one would desire to modify the load distribution factors in advance of the analysis run. Firstly, the mesh file will need to be converted to Matlab mat-file format using exo2mat (i.e. exo2mat modelfile.exo). Secondly, the mat-file (modelfile.mat) will be loaded into the Matlab workspace so that the new load factors can be computed.

The mat-file will contain the side sets and node sets which were defined in the meshing program. The load distribution factors, at this point, will all be set to one because this is the default of the meshing program. In this example, a new side set will be created using the to-be-modified side set as a template so that the original side set with unit load factors can be preserved for a future analysis. We are only interested in changing the load distribution factors, so we can simply copy the other required information (as described in Table 2) to the new side set variables. Firstly, we add one additional side set to the side set count as follows:

```
>> nssets = nssets +1;
```

and give the new side set a unique identification number (#100), for example, if we are adding a 7th side set

```
>> ssids(7) = 100;
```

Now, we look at copying some information that will not change in creating the new side set. This information describes the nodes, elements, sides in the side set. In this example, we are only changing the distribution factors for side set #01, so we copy the following information to the new side set #07.

```
>> ssnod07   = ssnod01;
>> ssnum07  = ssnum01;
>> sselem07 = sselem01;
>> ssside07 = ssside01;
>> nssdfac(7) = nssdfac(1);
>> nsssides(7) = nsssides(1);
```

The only remaining variable that must be specified for side set #07 is ssfac07, which contains the distribution factors as a function of position. The operations required to create these distribution factors are quite simple in Matlab. We simply need to identify the coordinates associated with the nodes in the side set ('X07','Y07', and 'Z07'), and then compute the distribution factors according to these nodal coordinates.

```
>> X07 = x0(ssnod01);
>> Y07 = y0(ssnod01);
>> Z07 = z0(ssnod01);

>> ssfac07 =  X07.*Y07.*sin(Z07);
```

We note that if a side set or node set is only to be modified without creating a new one, then the previous four lines of code are sufficient to modify those values. Of course, additionally the analyst may be interested in normalizing the distribution function (by dividing by the max value of the function to ease the scaling of loads in the analysis code). The additional considerations above are only needed to create a new side set. Here, we have arbitrarily chosen a distribution governed by $f(x, y, z) = xy\sin(z)$ as an example, but any function can be chosen.

It should also be noted that the above example assumes the typical case where the nodes are numbered sequentially in the "node_num_map" variable such that the coordinates of node 100; for example, are given by the 100th element of the nodal coordinate variable vectors ('x0','y0','z0'). If this were not the case one would need to do a simple search to find the proper coordinates of the nodes in the side set.

Finally, we clear the unnecessary variables, and save the new mat-file.

```
>> clear X07 Y07 Z07

>> save newmodelfile.mat
```

The new Exodus II model file will result from a call to mat2exo (i.e. mat2exo newmodelfile.mat) and the file newmodelfile.exo will be created.

Some other methods exist at SNL to perform this task which include specialized analyst written codes for modifying the distribution factors or using "field" specifications in Patran. It is expected that this path will be simpler, offering more capabilities because of the power of Matlab and the full bi-directional capability contained in exo2mat and mat2exo.

EXAMPLE #2. POST-PROCESSING: MANIPULATION AND STORAGE OF RESULTS VARIABLES FOR VISUALIZATION

The second example demonstrates a post-processing feature. Suppose that one desires to perform a transient analysis on a system using a solution method which is not available in a code such as Salinas. Here, the analyst is forced to write a specialized code to solve the problem off-line (using Matlab, for example). Suppose that it is desired to visualize this set of calculations. In this example, it is demonstrated how one can store a set of calculations (in the context of a transient structural dynamics problem) in an Exodus II database for visualization.

In this example, we desire to apply a new technique in a transient solution process -- a model size reduction technique enabling a solution to be accomplished by solving many fewer degrees of freedom. The reduced order model is developed using full order system matrices and dof/node mapping from Salinas and is solved using an analysis code in Matlab.

Once solved, we would like to visualize the displacement solution of the new reduced order model, and for the purpose of evaluating the reduction technique we desire to visualize the difference between the solution of the full order problem and the reduced order problem. There is no need to go into any details about the model size reduction technique, but it should be noted that a special transformation is needed to map from the reduced order system coordinates back to the A-set (Salinas Analysis set) coordinates and then from the A-set to the G-set (Global set) coordinates for storage in the Exodus II database. The method by which the G-set coordinates are reconstructed from A-set coordinates is reported in a separate document [2]. For more discussion on the A-set and G-set see References 3 and 4.

In our Matlab solution process, we deal with all nodal displacement solutions in G-set coordinates. These include three sets of solutions: 1) the full order system solved by Salinas, 2) the full order system solved in Matlab (just as a check of the dof/node mapping), and 3) the reduced order system solved in Matlab. Additionally, we compute the difference between solutions 1) and 2) and the difference in solutions 2) and 3). Thus we actually have 5 sets of solutions for nodal "displacements" which consist of a total of 15 nodal variables (x,y, and z displacements for each solution). The Exodus II output database from Salinas will serve as a template – this file can be translated to mat-file format using exo2mat. Here we will be manipulating variables listed in Table 3, in this case nodal variables. The variables nvar01, nvar02, and nvar03 in the template file already contain the x, y, and z displacements for the full order Salinas solution. All of the nodal variables have dimensions of number of rows equal to the number of nodes in the model and number of columns equal to the number of time steps solved. Firstly, we set the total number of nodal variables to 15 in Matlab:

```
>> nnvars = 15;
```

And, we must store our newly computed variable solutions resulting from the Matlab solution for the full order system

```
>> nvar04 = DispX_Aset;
>> nvar05 = DispY_Aset;
>> nvar06 = DispZ_Aset;
```

and, the newly computed variables for the reduced order system:

```
>> nvar07 = DispX_Reduced;
>> nvar08 = DispY_Reduced;
>> nvar09 = DispZ_Reduced;
```

The differences (or error solutions) are stored as follows using simple matrix subtraction capability in Matlab:

```
>> nvar10 = nvar01 - DispX_Aset;
>> nvar11 = nvar02 - DispY_Aset;
>> nvar12 = nvar03 - DispZ_Aset;

>> nvar13 = DispX_Aset - DispX_Reduced;
>> nvar14 = DispY_Aset - DispY_Reduced;
>> nvar15 = DispZ_Aset - DispZ_Reduced;
```

Finally, we save the workspace variables to a new mat-file and convert to Exodus II using mat2exo. Now, the solutions computed in Matlab can be visualized in a program such as Ensight including those of a reduced order model, and the differences in the nodal displacements. A key to this process is being able to map the results to the proper coordinates, the G-set. As mentioned previously, the details regarding mapping coordinates from A-set to G-set is described in a separate document [2].

To demonstrate this feature, Figures 1 and 2 show a comparison of the displacement solution at a particular time step for the Salinas solution and the reduced order solution in Matlab, respectively. Aside from the reduced order method providing an accurate solution, we note that mat2exo provides the means to assess visually the behavior and accuracy of the off-line solution. Figure 3, shows the difference in these solutions.

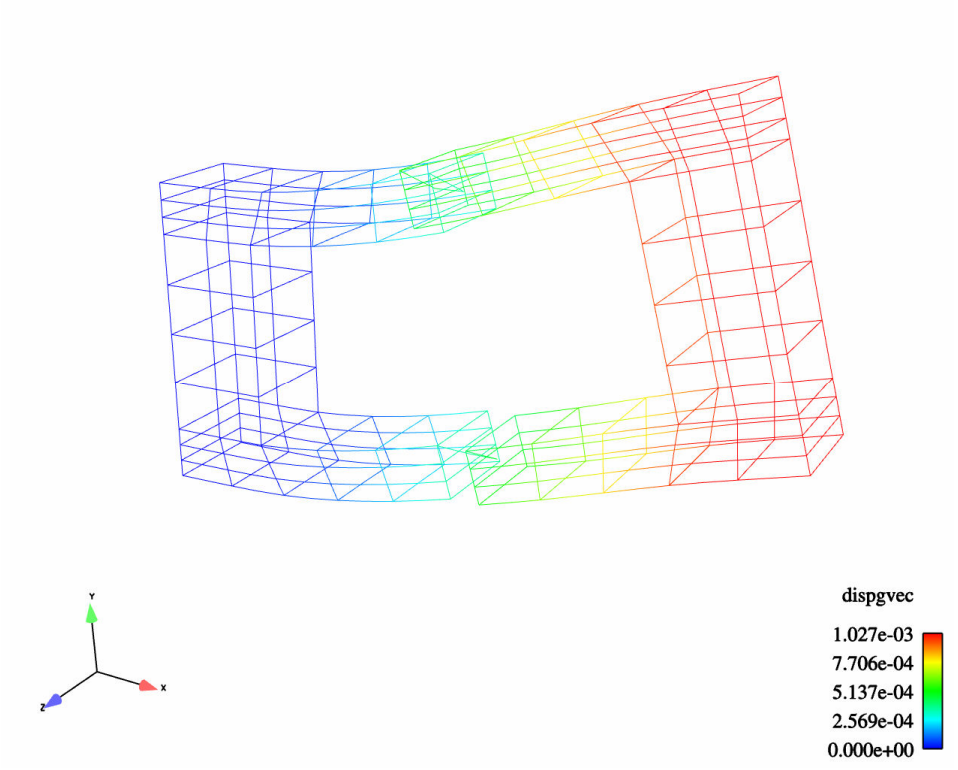

Figure 1. Displacement solution resulting from Salinas run (timestep = 72)

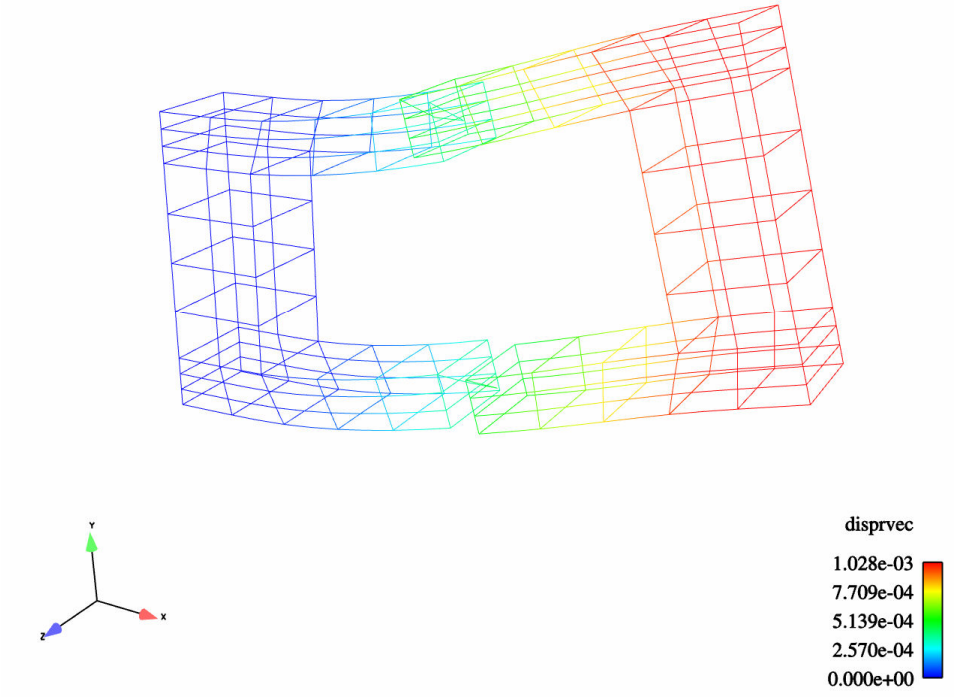

Figure 2. Displacement solution of reduced order model resulting from Matlab code (timestep = 72)

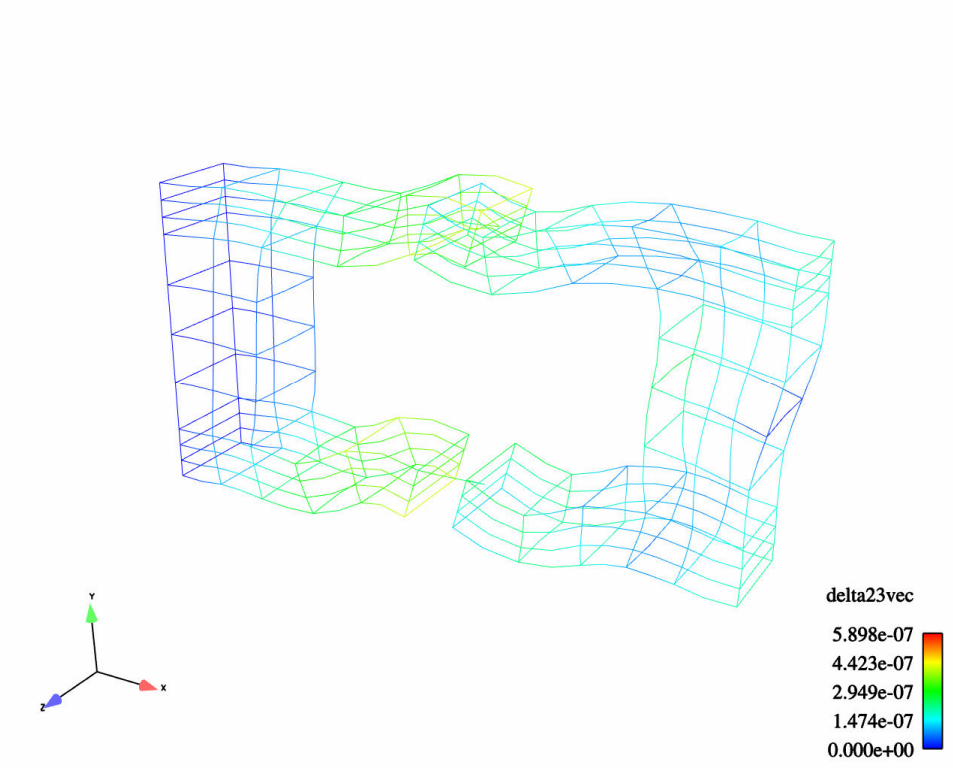

Figure 3. Visualization of error solution computed in Matlab for solutions shown in Figures 1 and 2

In the following section, we note how to rename these nodal variables by modifying the "nnames" variable. This is essentially a bookkeeping feature to be used to assign names to the variables so that when loaded into a visualization program, it is clear which variables are being viewed. For example, in Figures 1, 2 and 3 we have renamed the variables in Matlab as can be seen above the legend.

NOTE ON CHANGING RESULTS VARIABLE NAMES

In some cases, the manipulation of nodal, elemental, and global variables will result in the desire to rename the newly constructed variables. For example, when the displacement nodal variables ('DispX', 'DispY', 'DispZ') from two mesh files are differenced, we may wish to name the new quantities, "DeltaX", "DeltaY", and "DeltaZ". This name change/update can be accomplished in Matlab by modifiying the "nnames" string, in this example. Other examples may involve updating "blknames", "gnames", or "enames".

The mat2exo code requires that the character strings associated with each variable name are separated by line breaks in order to identify the individual variable names from the single stored string array. In Matlab, it is a simple matter to create this invisible line break in the following way:

```
>> linebreak = sprintf('\n');
```

where the notation should be very familiar to C programmers. For the nodal variables example above we would define the following:

```
>> nnames = ['DeltaX',linebreak,'DeltaY',linebreak,'DeltaZ',linebreak];
```

Of course, this can be automated (and should be for the case when many variables are to be named) using string manipulation functions in Matlab.

Additionally, it is noted that for a 3D problem, the vector variable for a triad of matching variable names ending in "X", "Y", and "Z" will be formed automatically in visualization programs such as Ensight. Thus, in this example the variable "DeltaVec" will be formed automatically if this convention is followed for naming of 'DeltaX', 'DeltaY', and 'DeltaZ'.

REVISION HISTORY FOR MAT2EXO AND EXO2MAT

Prior to these developments, exo2mat existed as a nearly fully mature code in its translation ability with a significant amount of coding performed in the past several years. Additional coding noted in this memo was made only to provide exo2mat with the ability to fully write side set and node set information to Matlab, and subsequently the ability to translate this information back to Exodus II using mat2exo. A significant amount of coding had been accomplished for the mat2exo code only in the core file input/output functions as well as the translation of the mesh and geometry related information. However, no results information was translated, and of course no side set or node set information was translated due to the previously mentioned deficiencies of exo2mat in writing this information.

The following summarizes the recent known revision history for mat2exo and exo2mat.

mat2exo:

$mat2exo, Version 1.1, 8/21/2003 by Richard Naething
- Core file input/output functionality
- Mesh/geometry information

$mat2exo, Version 1.2, 9/7/2005 by D. Todd Griffith
- Results information
  - Global, nodal and element variable names
  - Writes global, nodal and element variable results
- Writes complete set of time steps (Version 1.1 skipped first step)

$mat2exo, Version 1.3 (in SEACAS and Salinas repository), 12/16/2005 by D. Todd Griffith
- Complete node set information
  - node set numbers,
  - node set distribution. factors, etc
- Complete side set information
  - side set numbers
  - side list and element list numbers
  - side set distribution factors, etc.

exo2mat:

$exo2mat, Version 1.4 (in SEACAS repository), April 2003 by Garth Reese
- Core file input/output functionality

- Mesh/geometry information
- Results information

$exo2mat, Version 1.5 (in SEACAS repository), August 2003
- Minor modifications

$exo2mat, Version 1.6(in SEACAS repository), September 2004
- Provisions made for Exodus files containing no side set distribution factors

$exo2mat, Version 1.7(in SEACAS repository), 1/12/2006 by D. Todd Griffith
- Complete node set information
  - node set numbers,
  - node set distribution. factors, etc
- Complete side set information
  - side set numbers
  - side list and element list numbers
  - side set distribution factors, etc.
- Side set distribution factors written as double precision instead of integers
- Conditional provision made for Exodus file with no side set distribution factors

ACKNOWLEDGEMENTS

Dan Segalman provided several motivating examples which lead to the implementation of some of the key features written into the mat2exo program. Dan also carefully reviewed this document. Rick Naething provided a previous version of mat2exo, as well as much appreciated assistance in issues related to linking the code with the Matlab and ExodusII libararies. Garth Reese provided a previous working version of the exo2mat code, and provided assistance in making available the updated mat2exo code in the Salinas Team software repository. Jerry Rouse and Tim Walsh provided valuable feedback from the use of an earlier version of the mat2exo code. The SEACAS Tools Team, in particular Greg Sjaardema, is also acknowledged for helping to track versions of exo2mat and to make updates to their software repository.

REFERENCES

[1] Schoof, Larry A., and Yarberry, Victor R., EXODUS II: A Finite Element Data Model, SAND92-2137, November 1995.

[2] Griffith, D. Todd, "Reconstructing G-set nodal variables from A-set nodal variables and constraint relations," Internal Memo, Sandia National Laboratories, January 2006.

[3] MSC/NASTRAN Primer, pgs. 142-143.

[4] Cook, R. D. and D. S. Malkaus, M. E. P., *Concepts and Applications of Finite Element Analysis*, John Wiley & Sons, third edn., 1989, Chapter 9.1-3.