Introduction to CU-BEN

CORNELL UNIVERSITY

J. W. Kosianka
W. Wu
H. M. Reed
C. J. Stull
C. J. Earls

April 13, 2018

Contents

1 Overview of CU-BEN

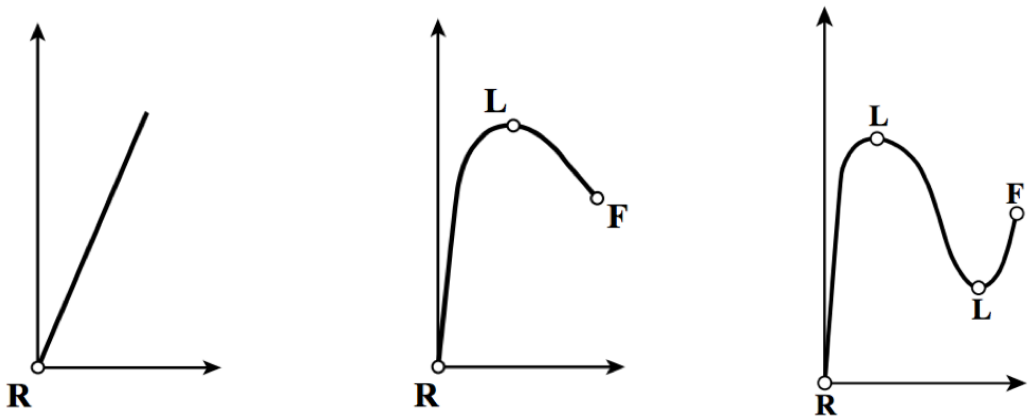
CU-BEN is a 3D nonlinear transient dynamic implicit finite element (FE) solver with linear, acoustic fluid-structure interaction (FSI) capabilities. The motivation for its creation was to create a fast, efficient, and accurate dedicated hull modeling FE software package for use in solving inverse problems and for use in other applications where many calls to a structural modeling tool are required. Sparse storage schemes enhance the analysis speed and storage efficiency of this computational structural dynamics (CSD) solver. CU-BEN also exploits high-order structural elements suitable for specialized force-space plasticity models in order to streamline the treatment of material nonlinearity. (A discussion on force-space plasticity can be found in the Course Notes described in Section 3.) Additionally, CU-BEN is free and open source; thus, allowing researchers and practitioners to modify the code directly in accordance with their needs.

1.1 Analysis Types

The analysis options available in CU-BEN are (see Figure 1) 1st order elastic (i.e. "linear elastic"), 2nd order elastic (i.e. "geometrically nonlinear"), and 2nd order inelastic (i.e. "geometrically and materially nonlinear"). Linear elastic analysis assumes infinitesimal strain, infinitesimal displacement, small rotation, and a linear stress-strain relation. Geometric nonlinearity refers to a nonlinear strain-displacement relation. Material nonlinearity refers to a nonlinear constitutive relation. CU-BEN also allows for acoustic FSI based on G. C. Everstine's approach of extending solid, continuum element formulation for use with acoustic fluid. More information can be found in [4]. The desired analysis will influence the choice of solution method.

1.2 Solution Methods

The solution algorithm options available in CU-BEN are all based on the Direct Stiffness (DS) method: Linear-elastic analysis with a single factorization of the system stiffness matrix (DS), the Newton-Raphson (NR) method, the Modified Newton-Raphson (MNR) method, the Modified Spherical Arc Length (MSAL) method, the Linear Newmark Implicit Integration method, and the Nonlinear Newmark Implicit Integration method (the latter two to be used for transient dynamic analyses). The DS method is used for static, 1st order elastic analysis. For static 2nd order analysis, both elastic and inelastic, the NR, MNR, or MSAL methods can be used within CU-BEN. The NR method operates through a load increment scheme controlled by user-defined convergence criteria. The NR method then begins predictor-corrector iterations, to minimize the residual force vector, until a user-specified



(a) (b) (c)

Figure 1: Idealized single degree of freedom load – displacement response curves showing different response patterns: (a) Linear response with the assumption that the structure exhibits perfect linear elasticity for any deformation; (b) Nonlinear response showing the softening of a structure entering the plastic regime; (c) Nonlinear response showing snap-through, or the softening and subsequent hardening of the structure, creating an unstable region on the equilibrium path between the limit points; (L represents a limit point on the response curve, F represents the failure point, and R represents the reference load).

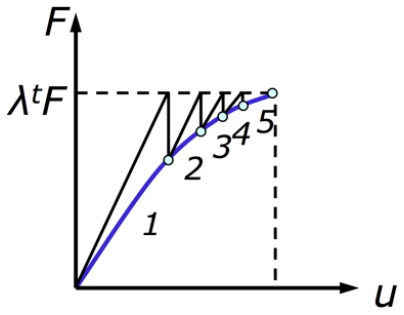


Figure 2: Schematic of Newton-Raphson method. Displacement is denoted as u, load as F, load proportionality factor as λ^t, and iterations as the numbers 1-5.

convergence measure is satisfied. A schematic of this approach is shown in Figure 2, where the tangent stiffness matrix, K^T is represented as the changing slope that corresponds with the load increment iterations, within the load increment shown. The MNR method differs from the NR method in that K^T is only calculated once within a given load increment; whereas for the NR method, K^T is updated at every predictor-corrector iteration. The MNR method may be expedient when repeated computation of the tangent stiffness matrix is deemed expensive, but this must be balanced against the likely requirement for a smaller load increment size within the MNR context. A shortcoming of the NR methods, in general, is their inability to traverse limit points in the equilibrium path, such as those shown in Figure 1(c). The MSAL method can be employed in such situations.

The implicit Newmark time integration method is implemented for both linear and nonlinear transient dynamic analyses. The Newmark method in CU-BEN applies to 2^{nd} order elastic nonlinear cases by implementing NR sub-iteration during each time step, to ensure the solutions at each time step satisfy the user-defined convergence criteria when nonlinear internal force vectors are employed at the element-level. To effectively implement the 2^{nd} order inelastic behavior in the transient dynamic case, an automatic, adaptive time step is implemented within CU-BEN. If the solution, within a prescribed time step, is unable to achieve equilibrium convergence, with the specialized force-space plasticity model, then the time step will automatically be decreased, and the load will linearly interpolate in accordance to that new time step. This time step reduction will repeat until the solution converges or until the minimum allowable time step is reached. More information about these methods can be found in the Course Notes described in Section 3, but it is important to emphasize that the user defined time histories, as specified within the model_def.txt input file, are respected during the solution. However, while it is that every user specified time step in the load history is contained within the ultimate CU-BEN solution, there may be additional time steps, included in between the user specified values, to allow for satisfaction of the numerical tolerances required for nonlinear equilibrium convergence. It is also pointed out that additional implicit, transient dynamic solution schemes are available in CU-BEN. As these additional methods introduce artificial numerical damping, they are described in Section 1.4.

1.3 Element Types

Discussion of CU-BEN element types will be restricted to the truss, frame, and shell structural elements. The fluid and solid bricks are formulated for use in acoustic FSI analyses, but are not discussed directly here, refer to [4]. Figures 3 - 5 contain graphical representations of the different structural element types. Degrees of freedom (DOFs) with single arrowheads denote translational DOFs, and those with double arrowheads denote rotational DOFs. For the formulation of the stiffness matrices for all elements, please refer to the Course Notes described in Section 3. All elements are intended for use in the updated Lagrangian reference frame, except for the truss element: the truss is formulated in a total Lagrangian context. The reference frame formulations respect the fact that all elements in the CU-BEN element library are suitable for use in large displacement, small strain response regimes, except the truss: the truss is a large displacement, large strain element. All elements require the specification of different sets of material and geometric properties for their use in CU-BEN; these input requirements will be described in Section 5 (they are also outlined in the initial comment block at the start of the main.c source file). The 3D nonlinear, large displacement, large strain truss element, shown in Figure 3, has translational DOFs in the 1-, 2-, and 3- directions at each node. The 3D frame element, shown in Figure 4, extends the truss element formulation to include the Bernoulli-Euler description of flexural kinematics for specifying rotational DOFs in the 1-, 2-, and 3- directions at each node, in addition to the displacement DOFs. The triangular shell element

is formulated by superimposing membrane effects over the top of a plate bending formulation employing explicit Kirchhoff thin plate kinematics. This results in a Discrete Kirchhoff theory (DKT) triangular shell element with translational and rotational DOFs in the 1-, 2-, and 3- elements at each node, shown in Figure 5. The DKT shell element is suitable for large deformation analysis, but not finite strain, as the thickness is not updated throughout the solution process and therefore the Green-Lagrange strain is not treated exactly within the formulation. Many more details about the element formulations and solution strategies within CU-BEN may be found in the class notes of C. Earls (available at http://earls.cee.cornell.edu/software/).

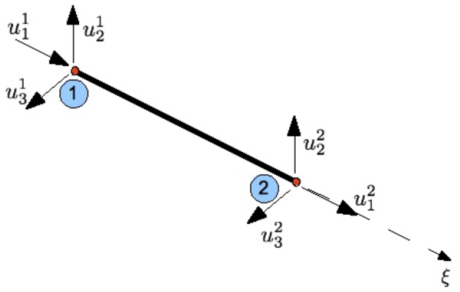


Figure 3: 3D 6-DOF space truss element.

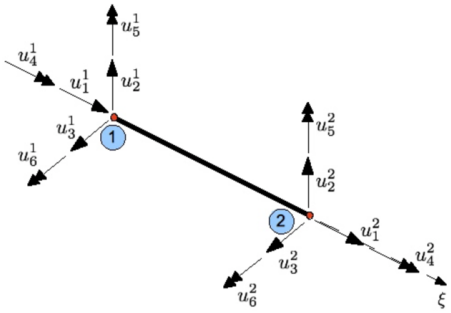


Figure 4: 3D 14-DOF space frame element, not shown is an additional warping DOF at each end.

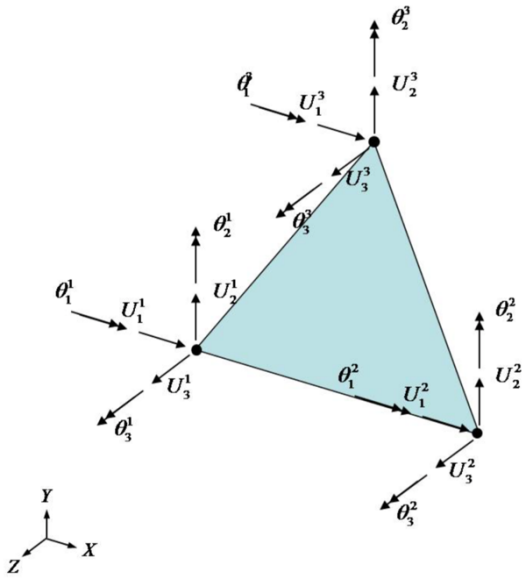


Figure 5: 3D 18-DOF DKT triangular shell element.

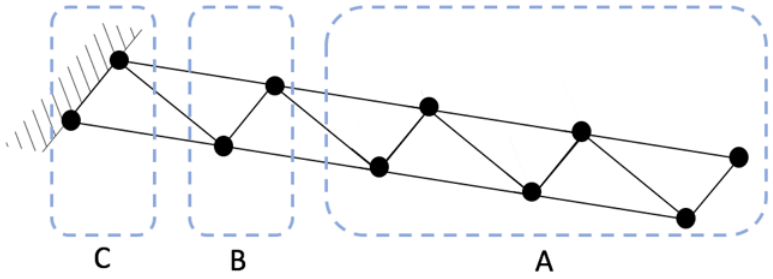


Figure 6: A cantilever beam meshed with triangular shell elements. Interior nodes, connecting nodes, and boundary nodes are partitioned in boxes A, B, and C respectively.

$$\begin{bmatrix} \bar{K}_{aa} & \bar{K}_{ab} \\ \bar{K}_{ba} & \bar{K}_{bb} \end{bmatrix} \begin{bmatrix} u_a \\ u_b \end{bmatrix} = \begin{bmatrix} \bar{q_a} \\ \bar{q_b} \end{bmatrix}$$

Figure 7: Matrix partitioning scheme. The stiffness matrices of interior nodes, connecting nodes, and boundary nodes are denoted as $\overline{K_{aa}}$, $\overline{K_{ab}}$, and $\overline{K_{bb}}$, respectively. The unknown displacement vector and prescribed displacement vector are denoted as u_a and u_b. The effective load vectors at interior and boundary nodes are denoted as $\overline{q_a}$ and $\overline{q_b}$.

1.4 Numerical Damping Schemes

Additional time integration schemes in CU-BEN include: Generalized-α; Hilber-Hughes-Taylor (HHT); and Wood-Bossack-Zienkiewicz (WBZ). These time integration scheme introduce artificial, numerical damping whose intensity is controllable by the user. Such numerical damping is useful when high frequency responses, that are spatially under-resolved within a given FE mesh, lead to pollution of the numerical solution - ultimately precipitating solution instability. Such solution instabilities can frequently arise in coupled physics contexts, such as fluid-structure interaction simulations (FSI). More information about the algorithm can be found in the [2].

1.5 Nonzero Boundary Conditions

Nonzero boundary conditions can be imposed on any translational DOF within CU-BEN for transient dynamic analyses. However, the imposition of these dynamic displacement boundary conditions poses two numerical concerns. Firstly, the formulation of such system results in a singular stiffness matrix - solving for displacements becomes infeasible as the stiffness matrix is not invertible. To circumvent analyzing this singular matrix, the effective stiffness matrix of the system is partitioned into submatrices consisting of interior node DOFs, boundary node DOFs, and connecting node DOFs as shown in Figures 6 and 7. The system of linear equations in Figure 7 is then rearranged to solve for unknown displacements.

Secondly, while the boundary nodes with prescribed displacement conditions may be translating globally, they are understood as fixed conditions locally, maintaining their position relative to neighboring nodes throughout the time history - i.e. local deflection does not occur at the translating boundary. This proves a challenge, as the locally fixed nodes have DOFs in the mass matrix that will contribute to the calculated nonlinear dynamic response for the complete structural mesh. The inclusion of these DOFs may induce high-frequency numerical noise when computing internal forces, degrading the accuracy of the solution. To remedy this issue, the mass contributions for nodes with nonzero boundary conditions are scaled by a factor of 10^6 to increase the inertia locally in the mesh, such that these nodes are treated as if fixed.

1.6 Analysis Restart

Checkpoint and restart capabilities are available in CU-BEN for transient dynamic analyses. The checkpoint flag signals to CU-BEN to back up information periodically and the restart flag signals to restart analysis from the

last saved time step in the time history. These functions could save computational cost tremendously should there be a hardware failure or power outage.

2 Downloading

CU-BEN can be downloaded from GitHub at https://github.com/nonlinearfun/CU-BENs or from http://earls.cee.cornell.edu/software/. It will be maintained on GitHub and all development will be documented through descriptive commits. CU-BEN will be continually updated on the GitHub repository, with date stamps on the files updated. The version released at the time this tutorial was written is CU-BENs v4.0. All CU-BEN files should be contained within one folder in order to run effectively. Throughout this document, all tutorials will assume the GitHub repository folder has been downloaded to the Desktop.

3 Documentation

The primary documentation of CU-BEN is the extensive commenting throughout the source code. Associated with every block of code is a detailed description of the theory behind the algorithm applied. An example of this is shown in Figure 8. The commenting is shown in gray. The header block in main.c describes the analysis and algorithm options available in CU-BEN, as in Section 1, and all details of usage. In addition to the use of comments, an unofficial "theory manual" of sorts is available at http://earls.cee.cornell.edu/software/. This theory manual was written by C. Earls as course notes for "CEE 7790 - Nonlinear Finite Element Methods: Structures" at Cornell University. It is sufficient for the majority of the theory behind the code in CU-BEN, but it is not complete (e.g. acoustic FSI and Newmark methods are not included in the course notes). Much of the code is based on theory and algorithms described in [1] and [3]. A more in-depth description of force space plasticity can be found in [5]. Information on the acoustic FSI capability can be found in [4]. Information on the implemented damping scheme can be found in [2].

```
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
```

```
// If load proportionality factor exceeds maximum, set equal to maximum
if (lpf > lpfmax) {
        lpf = lpfmax;
}
/* Set all temporary variables and variables which refer to the structure
 in its current configuration to values obtained at last successful
 load increment; this step is required so as not to overwrite structure
 properties prematurely if load increment is unsuccessful / invalid */
for (i = 0; i < NEQ; ++i) {
        /* Compute generalized total external load vector, accounting for
         generalized fixed-end load vector */
        qtot[i] = q[i] * lpf;
        /* Store generalized internal force vector from previous
         configuration */
        fp[i] = f[i];
        d_temp[i] = d[i];
        f_temp[i] = f[i];
}
```

Figure 8: Lines 1605-1622 of main.c of CU-BEN. These lines of code are within the Newton-Raphson method, initiating load incrementation.

4 Compiling to Obtain CU-BEN Executable

After downloading CU-BEN, it is necessary to compile the code to obtain a CU-BEN executable. This is accomplished by running the command "make" in the folder downloaded from GitHub. All files must be contained in one folder for the code to compile correctly. GNU Compiler Collection, or GCC, must be installed on the computer running CU-BEN. The compiler will create "object files" with the file extension ".o" for each *.c file contained in the downloaded folder. It will link these object files together in the form of an executable, in our case "ben.exe". Running the command "clean" will delete "ben.exe" and all *.o files. The program need only be compiled once for use with varying input files. If any updates or edits to the *.c files are made, the user will

need to re-compile the program. It is pointed out that the currently available make file in the GitHub repository for CU-BEN is intended for compilation on a Mac running OSX 10.11; Windows and Linux systems will require modifications to the make file, to respect the architecture being targeted for the build. Specifically, CU-BEN allows for the use of the BLAS and LAPACK solver suites, and so BLAS and LAPACK will be needed on the system running CU-BEN, and the path to the BLAS and LAPACK installations will need to be specified in the LIBS flag at the top of the CU-BEN make file. If compiling on a Linux system, the LBLAS library will need to be installed and specified. If compiling on OSX, one may face an error stemming from the Accelerate framework. This will not compile with newer versions of the gcc C compiler. This will compile correctly on gcc version 4.2.1, but may error out when compiling with newer versions. If this is the case, the user can comment out the "#include <Accelerate/Accelerate.h >" line in main.c and solve.c. This can be done as shown in Figure 9.

The following tutorial is provided for those compiling on a Windows system.

Figure 9: Cygwin's main page.

To install Cygwin terminal for PC, first go to Cygwin's main page: https://cygwin.com/install.html. Click on "setup-x86.exe" or "setup-x86_64.exe" to download the executable file for Cygwin.

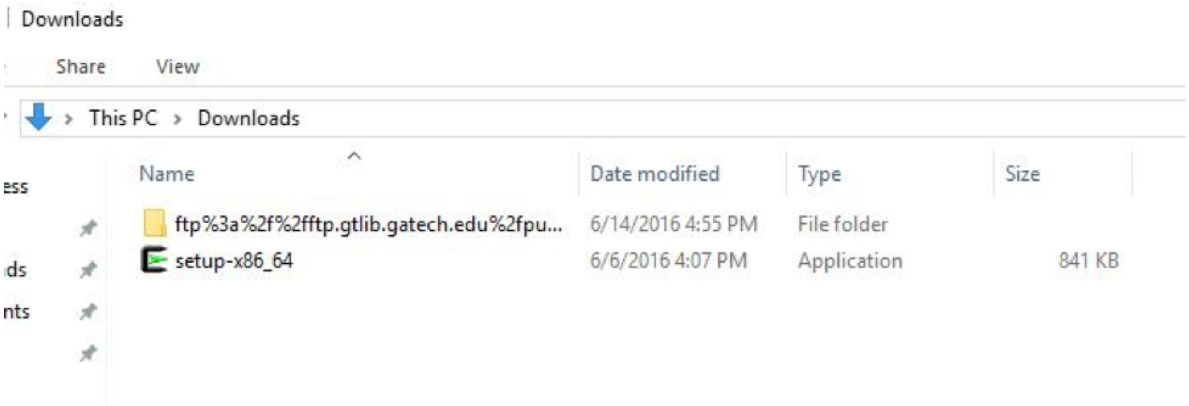


Figure 10: Cygwin executable file.

Go to "Downloads" directory where the executable file located and double click on "setup-x86_64" to set up Cygwin terminal, as shown in Figure 10.

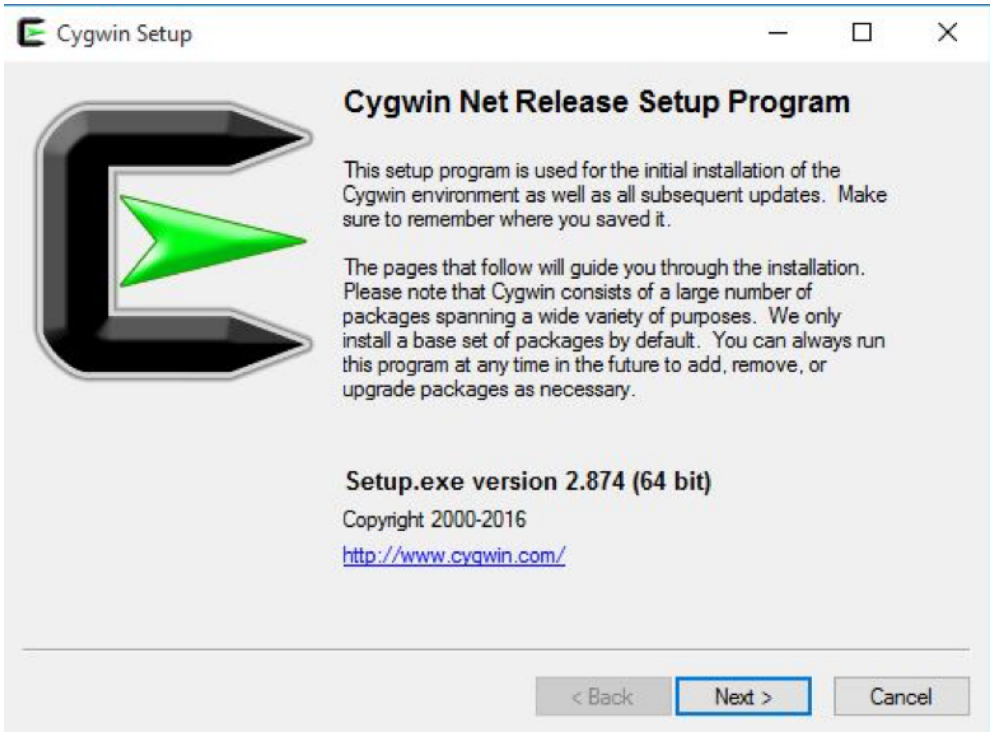


Figure 11: Cygwin setup window.

Click "Next" once Cygwin's setup window pops up, as shown in Figure 11.

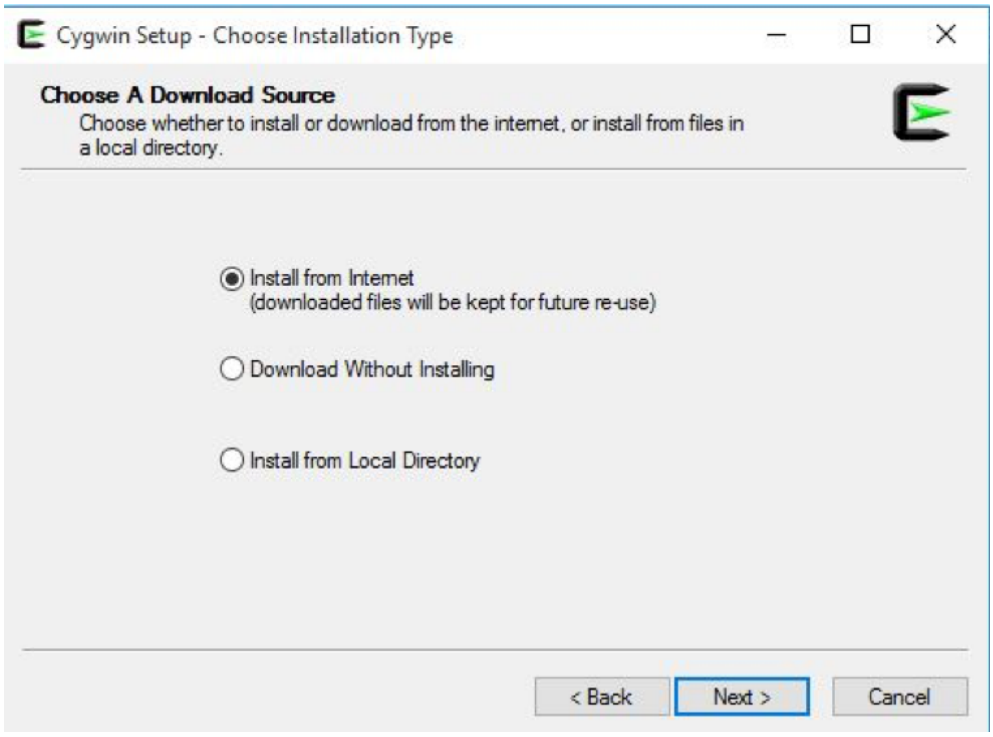


Figure 12: Cygwin Installation type window.

Choose "Install from Internet" installation type, then click "Next", as shown in Figure 12.

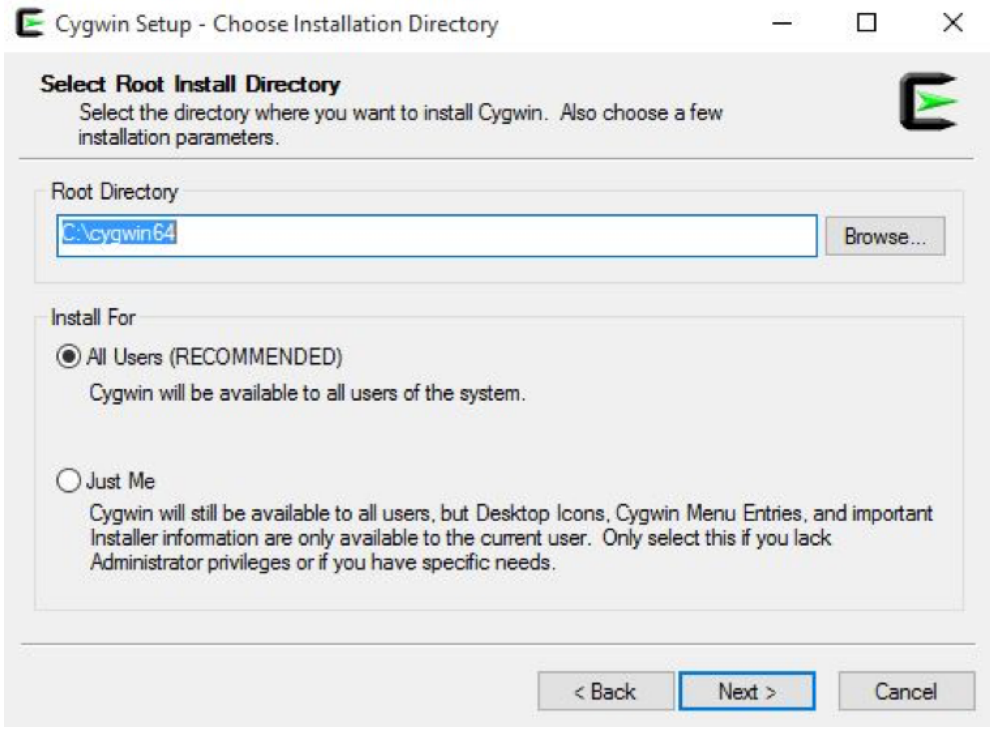


Figure 13: Cygwin Installation Directory window.

Set "C:\cygwin64" as the root directory. Select install for "All Users(RECOMMENDED)", then click "Next" to continue, as shown in Figure 13.

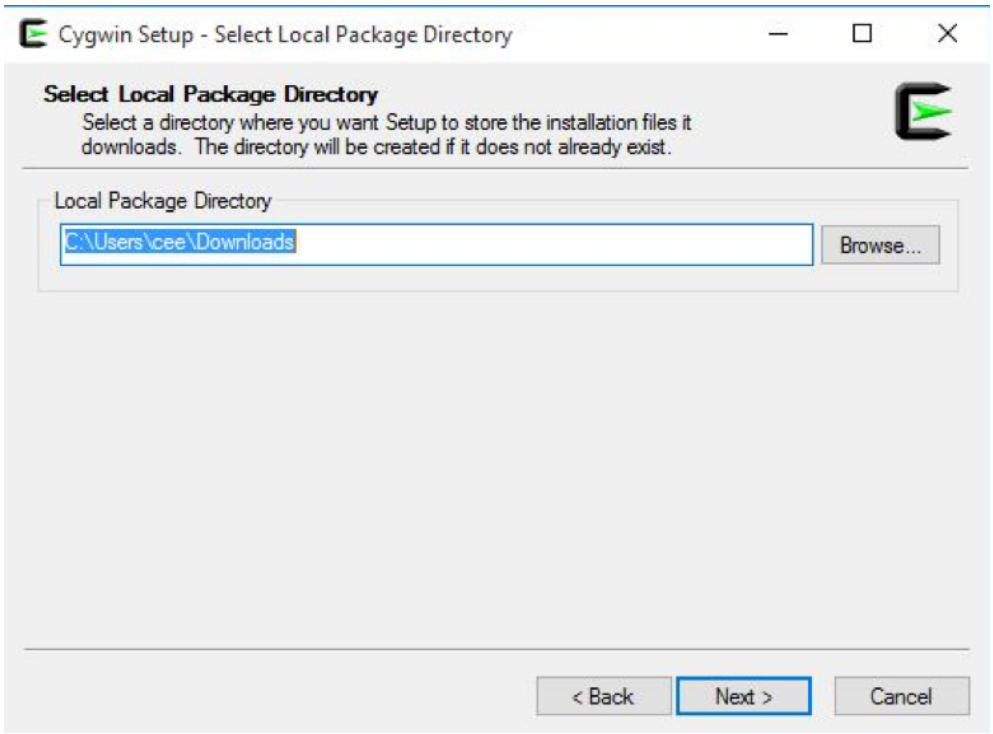


Figure 14: Cygwin local package directory window.

Save local package in the user's "Downloads" folder, then click "Next" to continue, as shown in Figure 14.

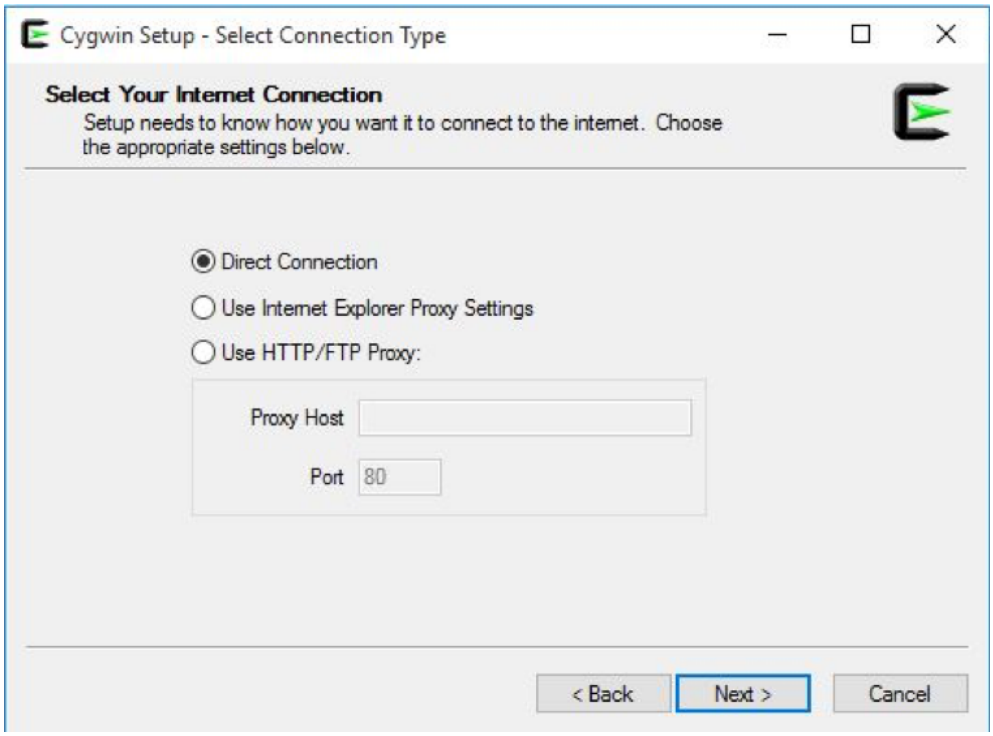


Figure 15: Cygwin connection type window.

Select "Direct Connection" to connect to the internet, then click "Next" to continue, as shown in Figure 15.

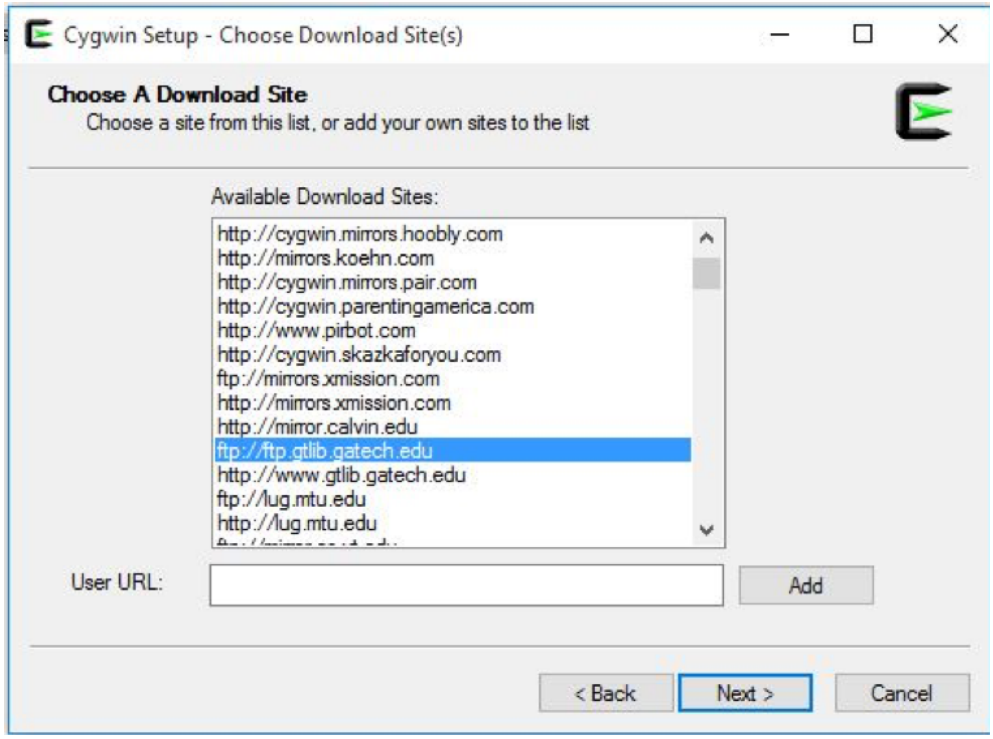


Figure 16: Cygwin download site window.

Choose any download site you like to download the necessary packages for Cygwin, then click "Next" to continue, as shown in Figure 16.

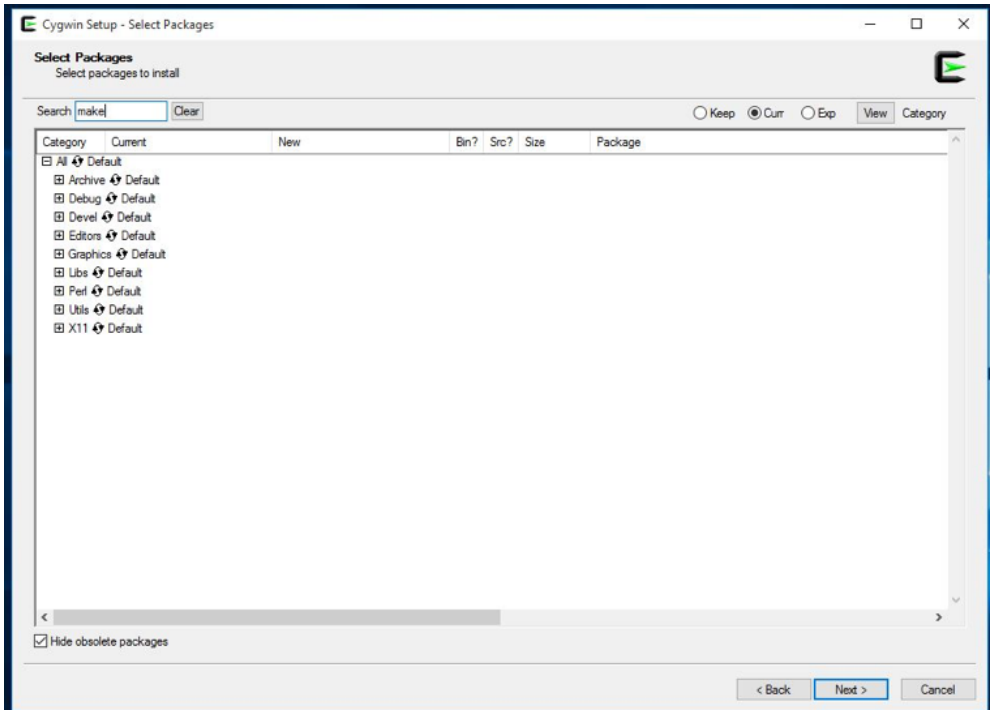


Figure 17: List of libraries in Cygwin.

Enter the name of the packages next to the search window to install packages. The packages **make**, **lapack**, **blas**, **patch**, **python**, **rebase**, **gcc**, **g++**, and **Cygwin**, should be installed for CU-BEN to run smoothly. Click "Default" next to each library to switch to "Install" as shown in the picture Figure 18.

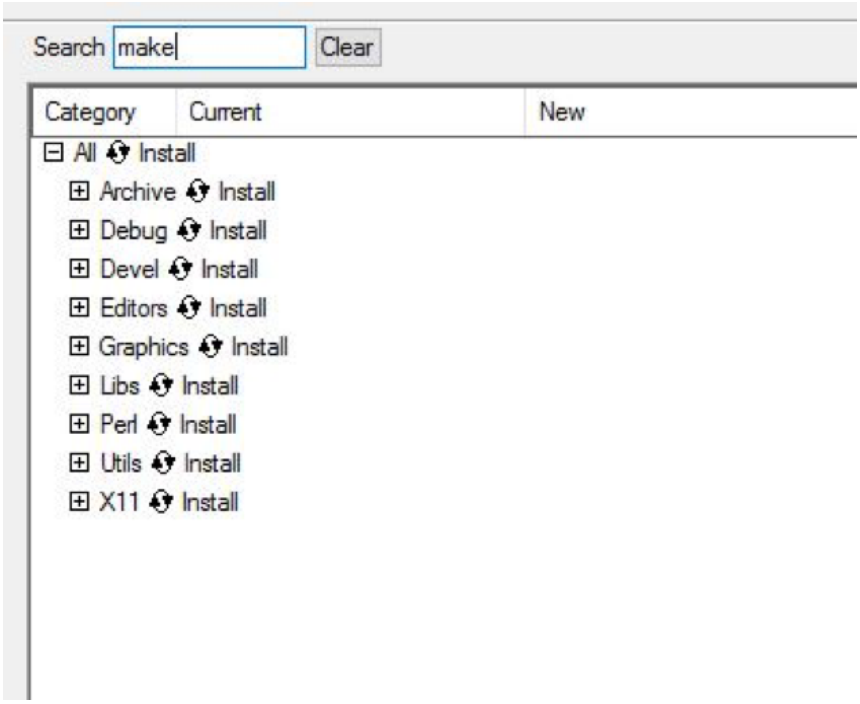


Figure 18: Install packages in Cygwin.

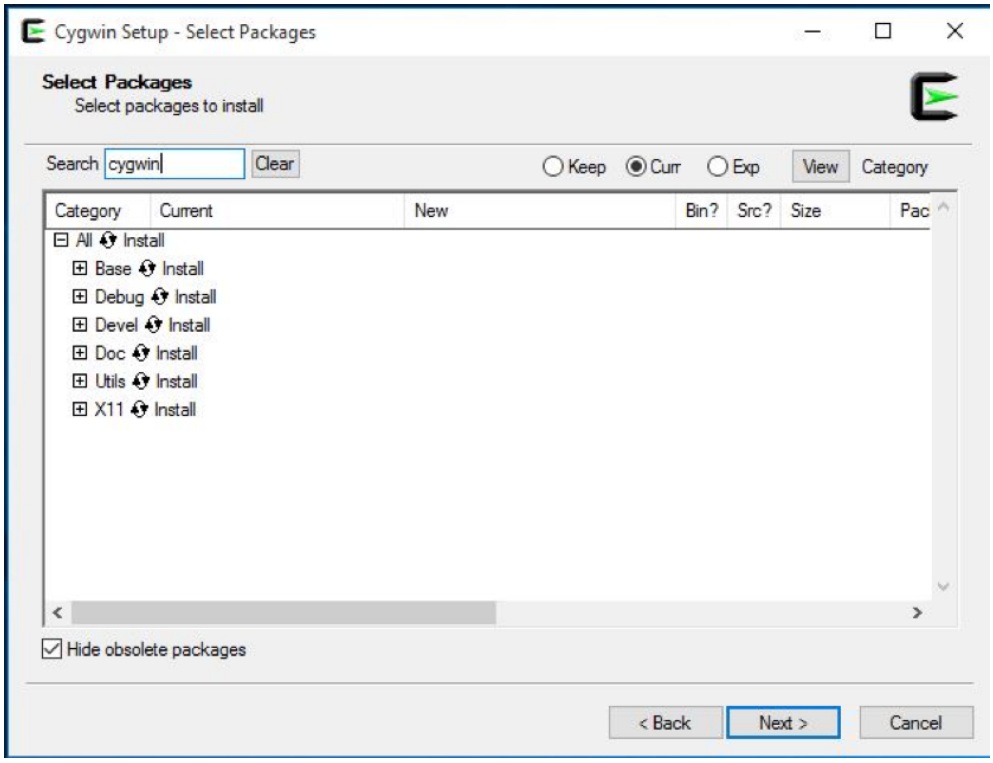


Figure 19: A list of libraries in Cygwin.

After all packages are selected to install, click "Next" to continue, as shown in Figure 19.

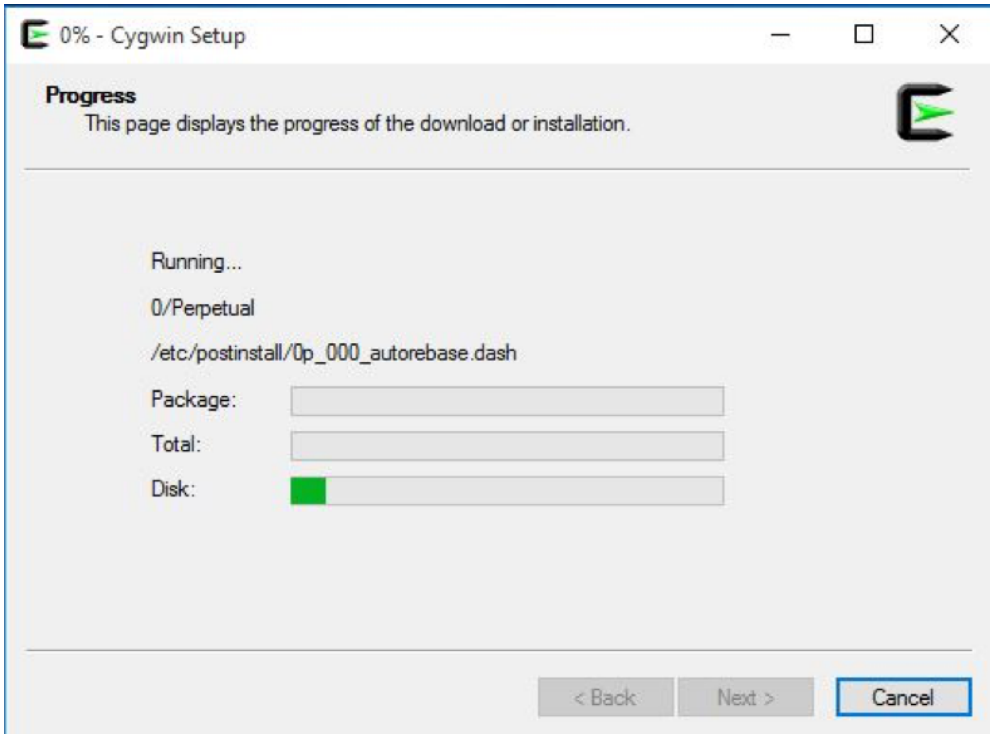


Figure 20: Cygwin setup window.

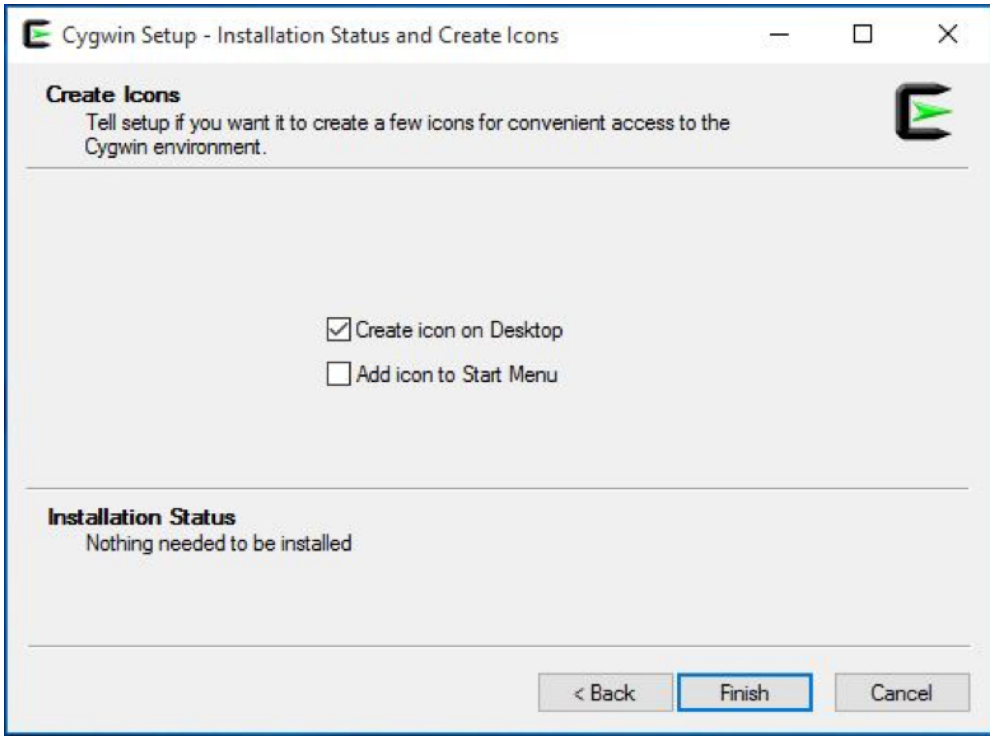


Figure 21: Cygwin installation status window.

Once Cygwin is successfully installed, check "Create icon on Desktop" and then click "finish", as shown in Figure 21. Double click Cygwin's icon on desktop to launch Cygwin terminal window.

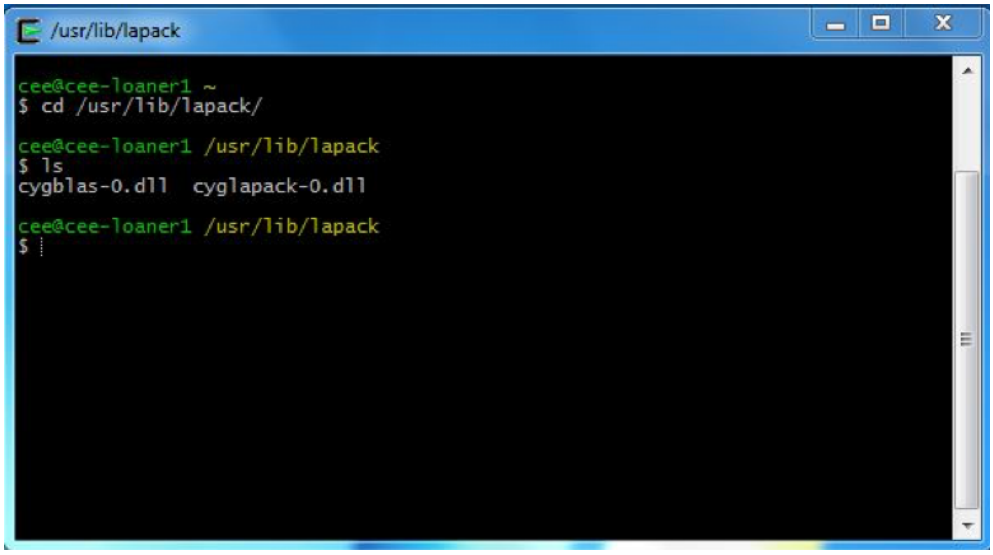


Figure 22: Location of BLAS and LAPACK libraries.

The dynamic libraries for BLAS and LAPACK, cygblas-0.dll and cyglapack-0.dll respectively, are located in "/usr/lib/lapack/".

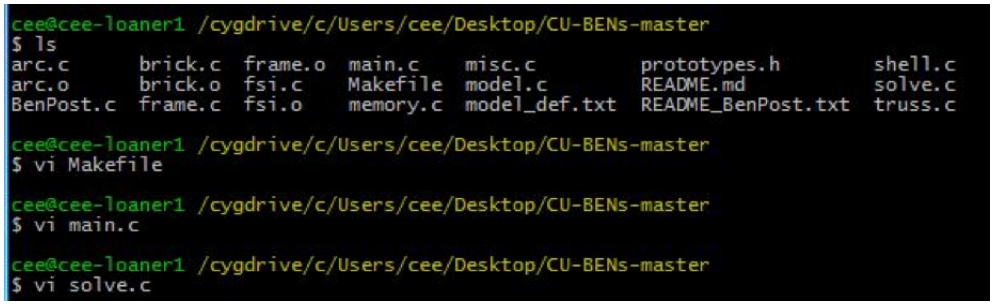


Figure 23: CU-BEN's master directory. (Note: Since this tutorial was prepared for the CU-BEN User Manual, developments to CU-BEN have prompted the inclusion of additional files.)

To access CU-BEN, open a second Cygwin terminal and change the present working directory to the path where CU-BEN is saved. When using Cygwin, it is important to specify the present working directory from within "/cygdrive/c/". If this is not specified, Cygwin will not be able to locate the necessary libraries for compiling.

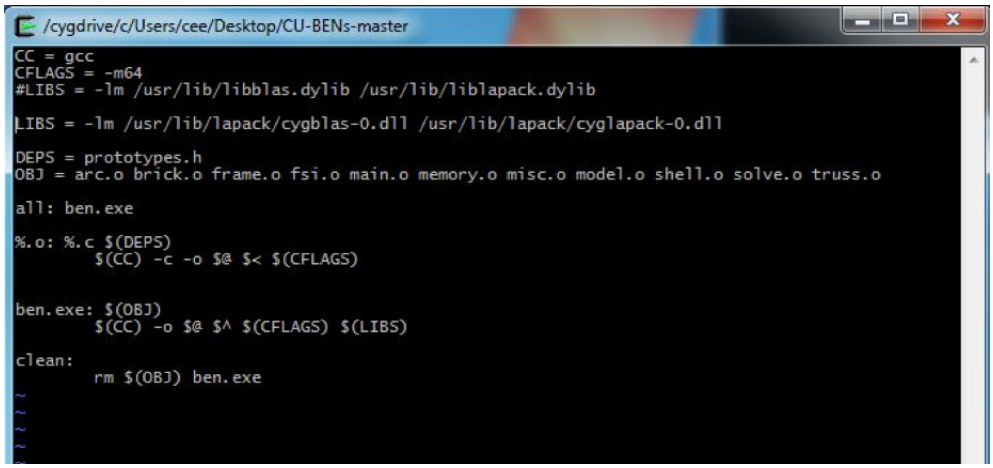


Figure 24: CU-BEN's makefile.

To modify CU-BEN's makefile for Windows, edit the makefile by commenting out or deleting the line "LIBS = -lm /usr/lib/libblas.dylib usr/lib/liblapack.dylib". To comment out the line, insert "#" in front of the line, as shown in Figure 24. Then, insert the following line: "LIBS = -lm /usr/lib/lapack/cygblas-0.dll /usr/lib/lapack/cyglapack-0.dll".

Figure 25: CU-BEN's main.c file.

Also, comment out "#include <Accelerate/Accelerate.h >" in main.c and solve.c. To do so, insert "//" in front of the line, as shown in Figure 25. This line is used only on Mac systems; it is an artifact from CU-BEN original creation using Xcode. The Accelerate Framework contains C APIs for vector and matrix math and large number handling, which optimize performance on Mac systems.

```
264                 // Calculate static force vector if generalized-alpha method applied
265                 if (alphaf != 0 ){
266                     if(ANAFLAG != 4 && SLVFLAG == 0){// Non-FSI analysis, using skyline function
267                         skymult (pmaxa, pss, pssd, pdd, fact, pdet);
268                     }
269                     else if (ANAFLAG == 4 || SLVFLAG == 1){// FSI analysis, cannot use skyline function
270                         double beta, gamma;
271                         int incx, incy;
272                         incx = incy = 1;
273                         beta = 1;
274                         gamma = 0;
275 
276                         for (i= 0; i<NEQ; ++i){
277                             *(pssd+i) = 0;
278                         }
279 
280                         char CblasRowMajor, CblasNoTrans;
281                         cblas_dgemv(CblasRowMajor, CblasNoTrans, m, n, beta, pss, lda, pdd, incx, gamma, pssd, incy);
282 
283                         for (i = 0; i < NEQ; ++i) {
284                             *(pdd+i) = *(pssd+i);
285                         }
286                     }
287                     for (i = 0; i < NEQ; ++i) {
288                         *(pReff+i) -= alphaf/(1-alphaf)*(*(pdd+i));
289                     }
290                 }
```

Figure 26: Declare variables in solve.c.

Declare variables CblasRowMajor and CblasNoTrans in solve.c as shown in Figure 26.
After all changes are made, enter the command "make" to compile CU-BEN, as shown in Figure 27.

```
/cygdrive/c/Users/cee/Desktop/CU-BENs-master

cee@cee-loaner1 /cygdrive/c/Users/cee/Desktop/CU-BENs-master
$ ls
arc.c      frame.c  main.o    misc.o         README.md           results4.txt  shell.o
arc.o      frame.o  Makefile  model.c        README_BenPost.txt  results5.txt  solve.c
BenPost.c  fsi.c    memory.c  model.o        results1.txt        results6.txt  solve.o
brick.c    fsi.o    memory.o  model_def.txt  results2.txt        results7.txt  truss.c
brick.o    main.c   misc.c    prototypes.h   results3.txt        shell.c       truss.o

cee@cee-loaner1 /cygdrive/c/Users/cee/Desktop/CU-BENs-master
$ vi Makefile

cee@cee-loaner1 /cygdrive/c/Users/cee/Desktop/CU-BENs-master
$ vi main.c

cee@cee-loaner1 /cygdrive/c/Users/cee/Desktop/CU-BENs-master
$ vi solve.c

cee@cee-loaner1 /cygdrive/c/Users/cee/Desktop/CU-BENs-master
$ make
gcc -o ben.exe arc.o brick.o frame.o fsi.o main.o memory.o misc.o model.o shell.o solve.o truss.o -m
64 -lm /usr/lib/lapack/cygblas-0.dll /usr/lib/lapack/cyglapack-0.dll

cee@cee-loaner1 /cygdrive/c/Users/cee/Desktop/CU-BENs-master
$
```

Figure 27: Compile CU-BEN.

For those compiling on a Unix operating system, it is necessary to download the BLAS and LAPACK libraries. Once those are downloaded, specify the file path in the make file under the LIBS flag, similarly to that shown in Figure 24. The user must also comment out the "#include <Accelerate/Accelerate.h >" line in main.c and solve.c. This can be done as shown in Figure 25. The user can make edits to the file using the vi text editor. As long as the GCC compiler is installed on the system used for compiling, and the discussed changes are made, the user should not have a problem with compiling.

5 Creating Input File

Detailed instructions on how to create an input file for any kind of analysis can be found within the comment header block of main.c, on lines 82-328. Input files must be named "model_def.txt" to be recognized by CU-BEN. The following are simple example input files for different analyses. These samples are available in a folder on GitHub titled "Sample_Input_Files." For use in CU-BEN, the user must rename the files "model_def.txt" and place them in the same directory as all other source code. Material properties consistent with steel are assumed in Examples 5.1-5.3, while material properties similar to those of polypropylene are specified for Example 5.4.

As units are not regulated by CU-BEN, the user must be careful and consistent with units throughout. For the examples provided, all coordinates and measures of length are provided in meters, Young's modulus in Pa, cross sectional area in m^2, density in kg/m^3, maximum allowable yield stress in Pa, shear modulus in Pa, moment of inertia in m^4, plastic section modulus in m^3, loads in N, and time in seconds. Time must always be specified in seconds, all other units can be adapted for the given analysis context, but must remain consistent. If using varied element types, the user must specify the element properties (connectivity and material parameters) in the following order: trusses, frames, shells, solid bricks, acoustic fluid bricks. The instructions on input file construction, as contained in the comment header to the main.c file, will elucidate further details.

5.1 Statically Loaded, Geometrically Nonlinear Truss Example

Figure 28 describes the geometry of the first example input file. It has 3 truss elements and 4 joints. The cross-sectional area of each of the elements is 0.0001 m^2. Three of the joints have pin constraints and the fourth has a horizontally applied load of 10,000 N. The joint with the load is constrained in the z- direction, to prevent out of plane translation. The analysis has 2 DOFs. This is a 2^{nd} order elastic analysis that uses the Modified Newton-Raphson (MNR) solution method.

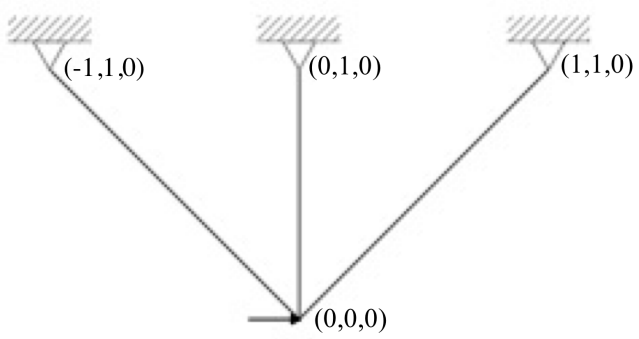


Figure 28: Geometry of example truss element. The coordinates shown are in meters.

2	Analysis Type: 2^{nd} order elastic
2	Solution Method: Modified Newton-Raphson
0	Solver Algorithm: Skyline for symmetric matrices *(See Note 5.1-1)*
1	Node Renumbering Algorithm: Off *(See Note 5.1-2)*
4	Number of Joints: 4
3,0,0,0,0	Elements: 3 truss, 0 frame, 0 shell, 0 solid brick, 0 acoustic fluid bricks
1,4	Truss Element 1 connectivity
2,4	Truss Element 2 connectivity
3,4	Truss Element 3 connectivity
1,1	Joint 1 Boundary Conditions: pinned
1,2	
1,3	
2,1	Joint 2 Boundary Conditions: pinned
2,2	
2,3	
3,1	Joint 3 Boundary Conditions: pinned
3,2	
3,3	
4,3	Joint 4 Boundary Conditions: fixed against out of plane translation

0,0	Signals to program that boundary condition specification is completed
-1,1,0	Joint 1 x, y, z coordinates
0,1,0	Joint 2 x, y, z coordinates
1,1,0	Joint 3 x, y, z coordinates
0,0,0	Joint 4 x, y, z coordinates
210000000000,0.0001,8050,345000000	Truss Element Properties: Elastic Modulus, Cross Sectional Area, Density, Maximum Allowable Yield Stress
210000000000,0.0001,8050,345000000	
210000000000,0.0001,8050,345000000	
4,1,-10000	Joint Load: Joint Number 4, X- Load Direction, Magnitude
0,0,0	Signals to program that load specification is complete
1.0,0.1,0.1,0.1,0.1	Load proportionality parameters: Maximum Lambda, Initial Lambda, Default Increment of Lambda, Maximum Increment of Lambda, Minimum Increment of Lambda *(See Note 5.1-3)*
10,30,10	Counter parameters: Maximum number of iterations within load increment, Maximum number of times to step back load due to unconverged solution, Maximum number of converged solutions before increasing increment of lambda *(See Note 5.1-3)*
0.001,0.001,0.001	Convergence Tolerances: displacement, out-of-balance forces, energy

Table 1: Input file for Static 2nd Order Elastic Truss with line-by-line annotation.

Note 5.1-1: The other option for a linear solver is CLAPACK for symmetric and non-symmetric matrices. CLAPACK is a linear algebra package used to solve systems of linear, algebraic equations by direct means. It is typically only used for acoustic FSI analyses within CU-BEN, as CLAPACK has the ability to handle non-symmetric matrices. It is more efficient to use skyline for all other types of FE analysis.

Note 5.1-2: Node renumbering algorithm refers to a scheme wherein nodes are renumbered to produce a smaller bandwidth in the global, system stiffness matrix. This will minimize the storage requirements during the FE solution. It is pointed out that node renumbering is also useful when using the skyline storage scheme.

Note 5.1-3: These input parameters refer to solution tuning parameters for the Newton - Raphson method. These are user defined and affect the accuracy and economy of the nonlinear solution.

5.2 Dynamic Materially and Geometrically Nonlinear Frame Example

Figure 29 describes the geometry of the second example input file. It is a cantilever with 10 frame elements and 11 joints. The frame has a square cross section and uses steel material properties. The cross-sectional area of the elements is 0.000441 m^2. The values for the moments, plastic sections, and other geometric properties were determined from the material properties and prescribed geometry. One joint has fixed constraints and the other joints are free. The joint at the far end of the cantilever has an axially applied load of 2796 N, which is 33% of the critical buckling load. The joint also has a transversely applied loading whose sinusoidal loading time history has a peak amplitude of 110 N. The frequency of this transverse loading is 6240 degs/sec. The total time of this

Figure 29: Geometry of example frame element. The coordinates shown are in meters.

analysis is 0.11538 sec, divided into 16 equal time steps. This is a 2nd order elastic analysis that uses the nonlinear Newmark implicit integration time integration scheme.

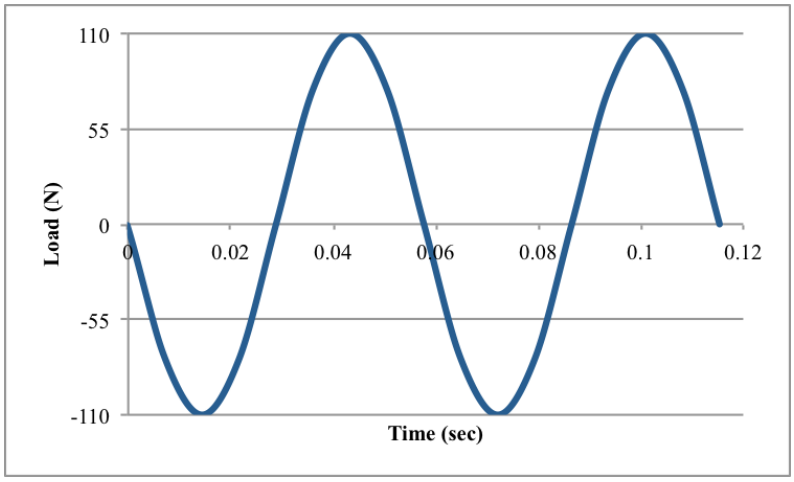


Figure 30: Plot of transverse load function over time. The amplitude is 110 N; the frequency is 17.3 Hz. The length of the analysis is 0.11538 sec with 16 time steps of 0.00721125 sec each.

3	Analysis Type: 2nd order inelastic
5	Solution Method: Nonlinear Newmark Implicit Integration Scheme
10,0	Data write checkpoint every 10 time steps, Restart from last saved data point: Off *(See Note 5.2-1)*
0	Solver Algorithm: Skyline for symmetric matrices *(See Note 5.2-2)*
1	Node Renumbering Algorithm: Off
11	Number of Joints: 11
0,10,0,0,0	Elements: 0 truss, 10 frame, 0 shell, 0 solid bricks, 0 acoustic fluid bricks
1,2	Frame Element 1 connectivity
2,3	Frame Element 2 connectivity
3,4	Frame Element 3 connectivity
4,5	Frame Element 4 connectivity
5,6	Frame Element 5 connectivity
6,7	Frame Element 6 connectivity
7,8	Frame Element 7 connectivity
8,9	Frame Element 8 connectivity
9,10	Frame Element 9 connectivity
10,11	Frame Element 10 connectivity
1,1	Joint 1 Boundary Conditions: fixed
1,2	
1,3	
1,4	
1,5	
1,6	
0,0	Signals to program that boundary condition specification is completed

0,0	Signals to program that prescribed displacement boundary condition specification is completed *(See Note 5.2-3)*
0,0,0	Joint 1 x, y, z coordinates
0.1,0,0	Joint 2 x, y, z coordinates
0.2,0,0	Joint 3 x, y, z coordinates
0.3,0,0	Joint 4 x, y, z coordinates
0.4,0,0	Joint 5 x, y, z coordinates
0.5,0,0	Joint 6 x, y, z coordinates
0.6,0,0	Joint 7 x, y, z coordinates
0.7,0,0	Joint 8 x, y, z coordinates
0.8,0,0	Joint 9 x, y, z coordinates
0.9,0,0	Joint 10 x, y, z coordinates
1,0,0	Joint 11 x, y, z coordinates
2100000000000,810000000000, ⋯	Frame Element Properties: Elastic Modulus, Shear Modulus, Density, Cross Sectional Area, Moment of Inertia – Strong Axis, Moment of Inertia – Weak Axis, Moment of Inertia – Polar, Moment of Inertia – Warping *(See Note 5.2-4)*
2100000000000,810000000000, ⋯	
2100000000000,810000000000, ⋯	
2100000000000,810000000000, ⋯	
2100000000000,810000000000, ⋯	
2100000000000,810000000000, ⋯	
2100000000000,810000000000, ⋯	
2100000000000,810000000000, ⋯	
2100000000000,810000000000, ⋯	
0,0,0,0,0,0,0	Signals to program that there are no frame member end offsets to apply *(See Note 5.2-5)*
0.5,1,0	Frame element auxiliary points *(See Fig 31)*
0.5,1,0	
0.5,1,0	
0.5,1,0	
0.5,1,0	
0.5,1,0	
0.5,1,0	
0.5,1,0	
0.5,1,0	
0.5,1,0	
345000000,0.0000015435,0.0000015435	Frame Element Yield Criteria: Yield Strength, Strong-Axis Plastic Section Modulus, Weak-Axis Plastic Section Modulus
345000000,0.0000015435,0.0000015435	
345000000,0.0000015435,0.0000015435	
345000000,0.0000015435,0.0000015435	
345000000,0.0000015435,0.0000015435	
345000000,0.0000015435,0.0000015435	
345000000,0.0000015435,0.0000015435	
345000000,0.0000015435,0.0000015435	
345000000,0.0000015435,0.0000015435	
345000000,0.0000015435,0.0000015435	
0,0,0,0,0	Signals to program that there are no frame member end releases to apply *(See Note 5.2-6)*
16,0.11538	Number of time steps, Total time for analysis (in sec)

11,1,-2796	Joint Load: Joint Number 11, X- Load Direction, Magnitude (Axial Load) *(See Note 5.2-7)*
11,1,-2796	
11,1,-2796	
11,1,-2796	
11,1,-2796	
11,1,-2796	
11,1,-2796	
11,1,-2796	
11,1,-2796	
11,1,-2796	
11,1,-2796	
11,1,-2796	
11,1,-2796	
11,1,-2796	
11,1,-2796	
11,1,-2796	
11,2,0	Joint Load: Joint Number 11, Y- Load Direction, Magnitude (Transverse Load) *(See Note 5.2-7)*
11,2,-77.78166	
11,2,-110	
11,2,-77.78199	
11,2,-0.000532187	
11,2,77.78133	
11,2,110	
11,2,77.78243	
11,2,0.001064373	
11,2,-77.78089	
11,2,-110	
11,2,-77.78276	
11,2,-0.001596562	
11,2,77.7832	
11,2,110	
11,2,77.7832	
11,2,0.002128742	
0,0,0	Signals to program that joint load specification is completed
0,0,0	Signals to program that prescribed displacement specification is completed
0,0,0,0,0	Signals to program that distributed load specification is completed
1,0.9	Damping scheme: Generalized-α method, Spectral radius of 0.9 *(See Note 5.2-8)*
1, 0.1, 0.1, 10, 0.1	Load proportionality parameters: Maximum Lambda, Initial Lambda, Initial Increment of Lambda, Maximum Increment of Lambda, Minimum Increment of Lambda *(See Note 5.1-3)*
1000, 1000, 1000	Counter parameters: Maximum number of iterations within load increment, Maximum number of times to step back load due to unconverged solution, Maximum number of converged before increasing increment of Lambda *(See Note 5.1-3)*

0.0001,0.0001,0.0001	Convergence Tolerances: displacement, out-of-balance forces, energy *(See Note 5.2-9)*

Table 2: Input file for Nonlinear Frame Using Modified Newton-Raphson with line-by-line annotation.

Note 5.2-1: The data write checkpoint specification and analysis restart capability are only applicable for use in transient dynamic analysis. The user must specify a checkpoint that ranges from 0 to the total number of time steps to signal to CU-BEN the frequency at which to back up necessary data structures. If a value of 0 is specified, the user is indicating not to generate any checkpoint data. For any value greater, the backup data will be output to the "result8.txt" file at each checkpoint. At each checkpoint, the data from the previous checkpoint will be overwritten. Additionally, the user must specify a value for restart flag: a restart flag of 0 indicates the analysis begins at the initial time step; a restart flag of 1 indicates the analysis resumes from the structural configuration saved in result8.txt, i.e. the last successfully analyzed time step written before the program crash. An analysis with a restart flag of 1 must have a backup file, i.e. "results8.txt," available to run successfully.

Note 5.2-2: The skyline algorithm organizes a symmetric matrix into an indexed array. This algorithm takes advantage of matrix sparsity and tracks a variable bandwidth to store primarily non-zero entries from the global system stiffness matrix. This is a highly efficient storage scheme. In one example, the use of the skyline storage scheme reduced the storage needs of a 46,000 DOF model from 55 GBs to 1 GB. All elements, except the solid and acoustic fluid bricks, employ a lumped mass matrix: the bricks all have consistent mass matrices.

Note 5.2-3: The dynamic displacement boundary condition algorithm allows the user to analyze the structural response of a dynamic system when nonzero displacement is applied at fixed joints on the structure. This capability is only applicable for transient dynamic analysis. Similar to the description of the fixed boundary conditions, the joint number and direction in which the prescribed displacement is applied need be specified before signaling to the program that specification is completed. It is also important to point out that all nodes that have dynamic boundary conditions applied must be excluded from the specification of fixed boundary conditions. The nodes with dynamic boundary conditions will not overwrite previously specified fixed boundary conditions and will lead to errors in analysis.

Note 5.2-4: When specifying material properties, each element must be described completely on a single line (i.e. do not separate material properties on numerous lines in the input file). In the sample input file included, the materials span several lines due to the limitations of the word processing file. In the annotated input file, it was chosen to truncate the lines for clarity.

Note 5.2-5: Frame member offsets refer to situations where both shell and frame elements are in use. This is meant to model stiffened panels in ship hulls. In FE analysis, thickness and cross-sectional area are specified as material properties, so if a frame is added to stiffen a shell panel, the frame mid-plane will be on the same axis as the shell mid-plane, which is not geometrically correct. The offset allows the elements to be in the same plane but adjusts the specification of the frame section to respect the separation. This allows the user to specify additional elements without having to specify additional nodes.

Note 5.2-6: When a frame member release is applied, the member connections are converted from the default moment connections to pin connections, where moment is not transferred. The user can specify strong or weak axis moment release at one or both ends.

Note 5.2-7: Even when a static load is applied, when running dynamic analysis, the point load on a particular joint has to be specified explicitly at each time step.

Note 5.2-8: For the versatile damping scheme implemented, the spectral radius specifies the level of numerical damping the user wishes to apply to the equations of motion. The spectral radius can range between 1 (for a completely undamped analysis) and 0 (completely damped), inclusive. This spectral radius will be used to calculate parameters used to scale the equation of motion based on the user specified damping scheme denoted by the flag on the same line. For more information, see [2].

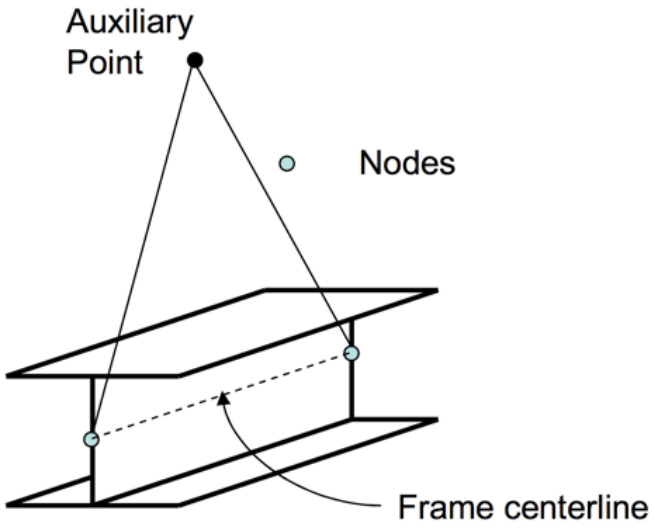


Figure 31: Frame element auxiliary points orient the strong axis of the frame element. In CU-BEN, a frame element is specified as a line with some properties associated with it. However, as the properties of a frame element depend upon the orientation of said element, with respect to some reference configuration, we must provide to CU-BEN information as to how the element is oriented. The so-called auxiliary point orients the strong-axis of the frame element.

Note 5b-9: These parameters control the convergence of the Newton-Raphson method. These will tune the accuracy of the solution.

5.3 Dynamic Linear Elastic Shell Example

The geometry for the third example of a linear elastic cantilever with a dynamic loading can be found in Figure 32. It has 6 joints and 4 elements. The thickness of the elements is 0.009 m. The near joints have fixed constraints and the other four joints are free. The joints at the far end of the cantilever each have a transversely applied sinusoidal load function. The load over time can be found in Figure 33. The analysis has 24 DOFs and uses the Newmark Implicit Time Integration method.

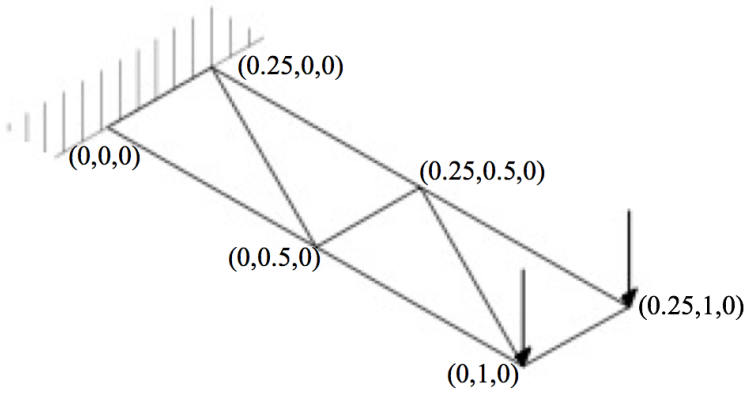


Figure 32: Geometry of example shell element. The coordinates shown are in meters.

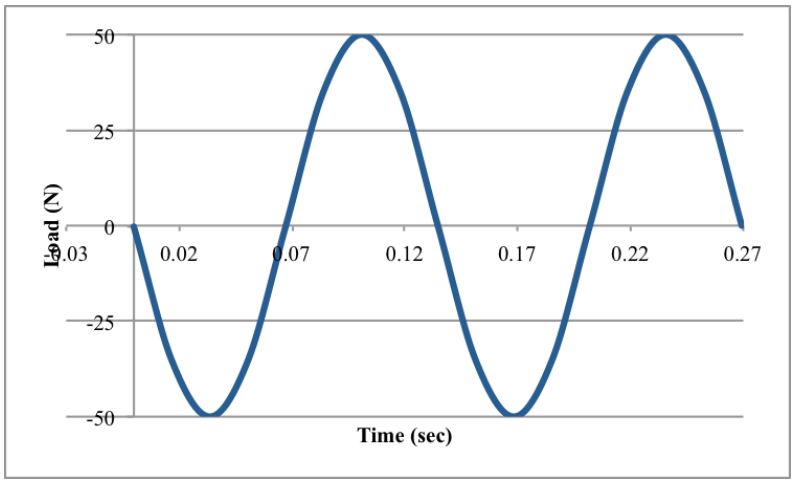


Figure 33: Plot of load function on each node at tip of cantilever over time. The amplitude is 50 N; the frequency is 7.4 Hz. The duration of the analysis is 0.26934 sec, with 16 equal time steps of 0.016833 sec each.

1	Analysis Type: 1st order elastic
4	Solution Method: Newmark Implicit Integration Method
0,0	Data write checkpoint every 4 time steps, Restart from last saved data point: Off *(See Note 5.2-1)*
0	Solver Algorithm: Skyline for symmetric matrices
1	Node Renumbering Algorithm: Off
6	Number of Joints: 6
0,0,4,0,0	Elements: 0 truss, 0 frame, 4 shell, 0 solid bricks, 0 acoustic fluid bricks
1,2,6	Shell Element 1 connectivity *(See Note 5.3-1)*
2,3,6	Shell Element 2 connectivity *(See Note 5.3-1)*
3,5,6	Shell Element 3 connectivity *(See Note 5.3-1)*
3,4,5	Shell Element 4 connectivity *(See Note 5.3-1)*
1,1	Joint 1 Boundary Conditions: fixed
1,2	
1,3	
1,4	
1,5	
1,6	
2,1	Joint 2 Boundary Conditions: fixed
2,2	
2,3	
2,4	
2,5	
2,6	
0,0	Signals to program that boundary condition specification is completed

0,0	Signals to program that prescribed displacement boundary condition specification is completed *(See Note 5.2-3)*
0,0,0	Joint 1 x, y, z coordinates
0.25,0,0	Joint 2 x, y, z coordinates
0.25,0.5,0	Joint 3 x, y, z coordinates
0.25,1,0	Joint 4 x, y, z coordinates
0,1,0	Joint 5 x, y, z coordinates
0,0.5,0	Joint 6 x, y, z coordinates
210000000000,0.3,0.009,8050,345000000	Shell Element Properties: Elastic Modulus, Poisson's Ratio, Thickness, Density, Maximum Allowable Yield Stress
210000000000,0.3,0.009,8050,345000000	
210000000000,0.3,0.009,8050,345000000	
210000000000,0.3,0.009,8050,345000000	
16,0.26934	Number of time steps, Total time for analysis (in sec)
5,3,0	Joint Load: Joint Number 5, Z-Load Direction, Magnitude *(See Note 5.2-7)*
5,3,-35.3546	
5,3,-50	
5,3,-35.3576	
5,3,-0.0042	
5,3,35.3517	
5,3,50	
5,3,35.3605	
5,3,0.0084	
5,3,-35.3487	
5,3,-50	
5,3,-35.3634	
5,3,-0.0125	
5,3,35.3457	
5,3,50	
5,3,35.36641	
5,3,0.0167	
0,0,0	Signals to program that joint load specification is completed
0,0,0	Signals to program that prescribed displacement specification is completed
0,0,0,0,0	Signals to program that specification of non-zero initial conditions for node displacement, velocity, and acceleration is completed
0,1	Damping scheme: No scheme chosen, Spectral radius of 1 *(See Note 5.3-2)*
0.001,0.001,0.001	Convergence Tolerances: displacement, out-of-balance forces, energy *(See Note 5.3-3)*

Table 3: Input file for Dynamic Linear Elastic Shell with line-by-line annotation.

Note 5.3-1: When defining the element connectivity for the shell element, it is important to define the elements to have normal vectors, according to the right-hand rule, oriented in the same direction. If the elements have normal vectors defined in varying directions, this can cause distortion and negatively influence the accuracy of the analysis.

Note 5.3-2: When the damping scheme flag is set to 0, this coincides with a Newmark time integration scheme with no damping. This analysis also is specified when the spectral radius is set to 1, which sets the Newmark integration parameters to $\alpha = 0.25$ and $\delta = 0.5$. When the Newmark integration parameters are set to $\alpha = 0.25$ and $\delta = 0.5$, the unconditionally stable constant average acceleration method (or trapezoidal rule) is attained. By choosing these values for the parameters, the value chosen for the time step size does not rely on the CFL con-

dition. It is important, however, to specify a time step sufficiently small to accurately describe the loading pattern.

Note 5.3-3: These parameters are not needed for linear elastic analysis, but because CU-BEN was created as a nonlinear code, the input file still requires the specification of the load proportionality factor and convergence tolerances.

5.4 Dynamic Geometrically Nonlinear Shell Example

The final example is a 2nd order elastic cantilever meshed with shell elements with prescribed dynamic boundary conditions. The geometry of the cantilever can be found in Figure 34. It has 6 joints and 4 elements. The thickness of the elements is 0.02 m. The near joints have fixed constraints and the other joints are constrained against x-rotation, y-rotation, and warping. The joints at the fixed end of the cantilever each have a transversely applied linear prescribed displacement function, described in Figure 35. Unlike the previous example, this model does not have any loading described. The analysis has 18 DOFs and uses the Nonlinear Newmark Implicit Time Integration method.

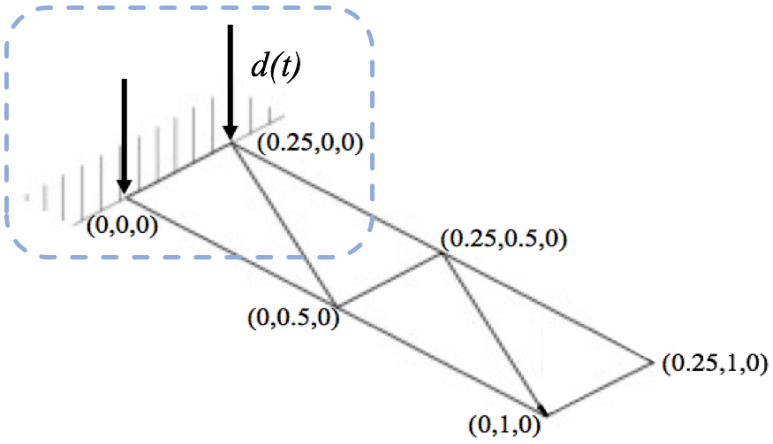


Figure 34: Geometry of example shell element. The coordinates shown are in meters. In this example, the arrows refer not to the application of load on a structure, but rather to the application of nonzero displacement boundary conditions over the time history of the structure, as denoted by *d(t)*.

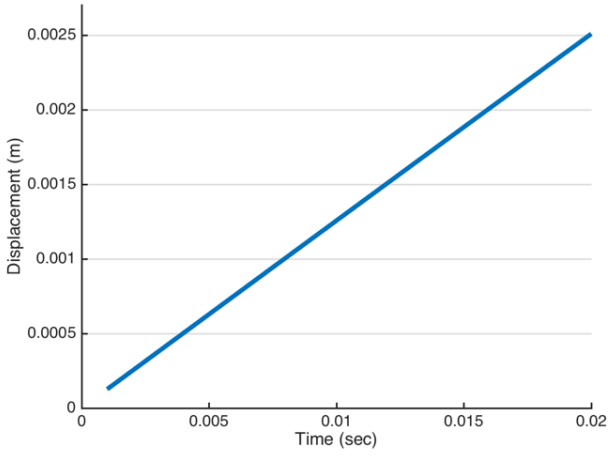


Figure 35: Plot of prescribed displacement function on each node at fixed end of cantilever over time. The duration of the analysis is 0.02 sec with 19 time steps of 0.0010526 sec each.

2	Analysis Type: 2nd order elastic
5	Solution Method: Newmark Implicit Integration Method
4,1	Data write checkpoint every 4 time steps, Restart from last saved data point: On *(See Note 5.2-1)*
0	Solver Algorithm: Skyline for symmetric matrices
1	Node Renumbering Algorithm: Off
6	Number of Joints: 6
0,0,4,0,0	Elements: 0 truss, 0 frame, 4 shell, 0 solid bricks, 0 fluid bricks
1,2,6	Shell Element 1 connectivity *(See Note 5.3-1)*
2,3,6	Shell Element 2 connectivity *(See Note 5.3-1)*
3,5,6	Shell Element 3 connectivity *(See Note 5.3-1)*
3,4,5	Shell Element 4 connectivity *(See Note 5.3-1)*
1,1	Joint 1 Boundary Conditions: fixed
1,2	
1,4	
1,5	
1,6	
2,1	Joint 2 Boundary Conditions: fixed
2,2	
2,4	
2,5	
2,6	
3,4	Joint 3 Boundary Conditions: constrained in x-rotation, y-rotation, and warping
3,6	
3,7	
4,4	Joint 4 Boundary Conditions: constrained in x-rotation, y-rotation, and warping
4,6	
4,7	
5,4	Joint 5 Boundary Conditions: constrained in x-rotation, y-rotation, and warping
5,6	
5,7	
6,4	Joint 6 Boundary Conditions: constrained in x-rotation, y-rotation, and warping
6,6	
6,7	
0,0	Signals to program that boundary condition specification is complete
1,3	Joint 1 Prescribed Displacement Boundary Condition *(See Note 5.2-3)*
2,3	Joint 2 Prescribed Displacement Boundary Condition *(See Note 5.2-3)*
0,0	Signals to program that prescribed displacement boundary condition specification is complete *(See Note 5.2-3)*
0,0,0	Joint 1 x, y, z coordinates
0.25,0,0	Joint 2 x, y, z coordinates
0.25,0.5,0	Joint 3 x, y, z coordinates
0.25,1,0	Joint 4 x, y, z coordinates
0,1,0	Joint 5 x, y, z coordinates

0,0.5,0	Joint 6 x, y, z coordinates
5600000.0,0.40,0.02,1000.0,2200000.0	Shell Element Properties: Elastic Modulus, Poisson's Ratio, Thickness, Density, Maximum Allowable Yield Stress
5600000.0,0.40,0.02,1000.0,2200000.0	
5600000.0,0.40,0.02,1000.0,2200000.0	
5600000.0,0.40,0.02,1000.0,2200000.0	
19,0.02	Number of time steps, Total time for analysis (in sec)
0,0,0.0	Signals to program that joint load specification is completed
1,3,1.256629e-04	Prescribed Displacement: Joint Number 1, Z Displacement Direction, Magnitude *(See Note 5.4-1)*
1,3,2.513208e-04	
1,3,3.769688e-04	
1,3,5.026019e-04	
1,3,6.282152e-04	
1,3,7.538037e-04	
1,3,8.793624e-04	
1,3,1.004886e-03	
1,3,1.130371e-03	
1,3,1.255810e-03	
1,3,1.381201e-03	
1,3,1.506536e-03	
1,3,1.631812e-03	
1,3,1.757024e-03	
1,3,1.882166e-03	
1,3,2.007234e-03	
1,3,2.132223e-03	
1,3,2.257128e-03	
1,3,2.381943e-03	
1,3,2.506665e-03	
2,3,1.256629e-04	Prescribed Displacement: Joint Number 2, Z Displacement Direction, Magnitude *(See Note 5.4-1)*
2,3,2.513208e-04	
2,3,3.769688e-04	
2,3,5.026019e-04	
2,3,6.282152e-04	
2,3,7.538037e-04	
2,3,8.793624e-04	
2,3,1.004886e-03	
2,3,1.130371e-03	
2,3,1.255810e-03	
2,3,1.381201e-03	
2,3,1.506536e-03	
2,3,1.631812e-03	
2,3,1.757024e-03	
2,3,1.882166e-03	
2,3,2.007234e-03	
2,3,2.132223e-03	
2,3,2.257128e-03	
2,3,2.381943e-03	
2,3,2.506665e-03	
0,0,0	Signals to program that prescribed displacement specification is completed *(See Note 5.4-1)*
0,0,0,0,0	Signals to program that specification of non-zero initial conditions for node displacement, velocity, and acceleration is completed

0,1.0	Damping scheme: No scheme chosen, Spectral radius of 1 *(See Note 5.3-2)*
1.0,0.1,0.1,10,0.1	Load proportionality parameters: Maximum Lambda, Initial Lambda, Increment of Lambda, Maximum Increment of Lambda, Minimum Increment of Lambda *(See Note 5.1-3)*
1000, 1000, 1000	Counter parameters: Maximum number of iterations within load increment, Maximum number of times to step back load due to unconverged solution, maximum number of converged solutions before increasing increment of lambda *(See Note 5.1-3)*
0.001,0.001,0.001	Convergence Tolerances: displacement, out of balance forces, energy *(See Note 5.2-9)*

Table 4: Input file for Dynamic 2nd Order Elastic Shell with line-by-line annotation.

Note 5.4-1: When running dynamic analysis with prescribed nonzero displacement boundary conditions, the displacement of a particular joint has to be specified explicitly at each time step.

6 Running Analysis

To run the analysis, first compile the code following the directions from Section 4. The model_def.txt input file created following the directions in Section 5 must be in the same folder as the executable ben.exe. Note that if the input file (model_def.txt) is not in the same folder as ben.exe, an error will **not** be returned, and the code will continue to run until the user force exits from it. If the input file is incomplete, the code will terminate with the error: "Segmentation fault: 11". To run the analysis, enter the command "./ben.exe" in the command line. The user must run the command while in the directory which contains the executable and input file. If running a nonlinear analysis, the code will output "LPF = ..." in the terminal window after every load increment iteration, as they are completed, and terminate. An example of this is shown in Figure 36. The value in the parentheses of the output line is the total number of sub-iterations to complete that load increment. If running a dynamic analysis, the terminal window will output "Time = ..." for each time step, as they are completed. An example of this is shown in Figure 37. If running a static linear elastic analysis, the terminal window will simply output "Analysis complete" during a successful solution and terminate. Every time CU-BEN is run, it will print 7 output files in the same folder as the input file and executable, named "results1.txt", "results2.txt", "results3.txt", "results4.txt", "results5.txt", "results6.txt", and "results7.txt". If running dynamic analysis and restart capability is turned on, CU-BEN will also print a "results8.txt" file. For all analysis methods, a summary of the analysis will be written into "results1.txt". If there is an error in the input file or in the way the model was specified, an error will print at the bottom of "results1.txt". The error message will guide the user to the source of the problem. Depending

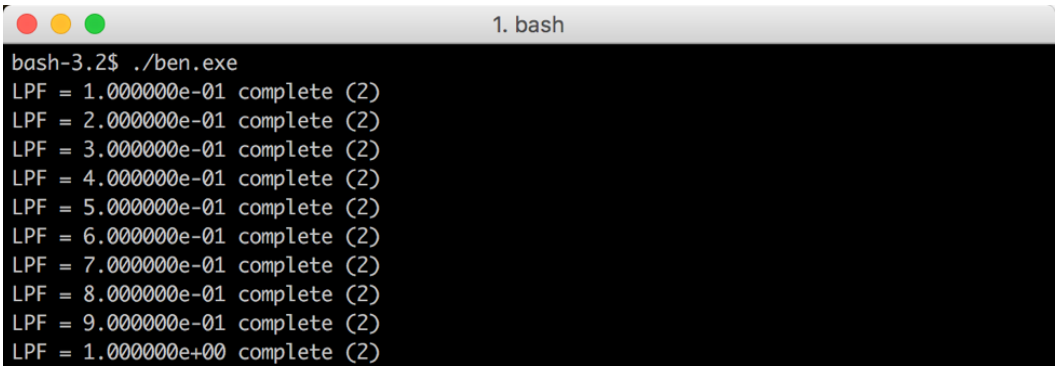


Figure 36: CU-BEN window output.

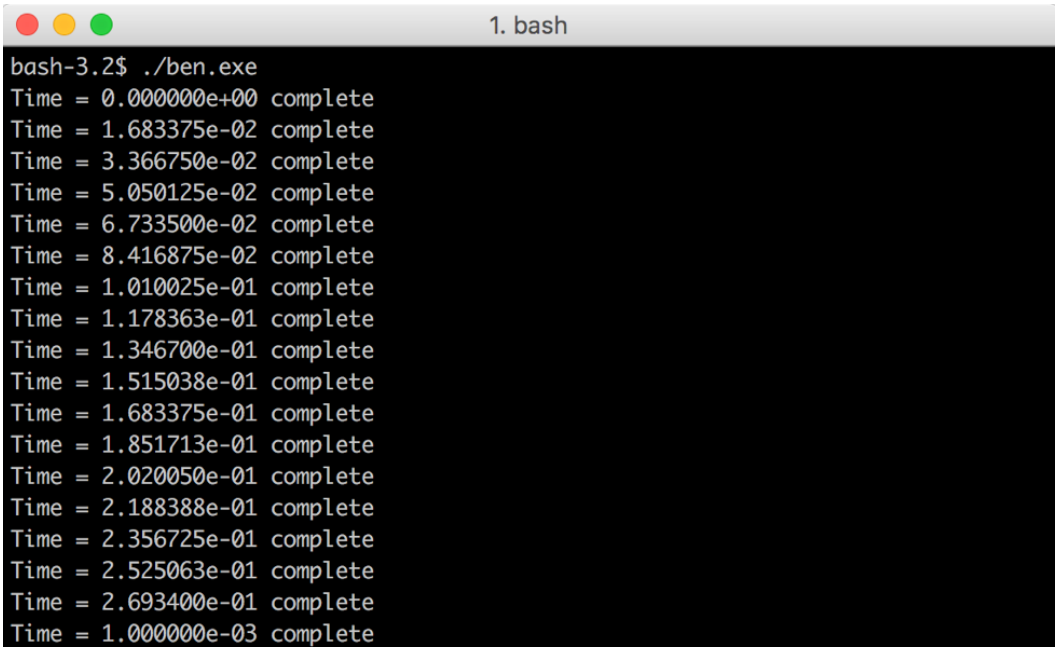


Figure 37: CU-BEN window output.

on the analysis of choice, different results will output in to different output files. For all solution methods, the deflection of the model will be output in "results2.txt". For DS, the displacements at each DOF are output. For nonlinear methods, the load proportionality factor (LPF), number of sub-iterations needed for convergence within that load increment iteration, and the displacement at each DOF are given in "results2.txt". For the Newmark method, the displacements at each DOF at each time step are furnished in "results2.txt". If using a truss element, "results3.txt" will contain truss element internal forces. If using a frame element, "results4.txt" will contain frame element internal forces and moments. If using a shell element, "results5.txt" will contain shell element internal forces and moments. The files "results6.txt" and "results7.txt" are used in the case of acoustic FSI. If running dynamic analysis and restart capability is turned on, "results8.txt" will contain all necessary data structures to restart analysis from the last successfully checkpoint time step in the time history. The examples described in Section 5 are included in the "Sample_Input_Files" folder on GitHub. These files must be renamed to "model_def.txt" and copied into the folder containing the CU-BEN executable to run correctly. To understand the expected run time for an input file, a model that has 7,417 shell elements, 3,938 joints, 23,484 DOFs, and 30 time steps takes about 4 minutes total to run with each time step taking approximately 0.5 seconds of wall clock time when running linear elastic analysis.

7 Using Abaqus® to Build Input File

While model_def.txt input files for simple analyses can be easily constructed using the examples in Section 5 and the detailed instruction in lines 81-272 of main.c, it can be difficult, in practice, to manually write an input file for more realistically complex structures or loading patterns. To remedy this, a protocol has been established to employ Abaqus to interactively define the geometry, boundary conditions, and load pattern for dynamic shell analysis (as an example). The user can run the rudimentary pre-processor, BenPre, to convert the data created in the Abaqus *.inp file to a model_def.txt input file for use in CU-BEN. BenPre.py is a rudimentary pre-processing routine for CU-BEN. This routine was written for Python 2.7. It operated on files that conform to the Abaqus *.inp formatting, along with supplemental command line prompts, to create an input file (i.e. model_def.txt) for CU-BEN. This routine is **only** intended to build models containing three node triangular **shell** elements (to be used in FE context). This code will not prepare a CU-BEN input file for other elements, or for a CU-BEN acoustic FSI analyses. While this code is helpful in constructing CU-BEN input files, it is **highly** recommended that the user reviews the input files generated: in order to ensure that she/he obtains the desired model specification. BenPre will offer some errors if the Abaqus file does not conform to the protocol outlined below or if inappropriate

analysis flags are input to the terminal; however, this will not necessarily guarantee a correct input file.

The following steps describe how Abaqus may be used to create a CU-BEN input file for the linear elastic cantilever described in Section 5.3.

To model the geometry of the structure, create a "Part" in Abaqus. Make sure that "Shell" is selected under "Base Feature >Shape." Because CU-BEN was designed for shell elements, using Shell as opposed to Solid will allow us to use the data from the *.inp file produced exactly. This is shown in Figure 38.

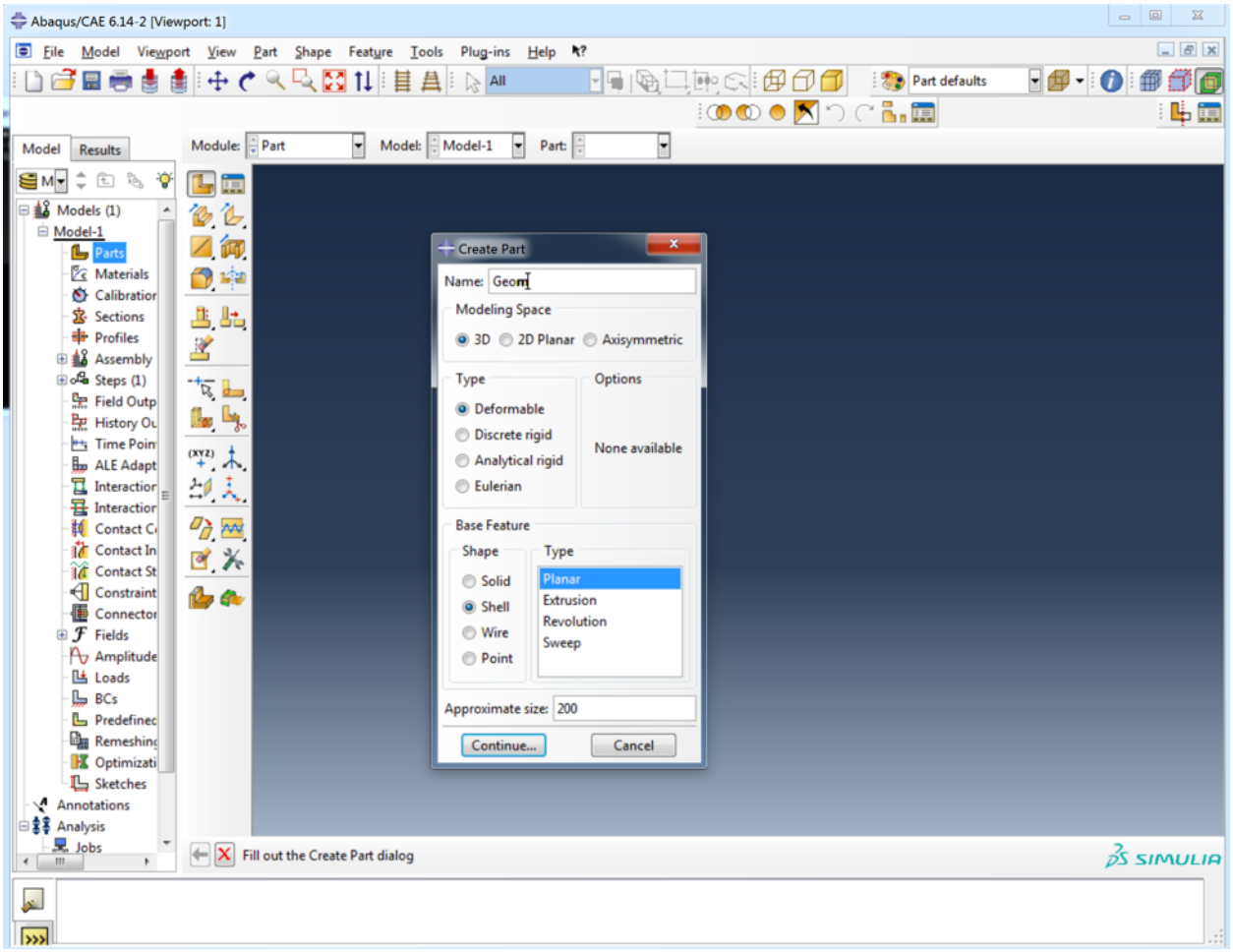


Figure 38: Controls for creating CU-BEN geometry using Abaqus' Parts.

To mesh the structure, first select "Tri" under "Mesh >Controls >Element Shape." A triangle must be specified in order to correctly describe the element for the DKT shell. This can be seen in Figure 39. A fully meshed cantilever is seen in Figure 40. If an incorrect mesh element shape is chosen in Abaqus, BenPre will prompt the user to re-mesh the model and to try converting the input file again.

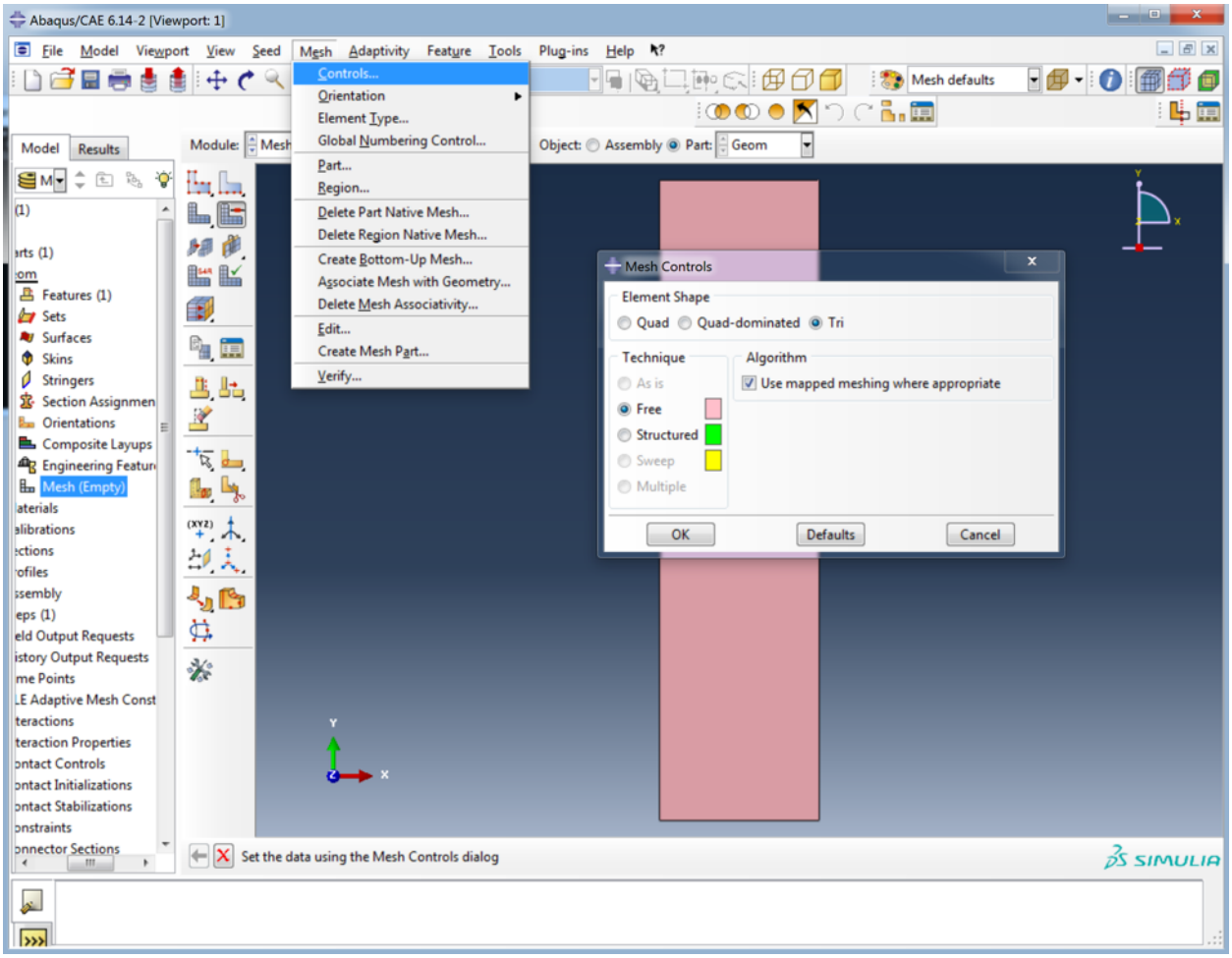


Figure 39: Mesh controls to define elements on geometry created in Abaqus.

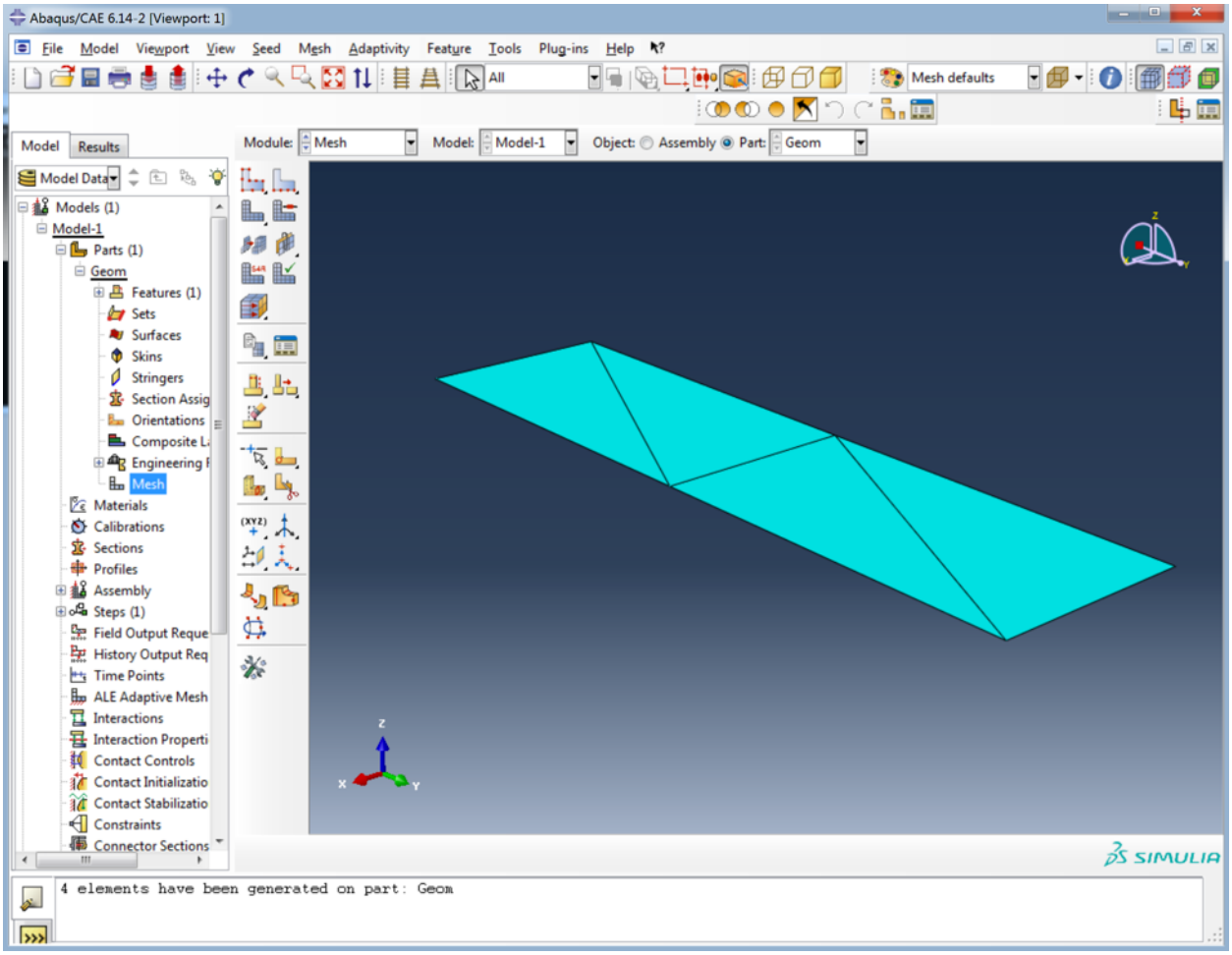


Figure 40: Meshed geometry created in Abaqus, analogous to the element specification in Figure 31.

To apply material properties, create a new "Material" in Abaqus. Under "General >Density," enter the "Mass Density." Under "Mechanical >Elasticity >Elastic," specify "Young's Modulus" and "Poisson's Ratio." Under "Mechanical >Plasticity >Plastic," specify "Yield Stress." This can be seen in Figure 41.

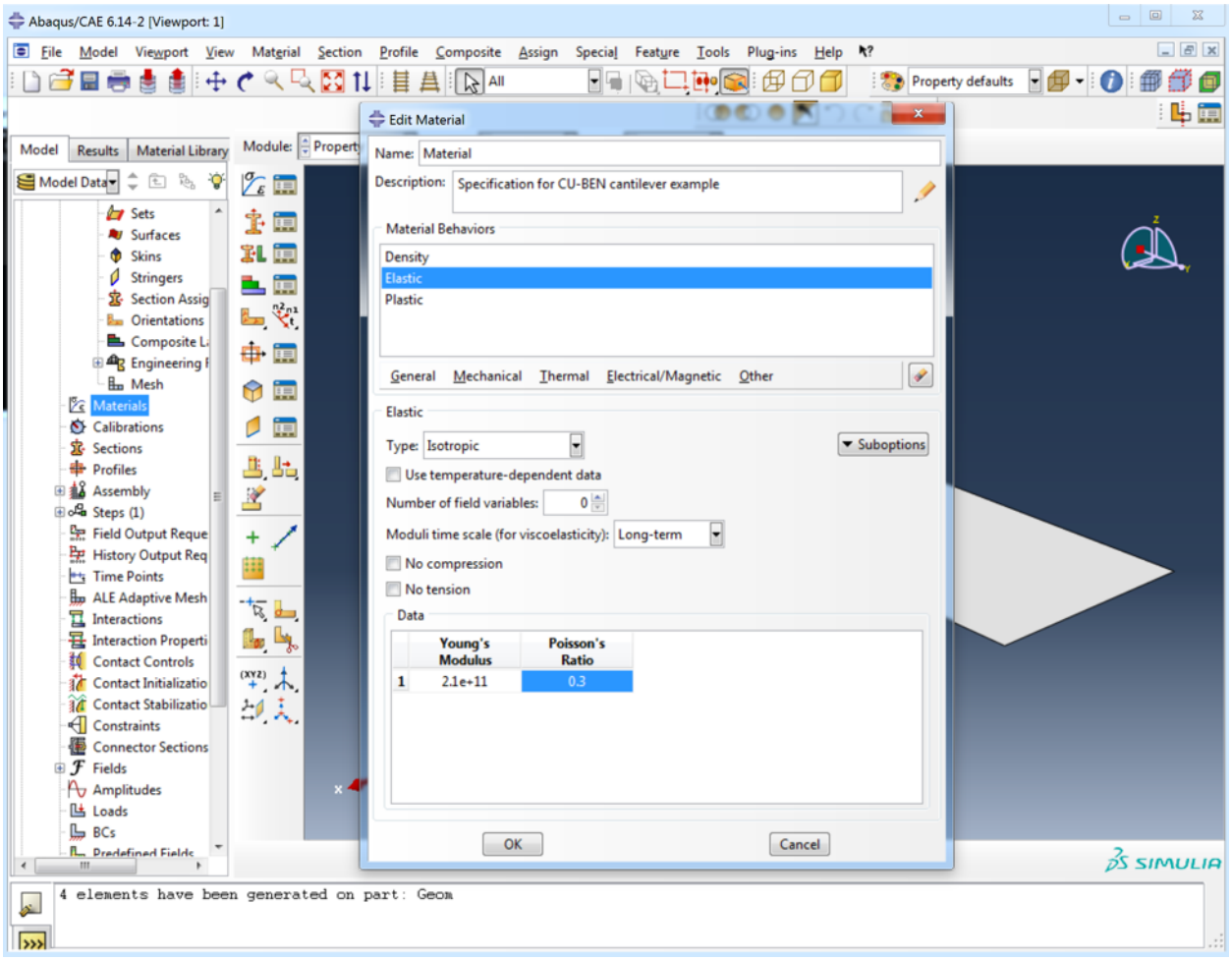


Figure 41: Material created in Abaqus, with appropriate material properties for CU-BEN analysis.

To specify the thickness in Abaqus, create a "Section." When creating the Section, be sure to choose "Shell" under "Category" and choose "Homogenous" under "Type." In the dialogue box under the "Basic" tab, the user can enter "Shell Thickness" by selecting "Value." If your model has various materials, you fill be prompted to specify the material for this particular section here. This can be seen in Figure 42.

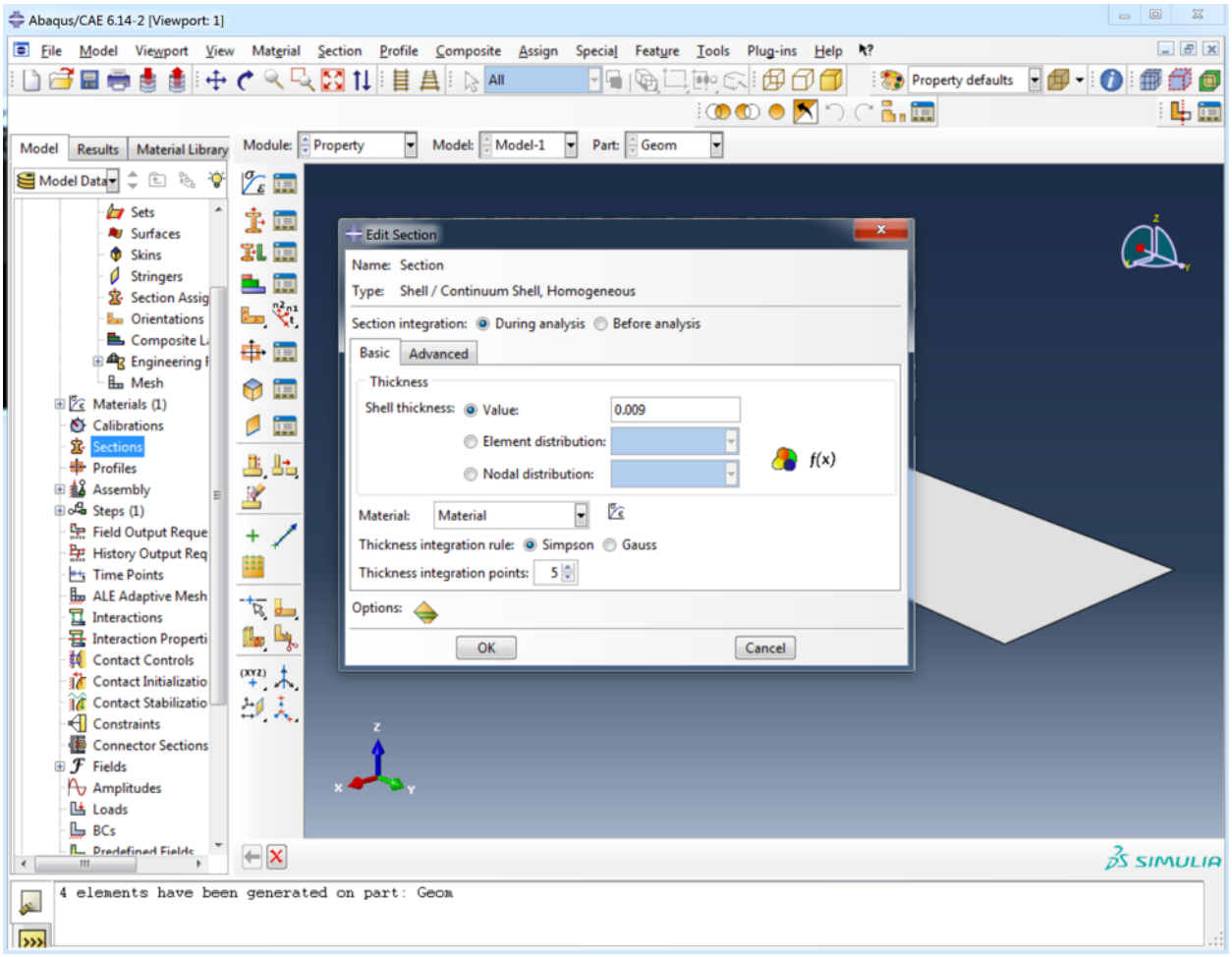


Figure 42: Section created in Abaqus, with appropriate geometric properties for CU-BEN analysis.

To apply the Material and Section created, under the Part created earlier, select "Section Assignment." Select the Part and specify the Section. If different areas have different thicknesses, apply each thickness to each separate area individually. Create a set from the region selected. In the dialogue box, specify the Section. Under "Thickness >Assignment," choose "From section." Under "Shell Offset >Definition," choose "Middle surface." This can be seen in Figure 43.

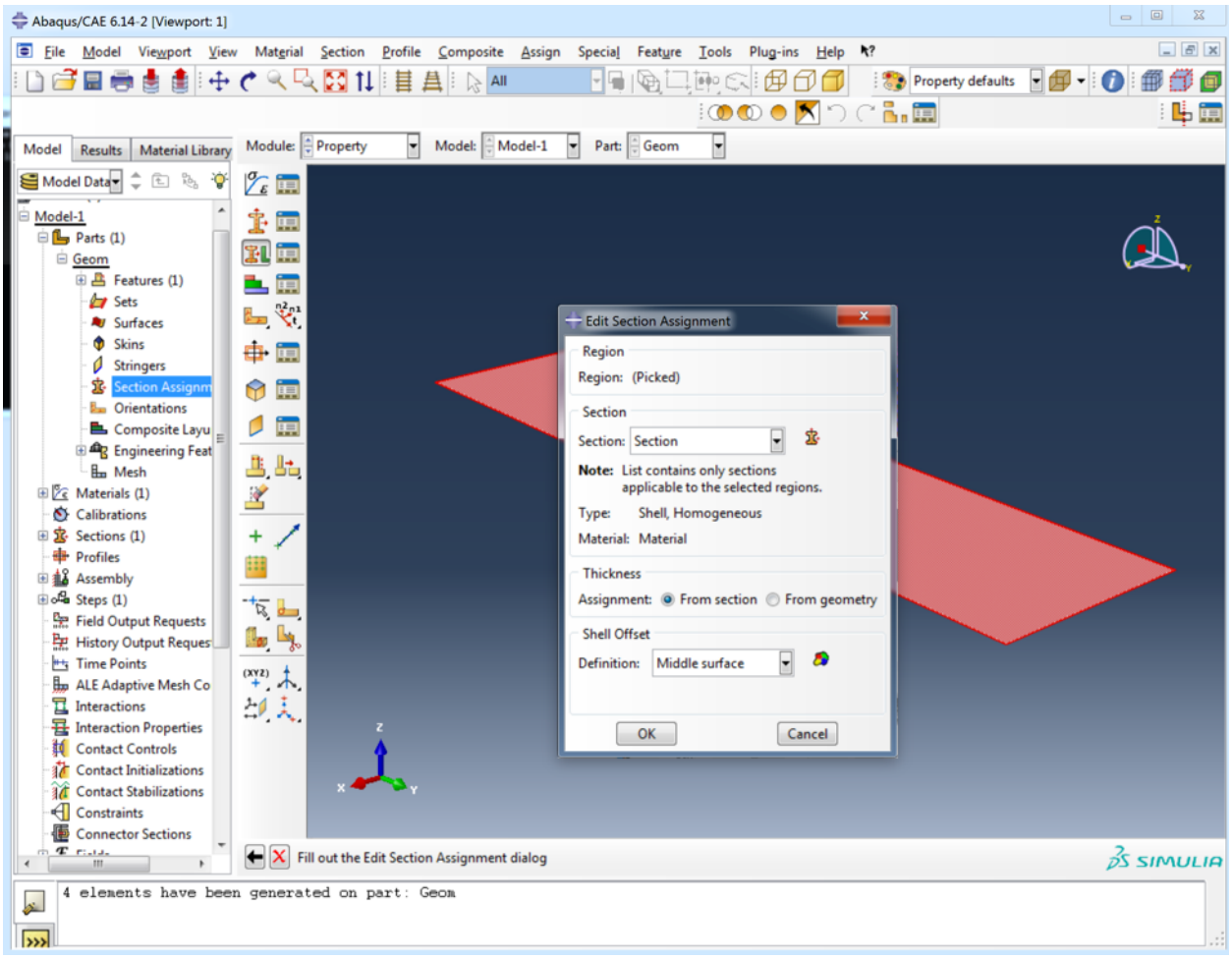


Figure 43: Section applied in Abaqus.

To create the time steps in the analysis, create a new "Step" named "TimeSteps." In the dialogue box, choose "Dynamic, Explicit." Under the "Basic" tab, specify the "Time Period" as the total time in the analysis. Under the "Incrementation" tab, select the "Type" as "Fixed." Under "Increment size selection," choose "User-defined time increment" and specify the size of the time step. This can be seen in Figure 44. If this step is omitted, BenPre will error out and prompt the user to try applying the load in the Abaqus file again.

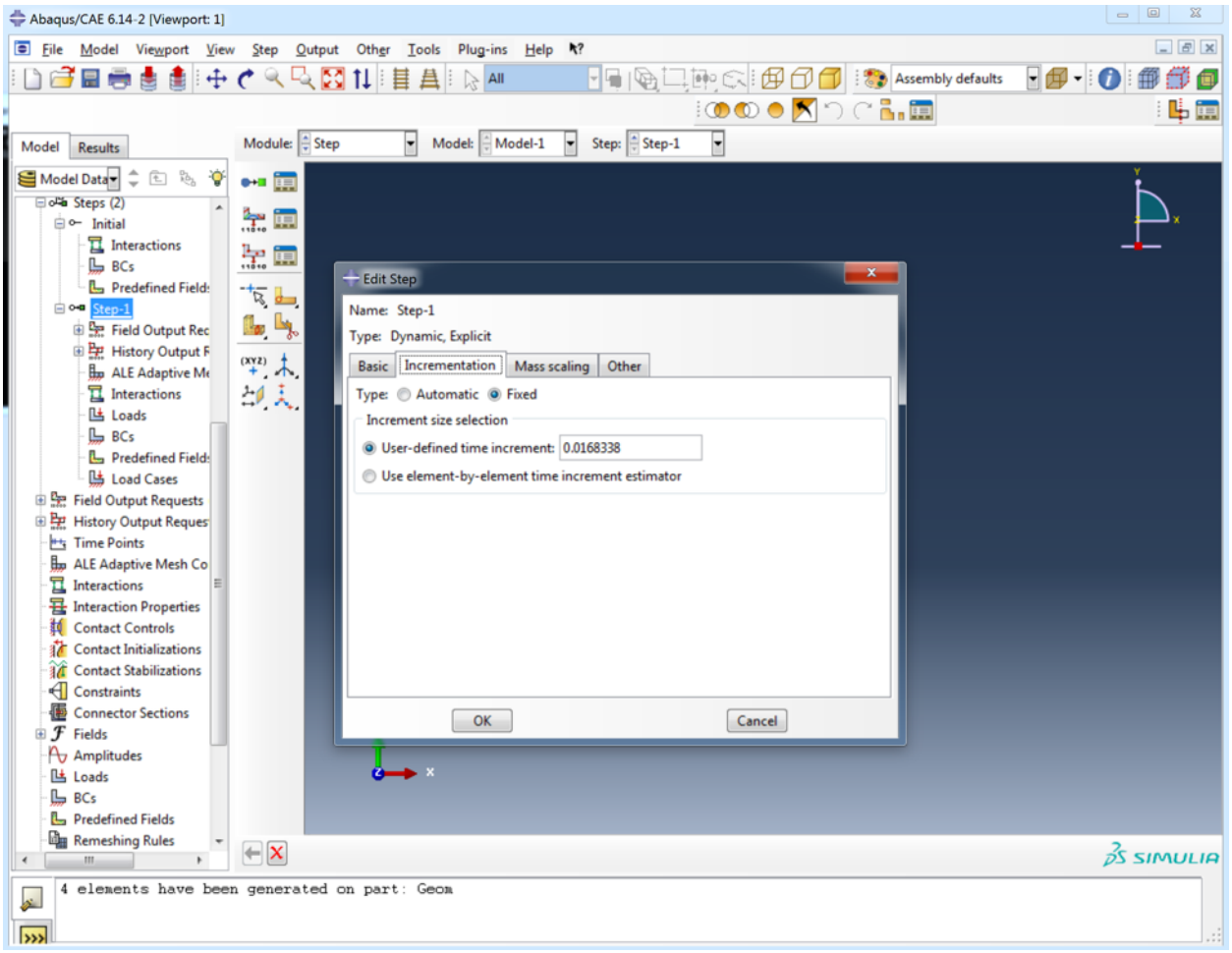


Figure 44: Specified time incrementation for dynamic analysis.

Before setting boundary conditions, the user must create an "Instance" of the geometry. This can be done under "Assembly." Now the user can apply boundary conditions. After creating "BCs," apply them to the Step created to define the time domain of the analysis. Under "Category," choose "Mechanical." Under "Types for Selected Step," choose "Displacement/Rotation." This can be seen in Figure 45.

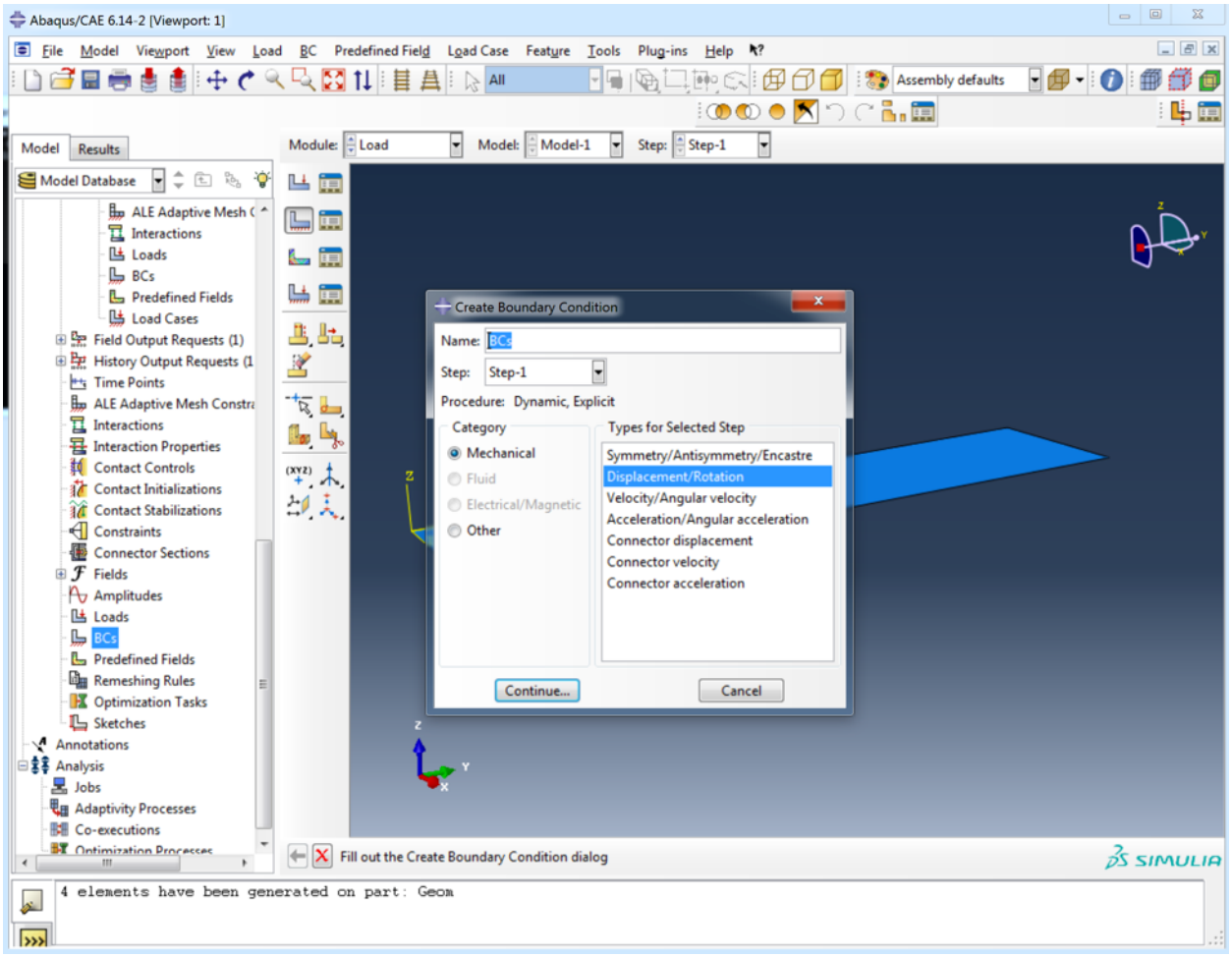


Figure 45: Specified time incrementation for dynamic analysis.

The user will be prompted to choose the location of the boundary conditions. Create a set from the region selected. Choose the appropriate constraints for the desired boundary conditions.

To apply loading, create a new "Load," choose "Mechanical" under "Category," and "Concentrated force" under "Types for Selected Step." Select the step generated to describe the time history. This will prompt the user to select the points that will be receiving the applied load. Create a set from the region selected. In the "Edit Load" dialogue box, specify the vector components of the direction of the load. To specify the magnitude, click "Amplitude." Because CU-BEN needs an explicit definition of the applied load at each time step, it is best to import a tabulated version of the load function. This can be constructed easily in MATLAB or Excel and subsequently imported in. This is shown in Figure 46. The load will start after the first time step. If no load history is specified, BenPre will assume that the model does not have any load applied to it over time and will leave that portion of the CU-BEN input file empty.

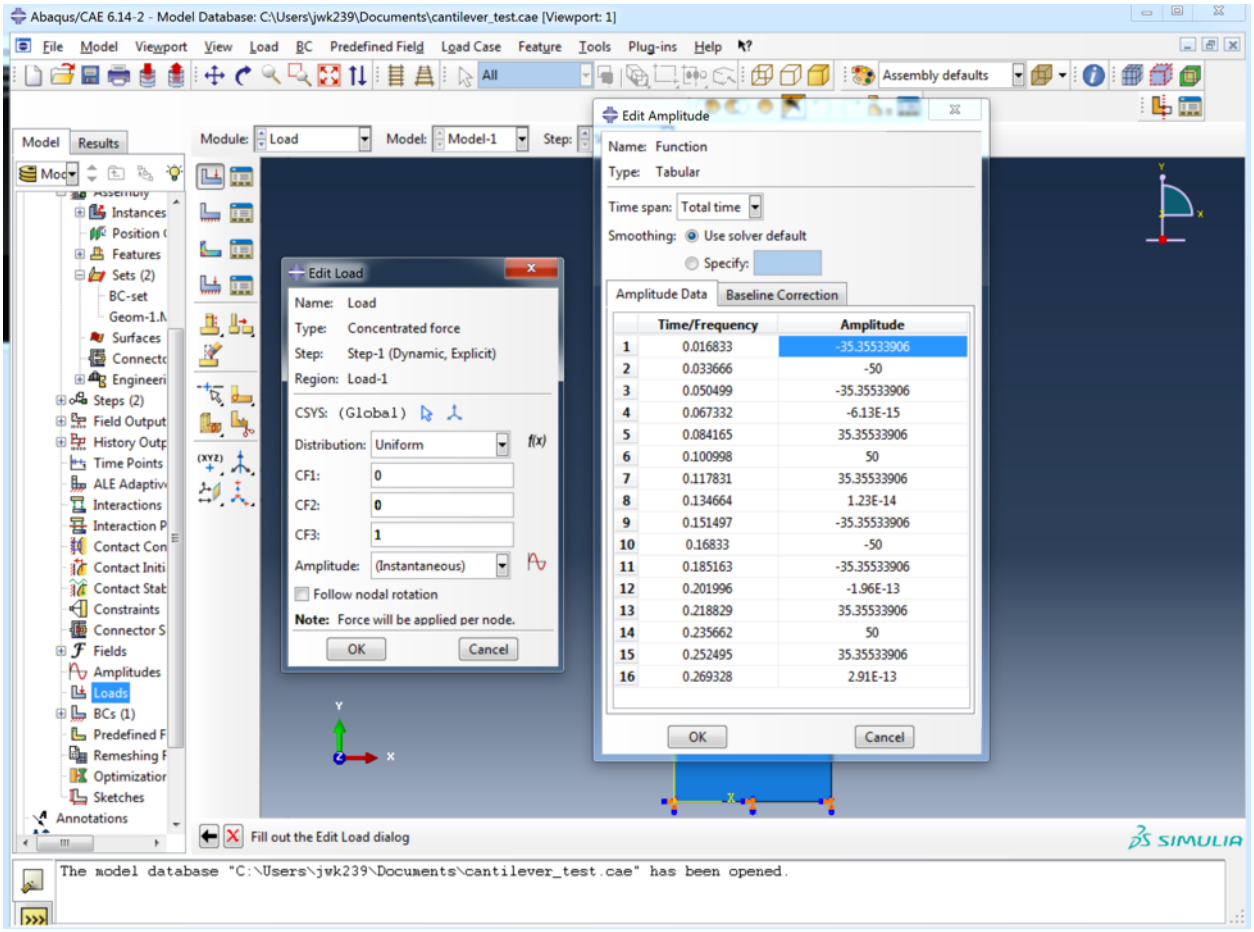


Figure 46: Specified load pattern for dynamic analysis.

If there are any loads on the structure at the initial time of zero seconds, create the Step "Time0" and apply non-zero loading on nodes within this step. If there are any initially prescribed displacements, velocities, or accelerations on any nodes in the model, also define them in Step "Time0". These prescribed conditions must be applied in Boundary Conditions and the user can specify the type (displacement, velocity, or acceleration), when creating the condition, seen in Figure 45. If the load on the structure is zero at the initial time of the analysis and there are no initial prescribed conditions, such as in the case of our example, this step can be skipped.

If applying nonzero displacement boundary conditions, i.e. imposing a translational time history on a degree of freedom, the user must define this within Abaqus as history data. For the example described in Section 5.3, nonzero displacement boundary conditions are not applied; thus, if the user is intending to build the CU-BEN input file described in the example, they may proceed to create their Abaqus file for use in BenPre.py. For completeness, the application of nonzero boundary conditions in Abaqus in a manner that will cohere to the protocol around which BenPre.py was developed will be described below. The tutorial images shown will apply the pre-

scribed root motion similar to that described in the example in Section 5.4 to the example described in Section 5.4.

To define nonzero displacement boundary conditions, create another set of "BCs," and apply them to the Step created to define the time domain of the analysis. Under "Category," choose "Mechanical." Under "Types for Selected Step," choose "Displacement/Rotation." This is identical to the definition of boundary conditions seen in Figure 45. The user will be prompted to choose the location of the boundary conditions. Create a set from the region selected. Choose the appropriate constraints for the desired boundary conditions. In a similar manner to the definition of the load time history, an "Amplitude" must be specified with a tabulated displacement value in a particular direction for each time step. The specification of constraints and definition of nonzero boundary condition time history can be seen in Figure 47. The displacement will begin after the first time step. Specify the appropriate Amplitude under the Amplitude drop-down tab. For the case seen in Figure 47, "Amp-2" would be specified.

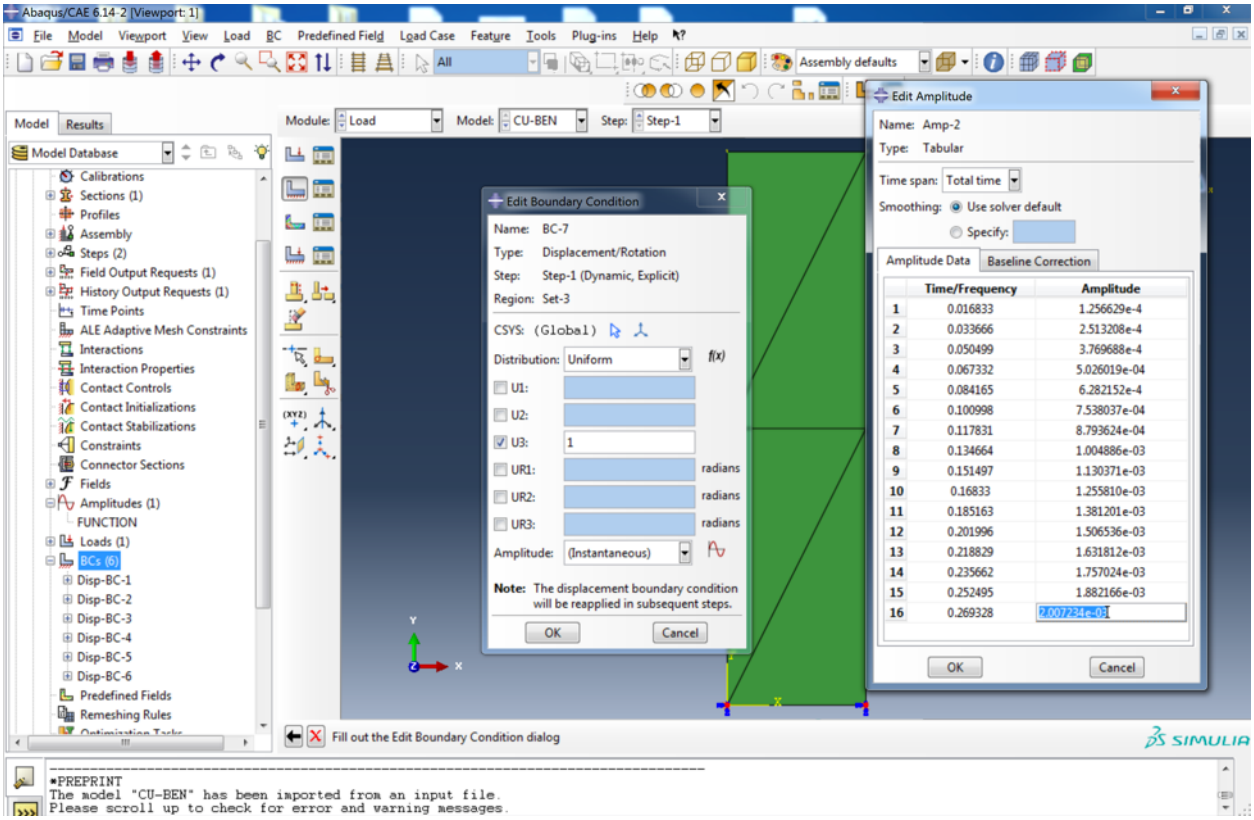


Figure 47: Specified translation time history if specifying nonzero displacement boundary conditions for dynamic analysis.

Once all features of the analysis have been defined, the user must create a "Job" named "CU-BEN." Once the Job has been created, right-click on the Job to write an input file. BenPre will prompt the user to specify the name of the Abaqus generated file. This file will be used by the BenPre code to create a model_def.txt input file. Place this file in the same folder as BenPre.

To execute BenPre.py, issue the command "python BenPre.py" (without quotes) within the command line.

The user will first be prompted to specify the file name of the Abaqus input file. The user will then enter the analysis type, solution algorithm type, solver algorithm type, and to choose to "execute node-renumbering" in the command line. Options for each of the analysis types and solution flags will be provided and the user is to enter the numerical flag associated with their choice. The code then will access the Abaqus input file to transcribe the geometry established in Abaqus to be read by CU-BEN. Depending on the choice of analysis/algorithm, the user will be prompted to enter additional parameters for the solution. If the prompt requires more than one value, a

guide will be printed below the prompt to help the user input the values correctly.

The code will output a file named "model_def.txt". This can be used as an input file in CU-BEN. But as a reminder, please review the contents of the input file to ensure the model was translated from the Abaqus file correctly and all necessary solution parameters are specified.

8 Using VTK Files to Visualize CU-BEN Results

Because the results printed in the output files are not always easily readable by humans (easy to import into Exel or read with a Python script), a rudimentary post processor was developed for CU-BEN, BenPost.c, and included in the GitHub download of CU-BEN. BenPost.c converts the information in the results*.txt files to *.vtu files that can be opened in ParaView and saved as a *.vtk file, or exported as a video or image file. ParaView is freeware that can be downloaded at http://www.paraview.org/. Many different versions of ParaView exist for different operating systems and each different version of ParaView can be downloaded. The user should download the ParaView that corresponds with their operating system. Figures 48 - 51 are taken from ParaView 4.1 running on Mac OS X 10.11.

BenPost must be kept in the same folder as CU-BEN's results*.txt output files. After creating and running the BenPost executable, a "master.pvd" file will output for ParaView accessibility. A folder will also be output named "vtu_files" which will contain a "slave_*.vtu" file for each time step of the CUBEN simulation if running a dynamic analysis, and for each LPF if running a nonlinear analysis. This folder and its contents can be opened in ParaViewn and can subsequently be saved as a *.vtk file. This can be seen in Figure 48. After opening the "slave_*.vtu" files, choose the variables you would like graphic representations for, and hit "Apply". Examples images are provided in Figures 49 - 51. The FE mesh shown is a portion of a tow tank model for the Joint High Speed Sealift (JHSS) vessel. It has 45,874 joints, 91,776 shell elements, and 271,428 DOFs with a horizontally applied linearly increasing load.

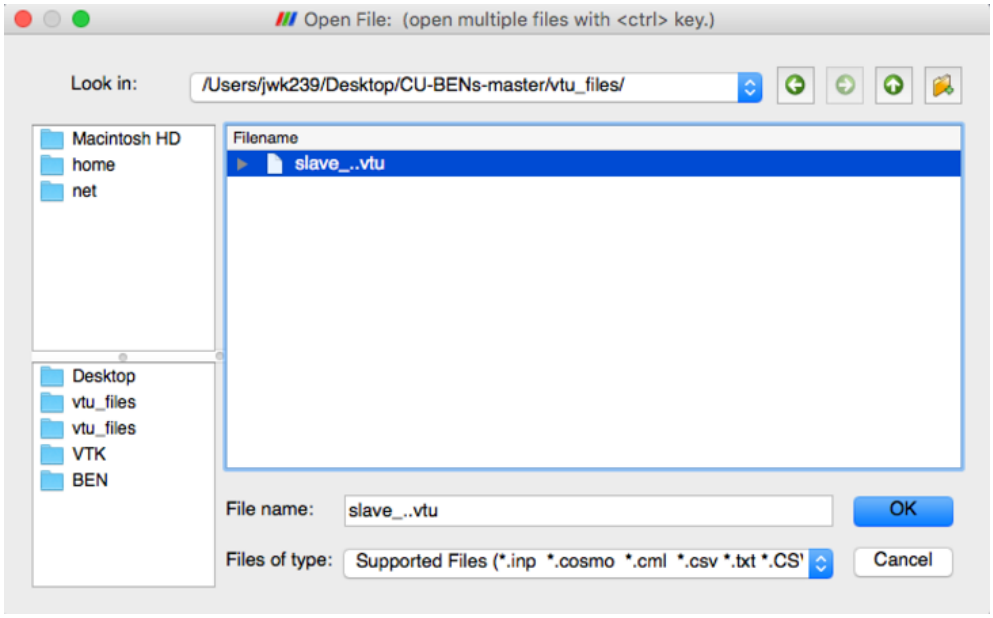


Figure 48: Opening files created by BenPost in ParaView.

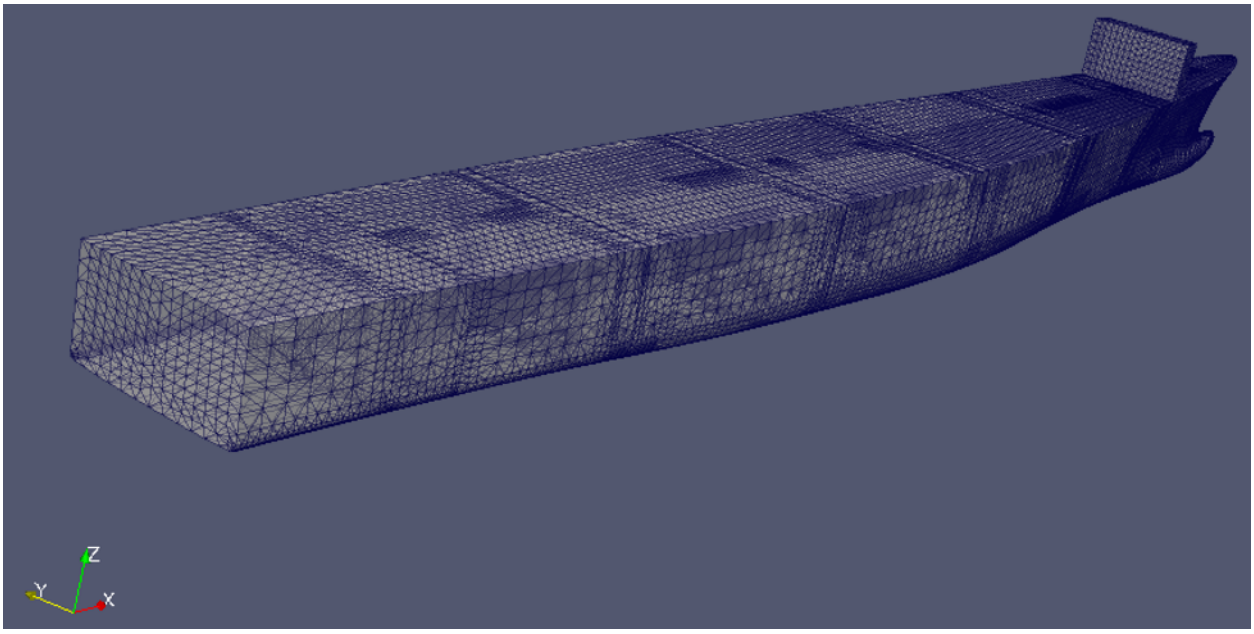


Figure 49: FE geometry for JHSS model.

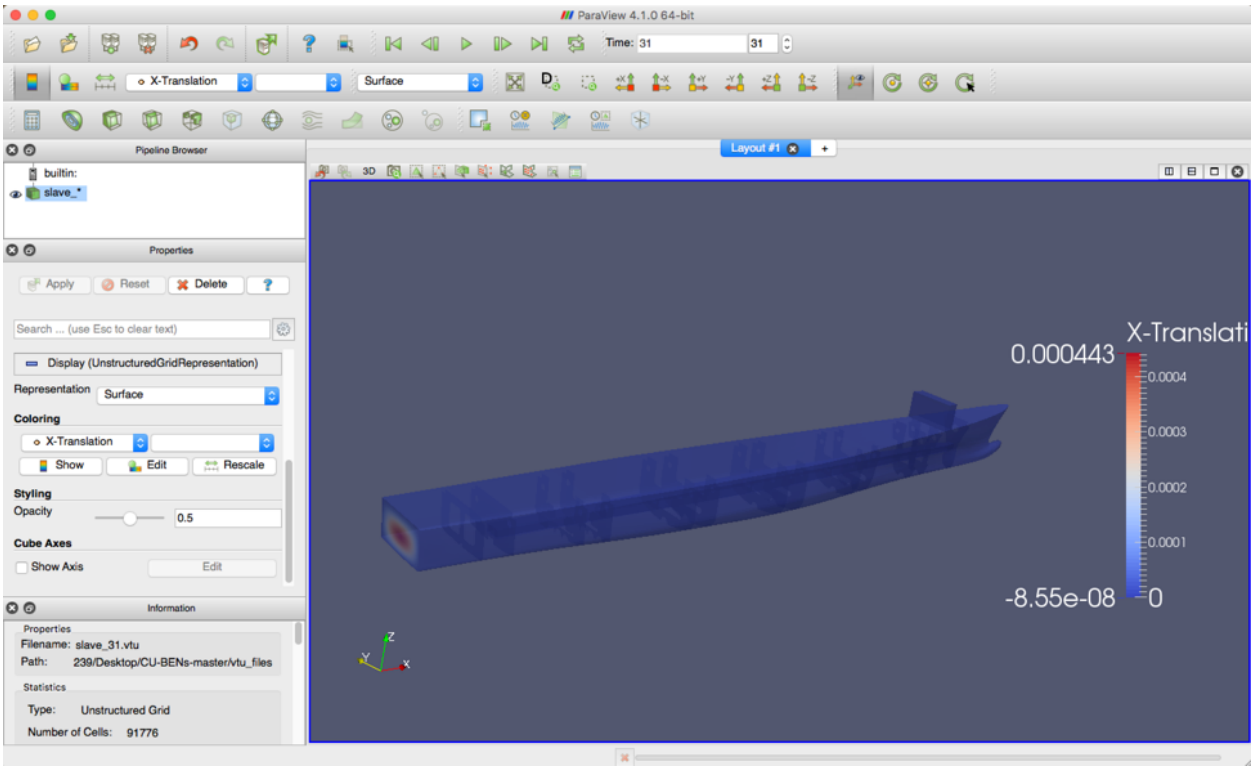


Figure 50: X- translation graphic created by BenPost and viewed in ParaView.

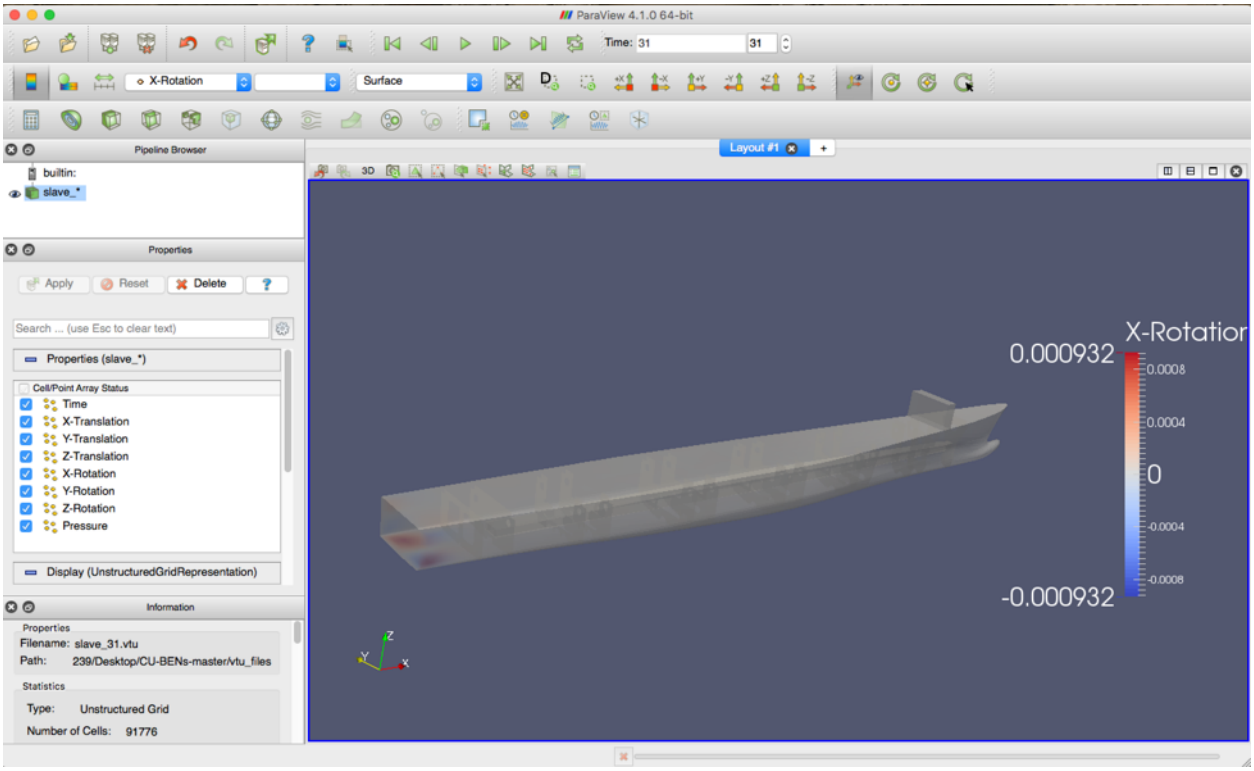


Figure 51: X- rotation graphic created by BenPost and viewed in ParaView.

When using ParaView, it is highly recommended to use a USB 3 button mouse if operating on a laptop with a touchpad. To zoom, hold down on the right mouse button. To rotate, hold down on the left mouse button. To pan, hold down on the middle mouse button. These camera motions can also be achieved through icons in the Camera Controls panel of the interface bar. When using the Selection Display Inspector to identify node numbers, it is important to be aware that node numbering begins at 1 in CU-BEN, but at 0 in ParaView. All node IDs identified in ParaView will correspond to the node ID + 1 in CU-BEN. For the most descriptive color scale, it is good practice to rescale the magnitude color bar at the time step where the model experiences the magnitudes are the largest in both the positive and negative directions for the motion the user is graphically representing. To export an animation, simply click on "File >Save Animation" and a dialogue box will prompt the user to input animation settings.

9 Plotting Structural Responses from CU-BEN

To assist in visualizing the displacement response for one joint, a rudimentary plotting routine, BenPlot.py, has been developed for CU-BEN. BenPlot extracts data from the displacement response file, results2.txt, and plots the structural response at a user-specified joint.

Additional Cygwin/X server packages and matplotlib are necessary in order for BenPlot.py to run successfully. Instructions for installing Cygwin/X server can be found at https://x.cygwin.com/docs/ug/setup.html.

To install matplotlib for plotting figures in Python:

1. Install pip package by entering "python -m ensurepip" in the command line.
2. Enter "pip install matplotlib".

To run BenPlot, keep the file in the in same folder as CU-BEN's results1.txt and results2.txt files. Users would first have to launch Cygwin/X server by entering "xlaunch" in the command line. Then, follow Figure 52 to Figure 56 to setup Cygwin/X server terminal.

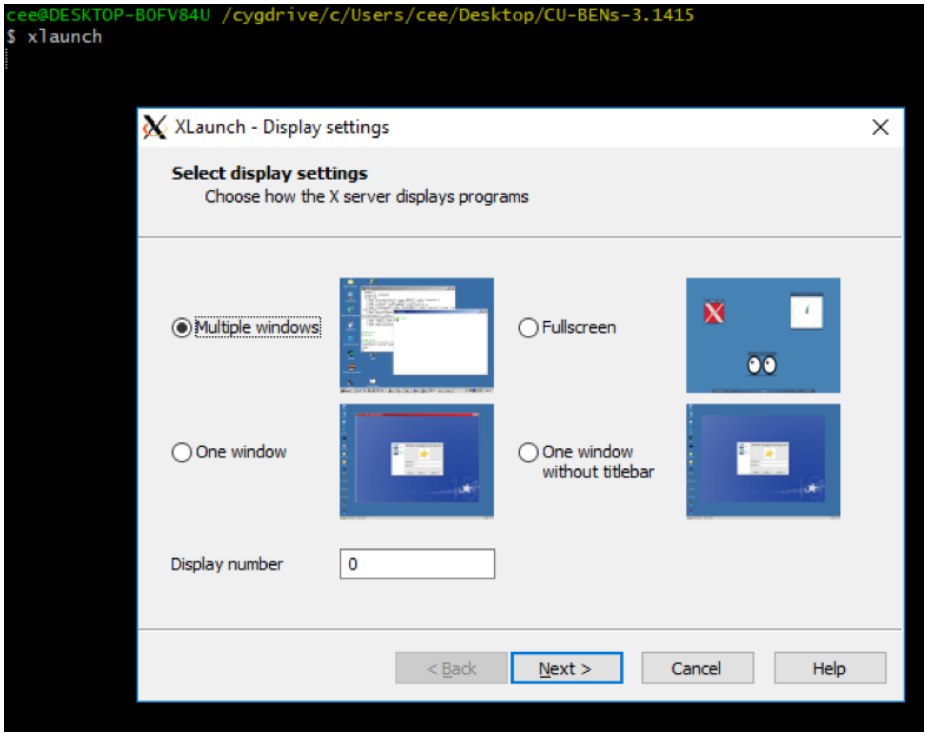


Figure 52: Specify display settings.

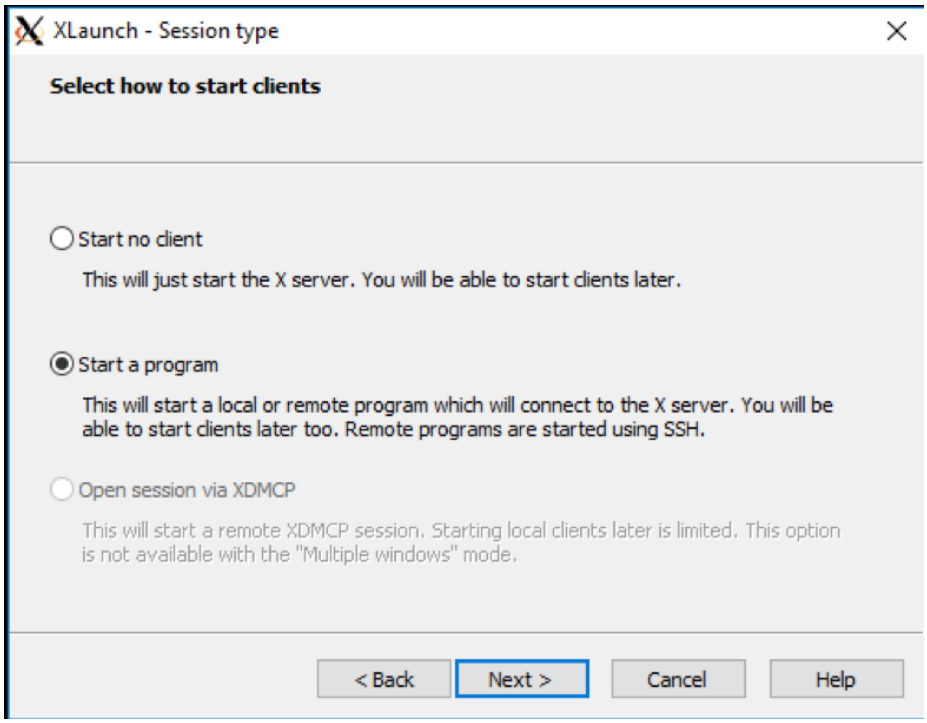


Figure 53: Specify session type.

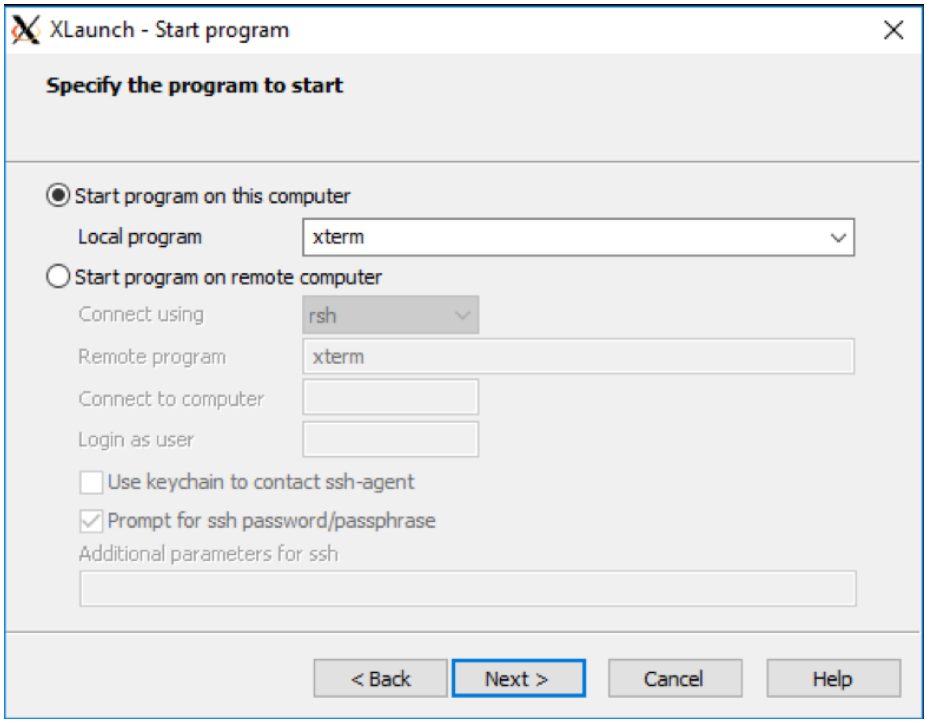


Figure 54: Start the program.

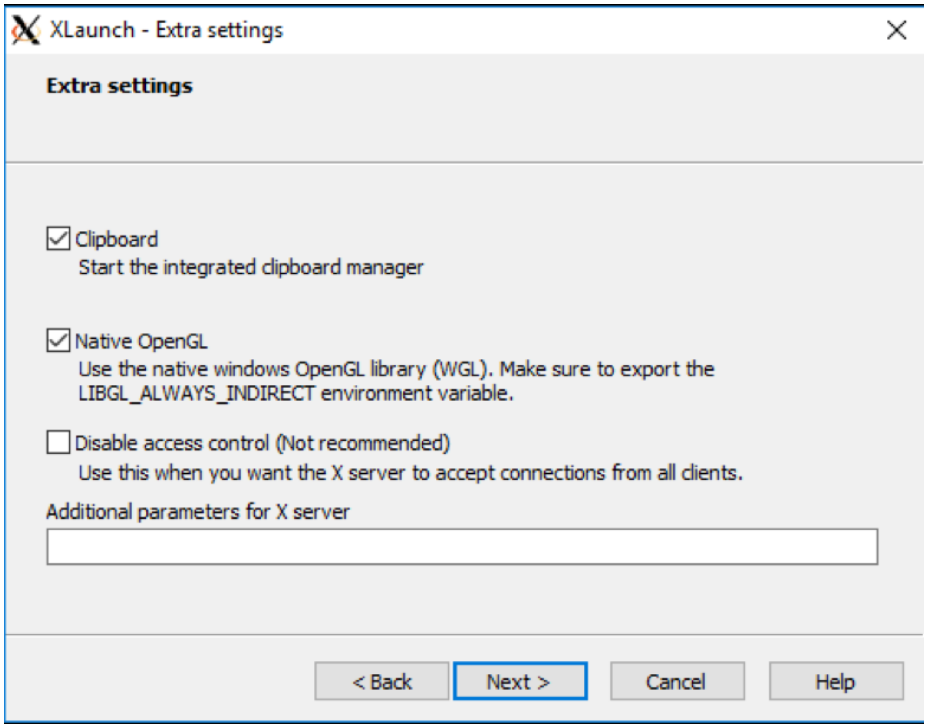


Figure 55: Specify extra settings.

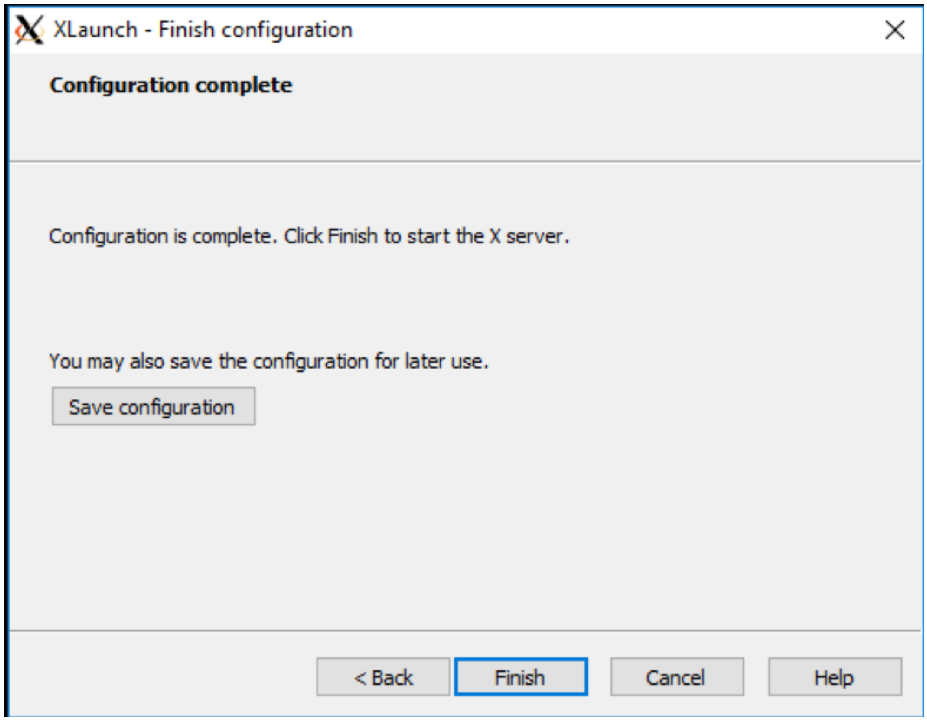


Figure 56: Configuration setup complete.

Once Cygwin/X terminal is launched successfully, access CU-BEN directory as shown in Figure 57. Then enter "python BenPlot.py" (without quotes) in the command line. The user will be prompted to input the xyz-coordinates pertaining to the joint of interest. If the combination of coordinate values is not found, the routine will output an error and terminate. If the coordinates do not have active degrees of freedom, the routine will output an error and terminate.

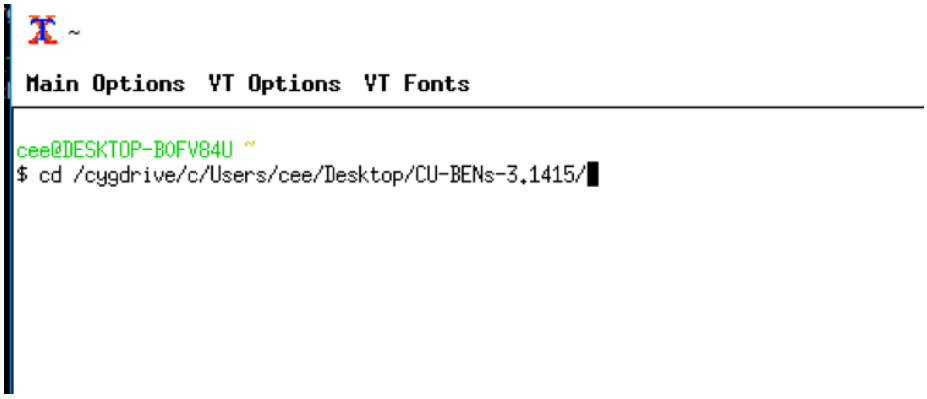


Figure 57: Access CU-BEN directory in Cygwin/X terminal.

For each DOF associated with the specified joint, the routine will create a plot. If the analysis is dynamic, the load response will be plotted against the solution time. If the analysis is static and nonlinear, the load proportionality factor will be plotted against the load response. The plots are outputted based on their DOF ordering. If the user would like to save the plot, they will be prompted about the path for the save. When the user closes the window containing the plot, the next DOF associated with the given joint will be plotted. An example of the plot created can be found in Figure 58. This plot was creating using results files from the shell cantilever example used in the Sections 5.3 and 7. The translation in the z- direction is plotted at a node at the tip of the cantilever.

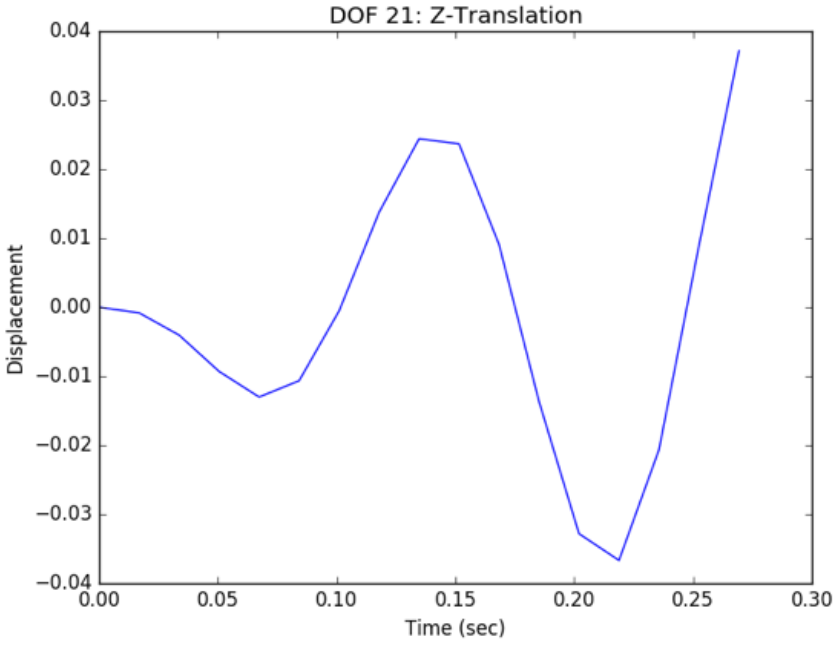


Figure 58: Plot of translation in the z- direction over time for the shell cantilever example used in Sections 5.3 and 7 created by BenPlot.

References

[1] K.J. Bathe and H. Saunders. *Finite element procedures*. Prentice Hall International, 1996.

[2] J. Chung and G.M. Hulbert. A time integration algorithm for structural dynamics with improved numerical dissipation: the generalized-α method. *Journal of applied mechanics*, 60(2):371–375, 1993.

[3] M.A. Crisfield, J.C. Remmers, and C.V. Verhoosel. *Nonlinear finite element analysis of solids and structures*. John Wiley & Sons, 2012.

[4] G.C. Everstine. Finite element formulations of structural acoustics problems. *Computers & Structures*, 65(3):307–321, 1997.

[5] W. McGuire, R.H. Gallagher, and R.D. Ziemian. *Matrix structural analysis*. John Wiley & Sons, 1979.