Notes in Coupling FERUM and ZEUS-NL

Nophi Ian Biton

January 2, 2022

1 Introduction

This document aims to outline basic steps in performing reliability analysis through **F**inite **E**lement **R**eliability **U**sing **M**atlab [2] coupled with external structural analysis software such as ZEUS-NL [3]. Uncertainties in engineering is inevitable, therefore, it should be accounted in design, analysis, and decision making. Structural reliability is a research field in estimating the reliability of structures under extreme loads and long term service loads considering in material, structural model applied etc. A detailed discussion about structural reliability can be found in [6] and [5].

The basic formulation of a reliability analysis problem is that given random variables X, the question is asked: what is the probability of failure $P_f = P(g(X) < 0)$ where $g(X)$ is the limit state function. The simplest expression of limit state function is $g(X) = R - S$ where R is the capacity (also called resistance) and S is the demand (also called load effects). In structural reliability, there are three states that are describe:

$$g(X) = R - S \begin{cases} > 0 & Safe\ State \\ = 0 & Limit\ State \\ < 0 & Failure\ State \end{cases}$$

Both R, S can be defined as random variables with corresponding probability density function. The uncertainty concerning the demands or load effects S can be associated to earthquake, wind loads or even live loads and dead loads while the uncertainty in the capacity or resistance R can be associated to concrete strength, yield strength, modulus of elasticity etc. There are a couple of methods to perform reliability analysis [5]. The most common ones are First Order Reliability Method (FORM) and Monte Carlo Simulation (MCS). The theory behind these methods will not be discussed in this document but rather how to use a freely available tool (i.e. FERUM) that is able to perform these analyses.

The limit state function described above is not always explicitly provided such in a case where complicated structural analysis is performed to determine non-linear behaviour of structures. In earthquake engineering, the structure may experience forces beyond the linear behaviour of components or system, thus, non-linear analysis must be used. Static and/or dynamic non-linear analysis that both considers geometrical and material non-linearity are commonly used. The inherent variability due to the ground motions as well as the uncertainty in the material properties can be accounted in performing reliability analysis. This is where the ZEUS-NL can be used to perform non-linear analysis of the structures. The reliability analysis using FERUM can be coupled with ZEUS-NL in order to evaluate the limit state function in terms of non-linear response of a structure. This methodology was first implemented in seismic fragility analysis [4].

1.1 Reliability Analysis using FERUM

Finite **E**lement **R**eliability **U**sing **M**atlab or FERUM is an open source Matlab code that is able to perform a couple of analysis in structural reliability. It was first developed by researchers from the University of California Berkeley [1]. The FERUM version 3.0 can be downloaded at `http://projects.ce.berkeley.edu/ferum/`. The latest version of FERUM is maintained at the Institut Français de Mécanique Avancée

(IFMA) in Clermont-Ferrand, France. The FERUM version 4.x includes new features such as Subset Simulation, Global sensitivity and Reliability-Based Design Optimization algorithm. It also provides readily available functionality for distributed computing. More details about the latest version can be found in its user manual [2].

Getting started with FERUM 4.x

In order to run FERUM in Matlab, do the following steps:

1. Visit the webpage `https://www.sigma-clermont.fr/en/ferum_download`. Fill-up the form and a download link will be sent to the registered email addres.
2. Download the FERUM4.0.zip and extract all the files at a preferred location in your computer.
3. Before FERUM can be used in Matlab, all the FERUM subdirectories must be included in the Matlab Search Path. Follow the following steps:
   - Select "Set Path" in the Home tab
   - Select "Add with Subfolders" the folder where the codes was unzipped previously
   - The directories will be shown in the pop window (see Figure 1) and click "Save".
4. Load an example input file in Matlab. Issue the command `>> inputfile_xxx` in the Matlab command prompt. This will load all necessary parameter to the current Matlab workspace.
5. Issue the command `>> ferum` into the command prompt in order to start the FERUM analysis
6. Choose a specific option (e.g. 10 for FORM). FERUM will prompt the user to choose an analysis type when `analysisopt.echo_flag` is set to `1` other wise set it to `0` and define `analysisopt.analysistype` to preferred analysis option (see `ferum.m` for the list of options).
7. The program will start the analysis and gives intermediate information as it runs.
8. If the analysis is successful, the results will be saved in the workspace. The results will be saved in different variable names for different methods used (e.g. `formresults` for FORM).

Input files for FERUM

The parameters that needs to be declared in order to perform analysis in FERUM is demonstrated in the file `inputfile_template.m`. Additionally, the different methods available in FERUM is discussed in the user manual [2]. The contents of the input files will be discussed briefly here.

A well defined reliability analysis problem should include information about the probability distribution of the considered random variables and definition of the limit state function (or sometimes called performance function). The parameters (with *structure array* data type in Matlab) needed in the input file for FERUM can be divided into the following:

1. `probdata` - All information about the probability distribution, correlation matrix and transformation method of jointly distributed variables are defined here. There are a couple of *fields* in `probdata` such as `probdata.name` an optional field for naming the random variable with array of strings, `probdata.marg` which is an array of values that defines the type, mean, standard deviaton, etc. See `inputfile_template.m` for more information of the required fields of `probdata`.
2. `analysisopt` - The necessary information regarding analysis is defined here. The options regarding the analysis mode, block size g-calls, options in FORM, SORM, MCS etc can be modified here. When the user wants to run a specific method only (e.g. FORM) then other fields for analysis type can be deleted. For trouble shooting the analysis, the default values may be changed in order to converge to a desirable result over a much lesser time.

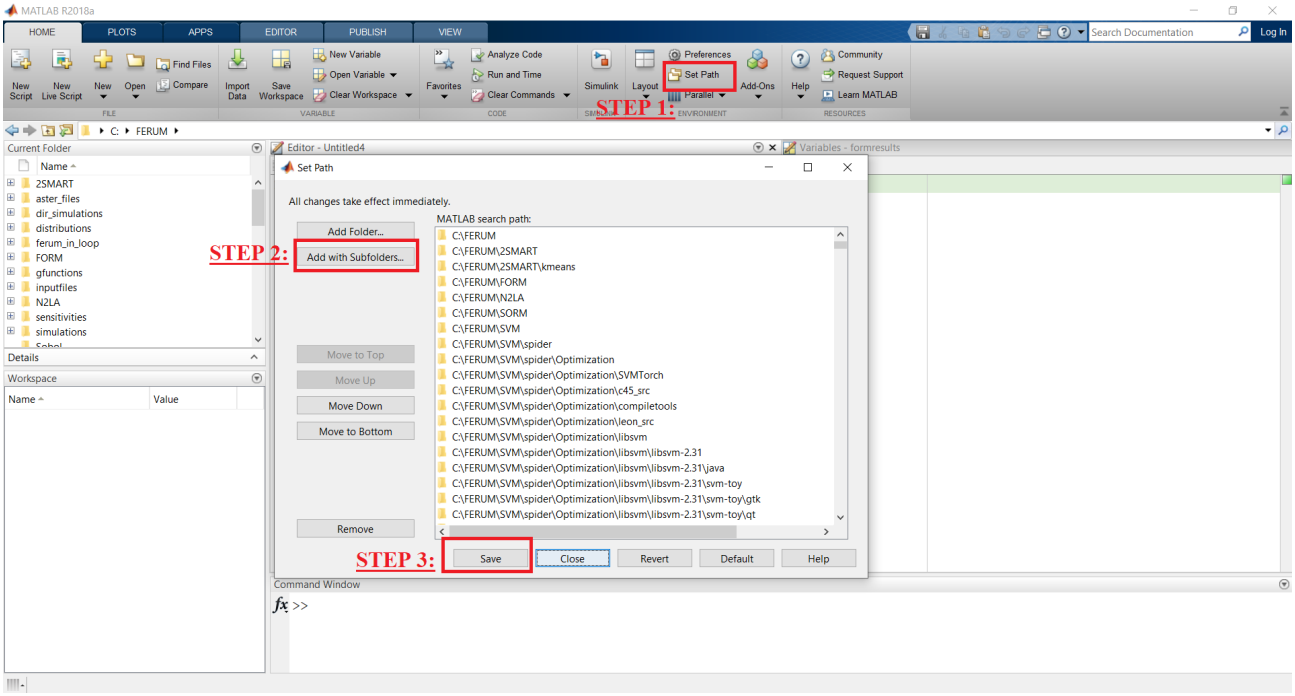

Figure 1: Add FERUM codes to Matlab Search Path

Additional notes

- When `analysisopt.echo_flag` is set to `1` (interactive mode) upon issuing the command `>> ferum`, the program will ask the user to enter the analysis type. However, if it is set to `0` (silent mode) then the user needs to define `analysisopt.analysistype` to a preferred analysis option (see `ferum.m` for the list of options).
- When `analysisopt.grad_flag` is set to `'ddm'` (direct differentiation) then an additional field in `gfundata.dgdq` must be defined which is an array of strings pertaining to the partial derivatives of the limit state function.
- When `analysisopt.rand_generator` is set to 1 (Mersenne Twister) an error will occur when MCS or IS is conducted. This occurs because the `twister.m` was considered as a Matlab function. In order to debug this, replace all lines that uses `twister` into `rand` function. The default random generator type of `rand` function in Matlab already uses the Mersenne Twister. Alternatively, the `analysisopt.rand_generator` can be set to `0` for default `rand` Matlab function.

3. `gfundata` - The information about the limit state function is defined in this parameter. There are several ways to define the limit state function in the *field* `gfundata(1).expression`. A separate matlab function (`limit_state.m`) file can be created to define the limit state function and equate the `gfundata(1).expression = 'limit_state(x1,x2)'` including the input arguments needed in the function. The random variables must be declared in the same order as in `probdata.marg` using the names defined in `probdata.name` (optional). If the names are not defined then random variables can be referred in as `x1,x2,x3...` Alternatively, the limit state function can be defined explicitly such as `gfundata(1).expression = 'x2-x1'`. If direct differentiation is used additional field `gfundata(1).dgdq` should be defined.

4. `femodel` - This parameter should include the information about the FEM model to be used (such as ASTER). Currently, only aster FEM software is available in FERUM. The directories of the aster, model inputs, mesh generator should be defined here.

5. `rbdo_fundata` and `rbdo_parameters` - When a reliability-based design optimization is to be performed, then parameters about the objective function, deterministic constraints and target reliability are defined in these parameters.

1.2 ZEUS-NL for Structural Analysis

ZEUS-NL is a computer software used and was developed by MAE center for earthquake engineering applications. Its advantages of being efficient, accurate and user-friendly made it easier to perform inelastic analyses of sophisticated structures considering large displacement through fiber approach. It has a wide range of flexible material models and elements that can be used. ZEUS-NL is available free-of-charge.

Getting started with ZEUS-NL

In order to use ZEUS-NL in your computer, follow the steps:

1. Visit the webpage `http://mae.cee.illinois.edu/software/software_zeusnl.html` and click the hyperlink 'Download ZEUS-NL 1.9.0'. You will be directed to a Google Code Archive site where the files are stored.

2. Download the file 'ZeusNLInstallation_files_for_Windows_system_v1.9.0' (for windows) and other relevant documentation (user manual, license etc.).
   Also, download 'ZeusNL_PostProcessor_ZEUSPost_Source_Code_MATLAB' for Matlab post processing.

3. Unzip the installation files into a folder. Run the Setup.exe

4. Follow the instructions and the automated installation program will proceed to copy ZeusNL onto the hard drive. Normally, the default settings suggested by the installation should work well. ZeusNL shortcuts will be added to the desktop and to the Start Menu under `Programs > ZeusNL`.

Features of ZEUS-NL

Upon completing the installation, two shortcut icons will be created in the desktop, namely: the 'ZeusNL' for model definition and analysis and 'ZeusView' for post processing of results.

An example interface of ZEUS-NL is shown in Figure 2. A model viewer of the location of nodes and element connectivity is embedded in the software. The step-by-step procedures in modelling and performing analysis is outlined in its user manual [3]. The steps consist of defining parameters in each modules (Materials, Sections Element Classes, Nodes, Element Connectivity, Restraints, Applied Loading). The modules to be defined also vary with the analysis to be performed. The different analysis that can be done in ZEUS-NL are:

- Eigenvalue Analysis
- Static Constant Analysis
- Static Pushover Analysis
- Static Adaptive Pushover Analysis
- Static Time History Analysis
- Dynamic Time History Analysis

There are a couple of features that the program can offer. It is able to account a wide range of tested concrete and steel models. It also accounts for both material and geometrical non-linearities. It models spread of inelasticity along the member length apart from other software that uses lumped inelasticity model. Its input and output files is well integrated to windows operating system environment. It also offers Template facility for faster model creation.

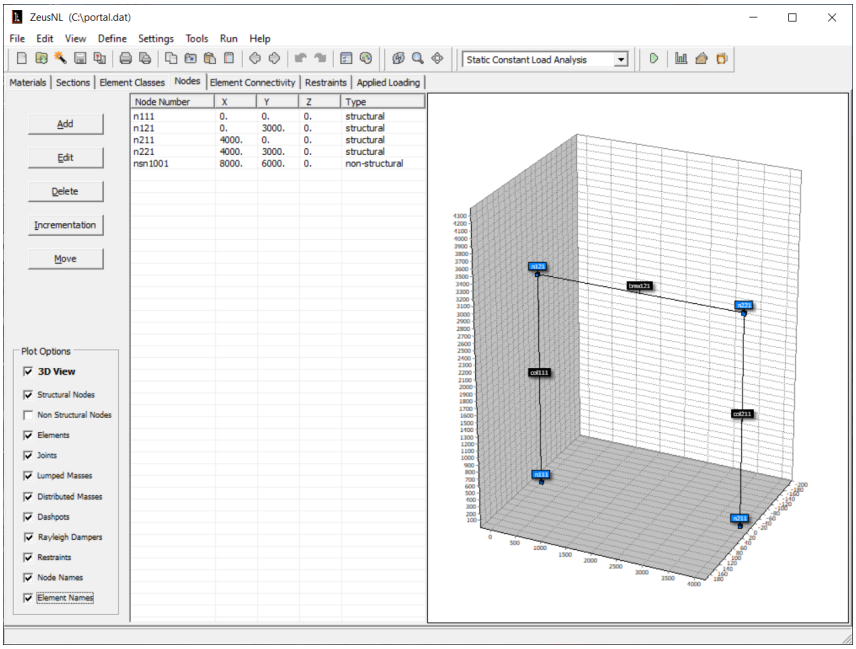

Figure 2: ZEUS-NL Platfrom

2 Running ZEUS-NL through MATLAB

There is no direct way of accessing or running ZEUS-NL through Matlab commands. However, a feature that lets ZEUS-NL to be run in command window through batch mode can be used.

Batch Mode

A detailed explanation of how to run in batch mode is included in the Batch_Mode.pdf downloaded along side installation files of ZEUS-NL.

ZEUS-NL analysis engine is basically composed of two executable files: "Reader" and "Solver". "Reader" reads the ZEUS-NL input files with "dat" extension and creates a set of files to be read by the "Solver". The user can check whether the input file is successfully read or not from the file with "out" extension created by the "Reader". This file is copy of the input file that includes errors (if any). For "Solver" to run there should be no error messages in this "out" file.

The syntax for running analysis in batch mode is provided in Figure 3. Note that the executable files "Reader" and "Solver" as well as the input ZEUS-NL file has to be in the same directory.

The set of commands provided in Figure 3 can be collected in a file with "bat" extension (for windows systems) such as "RunZEUS.bat" and can be run all at once by calling "RunZEUS.bat". Note that the commands that are shown in blue are dos commands. For other operating systems one has to replace these commands with their equivalents.

If this procedure is to be repeated for several analyses, the user has to make sure that the "result.num" file is deleted before running the next analysis, otherwise, ZEUS-NL will not write on the existing file. To do this automatically the following line could be added to the end of the above set of commands: "del result.num".

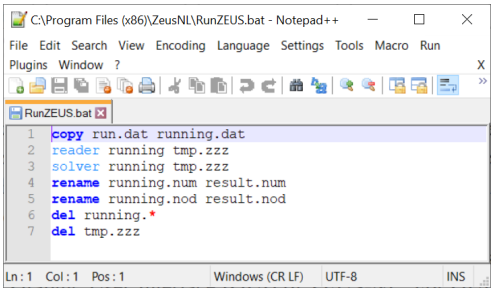

Figure 3: Syntax for running ZEUS-NL in batch mode.

Perform ZEUS-NL analysis through Matlab

The input and output files of ZEUS-NL can also be accessed through any text editor. This flexibility allows other scripting languages such as Matlab to use ZEUS-NL externally. The general steps in accessing ZEUS-NL through other scripting language are:

1. Create a ZEUS-NL input file in *.dat format following the Input_Output_File_Format_Description.pdf that can be downloaded along side installation files of ZEUS-NL. Matlab functions `edit` to create a file and `fopen` to manipulate the contents of the file in accordance to the FERUM format can be used.

2. Run ZEUS-NL through batch mode. If the commands are save in a batch file (e.g. see Figure 3), then the command `system('RunZEUS.bat')` can be issued in Matlab.

3. Temporarily pause Matlab using `pause()` command until the ZEUS-NL is finished running and produces the *.num file. A loop with `exist()` command of Matlab can be used until the *.num file is created.

4. Read the contents of the results of *.num using `fopen` command of Matlab. Alternatively, the `ZEUSPostM.m` can be used.

5. If the analysis is to be repeated, then *.num file should be deleted using dos command `del *.num`.

3 Couple FERUM and ZEUS-NL

In order to perform a reliability analysis using FERUM and ZEUS-NL together several codes or files must be prepared:

- **ferum_inputfile.m** - This will be the input file that is needed to run FERUM. The parameters declared in this file must be loaded first in the Matlab workspace before FERUM analysis can start. An example file is shown in Listing 1.

Listing 1: Sample FERUM input file.

```
clear probdata femodel analysisopt gfundata randomfield systems results output_filename
output_filename = 'simple.txt';


%% PROBDATA
% marginal distribution for each random variable
probdata.marg(1,:) = [ 1 1000 100 1000 0 0 0 0 0];
probdata.marg(2,:) = [ 1 200e3 1000 200e3 0 0 0 0 0];

%correlation matrix
probdata.correlation = eye(2);
% type of joint distribution
probdata.transf_type = 3;

% Method for computation of the modified Nataf correlation matrix
probdata.Ro_method   = 1;
```

```
17
18 % Flag for computation of sensitivities w.r.t. means, standard deviations, parameters and
       correlation coefficients
19 probdata.flag_sens   = 1;
20
21 probdata.parameter = distribution_parameter(probdata.marg);
22
23 %% ANALYSISOPT
24 analysisopt.echo_flag = 1;                    % 1: FERUM interactive mode, 0: FERUM silent mode
25 %analysisopt.analysistype = 10;
26
27 analysisopt.multi_proc           = 1;         % 1: block_size g-calls sent simultaneously
28                                                %    - gfunbasic.m is used and a vectorized version of
                                                       gfundata.expression is available.
29                                                %      The number of g-calls sent simultaneously (
                                                       block_size) depends on the memory
30                                                %      available on the computer running FERUM.
31                                                %    - gfunxxx.m user-specific g-function is used and
                                                       able to handle block_size computations
32                                                %      sent simultaneously, on a cluster of PCs or any
                                                       other multiprocessor computer platform.
33                                                % 0: g-calls sent sequentially
34 analysisopt.block_size           = 500000;    % Number of g-calls to be sent simultaneously
35
36 % FORM analysis options
37 analysisopt.i_max                = 100;       % Maximum number of iterations allowed in the search
       algorithm
38 analysisopt.e1                   = 0.001;     % Tolerance on how close design point is to limit-
       state surface
39 analysisopt.e2                   = 0.001;     % Tolerance on how accurately the gradient points
       towards the origin
40 analysisopt.step_code            = 0;         % 0: step size by Armijo rule, otherwise: given value
       is the step size
41 analysisopt.Recorded_u           = 1;         % 0: u-vector not recorded at all iterations, 1: u-
       vector recorded at all iterations
42 analysisopt.Recorded_x           = 1;         % 0: x-vector not recorded at all iterations, 1: x-
       vector recorded at all iterations
43
44 % FORM, SORM analysis options
45 analysisopt.grad_flag            = 'ffd';     % 'ddm': direct differentiation, 'ffd': forward finite
        difference
46 analysisopt.ffdpara              = 50;      % Parameter for computation of FFD estimates of
       gradients - Perturbation = stdv/analysisopt.ffdpara;
47                                               % Recommended values: 1000 for basic limit-state
                                                       functions, 50 for FE-based limit-state functions
48 analysisopt.ffdpara_thetag      = 50;      % Parameter for computation of FFD estimates of
       dbeta_dthetag
49                                               % perturbation = thetag/analysisopt.ffdpara_thetag if
                                                       thetag ~= 0 or 1/analysisopt.ffdpara_thetag if
                                                       thetag == 0;
50                                               % Recommended values: 1000 for basic limit-state
                                                       functions, 100 for FE-based limit-state functions
51
52 % Simulation analysis (MC,IS,DS,SS) and distribution analysis options
53 analysisopt.num_sim              = 1000;      % Number of samples (MC,IS), number of samples per
       subset step (SS) or number of directions (DS)
54 analysisopt.rand_generator       = 0;         % 0: default rand matlab function, 1: Mersenne Twister
        (to be preferred)
55
56 % Simulation analysis (MC, IS) and distribution analysis options
57 analysisopt.sim_point            = 'origin'; % 'dspt': design point, 'origin': origin in standard
       normal space (simulation analysis)
58 analysisopt.stdv_sim             = 1;         % Standard deviation of sampling distribution in
       simulation analysis
59
60 % Simulation analysis (MC, IS)
61 analysisopt.target_cov           = 0.05;      % Target coefficient of variation for failure
       probability
62 analysisopt.lowRAM               = 0;         % 1: memory savings allowed, 0: no memory savings
       allowed
63
64 % Directional Simulation (DS) analysis options
65 analysisopt.dir_flag             = 'random'; % 'det': deterministic points uniformly distributed on
        the unit hypersphere using eq_point_set.m function
66                                               % 'random': random points uniformly distributed on the
                                                       unit hypersphere
67 analysisopt.rho                  = 8;         % Max search radius in standard normal space for
       Directional Simulation analysis
68 analysisopt.tolx                = 1e-5;      % Tolerance for searching zeros of g function
69 analysisopt.keep_a               = 1;         % Flag for storage of a-values which gives axes along
       which simulations are carried out
```

```
analysisopt.keep_r                = 1;        % Flag for storage of r-values for which g(r) = 0
analysisopt.sigmafun_write2file   = 0;        % Set to 0

% Subset Simulation (SS) analysis options
analysisopt.width                 = 2;        % Width of the proposal uniform pdfs
analysisopt.pf_target             = 0.1;      % Target probability for each subset step
analysisopt.flag_cov_pf_bounds    = 1;        % 1: calculate upper and lower bounds of the
     coefficient of variation of pf, 0: no calculation
analysisopt.ss_restart_from_step = -inf;      % i>=0 : restart from step i, -inf : all steps, no
     record (default), -1 : all steps, record all
analysisopt.flag_plot             = 0;        % 1: plots at each step (2 r.v. examples only), 0: no
     plots
analysisopt.flag_plot_gen         = 0;        % 1: intermediate plots for each MCMC chain (2 r.v.
     examples only), 0: no plots

% 2SMART analysis options
analysisopt.num_sim_sdu           = [ 50 50 ];         % Nu number of points ( [ first subset-like
     step, next steps], same values by default )
analysisopt.num_sim_norm          = [ 100 100 ];       % Nn number of points ( [ first subset-like
     step, next steps], same values by default )
nrv = size(probdata.marg,1);
analysisopt.buf_size              = round(1.25e7/nrv); % RAM-size dependent parameter (important for
      large number of random variables)
analysisopt.svm_buf_size          = 3500^2;            % RAM-size dependent parameter for eval_svm.m
      function
analysisopt.flag_var_rbf          = 0;                 % 0: cross validation at the 1st subset-like
     step only, 1: cross validation at all steps


%% GFUNDATA
gfundata(1).evaluator = 'basic';
gfundata(1).type = 'expression';
gfundata(1).parameter = 'no';
gfundata(1).expression = 'limit_state_zeus(x1,x2)';

gfundata(1).flag_sens  = 0;

%% FEMODEL
femodel = 0;
randomfield.mesh = 0;
```

- **limit_state_zeus.m** - This will be the Matlab function that contains the limit state function to be called from ferum_inputfile.m. An example file is shown in Listing 2. This Matlab function should have the capabilities of:
  - Modify the input file *.dat depending on the value of the random variable that FERUM will evaluate. The lines 6 to 29 facilitates the modification of necessary inputs to the ZEUS-NL. For the case of file presented in Listing 2, the random variables considered are the applied and material property modulus of elasticity. There are couple of ways to modify the *.dat file using Matlab functions and this varies depending on the random variable to be considered.
  - Run the *.bat file where the dos commands in running ZEUS-NL are saved.
  - Read *.num results of ZEUS-NL. It is necessary to know what performance indicator (i.e. displacement, rotation, interstory drift,stress, etc.) is being considered in the reliability analysis. The lines 43 to 52 facilitates this reading of a specific displacement of the node. Since the performance indicator may vary from structure to structure, then the user should specify this uniquely.
  - Delete *.num file after the results are acquired.
- **RunZEUS.bat** - This file should include all the dos commands that will run the ZEUS-NL similar to Figure 3.
- **Reader.exe and Solver.exe** - These applications should be under the same folder where Matlab files (ferum_inputfile.m and limit_state_zeus.m) are called. These application files can be found within the installation directory of ZEUS-NL.

Listing 2: Sample Limit State funtion in Matlab.

```
function g = limit_state_zeus(x1,x2)
        for i=1:length(x1)
        P=x1(i);
```

```
E=x2(i);

% Change input file for Zeus
fid = fopen('simple.dat'); % open file
fileContents = textscan(fid,'%s','Delimiter','\n');
fileContents = fileContents{1};
fclose(fid); % close file

% Change modulus of elasticity of material
matStarts = strmatch('#                                                    Material Types',
    fileContents);
tmp = regexp(fileContents{matStarts+5}, '\s+', 'split');
tmp{3} = num2str(E); % column 3 is modulus of elasticity
tmp = strjoin(tmp);
fileContents{matStarts+5} = tmp;

% Change load applied
loadStarts = strmatch('initial.loads',fileContents);
tmp = regexp(fileContents{loadStarts+2}, '\s+', 'split');
tmp{4} = num2str(P); % column 4 is load applied
tmp = strjoin(tmp);
fileContents{loadStarts+2} = tmp;

% write to file
fid = fopen('run.dat','wt') ;
fprintf(fid,'%s\n',fileContents{:});
fclose(fid) ;

% Perform structural analysis
system('RunZEUS.bat');
count = 0;
while exist('result.num','file')==0
 pause();
 count = count+1;
 if count == 100
   disp('no file was generated in allowed time');
   break;
 end
end
% Read result.num
rfile = fopen('result.num'); % open file
resultContents = textscan(rfile,'%s','Delimiter','\n');
resultContents = resultContents{1};
fclose(rfile); % close file

% Determine x - displacement of node 3
dispStarts = strmatch('DISPLACEMENTS',resultContents);
node_ID = 3;
tmp = regexp(resultContents{dispStarts+1+node_ID}, '\s+', 'split');
delta_x = str2num(tmp{2});

system('deleteFILE.bat');

g(i) = 1.5 - delta_x;
end
```

Once all these files are properly prepared in one directory. The reliability analysis can now be performed. The **ferum_inputfile.m** should be loaded first and then the user should issue the `>> ferum` and then choose the analysis type (or predefine the analysis type before hand).

4 Conclusion

This document briefly outlines how to use FERUM and ZEUS-NL together in performing reliability analysis. The technique considered here can be extended to other structural analysis software as long as Matlab is able to communicate with the structural analysis software.

References

[1] FERUM Version 3.0 User's Guide, 2001.

[2] J Bourinet. FERUM 4.1 User's Guide, 2010.

[3] Elnashai, Amr S., Vassilis K. Papanikolaou, and Do H. Lee. *ZEUS-NL: A System for Inelastic Analysis of Strutures User's Manual*. Number January. 2012.

[4] Young-joo Lee and Do-soo Moon. A new methodology of the development of seismic fragility curves. *Smart Structures and Systems*, 14(5):847–867, 2014.

[5] Robert E Melchers and André T Beck. *Structural Reliability Analysis and Prediction Third Edition*. John Wiley & Sons, Hoboke, New Jersey, third edition, 2018.

[6] Andrzej S. Nowak and Kevin R. Collins. *Reliability of Structures*. McGraw-Hill Education, United States of America, 2000.