Recent development in Ansys LS-DYNA's NVH solvers

Yun Huang, Tom Littlewood, Zhe Cui, Ushnish Basu
Ansys Inc.

Abstract

LS-DYNA has been used in automotive industries for many years, especially in vehicle crashworthiness and occupant safety analysis areas. Besides that, LS-DYNA also provides many useful features for NVH (Noise, Vibration and Harshness) analysis. During the past few years, based on the feedback and suggestions from users, many updates and enhancements in the NVH solvers have been implemented, including

- Fast FRF analysis with reduced eigenvectors.
- Frequency dependent adaptive remeshing for BEM acoustics.
- Frequency interpolation for BEM acoustic solvers.
- Fluid added mass computation and its application in modal and vibration analysis.
- Coupling of acoustic spectral element method and piezoelectric materials for ultrasonic sensor simulation.
- Enhanced d3max output.
- New options in fatigue solvers.
- Other enhancements.

This paper aims at introducing some of these updates and enhancements to LS-DYNA users. Some examples are included in the paper for illustration and validation purposes.

1 Introduction

LS-DYNA has been widely used in automotive industries for many different applications, including crashworthiness, passenger safety evaluation, NVH and durability analysis. Particularly, new features are consistently added to improve its strength and capabilities in NVH analysis, based on the feedback and suggestions from users. The updates and enhancements have been made in different areas in vibration analysis, acoustic analysis and fatigue analysis. This paper aims to give a brief introduction on some of the recent updates and enhancements in LS-DYNA's NVH solvers.

2 Fast FRF analysis with reduced eigenvectors

FRF (keyword: **`*FREQUENCY_DOMAIN_FRF`**) provides a transfer function from loading to response, and it has important applications in auto NVH problems. For example, it can be used to characterize some important properties of BIW like the dynamic stiffness and effective mass, etc. It can also be used to identify the energy transfer path for a vibration problem or a vibro-acoustic problem.

FRF is computed using modal superposition. Thus, extracting modal shape (or eigenvectors) from previous modal analysis is the first step. In the past, LS-DYNA always extracted the full eigenvectors (including all the nodes in the model) from d3eigv and used them in modal superposition. However, for many FRF problems, the number of nodes (or elements) involved in loading and in response are very limited. For example, for vibration analysis of vehicles, people are more concerned about the excitation from engine attachment points, suspensions or wheels, and the response only on seats, steering wheels, etc. In other words, the nonzero components in load vectors, and in response vectors are very less, compared to the total dof of the original model. For the loading cases like nodal force and pressure, once we get modal shape (or eigenvectors) on these nodes, we can run FRF analysis.

Recently a reduced eigenvector output has been implemented for fast Lanczos eigen solver (**`eigmth`** = 103). The reduced eigenvectors are provided for selected nodes only (the set of nodes for eigenvector output is defined in card 3 in **`*CONTROL_IMPLICIT_EIGENVALUE`**). They are dumped in a lsda binary file "FastLnzEigenVectors". Using this binary file, we can now run FRF with the reduced eigenvectors.

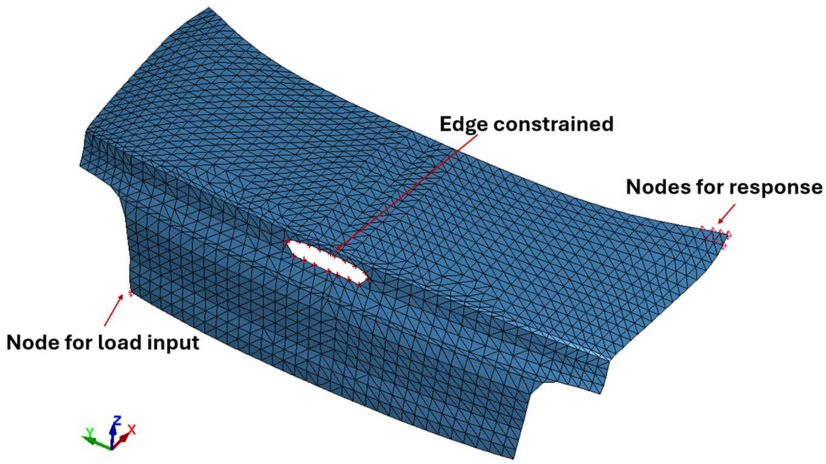

Fig.1: A trunk lid model for FRF analysis

For illustration purposes, FRF analysis for a simplified trunk lid model (see Figure 1) is performed, using both the full eigenvectors from d3eigv and the reduced eigenvectors from “FastLnzEigenVectors”. The model has 2695 shell elements and is constrained to a static shaker table through the hole in the center. Harmonic nodal force excitation is defined on a corner of the model and the displacement response on the other corner is computed. The FRF (the ratio of displacement over nodal force, or “dynamic compliance”) results by the two methods are given in Figure 2. The two FRF curves basically overlap with each other.

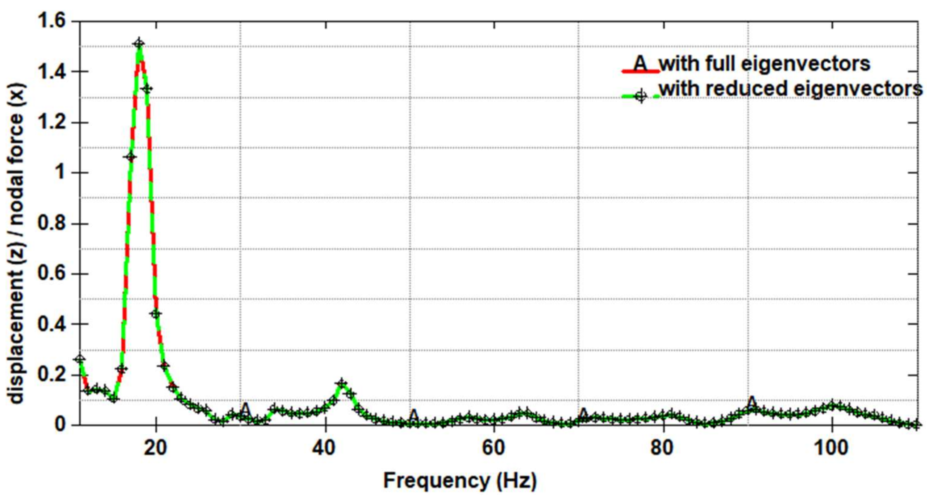

Fig.2: FRF results using full or reduced eigenvectors

Here are the Keyword cards for running FRF with reduced eigenvectors.

```
*CONTROL_IMPLICIT_EIGENVALUE
$:    neig    center     lflag    lftend     rflag    rhtend    eigmth    shfscl
        20       0.0         0       0.0         0       0.0       103       0.0
$:  isolid     ibeam    ishell   itshell    mstres    evdump   mstrscl
         0         0         0         0         0         0       0.0
$ Card 3c
                             1
*FREQUENCY_DOMAIN_FRF_REDUCED
$:      n1     n1typ      dof1      vad1    relatv     fnmax     mdmin     mdmax
    428297         0         1         3
$:   dampf     lcdam     lctyp    dmpmas    dmpstf    dmpflg
      0.02         0         0       0.0       0.0         0
$:      n2     n2typ      dof2      vad2      vid2
         3         1         3         2
$:    fmin      fmax     nfreq    fspace    lcfreq
      11.0     110.0       100         0         0
```

Fig.3: FRF keyword card for running with reduced eigenvectors

In terms of hard drive space usage and CPU time, the one based on reduced eigenvectors is more efficient than the traditional one which is based on full d3eigv files, as can be seen from Table 1 and Table 2.

File name for eigenvectors	Size (bytes)
d3eigv	1,552,384
FastLnzEigenVectors	12,301

Table 1: Disk consumption comparison

Approach	CPU time (seconds)
FRF with full eigenvectors (from d3eigv)	0.758
FRF with reduced eigenvectors (from “FastLnzEigenVectors”)	0.656

Table 2: CPU time cost comparison

The difference in CPU time could be larger if a more complicated model is used and thousands of eigenmodes are considered in the modal superposition. For the current model, only 20 eigenmodes are considered.

3 Frequency dependent adaptive remeshing for BEM Acoustic solver

In BEM acoustics, for accuracy consideration, the element size should not exceed 1/8 of the wavelength (see [1]), where the wavelength λ is defined by equation (1):

$$\lambda = c/f \tag{1}$$

In equation (1), c is the sound speed. It is a constant number based on the medium (e.g. c=340 m/s in air). f is the frequency.

In LS-DYNA, we used to use one mesh for the computation of the whole range of frequencies. For the mesh to be good even for the highest frequency, it must be dense enough. In other words, the element size is controlled or decided by the highest frequency, and it must be less than 1/8 of the wavelength at the highest frequency. This could result in significant waste of CPU time, since for lower frequencies, it is ok to use a coarser mesh and get the solution faster.

With the latest version of LS-DYNA, a frequency dependent adaptive remeshing for the acoustic BEM has been implemented. The idea is to start BEM computation from a coarser mesh, at lower frequencies. And then the boundary element mesh is refined with increasing frequencies when needed (the element size becomes larger than 1/8 of the wavelength). In that case, the original boundary element is divided into smaller ones, as shown in Figure 4.

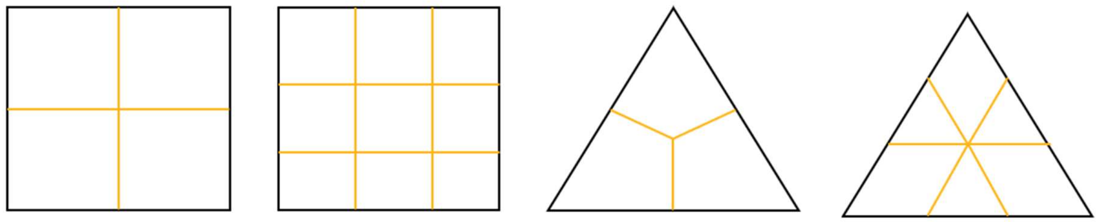

Fig.4: Boundary element mesh refinement

The model concerned is a simplified vehicle compartment as shown in Figure 5. It has 1,264 boundary elements in the original configuration.

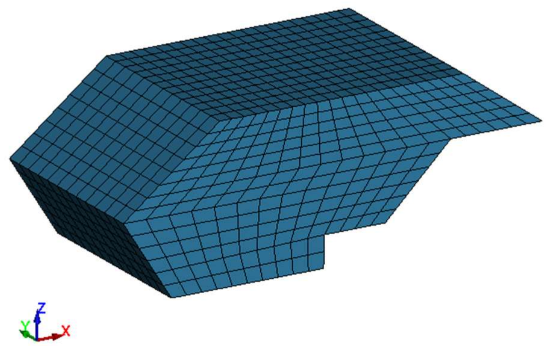

Fig.5: A compartment model for BEM acoustic analysis

When a vehicle runs, the compartment is subjected to vibration from the ground. To model this, a uniform normal velocity (7 mm/s) boundary condition is assumed on the **`set_segment`**, defined on the bottom, as shown in Figure 6. The rest of the surface of the compartment is assumed to be rigid (in other words, $v_n = 0$, where v_n is the normal velocity).

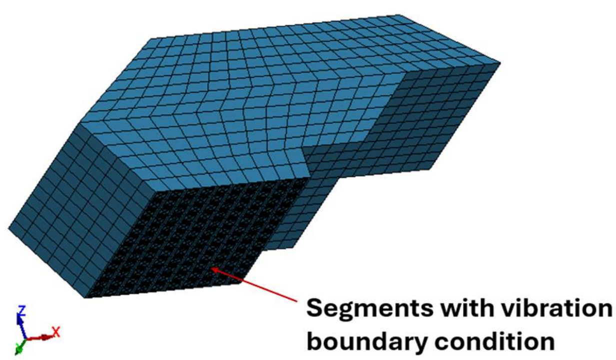

Fig.6: Vibration condition on the bottom

For this problem, we are looking for solution at 10 frequencies in the range 100-2000 Hz, as defined by the keyword card in Figure 7. The solution is requested for a field point inside the compartment, defined by **`set_node`** 200.

```
*FREQUENCY_DOMAIN_ACOUSTIC_BEM
$#      ro         c      fmin      fmax     nfreq    dt_out   t_start      pref
1.2300e-12 3.4000e+5  100.0000   2000.00        10         0    0.0002.0000e-11
$#nsid_ext  type_ext  nsid_int  type_int   fft_win
         0         0       200         1         4
$#  method     maxit       res       ndd
       312      1000 1.0000E-6
$#                nbc    restrt     iedge      noel     nfrup
                    2
$#    ssid    sstype      norm  bem_type       lc1       lc2
         1         2         0        -3
$#    ssid    sstype      norm  bem_type       lc1       lc2
         2         2         0        -2
```

Fig.7: Keyword for running BEM acoustic solver

The boundary element mesh is refined if the maximum element size becomes larger than 1/8 of the wave length for the current frequency. Table 3 shows the frequency, 1/8 of wave length, and the number of division needed on the side of each element (for dividing one element into smaller ones). For the original mesh, the max element size is 61.63 mm and it is good for the first 3 frequencies.

Index	Frequency (Hz)	1/8 of wave length (mm)	Element side division
1	100.0	425.00	NA
2	311.1	136.61	NA
3	522.2	81.38	NA
4	733.3	57.95	2
5	944.4	45.00	2
6	1155.6	36.78	2
7	1366.7	31.10	2
8	1577.8	26.94	3
9	1788.9	23.76	3
10	2000.0	21.25	3

Table 3: Frequency dependent mesh refinement

As Table 3 indicated, two mesh refinements are needed. One is needed at frequency 733.3 Hz and the other is needed at frequency 1577.8 Hz. The two refined meshes are shown in Figures 8 and 9.

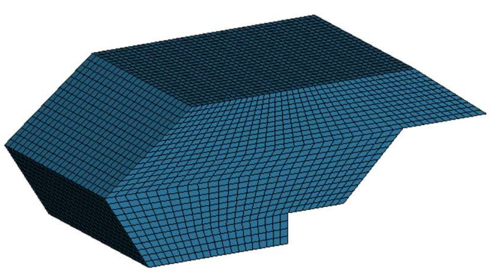

Fig.8: BEM mesh after the 1st refinement (5,056 element)

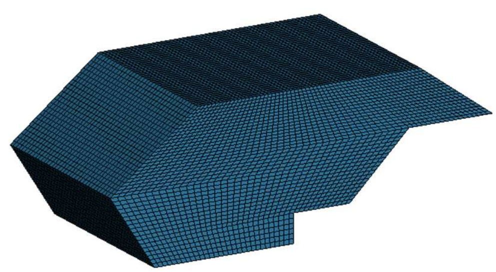

Fig.9: BEM mesh after the 2nd refinement (11,376 element)

To check the results, the same problem was run with a constant mesh (the BEM mesh with 11,376 elements, which is the same one after the 2nd mesh refinement) for the whole range of frequency. The acoustic results computed by the two approaches are given in Figure 10.

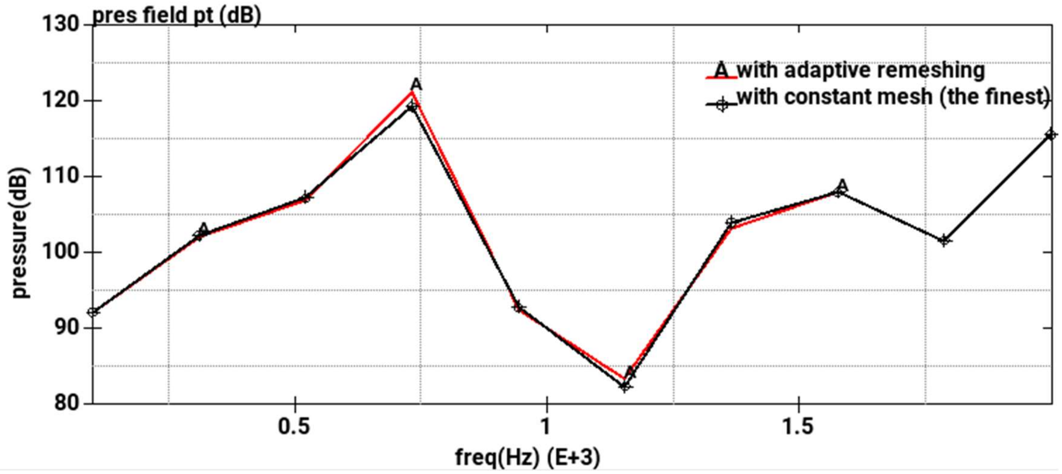

Fig.10: Acoustic results by the two approaches (adaptive mesh and constant mesh)

One can see that the two results are very close to each other.

But in terms of CPU time, the adaptive remeshing one is obviously faster.

Mesh option	**CPU time**
With constant mesh	28 minutes 20 seconds
With adaptive remeshing	12 minutes 6 seconds

Table 4: CPU time for the acoustic problem (1 thread SMP run with Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz)

4 Frequency interpolation for BEM acoustic solvers

In the past, LS-DYNA BEM acoustic solvers always run the computation on the frequencies given from FFT (FFT is used to convert the time domain vibration data to frequency domain) or the frequencies defined in the frequency domain vibration analysis directly. After that, a frequency interpolation is performed in post-processing, to get the acoustic results at the frequencies requested by users. This could be inefficient since the number of the frequencies requested by users may be much less than the number of frequencies from FFT, or from the frequency domain vibration analysis. Thus, a new option is implemented to run the frequency interpolation on boundary condition first, before jumping into the BEM equation systems formulation and solution.

With the new keyword **`*FREQUENCY_DOMAIN_ACOUSTIC_FREQUENCY`**, a variety of ways to define the output frequencies and the interpolation option have been provided.

Card 1	1	2	3	4	5	6	7	8
Variable	FMIN	FMAX	NFREQ	FSPACE	LCFREQ	BIAS	SPREADF	FRACTN
Type	F	F	I	I	I	F	F	I
Default	0.0	0.0	0	0	0	3.0	0.1	0

*Fig.11: New keyword **`*FREQUENCY_DOMAIN_ACOUSTIC_FREQUENCY`***

More details of the keyword parameters are given below.

VARIABLE	DESCRIPTION
FMIN	Minimum frequency for output (cycles/time)
FMAX	Maximum frequency for output (cycles/time)
NFREQ	Total number of frequencies for output

	>0: number of frequencies for the whole range. <0: number of frequencies for each interval.
FSPACE	Frequency spacing option for output: EQ.0: Linear EQ.1: Logarithmic EQ.2: Biased spacing (range) EQ.3: Eigenfrequencies only EQ.4: Biased spacing (eigenfrequency) EQ.5: Biased spacing (eigenfrequency spread) EQ.6: Octave frequencies starting with FMIN
LCFREQ	Load curve ID defining the frequencies for output
BIAS	Bias parameter
SPREADF	Spread ratio
FRACTN	Octave fraction (for **FSPACE**=6). For example, **FRACTN**=3 means 1/3 octave spacing. **FMAX** is ignored.

Particularly the different spacing options are given as Figure 12.

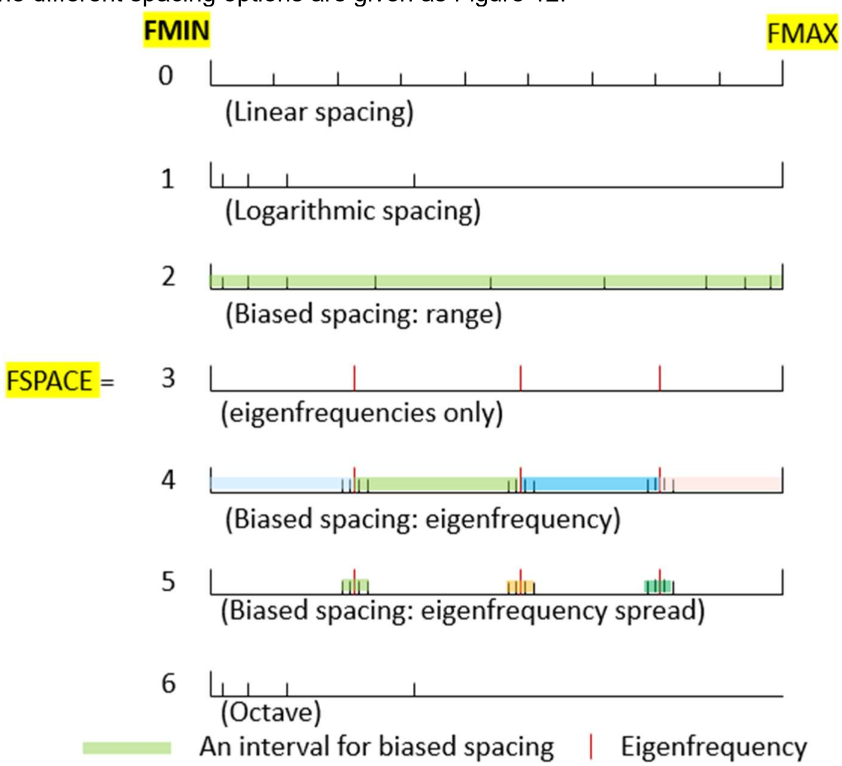

Fig.12: Different spacing options for output frequencies

5 Enhanced d3max database

D3max is a quick and convenient tool to provide the envelope of stress during a transient process. This is useful if the users are only concerned about the maximum value of stress in the process, instead of the lengthy time history of the stress. For example, for users running drop tests of electronic products, the maximum value of the stress shown up in this process is more important and more interesting to them, and sometimes that is the physical variable which is used in the safety evaluation of the parts.

For d3max, some new options have been implemented to improve the capabilities of this feature. They include:

- Supporting thick shell elements
- Supporting particle elements
- Supporting small restart and simple restart
- Supporting ALE results

6 New options in fatigue solvers

For fatigue solvers, a new option `_STATIC` was implemented to allow users to run transient fatigue analysis with a set of linear static stress results, and the time history curve for the loading. This option is useful if the structure is undergoing transient linear vibration under proportional loading (or the loading direction is not changed), and the stress level is low.

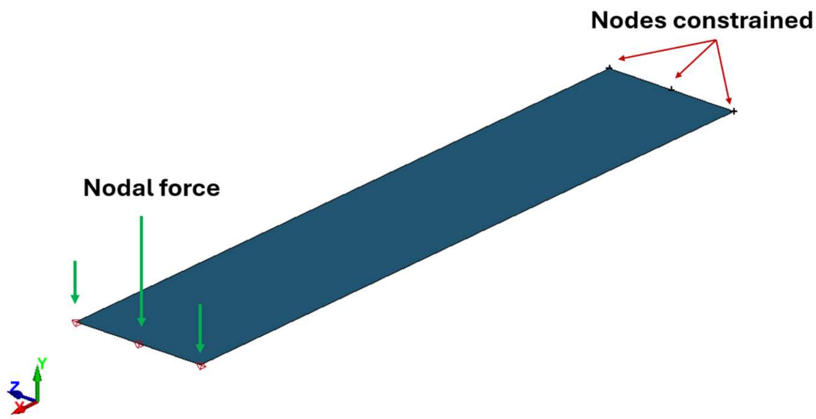

Fig.13: A rectangular beam under time varying nodal force on the edge

The problem considered here is a rectangular beam, under time varying (cyclic) nodal force on one edge. The beam is constrained on the other edge.

Two approaches were used to run fatigue analysis for this model. The first one, which is based on extracting stress cycle from full d3plot binary database, uses keyword setting like this

```
*CONTROL_IMPLICIT_GENERAL
$#  imflag       dt0    imform      nsbs       igs     cnstn      form    zero_v
         1   0.00050
*CONTROL_IMPLICIT_SOLUTION
$#  nsolvr    ilimit    maxref     dctol     ectol     rctol     lstol    abstol
         1         0         0     0.000     0.000     0.000     0.000     0.000
$#   dnorm    diverg     istif   nlprint    nlnorm   d3itctl
         0         0     99999
*CONTROL_IMPLICIT_DYNAMICS
$#   imass     gamma      beta    tdybir    tdydth    tdybur     irate     alpha
         0      0.60      0.38       0.0       0.5      1.00         1
*CONTROL_TERMINATION
$#  endtim    endcyc     dtmin    endeng    endmas
  1.000000         0     0.000     0.000     0.000
*DATABASE_BINARY_D3PLOT
$#      dt      lcdt      beam     npltc    psetid
   0.01000         0         0         0         0
$#   ioopt
         0
*LOAD_NODE_POINT
$#     nid       dof      lcid        sf       cid        m1        m2        m3
        16         2        98  -0.00500         0         0         0         0
        32         2        98  -0.01000         0         0         0         0
        48         2        98  -0.00500         0         0         0         0
*FATIGUE_D3PLOT
$#    ssid    sstype

$#      dt

$#   stres     index
         0         0
*MAT_ADD_FATIGUE
$#     mid      lcid     ltype         a         b    sthres    snlimt
         1       400
*DATABASE_FREQUENCY_BINARY_D3FTG
         1
```

Fig.14: Keyword setting for running fatigue analysis using ***FATIGUE_D3PLOT***

The keyword indicates that a transient analysis is performed with load history curve 98, and the stress results are dumped to d3plot binary database every 0.01 seconds. After that, fatigue analysis is performed using von Mises stress time history (obtained from the stress component history saved in d3plot). Please note that curve 98 is depicted in Figure 15.

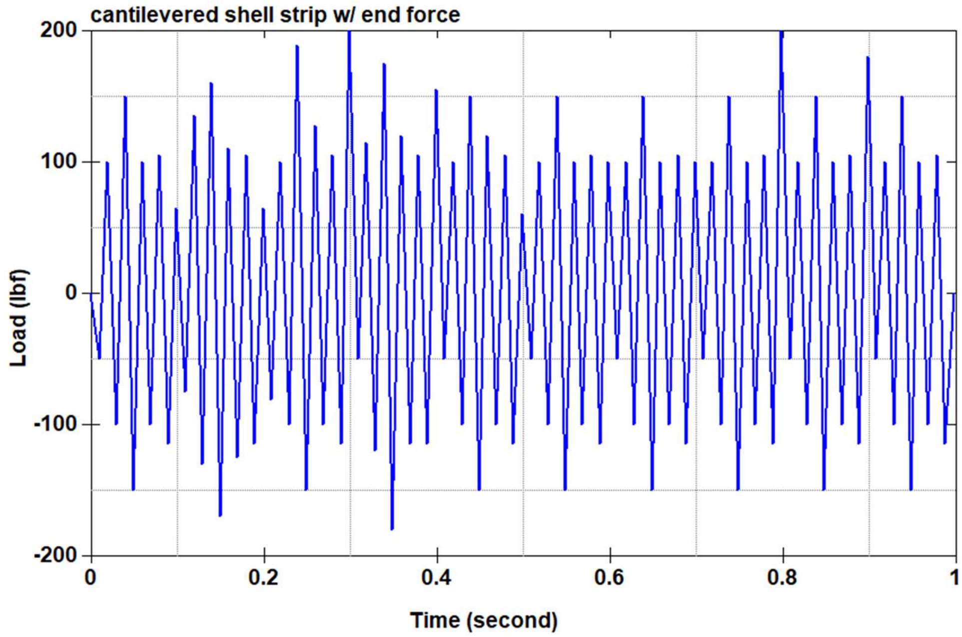

Fig.15: Load history curve (curve ID 98)

For the second approach, the full time history of stress for each element is reconstructed using a static stress solution and the time history curve of the load scale. It is assumed that the structural response is linear, thus the whole stress history can be reconstructed using a static stress result and the scale curve of the load. The static stress solution is obtained by running a linear static analysis, using a step nodal force to the structure, shown in Figure 16.

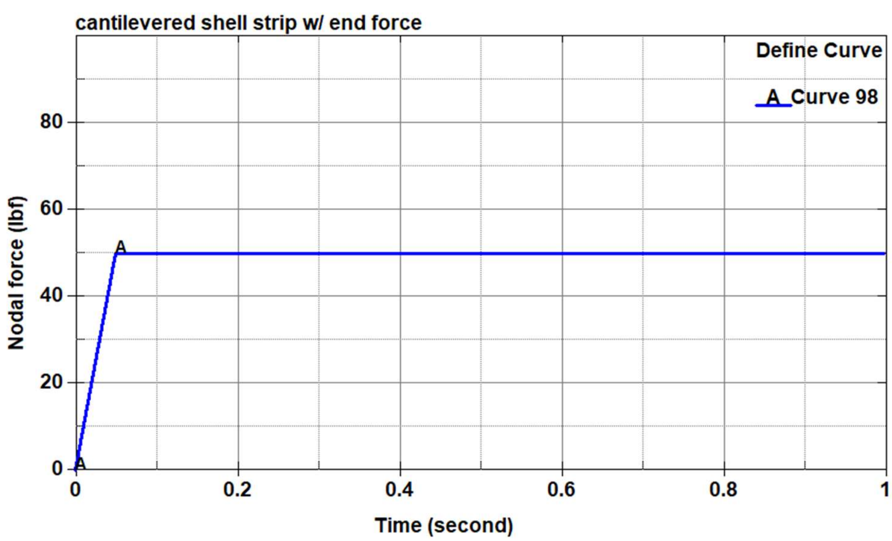

Fig.16: A step force for linear static computation

The static stress results are saved in a d3plot database, under the folder “static.stress”.

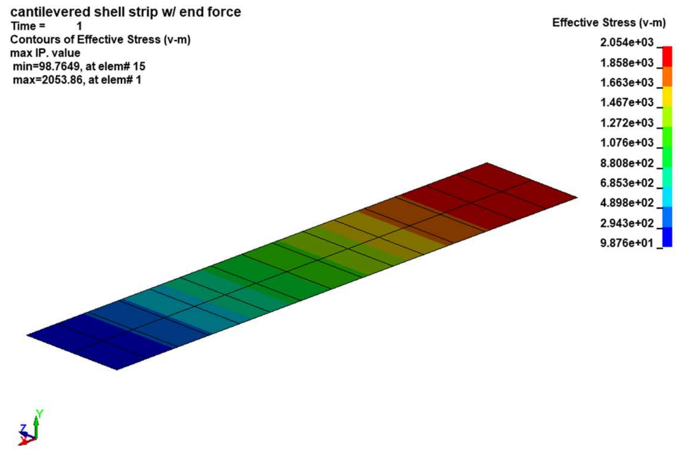

Fig.17: Static stress state

The new keyword for this approach is ***FATIGUE_STATIC**. in Figure 18. "dt" defined in card 2 corresponds to "dt" defined in ***DATABASE_BINARY_D3PLOT**, and it defines the time step for reconstructing the stress history curve. The filename of the file which saves the static stress, is defined in card 3, like "static.stress/d3plot". In card 4, the time history curve of the load scale factor is defined by curve ID 1. "**nstate**=2" means LS-DYNA needs to extract the static stress data from the 2nd state of d3plot database.

```
*FATIGUE_STATIC
$#    ssid    sstype

$#      dt
  0.01E+00
$#   stres     index

$ filename
 static.stress/d3plot
$#    lcid    nstate
         1         2
*MAT_ADD_FATIGUE
$#     mid      lcid     ltype         a         b    sthres    snlimt
         1       400
*DATABASE_FREQUENCY_BINARY_D3FTG
         1
```

*Fig.18: Keyword setting for running fatigue analysis using ***FATIGUE_STATIC***

Particularly, the load scale time history is given as Figure 19.

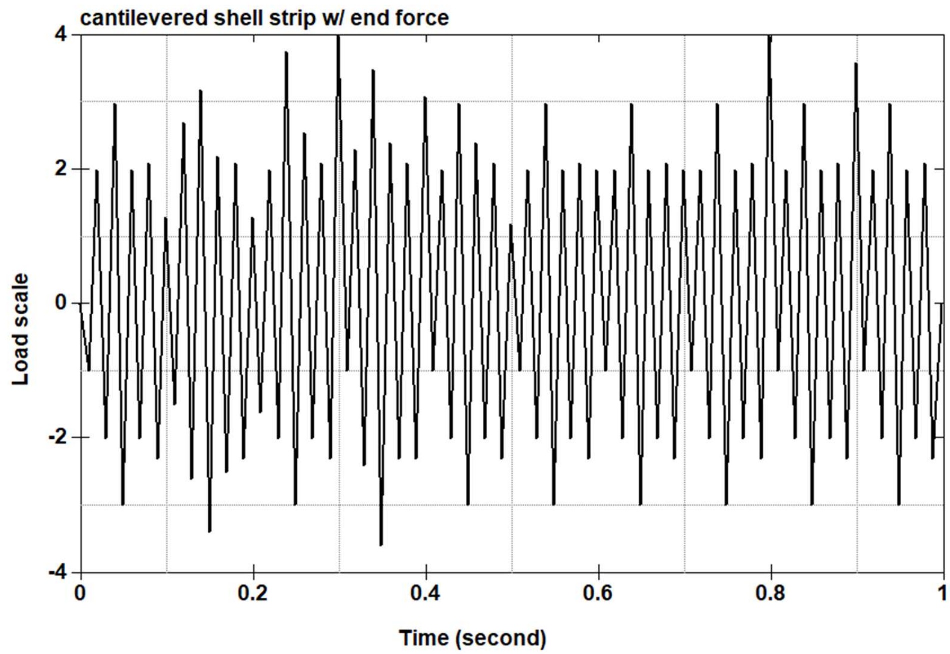

Fig.19: Load scale time history curve

The results of fatigue damage ratio (in d3ftg) are compared in the following figures.

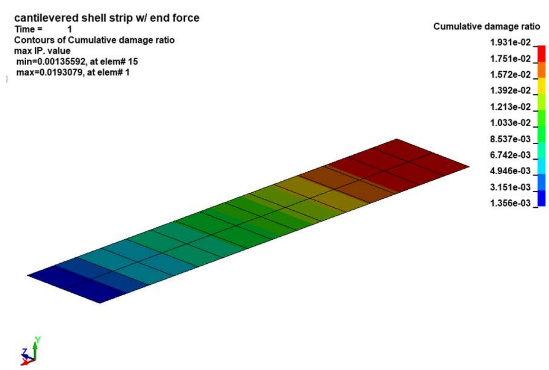

Fig.20: Fatigue damage ratio from ****FATIGUE_D3PLOT***

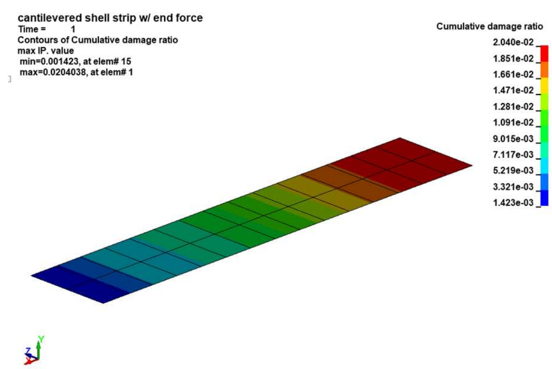

Fig.21: Fatigue damage ratio from ****FATIGUE_STATIC***

As can be seen, with the full time domain fatigue approach (**`*FATIGUE_D3PLOT`**), the maximum value of the fatigue damage ratio is 0.0193, and with the static approach (**`*FATIGUE_STATIC`**), the maximum value of the fatigue damage ratio is 0.0204. The percentage difference is only around 5.7%. The maximum damage ratio takes place at the same element (element 1) for the two approaches. In addition, the distribution of the fatigue damage ratio is very close for the two approaches.

In terms of Hard drive space consumption, as shown in Table 5, the new approach based on **`*FATIGUE_STATIC`** is more efficient since it needs to save only 1 static stress state results to d3plot. The approach based on **`*FATIGUE_D3PLOT`** needs to save d3plot for multiple time steps and the d3plot family files (d3plot, d3plot01, d3plot02, etc.) could easily take hundreds of GB for a large-scale model, for thousands of time steps.

In terms of CPU cost, the new approach based on **`*FATIGUE_STATIC`** is also cheaper, since it needs to run only one step static calculation to get one set of static stress data. For the example shown in this paper (Figure 13), the total CPU time for **`*FATIGUE_STATIC`** approach (combing the CPU time for running a one step static computation, and the CPU time for running fatigue analysis), is still much less than the CPU time for using **`*FATIGUE_D3PLOT`** approach.

Approach	Hard drive space (byte)
`*FATIGUE_D3PLOT`	32,768
`*FATIGUE_STATIC`	1,196,032

Table 5: Hard drive space by the two approaches

Approach	CPU time (seconds)
`*FATIGUE_D3PLOT`	122.69
`*FATIGUE_STATIC`	1.0887[1]+1.1885[2]=2.2772

*Table 6: CPU time by the two approaches (for **`*FATIGUE_STATIC`**: 1- CPU time for static stress computation; 2- CPU time for fatigue computation)*

Please note, the CPU time in Table 6 is based on a 4 threads SMP run with Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz.

Of course, the **`*FATIGUE_STATIC`** approach is only valid if the load is small, and the structural response is still in the linear range. As a contrary, though it is more expensive, **`*FATIGUE_D3PLOT`** (and **`*FATIGUE_ELOUT`**) is more flexible and can work with different problems with dynamic and plastic behaviors.

7 Summary

This paper provides a brief introduction on some of the new developments and enhancements for LS-DYNA NVH solvers, from the past few years. Some examples are included for illustration and validation purposes. The new developments and enhancements were implemented to meet the requirement from many users, for using LS-DYNA more effectively on NVH analysis.

Feedback and suggestions from the users, especially for the continued development of the NVH solvers, are highly welcomed and appreciated.

8 Literature

[1] LS-DYNA® Keyword User's Manual, Volume I, 2024