Enabling the Network for Earthquake Engineering Simulation

TR-2009-[ID]

OpenFresco Framework for Hybrid Simulation: LS-DYNA Example

Andreas Schellenberg, Yuli Huang, Hong K. Kim,
Yoshikazu Takahashi, Gregory L. Fenves,
and Stephen A. Mahin

Department of Civil and Environmental Engineering,
University of California, Berkeley

Last Modified: 2009-08-03 Version: 2.6

Acknowledgment: This work was supported by the George E. Brown, Jr. Network for Earthquake Engineering Simulation (NEES) Program of the National Science Foundation under Award Number CMS-0402490. Visit http://it.nees.org/ for more information.

Table of Contents

Table of Figures

1 Introduction: Distributed Hybrid Simulation Example Using LS-DYNA

This document shows how LS-DYNA (LSTC 2006) can be used as the computational driver for a hybrid simulation with OpenFresco. The demonstration example is a simple two-DOF model that utilizes the LS-DYNA explicit solver, which is a modified central difference time integrator (Hallquist 2006), to perform the time history analysis. A distributed hybrid simulation is performed in this example. The test is a fully simulated test, meaning that the experimental control is set to simulation mode. Thus, it does not require a physical specimen or setup to run. The response results from the simulation are provided for comparison. In this example, LS-DYNA v971 was used.

The document also provides a guideline for programming a user-defined element in LS-DYNA.

2 Required Files

For this example, the following files are necessary. These are located in:

`User’s Directory\OpenFresco\trunk\EXAMPLES\OneBayFrame\LSDyna`

if OpenFresco was installed in the default location, the `User’s Directory` is `C:\Program Files`.

The following files should be in this directory:

- `OneBayFrame_LSDyna.k`

Some Tcl files are needed in addition to the files above. These are in:

`User’s Directory\OpenFresco\trunk\EXAMPLES\OneBayFrame\OpenSees`

The following files should be in this directory:

- `OneBayFrame_Local_SimAppServer.tcl`
- `OneBayFrame_Distr_LabServer.tcl`
- `OneBayFrame_Distr_SimAppServer.tcl`

3 Structural Model

The structural model is defined to be equivalent to the OpenSees One-Bay Frame model found in the OpenFresco Installation and Getting Started Guide (Figure 5.1 of that report). The model consists of a four-node shell element, Element 1, which is the user-defined element representing the experimental column. It also has two beam elements, Elements 2 and 3, as illustrated in Figure 1. Lumped masses are placed at nodes 3 and 4 as in the OpenSees model. In LS-DYNA, a user-defined element must be a shell element. The shell element can be made to behave like a beam element by restraining certain degrees-of-freedom of the nodes.

The following LS-DYNA input file from `OneBayFrame_LSDyna.k` defines the geometry of the model:

```
*ELEMENT_SHELL
$#   eid      pid       n1       n2       n3       n4       n5       n6       n7       n8
       1        1        1        3        5        6
```

```
*ELEMENT_DISCRETE
$#   eid      pid       n1       n2      vid                s       pf          offset
       2        2        2        4        1
       3        3        3        4        1
*ELEMENT_MASS
$#   eid      nid             mass      pid
      13        3        0.0400000        1
      14        4        0.0200000
*NODE
$#   nid                x                y                z       tc       rc
       1            0.000            0.000            0.000        7        7
       2      100.0000000            0.000            0.000        7        7
       3            0.000       54.0000000            0.000        5        7
       4      100.0000000       54.0000000            0.000        5        7
       5 -1.0000000e-006       54.0000000            0.000        7        7
       6 -1.0000000e-006            0.000            0.000        7        7
```

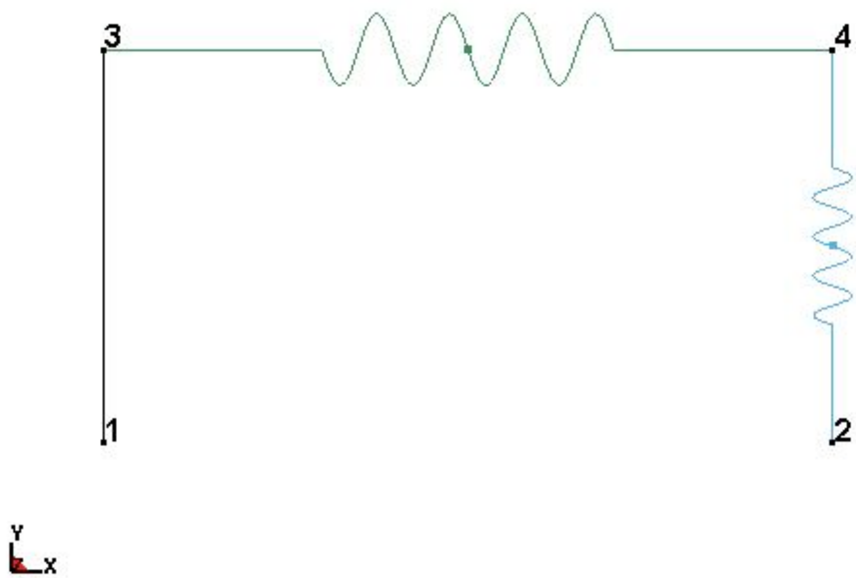

Figure 1: LS-DYNA One-Bay Frame Model.

4 Ground Motion

The structure is subjected horizontally to the north-south component of the ground motion recorded at a site in El Centro, California during the Imperial Valley earthquake of May 18, 1940 (Chopra 2006). The file, elcentro.txt, contains the acceleration data recorded at every 0.02 seconds (Figure 2).

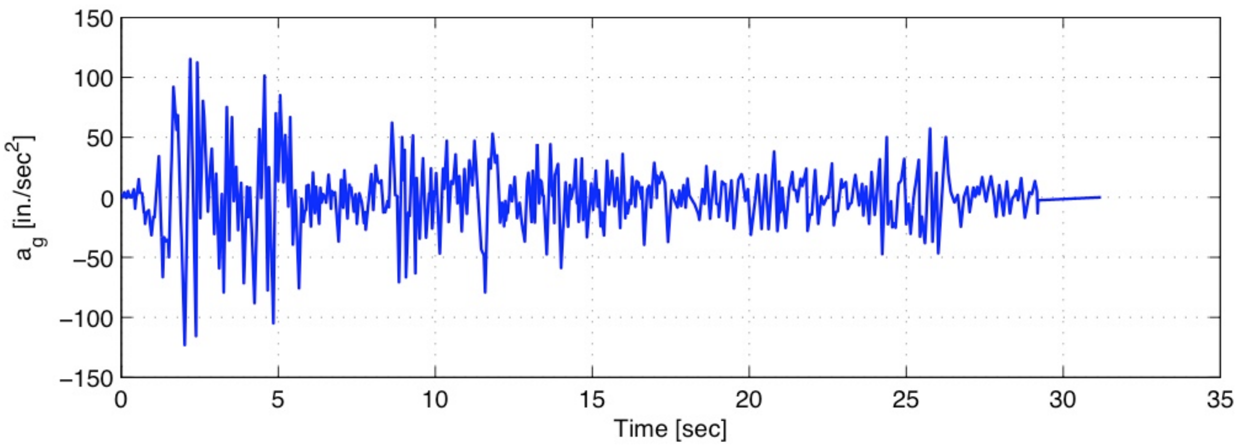

Figure 2: 1940 El Centro Ground Motion.

5 LS-DYNA User-Defined Element

5.1 Interfacing LS-DYNA with OpenFresco via User-Defined Element

LS-DYNA provides a user-defined element interface (LSTC 2006). The interface can accommodate either an integrated or a resultant/discrete element. For resultant/discrete element formulations, the force and stiffness assembly must be implemented. History variables can be associated with the user-defined elements. Each time nodal displacements are passed to the element it returns the nodal forces and consistent stiffness (Figure 3).

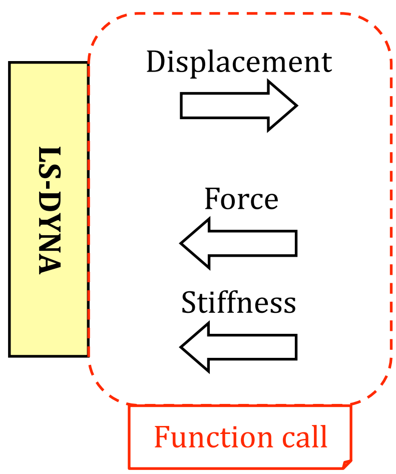

Figure 3: LS-DYNA User-Defined (Generic-Client) Element.

In OpenFresco, the SimAppElemServer (Simulation Application Element Server) is used as an interface to the generic client element in LS-DYNA. This middle-tier server can accommodate network communications from the user-defined generic client element in LS-DYNA to the OpenFresco experimental elements. After displacements are sent to the middle-tier server, the corresponding forces and stiffness can be obtained, as illustrated in Figure 4.

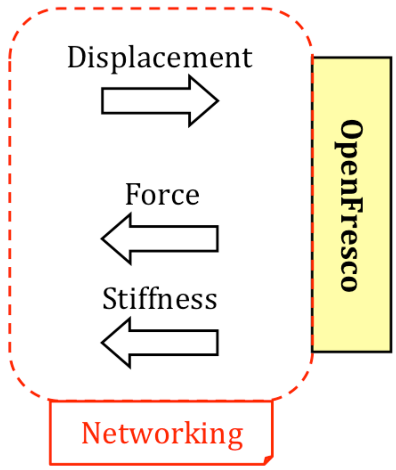

Figure 4: OpenFresco SimAppElemServer.

Initialization and termination phases are added to open and close the socket, required for the network communications. The network ports and other flags are stored as history variables in the LS-DYNA user-defined element. To run the example and communicate with OpenFresco, the user-defined element and socket interface codes are compiled and linked with the LS-DYNA library (Figure 5).

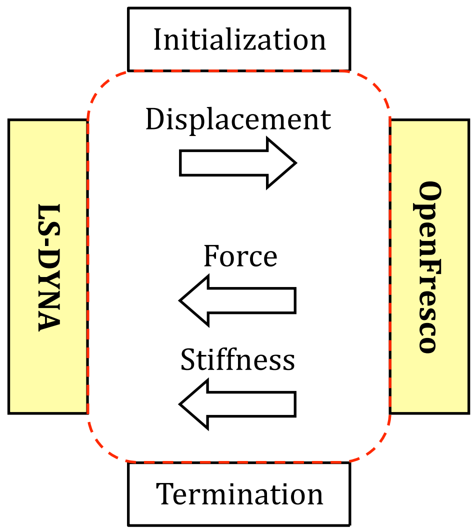

Figure 5: Interface between LS-DYNA and OpenFresco.

5.2 Implementation of User-Defined Element genericClient_1d

The Fortran source code for the user-defined element is in the file `genericClient_1d.f`. The following portion of the LS-DYNA input file `OneBayFrame_LSDyna.k` defines the left column of the model, shown in Figure 1.1, using a shell section. The shell section utilizes element `genericClient_1d` with a port number of `8090`.

```
*PART
$# title
leftcolumn
$#     pid     secid       mid     eosid      hgid      grav    adpopt      tmid
         1         1         1
*SECTION_SHELL
$#   secid    elform      shrf       nip     propt   qr/irid     icomp     setyp
         1       101
$#      t1        t2        t3        t4      nloc     marea      idof    edgset
  1.000000  1.000000  1.000000  1.000000
$#    nipp     nxdof      iunf      ihgf      itaj       lmc      nhsv      iloc
         0                             0                   1         6         1
      8090
*MAT_ELASTIC
 $#     mid        ro         e        pr        da        db  not used
          1 1.0000E-51.0000E-15
```

A user-defined element should have mechanisms for establishing a network connection, sending data, and receiving data, regardless of which finite element (FE) software is used. The following excerpts are from `genericClient_1d.f` for LS-DYNA. This file can be found in:

`User's Directory\OpenFresco\trunk\SRC\simApplicatonClient\lsDyna`

The element `genericClient_1d` uses the Fortran interface file `socketf.c`, which is located in:

`User's Directory\OpenFresco\trunk\SRC\simApplicatonClient\fortran`

The interface file calls functions from `socket.c`, which is located in:

`User's Directory\OpenFresco\trunk\SRC\simApplicatonClient\c`

A network connection is established with the OpenFresco middle-tier server, and the data size is set using the following code.

```
if (nint(hsv(i,1)) .eq. 0) then
            port=nint(cm(1))
            sizeMachineInet = 9+1
            call setupconnectionclient (port,
     *                                 '127.0.0.1'//char(0),
     *                                 sizeMachineInet,
     *                                 socketID)
            if (socketID .le. 0) then
               write(*,*) 'Warning: cannot connect'
            else
               hsv(i,1)=1
               hsv(i,2)=socketID
c
c ...          set the data size for the experimental site
c
c ...          sizeCtrl
c              disp
               iData(1)  = sizeDOF
c              vel
               iData(2)  = 0
c              accel
               iData(3)  = 0
```

```
c               force
                iData(4)  = 0
c               time
                iData(5)  = 0
c ...           sizeDaq
c               disp
                iData(6)  = 0
c               vel
                iData(7)  = 0
c               accel
                iData(8)  = 0
c               force
                iData(9)  = sizeDOF
c               time
                iData(10) = 0
c ...           dataSize
                iData(11) = sizeSendData
c
                call senddata(socketID, sizeInt, iData, 11, stat)
             end if
```

The trial displacements are sent using the code below. Setting the action `sData(1) = 3` informs the middle-tier server that a trial response is being sent.

```
c ...     send trial response to experimental site
c
          sData(1) = 3
          do j = 1, sizeDOF
             sData(1+j) = hsv(i,2+j)
          end do
c
          call senddata(socketID, sizeDouble, sData, sizeSendData, stat)
```

Then the measured force is obtained from the middle-tier server (action `sData(1) = 10`) and is stored in the variable `force`.

```
c ...     get measured resisting forces
c
          sData(1) = 10
          call senddata(socketID, sizeDouble, sData, sizeSendData, stat)
c
          call recvdata(socketID, sizeDouble, rData, sizeSendData, stat)
c
          do j=1,2
             force(i,j)   = rData(j)
             force(i,j+6) = rData(j+2)
          end do
```

Other actions can be communicated to the server by setting `sData(1)` to some integer value. The following is a complete list of all the possible actions.

`sData(1)` Inputs:

- `1` = start server process (optional)
- `2` = setup test (not used here)
- `3` = set trial response
- `4` = execute (obsolete)
- `5` = commit state
- `6` = get DAQ response vectors
- `7` = get displacement vector
- `8` = get velocity vector
- `9` = get acceleration vector
- `10` = get force vector
- `11` = get time vector
- `12` = get initial stiffness matrix
- `13` = get tangent stiffness matrix
- `14` = get damping matrix
- `15` = get mass matrix
- `99` = end server process

6 OpenFresco Tcl Commands

This section contains explanations of the common OpenFresco Tcl commands used in both local and distributed tests. Each subsection highlights a Tcl command and the script that contains the command.

6.1 Experimental Control

The experimental control is set to `SimUniaxialMaterials` for this example. `SimUniaxialMaterials` uses the Steel02 material, which has a matTag of 1, to simulate the response of the experimental element. The following script is located in `OneBayFrame_Local_SimAppServer.tcl` for the local test and `OneBayFrame_Distr_LabServer.tcl` for the distributed test. The Tcl command is invoked by `expControl`.

```
# Define materials
# ----------------
# uniaxialMaterial Steel02 $matTag $Fy $E $b $R0 $cR1 $cR2 $a1 $a2 $a3 $a4
#uniaxialMaterial Elastic 1 2.8
uniaxialMaterial Steel02 1 1.5 2.8 0.01 18.5 0.925 0.15 0.0 1.0 0.0 1.0

# Define experimental control
# ---------------------------
# expControl SimUniaxialMaterials $tag $matTags
expControl SimUniaxialMaterials 1 1
```

The `expControl` command parameters for `SimUniaxialMaterials` are:

- `$tag` is the unique control tag.
- `$matTags` are the tags of previously defined uniaxial material objects.

6.2 Experimental Setup

The `OneActuator` setup is used for the experimental setup (Figure 6). The following script is located in `OneBayFrame_Local_SimAppServer.tcl` for the local test and `OneBayFrame_Distr_LabServer.tcl` for the distributed test. The Tcl command for the experimental setup is `expSetup`.

```
# Define experimental setup
# -------------------------
# expSetup OneActuator $tag <-control $ctrlTag> $dir -sizeTrialOut $sizeTrial $sizeOut
<-trialDispFact $f> ...
expSetup OneActuator 1 -control 1 1 –sizeTrialOut 1 1
```

The `expSetup` command parameters for `OneActuator` are:

- `$tag` is the unique setup tag.
- `$ctrlTag` is the tag of a previously defined control object. In this case, it is SimUniaxialMaterials.
- `$dir` is the direction of the imposed displacement in the element basic reference coordinate system.
- `$sizeTrial` and `$sizeOut` are the sizes of the element trial and output data vectors, respectively.
- `$f` are trial displacement factor, output displacement factor, and output force factor, respectively. These optional fields are used to factor the imposed and the measured data. The default values are 1.0.

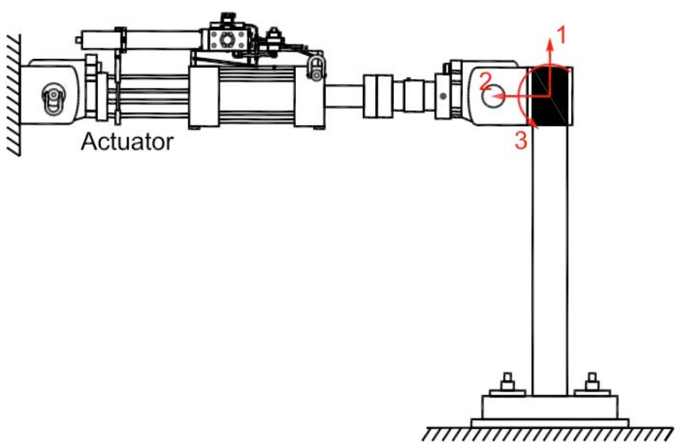

Figure 6: OneActuator Experimental Setup.

6.3 Experimental Element

The experimental element is set to a `twoNodeLink` element (Figure 7). The following script is located in `OneBayFrame_Local_SimAppServer.tcl` for the local test and `OneBayFrame_Distr_SimAppServer.tcl` for the distributed test.

```
# Define experimental element
# ---------------------------
```

```
# left column
# expElement twoNodeLink $eleTag $iNode $jNode -dir $dirs -site $siteTag -initStif $Kij
<-orient <$x1 $x2 $x3> $y1 $y2 $y3> <-pDelta (4 $Mratio)> <-iMod> <-mass $m>
expElement twoNodeLink 1 1 3 -dir 2 -site 1 -initStif 2.8 -orient -1 0 0
```

The `expElement` command parameters for `twoNodeLink` are:

- `$eleTag` is the unique element tag.
- `$iNode` and `$jNode` are the end nodes that the twoNodeLink element connects to.
- `$siteTag` is the tag of a previously defined site object. In this example, it is the `LocalSite` for the local test and the `RemoteSite` for the distributed test.
- `$dirs` are the force-deformation directions in the element local reference coordinate system.
- `$Kij` are the (row wise) initial stiffness matrix components of the element.
- `$x1 $x2 $x3 $y1 $y2 $y3` set the orientation vectors for the element. `x1`, `x2`, and `x3` are vector components in the global coordinates defining the local x-axis. `y1`, `y2`, and `y3` are the same except that they define the y vector which lies in the local x-y plane for the element. `<-orient <$x1 $x2 $x3> $y1 $y2 $y3>` field is optional with default being the global X and Y.
- $Mratio are the optional P-Delta moment contribution ratios. The size of the ratio vector is 4 (entries: [My-$iNode, My-$jNode, Mz-$iNode, Mz-$jNode]) My-$iNode + My-$jNode <= 1.0, Mz-$iNode + Mz-$jNode <= 1.0. The remaining P-Delta moments are resisted by shear couples. (default = [0.0 0.0 0.0 0.0])
- `-iMod` and `$m` are also optional fields. `-iMod` allows for error correction using Nakashima's initial stiffness modification. Default for `-iMod` is false. `$m` is used to assign mass to the element. Its default is zero.

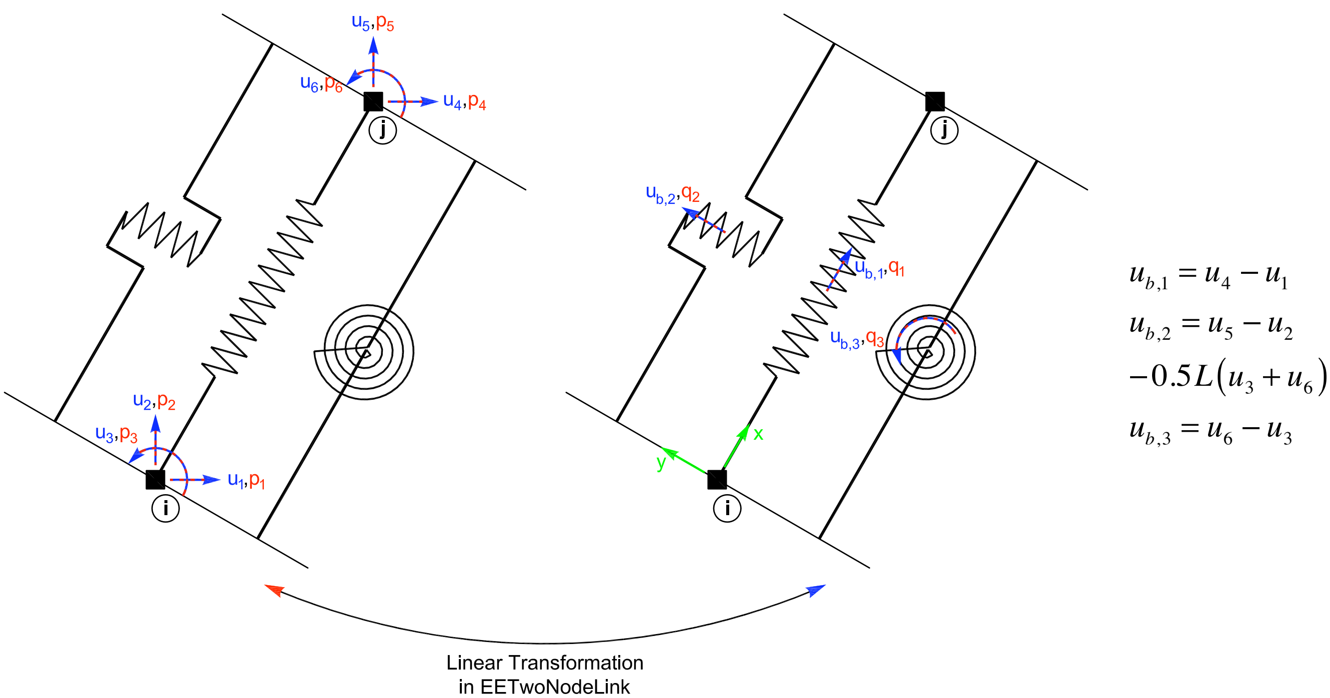

Figure 7: twoNodeLink Experimental Element.

The experimental element works in conjunction with a user-defined generic client element in the FE-software. In this example, the LS-DYNA user-defined element is `e101`, as described in Section 5.2.

6.4 Experimental Site

The experimental site is set to `LocalSite` in `OneBayFrame_Local_SimAppServer.tcl`. This example uses a client to middle-tier-server communication. The following script below is in `OneBayFrame_Local_SimAppServer.tcl`.

```
# Define experimental site
# ------------------------
# expSite LocalSite $tag $setupTag
expSite LocalSite 1 1
```

The `expSite` command parameters for `LocalSite` are:

- `$tag` is the unique site tag.
- `$setupTag` is the tag of a previously defined experimental setup object.

For the geographically distributed test, the script below, `OneBayFrame_Distr_SimAppServer.tcl`, shows that the `expSite` on the middle-tier server is set to `ShadowSite`:

```
# Define experimental site
# ------------------------
# expSite ShadowSite $tag <-setup $setupTag> $ipAddr $ipPort <-ssl> <-dataSize $size>
expSite ShadowSite 1 "127.0.0.1" 8091
```

The `expSite` command parameters for `ShadowSite` are:

- `$tag` is the unique site tag.
- `$setupTag` is the optional tag of a previously defined experimental setup object.
- `$ipAddr` is the IP address of the corresponding `ActorSite`.
- `$ipPort` is the IP port number of the corresponding `ActorSite`.
- `-ssl` is an option that uses OpenSSL. The default is off.
- `$size` is the optional data size being sent.

The `expSite` is set to `ActorSite` on the laboratory server side. The following script is in `OneBayFrame_Distr_LabServer.tcl`:

```
# Define experimental site
# ------------------------
# expSite ActorSite $tag -setup $setupTag $ipPort <-ssl>
expSite ActorSite 1 -setup 1 8091
```

The `expSite` command parameters for `ActorSite` are:

- `$tag` is the unique site tag.
- `$setupTag` is the tag of a previously defined experimental setup object.
- `$ipPort` is the IP port number of the `ActorSite`.
- `-ssl` is an option that uses OpenSSL. The default is off.

7 Running Distributed Hybrid Simulation with Setup on Server Side

A distributed test consists of the multi-tier client-server architecture shown in Figure 8. This section shows how OpenFresco can be used to run a distributed test with the LS-DYNA client, the middle-tier servers and the laboratory server running as four separate processes. For this example, the processes are run on the same computer, but in most applications, the client and the first middle-tier server processes would be run on one machine and the second middle-tier server and the backend server processes would be run on a different machine across a network. The experimental setup can be defined on either the first or second middle-tier server side. In this example, the setup is located on the second middle-tier server side.

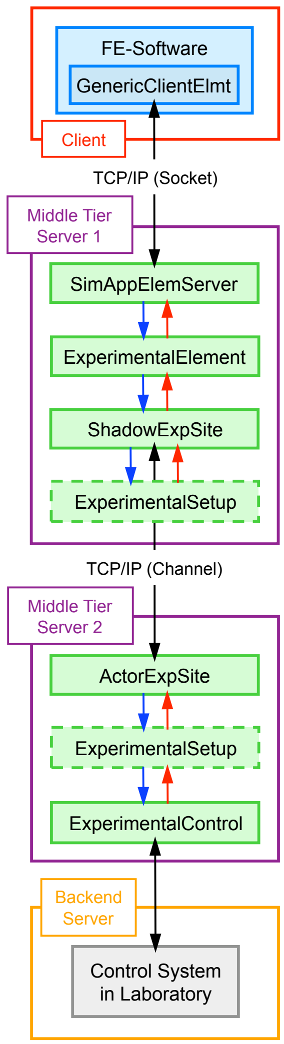

Figure 8: Distributed Hybrid Simulation using the Generic-Client Element.

To run this simulation perform the following steps:

- Start the OpenFresco executable file (OpenFresco.exe) from the directory where the `OneBayFrame_Distr_LabServer.tcl` resides.

- At the prompt, type **`source OneBayFrame_Distr_LabServer.tcl`** and hit **enter** (Figure 9).

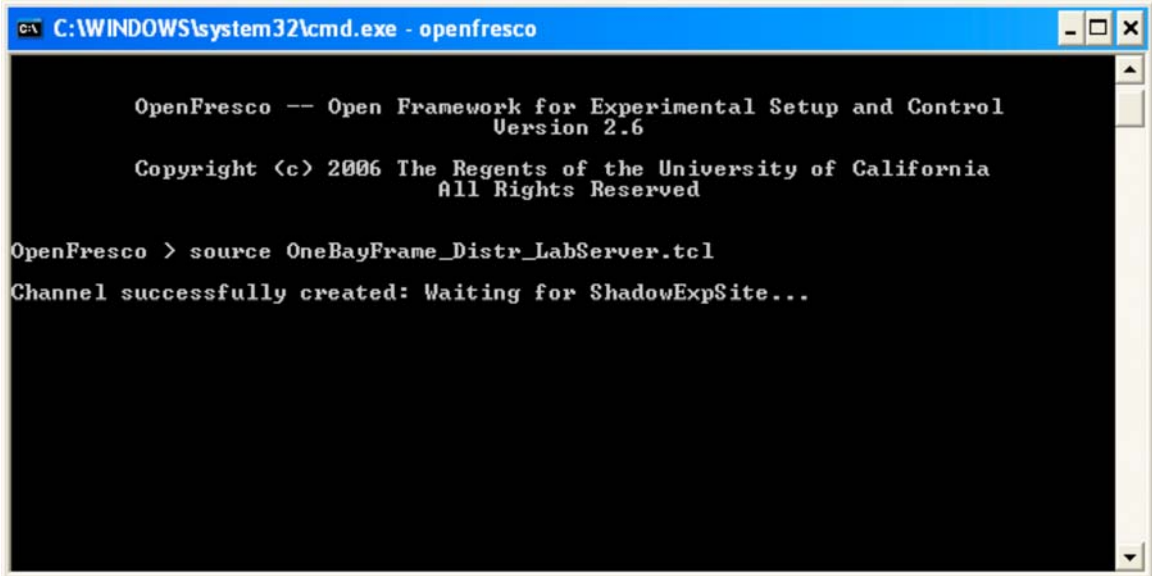

Figure 9: OpenFresco 2nd Middle-Tier & Lab Server Window for Distributed Test.

- Start the OpenFresco executable again (OpenFresco.exe) from the directory where the `OneBayFrame_Distr_SimAppServer.tcl` resides. This opens another OpenFresco command window.
- Make sure that the startSimAppElemServer instead of the startSimAppSiteServer command is used in **`OneBayFrame_Distr_SimAppServer.tcl`**.
- At the prompt, type **`source OneBayFrame_Distr_SimAppServer.tcl`** and hit **enter** (Figure 10).

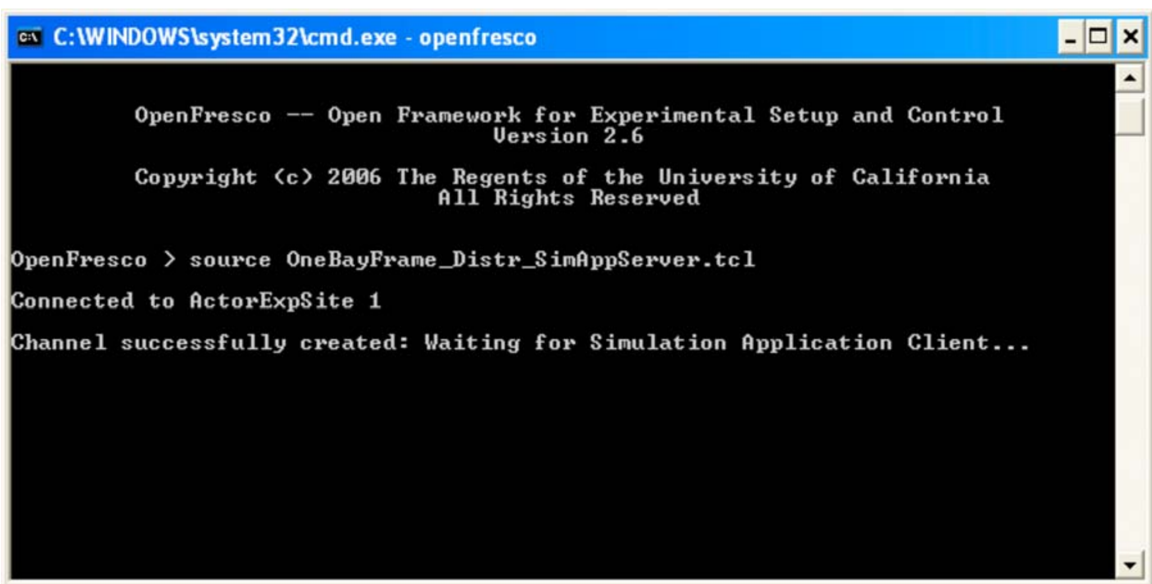

Figure 10: OpenFresco 1st Middle-Tier Server Window for Distributed Test.

- The OpenFresco lab server window now looks like Figure 11.

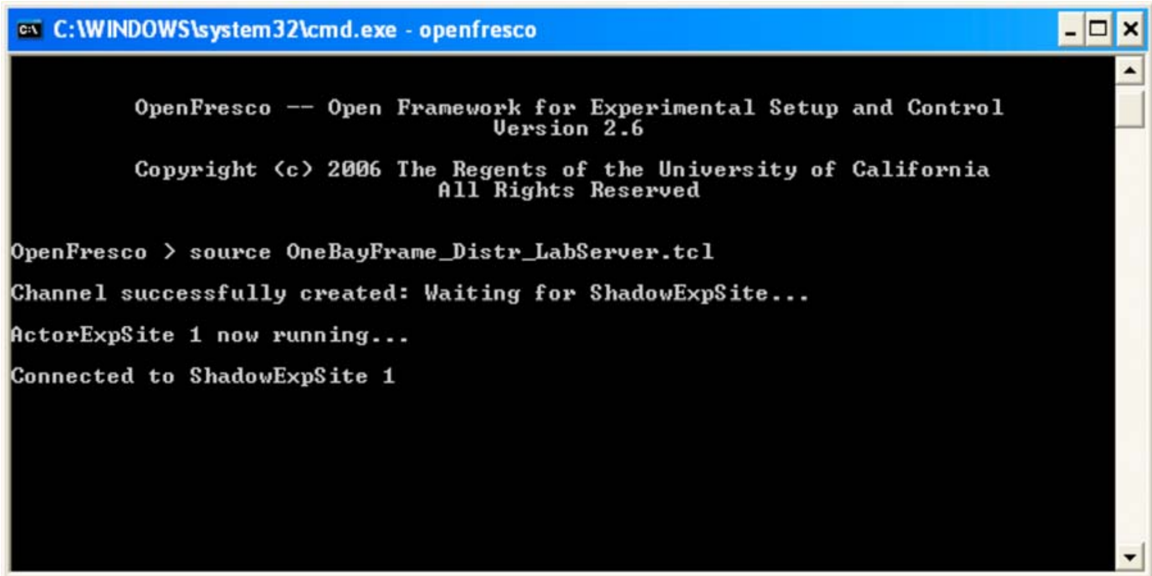

Figure 11: OpenFresco 2nd Middle-Tier & Lab Server Window for Distributed Test during Simulation.

- Start LS-DYNA using the input file `OneBayFrame_LSDyna.k` (Figure 12).

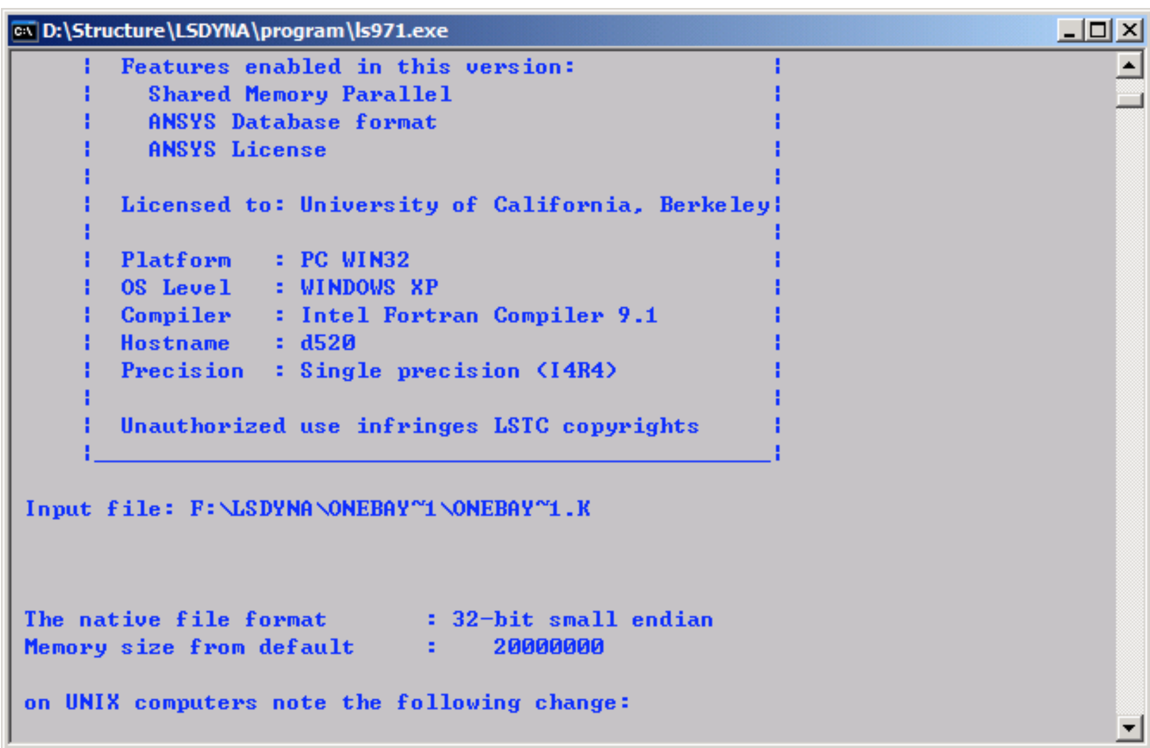

Figure 12: LS-DYNA Client Command Window for Distributed Test.

- After the simulation is complete, the LS-DYNA Client, OpenFresco laboratory server, and simulation application command windows look like Figures 13, 14, and 15 respectively.

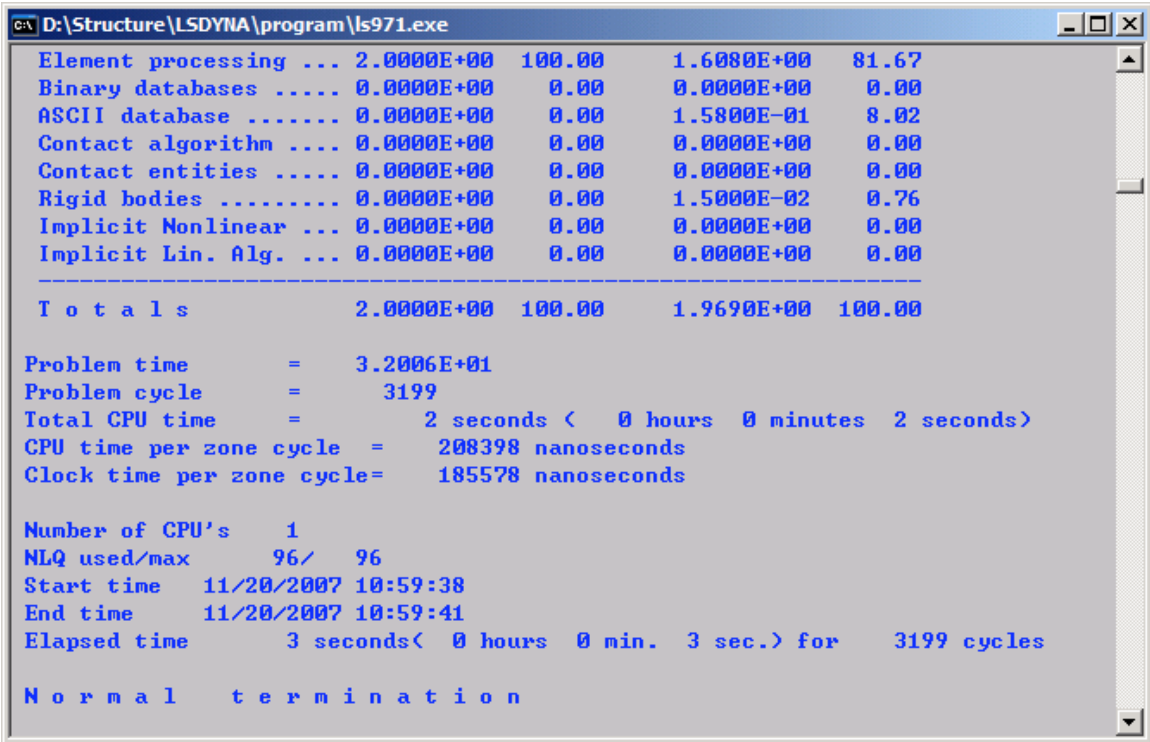

Figure 13: LS-DYNA Client Command Window for Distributed Test after Simulation.

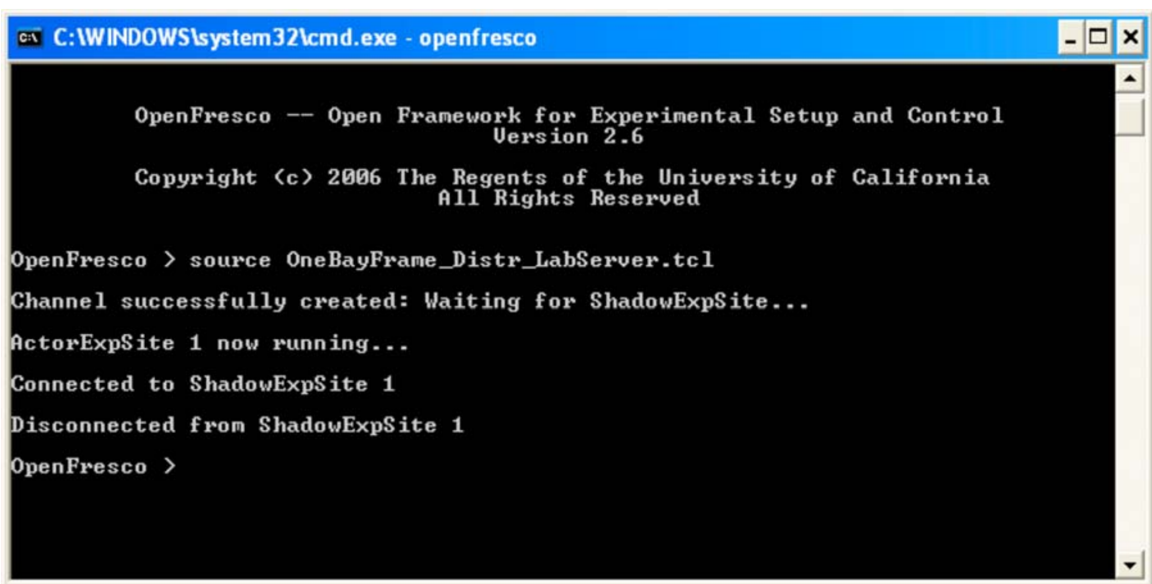

Figure 14: OpenFresco 2nd Middle-Tier & Lab Server Window for Distributed Test after Simulation.

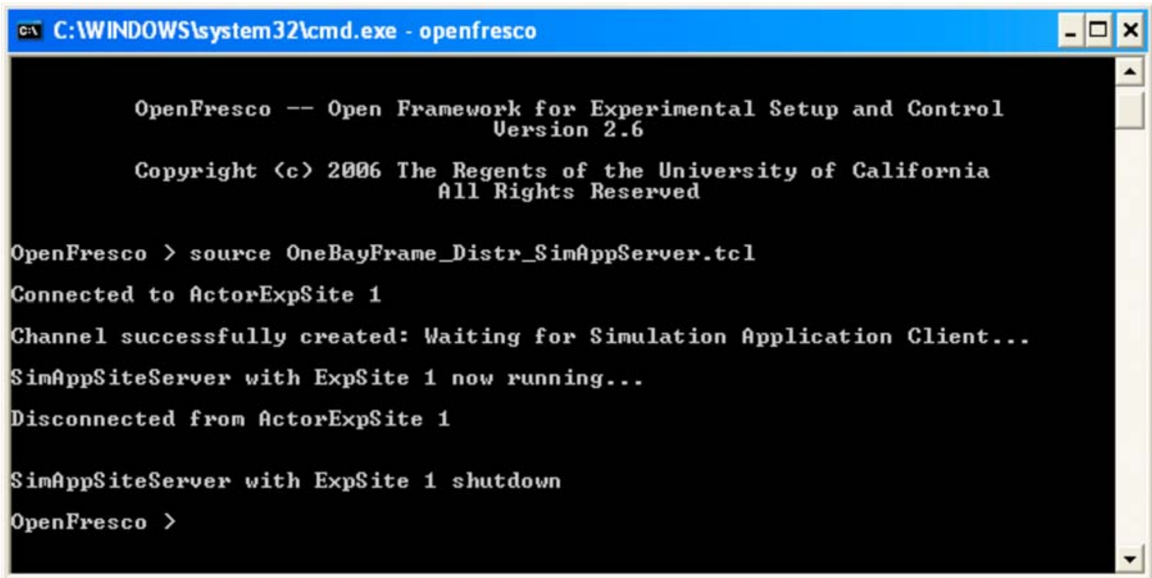

Figure 15: OpenFresco 1st Middle-Tier Server Window for Distributed Test after Simulation.

8 Results

The response quantities for this example are plotted in Figures 16 to 19.

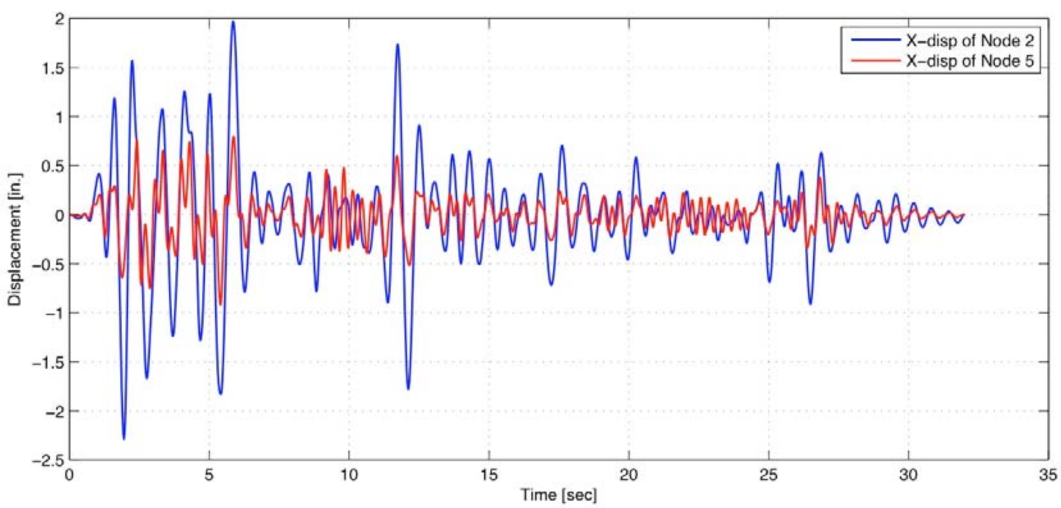

Figure 16: Displacements vs. Time for LS-DYNA Example.

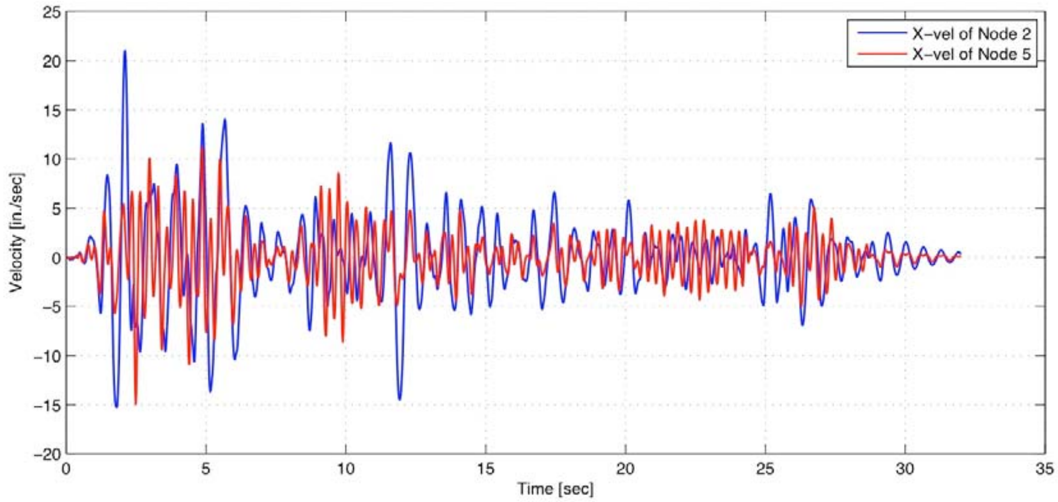

Figure 17: Velocity vs. Time for LS-DYNA Example.

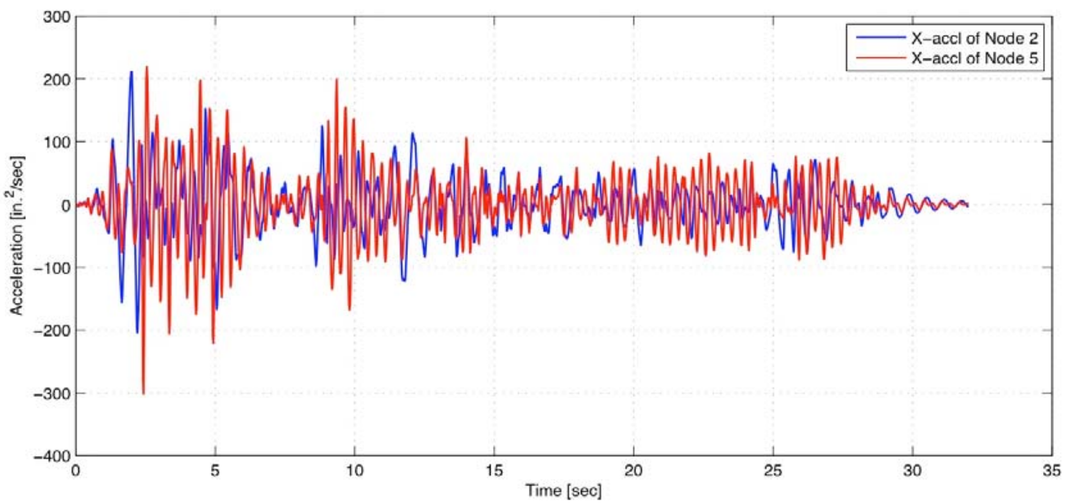

Figure 18: Acceleration vs. Time for LS-DYNA Example.

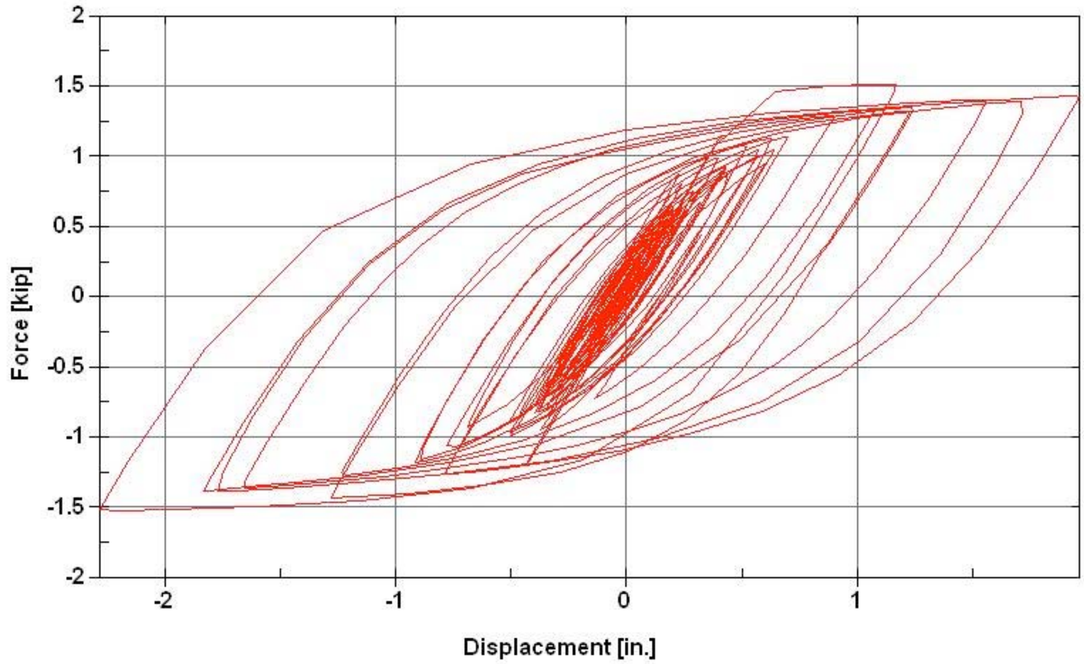

Figure 19: Experimental Element Hysteresis Loop for LS-DYNA Example.

9 References

Chopra, A.K., “Dynamics of Structures, Theory and Applications to Earthquake Engineering”, 3rd edition, Prentice Hall, 2006, 912 pp.

Hallquist, J.O. (2006) “LSDYNA® theory manual”, from http://www.lstc.com/pages/manuals_down.htm

LSTC (2006). “LSDYNA® keyword user’s manual”, from http://www.lstc.com/pages/manuals_down.htm