



*Enabling the Network for
Earthquake Engineering Simulation*



TR-2009-[ID]

OpenFresco Framework for Hybrid Simulation: MATLAB[®] Example

Andreas Schellenberg, 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

Table of Contents

1	Introduction: MATLAB® Example Using One-Bay Frame Model	3
2	Required Files.....	3
3	Structural Model.....	4
4	Ground Motion	4
5	The MATLAB® Script	5
5.1	Element Properties.....	5
5.2	Experimental Element	6
5.3	Mass and Damping Matrices	8
5.4	Hybrid Simulation Time-Integration Schemes	8
6	OpenFresco Tcl Commands	9
6.1	Experimental Control.....	9
6.2	Experimental Setup.....	9
6.3	Experimental Element	10
7	Running the Example	10
7.1	Local Hybrid Simulation	10
7.2	Distributed Hybrid Simulation with Setup on Server Side.....	13
8	Results	18
9	References	20

Table of Figures

Figure 1: MATLAB® One-Bay Frame Model.....	4
Figure 2: 1940 El Centro Ground Motion.	5
Figure 3: OneActuator Experimental Setup.	10
Figure 4: Local Hybrid Simulation using the Experimental Element.	11
Figure 5: OpenFresco Command Window for Local Test.	12
Figure 6: MATLAB® Client Command Window for Local Test after Simulation.	12
Figure 7: OpenFresco Command Window for Local Test after Simulation.	13
Figure 8: Distributed Hybrid Simulation using the Experimental Element.	14
Figure 9: OpenFresco Lab Server Window for Distributed Test.	15
Figure 10: OpenFresco Simulation Application Server Window for Distributed Test.	16
Figure 11: OpenFresco Lab Server Window for Distributed Test during Simulation.	16
Figure 12: MATLAB® Client Command Window for Distributed Test after Simulation.	17
Figure 13: OpenFresco Lab Server Window for Distributed Test after Simulation.....	17
Figure 14: OpenFresco Simulation Application Server Window for Distributed Test after Simulation. .	18
Figure 15: Displacements vs. Time for One-Bay Frame Example, Explicit Newmark Integration.....	18
Figure 16: Element Hysteresis Loops for One-Bay Frame Example, Explicit Newmark Integration.	19
Figure 17: Displacements vs. Time for One-Bay Frame Example, Alpha-OS Integration.....	19
Figure 18: Element Hysteresis Loops for One-Bay Frame Example, Alpha-OS Integration.	20



1 Introduction: MATLAB® Example Using One-Bay Frame Model

This example shows how MATLAB® can be used as the computational driver for a hybrid simulation with OpenFresco. It uses a simple One-Bay Frame model with the explicit Newmark and Alpha-OS time integration schemes. It shows how to run both local and distributed hybrid simulations. The One-Bay Frame example is a fully simulated test, meaning that the experimental control is set to simulation mode. It does not require a physical specimen to run. The response results from the simulation are provided for comparison.

2 Required Files

For the MATLAB® example, the following files are necessary. Some are located in:

User's Directory\OpenFresco\trunk\EXAMPLES\OneBayFrame\Matlab

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

The following MATLAB® files should be in this directory:

- OneBayFrame_NewmarkExplicit.m
- OneBayFrame_AlphaOS.m
- Elastic.m
- EP.m
- EP_spring.m
- Experimental.m
- zForce.m
- elcentro.txt

Others are located in the following directory, which needs to be added to the MATLAB® path:

User's Directory\OpenFresco\trunk\src\simApplicationClient\matlab

The files in this directory are:

- TCPSocket.m
- TCPSocket.mexw32¹

¹ This mex file only works in conjunction with MATLAB® R2009a. If an older or newer version of MATLAB® is used, this mex file needs to be replaced with a mex file created from the version of MATLAB® being used. To create a new mex file, OpenFresco must be installed using the Full Installation option. Change the MATLAB® working directory to User's Directory\OpenFresco\trunk\src\simApplicationClient\matlab. Rename or delete the existing TCPSocket.mex32 file in this directory. At the MATLAB® command prompt type `mex -v -O TCPSocket.c "User's Directory\OpenFresco\trunk\src\simApplicationClient\c\socket.c" ws2_32.lib`. Make sure to add ... \OpenFresco\trunk\src\simApplicationClient\matlab to the MATLAB® path if not done so yet before running the examples described below. The default C compiler in MATLAB® does not create the mex file correctly. Before using the mex command, use `mex -setup` to change to a different C compiler. MS Visual Studio 6.0 and later work well.



Some Tcl files are needed in addition to the ones 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 model consists of two columns, Element 1 and 2, connected by a spring, Element 3. A lumped mass is placed at the top of each column. The two column bases are fixed. The columns are axially rigid, and the tops are free to rotate. Imperial units are used [inches, kips, sec].

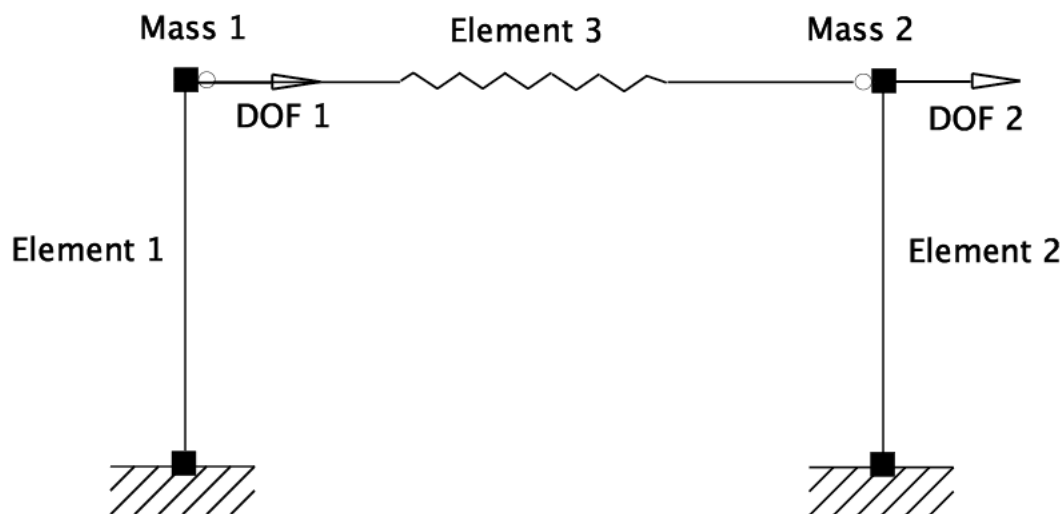


Figure 1: MATLAB® One-Bay Frame Model.

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).



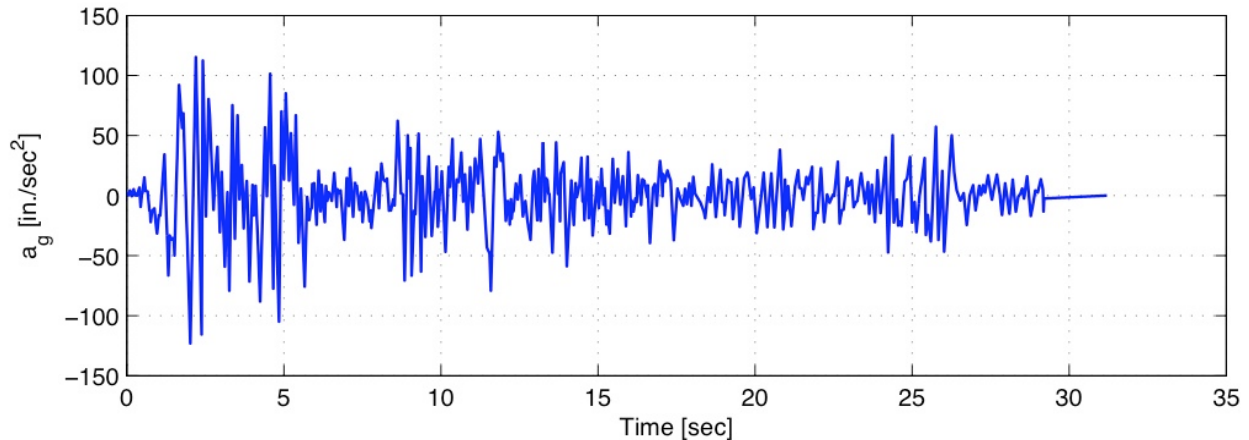


Figure 2: 1940 El Centro Ground Motion.

5 The MATLAB® Script

This section contains explanations of the MATLAB® script, `OneBayFrame_NewmarkExplicit.m`. It details how the model is represented in MATLAB®. The section also includes a discussion about setting the parameters for the explicit Newmark and Alpha-OS time-integration schemes.

5.1 Element Properties

Four settings are available for each element. In this example, Element 1 is set to `Experimental` and Element 2 and 3 are set to `Analytical Elastic`.

```
% Element 1 Parameters - Column Element
ElementData(1).type      = 3;                                % Type of Element:
                                                                % 0=turned off,
                                                                % 1=analytical elastic,
                                                                % 2=analytical-EP with kinematic hard,
                                                                % 3=experimental

ElementData(1).len       = 54;                                % Element Length (in.)
ElementData(1).k_elem    = 2.8;                                % Elastic Stiffness (kips/in)
ElementData(1).My        = 81;                                % Plastic Capacity (kip*in)
ElementData(1).hk        = 0.01;                               % Kinematic Hardening Ratio
ElementData(1).qb        = 0;                                  % Back force
ElementData(1).v_pl      = 0;                                  % Element Plastic Deformation
ElementData(1).socketID  = socketID;                           % Socket ID
ElementData(1).dataSize  = dataSize;                           % size of send and receive vectors

% Element 2 Parameters - Column Element
ElementData(2).type      = 1;                                % Type of Element:
                                                                % 0=turned off,
                                                                % 1=analytical elastic,
                                                                % 2=analytical-EP with kinematic hard,
                                                                % 3=experimental

ElementData(2).len       = 54;                                % Element Length (in.)
ElementData(2).k_elem    = 5.6;                                % Elastic Stiffness (kips/in)
ElementData(2).My        = 162;                                % Plastic Capacity (kip*in)
ElementData(2).hk        = 0.01;                               % Kinematic Hardening Ratio
ElementData(2).qb        = 0;                                  % back force
ElementData(2).v_pl      = 0;                                  % Element Plastic Deformation
% ElementData(2).socketID = socketID;                           % Socket ID
% ElementData(2).dataSize = dataSize;                           % size of send and receive vectors
```



```

% Element 3 Parameters - Spring Element
ElementData(3).type      = 1;           % Type of Element:
                                     % 0=turned off,
                                     % 1=analytical elastic,
                                     % 2=analytical-EP with kinematic hard,
                                     % 3=experimental
ElementData(3).k_elem    = 2.0;         % Elastic Stiffness (kips/in)
ElementData(3).Fy        = 25;          % Yield Force (kips)
ElementData(3).hk        = 0.01;        % Kinematic Hardening Ratio
ElementData(3).qb        = 0;           % Back force
ElementData(3).v_pl      = 0;           % Element Plastic Deformation
% ElementData(3).socketID = socketID;    % Socket ID
% ElementData(3).dataSize = dataSize;    % size of send and receive vectors

```

ElementData Inputs:

- `ElementData($ElementTag).type` toggles between a zero force element, a linear-elastic element, an elastic-plastic element with kinematic hardening, and an experimental element. If an experimental element is used the `socketID` and `dataSize` lines need to be uncommented.
- `ElementData($ElementTag).len` is the length of the element.
- `ElementData($ElementTag).k_elem` is the initial stiffness of the element.
- `ElementData($ElementTag).My` is the plastic hinge capacity. This is to be used with the elastic-plastic element with kinematic hardening.
- `ElementData($ElementTag).hk` is the kinematic hardening ratio and not the kinematic hardening stiffness. This is to be used with elastic-plastic elements with kinematic hardening.
- `ElementData($ElementTag).qb` and `ElementData($ElementTag).v_pl` are the back force and the plastic deformation in the element. These too are to be used with elastic-plastic elements.
- `ElementData($ElementTag).socketID` is the id of the socket connecting to the OpenFresco middle-tier server. This field should be commented out unless the element is set to experimental element.
- `ElementData($ElementTag).dataSize` is the size of the data being sent and received. Sending less data usually does not result in a shorter run time. This field allows the user to adjust the data size to achieve an optimal run time. It should be commented out unless the element is set to experimental element.

5.2 Experimental Element

In order to use OpenFresco with a finite element (FE) software, the FE software must support the addition of user-defined elements. This example shows how a user-defined experimental element can be programmed in MATLAB®. The Matlab user-defined elements use the MATLAB® mex file, `TCPSocket.mexw32`, to communicate with the OpenFresco middle-tier server.

To use this functionality, add the following path to MATLAB®:

User's Directory\OpenFresco\trunk\SRC\simApplicationClient\matlab

Then typing `help TCPSocket` at the MATLAB® prompt shows how to use the `TCPSocket` command. This example uses the experimental element programmed in `Experimental.m`.



This directory contains other experimental elements for the MATLAB® toolbox FEDEASLab v2.3. Below is a list of elements available:

- EEFrame2d.m
- EEFrame3d.m
- EETruss.m
- EETwoNodeLink2d.m
- GenericClient2d.m

Below are excerpts from OneBayFrame_NewmarkExplicit.m. The first one shows how to open a client connection to a server utilizing the TCPSocket command with an ip-address of "127.0.0.1" and an ip-port number of 8090.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Setup Connection %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
socketID = TCPSocket('openConnection','127.0.0.1',8090);
```

The data size at the middle-tier server is set also using the TCPSocket command.

```
% set the data size for the experimental site
dataSize = 2;
sData = zeros(1,dataSize);
dataSizes = int32([1 0 0 0 0, 0 0 0 1 0, dataSize]);
TCPSocket('sendData',socketID,dataSizes,11);
```

The connection is closed after the simulation has completed.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Disconnect from Experimental Site %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sData(1) = 99;
TCPSocket('sendData',socketID,sData,dataSize);
TCPSocket('closeConnection',socketID);
```

The following code is an excerpt from Experimental.m. It sends the DOF 1 (Figure 1) displacement to the middle-tier server.

```
% send trial response to experimental site
sData(1) = 3;
sData(2) = u;
TCPSocket('sendData',socketID,sData,dataSize);
```

In this code, the measured force from the middle-tier server is obtained:

```
% get measured resisting forces
sData(1) = 10;
TCPSocket('sendData',socketID,sData,dataSize);
rData = TCPSocket('recvData',socketID,dataSize);
```

Then, the current state is committed:

```
% commit state
sData(1) = 5;
TCPSocket('sendData',socketID,sData,dataSize);
```



Finally, the element resisting force is set.

```
% Set resisting force
ElementPost.p_r = rData(1);
```

Actions are communicated to the server by setting `sData(1)` to some integer value.

`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

5.3 Mass and Damping Matrices

This example uses a diagonalized mass matrix and mass proportional damping. Rayleigh damping can be used by uncommenting the appropriate lines.

```
% Mass Matrix, M
M = [mass(1) 0; 0 mass(2)];

% Mass Proportional Damping Matrix, C
a_o = zeta(1) * 2 * w(1);
C = a_o*M;

% Rayleigh Damping Matrix, C
% a_o = zeta(1) * 2 * w(1) * w(2) / (w(1)+w(2));
% a_1 = zeta(1) * 2 / (w(1)+w(2));
% C = a_o*M + a_1*K_el;
```

5.4 Hybrid Simulation Time-Integration Schemes

The example uses the explicit Newmark integration method with $\beta = 0$ and $\gamma = 0.5$.

```
% Newmark Parameters Beta and Gamma
beta = 0; %Set to be explicit
gamma = 1/2;
```



The Alpha-Operator Splitting scheme can be used by running `OneBayFrame_AlphaOS.m`. `OneBayFrame_AlphaOS.m` is identical to `OneBayFrame_NewmarkExplicit.m` except for the time-integration script. To introduce some numerical damping, α is set to $-1/6$ in the file.

```
% Alpha OS Parameters - alpha, beta and gamma
alpha = -1/6;           %alpha [-1/3,0]
beta  = (1-alpha)^2/4;
gamma = (1-2*alpha)/2;
```

6 OpenFresco Tcl Commands

This section contains explanations of the common OpenFresco Tcl commands used in both the local and the distributed tests. The test specific commands are explained in Section 7. Each subsection highlights a Tcl command and the script that contains the command.

6.1 Experimental Control

The experimental control is set to be `SimUniaxialMaterials` for this example. `SimUniaxialMaterials` uses the `Steel02` material, which has a `matTag` of 1, to simulate the response of the experimental element. The script below is in `OneBayFrame_Local_SimAppServer.tcl` for the local test and `OneBayFrame_Distr_LabServer.tcl` for the distributed test. The experimental control is defined by the Tcl command `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 in Figure 3. The script below 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:

```
# 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.

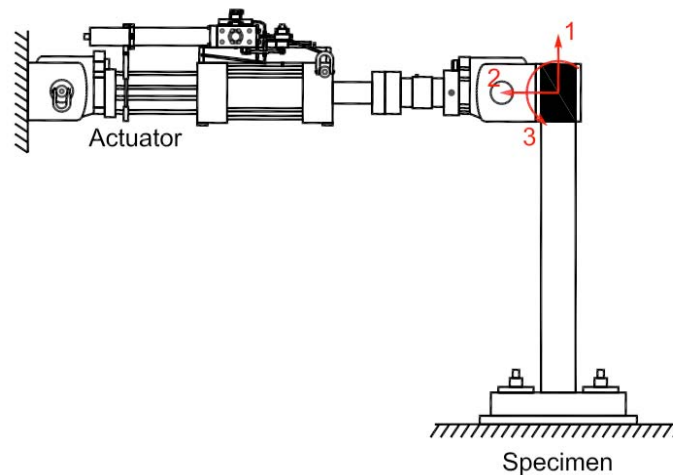


Figure 3: OneActuator Experimental Setup.

6.3 Experimental Element

OpenFresco provides two ways to define an experimental element. The first, and more common, is to define the experimental element on the middle-tier server side and to use a generic-client element in the finite element software. A second option is to define an experimental element directly in the finite element software and therefore not use any element on the middle-tier server side. This approach is illustrated in Figures 4 and 8. The experimental element is defined directly in MATLAB® (Section 5.2), therefore no experimental element is defined on the middle-tier server side for this example.

7 Running the Example

7.1 Local Hybrid Simulation

This example uses the client-middle-tier-server architecture for running a local hybrid simulation (Figure 4). As mentioned earlier, the simple MATLAB® computational simulation program uses an experimental element instead of a generic-client element.



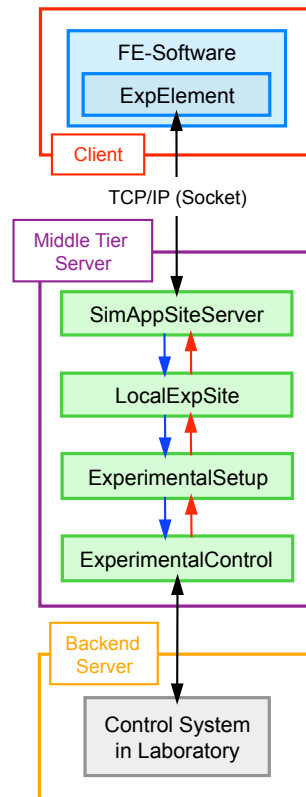


Figure 4: Local Hybrid Simulation using the Experimental Element.

The experimental site is set to `LocalSite`. There is a client to middle-tier-server communication in this example. The code segment below is in the `OneBayFrame_Local_SimAppServer.tcl` script.

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

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

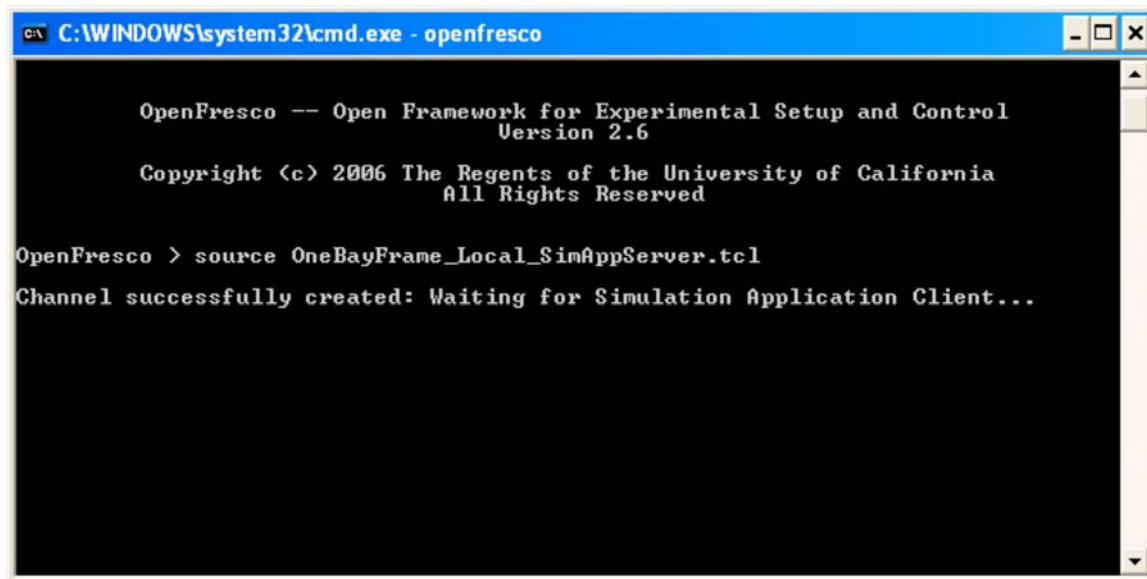
- `$tag` is the unique site tag.
- `$setupTag` is the tag of a previously defined experimental setup object.

To run this simulation perform the following steps:

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



- At the prompt, type **source OneBayFrame_Local_SimAppServer.tcl** and hit **enter** (Figure 5).



```

C:\WINDOWS\system32\cmd.exe - openfresco

OpenFresco -- Open Framework for Experimental Setup and Control
Version 2.6

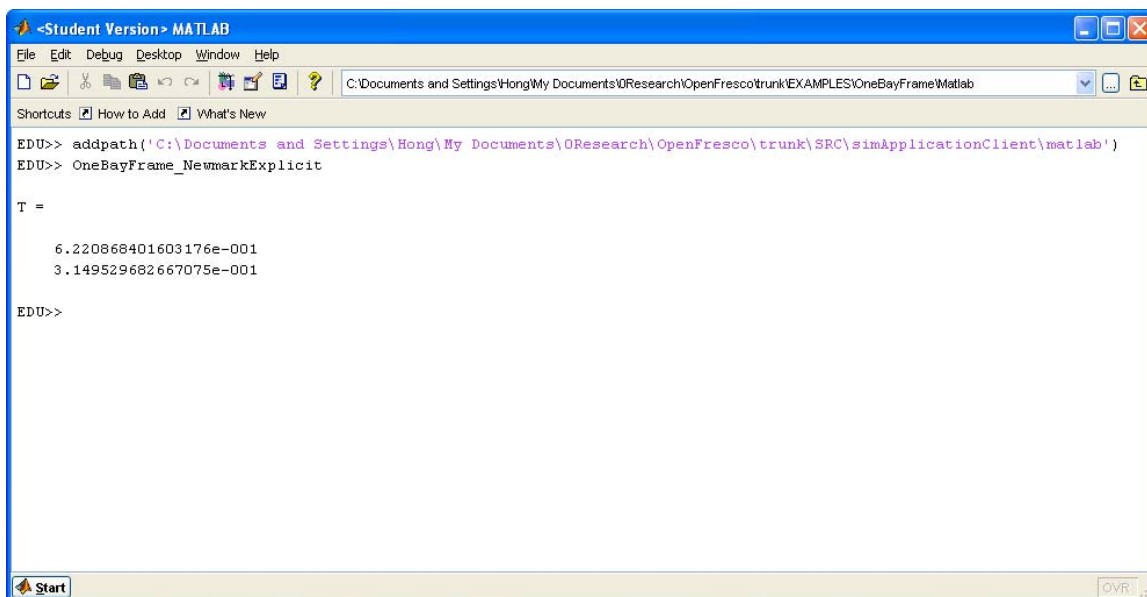
Copyright (c) 2006 The Regents of the University of California
All Rights Reserved

OpenFresco > source OneBayFrame_Local_SimAppServer.tcl
Channel successfully created: Waiting for Simulation Application Client...

```

Figure 5: OpenFresco Command Window for Local Test.

- Start MATLAB®.
- Add the folder that contains the `TCP Socket.mexw32` file to the MATLAB® path.
- Switch to the directory with the MATLAB® examples, User's Directory\OpenFresco\trunk\EXAMPLES\OneBayFrame\Matlab.
- At the prompt, type **OneBayFrame_NewmarkExplicit** and hit **enter** (Figure 6).



```

<Student Version> MATLAB
File Edit Debug Desktop Window Help
C:\Documents and Settings\Hong\My Documents\OResearch\OpenFresco\trunk\EXAMPLES\OneBayFrame\Matlab
Shortcuts How to Add What's New
EDU>> addpath('C:\Documents and Settings\Hong\My Documents\OResearch\OpenFresco\trunk\src\simApplicationClient\matlab')
EDU>> OneBayFrame_NewmarkExplicit

T =

    6.220868401603176e-001
    3.149529682667075e-001

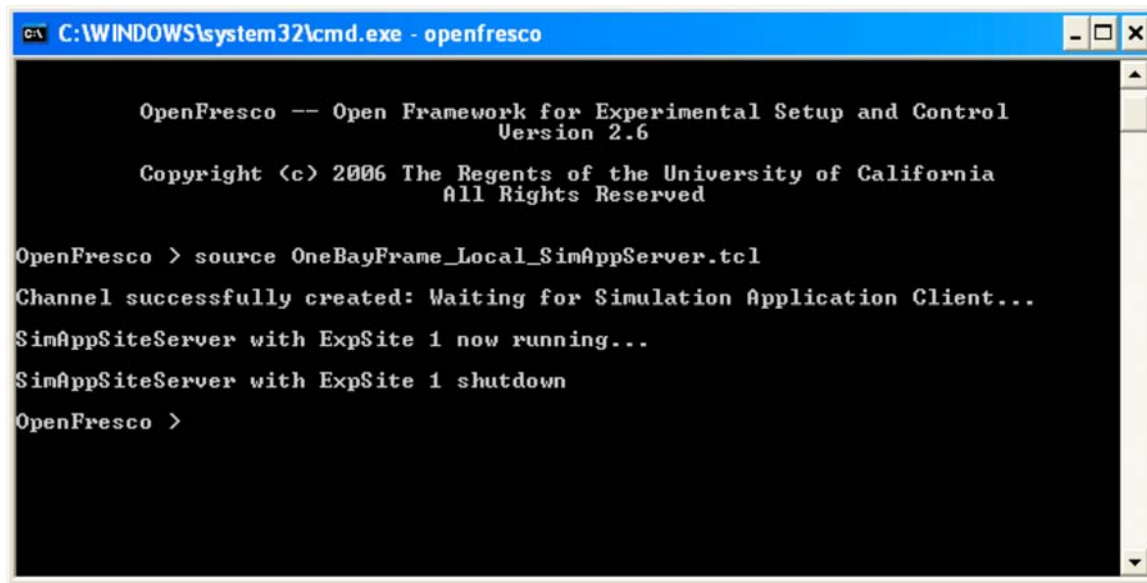
EDU>>

```

Figure 6: MATLAB® Client Command Window for Local Test after Simulation.



- After the simulation has finished, the OpenFresco command window should look like Figure 7. Type **exit** or press **ctrl+c** to exit the OpenFresco command prompt.



```
C:\WINDOWS\system32\cmd.exe - openfresco

OpenFresco -- Open Framework for Experimental Setup and Control
Version 2.6

Copyright (c) 2006 The Regents of the University of California
All Rights Reserved

OpenFresco > source OneBayFrame_Local_SimAppServer.tcl
Channel successfully created: Waiting for Simulation Application Client...
SimAppSiteServer with ExpSite 1 now running...
SimAppSiteServer with ExpSite 1 shutdown
OpenFresco >
```

Figure 7: OpenFresco Command Window for Local Test after Simulation.

7.2 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 demonstrates how OpenFresco can be used to run a distributed test with the client, the first middle-tier server and the second middle-tier and backend server running as three separate processes. For the example, the processes are run on the same computer, but in a practical application the client and the first middle-tier server processes would be run on one machine and the second middle-tier and 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.



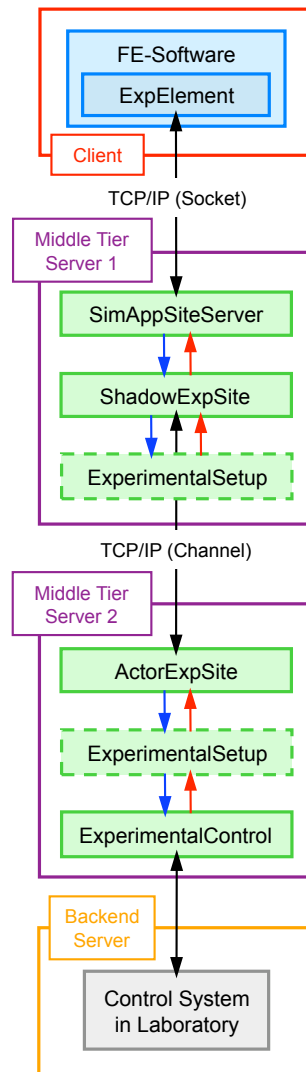


Figure 8: Distributed Hybrid Simulation using the Experimental Element.

In the script below, `OneBayFrame_Distr_SimAppServer.tcl`, shows that the `expSite` on the first 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 second middle-tier server. The script below is found 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.

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).

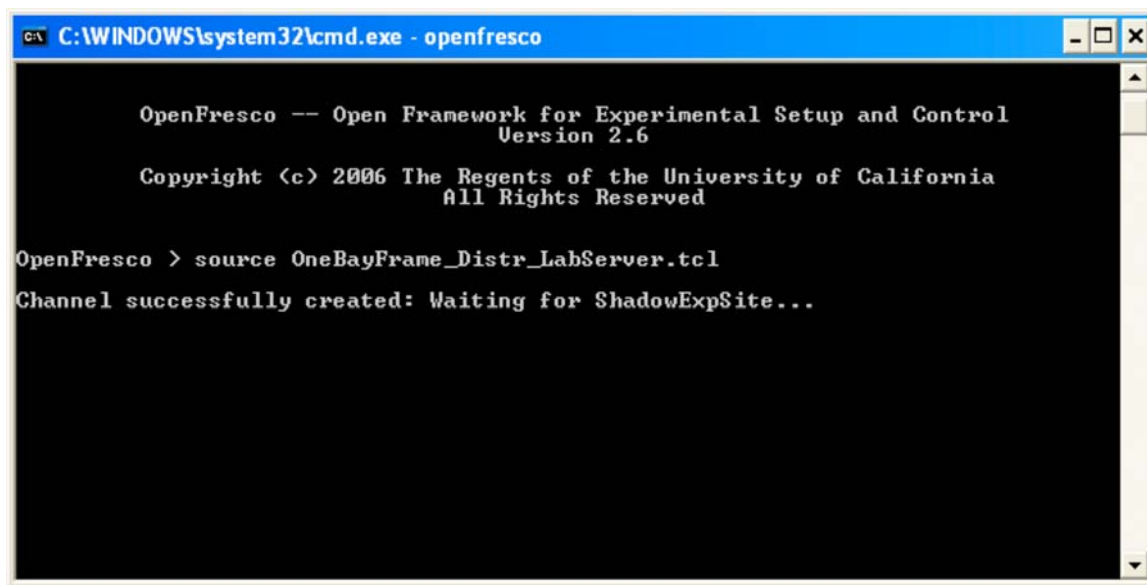
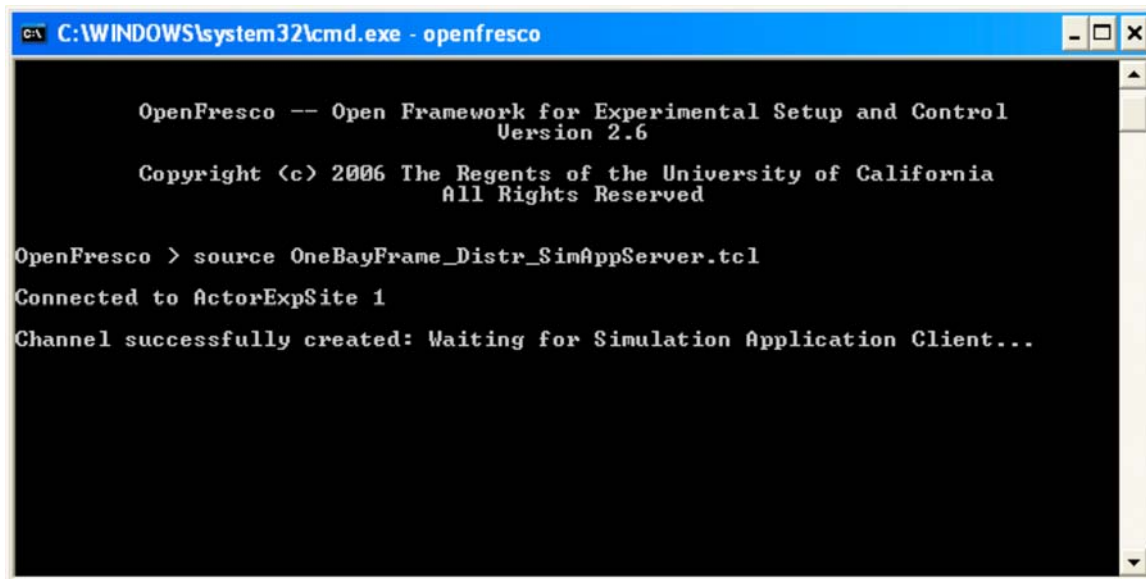


Figure 9: OpenFresco Lab Server Window for Distributed Test.

- Start the `OpenFresco` executable again (`OpenFresco.exe`) from the directory where the `OneBayFrame_Distr_SimAppServer.tcl` resides. This opens another `OpenFresco` command window.



- At the prompt, type `source OneBayFrame_Distr_SimAppServer.tcl` and hit `enter` (Figure 10).



```
C:\WINDOWS\system32\cmd.exe - openfresco

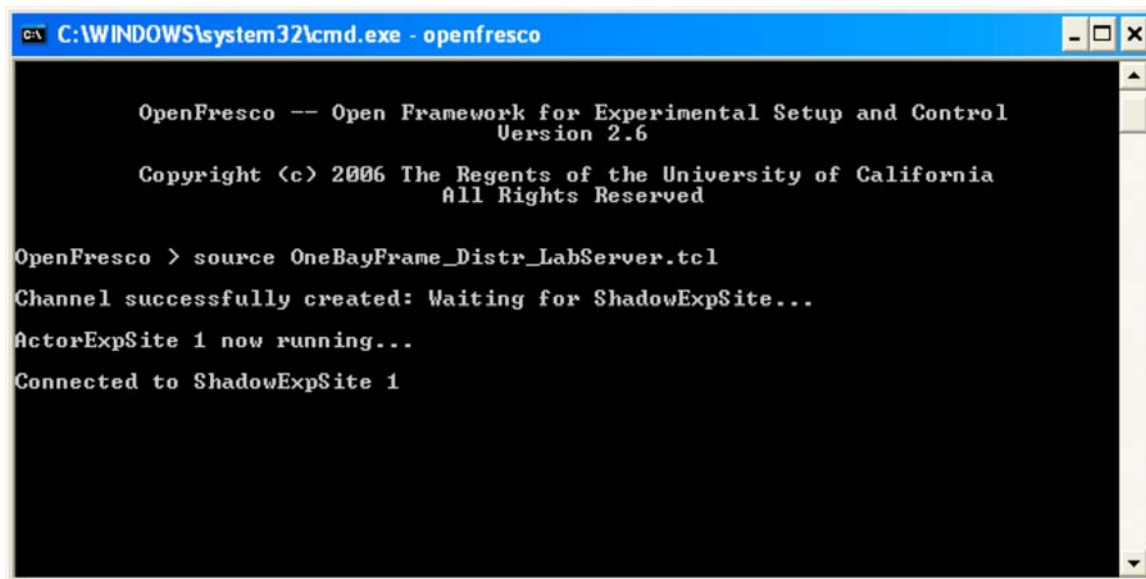
OpenFresco -- Open Framework for Experimental Setup and Control
Version 2.6

Copyright (c) 2006 The Regents of the University of California
All Rights Reserved

OpenFresco > source OneBayFrame_Distr_SimAppServer.tcl
Connected to ActorExpSite 1
Channel successfully created: Waiting for Simulation Application Client...
```

Figure 10: OpenFresco Simulation Application Server Window for Distributed Test.

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



```
C:\WINDOWS\system32\cmd.exe - openfresco

OpenFresco -- Open Framework for Experimental Setup and Control
Version 2.6

Copyright (c) 2006 The Regents of the University of California
All Rights Reserved

OpenFresco > source OneBayFrame_Distr_LabServer.tcl
Channel successfully created: Waiting for ShadowExpSite...
ActorExpSite 1 now running...
Connected to ShadowExpSite 1
```

Figure 11: OpenFresco Lab Server Window for Distributed Test during Simulation.

- Start MATLAB®.
- Add the folder that contains the `TCPsocket.mexw32` file to the MATLAB® path if not done so yet in the previous local hybrid simulation.



- Switch to the directory with the MATLAB® examples, User's Directory\OpenFresco\trunk\EXAMPLES\OneBayFrame\Matlab.
- At the prompt, type **OneBayFrame_NewmarkExplicit** and hit **enter**. The simulation should now be running.
- After it has finished, the MATLAB® window should look like Figure 12.

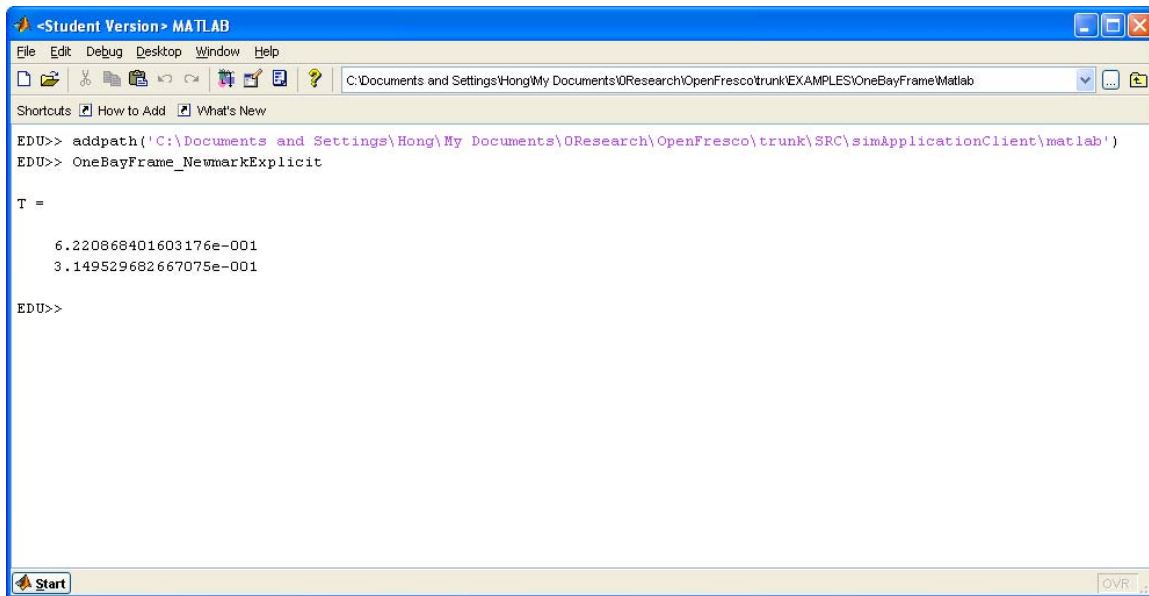


Figure 12: MATLAB® Client Command Window for Distributed Test after Simulation.

- The OpenFresco lab server and simulation application command windows look like Figures 13 and 14, respectively.

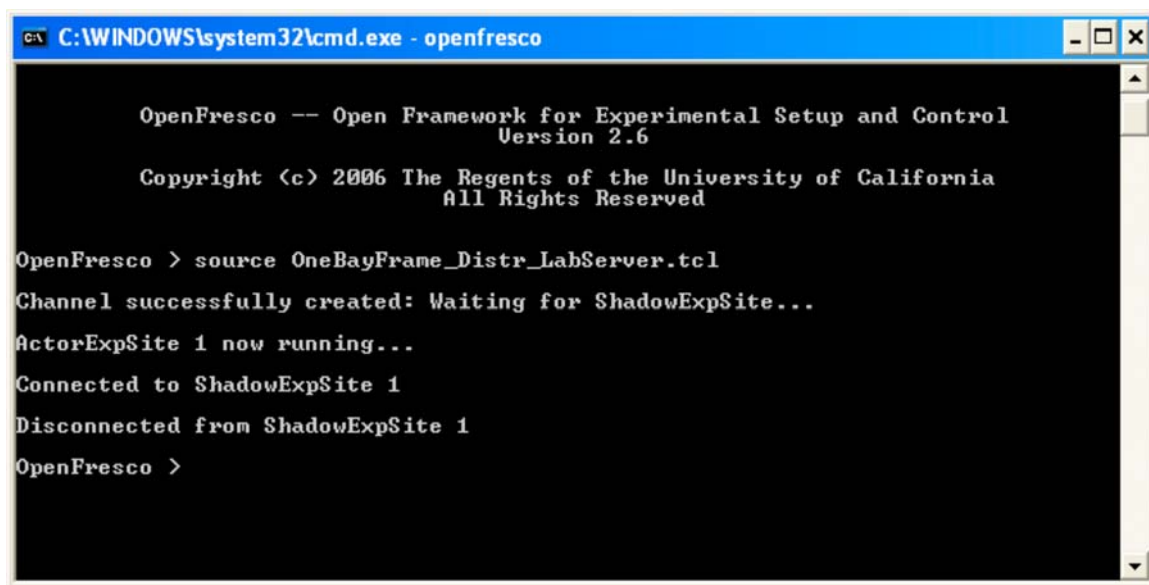
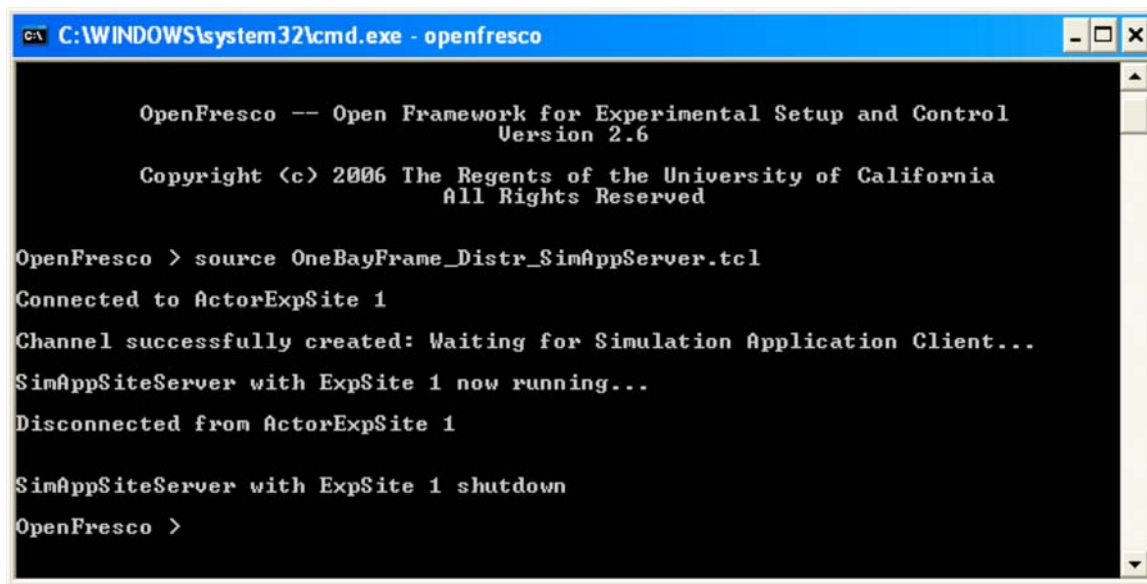


Figure 13: OpenFresco Lab Server Window for Distributed Test after Simulation.





```

C:\WINDOWS\system32\cmd.exe - openfresco

OpenFresco -- Open Framework for Experimental Setup and Control
Version 2.6

Copyright (c) 2006 The Regents of the University of California
All Rights Reserved

OpenFresco > source OneBayFrame_Distr_SimAppServer.tcl
Connected to ActorExpSite 1
Channel successfully created: Waiting for Simulation Application Client...
SimAppSiteServer with ExpSite 1 now running...
Disconnected from ActorExpSite 1
SimAppSiteServer with ExpSite 1 shutdown
OpenFresco >

```

Figure 14: OpenFresco Simulation Application Server Window for Distributed Test after Simulation.

8 Results

The MATLAB® command window should display the following results:

$T_1 = 0.622 \text{ sec}$

$T_2 = 0.315 \text{ sec}$

The response quantities for the explicit Newmark time integration method are plotted in Figures 15 and 16.

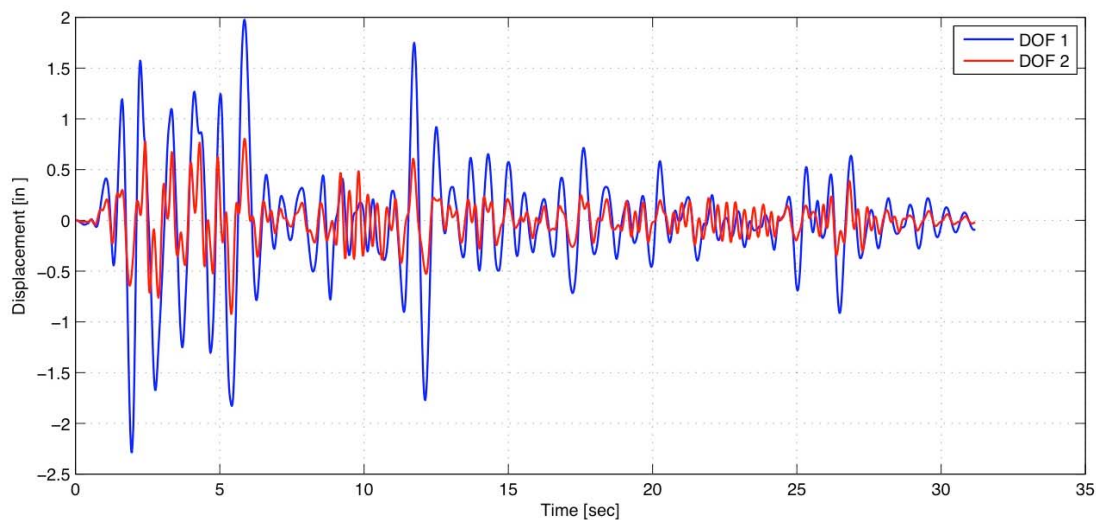


Figure 15: Displacements vs. Time for One-Bay Frame Example, Explicit Newmark Integration.



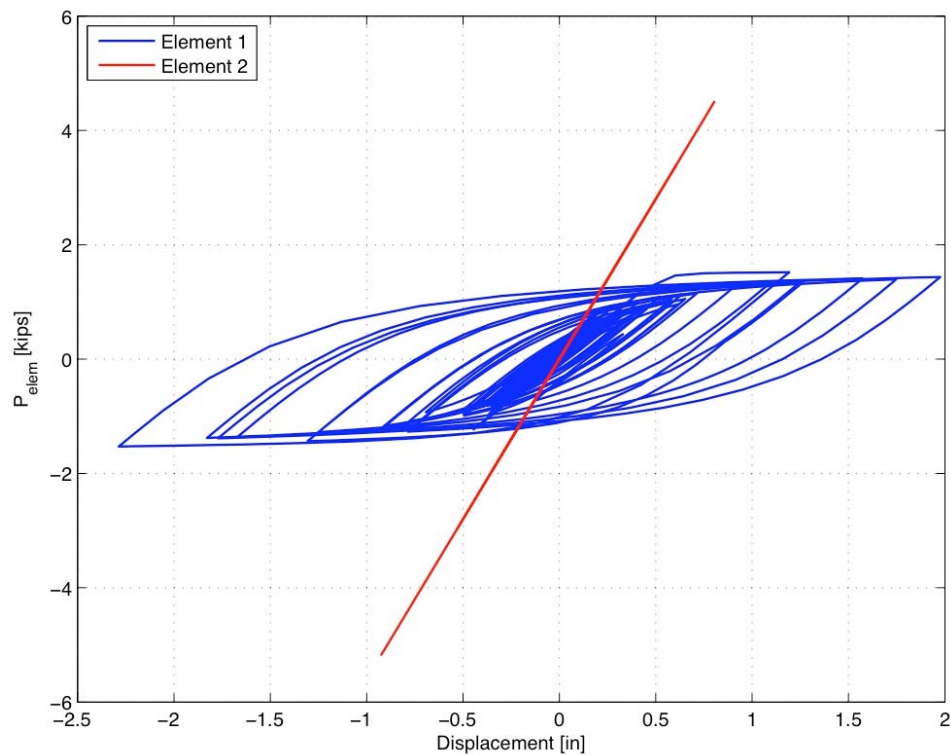


Figure 16: Element Hysteresis Loops for One-Bay Frame Example, Explicit Newmark Integration.

The response quantities for the Alpha-OS time integration method are plotted in Figures 17 and 18.

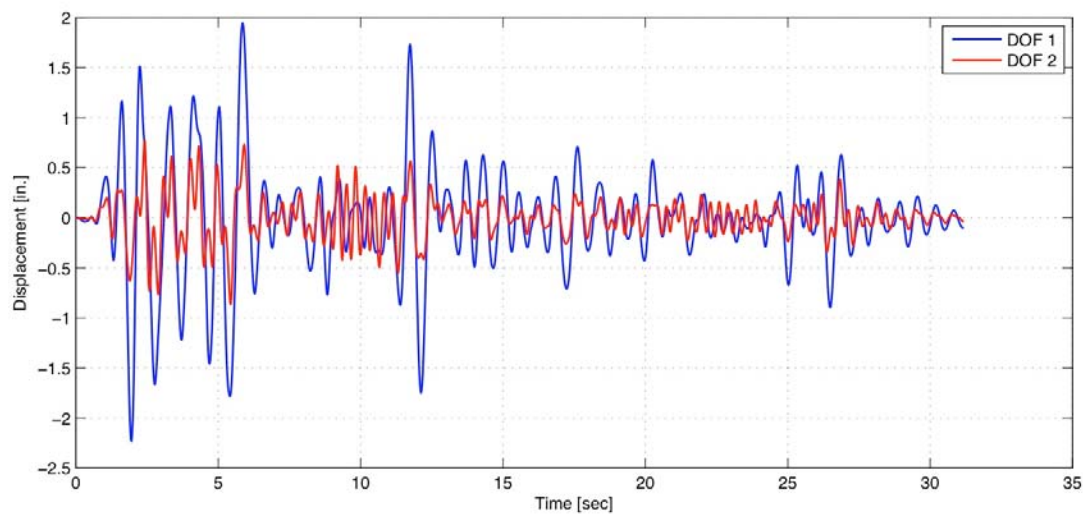


Figure 17: Displacements vs. Time for One-Bay Frame Example, Alpha-OS Integration.



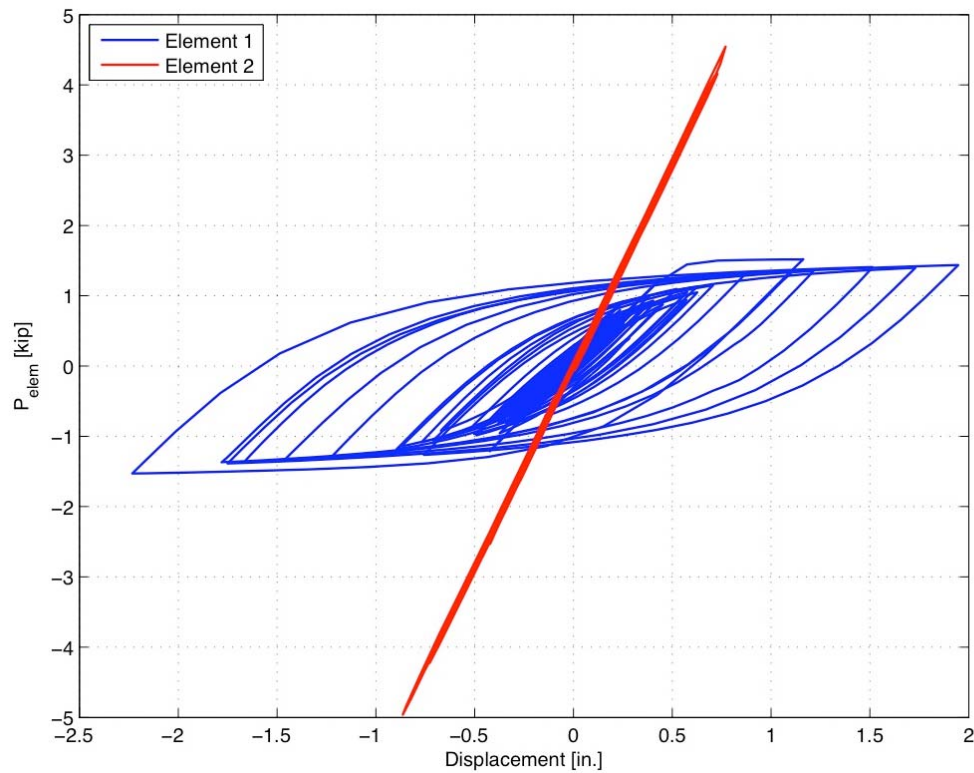


Figure 18: Element Hysteresis Loops for One-Bay Frame Example, Alpha-OS Integration.

9 References

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

