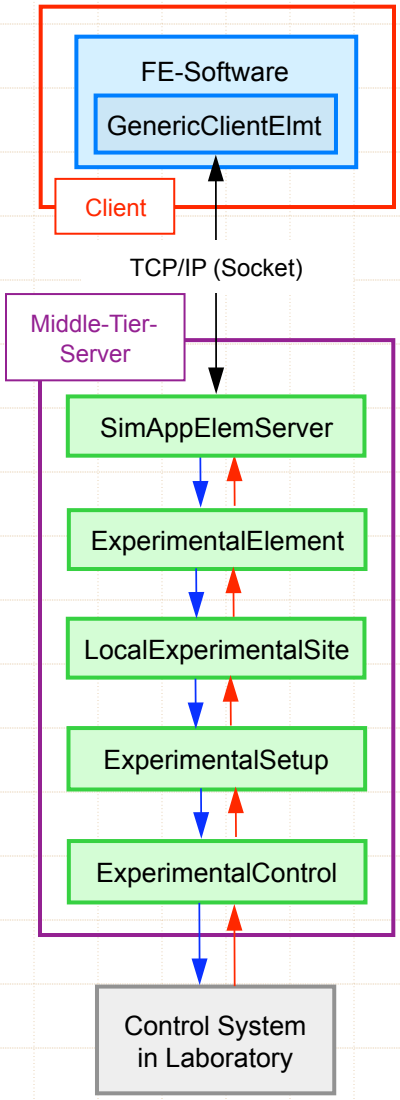
Interfacing FE-Software: Generic Client Element and Adapter Element

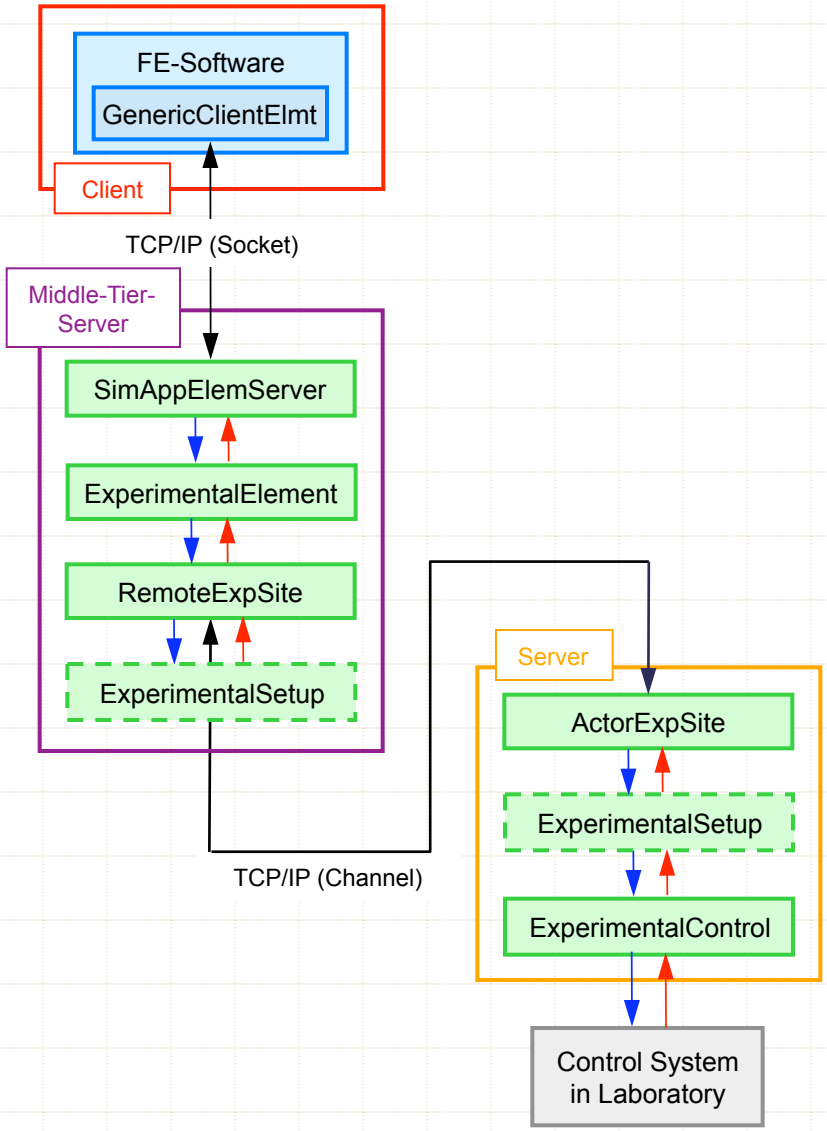
Andreas Schellenberg, Hong K. Kim, Yuli Huang, Yoshikazu Takahashi, Gregory L. Fenves, and Stephen A. Mahin

Department of Civil and Environmental Engineering
University of California, Berkeley

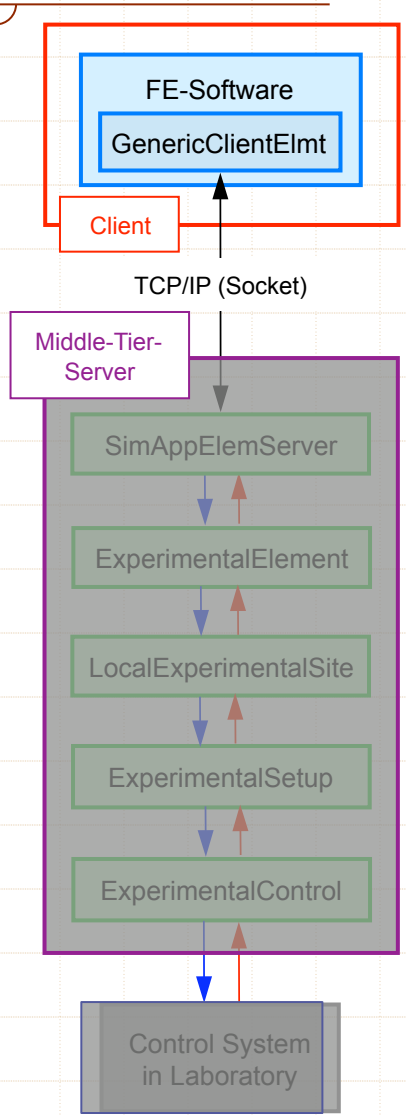
OpenFresco

OpenFresco Architecture



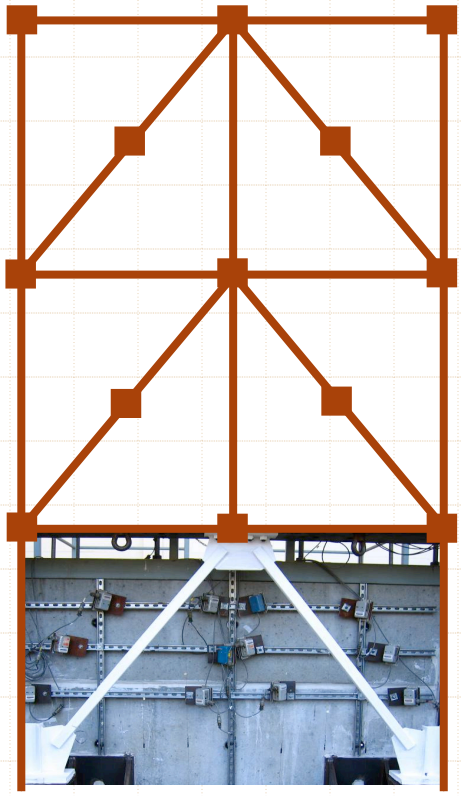


Generic Client Element (1)



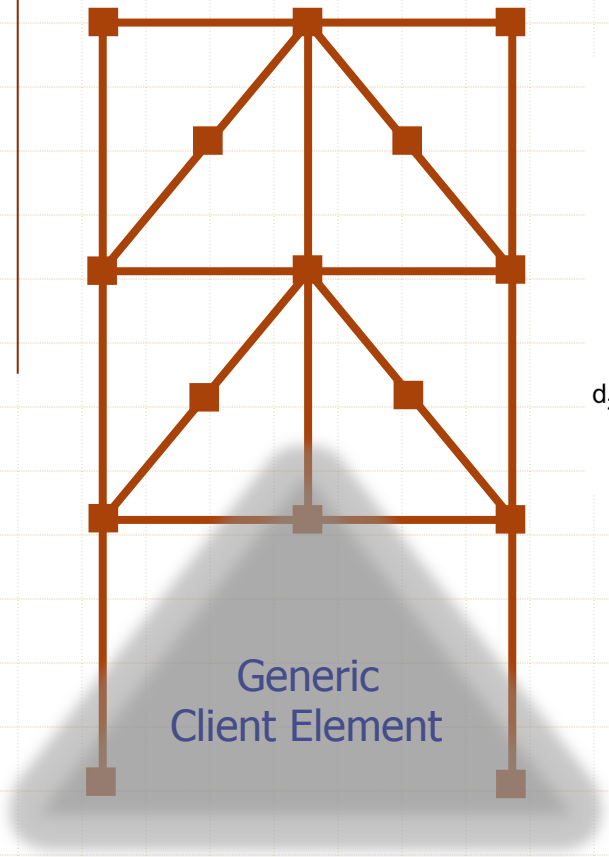
- FE-Software must have user-defined elements
- Generic Client Element is an interface to OpenFresco
- Has arbitrary number of nodes and degrees of freedom
- Uses TCP sockets for communication
- Makes use of Experimental Elements already in OpenFresco
- Generic Client Element is programmed once for a specific FE-Software

Generic Client Element (2)

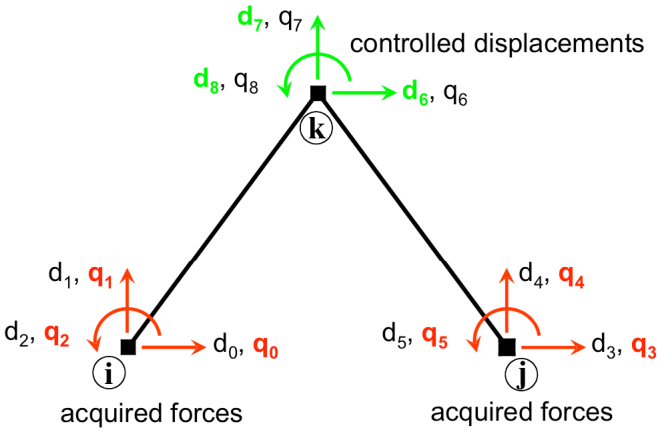


- analytical model of structural energy dissipation and inertia
- physical model of structural resistance

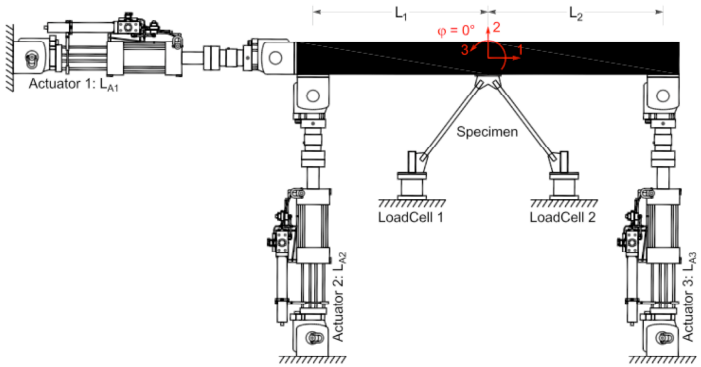
Generic Client Element (3)



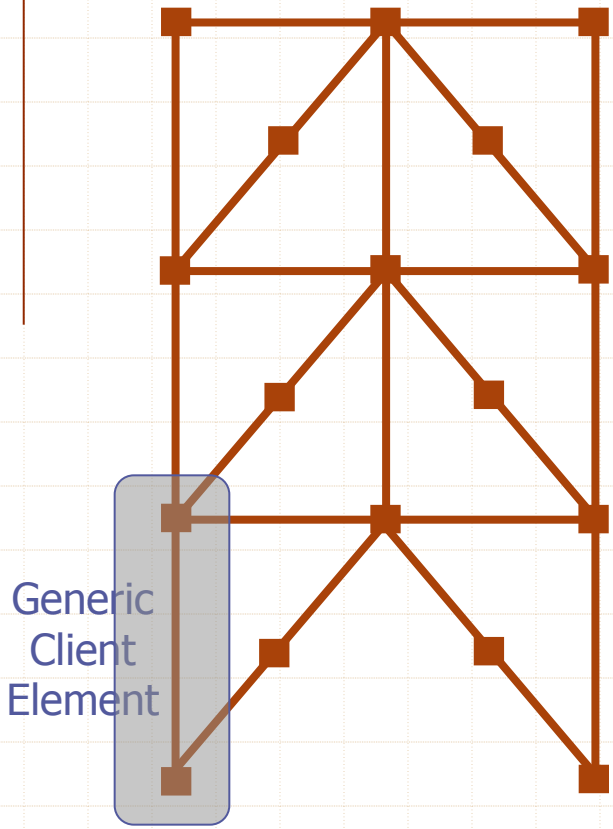
OpenFresco Experimental Element: invertedVBrace (2D)



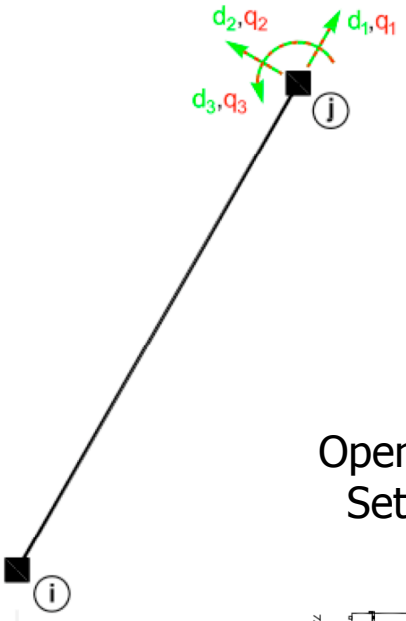
OpenFresco Experimental Setup: invertedVBrace



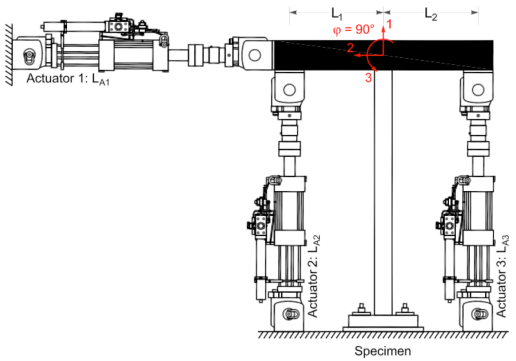
Generic Client Element (4)



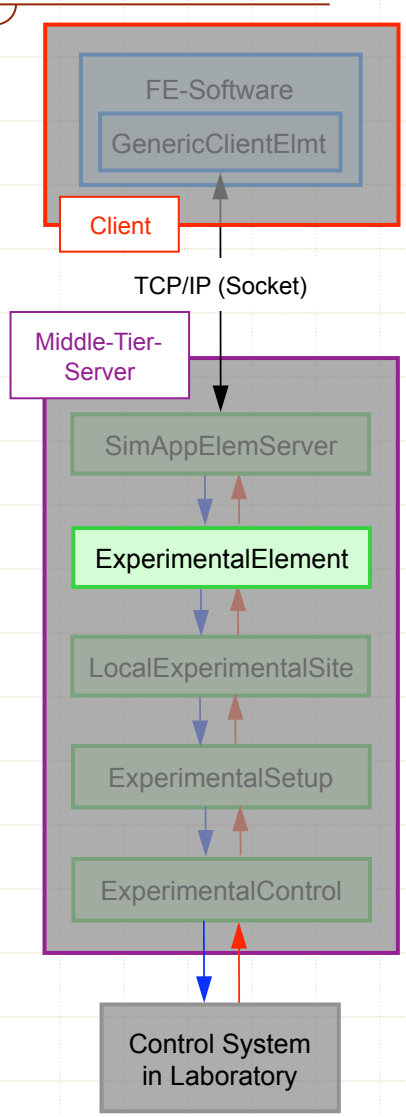
OpenFresco Experimental
Element: beamColumn (2D or 3D)



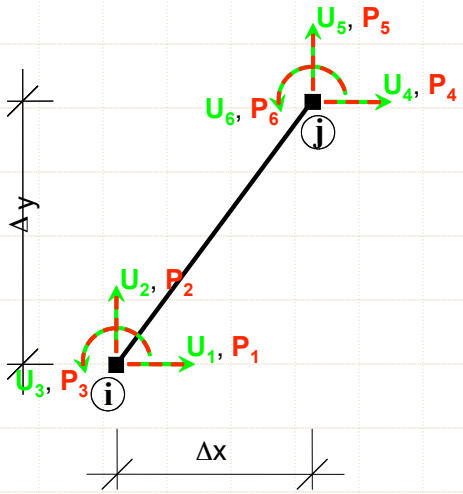
OpenFresco Experimental
Setup: ThreeActuators



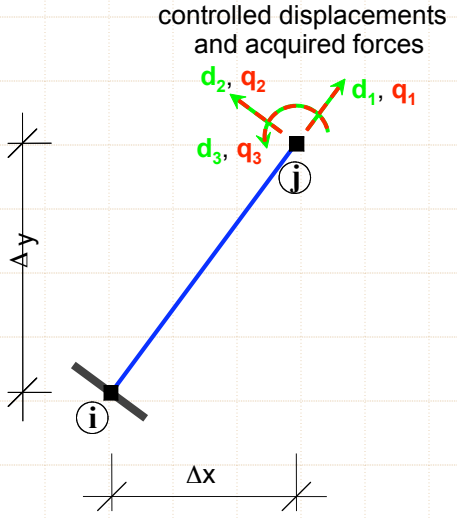
Experimental Element



- Transforms between global element DoF in FE-Software and basic element DoF in experimental element
- Experimental Element Tcl commands are in the Command Language Manual
- Recorder command available for each element



Global System



Cantilever Basic System

OpenFresco Experimental Elements

- beamColumn (2D or 3D)
- generic (1D, 2D, or 3D)
- invertedVBrace (2D)
- truss (1D, 2D, or 3D)
- corotTruss (1D, 2D, or 3D)
- twoNodeLink (1D, 2D, or 3D)

LS-DYNA genericClient_1d Element (1)

- NEESforge website: OpenFresco LS-Dyna Example Manual
- /trunk/SRC/simApplicationClient/lsDyna
  - Trial Displacements Being sent

```
c ...   send trial response to experimental site
c
        sData(1) = 3
        do 5, j = 1,4
          sData(1+j) = hsv(i,2+j)
    5   continue
c
        dataTypeSize = sizeDouble
        nleft        = sizeSendData
        call senddata(socketID, dataTypeSize, sData, nleft, stat)
```

LS-DYNA genericClient_1d Element (2)

- Get Measured Force

```
c ...   get measured resisting forces
c
        sData(1) = 10
        dataTypeSize = sizeDouble
        nleft        = sizeSendData
        call senddata(socketID, dataTypeSize, sData, nleft,
stat)
c
        dataTypeSize = sizeDouble
        nleft        = 4
        call recvdata(socketID, dataTypeSize, r, nleft, stat)
c
        do j=1,2
           force(i,j)   = r(j)
           force(i,j+6) = r(j+2)
        enddo
```

- socket.c using socketf.c

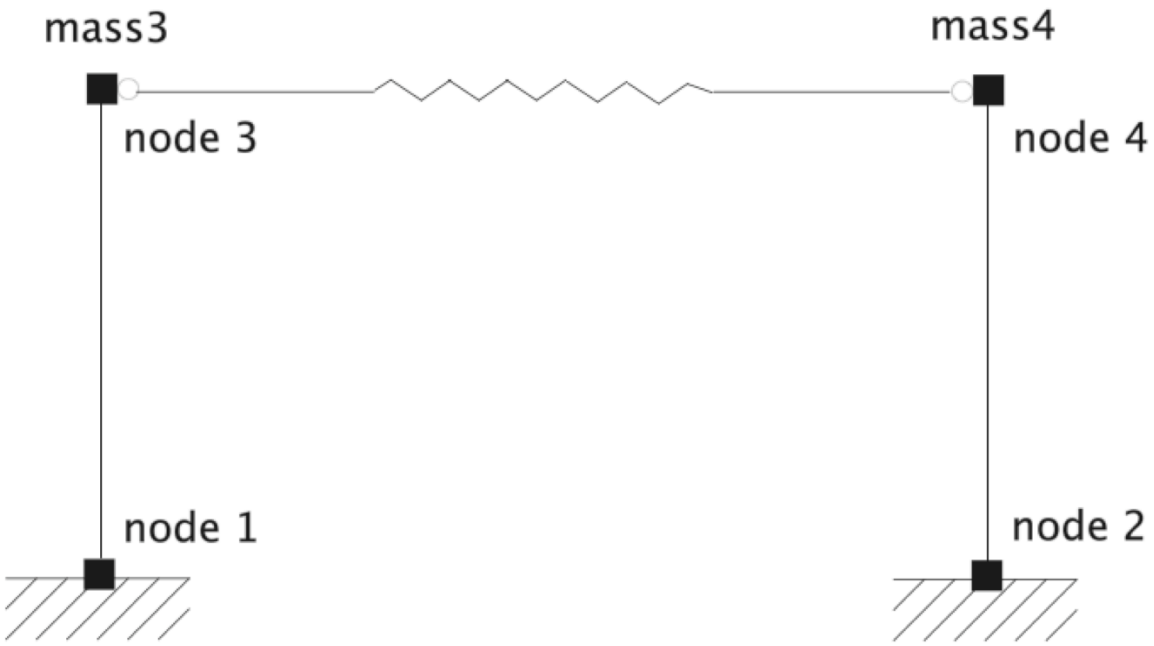
sData(1) actions (FrescoGlobals.h)

- /trunk/SRC/

  - 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.
  - 7 = get displacement.
  - 8 = get velocity.
  - 9 = get acceleration.
  - 10 = get force.
  - 11 = get time.
  - 12 = get initial stiffness.
  - 13 = get tangent stiffness.
  - 14 = get damping values.
  - 15 = get mass values.
  - 99 = end server process.

LS-DYNA OneBayFrame Example

- /trunk/EXAMPLES/OneBayFrame/LSDyna
  - OneBayFrame_LSDyna.k
- Structural Model



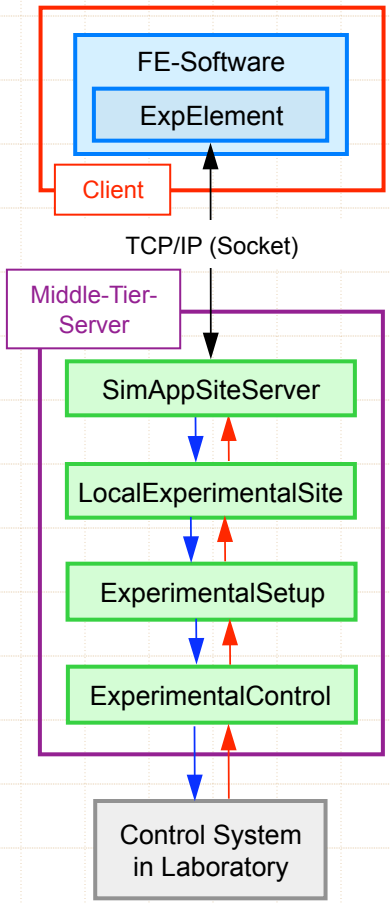
OpenSees genericClient Element

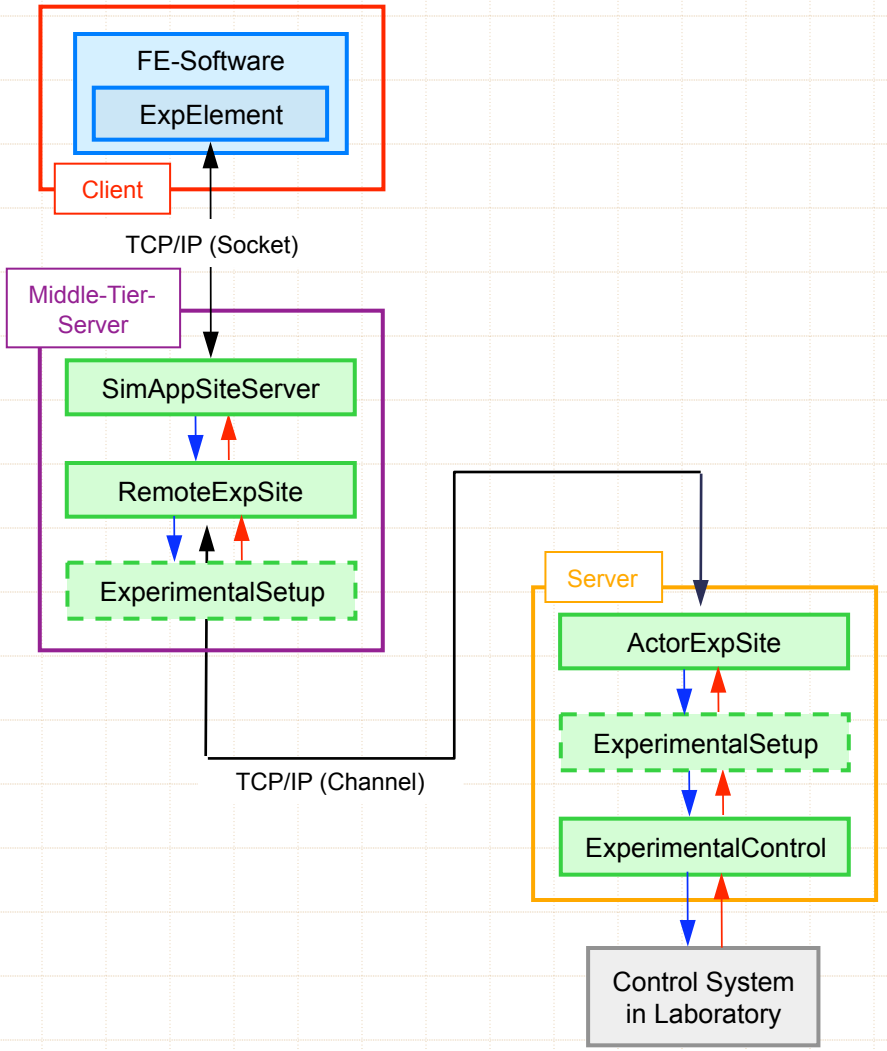
- Included in OpenSees 2.1.1
- Tcl command:

element genericClient $tag –node $Ndi $Ndj…
–dof $dofNdi … –dof $dofNdj … –server
$ipPort <$ipAddr> <-ssl> <-datasize $size>

- /trunk/EXAMPLES/OneBayFrame
  - Local Test Files: OneBayFrame_Local_Client.tcl and OneBayFrame_Local_SimAppServer.tcl
  - Distributed Test Files: OneBayFrame_Distr_Client.tcl, OneBayFrame_Distr_SimAppServer.tcl, and OneBayFrame_Distr_LabServer.tcl

Experimental Element Directly in FE-Software



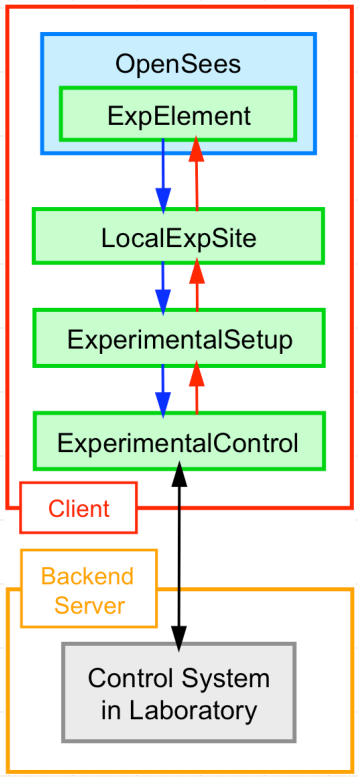


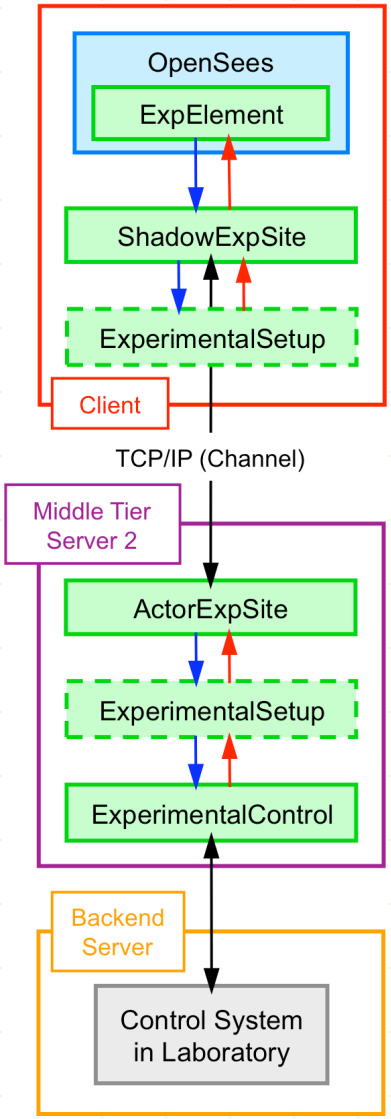
Matlab Experimental Element

- OpenFresco Matlab Example Manual
- /trunk/EXAMPLES/OneBayFrame/Matlab
- Experimental.m, 1D element, uses a TCPSocket.mex file

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

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

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

Experimental Element In OpenSees





OpenSees OneBayFrame Example

- /trunk/EXAMPLES/OneBayFrame
- Local Test File: OneBayFrame_Local.tcl
- Distributed Test Files: OneBayFrame Client1.tcl, OneBayFrame_Server1a.tcl (Left Column), and OneBayFrame_Server1b.tcl (Right Column)

Abaqus genericClient_exp Element (1)

- NEESforge website: OpenFresco Abaqus Example Manual (available soon)
- /trunk/SRC/simApplicationClient/abaqus
  - build instructions.txt
  - genericClient_exp.for
  - genericClient_exp.obj
  - vaba_param.inc
- /trunk/SRC/simApplicationClient/c
  - socket.c
- /trunk/SRC/simApplicationClient/fortran
  - socketf.c

Abaqus genericClient_exp Element (2)

```
                if (time(iTotalTime) .gt. 0.0) then
                   if (nint(svars(kblock,1)) .eq. 0) then
                      port = jprops(1)
                      sizeMachineInet = 9+1
                      call setupconnectionclient(port,
     * * *                                       '127.0.0.1'//char(0),
     * * *                                       sizeMachineInet,
     * * *                                       socketID)
                      if (socketID .le. 0) then
                         write(*,*) 'ERROR - failed to setup connection'
                         call xplb_exit
                      endif
                      svars(kblock,1) = socketID

c                     set the data size for the experimental element
c                     sizeCtrl(disp)
                      iData(1)  = ndofel
c                     sizeCtrl(vel)
                      iData(2)  = ndofel
c                     sizeCtrl(accel)
                      iData(3)  = ndofel
c                     sizeCtrl(force)
                      iData(4)  = 0
c                     sizeCtrl(time)
                      iData(5)  = 1
c                     sizeDaq(disp)
                      iData(6)  = 0
c                     sizeDaq(vel)
                      iData(7)  = 0
c                     sizeDaq(accel)
                      iData(8)  = 0
c                     sizeDaq(force)
                      iData(9)  = ndofel
c                     sizeDaq(time)
                      iData(10) = 0
c                     dataSize
                      iData(11) = dataSize

                      call senddata(socketID, sizeInt,
     *                              iData, 11, stat)
                   else
                      socketID = nint(svars(kblock,1))
                   endif
```

Abaqus genericClient_exp Element (3)

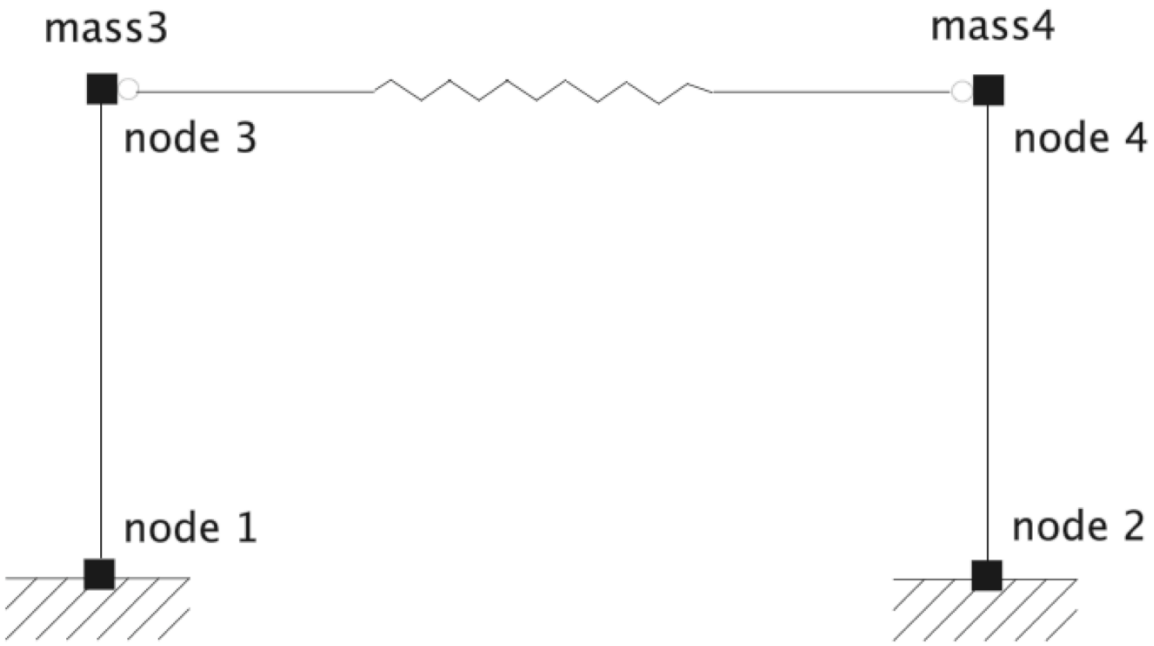
```
c                  commit state
                   if (time(iTotalTime) .gt. svars(kblock,2)) then
                      sData(1) = 5
                      call senddata(socketID, sizeDouble,
     *                                 sData, dataSize, stat)
                      svars(kblock,2) = time(iTotalTime)
                   endif
```

```
c                  send trial response to experimental element
                   sData(1) = 3
                   do i = 1, ndofel
                      sData(1+i) = u(kblock,i)
                      sData(1+ndofel+i) = v(kblock,i)
                      sData(1+2*ndofel+i) = a(kblock,i)
                   enddo
                   sData(1+3*ndofel+1) = time(iTotalTime)

                   call senddata(socketID, sizeDouble,
     *                           sData, dataSize, stat)
```

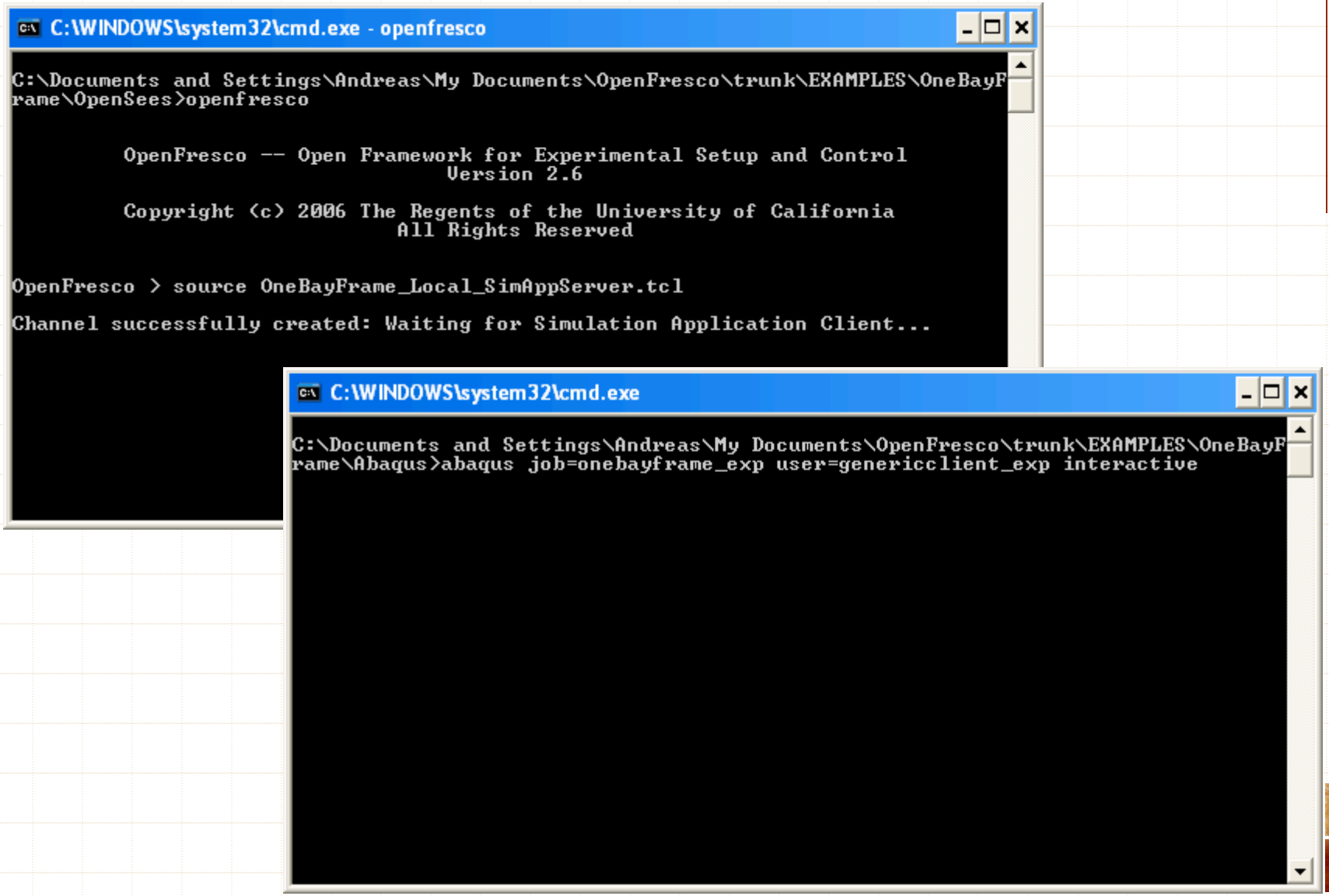
```
c                  get measured resisting forces
                   sData(1) = 10
                   call senddata(socketID, sizeDouble,
     *                           sData, dataSize, stat)

                   call recvdata(socketID, sizeDouble,
     *                           rData, dataSize, stat)

                   do i = 1, ndofel
                      rhs(kblock,i) = rData(i)
                   enddo
```

Abaqus OneBayFrame Example (1)

- /trunk/EXAMPLES/OneBayFrame/Abaqus
  - OneBayFrame_Exp.inp
- Structural Model

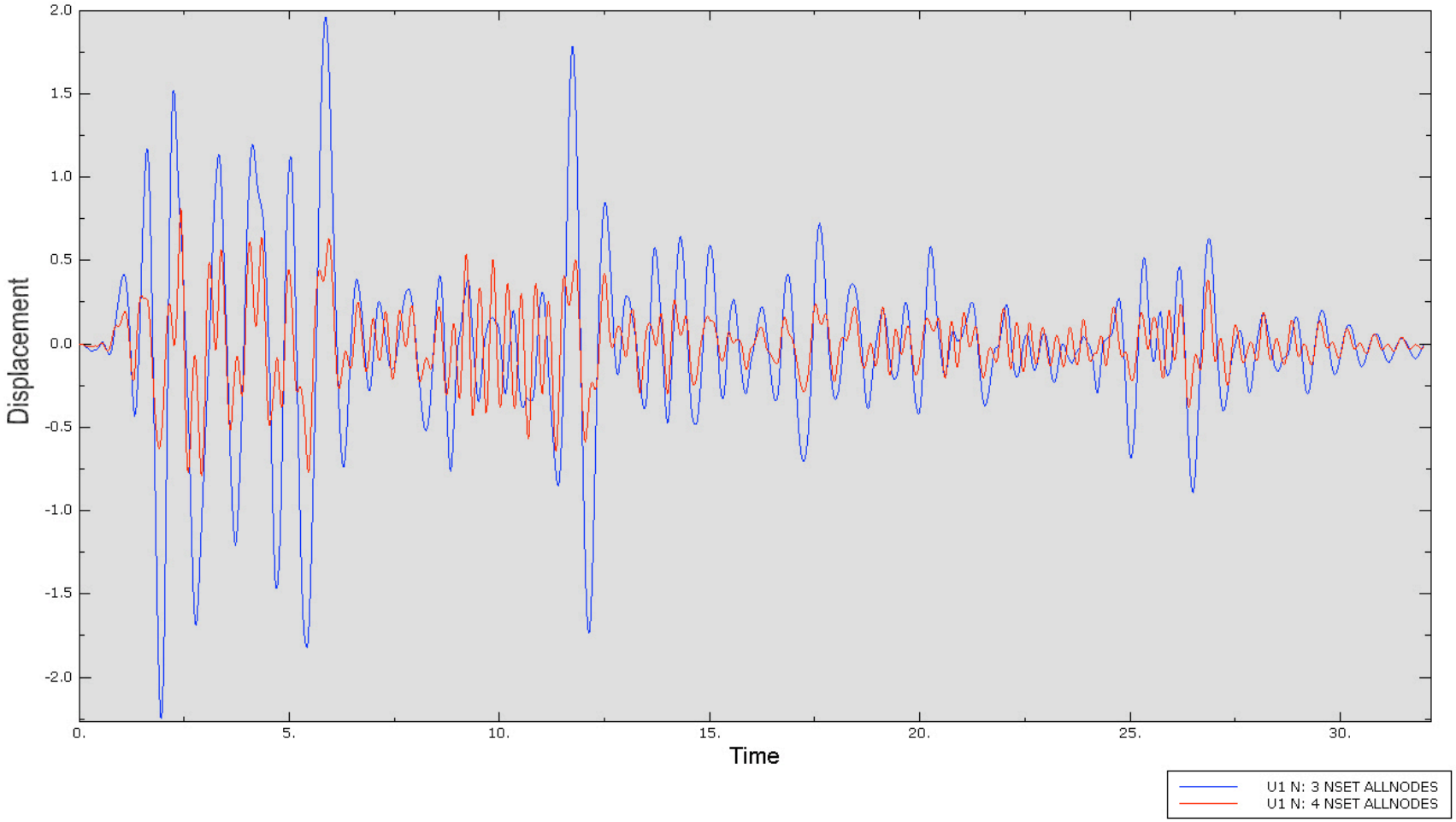


Abaqus OneBayFrame Example (2)

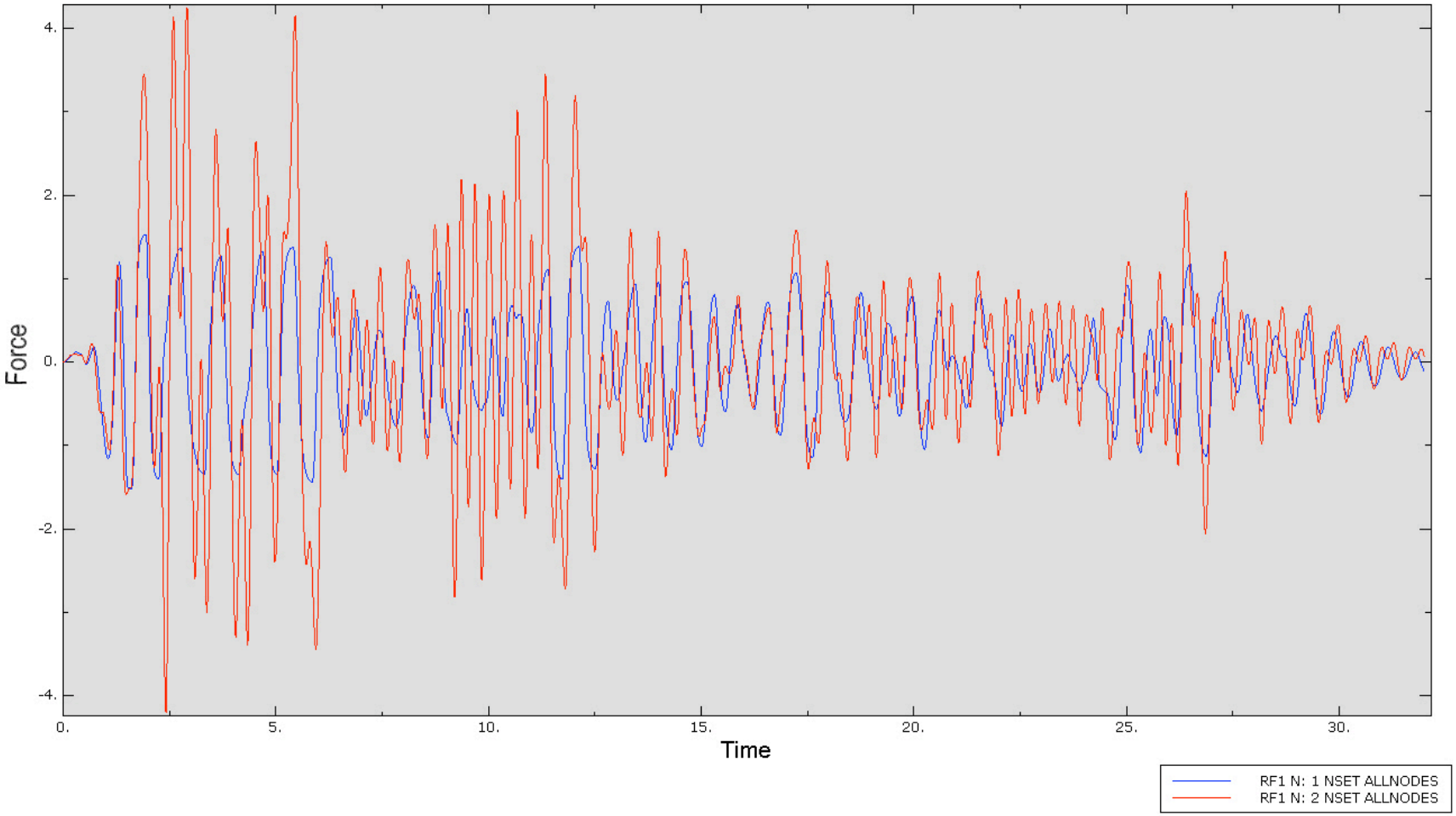


Abaqus OneBayFrame Example (3)

Abaqus OneBayFrame Example (4)



Abaqus OneBayFrame Example (5)



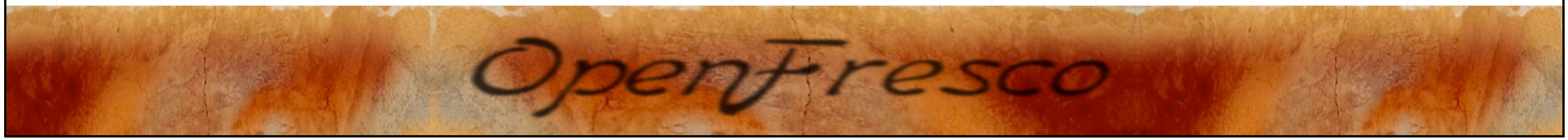
Conclusion

- Two Ways to Interface FE-Software
  - Generic Client Element
  - Experimental Element Directly in FE-Software
- Generic Client Element to be Programmed by the Developers
- OpenFresco works with Matlab/Simulink, LS-Dyna, Abaqus, OpenSees, UI SimCor

Structural FE-Software Coupling through the Experimental Software Framework, OpenFresco

Andreas Schellenberg, Yuli Huang and Stephen A. Mahin

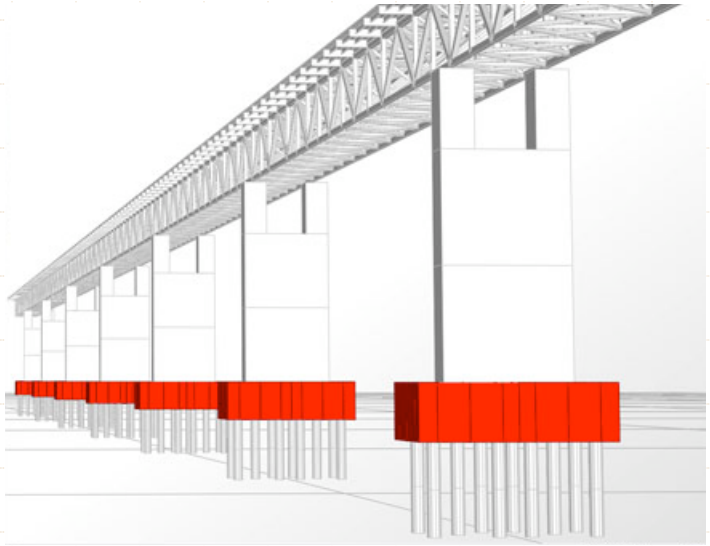
Department of Civil and Environmental Engineering
University of California, Berkeley

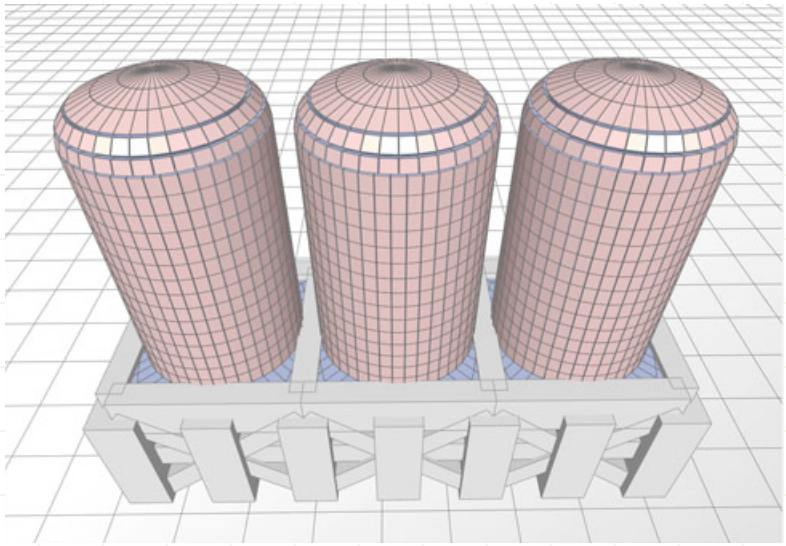


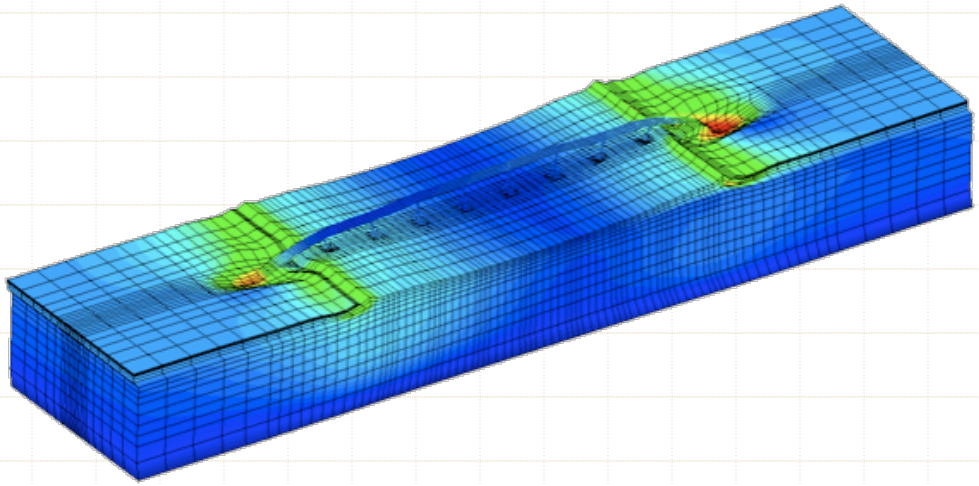
Motivation

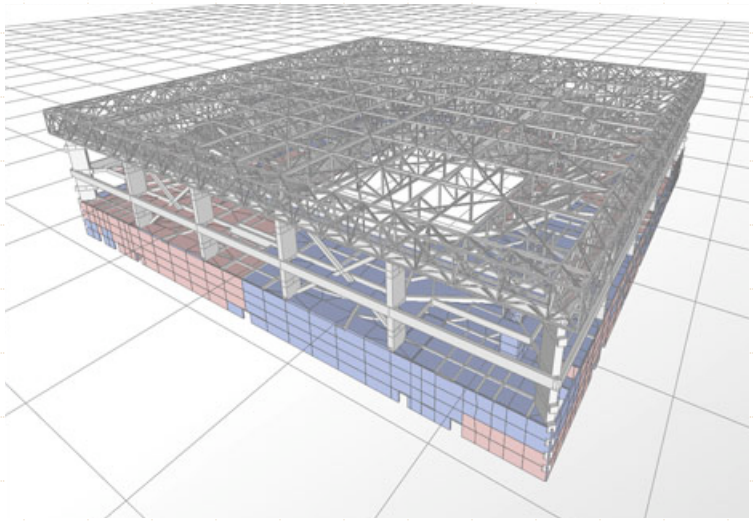
- Model and simulate entire systems and not just isolated structures or components
- FE-software packages are often highly specialized for some areas, but might lack necessary features in others
- Thus, couple specialized software packages together to take advantage of unique modeling and analysis capabilities
- This provides more flexibility and greater realism in simulating large systems

Motivation Cont.









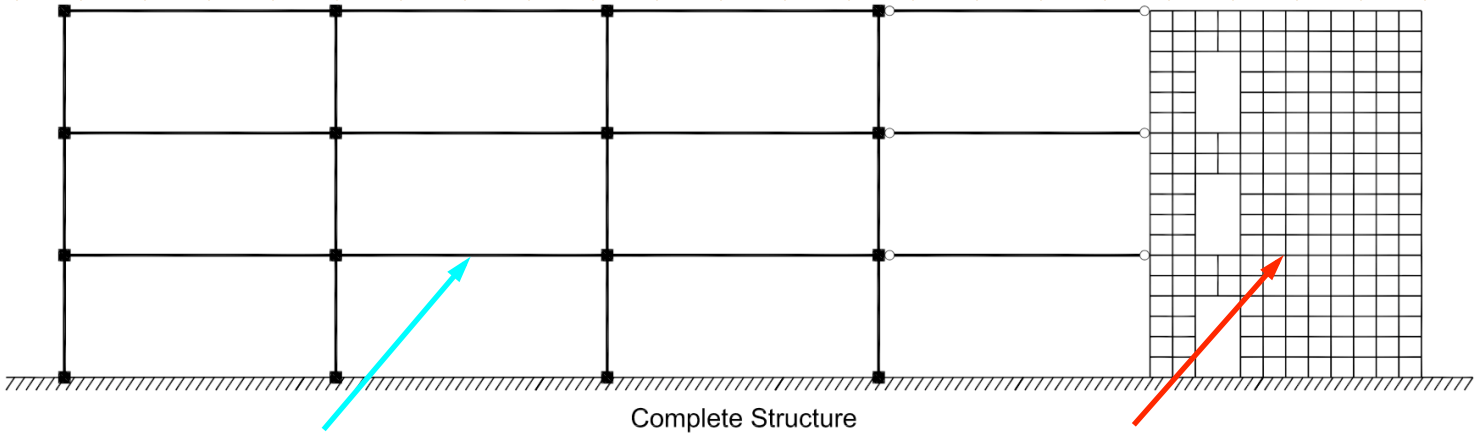
Outline of Presentation

1. Software Coupling Concept
2. Adapter Element Theory
3. Implementation Details
4. Three-Story Five-Bay Frame Example
5. Summary and Conclusions

Concept

- Select one program to act as the master solving the complete system
- Select remaining programs to act as slaves modeling different subassemblies
  - Subassemblies act as super-elements
  - Connected to master via interface DOFs
- Generic adapter elements provide interfaces in slave programs
- Generic super-elements provide interfaces in master program

Concept Cont.

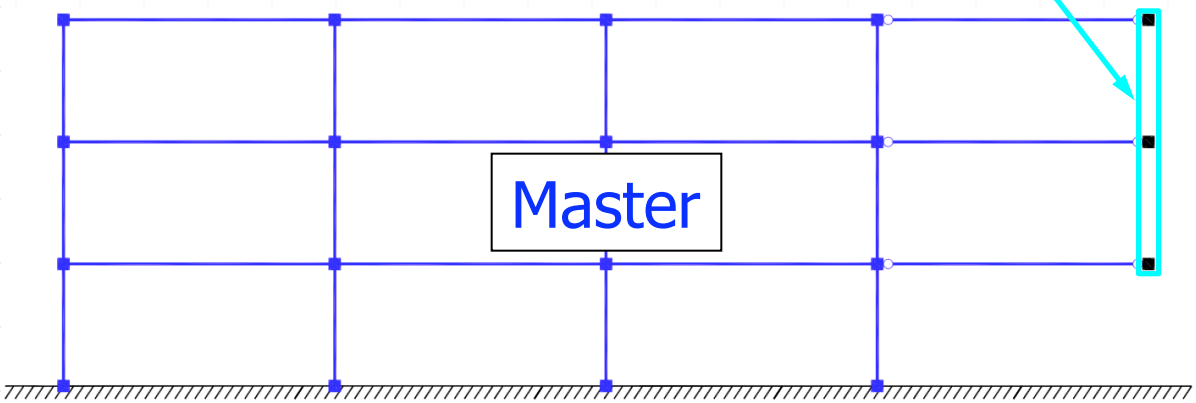


FE-Software 1: Generic Super-Element:
- frame element
- steel materials

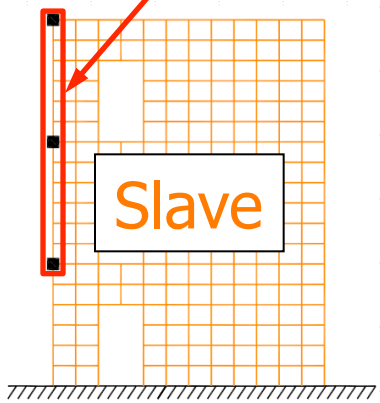
FE-Software 2: Adapter-Element:
- shell element
- concrete materials

- trial displacements
- resulting forces

- impose displ.
- acquire forces

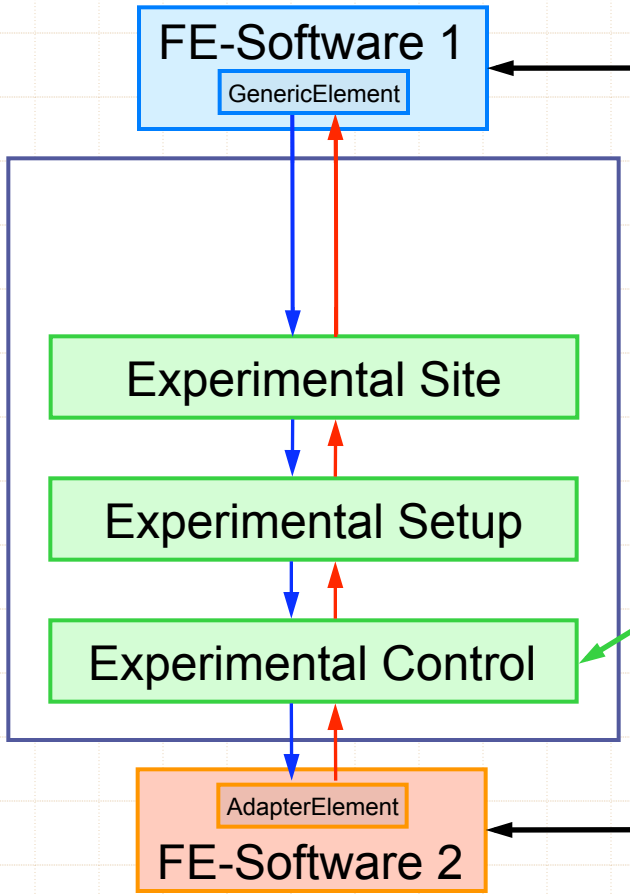


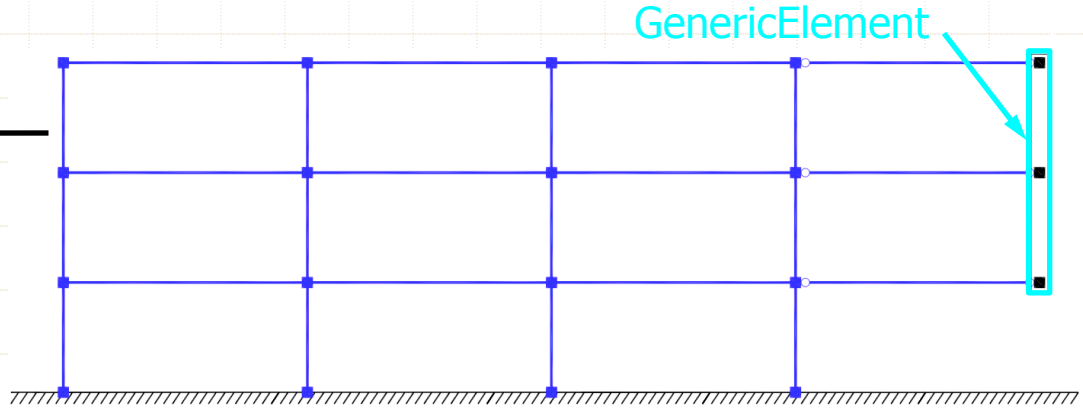
Moment Resisting Frame Subassembly
in Master FE-Software



Shear Wall Subassembly
in Slave FE-Software

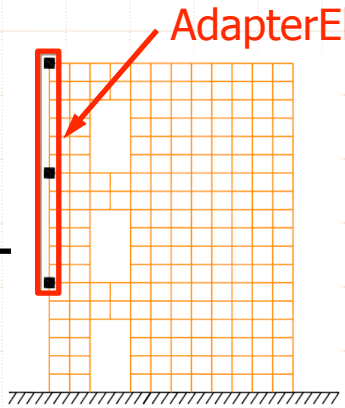
Middleware: OpenFresco





Moment Resisting Frame Subassembly
in Master FE-Software

ECSimFEAdapter (new experimental control that talks to Adapter element)



Shear Wall Subassembly
in Slave FE-Software

Advantages

- No file system is utilized to exchange data among the different software packages
- Runs continuously and concurrently
- No program-specific modifications need to be made to achieve the coupling
- This means that no access to the entire source-code of a FE-software is required
- A published programming interface (e.g. user-defined elements) is the only requirement

Adapter Element Theory

$$\mathbf{M}\ddot{\mathbf{U}}(t)+\mathbf{P}_{\mathbf{r}}\left(\mathbf{U}(t),\dot{\mathbf{U}}(t)\right)=\mathbf{P}(t)-\mathbf{P}_{\mathbf{0}}(t)$$

where:

$$\mathbf{M}=\mathbf{M}_{\mathbf{nd}}\ddot{\mathbf{U}}(t)+\mathop{\mathbf{A}}_{el}\mathbf{m}_{el}\ddot{\mathbf{u}}_{el}(t)$$

$$\mathbf{P}_{\mathbf{r}}=\mathop{\mathbf{A}}_{el}\mathbf{p}_{\mathbf{r},el}\left(\mathbf{u}_{el}(t),\dot{\mathbf{u}}_{el}(t)\right)$$

$$\mathbf{P}_{\mathbf{0}}=\mathop{\mathbf{A}}_{el}\mathbf{p}_{\mathbf{0},el}(t)$$

$$\mathbf{m}_{el}=\int_{V_{el}}\mathbf{N}^{T}\rho\mathbf{N}\,dV$$

$$\mathbf{p}_{\mathbf{r},el}=\int_{V_{el}}\mathbf{B}^{T}\boldsymbol{\sigma}(\boldsymbol{\varepsilon})\,dV$$

$$\mathbf{p}_{\mathbf{0},el}=-\int_{V_{el}}\mathbf{N}^{T}\mathbf{b}\,dV-\int_{\partial V_{t,el}}\mathbf{N}^{T}\mathbf{t}\,dS-\int_{V_{el}}\mathbf{B}^{T}\mathbf{D}\boldsymbol{\varepsilon}_{\mathbf{0}}\,dV+\int_{V_{el}}\mathbf{B}^{T}\boldsymbol{\sigma}_{\mathbf{0}}\,dV$$

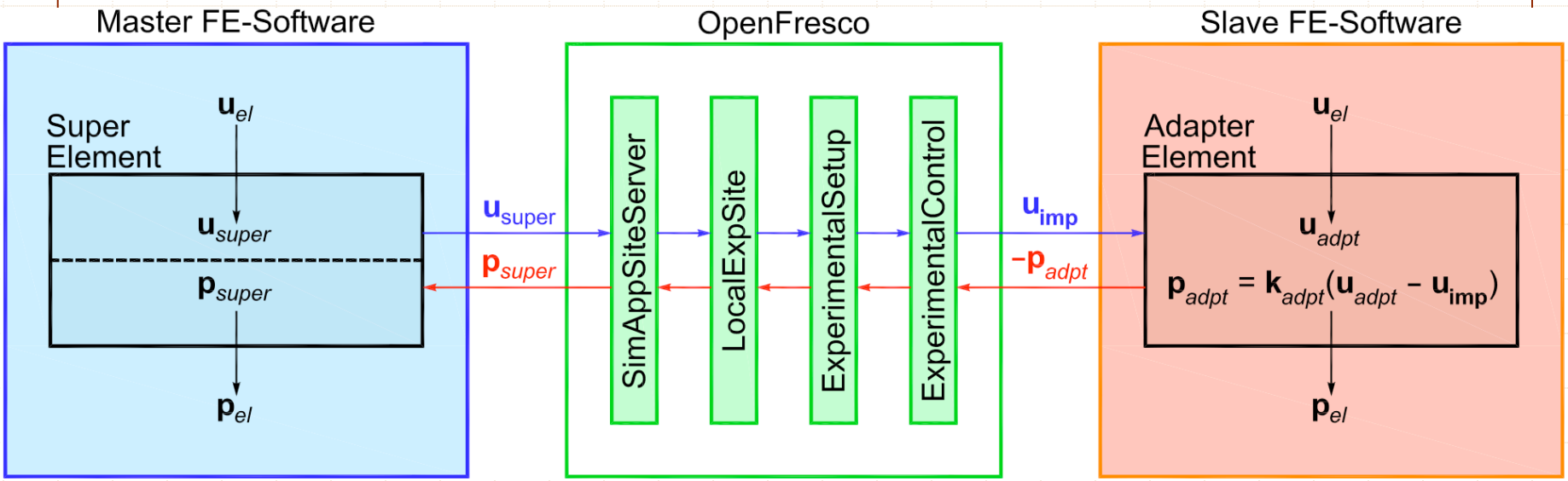
for AdpaterElement

$$\mathbf{p}_{\mathbf{0},adpt}=-\mathbf{k}_{adpt}\,\mathbf{u}_{\mathbf{imp}}(t)$$

Nodal force vector of AdapterElement:

$$\mathbf{p}_{adpt}=\mathbf{p}_{\mathbf{r},adpt}+\mathbf{p}_{\mathbf{0},adpt}=\mathbf{k}_{adpt}\left(\mathbf{u}_{adpt}(t)-\mathbf{u}_{\mathbf{imp}}(t)\right)$$

Implementation Details

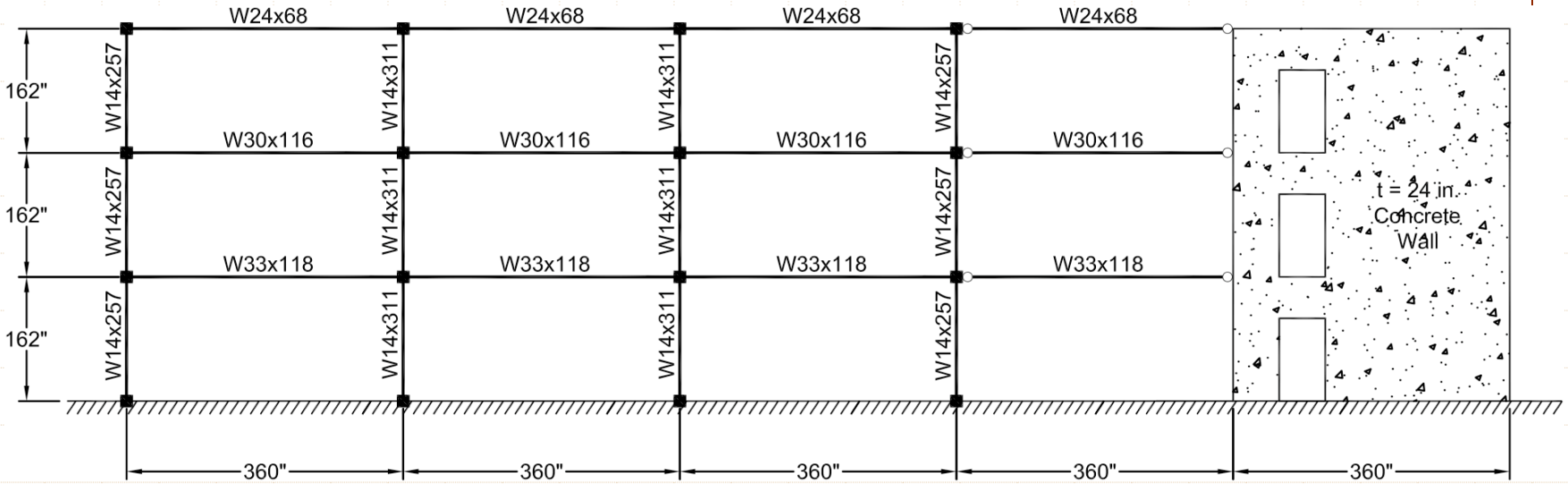


Both, the super-element in the master FE-software and the adapter element in the slave FE-software need to be once implemented as user-defined elements using the software packages' published programming interfaces

Implementation Details Cont.

- Integration methods in the master and slave programs need to be compatible
- If the behavior of the subassembly in the slave program is time-dependent:
  - Time in master and slave programs need to advance at the same rate
  - Thus, use special integration methods in master program
  - Utilize any integration method in slave program, as long as time step size is compatible with master program

3-Story 5-Bay Frame Example

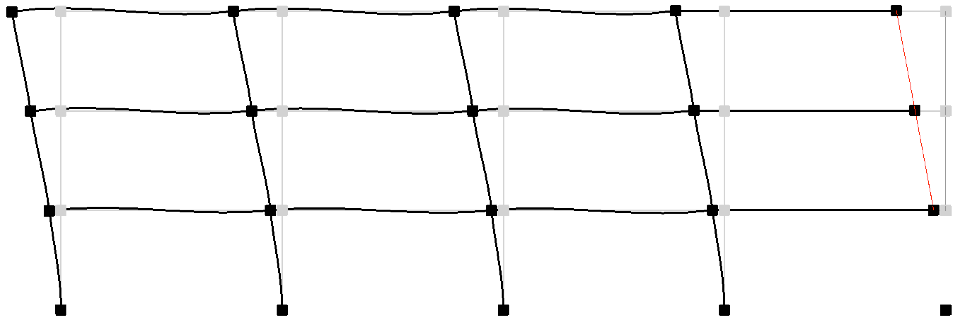


Properties of Model:

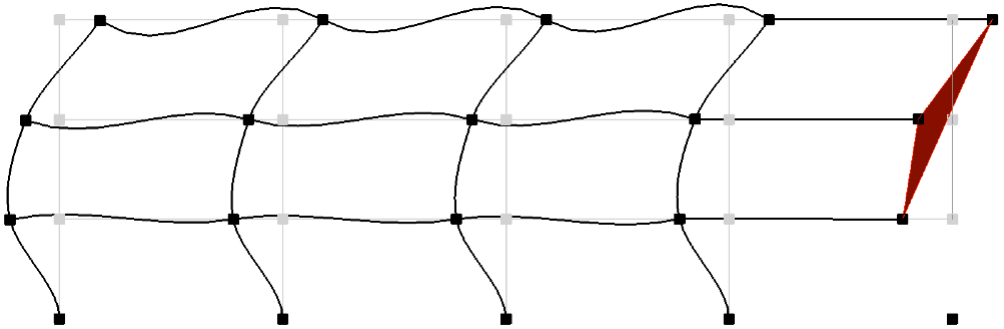
- NDOF = 27+414
- Damping: $\zeta_1 = 0.05$
- SACNF01: pga = 0.755g
- $k_{ii,adpt}$ = 1E12 kips/in.

$$\mathbf{k}_{super} = \begin{bmatrix} 93204.7 & -51063.1 & 10190.1 \\ -51063.1 & 82381.2 & -36235.3 \\ 10190.1 & -36235.3 & 23304.8 \end{bmatrix}$$

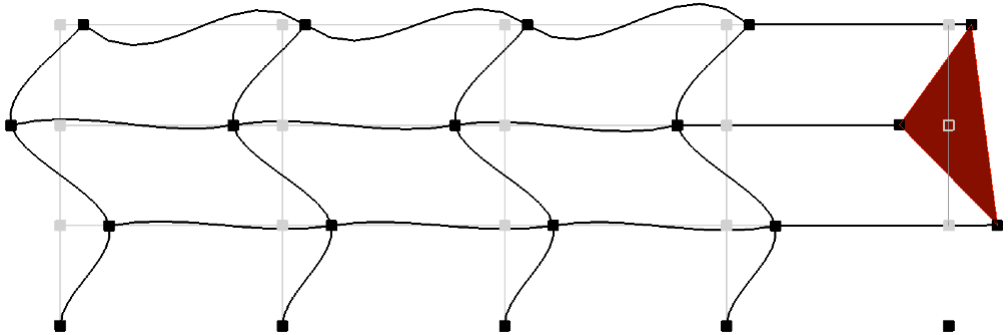
Mode Shapes



1st Mode:
T_1 = 0.312 sec

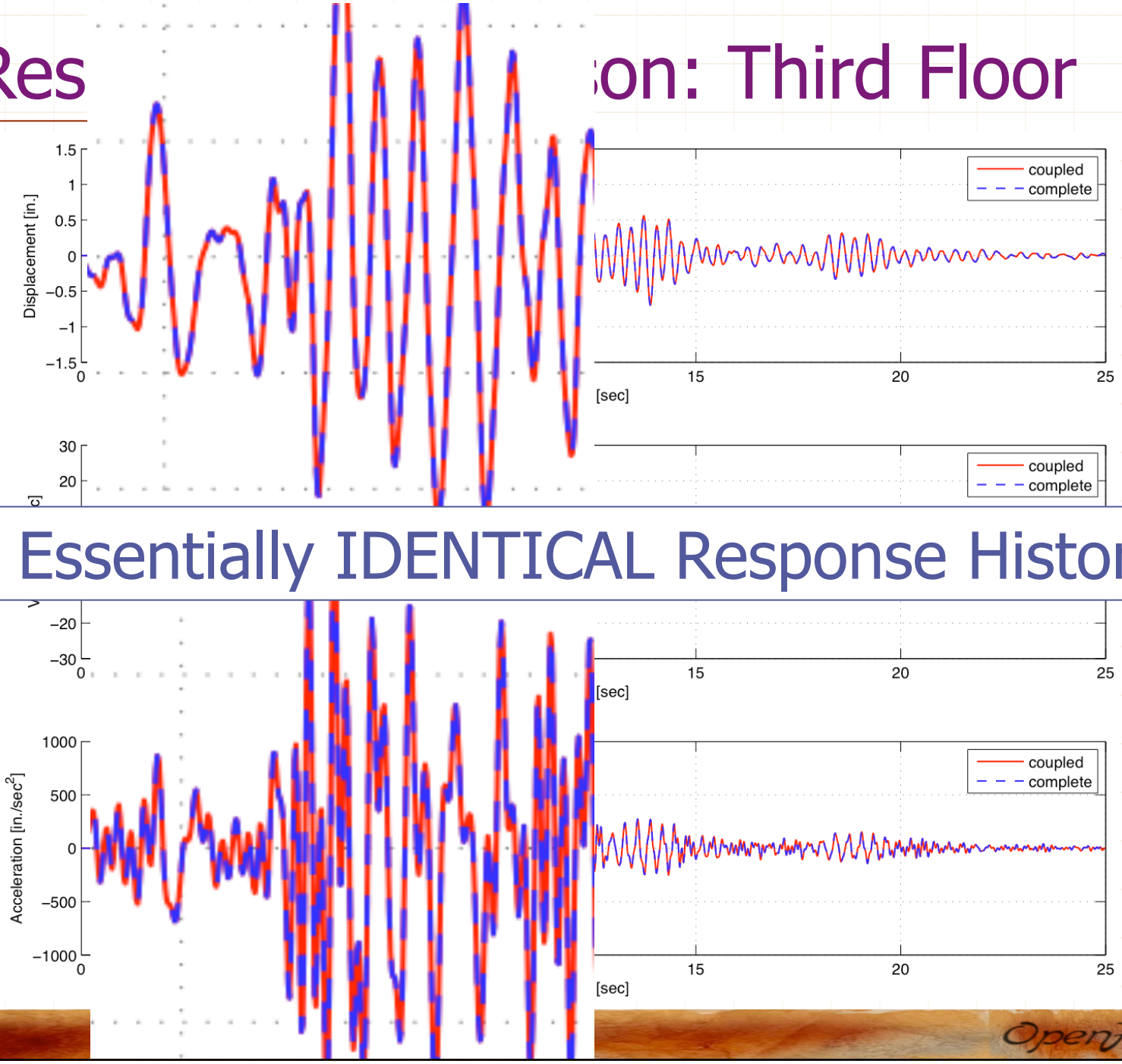


2nd Mode:
T_2 = 0.089 sec



3rd Mode:
T_3 = 0.051 sec

Res son: Third Floor

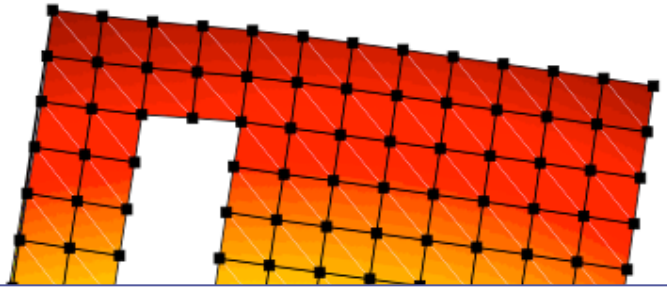


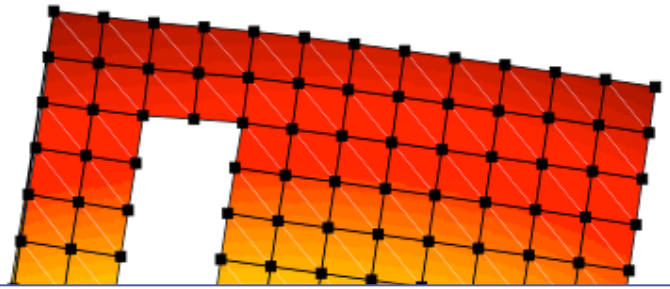
Essentially IDENTICAL Response Histories

Response Comparison: Wall Defo.

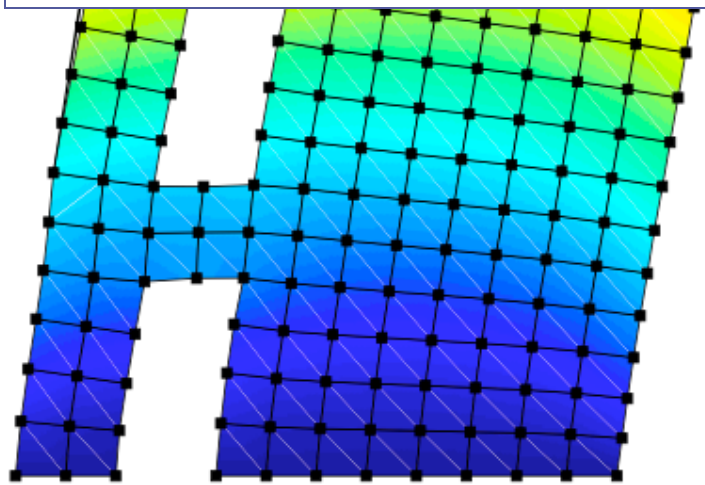
Coupled (at 6.33 sec)

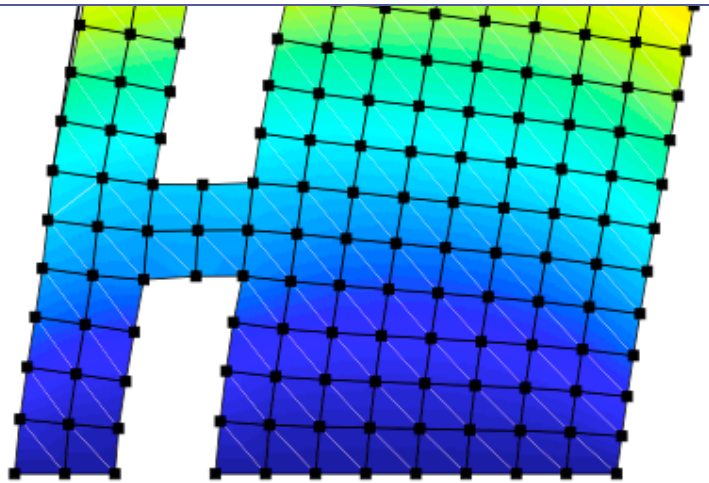
Complete (at 6.33 sec)





Essentially IDENTICAL Wall Deformations





50x magnified

50x magnified

Analysis Time Consumption

- Measure elapsed times for 2500-step-long transient analyses
- For coupled analysis this corresponded to 9990 network transactions across the TCP/IP connection
- Coupled simulation ~3-times slower

	complete simulation	coupled simulation
elapsed time	28.7 sec	84.4 sec
number of analysis steps	2500	2500/4995 (master/slave)

Conclusions

- Software coupling is important for large structural systems that require unique modeling capabilities of diff. programs
- The adapter element approach and the OpenFresco middleware provide a useful and effective set of modules for coupling structural analysis software
- The novel approach avoids the need to repetitively shutdown/restart programs and read/write data files

Questions?
Thank you!

http://openfresco.neesforge.nees.org

The development of OpenFresco has been sponsored in parts by the National Science Foundation through grants from the NEES Consortium, Inc.

Department of Civil and Environmental Engineering
University of California, Berkeley

OpenFresco