Conducting Hybrid Simulations with OpenSees/OpenFresco

Andreas Schellenberg

Pacific Earthquake Engineering Research Center, University of California, Berkeley
Outline of Presentation

1. Introduction to Hybrid Simulation
2. OpenFresco Architecture and TCL commands
3. Downloading and Installing OpenFresco
4. Building a Hybrid Model in OpenSees/OpenFresco
5. Simulated vs. Real Controllers
6. Using other Computational Drivers
7. Summary & Conclusions
Introduction to Hybrid Simulation

\[
M \ddot{u} + C \dot{u} + P_r(u) = P(t)
\]
## Comparison of Exp. Test Methods

<table>
<thead>
<tr>
<th></th>
<th>Quasi-Static</th>
<th>Shaking Table</th>
<th>Hybrid Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamics</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Strain Rate Effects</td>
<td>NO</td>
<td>YES</td>
<td>YES (if real-time test)</td>
</tr>
<tr>
<td>Large- or Full-Scale</td>
<td>YES (limited by table)</td>
<td>NO</td>
<td>YES</td>
</tr>
</tbody>
</table>

- First Hybrid Simulation (Online Test) in 1975 by Takanashi et al.
Hybrid Simulation

\[ M \cdot \ddot{u} + C \cdot \dot{u} + P_r(u) = P(t) \]

- Inertia
- Energy dissipation
- Resistance
Hybrid Simulation

\[ M \ddot{u} + C \dot{u} + P_r(u) = P(t) \]

Dynamic Loading:
- Seismic
- Wind
- Blast/Impact
- Wave
- Traffic

Static Loading:
- Gravity
- Prestress

Analytically add nonlinear geometric effects to measured resisting forces

- Analytical model of structural energy dissipation and inertia
- Physical model of structural resistance
Hybrid Simulation

- Model the well understood parts of a structure in a finite element program on one or more computers
- Leave the construction and testing of the highly nonlinear and/or numerically hard to model parts of the structure in one or more laboratories
- Can be considered as a conventional finite element analysis where physical models of some portions of the structure are embedded in the numerical model
Required Components

1. Discrete model of the structure to be analyzed, including the static and dynamic loading
2. Servo-hydraulic control system with static or dynamic actuators
3. Physical test specimen, including a reaction-frame
4. Data acquisition system with instrumentation
Testing Methods

- Conventional hybrid simulation test where specimen is loaded using a ramp-and-hold loading procedure
- Continuous test where specimen is loaded at a continuous slow to moderately slow rate to avoid load relaxations
- Real-time test where specimen is loaded at correct velocities to account for rate-dependent material behaviors
- Geographically distributed network test
Advantages

- Enables dynamic testing of full-scale specimens
- Quasi-static testing equipment sufficient
- Fewer restrictions on size, weight and strength of a specimen
Advantages

- Geometric nonlinearities, three-dimensional effects, multi-support excitations and soil-structure interactions can be incorporated into the analytical model.

- Internationally, geographically distributed testing is made possible.
Advantages

- **Quasi-static applications**
  - Drifts may concentrate in one story
  - Numerical model of portions of system that may alleviate weak story response

- **Dynamic Applications**
  - Shaking Table
  - Smart shaking table

- **Hydrodynamic effects**

- **Moving Loads**
  - Including impact
  - Interaction with moving loads, including impact
OpenFresco
Software Framework
What is OpenFresco?

- Open source Framework for Experimental Setup and Control
- Secure, object oriented, network enabled "middleware" -- Pairs computer analysis software with laboratory control systems and other software to enable hybrid and collaborative computing:

- **Computational Drivers**
  - OpenSees
  - OpenFresco Express
  - Abaqus
  - LS-DYNA
  - Matlab
  - Simulink
  - Ansys
  - UI-SimCor

- **Control Systems**
  - dSpace
  - MTS
    - STS family
    - Flextest/CSI
    - Flextest/SCRAMNet
  - National Instruments
  - Pacific Instruments
  - ADwin
Why a Software Framework?

- Lack of a common framework for development and deployment of HS
- Problem specific implementations which are site and control system dependant
- Such highly customized software implementations are difficult to adapt to different structural problems

Need a robust, transparent, adaptable, and easily extensible software framework for research and deployment
Rethinking implementation strategies

- Embed test specimen(s) in an existing computational framework of user’s choice

Typical features of an analysis framework

Proper numerical model uncertain
Rethinking implementation strategies

- Embed test specimen(s) in an existing computational framework of user’s choice

**Typical features of an analysis framework**

- Define element as an “Experimental Element”

**OpenFresco**

**Laboratory**
OpenFresco Components

FE-Software

provides all features of unmodified computational framework, including parallel and network computing

OpenFresco
(Middleware)

Control System in Laboratory

provides control of physical actuators as well as data acquisition using physical instrumentation devices
OpenFresco Components

FE-Software provides all features of unmodified computational framework, including parallel and network computing.

Experimental Element represents the part of the structure that is physically tested and provides the interface between the FE-software and the experimental software framework.

Experimental Site stores data and provides communication methods for distributed testing.

Experimental Setup transforms between the experimental element degrees of freedom and the actuator degrees of freedom (linear or non-linear transformations).

Experimental Control interfaces to the different control and data acquisition systems in the laboratories.

Control System in Laboratory provides control of physical actuators as well as data acquisition using physical instrumentation devices.
Requirements for Architecture

- Provide connectivity to a wide variety of FE-software (clients), independent of the language, such analysis software is programmed in
- Enable distributed testing and support different communication protocols
- Interface with rapidly evolving control and data acquisition systems deployed at testing facilities all over the world
OpenFresco Components

Client

TCP/IP (Socket)

Middle Tier Server

SimAppElemServer

ExperimentalElement

LocalExpSite

ExperimentalSetup

ExperimentalControl

Backend Server

Control System in Laboratory

OpenSees

ExpElement

LocalExpSite

ExperimentalSetup

ExperimentalControl

Backend Server

Control System in Laboratory

FE-Software

GenericClientElmt

FE-Software

ExpElement

FE-Software

ExpElement

local deployment
OpenFresco Components

FE-Software

Experimental Element

Transforms between the global element degrees of freedom in the FE-Software and the basic element degrees of freedom in the experimental element.

Consider element in structure

Two coordinate systems used in FE analysis.

Experimental Site

Experimental Setup

Experimental Control

Control System in Laboratory

Global System

Cantilever Basic System

d_1, q_1
d_2, q_2
d_3, q_3
OpenFresco Components

Experimental Site
Stores data and provides communication methods for distributed testing

LocalExpSite available for local testing and RemoteExpSite/ActorExpSite pair available for geographically distributed testing

Utilizes communication channels with TCP, TCP+SSL or UDP communication protocols
OpenFresco Components

**FE-Software**
- GenericElement

**Experimental Element**

**Experimental Site**

**Experimental Setup**

**Experimental Control**

**Control System in Laboratory**

---

**Experimental Setup**
Transforms between the basic experimental element degrees of freedom in OpenFresco and the actuator degrees of freedom in the laboratory (linear vs. non-linear transformations are available).

**Cantilever Basic System**

 controlled displacements and acquired forces

\[
T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -L_0 \\ 0 & 1 & L_1 \end{bmatrix}
\]

**Actuator Setup**
OpenFresco Components

Experimental Control
Interfaces to the different control and data acquisition systems in the laboratories (IP addresses and port numbers)
OpenFresco TCL-Commands
Experimental Elements

EEBeamColumn (2D,3D)

```
expElement beamColumn $eleTag $iNode $jNode $tranTag -site $siteTag -initStif $Kij <-iMod> <-rho $rho>
```

- `$eleTag` unique element tag
- `$iNode,$jNode` end nodes
- `$tranTag` tag of previously defined crd-transf object
- `$siteTag` tag of previously defined site object
- `$Kij` initial stiffness matrix elements (ndf x ndf)
- `-iMod` flag for I-Modification (optional, default=false)
- `$rho` mass per unit length (optional, default=0.0)
Experimental Sites

LocalExpSite

expSite LocalSite $tag $setupTag

$tag unique site tag
$setupTag tag of previously defined setup object
Experimental Setups

ESThreeActuators

expSetup ThreeActuators $tag <-control $ctrlTag>
$La1 $La2 $La3 $L1 $L2 <-nlGeom> <-posAct1 $pos>
<-phiLocX $phi> <-trialDispFact $f> ...
Experimental Controls

ECxPCtarget

```bash
expControl xPCtarget $tag $type ipAddr $ipPort
appName appPath
```

- `$tag` unique control tag
- `$type` predictor-corrector type
- `ipAddr` IP address of xPC Target
- `$ipPort` IP port of xPC Target
- `appName` name of Simulink application to be loaded
- `appPath` path to Simulink application
OpenFresco Command Language Manual

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

OpenFresco.exe & OpenFresco.dll
Version 2.6

July 2009

Created on 9/26/07
Downloading and Installing OpenFresco
OpenFresco (the Open-source Framework for Experimental Setup and Control) is an environment-independent software framework that connects finite element models with control and data acquisition systems in laboratories to facilitate hybrid simulation of structural and geotechnical systems.

Hybrid simulation is an experimental testing technique where a test is executed based on a step-by-step numerical solution of the governing equations of motion for a hybrid model, formulated considering both the numerical and physical portions of a structural system. In order for the earthquake engineering community to take full advantage of this technique, OpenFresco standardizes the deployment of hybrid simulation and extends its capabilities to applications where advanced numerical techniques are utilized, boundary conditions are imposed in real-time, and dynamic loading conditions caused by wind, blast, impact, waves, fire, traffic, and, in particular, seismic events are considered.

Accordingly, the architecture of the OpenFresco software package provides a great deal of flexibility, extensibility, and re-usability to the researcher or developer interested in hybrid simulation.
First, log in if you already have an account or register for a new account.
Download & Install OpenFresco

Note: The executable and dynamic link library for OpenFresco are only provided for Windows based PC computers. For other operating systems you will need to download and build the source code yourself.

1. Install Tcl/Tk
   If you have not installed Tcl/Tk on your computer, please download the Tcl/Tk installation file below and double-click to install it (OpenFresco employs Tcl/Tk 8.5).
   When installing Tcl/Tk it is essential that you change the installation directory to "C:\Program Files\Tcl" during the course of the installation. If you run OpenFresco and you see an error message to the effect, "Cannot find tcl85.dll", you have skipped this step and must reinstall Tcl/Tk. Note that you will probably have to uninstall the version you just installed first. Download Tcl/Tk 8.5

2. Install OpenSSL (optional)
   If you would like to use OpenFresco's capability to encrypt signals during geographically distributed hybrid simulations through Secure Socket Layer, please download the OpenSSL installation file from the website below and install it (OpenFresco employs the full (not light) version of OpenSSL 1.0). When installing OpenSSL it is essential that you change the installation directory to "C:\Program Files\OpenSSL" during the course of the installation. Download OpenSSL

3. Install Computational Driver
   Install the structural analysis software package of your choice. OpenFresco can interface with many software packages including OpenSees, Abaqus, LS-DYNA, Matlab, Simulink, UI-SimCor, ANSYS (coming soon!)

4. Download & Extract OpenFresco
   Download OpenFresco 2.6.2 (.zip file, 5 MB)
Run OpenFresco
SVN

The OpenFresco source code is stored using the Apache Subversion (SVN) software. SVN provides a means to store not only the current version of a piece of source code, but a record of all changes that have occurred to that source code over time and a record of who made those changes. The use of SVN is particularly common for software projects with multiple developers, because SVN guarantees that changes made by one developer are not accidentally removed when another developer commits changes to the source code. For the OpenFresco software project anyone can check out the code via anonymous SVN access, but only trusted developers have the ability to commit changes and additions to the code repository.

Getting the Code

To download the OpenFresco source code from the repository you can do so on your local machine first. You can download SVN for all major Linux, Windows, and MacOSX. If you are working on Windows, this is particularly nice and easy to use. It lets you control SVN functions in menu as you navigate the file system in Windows Explorer.

Once you have SVN installed, you can download the OpenFresco code:

```
svn co svn://openfresco.berkeley.edu/user/local/svn
```

The checkout command makes a local copy of the entire OpenFresco repository into your current working directory. By requesting `./OpenFresco/trunk` you can browse the source code.

Browsing the Code

You can browse the source code online using WebSVN. The spider-view presents a high-level overview of the OpenFresco repository that has been downloaded. You can see the structure of the files changed, added or deleted in any given revision. You can also see differences between two versions of a file as to see exactly what has changed between different revisions.

OpenFresco

openfresco.berkeley.edu/developers/svn
Example:
Building a Hybrid Model using OpenSees & OpenFresco
HS of Structural Collapse
Structural Collapse

- On shaking tables, simulation of collapse is dangerous and expensive.
- In hybrid simulations:
  - Gravity loads and resulting geometric nonlinearities can be modeled analytically:
    - Therefore, no complex active or passive gravity load setups are necessary.
  - Actuator movements will limit displacements during collapse (safety):
    - Thus, there is no need to protect expensive test equipment from specimen impact.
  - Only critical, collapse-sensitive elements of a structure need to be physically modeled.
Implementation in a Hybrid Model

- Physical portion of the model:
  - Test material and cross-section level response

- Analytical portion of the model:
  - Apply the gravity and/or prestress loads
  - Provide the geometric transformations such that the second-order effects due to axial loads are accounted for
  - Model the rest of the structure
Properties of Model:

- NDOF = 8 (4 with mass)
- Period: $T_1 = 0.49$ sec
- Damping: $\zeta_1 = 0.05$
- $P = 50\%$ of $\varphi P_n$

- Crd-Trans: P-Delta, Corotational
- ExpElements: EEBeamColumn2d
- ExpSetups: ESOneActuator
- ExpControl: ECxPCtarget
- SACNF01: $pga = 0.906g$
OpenFresco Local Architecture

Exp. beamColumn element defined in OpenSees

Communication methods for distributed testing. In this case, we are using a local site.

Transforms between experimental element DOFs in OPF and the actuator DOFs in the laboratory. Linear and non-linear transformations are available.

Interface with control and data acquisition systems. In this example, SimUniaxialMaterials will simulate the response of the experimental element using a material defined in OpenSees, Steel02.
Tcl File Components

**Geometry**
- Materials
- Experimental Control
- Experimental Setup
- Experimental Site
- Geometric Transformation
- Experimental Elements
- Numerical Elements
- Gravity Loads
- Gravity Analysis
- Dynamic Loads
- Dynamic Analysis
Portal Frame Model

```python
# Start of model generation

# create ModelBuilder (with two-dimensions and 3 DOF/node)
model BasicBuilder -ndm 2 -ndf 3

# Load OpenFresco package

# (make sure all dlls are in the same folder as openSees.exe)
loadPackage OpenFresco
```

![Diagram of Portal Frame Model]

- $P = 5.32$ kip
- $W6x12$ Beam
- $m_3 = 0.0138$ kip·sec²/in.
- $m_4 = 0.0138$ kip·sec²/in.
- $k_{y_{init}} = 2.8$ kip/in.
- $k_{x_{init}} = 2.8$ kip/in.
- 50.0"
- 100.0"
Geometry

- `withGravity 0`: turns off gravity loads (no P-delta)
- `withGravity 1`: turns on gravity loads (P-delta)
- Assigned no rotational mass – must use implicit integration method

```plaintext
# Define geometry for model
# -----------------------------------
set withGravity 1;
set Pc 10.638;
set P [expr 0.5*$Pc];
set mass3 [expr $P/386.1];
set mass4 [expr $P/386.1];

# node $tag $xCrd $yCrd $mass
node 1 0.0 0.0
node 2 100.0 0.0
node 3 0.0 50.0 -mass $mass3 $mass3 0.0
node 4 100.0 50.0 -mass $mass4 $mass4 0.0

# set the boundary conditions
# fix $tag $DX $DY $SRZ
fix 1 1 1 0
fix 2 1 1 0
```
Tcl File Components

Geometry
Materials
Experimental Control
Experimental Setup
Experimental Site
Geometric Transformation
Experimental Elements
Numerical Elements
Gravity Loads
Gravity Analysis
Dynamic Loads
Dynamic Analysis
Materials/Experimental Control

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td># Define materials</td>
</tr>
</tbody>
</table>
| 49 | # -----------------
| 50 | # uniaxialMaterial Steel02 $matTag $Fy $E $b $RO $cR1 $cR2 $a1 $a2 $a3 $a4
| 51 | uniaxialMaterial Steel02 1 1.5 2.8 0.01 18.5 0.925 0.15 0.0 1.0 0.0 1.0
| 52 | #uniaxialMaterial Elastic 1 2.8
| 53 |   |
| 54 | # Define experimental control |
| 55 | # ----------------------------
| 56 | # expControl SimUniaxialMaterials $tag $matTags
| 57 | expControl SimUniaxialMaterials 1 1
| 58 | expControl SimUniaxialMaterials 2 1 |

- **Want to control two columns**
- **SimUniaxialMaterials** used to simulate a specimen
- **Need to create a separate experimental control for each element** so create experimental control with tags “1” and “2”
- **Assign a material tag to each**

---

Column 1 (left)

Column 2 (right)
Tcl File Components

- Geometry
- Materials
- Experimental Control
- Experimental Setup
- Experimental Site
- Geometric Transformation
- Experimental Elements
- Numerical Elements
- Gravity Loads
- Gravity Analysis
- Dynamic Loads
- Dynamic Analysis
### Experimental Setup and Site

- **OneActuator** direction in element’s local coordinate system
- **sizeTrialOut** are equal to number of basic DOFs of element
- Optional Factors: all factors are multipliers
- e.g.
  - `-crtlDispFact 2` = target displacement $\times$ 2
  - `-daqDispFact 0.5` = measured displacement $\times$ 0.5

![Diagram of experimental setup with actuator and specimen](image)

Refers back to SimUniaxialMaterials Experimental Control

Left and right columns (tags 1 and 2)
Tcl File Components

- Geometry
- Materials
- Experimental Control
- Experimental Setup
- Experimental Site
- Geometric Transformation
- Experimental Elements
- Numerical Elements
- Gravity Loads
- Gravity Analysis
- Dynamic Loads
- Dynamic Analysis
# Define geometric transformation
# -------------------------------------
#geomTransf PDelta 1
gemTransf Corotational 1

# Define experimental elements
# -------------------------------------
# left and right columns
# expElement beamColumn $eleTag $iNode $jNode $transTag -site $siteTag -initStif $Kij <-iMod
expElement beamColumn 1 3 1 -site 1 -initStif 1310.8 0 0 0 11.2 -280.0 0 -280.0 9333.3333
expElement beamColumn 2 4 2 -site 2 -initStif 1310.8 0 0 0 11.2 -280.0 0 -280.0 9333.3333

# Define numerical elements
# -------------------------------------
# element elasticBeamColumn $eleTag $iNode $jNode $A $E
element elasticBeamColumn 3 3 4 3.55 29000 22.1 1

\[
K_{init} = \begin{bmatrix}
\frac{EA}{L} & 0 & 0 \\
0 & \frac{12EI}{L^3} & -\frac{6EI}{L^2} \\
0 & -\frac{6EI}{L^2} & \frac{4EI}{L}
\end{bmatrix} = \begin{bmatrix}
1310.8 & 0 & 0 \\
0 & 11.2 & -280.0 \\
0 & -280.0 & 9333.3
\end{bmatrix}
\]

- L = 50”
- A = 2.26 in²
- I = 4.02 in⁴
- E = 29000 ksi
Tcl File Components

- Geometry
- Materials
- Experimental Control
- Experimental Setup
- Experimental Site
- Geometric Transformation
- Experimental Elements
- Numerical Elements
  - Gravity Loads
  - Gravity Analysis
- Dynamic Loads
- Dynamic Analysis
Gravity Loads

if {$withGravity} {
  # Define gravity loads
  # ------------------
  # Create a Plain load pattern with a Linear TimeSeries
  pattern Plain 1 "Linear" {
    # Create nodal loads at nodes 2
    #   nd   FX  FY   MZ
    load 3 0.0 [expr -$P] 0.0
    load 4 0.0 [expr -$P] 0.0
  }
  # ------------------
  # End of model generation
  # ------------------
}

Loads in the negative Y-direction at nodes 3 and 4
Gravity Analysis Options

Banded General SOE

DOFs assigned arbitrarily (ok for small models)

Only using homogeneous single point constraints

Test EnergyIncr $tol $maxNumIter

Load Control with 10 steps

Newton-Raphson algorithm

Perform a static analysis

# Start of analysis generation
# -----------------------------------
# Create the system of equation
system BandGeneral
# Create the DOF numberer
numberer Plain
# Create the constraint handler
constraints Plain
# Create the convergence test
test EnergyIncr 1.0e-6 10
# Create the integration scheme
integrator LoadControl 0.1
# Create the solution algorithm
algorithm Newton
# Create the analysis object
analysis Static
# -----------------------------------
# End of analysis generation
# -----------------------------------
Gravity Recorders and Analysis

# Start of recorder generation
# --------------------------------------------------
# create a Recorder object for the nodal displacements at node 2
recorder Node -file Gravity_Dsp.out -time -node 3 4 -dof 1 2 3 disp
recorder Element -file Gravity_Frc.out -time -ele 1 2 3 force
# --------------------------------------------------
# End of recorder generation
# --------------------------------------------------


# Perform the gravity analysis
# --------------------------------------------------
# perform the gravity load analysis, requires 10 steps to reach the load level
if {[analyze 10] == 0} {
    puts "\nGravity load analysis completed"
} else {
    puts "\nGravity load analysis failed"
    exit -1
}
Tcl File Components

- Geometry
- Materials
- Experimental Control
- Experimental Setup
- Experimental Site
- Geometric Transformation
- Experimental Elements
- Numerical Elements
- Gravity Loads
- Gravity Analysis
- Dynamic Loads
- Dynamic Analysis
Dynamic Loads

```tcl
# Define dynamic loads
# ------------------
# set time series to be passed to uniform excitation
set dt 0.01
set scale 1.2
timeSeries Path 1 filePath SACNF01.txt -dt $dt -factor [expr 386.1*$scale]

# create UniformExcitation load pattern
# pattern UniformExcitation $tag $dir -accel $tsTag <-vel0 $vel0>
pattern UniformExcitation 2 1 -accel 1

# calculate the rayleigh damping factors for nodes & elements
set alphaM 1.2797; # D = alphaM*M
set betaK 0.0; # D = betaK*Kcurrent
set betaKinit 0.0; # D = betaKinit*Kinit
set betaKcomm 0.0; # D = betaKcomm*KlastCommit

# set the rayleigh damping
rayleigh $alphaM $betaK $betaKinit $betaKcomm
# ------------------
# End of model generation
# ------------------
```

Place ground motion file in the same folder as the PortalFrame_Local.tcl file
Dynamic Analysis Options

# Start of analysis generation
# -----------------------------------
# create the system of equations
system BandGeneral

# create the DOF numberer
numberer Plain

# create the constraint handler
constraints Plain

# create the convergence test
test FixedNumIter 5

# create the integration scheme
integrator NewmarkHSFixedNumIter 0.5 0.25

# create the solution algorithm
algorithm Newton

# create the analysis object
analysis Transient

# -----------------------------
# End of analysis generation
# -----------------------------

Same as gravity analysis

5 iterations/time-step

NewmarkHSFixedNumIter: implicit Newmark method with 5 iterations/time-step

γ=0.5: second order accuracy, no numerical damping

β=0.25: average acceleration, unconditional stability
# Start of recorder generation

# create the recorder objects

recorder Node -file Node_Dsp.out -time -node 3 4 -dof 1 2 3 disp
recorder Node -file Node_Vel.out -time -node 3 4 -dof 1 2 3 vel
recorder Node -file Node_Acc.out -time -node 3 4 -dof 1 2 3 accel
recorder Node -file Node_Rxn.out -time -node 1 2 3 4 -dof 1 2 3 reactionIncludingInertia

recorder Element -file Elmt_glbFrc.out -time -ele 1 2 3 forces
expRecorder Control -file Control_ctrlDsp.out -time -control 1 2 ctrlDisp
expRecorder Control -file Control_daqDsp.out -time -control 1 2 daqDisp
expRecorder Control -file Control_daqFrc.out -time -control 1 2 daqForce

# End of recorder generation

#
Dynamic Analysis

```plaintext
# Finally perform the analysis
#
# perform an eigenvalue analysis
set pi [expr acos(-1.0)]
set lambda [eigen -fullGenLapack 4]
puts "\nEigenvalues at start of transient:"
puts "|   lambda   |   omega   |   period   |   frequency   |
foreach lambda $lambda {
    set omega [expr pow($lambda, 0.5)]
    set period [expr 2.0*$pi/$omega]
    set frequ [expr 1.0/$period]
    puts [format "|   %5.3e   |   %8.4f   |   %7.4f   |   %9.4f   |
           $lambda   $omega   $period   $frequ"
}

# open output file for writing
set outFileID [open elapsedTime.txt w]
# perform the transient analysis
set tTot [time {
    for {set i 1} {($i < 2500)} {incr i} {
        set t [time {analyze 1 [expr $dt/1.0]}]
        puts $outFileID $t
        #puts "step $i"
    }
}]
```
Running the Hybrid Simulation

- Start the OpenSees executable file from the directory where you saved PortalFrame_Local.tcl
- At the prompt, type `source PortalFrame_Local.tcl` and press enter
Run Simulation

OpenSees > source PortalFrame_Local.tcl

OpenFresco -- Open Framework for Experimental Setup and Control
Version 2.6.2
Copyright (c) 2006 The Regents of the University of California
All Rights Reserved

WARNING: EEBeamColumn2d::getTangentStiff() - Element: 1
TangentStiff cannot be calculated.
Return InitialStiff including GeometricStiff instead.
Subsequent getTangentStiff warnings will be suppressed.

WARNING: EEBeamColumn2d::getTangentStiff() - Element: 2
TangentStiff cannot be calculated.
Return InitialStiff including GeometricStiff instead.
Subsequent getTangentStiff warnings will be suppressed.

Gravity load analysis completed

WARNING: NewmarkHSFixedNumIter::domainChanged() - assuming Ut-1 = Ut

Eigenvalues at start of transient:

<table>
<thead>
<tr>
<th>lambda</th>
<th>omega</th>
<th>period</th>
<th>frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.639e+02</td>
<td>12.8040</td>
<td>0.4907</td>
<td>2.0370</td>
</tr>
<tr>
<td>9.515e+04</td>
<td>308.4630</td>
<td>0.0204</td>
<td>49.0934</td>
</tr>
<tr>
<td>9.532e+04</td>
<td>308.7421</td>
<td>0.0204</td>
<td>49.1378</td>
</tr>
<tr>
<td>1.496e+05</td>
<td>386.7612</td>
<td>0.0162</td>
<td>61.5550</td>
</tr>
</tbody>
</table>

Elapsed Time = 815445 microseconds per iteration

C:\Users\Andreas\Documents\OpenFresco\SourceCode\trunk\EXAMPLES\PortalFrame>
Recorders Save Output Files

<table>
<thead>
<tr>
<th>Name</th>
<th>Date modified</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>PortalFrame_Local.tcl</td>
<td>3/6/2013 7:20 PM</td>
<td>TCL File</td>
<td>9 KB</td>
</tr>
<tr>
<td>Node_Vel.out</td>
<td>3/6/2013 8:50 PM</td>
<td>OUT File</td>
<td>164 KB</td>
</tr>
<tr>
<td>Node_Rxn.out</td>
<td>3/6/2013 8:50 PM</td>
<td>OUT File</td>
<td>286 KB</td>
</tr>
<tr>
<td>Node_Dsp.out</td>
<td>3/6/2013 8:50 PM</td>
<td>OUT File</td>
<td>168 KB</td>
</tr>
<tr>
<td>Node_Acc.out</td>
<td>3/6/2013 8:50 PM</td>
<td>OUT File</td>
<td>144 KB</td>
</tr>
<tr>
<td>Gravity_Frc.out</td>
<td>3/6/2013 8:50 PM</td>
<td>OUT File</td>
<td>2 KB</td>
</tr>
<tr>
<td>Gravity_Dsp.out</td>
<td>3/6/2013 8:50 PM</td>
<td>OUT File</td>
<td>1 KB</td>
</tr>
<tr>
<td>Elmt_glbFrc.out</td>
<td>3/6/2013 8:50 PM</td>
<td>OUT File</td>
<td>381 KB</td>
</tr>
<tr>
<td>Control_daqFrc.out</td>
<td>3/6/2013 8:50 PM</td>
<td>OUT File</td>
<td>63 KB</td>
</tr>
<tr>
<td>Control_daqDsp.out</td>
<td>3/6/2013 8:50 PM</td>
<td>OUT File</td>
<td>61 KB</td>
</tr>
<tr>
<td>Control_ctrlIDsp.out</td>
<td>3/6/2013 8:50 PM</td>
<td>OUT File</td>
<td>61 KB</td>
</tr>
<tr>
<td>SACNF01.txt</td>
<td>10/13/2010 10:47 PM</td>
<td>Notepad++ Document</td>
<td>78 KB</td>
</tr>
<tr>
<td>elapsedTime.txt</td>
<td>3/6/2013 8:50 PM</td>
<td>Notepad++ Document</td>
<td>79 KB</td>
</tr>
<tr>
<td>PlotOutput.m</td>
<td>3/6/2013 8:59 PM</td>
<td>MATLAB Code</td>
<td>9 KB</td>
</tr>
</tbody>
</table>
Compare Deformations

Deformation Histories

- Deformation Histories without Gravity Loads
- Deformation Histories with Gravity Loads

Time [sec]

Deformation [in.]
Compare Hysteresis Loops

- w/o Gravity
- with Gravity
Simulated Controllers vs. Real Controllers
Connecting to MTS 493 controller

OpenSees Finite Element Model

OpenFresco Middleware

xPC-Target real-time Predictor-Corrector

MTS 493 real-time Controller

Physical Specimen in μNEES Lab

Connecting to MTS 493 controller

OpenSees

ExpElement

LocalExpSite

ExperimentalSetup

ExperimentalControl

Client

Backend Server

Control System in Laboratory

xPC-Target real-time Predictor-Corrector

MTS 493 real-time Controller

Physical Specimen in μNEES Lab

OpenFresco
Modify Experimental Control

- Want to control two columns
- xPCtarget used for left column
- SimUniaxialMaterials used to simulate right column
Computational Drivers
How to Interface

- Two Ways to Interface with FE-Software
  - Generic Client Element
  - Experimental Element Directly in FE-Software

- Generic Client Element to be Programmed by the Developers

- Several generic client elements available: /trunk/SRC/simApplicationClient
Computational Drivers

- OpenSees
- LS-DYNA
- Abaqus
- Matlab/Simulink
- UI-SimCor
Integration Methods

\[ M \cdot \ddot{u}_n + C \cdot \dot{u}_n + \mathbf{P}_r (u_n) = \mathbf{P}(t_n) \]

- Mass matrix \( M \) is often singular
  -> second order differential equation
  infinitely stiff -> fully implicit numerical methods

- Make as few function calls as possible

- Use constant Jacobian in the numerical methods since tangent stiffness is not available
Direct Integration Methods

- **Explicit Integrators**
  - explicit Newmark Method
  - Central-Difference Method
  - explicit Alpha Method
  - explicit Generalized-Alpha Method

- **Implicit Integrators (do not use for HS)**
  - Newmark Method
  - Alpha Method
  - Generalized-Alpha Method
  - Collocation Method
Direct Integration Methods

- Implicit Integrators with increment reduction factors
  - Newmark HS IncrReduct Method
  - Generalized-Alpha HS IncrReduct Method
  - Collocation HS IncrReduct Method

- Implicit Integrators with increment limits
  - Newmark HS IncrLimit Method
  - Generalized-Alpha HS IncrLimit Method
  - Collocation HS IncrLimit Method
Direct Integration Methods

- Implicit Integrators with sub-stepping (constant number)
  - Newmark HS FixedNumIter Method
  - Generalized-Alpha HS FixedNumIter Method
  - Collocation HS FixedNumIter Method

- Predictor-Corrector Integrators
  - Alpha-OS Method
  - Generalized-Alpha-OS Method
OpenFresco (the Open-source Framework for Experimental Setup and Control) is an environment-independent software framework, that connects finite element models with control and data acquisition systems in laboratories to facilitate hybrid simulation of structural and geotechnical systems.

Hybrid simulation is an experimental testing technique where a test is executed based on a step-by-step numerical solution of the governing equations of motion for a hybrid model, formulated considering both the numerical and physical portions of a structural system. In order for the earthquake engineering community to take full advantage of this technique, OpenFresco standardizes the deployment of hybrid simulation and extends its capabilities to applications where advanced numerical techniques are utilized, boundary conditions are imposed in real-time, and dynamic loading conditions caused by wind, blast, impact, waves, fire, traffic, and, in particular, seismic events are considered. Accordingly, the architecture of the OpenFresco software package provides a great deal of flexibility, extensibility, and re-usability to the researcher or developer interested in hybrid simulation.

Hybrid Simulation

\[ M \ddot{u} + C \dot{u} + P_e(u) = P(t) \]

Dynamic Loading:
- Seismic
- Wind
- Blast/Impact
- Wave
- Traffic

Static Loading:
- Gravity
- Prestress

Download documentation and software:
http://openfresco.berkeley.edu

OpenFrescoExpress is a self-contained software package, including a easy-to-use graphical user interface, that facilitates hybrid testing of systems having up to two degrees of freedom. OpenFrescoExpress addresses the needs of a wide range of users including:

- Laboratory staff and research students learning about hybrid simulation and starting to use this experimental testing method.
- Staff and students at laboratories that regularly use hybrid simulation but desire a tool for quick demonstration of the hybrid simulation testing method.
- Researchers who are conducting simple tests and would like to take advantage of a graphical user interface that quickly and easily displays useful real-time test data.
- Graduate students and researchers who are not at a laboratory but wish to run the software as a pure simulation tool to learn more about hybrid simulation and how it works.

Download report from:
http://peer.berkeley.edu/publications/
Select Structure

Period is immediately calculated

Assign Mass

Assign Stiffness

Select & Assign Damping

GUI: OpenFresco Express
Summary & Conclusions

- Hybrid simulation inherently requires close collaboration amongst experts from many different fields.
  - Structural behavior
  - Laboratory testing and control
  - Computational simulation
  - Information technology

- Hence, hybrid simulation fosters collaboration and communication among distant researchers in different labs.
Summary & Conclusions

- OpenFresco, the environment-independent software framework for the development and deployment provides an excellent platform for this collaboration (on-site and geographically distributed).

- The modularity and transparency of the framework permits existing components to be modified and new components to be added without much dependence on other objects.
Summary & Conclusions

- Large libraries of hybrid simulation direct integration methods, experimental elements, experimental setups, controller models, and event-driven solution strategies are available to the researchers to choose or adapt from.

- Needs:
  - User feedback on refinements and new features
  - Developer contributions to extend libraries
Questions?
Thank you!

http://openfresco.berkeley.edu/

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