ANEXO

ANEXO 1 TUTORIAL BÁSICO PARA EL USO DE LA RUTINA PSC.

1. Ejemplo de Comandos Básicos de OpenSees

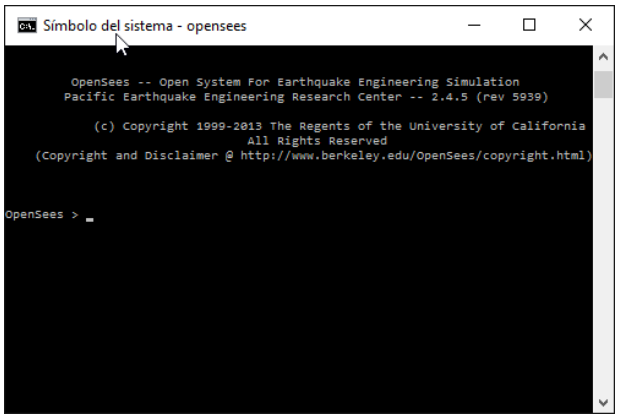
OpenSees cuenta con una interface de comandos mediante un intérprete de comandos del lenguaje tcl, el cual esta ampliado para incorporar las funcionalidades de OpenSees.

Todas las funcionalidades de tcl están disponibles en el intérprete el cual se instala como opensees.exe, para funcionar requiere la instalación de la aplicación activetcl.

OpenSees es un marco de trabajo itegrado y no un programa solamente, si se requiere o se prefiere el uso de lenguajes de programación como c, c++ o fortran es posible también el uso de OpenSees.

Una vez instalado OpenSees, desde la línea de comandos (cmd.exe) se puede llamar a OpenSees mediante el comando “opensees”, una vez hecho esto se estará en la línea de comandos del intérprete de tcl de OpenSees.

En las siguientes páginas se presenta un ejemplo elemental del programa, si se desea ampliar la información se recomienda consultar el manual [1], o bien la página de soporte: http://opensees.berkeley.edu/ , donde además se puede descargar el programa con código fuente e instrucción de instalación.



Aunque es posible, realizar en su totalidad la construcción y análisis de un modelo desde la línea de comandos, es más recomendable elaborar un archivo script con los comandos, tal cual se deberían incorporar en el programa, para cargar este archivo se empela la instrucción “source”.

```
source util/DisplayModel2D.tcl;        # procedure for displaying a 2D perspective of model
```

La construcción paso a paso de un modelo es como sigue:

Se crea un objeto que contiene el modelo definiendo del numero de dimensiones (-ndm) y el número de grados de libertad (-ndf).

En este caso se trata de un modelo de 2 dimensiones y 3 grados de libertad.

```
model basic -ndm 2 -ndf 3
```

Con la instrucción “node” se definen los nodos o puntos, como cada elemento en OpenSees este debe tener un identificador único compuesto por un número entero.

```
# Create nodes
#    tag        X        Y
node  1        0.0      0.0
```

Y se definen las condiciones de frontera:

```
# Fix supports at base of columns
#    tag   DX   DY   RZ
fix   1     1    1    1
fix   2     1    1    1
```

Dependiendo de los recursos de memoria es posible contener hasta 2^{32} nodos en el dominio, sin embargo, el tiempo de análisis que implica un modelo tal limita considerablemente esta capacidad en la realidad, este límite también aplica para todos los elementos en OpenSees.

Con los nodos definidos es posible definir los demás elementos como barras, áreas, solidos, etc. en este caso se define una viga lineal elástica.

```
# Geometry of column elements
#                tag
geomTransf Linear 1

# Create the beam element
#                          tag ndI ndJ     A       E     Iz    transfTag
element elasticBeamColumn   1   1   3    0.009    24855578  0.000675    1
```

En la librería de OpenSees existen más de 100 elementos y es posible implementar nuevos de ser necesario.

Una vez definidos los elementos del modelo, se definen las cargas:

```
# Create a Plain load pattern with a Linear TimeSeries
pattern Plain 1 "Linear" {

    # Create nodal loads at nodes 3 & 4
    #    nd    FX          FY  MZ
    load  4   0.0  [expr -$P] 0.0
}
```

Dentro de la instrucción “pattern” es posible ejecutar cualquier conjunto de instrucciones necesarios para aplicar las cargas, desplazamientos o relaciones entre dof que se requieran, se pueden tener tantos patrones como se requieran.

La expresión “Linear” en este caso hace referencia a la forma como se aplica la carga a lo largo del tiempo de integración y no implica un análisis lineal.

Para continuar con el la modelación se deben definir parámetros como, la forma en que se conformará el sistema (matriz), los tipos de restricciones entre los nodos, el algoritmo a emplear para numerar los dof:

```
# Create the system of equation, a sparse solver with partial pivoting
system BandGeneral

# Create the constraint handler, the transformation method
constraints Transformation

# Create the DOF numberer, the reverse Cuthill-McKee algorithm
numberer RCM

# Create the convergence test, the norm of the residual with a tolerance of
# 1e-12 and a max number of iterations of 10
test NormDispIncr 1.0e-12  10 3
```

Se define el algoritmo de solución del sistema y el método mediante el cual se va aplicando y controlando el incremento de carga en cada tiempo llamado “integrator”, en el caso de análisis no lineal también se debe definir el “test”, que es el criterio a emplear para determinar si hay o no convergencia, por último en esta etapa se crea el objeto “analysis“ especificándole si el análisis es estático “Static” o “Trasient”, la diferencia está en la consideración de los parámetros dinámicos (matriz de masa, amortiguación, etc).

```
# Create the analysis object
analysis Static
```

Finalmente se aplica el comando “analize”, con el cual se ejecuta la solución del sistema, si se adjunta un parámetro numérico, indica la cantidad de pasos en los cuales se dividirá el seudotiempo para la aplicación del patrón de carga.

```
# perform the gravity load analysis, requires 10 steps to reach the load level
analyze 10

# Print out the state of nodes 3 and 4
print node 4
```

En modelos relativamente sencillos se puede utilizar la instrucción “print” que presenta en pantalla los resultados del sistema completo, si se adjunta a la instrucción print un nombre de archivo generará un archivo con la información.

Para modelos más complejos se debe utilizar el objeto “recorder” que una vez finalizado el programa genera un archivo con la información especificada del objeto especificado.

```
# ------------------------------
recorder Element -file eleForces1.out -ele 1 2 3 4 5  forces
recorder Element -file elebForces1.out -ele 1 2 3 4 5  basicForces
```

Para la elaboración de scrips más complejos se requiere de conocimientos de básicos de programación y del lenguaje tcl.

2. Ejemplo de un modelo en PSC.

Para el análisis de estructuras en la rutina para colapso progresivo, se implementaron algunas funcionalidades adicionales que facilitan la definición del modelo.

Para su ejecución en OpenSees se ejecuta la instrucción:

```
source psc.tcl
```

desde la carpeta en la cual se localizan los archivos de la rutina psc.tcl, esta genera un modelo con base en el archivo Model4.tcl; en este tutoria se explicara la estructura e instrucción necesarios para definir un modelo editando el fichero Model4.tcl

En primera medida en la carpeta que contiene los archivos ejecutables de tcl, debe existir una carpeta llamada “Models”, dentro de esta carpeta se debe generar una carpeta para cada modelo que se quiera analizar, dentro de esta carpeta se encuentra el archivo Model4.tcl, al efectuar el análisis en esta carpeta se guardaran los resultados en carpetas nombradas según el tipo de análisis efectuado.

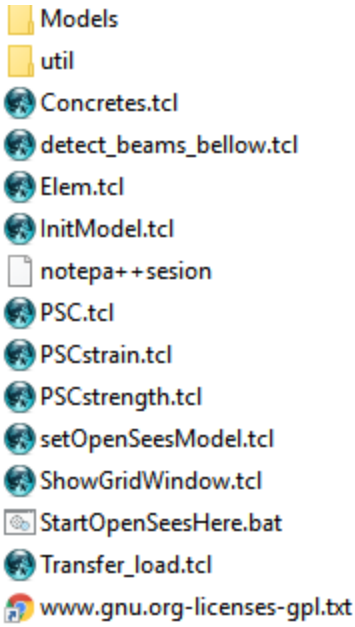


Figura 2-1 Estructura de Archivos

En la Figura 2-1 se muestran los archivos que conforman la aplicación.

Para seleccionar el modelo a analizar se debe modificar la línea 67 del archivo PSC.tcl, definiendo la variable g_data(Name) con el nombre del archivo a ejecutar, el mismo de la carpeta ubicada en la carpeta “Models”. No se requiere modificar nada más en el archivo PSC.

```
#Model path, where all the info is placed, specially Model4.tcl
set g_data(Name) "test02"
```

El archivo Model4.tcl sigue el orden de generación del modelo en el OpenSees u otros programas de análisis.
Las primeras líneas del archivo, permiten definir el nombre del modelo, por simple referencia, este valor solamente se utiliza para presentar este nombre en pantalla.

La siguiente línea es la definición del tipo de criterio a emplear para considerar la falla de los elementos estructurales, esto se hace definiendo el valor de la variables Fail_Criteria, si se define como Strenght se emplearan los criterios de falla por resistencia, si se

emplea el valor Strain, se considerará la deformación límite establecida para los materiales.

Posteriormente se establecen dos variables del aplicativo las cuales controlan el número de pasos en los cuales se aplicara la carga durante el análisis por colapso NSL y el número interno de pasos en el análisis no lineal realizado por OpenSees NLAS, estos valores se establecen buscando un equilibrio entre tiempo de cálculo y facilidad de convergencia.

Por último la variable mesh define el número de subelementos que conforman cada elemento estructural definido.

```
set ModelName "test01"
set Fail_Criteria "Strength" ; # "Strain" or "Strength"
set Fail_Criteria "Strain"
#set Fail_Criteria "none"

set NSL 25 ;# Number of load steps in PC analysis
set NLAS 25; #Number of steps/load increments for the nl anlysis (for each NSL increment)
set damageLimit 0.2

set mesh 9
```

Dentro del archivo es posible realizar operaciones entre variables e incluso realizar rutinas complejas con ciclos como el for o while, por ejemplo en este modelo se define el valor de la carga en las vigas por medio de operaciones aritméticas.

```
set DL -3.85; #kN/m2
set LL -2.0; #kN/m2
set floorLenght 2.0; #m

set beamsLoad [expr (1.2*$DL+0.5*$LL)*$floorLenght]; #kN/m -->> equ. 3-10 UFC 402303
```

La definición de los materiales se realiza especificando 3 argumentos de un arreglo denominado Mat, el primero es el tipo (type) de material el cual puede ser 1: Concreto inconfinado, 2: Concreto Confinado y 3:Acero, el segundo valor es el identificado del material que puede ser cualquier numero entero sucesivo (tagMaterial) y por último el valor F el cual corresponde al f'c en el caso de concretos o al fy en el caso de acero.

```
set n_Materials 3
#Confined Concrete
set Mat(1,type) 1
set Mat(1,tagMaterial) 1
set Mat(1,F) -28000.0
#Unconfined Concrete
set Mat(2,type) 2
set Mat(2,tagMaterial) 2
set Mat(2,F) -28000.0
#Rebar steel1
set Mat(3,type) 3
set Mat(3,tagMaterial) 3
set Mat(3,F) 420000.0
```

La definición de vigas o columnas requiere que previamente se haya definido su sección transversal, esto se realiza mediante la función set_template_section, esta función tiene el siguiente prototipo:

```
proc set_template_section { {type 2}
                            {h 0.6} {b 0.6}
                            {idCoverCte 1}
                            {idCoreCte 2}
                            {idSteel 3}
                            {cover 0.05}
                            {Astop 0.02}
                            {Asbott 0.02}
                            {Asral 0.02}
                            {Asv 0.000946}
                            {nx 5} {nz 5}
                            {dd 10}  {phiy 5.6e-3}
                            {phipra 0.15} {M0 0.0} } {
                                #the template s
```

Donde cada parámetro se debe especificar así:

Type: el tipo de sección 1 para lineal y 2 para no lineal
h: La altura de la sección
b: El ancho de la sección
idCoverCte : El identificador del material de concreto de recubrimiento
idCoreCte : El identificador del material de concreto del nucleo confinado.
idSteel : El identificador del material del acero de refuerzo.
Cover: espesor del recubrimiento

Astop: Area de refuerzo en la cara superior de la sección

Asbott Area de refuerzo en la cara inferior de la sección

Asral: Area de refuerzo en las dos caras laterales de la sección Asv: Area de refuerzo transversal de confinamiento.

nx Numero de varillas en cada una de las caras inferior y superior.

nz: Numero de varillas en cada una de las caras laterales

dd :Número de elementos en que se discretica la sección no lineal

phiy Rotación de fluencia de la sección.

Phipra Rotación ultima de la sección.

M0 Momento de fluencia de la sección.

Esta función da como resultado un número entero y puede ser asignada a vigas o columnas indiferentemente.

```
set ind_col_01 [set_template_section 2 0.4 0.4 1 2 3 0.05 0.0004
                0.0004  0.0004 0.00946 4 3 10 0.0081 0.13 250.0]
```

Las líneas anteriores se definieron una sección no lineal de 40x40 cm y el índice de la nueva sección creada se guardó en la variable ind_col_01, posteriormente se utiliza esta variable para crear las columnas que tienen esta sección.

Para la definición de los nodos del modelo se emplea la función set_node definiendo en este orden su identificador, la coordenada X, la coordenada Y, el valor 1 si es un apoyo o 0 si no, el valor 0, el valor 0, la carga en X, la carga en Y.

Los valores que se colocan como 0 son con el objeto de inicializar la variable, los valores de carga son opcionales y se pueden omitir, sin embargo, si se define un valor para la carga en Y y no para la carga en X, esta última se debe asignar como 0.0.

```
#    tag        X       Y   Fixed?  Removed? Supported? LoadX LoadY
#ground level first
set_node 1 0.0 0.0 1 0 1
set_node 2 $L1 0.0 1 0 1
set_node 3 [expr $L1+$L2] 0.0 1 0 1
```

Las columnas se definen empleando las funciones set_col cuyos argumentos son identificador, nodo inicial, nodo final, valor 0, valor 0 y el identificador de la sección.

Las vigas se definen mediante la función set_beam cuyos argumentos son identificador, nodo inicial, nodo final, carga distribuida, valor 0, valor 0 e identificador de la sección. Los valores que se colocan como 0 son con el objeto de inicializar la variable.

```
set_col 1 1 4 0 0 $ind_col_01
set_col 2 2 7 0 0 $ind_col_01
set_col 3 3 10 0 0 $ind_col_01

set_beam 4 4 5  $beamsLoad 0 0 $ind_bea_04
set_beam 5 5 6  $beamsLoad 0 0 $ind_bea_05
```

Con esto termina la definición del archivo Model4.tcl y se puede proceder a su ejecución .

3. Resultados

El programa guarda los resultados del análisis en una carpeta ubicada en el mismo directorio donde se encuentra el archivo model4.tcl, los archivos producidos son los siguientes:

log.txt El texto que se presenta en pantalla, durante la ejecución.

displacements(a b c).txt una tabla con los desplazamientos de cada nodo en para cada grado de libertad, para iteración a, el paso b y el intento c.

reaction(a b c).txt una tabla con las acciones internas en los extremos de cada subelemento, para iteración a, el paso b y el intento c.

strain.txt en el caso de análisis por deformaciones este archivo se genera guardando información detallada de las deformaciones controladas, el dañp acumulado y la causa de falla alcanzada.

El volumen de información producida es considerable, por lo tanto se requiere de una herramienta adicional para el procesamiento de esta información.

4. Información digital

En el anexo 2 se encuentra el código fuente de las rutinas mencionadas, así como modelos de ejemplo, que permiten un punto de partida para modelos nuevos.

ANEXO 2 CÓDIGO FUENTE Y PROGRAMAS