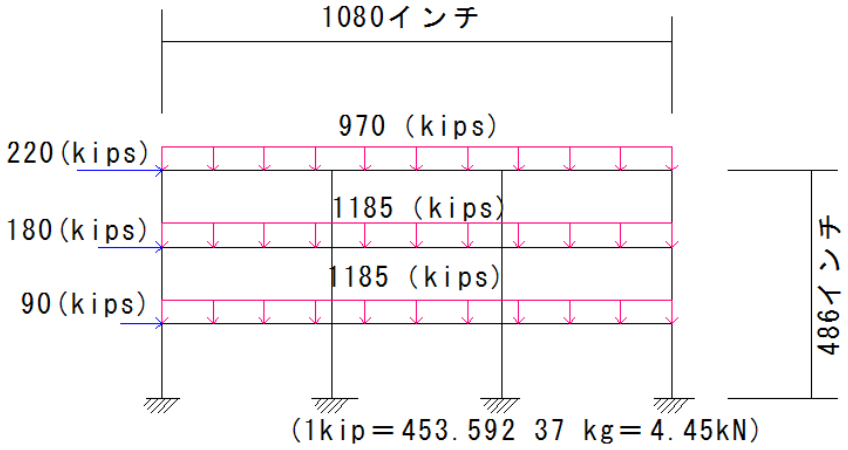
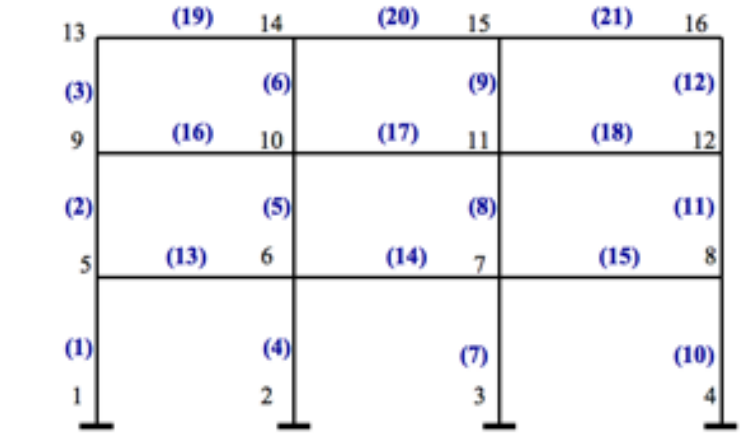
**Elastic Frame**

弾性ラーメン構造の静的荷重を受けた時の解析。

節点は剛接合。




FE Model: Node & Element Numbering

注:

1. 破線のボックス内は、入力ファイルの行です。
2. # コメントから始まるすべての行はプログラムでは無視されますが、コードを文書化するのに便利です。入力のスクリプトを作成するときはコメントを入れることをお勧めします。
3. 簡潔にするためを、「；」使用して、コマンドの終わりを意味する。
4. 画面への情報出力は必要以上に複雑になります。気にならないのであれば、無視してください。

パラメーター

モデルを構築する前にいくつかのパラメーターを設定します。

円周率 PI、重力定数 g、各床重量の変数を設定しています。

```
set PI   [expr 2.0 * asin(1.0)]
set g    386.4
set ft   12.0
set m1   [expr 1185.0/(4*$g)]; # 4 nodes per floor   #節点 1 つ当たりの荷重
set m2   [expr 1185.0/(4*$g)]
set m3   [expr   970.0/(4*$g)]
set w1   [expr 1185.0/(90*$ft)];  #1m あたりの荷重      ※(90×12＝1080)
set w2   [expr 1185.0/(90*$ft)]
set w3   [expr   970.0/(90*$ft)]
```

モデル

モデルは 16 の節点、21 本の要素、ロードパターンは分布荷重、支持点は全て固定。弾性の要素には、関連付けられた材料のオブジェクトはありませんが、幾何学的変換をします。この例ではすべての柱は、PDelta 変換、およびすべての梁は、Linear 変換される。

```
# Units: kips, in, sec
#単位は kips、インチ、秒。
# Remove existing model
#既存のモデルを削除する。
wipe

# Create ModelBuilder (with two-dimensions and 3 DOF/node)
#モデルの空間を定義(2 次元の節点自由度は最大 3)
model BasicBuilder -ndm 2 -ndf 3


# Create nodes & add to Domain –
#節点と自重を設定。
# command: node nodeId xCrd yCrd <-mass $massX $massY $massRz>
#                  番号          座標                        自重？
#    NOTE: mass is optional
node 1       0.0     0.0
node 2     360.0     0.0
node 3     720.0     0.0
```

```
node 4    1080.0     0.0
node 5       0.0    162.0 -mass $m1 $m1 0.0
node 6    360.0    162.0 -mass $m1 $m1 0.0
node 7    720.0    162.0 -mass $m1 $m1 0.0
node 8 1080.0    162.0 -mass $m1 $m1 0.0
node 9         0.0 324.0 -mass $m2 $m2 0.0
node 10    360.0 324.0 -mass $m2 $m2 0.0
node 11    720.0 324.0 -mass $m2 $m2 0.0
node 12 1080.0 324.0 -mass $m2 $m2 0.0
node 13       0.0 486.0 -mass $m3 $m3 0.0
node 14    360.0 486.0 -mass $m3 $m3 0.0
node 15    720.0 486.0 -mass $m3 $m3 0.0
node 16 1080.0 486.0 -mass $m3 $m3 0.0

# Set the boundary conditions - command: fix nodeID xResrnt? yRestrnt? rZRestrnt?
#境界条件      1=固定   0=自由
fix 1 1 1 1
fix 2 1 1 1
fix 3 1 1 1
fix 4 1 1 1

# Define geometric transformations for beam-column elements
#座標変換を定義。
geomTransf Linear 1; # beams #梁
geomTransf PDelta 2; # columns #柱

# Define elements
#要素の定義。
# Create elastic beam-column elements
#梁と柱の要素を作成。
# command: element elasticBeamColumn eleID node1 node2 A E Iz geomTransfTag
# element elasticBeamColumn　タグ番号　節点番号　断面積　ヤング率　断面二次モーメント　座標変換のタグ番号：梁=1　柱=2
# Define the Columns
#柱を定義。
element elasticBeamColumn   1   1   5 75.6 29000.0 3400.0 2; # W14X257
```

```
element elasticBeamColumn    2    5    9 75.6 29000.0 3400.0 2; # W14X257
element elasticBeamColumn    3    9 13 75.6 29000.0 3400.0 2; # W14X257
element elasticBeamColumn    4    2    6 91.4 29000.0 4330.0 2; # W14X311
element elasticBeamColumn    5    6 10 91.4 29000.0 4330.0 2; # W14X311
element elasticBeamColumn    6 10 14 91.4 29000.0 4330.0 2; # W14X311
element elasticBeamColumn    7    3    7 91.4 29000.0 4330.0 2; # W14X311
element elasticBeamColumn    8    7 11 91.4 29000.0 4330.0 2; # W14X311
element elasticBeamColumn    9 11 15 91.4 29000.0 4330.0 2; # W14X311
element elasticBeamColumn 10    4    8 75.6 29000.0 3400.0 2; # W14X257
element elasticBeamColumn 11    8 12 75.6 29000.0 3400.0 2; # W14X257
element elasticBeamColumn 12 12 16 75.6 29000.0 3400.0 2; # W14X257

# Define the Beams
#梁の定義。
element elasticBeamColumn 13    5    6 34.7 29000.0 5900.0 1; # W33X118
element elasticBeamColumn 14    6    7 34.7 29000.0 5900.0 1; # W33X118
element elasticBeamColumn 15    7    8 34.7 29000.0 5900.0 1; # W33X118
element elasticBeamColumn 16    9 10 34.2 29000.0 4930.0 1; # W30X116
element elasticBeamColumn 17 10 11 34.2 29000.0 4930.0 1; # W30X116
element elasticBeamColumn 18 11 12 34.2 29000.0 4930.0 1; # W30X116
element elasticBeamColumn 19 13 14 20.1 29000.0 1830.0 1; # W24X68
element elasticBeamColumn 20 14 15 20.1 29000.0 1830.0 1; # W24X68
element elasticBeamColumn 21 15 16 20.1 29000.0 1830.0 1; # W24X68

# Create a Plain load pattern with a linear TimeSeries:
#荷重パターンを作成。
#    command pattern Plain $tag $timeSeriesTag { $loads }
#梁に分布荷重で与える。
pattern Plain 1 1 {
    eleLoad -ele 13 14 15 -type -beamUniform -$w1
    eleLoad -ele 16 17 18 -type -beamUniform -$w2
    eleLoad -ele 19 20 21 -type -beamUniform -$w3
}
```

**分析 - 重力加速度荷重**

重力負荷解析を実行するコマンドを示します。モデルは弾性で線形アルゴリズムを使用し、負荷の制御のステップを使用して、必要な負荷レベルを取得します。

```
# Create the system of equation
system BandSPD

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

# Create the constraint handler, a Plain handler is used as homo constraints
constraints Plain

# Create the integration scheme, the LoadControl scheme using steps of 1.0
integrator LoadControl 1.0

# Create the solution algorithm, a Linear algorithm is created
algorithm Linear

# create the analysis object
analysis Static
```

**重力の分析を実行します。**

モデルオブジェクト作成後、分析と出力が定義されて解析を実行します。

```
analyze 1
```

**確認するために結果を画面に表示します。**

レコーダーを使用することに加えて、結果は、print と puts コマンドを使って指定できます。ファイル識別子が指定されていない場合は、これらのコマンドの結果が画面に表示されます。**nodeReaction** コマンドを使用して、各節点の挙動を返します。tcl **lindex** コマンドはこれらのリストから値を取得するために使用します。

```
# invoke command to determine nodal reactions
#節点の挙動を確認する。
reactions

set node1Rxn [nodeReaction 1];
# nodeReaction command returns nodal reactions for specified node in a list
# nodeReaction command でリスト内にある指定した節点の挙動を返す。
set node2Rxn [nodeReaction 2]
set node3Rxn [nodeReaction 3]
set node4Rxn [nodeReaction 4]

set inputedFy   [expr -$Load1-$Load2-$Load3]; # loads added negative Fy direction to ele
set computedFx [expr [lindex $node1Rxn 0]+[lindex $node2Rxn 0]+[lindex $node3Rxn 0]+[lindex $node4Rxn 0]]
set computedFy [expr [lindex $node1Rxn 1]+[lindex $node2Rxn 1]+[lindex $node3Rxn 1]+[lindex $node4Rxn 1]]

#   0 が X 方向、1 が Y 方向
#   lindex はデータを引用するときに必要

puts "¥nEqilibrium Check After Gravity:"
puts "SumX: Inputed: 0.0 + Computed: $computedFx = [expr 0.0+$computedFx]"
puts "SumY: Inputed: $inputedFy + Computed: $computedFy = [expr $inputedFy+$computedFy]"
```

**水平荷重を加えます。**

モデルに水平荷重を加えます。

最初は **仮定定数重力加速度荷重を設定**する必要があります。

すなわち、モデルへの負荷の大きさを変更しないために。

節点にかける荷重のロードパターンを作成します。

```
# set gravity loads constant and time in domain to 0.0
loadConst -time 0.0

timeSeries Linear 2
pattern Plain 2 2 {
    load 13 220.0 0.0 0.0
    load  9 180.0 0.0 0.0
    load  5  90.0 0.0 0.0
}
```

**レコーダー**

下層の柱に力を記録するために、要素レコーダーを作成します。

```
recorder Element -file eleForces.out -ele 1 4 7 10 forces
```

**水平荷重解析を実行します。**

モデル、分析と出力が定義されたので分析を実行します。

```
analyze 1
```

**確認するために結果を画面に表示します。**

レコーダーを使用することに加えて、結果は、print と puts コマンドを使って指定できます。ファイル識別子が指定されていない場合は、これらのコマンドの結果が画面に表示されます。

```
reactions

set node1Rxn [nodeReaction 1];
# nodeReaction command returns nodal reactions for specified node in a list
# nodeReaction command でリスト内にある指定した節点の挙動を返す。
set node2Rxn [nodeReaction 2]
set node3Rxn [nodeReaction 3]
set node4Rxn [nodeReaction 4]

set inputedFx  [expr 220.0+180.0+90.0]
set computedFx [expr [lindex $node1Rxn 0]+[lindex $node2Rxn 0]+[lindex $node3Rxn
```

```
0]+[lindex $node4Rxn 0]]
set computedFy [expr [lindex $node1Rxn 1]+[lindex $node2Rxn 1]+[lindex $node3Rxn 1]+[lindex $node4Rxn 1]]

#  lindex はデータを引用するときに必要
#  0 が X 方向、1 が Y 方向

puts "¥nEqilibrium Check After Lateral Loads:"
puts "SumX: Inputed: $inputedFx + Computed: $computedFx = [expr $inputedFx+$computedFx]"
puts "SumY: Inputed: $inputedFy + Computed: $computedFy = [expr $inputedFy+$computedFy]"

# print ele information for columns at bottom
#下部の柱の情報を出力する。
print ele 1 4 7 10
```

**固有値の計算。**

水平荷重解析が完了した後、構造物の周期を確認します。

これを行うには、固有値を取得するコマンド使用します。これらは、tcl **リスト**に返されます。リストから tcl **lindex** コマンドを使用して、固有値を取得し、 そして **expr** コマンドを使用して、周期を決定します。

```
set eigenValues [eigen 5]

puts "¥nEigenvalues:"
set eigenValue [lindex $eigenValues 0]
puts "T[expr 0+1] = [expr 2*$PI/sqrt($eigenValue)]"
set eigenValue [lindex $eigenValues 1]
puts "T[expr 1+1] = [expr 2*$PI/sqrt($eigenValue)]"
set eigenValue [lindex $eigenValues 2]
puts "T[expr 2+1] = [expr 2*$PI/sqrt($eigenValue)]"
set eigenValue [lindex $eigenValues 3]
puts "T[expr 3+1] = [expr 2*$PI/sqrt($eigenValue)]"
set eigenValue [lindex $eigenValues 4]
puts "T[expr 4+1] = [expr 2*$PI/sqrt($eigenValue)]"
```

# create a recorder to record eigenvalues at all free nodes
#固定されてないすべての節点の固有値を記録するためのレコーダーを作成
recorder Node -file eigenvector.out -nodeRange 5 16 -dof 1 2 3 eigen 0

# record the results into the file
#結果をファイルに記録する。
record

**検索結果**

このスクリプトを実行すると、結果が以下のように出ます。