Generalized Single Degree Of Freedom Method For Wind Tower Structure

Initialization Code (optional)

Manipulate

```
Manipulate[
 gTick;

 Module[{finalResult, g, m = 10},

  If[setIC == True,
   setIC = False;
   nSteps = 0;
   currentTime = 0;
   {effectiveLengthOfTower, totalMass, effectiveMass, effectiveFlexuralStiffness, omega, result} =
    updateResult[data, shapeFunction, x, L]
  ];

  If[runningState == "RUNNING" || runningState == "STEP",
   currentTime = currentTime + delT;
   nSteps = nSteps + 1
  ];

  g = getPosition[omega, currentTime, plotType, 2, result];
  finalResult = Grid[{
      {
       Grid[{
         {"estimated natural frequency", padIt2[omega / (2 Pi), {5, 4}], "hz"},
         {"effective flexural stiffness", padIt2[effectiveFlexuralStiffness, {10, 0}], "N/m"},
         {"effective mass", effectiveMass, "kg"},
         {"actual mass", totalMass, "kg"},
         {"mass ratio", (effectiveMass / totalMass) * 100, "percent"},
         {"tower height", effectiveLengthOfTower , "meter"}
        }, Spacings → {.5, .5}, Alignment → Left,
        Frame → All, FrameStyle -> Directive[Thickness[.001], Gray]], SpanFromLeft
      }
      ,
      {
       If[plotType == "2D",
        Deploy@Graphics[g, PlotRange → {{-m, m}, {0, 1.05 * effectiveLengthOfTower}}, ImageSize → imsize]
        ,
        Deploy@Graphics3D[{Opacity[1], EdgeForm[Opacity[1.3]], g},
          ImageSize → imsize, Boxed → False, AxesOrigin → {0, 0, 0}, PlotRange → {{-m / 2, m / 2}, {-m, m}, All},
          Axes → False, AxesLabel → None, PreserveImageOptions → False, Ticks → False,
          TicksStyle → Directive[Black, 8], ImagePadding → 0, ImageMargins → 0, BoxRatios → {1, 1, 7}
         ]
       ], SpanFromLeft}
    }, Alignment → Center, Frame → All, FrameStyle -> Directive[Thickness[.001], Gray]];

  Which[runningState == "RUNNING" || runningState == "STEP",
   If[runningState == "RUNNING", gTick += del]
  ];
```

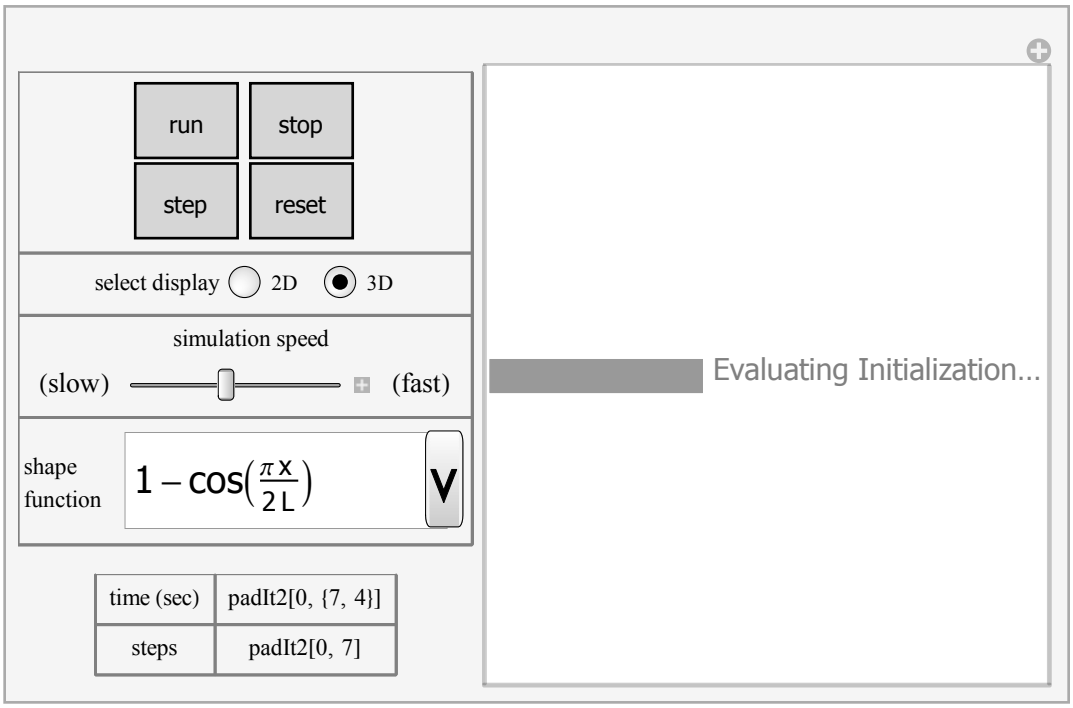
```
 Text@finalResult
]
,
Text@Grid[{
   {
    Grid[{
      {
       Grid[{
          {Button[Style["run", 11], {runningState = "RUNNING"; gTick += del}, ImageSize -> {48, 35}],
           Button[Style["stop", 11], {runningState = "STOP"; gTick += del}, ImageSize -> {48, 35}]},
          {Button[Style["step", 11], {runningState = "STEP"; gTick += del}, ImageSize -> {48, 35}],
           Button[Style["reset", 11],
            {setIC = True; runningState = "STOP"; gTick += del}, ImageSize -> {48, 35}]
          }
         }, Spacings → {0.4, .2}, Alignment → Center
       ], SpanFromLeft
      }
      ,
      {
       Grid[{
          {
           Text@Style["select display", 11],
           RadioButtonBar[Dynamic[plotType, {plotType = #; gTick += del} &],
            {"2D" → Style["2D", 10], "3D" → Style["3D", 10]
            }]
          }
         }, Spacings → {0, 0}
       ], SpanFromLeft
      },
      {
       Grid[{
          {"", Text@Style["simulation speed", 11], ""},
          {"(slow)", Manipulator[Dynamic[delT, {delT = #} &],
            {0.01, 0.1, 0.01}, ImageSize → Small, ContinuousAction -> False], "(fast)"}
         }], SpanFromLeft
      },
      {
       With[{lambda = {1.87510407, 4.6940913, 7.85475744, 10.99554073, 14.13716839},
          gama = {0.734095514, 1.018467319, 0.999224497, 1.00003355, 0.999998550}},
         Grid[{
            {Text@Style[Column[{"shape", "function"}], 11],
             PopupMenu[Dynamic[shapeFunction, {shapeFunction = #; gTick += del; setIC = True
                } &],
              {
               1/2 (Cosh[lambda[[1]] x / L] - Cos[lambda[[1]] x / L] -
                    gama[[1]] (Sinh[lambda[[1]] x / L] - Sin[lambda[[1]] x / L])) → TraditionalForm[f_1[x]],

               1/2 (Cosh[lambda[[2]] x / L] - Cos[lambda[[2]] x / L] -
                    gama[[2]] (Sinh[lambda[[2]] x / L] - Sin[lambda[[2]] x / L])) -> TraditionalForm[f_2[x]],
```

```
1 / 2 (Cosh[lambda[[3]] x / L] - Cos[lambda[[3]] x / L] -
     gama[[3]] (Sinh[lambda[[3]] x / L] - Sin[lambda[[3]] x / L])) -> TraditionalForm[f_3[x]],

1 / 2 (Cosh[lambda[[4]] x / L] - Cos[lambda[[4]] x / L] -
     gama[[4]] (Sinh[lambda[[4]] x / L] - Sin[lambda[[4]] x / L])) -> TraditionalForm[f_4[x]],
1 - Cos[Pi x / (2 L)] -> TraditionalForm[1 - Cos[Pi x / (2 L)]],

(3 L x^2 - x^3) / (2 L^3) -> TraditionalForm[(3 L x^2 - x^3) / (2 L^3)],

(x^4 - 4 L x^3 + 6 L^2 x^2) / (3 L^4) -> TraditionalForm[(x^4 - 4 L x^3 + 6 L^2 x^2) / (3 L^4)],
x^2 / L^2 -> TraditionalForm[x^2 / L^2]
}, ImageSize -> All]
}
}
]
], SpanFromLeft
}
}, Frame → All, FrameStyle -> Directive[Thickness[.001], Gray], Alignment → Center, Spacings → {.4, .8}
]
},
{
Grid[{
{Style["time (sec)", 11], Style[Dynamic@padIt2[currentTime, {7, 4}], 11]},
{Text@Style["steps", 11], Style[Dynamic@padIt2[nSteps, 7], 11]}
}, Spacings → {1, .8}, Frame → All, FrameStyle -> Directive[Thickness[.001], Gray]
]
}
}, Alignment → Center, Spacings → {0, 1}],

{{imsize, {150, 310}}, None},
{{delT, 0.05}, None},
{{gTick, 0}, None},
{{del, $MachineEpsilon}, None},
{{currentTime, 0}, None},
{{nSteps, 0}, None},
{{shapeFunction, 1 - Cos[Pi x / (2 L)]}, None},
{{setIC, True}, None},
{{runningState, "STOP"}, None},
{{plotType, "3D"}, None},
{{effectiveLengthOfTower, 0}, None},
{{totalMass, 0}, None},
{{effectiveMass, 0}, None},
{{effectiveFlexuralStiffness, 0}, None},
{{omega, 0}, None},
{{result, 0}, None},
```

```
ControlPlacement → Left,
SynchronousUpdating → False,
SynchronousInitialization → False,
ContinuousAction → False,
Alignment -> Center,
ImageMargins → 0,
FrameMargins → 0,
Paneled → True,
Frame → False,
AutorunSequencing → {1},
LocalizeVariables → True,
TrackedSymbols :→ {gTick},
Initialization :→
 {
  (*definitions used for parameter checking*)
  integerStrictPositive = (IntegerQ[#] && # > 0 &);
  integerPositive = (IntegerQ[#] && # ≥ 0 &);
  numericStrictPositive = (Element[#, Reals] && # > 0 &);
  numericPositive = (Element[#, Reals] && # ≥ 0 &);
  numericStrictNegative = (Element[#, Reals] && # < 0 &);
  numericNegative = (Element[#, Reals] && # ≤ 0 &);
  bool = (Element[#, Booleans] &);
  numeric = (Element[#, Reals] &);
  integer = (Element[#, Integers] &);
  (*-------------------------------------------*)
  (* helper function for formatting           *)
  (*-------------------------------------------*)
  padIt1[v_?numeric, f_List] :=
   AccountingForm[Chop[v] , f, NumberSigns → {"-", "+"}, NumberPadding → {"0", "0"}, SignPadding → True];
  (*-------------------------------------------*)
  (* helper function for formatting           *)
  (*-------------------------------------------*)
  padIt2[v_?numeric, f_List] :=
   AccountingForm[v , f, NumberSigns → {"", ""}, NumberPadding → {"0", "0"}, SignPadding → True];
  padIt2[v_?numeric, f_Integer] := AccountingForm[Chop[v] , f, NumberSigns → {"", ""},
    NumberPadding → {"0", "0"}, SignPadding → True];

  (*-------------------------------------------------------------------*)
  updateResult[data_, shapeFunction_, x_, L_] :=
   Module[{i, nRows, numberOfColumns = 13, kID = 1, kH = 2, kT = 3, kD = 4, kMASS = 5, kE = 6, kFLEXEFFECTIVE = 7,
     kCURRENTH = 8, kPHI = 9, kCURVETURE = 10, kANGLE = 11, kI = 12, kMEFFECTIVE = 13, result, lengthOfTower,
     totalMass, effectiveMass, effectiveFlexuralStiffness, effectiveStiffness, omega},

    nRows = Length[data];
    result = Table[0, {nRows}, {numberOfColumns}];
    result[[All, 2 ;; 6]] = data;
    result[[All, 2 ;; 4]] = result[[All, 2 ;; 4]] / 1000;
    result[[All, kE]] = result[[All, kE]] * 10^6;
    lengthOfTower = Total[result[[All, kH]]];
    totalMass = Total[result[[All, kMASS]]];

    Do[
     result[[i, kID]] = i;
     If[i == nRows, result[[i, kCURRENTH]] = result[[i, kH]],
      result[[i, kCURRENTH]] = result[[i + 1, kCURRENTH]] + result[[i, kH]]
     ];

     If[i > 1, (*top mass not part of the following computation*)
      result[[i, kPHI]] = shapeFunction /. {x -> result[[i, kCURRENTH]], L → lengthOfTower};
      result[[1, kMEFFECTIVE]] = result[[1, kMASS]] * result[[i, kPHI]]^2;
      result[[i, kCURVETURE]] =
       D[shapeFunction, {x, 2}] /. {x → result[[i, kCURRENTH]], L → lengthOfTower};
```

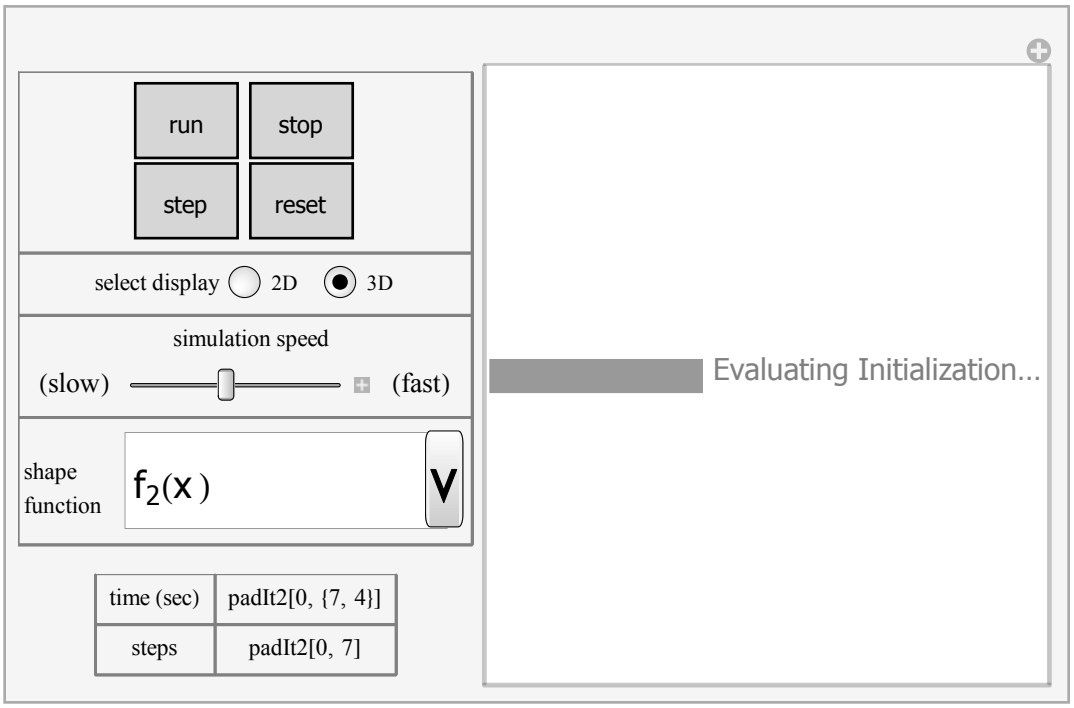
```
    result[[i, kANGLE]] = result[[i, kCURVETURE]] * result[[i, kH]];
    result[[i, kMEFFECTIVE]] = result[[i, kMASS]] * result[[i, kPHI]] ^ 2;
    result[[i, kI]] = momentOfInertia[result[[i, kT]], result[[i, kD]]];
    result[[i, kFLEXEFFECTIVE]] =
     result[[i, kI]] * result[[i, kE]] * result[[i, kCURVETURE]] * result[[i, kANGLE]]

    , (*below for top mass*)
    result[[i, kFLEXEFFECTIVE]] = result[[1, kMASS]];
    result[[1, kPHI]] = shapeFunction /. {x -> result[[1, kCURRENTH]], L → lengthOfTower};
   ]
   ,
   {i, nRows, 1, -1}
  ];

  effectiveMass = Total[result[[All, kMEFFECTIVE]]];
  effectiveFlexuralStiffness = Total[result[[All, kFLEXEFFECTIVE]]];
  effectiveStiffness = Total[result[[All, kFLEXEFFECTIVE]]];
  omega = Sqrt[effectiveStiffness / effectiveMass];
  {lengthOfTower, totalMass, effectiveMass, effectiveFlexuralStiffness, omega, result}
 ];

(*--------------------------------------------------------------------*)
getPosition[omega_, t_, plotType_String, scale_, result_] := Module[{i, g, color = LightBlue, cx, cz,
   cy, leftLowerCorner, rightLowerCorner, rightTopCorner, leftTopCorner, dia, kH = 2, kD = 4,
   kCURRENTH = 8, kPHI = 9, nRows},

  nRows = Length[result];
  g = Table[0, {nRows}];

  If[plotType == "2D",
   Do[
    cx = result[[i, kPHI]] * scale * Cos[omega * t];
    cy = result[[i, kCURRENTH]];

    If[i == 1,
     dia = result[[2, kD]] * 4,
     dia = result[[i, kD]]
    ];

    leftLowerCorner = {cx - dia / 2, cy - result[[i, kH]]};
    rightLowerCorner = {cx + dia / 2, cy - result[[i, kH]]};
    rightTopCorner = {cx + dia / 2, cy};
    leftTopCorner = {cx - dia / 2, cy};

    If[i == nRows || i == 1,
     color = Red,
     If[result[[i, kH]] < 1,
      color = Red,
      color = Black
     ]
    ];

    g[[i]] = {EdgeForm[{Thin, color}], FaceForm[],
      Polygon[{ leftLowerCorner, rightLowerCorner, rightTopCorner, leftTopCorner}]}

    , {i, 1, nRows}
   ]
   ,
   Do[
    cx = result[[i, kPHI]] * scale * Cos[omega * t];
    cz = result[[i, kCURRENTH]];

    If[i == 1,
     g[[1]] = {Red, Cylinder[{{cx, 0, cz - result[[1, kH]]}, {cx, 0, cz}}, result[[2, kD]]]}
```

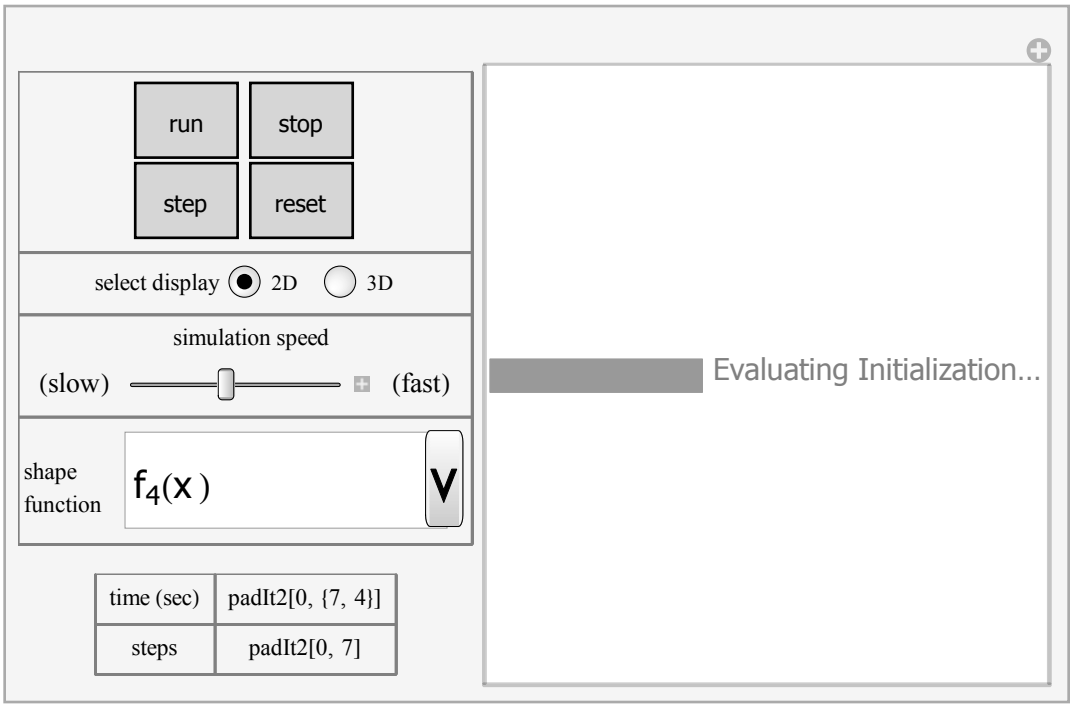
```
      ,

      If[i == nRows,
       color = Red,
       If[result[[i, kH]] < 1,
        color = Black,
        color = LightBlue
       ]
      ];

      If[i == nRows,
       g[[i]] =
        {color, Opacity[.9], Cylinder[{{0, 0, cz - result[[i, kH]]}, {0, 0, cz}}, result[[i, kD]] / 2]},
       g[[i]] = {color, Opacity[.9], Cylinder[{{cx, 0, cz - result[[i, kH]]}, {cx, 0, cz}},
          result[[i, kD]] / 2]}
      ]

     ]
     , {i, 1, nRows}
    ]
   ];
   g
  ];
(*-----------------------------------------------------------------------*)
momentOfInertia[t_, d_] := Module[{r2 = d / 2, r1},
  r1 = r2 - t;
  Pi * (r2^4 - r1^4) / 4
 ];
(*-----------------------------------------------------------------------*)
(*data=Import["data.xlsx",{"Sheets",1}]*)
(*this is the actual wind tower data*)
data = {{1340.`, 0.`, 0.`, 130000.`, 210000.`}, {295.`, 121.`, 2800.`, 2374.8262835560467`, 210000.`},
  {2300.`, 15.`, 2800.`, 2386.1406618751685`, 210000.`}, {2940.`, 15.`, 2822.`,
   3062.1786695954443`, 210000.`}, {2940.`, 15.`, 2844.`, 3086.2771505855408`, 210000.`},
  {2935.`, 15.`, 2868.`, 3106.171133487974`, 210000.`}, {2935.`, 15.`, 2890.`,
   3131.317891220979`, 210000.`}, {2935.`, 15.`, 2912.`, 3155.3713116611934`, 210000.`},
  {2935.`, 16.`, 2934.`, 3390.2201462719627`, 210000.`}, {2930.`, 17.`, 2956.`,
   3621.949609431842`, 210000.`}, {2930.`, 18.`, 2978.`, 3862.510622879164`, 210000.`},
  {2925.`, 19.`, 3000.`, 4099.092018136769`, 210000.`}, {280.`, 180.`, 3000.`,
   3529.647958691686`, 210000.`}, {2885.`, 20.`, 3052.`, 4307.746324352069`, 210000.`},
  {2885.`, 20.`, 3124.`, 4396.595429624945`, 210000.`}, {2880.`, 21.`, 3196.`,
   4715.074344034012`, 210000.`}, {2880.`, 21.`, 3268.`, 4823.225197872331`, 210000.`},
  {2880.`, 22.`, 3340.`, 5164.629736275286`, 210000.`}, {2875.`, 22.`, 3412.`,
   5268.768980113425`, 210000.`}, {2875.`, 22.`, 3484.`, 5381.873202425646`, 210000.`},
  {2870.`, 23.`, 3556.`, 5733.120246174896`, 210000.`}, {2870.`, 23.`, 3628.`,
   5851.159957727482`, 210000.`}, {2860.`, 23.`, 3700.`, 5947.936112191075`, 210000.`},
  {330.`, 230.`, 3700.`, 6540.681440845615`, 210000.`}, {2710.`, 24.`, 3760.`,
   5986.441195326227`, 210000.`}, {2710.`, 24.`, 3825.`, 6087.399843179338`, 210000.`},
  {2710.`, 24.`, 3890.`, 6192.396836946592`, 210000.`}, {2705.`, 25.`, 3955.`, 6546.00438555125`,
   210000.`}, {2705.`, 25.`, 4020.`, 6655.174489988609`, 210000.`}, {2705.`, 25.`, 4085.`,
   6764.344594425926`, 210000.`}, {2685.`, 26.`, 4150.`, 7093.359001961084`, 210000.`},
  {360.`, 240.`, 4150.`, 8389.619160172944`, 210000.`}, {2410.`, 26.`, 4150.`,
   6417.424886453375`, 210000.`}, {2410.`, 27.`, 4150.`, 6662.63295330221`, 210000.`},
  {2410.`, 28.`, 4150.`, 6907.721318873599`, 210000.`}, {2410.`, 29.`, 4150.`,
   7152.689983167541`, 210000.`}, {2405.`, 29.`, 4150.`, 7137.850377393334`, 210000.`},
  {2405.`, 30.`, 4150.`, 7382.1913550093805`, 210000.`}, {440.`, 390.`, 4150.`,
   16023.481209298769`, 210000.`}, {2400.`, 31.`, 4150.`, 7610.557551458674`, 210000.`},
  {2400.`, 32.`, 4150.`, 7854.152134490976`, 210000.`}, {2395.`, 34.`, 4150.`,
   8323.606637433179`, 210000.`}, {2395.`, 60.`, 4150.`, 14595.93172144645`, 210000.`},
  {2395.`, 60.`, 4150.`, 14595.93172144645`, 210000.`}, {240.`, 400.`, 4150.`,
   8940.344373585833`, 210000.`}, {700.`, 55.`, 4150.`, 3915.312064107245`, 210000.`}};

}
```



Caption

This demonstration shows how to use different shape functions to estimate the natural frequency for an actual industrial wind tower using the method of generalized single degree of freedom. The data for the structure of the wind tower was made available and used to calculate the effective flexural stiffness and the effective mass to estimate the natural frequency of the tower. The shape function which produces the lowest natural frequency will be the best approximation of the of the motion of the tower in free vibration mode.





Details (optional)

The method of generalized single degree of freedom is used to model a multi-degree of freedom system as one degree of freedom system in order to obtain an estimate of the (first) undamped natural frequency of the original system. A shape function is assumed for the solution and this shape function is then used to obtain the effective stiffness and effective mass of the single undamped degree of freedom model in order to obtain the natural frequency estimate.
This demonstration uses a number of shape functions. The shape function which produces the lowest natural frequency estimate will be the best approximation to the actual solution. You select the shape function from the pull-down menu, and the natural frequency is

calculated and displayed at the top of the main display area.
The author's report contains more information about this method.

References

[1] Mario Paz, William Leigh, Structural Dynamics, Boston: Kluwer academic publishers, 2004.

[2] Ray W. Clough, Joseph Penzien, Dynamics of structures, New Jersey: McGraw-Hill,1975.

Control Suggestions (optional)

- [x] Resize Images
- [ ] Rotate and Zoom in 3D
- [ ] Drag Locators
- [ ] Create and Delete Locators
- [ ] Slider Zoom
- [ ] Gamepad Controls
- [x] Automatic Animation
- [ ] Bookmark Animation

Search Terms (optional)

single degree of freedom

vibration

wind tower

Related Links (optional)

Authoring Information

Contributed by: Nasser M. Abbasi