In[1]:=
```
Manipulate[
 (*by Nasser M. Abbasi,
 simple rigid frame solution by direct stiffness method, 6/17/2015*)
 tick;
 Module[{dL0 = 10, L0 = len * 12, kElement, T0, k, i, j,
   ele, theta, globalK, force, mglobalK, I013, I02, coord, frame},
  coord = {{0, 0}, {0, dL0}, {0, 2 dL0}, {dL0, 2 dL0}, {dL0, dL0},
    {dL0, 0}, {2 dL0, 0}, {2 dL0, dL0}, {2 dL0, 2 dL0}};
  frame = {Line[{coord[[1]], coord[[3]]}], Line[{coord[[6]], coord[[4]]}],
    Line[{coord[[7]], coord[[9]]}], Line[{coord[[2]], coord[[8]]}],
    Line[{coord[[3]], coord[[9]]}]};

  (*make element stiffness matrix*)
  kElement = getElementMatrix[theta, I0, L0, E0 * 10^6, A0];
  For[i = 2, i <= Length[kElement], i++,
   For[j = 1, j ≤ i - 1, j++,
    kElement[[i, j]] = kElement[[j, i]]
   ]
  ];

  (*build the global stiffness matrix, using con, which is connectivity matrix*)
  globalK = Table[0, {i, 3 * 9}, {j, 3 * 9}];
  For[k = 1, k ≤ 10, k++, (*10 elements only*)
   T0 = con[[k]];
   ele = kElement /. theta → angles[[k]];
   (*adjust element stiffness matrix for angle*)
   (*this below adds the element to the global stiffness matrix *)
   For[i = 1, i ≤ 6, i++,
    For[j = 1, j ≤ 6, j++,
     globalK[[T0[[i]], T0[[j]]]] += ele[[i, j]]
    ]
   ]
  ];

  force = Table[0, {3 * 9}];
  force[[4]] = f2x;
  force[[7]] = f3x;
  force[[6]] = m2;
  force[[9]] = m3;
  force[[8]] = f3y;
  force[[11]] = f4y;
  force[[26]] = f9y;
```

```
(*Now adjust the global stiffness matrix for boundary conditions,
keep old copy for later use*)
mglobalK = globalK;
mglobalK[[1, ;;]] = 0; mglobalK[[ ;; , 1]] = 0; mglobalK[[1, 1]] = 1;
mglobalK[[2, ;;]] = 0; mglobalK[[ ;; , 2]] = 0; mglobalK[[2, 2]] = 1;
mglobalK[[3, ;;]] = 0; mglobalK[[ ;; , 3]] = 0; mglobalK[[3, 3]] = 1;
mglobalK[[16, ;;]] = 0;
mglobalK[[ ;; , 16]] = 0;
mglobalK[[16, 16]] = 1;
mglobalK[[17, ;;]] = 0;
mglobalK[[ ;; , 17]] = 0;
mglobalK[[17, 17]] = 1;
mglobalK[[18, ;;]] = 0;
mglobalK[[ ;; , 18]] = 0;
mglobalK[[18, 18]] = 1;
mglobalK[[19, ;;]] = 0;
mglobalK[[ ;; , 19]] = 0;
mglobalK[[19, 19]] = 1;
mglobalK[[20, ;;]] = 0;
mglobalK[[ ;; , 20]] = 0;
mglobalK[[20, 20]] = 1;
mglobalK[[21, ;;]] = 0;
mglobalK[[ ;; , 21]] = 0;
mglobalK[[21, 21]] = 1;

sol = LinearSolve[mglobalK, force];
(force = globalK.sol); (*Now solve back for forces,
this finds the reactions now for free*)
(*Print[MatrixForm[N@force]];*)

Grid[{
  {
   Graphics[
    {{Thick, frame},
     Rectangle[{-0.1 dL0, -0.01 dL0}, {0.1 dL0, 0.01 dL0}],
     Rectangle[{0.9 dL0, -0.01 dL0}, {1.1 dL0, 0.01 dL0}],
     Rectangle[{1.9 dL0, -0.01 dL0}, {2.1 dL0, 0.01 dL0}],


     addNodeLabel[coord[[1]], dL0, "1"],
     addNodeLabel[coord[[2]], dL0, "2"],
     addNodeLabel[coord[[3]], dL0, "3"],
```

```
addNodeLabel[coord[[4]], dL0, "4"],
addNodeLabel[coord[[5]], dL0, "5"],
addNodeLabel[coord[[6]], dL0, "6"],
addNodeLabel[coord[[7]], dL0, "7"],
addNodeLabel[coord[[8]], dL0, "8"],
addNodeLabel[coord[[9]], dL0, "9"],

(*add applied loads*)
addHorizontalForceArrow[coord[[2]], dL0, N@f2x, Blue, "startLeft"],
addHorizontalForceArrow[coord[[3]], dL0, N@f3x, Blue, "startLeft"],
addVerticalForceArrow[coord[[3]], dL0, N@f3y, Blue, "startAbove"],
addVerticalForceArrow[coord[[4]], dL0, N@f4y, Blue, "startAbove"],
addVerticalForceArrow[coord[[9]], dL0, N@f9y, Blue, "startAbove"],

(*add reactions*)
addHorizontalForceArrow[coord[[1]], dL0, N@force[[1]], Red, "startLeft"],
addVerticalForceArrow[coord[[1]], dL0, N@force[[2]], Red, "startBelow"],
addHorizontalForceArrow[
 coord[[6]], dL0, N@force[[16]], Red, "startRight"],
addVerticalForceArrow[coord[[6]], dL0, N@force[[17]], Red, "startBelow"],
addHorizontalForceArrow[
 coord[[7]], dL0, N@force[[19]], Red, "startRight"],
addVerticalForceArrow[coord[[7]], dL0, N@force[[20]], Red, "startBelow"],

(*add moments*)
addMoment[coord[[1]], dL0, force[[3]], Red],
addMoment[coord[[6]], dL0, force[[18]], Red],
addMoment[coord[[7]], dL0, force[[21]], Red],

addMoment[coord[[2]], dL0, force[[6]], Blue],
addMoment[coord[[3]], dL0, force[[9]], Blue],


If[showDeflection,
 {
  {Red, Dashed, Line[{
     coord[[1]],
     coord[[2]] + {exgH * sol[[4]], (exgV * sol[[5]])},
     coord[[3]] + {exgH * sol[[7]], (exgV * sol[[8]])},
     coord[[4]] + {exgH * sol[[10]], (exgV * sol[[11]])},
     coord[[9]] + {exgH * sol[[25]], (exgV * sol[[26]])},
     coord[[8]] + {exgH * sol[[22]], (exgV * sol[[23]])},
     coord[[7]]}]},
```

```
          {Red, Dashed, Line[{
             coord[[6]],
             coord[[5]] + {exgH * sol[[13]], (exgV * sol[[17]])},
             coord[[4]] + {exgH * sol[[10]], (exgV * sol[[11]])}
            }]
          }
         }
        ]

       }, PlotRange → {{-7, 27}, {-5, 24}}, ImageSize → 450
      ]
     }
    }]
],

Style[Text@Grid[{
     {"Element Length (ft)", Manipulator[Dynamic[len, {len = #;
          tick = Not[tick]} &], {9, 11, .1}, ImageSize → Tiny],
      Dynamic[padIt2[len, {2, 1}]]},
     {"Horizontal force at node 2", Manipulator[Dynamic[f2x, {f2x = #;
          tick = Not[tick]} &], {-20 000, 20 000, 10}, ImageSize → Tiny],
      Dynamic[padIt1[f2x, 5]]},
     {"Horizontal force at node 3", Manipulator[Dynamic[f3x, {f3x = #;
          tick = Not[tick]} &], {-20 000, 20 000, 10}, ImageSize → Tiny],
      Dynamic[padIt1[f3x, 5]]},
     {"Vertical force at node 3", Manipulator[Dynamic[f3y, {f3y = #;
          tick = Not[tick]} &], {-20 000, 20 000, 10}, ImageSize → Tiny],
      Dynamic[padIt1[f3y, 5]]},
     {"Vertical force at node 4", Manipulator[Dynamic[f4y, {f4y = #;
          tick = Not[tick]} &], {-20 000, 20 000, 10}, ImageSize → Tiny],
      Dynamic[padIt1[f4y, 5]]},
     {"Vertical force at node 9", Manipulator[Dynamic[f9y, {f9y = #;
          tick = Not[tick]} &], {-20 000, 20 000, 10}, ImageSize → Tiny],
      Dynamic[padIt1[f9y, 5]]},
     {"moment at node 2", Manipulator[Dynamic[m2, {m2 = #;
          tick = Not[tick]} &], {-10 000, 10 000, 10}, ImageSize → Tiny],
      Dynamic[padIt1[m2, 5]]},
     {"moment at node 3", Manipulator[Dynamic[m3, {m3 = #;
          tick = Not[tick]} &], {-10 000, 10 000, 10}, ImageSize → Tiny],
      Dynamic[padIt1[m3, 5]]},
     {Grid[{
        {"I (inch^4)", Manipulator[Dynamic[I0,
           {I0 = #;
```

```
            tick = Not[tick]} &],
          {10, 500, 1}, ImageSize → Tiny], Dynamic[padIt2[I0, 3]]},
        {"A (corss section area, inch^2)", Manipulator[Dynamic[A0,
           {A0 = #;
             tick = Not[tick]} &],
          {1, 100, 1}, ImageSize → Tiny], Dynamic[padIt2[A0, 3]]},
        {"E (10^6 psi)", Manipulator[Dynamic[E0,
           {E0 = #;
             tick = Not[tick]} &],
          {5, 50, 1}, ImageSize → Tiny], Dynamic[padIt2[E0, 2]]}
       }, Frame → True], SpanFromLeft
    },
    {Grid[{
       {"show deflection", Checkbox[Dynamic[showDeflection, {showDeflection = #;
            tick = Not[tick]} &]]},
       {"Exaggeration factor (horizontal)", Manipulator[Dynamic[exgH,
           {exgH = #;
             tick = Not[tick]} &], {1, 10, 1}, ImageSize → Tiny],
        Dynamic[padIt2[exgH, 2]]},
       {"Exaggeration factor (horizontal)", Manipulator[Dynamic[exgV,
           {exgV = #;
             tick = Not[tick]} &],
          {1, 1000, 1}, ImageSize → Tiny], Dynamic[padIt2[exgV, 3]]}
       }, Frame → True], SpanFromLeft
    }
   }, Alignment → Left, Frame → True], Medium],
Style[Text@Grid[{
    {"Solution: Displacements and rotations solution", SpanFromLeft},
    {"node 2 Ux (inch)", Dynamic@padIt1[N@sol[[4]], {5, 4}]},
    {"node 2 Vy (inch)", Dynamic@padIt1[N@sol[[5]], {7, 6}]},
    {"node 2 (angle)", Dynamic@padIt1[sol[[6]] * 180. / Pi, {5, 4}]},
    {"node 3 Ux (inch)", Dynamic@padIt1[N@sol[[7]], {5, 4}]},
    {"node 3 Vy (inch)", Dynamic@padIt1[N@sol[[8]], {7, 6}]},
    {"node 3 (angle)", Dynamic@padIt1[sol[[9]] * 180. / Pi, {5, 4}]},
    {"node 4 Ux (inch)", Dynamic@padIt1[N@sol[[10]], {5, 4}]},
    {"node 4 Vy (inch)", Dynamic@padIt1[N@sol[[11]], {7, 6}]},
    {"node 4 (angle)", Dynamic@padIt1[sol[[12]] * 180. / Pi, {5, 4}]},
    {"node 5 Ux (inch)", Dynamic@padIt1[N@sol[[13]], {5, 4}]},
    {"node 5 Vy (inch)", Dynamic@padIt1[N@sol[[14]], {7, 6}]},
    {"node 5 (angle)", Dynamic@padIt1[sol[[15]] * 180. / Pi, {5, 4}]},
    {"node 8 Ux (inch)", Dynamic@padIt1[N@sol[[22]], {5, 4}]},
    {"node 8 Vy (inch)", Dynamic@padIt1[N@sol[[23]], {7, 6}]},
    {"node 8 (angle)", Dynamic@padIt1[sol[[24]] * 180. / Pi, {5, 4}]},
```

```
    {"node 9 Ux (inch)", Dynamic@padIt1[N@sol[[25]], {5, 4}]},
    {"node 9 Vy (inch)", Dynamic@padIt1[N@sol[[26]], {7, 6}]},
    {"node 9 (angle)", Dynamic@padIt1[sol[[27]] * 180. / Pi, {5, 4}]}
   }, Alignment → Left, Spacings → {.5, .5}, Frame → All], 11],

{{tick, False}, None},
{{showDeflection, True}, None},
{{I0, 100}, None},
{{E0, 30}, None},
{{A0, 10}, None},
{{exgH, 5}, None},
{{exgV, 100}, None},
{{len, 10}, None},
{{f2x, 0}, None},
{{f3x, 1000}, None},
{{f3y, -1000}, None},
{{f4y, -10 000}, None},
{{f9y, -1000}, None},
{{m3, 5000}, None},
{{m2, 1000}, None},
{{sol, Table[0, {3 * 9}]}, None},
{{con, {{1, 2, 3, 4, 5, 6},
   {4, 5, 6, 7, 8, 9},
   {7, 8, 9, 10, 11, 12},
   {4, 5, 6, 13, 14, 15},
   {13, 14, 15, 16, 17, 18},
   {10, 11, 12, 13, 14, 15},
   {10, 11, 12, 25, 26, 27},
   {13, 14, 15, 22, 23, 24},
   {19, 20, 21, 22, 23, 24},
   {22, 23, 24, 25, 26, 27}}}, None}, (*connectivity matrix*)

{{angles, {Pi / 2, Pi / 2, 0, 0, -Pi / 2, -Pi / 2, 0, 0, Pi / 2, Pi / 2}}, None},
TrackedSymbols :⇢ {tick},
SynchronousUpdating → False, ControlPlacement → Left,
Alignment → Center, ImageMargins → 0, FrameMargins → 0,
Initialization :⇢
 (
  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] &);
padIt1[v_?numeric, f_List] := AccountingForm[v, f,
  NumberSigns → {"-", "+"}, NumberPadding → {"0", "0"}, SignPadding → True];
padIt1[v_?numeric, f_Integer] := AccountingForm[Chop[v], f,
  NumberSigns → {"-", "+"}, NumberPadding → {"0", "0"}, SignPadding → True];
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];

getElementMatrix[angle_, I0_, L0_, E0_, A0_] :=
 Module[{c = Cos[angle], s = Sin[angle]},
  E0 / L0 {{A0 c^2 + 12 I0 / L0^2 s^2, (A0 - 12 I0 / L0^2) c * s, -6 I0 / L0 * s,
     -(A0 c^2 + 12 I0 / L0^2 s^2), -(A0 - 12 I0 / L0^2) * c * s, -6 * I0 / L0 * s},
    {0, A0 * s^2 + 12 I0 / L0^2 * c^2, 6 * I0 / L0 * c, -(A0 - 12 I0 / L0^2) c * s,
     -(A0 * s^2 + 12 I0 / L0^2 * c^2), 6 * I0 / L0 * c},
    {0, 0, 4 * I0, 6 * I0 / L0 * s, -6 * I0 / L0 * c, 2 I0},
    {0, 0, 0, A0 * c^2 + 12 I0 / L0^2 s^2, (A0 - 12 I0 / L0^2) c * s, 6 * I0 / L0 * s},
    {0, 0, 0, 0, A0 * s^2 + 12 I0 / L0^2 * c^2, -6 * I0 / L0 * c},
    {0, 0, 0, 0, 0, 4 * I0}}
 ];

(*adds label of node using node coordinates*)
addNodeLabel[{x_, y_}, dL0_, label_] := Module[{},
  Style[Text[label, {x + 0.1 dL0, y - 0.1 dL0}], Red, 16]
 ];

(*draw horizontal force arrow and puts label next to it*)
addHorizontalForceArrow[{x_, y_}, dL0_, value_, color_, start_] := Module[{},
  If[value ≥ $MachineEpsilon,
   If[start == "startLeft",
    {
     {color, Arrow[{{x - 0.3 dL0, y}, {x , y}}]},
     Text[ToString[value], {x - 0.4 dL0, y - 0.05 dL0}]
    },
    {
     {color, Arrow[{{x, y}, {x + 0.3 dL0, y}}]},
     Text[ToString[value], {x + 0.4 dL0, y - 0.05 dL0}]
    }
   ],
```

```
    If[Abs@value > $MachineEpsilon,
     If[start == "startLeft",
      {
       {color, Arrow[{{x, y}, {x - 0.3 dL0, y}}]},
       Text[ToString[Abs@value], {x - 0.4 dL0, y - 0.05 dL0}]
      },
      {
       {color, Arrow[{{x + 0.3 dL0, y}, {x, y}}]},
       Text[ToString[Abs@value], {x + 0.4 dL0, y - 0.05 dL0}]
      }
     ]
    ]
   ]
  ];

addMoment[{x_, y_}, dL0_, value_, color_] := Module[{k},
  If[value > 0.001,
   {
    {color, Arrowheads[Medium], Arrow[BSplineCurve[
       Table[{Cos[k], Sin[k]} + {x, y}, {k, -115 Degree, 170 Degree, 1/5}]]]},
    Text[N@value, {x + 0.15 dL0, y + 0.1 dL0}]
   },
   If[Abs@value > 0.001,
    {
     {color, Arrowheads[Medium], Arrow[BSplineCurve[
        Table[{Cos[k], Sin[k]} + {x, y}, {k, 170 Degree, -115 Degree, -1/5}]]]},
     Text[N@value, {x + 0.15 dL0, y + 0.1 dL0}]
    }
   ]
  ]
 ];

(*draw vertical force arrow and puts label next to it*)
addVerticalForceArrow[{x_, y_}, dL0_, value_, color_, start_] := Module[{},
  If[value ≥ $MachineEpsilon,
   {
    If[start == "startBelow",
     {
      {color, Arrow[{{x, y - 0.3 dL0}, {x , y}}]},
      Text[ToString[value], {x + 0.1 dL0, y - 0.35 dL0}]
     }
     ,
     {
```

```
          {color, Arrow[{{x, y}, {x, y + 0.3 dL0}}]},
          Text[ToString[value], {x + 0.1 dL0, y + 0.35 dL0}]
         }
        ]
       },
       If[Abs@value > $MachineEpsilon,
        If[start == "startBelow",
         {
          {color, Arrow[{{x, y}, {x, y - 0.3 dL0}}]},
          Text[ToString[Abs@value], {x + 0.1 dL0, y - 0.35 dL0}]
         },
         {
          {color, Arrow[{{x, y + 0.3 dL0}, {x, y}}]},
          Text[ToString[Abs@value], {x + 0.1 dL0, y + 0.35 dL0}]
         }
        ]
       ]
      ]
     ]

  )

]
```