```
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, aLoc, bLoc, cLoc, dLoc, eLoc, fLoc, coord, frame},
          coord = {{0, 0}, {0, dL0}, {dL0, dL0}, {dL0, 0}};
          frame = Table[Line[{coord[[i]], coord[[i + 1]]}], {i, 1, Length[coord] - 1}];
          aLoc = {{-0.3 dL0, 0}, {-0.1 dL0, 0}};
          bLoc = {{0, -0.2 dL0}, {0, 0}};
          cLoc = {{-0.3 dL0, dL0}, {0, dL0}};
          dLoc = 0.08 dL0 + {dL0, dL0};
          eLoc = {{0.75 dL0, 0}, {0.89 dL0, 0}};
          fLoc = {{dL0, -0.2 dL0}, {dL0, 0}};

          (*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, 12}, {j, 12}];
          For[k = 1, k ≤ 3, k++, (*3 elements only*)
           T0 = con[[k]];
           If[k == 1 || k == 3,
            ele = kElement /. theta → angles[[k]]
             (*adjust element stiffness matrix for angle*)
             ,
            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 = {0, 0, 0, f2x, f2y, m2, f3x, f3y, m3, 0, 0, 0};

(*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[[10, ;;]] = 0;
mglobalK[[ ;; , 10]] = 0;
mglobalK[[10, 10]] = 1;
mglobalK[[11, ;;]] = 0;
mglobalK[[ ;; , 11]] = 0;
mglobalK[[11, 11]] = 1;
mglobalK[[12, ;;]] = 0;
mglobalK[[ ;; , 12]] = 0;
mglobalK[[12, 12]] = 1;
sol = LinearSolve[mglobalK, force];
(force = globalK.sol); (*Now solve back for forces,
this finds the reactions now for free*)
(*Print[InputForm@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}],



     addNodeLabel[coord[[1]], dL0, "1"],
     addNodeLabel[coord[[2]], dL0, "2"],
     addNodeLabel[coord[[3]], dL0, "3"],
     addNodeLabel[coord[[4]], dL0, "4"],

     (*add applied loads*)
     addHorizontalForceArrow[coord[[2]], dL0, N@f2x, Blue, "startLeft"],
     addVerticalForceArrow[coord[[2]], dL0, N@f2y, Blue, "startAbove"],
     addHorizontalForceArrow[coord[[3]], dL0, N@f3x, Blue, "startRight"],
     addVerticalForceArrow[coord[[3]], dL0, N@f3y, Blue, "startAbove"],

     (*add reactions*)
     addHorizontalForceArrow[coord[[1]], dL0, N@force[[1]], Red, "startLeft"],
      addVerticalForceArrow[coord[[1]], dL0, N@force[[2]], Red, "startBelow"],
      addHorizontalForceArrow[
       coord[[4]], dL0, N@force[[10]], Red, "startRight"],
      addVerticalForceArrow[coord[[4]], dL0, N@force[[11]], Red, "startBelow"],

      (*add moments*)
      addMoment[coord[[1]], dL0, force[[3]], Red],
      addMoment[coord[[4]], dL0, force[[12]], Red],
      addMoment[coord[[2]], dL0, force[[6]], Blue],
      addMoment[coord[[3]], dL0, force[[9]], Blue],

      If[showDeflection,
       {Red, Dashed, Line[{{0, 0}, {exgH * sol[[4]], dL0 + (exgV * sol[[5]])},
          {dL0 + (exgH * sol[[7]]), dL0 + (exgV * sol[[8]])}, {dL0, 0}}]}
      ]

     }, PlotRange → {{-7, 14}, {-5, 13}}, ImageSize → 450
    ]
   }
  }]
],

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]]},
   {"Vertical force at node 2", Manipulator[Dynamic[f2y, {f2y = #;
        tick = Not[tick]} &], {-20 000, 20 000, 10}, ImageSize → Tiny],
    Dynamic[padIt1[f2y, 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]]},
   {"moment at node 3", Manipulator[Dynamic[m3, {m3 = #;
        tick = Not[tick]} &], {-10 000, 10 000, 10}, ImageSize → Tiny],
    Dynamic[padIt1[m3, 5]]},
   {"moment at node 2", Manipulator[Dynamic[m2, {m2 = #;
        tick = Not[tick]} &], {-10 000, 10 000, 10}, ImageSize → Tiny],
     Dynamic[padIt1[m2, 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],
  Text@Grid[{
    {"Solution: Displacements and rotations solution", SpanFromLeft},
    {"node 2 U_x (inch)", Dynamic@padIt1[N@sol[[4]], {5, 4}]},
    {"node 2 V_y (inch)", Dynamic@padIt1[N@sol[[5]], {7, 6}]},
    {"node 2 (angle)", Dynamic@padIt1[sol[[6]] * 180. / Pi, {5, 4}]},
    {"node 3 U_x (inch)", Dynamic@padIt1[N@sol[[7]], {5, 4}]},
    {"node 3 V_y (inch)", Dynamic@padIt1[N@sol[[8]], {7, 6}]},
    {"node 3 (angle)", Dynamic@padIt1[sol[[9]] * 180. / Pi, {5, 4}]}
   }, Alignment → Left, Spacings → {.5, .5}, Frame → All],

  {{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, 10 000}, None},
{{f2y, 0}, None},
{{f3x, 0}, None},
{{f3y, 0}, None},
{{m3, 5000}, None},
{{m2, 0}, None},
{{sol, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}}, None},
{{con, {{1, 2, 3, 4, 5, 6}, {4, 5, 6, 7, 8, 9}, {7, 8, 9, 10, 11, 12}}}, None},
(*connectivity matrix*)

{{angles, {Pi / 2, 0, -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 - 0.2 dL0, y}, {x , y}}]},
     Text[ToString[value], {x - 0.2 dL0, y - 0.05 dL0}]
    },
    {
     {color, Arrow[{{x, y}, {x + 0.2 dL0, y}}]},
     Text[ToString[value], {x + 0.2 dL0, y - 0.05 dL0}]
    }
   ],
   If[Abs@value > $MachineEpsilon,
    If[start == "startLeft",
     {
      {color, Arrow[{{x, y}, {x - 0.2 dL0, y}}]},
      Text[ToString[Abs@value], {x - 0.2 dL0, y - 0.05 dL0}]
     },
     {
      {color, Arrow[{{x + 0.2 dL0, y}, {x, y}}]},
      Text[ToString[Abs@value], {x + 0.2 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 - 0.2 dL0}, {x , y}}]},
      Text[ToString[value], {x + 0.1 dL0, y - 0.2 dL0}]
     }
     ,
     {
      {color, Arrow[{{x, y}, {x, y + 0.2 dL0}}]},
      Text[ToString[value], {x + 0.1 dL0, y + 0.2 dL0}]
     }
    ]
   },
   If[Abs@value > $MachineEpsilon,
    If[start == "startBelow",
     {
      {color, Arrow[{{x, y}, {x, y - 0.2 dL0}}]},
      Text[ToString[Abs@value], {x + 0.1 dL0, y - 0.2 dL0}]
     },
     {
      {color, Arrow[{{x, y + 0.2 dL0}, {x, y}}]},
      Text[ToString[Abs@value], {x + 0.1 dL0, y + 0.2 dL0}]
     }
    ]
   ]
```

```
          ]
         ]

      )

   ]
```