Dynamic Analysis of a Second-Order System with Harmonic Loading

Initialization Code (optional)

Manipulate

```
Manipulate[None,
 Item[Grid[{
    {
     Dynamic[Row[{
        gTick;
        forcingFrequencyInRadian = forcingFrequency * 2 * Pi;

        {naturalFrequency, r, cc, zeta, dampedFrequency, tdd, timeConstant} =
         convertToStandardValues[k0, mass, forcingFrequencyInRadian, forceCritical, zeta];

        {transient, steadyState} =
         oneDegreeOfFreedomSolution[u0, v0, k0, zeta, f0, forcingFrequencyInRadian, naturalFrequency, t];

        magnificationFactor =
         calculateMagnificationFactor[zeta, f0, forcingFrequencyInRadian, naturalFrequency];
        dynamicMagnificationPlot = makeGenericDynamicMagnificationFactorPlot[{0.1, .7, 1, zeta}, r];
        phasePlot = makeGenericPhasePlot[{0.01, 0.1, 0.7, zeta}, r]; (*list of zeta values to use*)

        phaseLagPlot = If[phaseDiagramType == "standard",
          makePhaseDifferencePlot[zeta, f0, forcingFrequencyInRadian, naturalFrequency]
          ,
          makeArgandDiagram[zeta, f0, forcingFrequencyInRadian, naturalFrequency, tscale]
         ];

       }]
     ]
    },
    {
     Grid[{
       {Text[Style["differential equation", Bold, 11]]},
       {
        Grid[{
          {Text[TraditionalForm@Style[HoldForm[
                u''[t] + 2 ξ Subscript[ω, n] u'[t] + ω_n^2 u[t]] == HoldForm[F/m] Sin[HoldForm[ϖ t]], Medium]]
          }
         }, Spacings → {.7, .5},
         Alignment → Center, Frame → True, FrameStyle -> Directive[Thickness[.005], Gray]]
       },
       {Text[Style["system parameters", Bold, 11]]},
       {Grid[{
          {
```

```
      Row[{Style["damping", 11], Spacer[3], Text@Style[ξ, 11]}] ,
      Manipulator[Dynamic[zeta, {zeta = #; forceCritical = False; gTick += del} &],
       {0, 2, .01}, ImageSize → Tiny, ContinuousAction → True
      ],
      Style[Dynamic@padIt2[zeta, {3, 2}], 11],
      Button[Style["1", 9], {zeta = 1; forceCritical = True; gTick += del},
       Appearance → "Palette", Background → LightBlue, ImageSize → {25, 18}], SpanFromLeft
     }
     ,
     {
      Row[{Style["stiffness", 11], Spacer[3], Text@Style["k", Italic]}] ,
      Manipulator[Dynamic[k0, {k0 = #; gTick += del} &],
       {1, 10, 0.1}, ImageSize → Tiny, ContinuousAction → True],
      Style[Dynamic@padIt2[k0, {3, 1}], 11], SpanFromLeft
     }
     ,
     {
      Row[{Style["mass", 11], Spacer[3], Text@TraditionalForm@Style[m]}],
      Manipulator[Dynamic[mass, {mass = #; gTick += del} &], {1, 10, 0.1},
       ImageSize → Tiny, ContinuousAction → True], Style[Dynamic@padIt2[mass, {3, 1}], 11]
     }
     ,
     {
      Row[{Style["load", 11], Spacer[3], F}],
      Manipulator[Dynamic[f0, {f0 = #; gTick += del} &], {0, 10, 0.1},
       ImageSize → Tiny, ContinuousAction → True], Style[Dynamic@padIt2[f0, {3, 1}], 11]
     },
     {
      Row[{Style["load", 11], Spacer[3], ϖ}],
      Manipulator[Dynamic[forcingFrequency,
         {forcingFrequency = #; forcingFrequencyInRadian = forcingFrequency * 2 * Pi; r =
            forcingFrequencyInRadian / Sqrt[k0 / mass]; gTick += del} &], {0, 1, 0.01}, ImageSize → Tiny,
       ContinuousAction → True], Style[Dynamic@padIt2[N@forcingFrequency, {4, 3}], 11],
      Button[Style["ωn", 9], {forcingFrequencyInRadian = Sqrt[k0 / mass];
         forcingFrequency = forcingFrequencyInRadian / (2 * Pi); r = 1; gTick += del},
       Appearance → "Palette", Background → LightBlue, ImageSize → {25, 18}], SpanFromLeft
     }
    }, Alignment → Left, Spacings → {.3, .2},
    Frame → True, FrameStyle -> Directive[Thickness[.005], Gray], Alignment → Center
   ]
  },
  {Text[Style["initial conditions", Bold, 11]]},
  {Grid[{
     {
      Row[{Style["initial", 11], Spacer[3], Text@TraditionalForm@Style[u[t], 11]}],
      Manipulator[Dynamic[u0, {u0 = #; gTick += del} &],
       {-2, 2, 0.1}, ImageSize → Tiny, ContinuousAction → True],
      Style[Dynamic@padIt1[u0, {2, 1}], 11],
      Button[Style["0", 9], {u0 = 0; gTick += del},
       Appearance → "Palette", Background → LightBlue, ImageSize → {25, 18}], SpanFromLeft
     }
     ,
     {
      Row[{Style["initial", 11], Spacer[3], Text@TraditionalForm@Style[u'[t], 11]}],
      Manipulator[Dynamic[v0, {v0 = #; gTick += del} &],
       {-2, 2, 0.1}, ImageSize → Tiny, ContinuousAction → True],
      Style[Dynamic@padIt1[v0, {2, 1}], 11],
      Button[Style["0", 9], {v0 = 0; gTick += del},
       Appearance → "Palette", Background → LightBlue, ImageSize → {25, 18}], SpanFromLeft
     }
    }, Alignment → Left, Spacings → {.3, .2},
    Frame → True, FrameStyle -> Directive[Thickness[.005], Gray], Alignment → Center
   ]
  },
```

```
{Text[Style["model information", Bold, 11]]},
{
 Grid[{
   {Text@Style[Row[{"frequency ratio", Spacer[3], ϖ, "/", ωStyle[n,Italic]}], 11],
    Style[Dynamic@padIt2[N@r, {3, 2}], 11], Spacer[39]},

   {Text@Style[Row[{"natural frequency", Spacer[3], ωStyle[n,Italic]}], 11],
    Style[Dynamic@padIt2[N@naturalFrequency / (2.0 * Pi), {4, 3}], 11], Style["Hz", 11]},

   {Text@Style[Row[{"damped frequency", Spacer[3], ωStyle[d,Italic]}], 11],
    Style[Dynamic@padIt2[N@dampedFrequency / (2.0 * Pi), {4, 3}], 11], Style["Hz", 11]},

   {Text@Style[Row[{"natural period", Spacer[3], 2 π, "/", ωStyle[n,Italic]}], 11],
    Style[Dynamic@padIt2[(2. Pi) / naturalFrequency, {5, 3}], 11], Style["sec", 11]},

   {Text@Style[Row[{"damped period", Spacer[3], 2 π, "/", ωStyle[d,Italic]}], 11],
    Style[Dynamic[If[tdd === Infinity, Infinity, padIt2[N@tdd, {5, 3}]]], 11], Style["sec", 11]},

   {Text@Style[Row[{"damping coefficient", Spacer[3], Style[c, Italic]}], 11],
    Style[Dynamic@padIt2[N@zeta * 2 * Sqrt[mass * k0], {5, 3}], 11], ""},

   {Text@Style[Row[{"magnification factor", Spacer[3], β}], 11] ,
    Style[Dynamic[If[magnificationFactor === Infinity,
       magnificationFactor, padIt2[N@magnificationFactor, {7, 3}]]], 11], ""},

   {Text@Style[Row[{"static displacement", Spacer[3], F, "/", Style[k, Italic]}], 11],
    Style[Dynamic[padIt2[N[f0 / k0], {7, 3}]], 11], ""},

   {Text@Style[Row[{"time constant", Spacer[3], τ}], 11],
    Style[Dynamic[If[timeConstant === Infinity, Infinity, padIt2[N@timeConstant, {7, 3}]]], 11],
    Style["sec", 11]}

  }, Spacings → {.7, .4}, Frame → {All, All},
  FrameStyle -> Directive[Thickness[.5], Gray], Alignment → Left
 ]
},
{Grid[{
   {Text[Style["test cases", Bold, 11]],
    PopupMenu[Dynamic[testCase, {testCase = #;
        Which[testCase == 1,
         k0 = 2.; mass = 4.77; forcingFrequencyInRadian = 0.6; forceCritical = False; zeta = 0.;
         {naturalFrequency, r, cc, zeta, dampedFrequency, tdd, timeConstant} =
          convertToStandardValues[k0, mass, forcingFrequencyInRadian, forceCritical, zeta];
         forcingFrequency = forcingFrequencyInRadian / (2 * Pi); u0 = 0.; v0 = 0.; f0 = 1.;
         tscale = 500; responsePlotType = "full solution"; gTick += del
         ,
         testCase == 2,
         k0 = 1.; mass = k0 / 0.4 ^ 2; forcingFrequencyInRadian = 0.4; forceCritical = False; zeta = 0.;
         {naturalFrequency, r, cc, zeta, dampedFrequency, tdd, timeConstant} =
          convertToStandardValues[k0, mass, forcingFrequencyInRadian, forceCritical, zeta];

         u0 = 0; v0 = 0.0; f0 = 1.; forcingFrequency = forcingFrequencyInRadian / (2 * Pi);
         tscale = 300; responsePlotType = "full solution"; gTick += del
         ,
         testCase == 3,
         k0 = 1; mass = 1; naturalFrequency = Sqrt[k0 / mass];
         zeta = 2 * (0.01) / (2 * Sqrt[mass * k0]); cc = zeta * (2 mass naturalFrequency);
         forcingFrequencyInRadian = Sqrt[k0/mass - cc^2/(2 k0 mass)]; forceCritical = False;
         {naturalFrequency, r, cc, zeta, dampedFrequency, tdd, timeConstant} =
          convertToStandardValues[k0, mass, forcingFrequencyInRadian, forceCritical, zeta];
```



```
u0 = 0.; v0 = 0.; f0 = 1; forcingFrequency = forcingFrequencyInRadian / (2 * Pi);
tscale = 90; responsePlotType = "full solution"; gTick += del
,
testCase == 32,
k0 = 1.; mass = 1; naturalFrequency = Sqrt[k0 / mass];
zeta = 2 * (0.1) / (2 * Sqrt[mass * k0]); cc = zeta * (2 mass naturalFrequency);
forcingFrequencyInRadian = Sqrt[k0/mass - cc^2/(2 k0 mass)]; forceCritical = False;
{naturalFrequency, r, cc, zeta, dampedFrequency, tdd, timeConstant} =
 convertToStandardValues[k0, mass, forcingFrequencyInRadian, forceCritical, zeta];

u0 = 0.; v0 = 0.; f0 = 1; tscale = 90; responsePlotType = "full solution";
forcingFrequency = forcingFrequencyInRadian / (2 * Pi); gTick += del
,
testCase == 4,
k0 = 7.8; mass = 6.07; zeta = 4.18 / (2 * Sqrt[mass * k0]);
forcingFrequencyInRadian = 0 ; forceCritical = False;

{naturalFrequency, r, cc, zeta, dampedFrequency, tdd, timeConstant} =
 convertToStandardValues[k0, mass, forcingFrequencyInRadian, forceCritical, zeta];

u0 = 0.; v0 = 0.; f0 = 1.; forcingFrequency = 0;
tscale = 30; responsePlotType = "full solution"; gTick += del,

testCase == 5,
k0 = 7.8; mass = 2.4; forcingFrequencyInRadian = 0;
forceCritical = False; zeta = 4.18 / (2 * Sqrt[mass * k0]);
{naturalFrequency, r, cc, zeta, dampedFrequency, tdd, timeConstant} =
 convertToStandardValues[k0, mass, forcingFrequencyInRadian, forceCritical, zeta];

u0 = 0.; v0 = 0.; f0 = 1.; forcingFrequency = 0 ;
tscale = 10; responsePlotType = "full solution"; gTick += del,

testCase == 6,
k0 = 7.8; mass = 6.07;
forcingFrequencyInRadian = 0; forceCritical = False; zeta = 10 / (2 * Sqrt[mass * k0]);
{naturalFrequency, r, cc, zeta, dampedFrequency, tdd, timeConstant} =
 convertToStandardValues[k0, mass, forcingFrequencyInRadian, forceCritical, zeta];

u0 = 0.; v0 = 0.; f0 = 1.; forcingFrequency = 0 ;
tscale = 30; responsePlotType = "full solution"; gTick += del,

testCase == 7,
k0 = 1; mass = 9.96; forcingFrequencyInRadian = 0.317 ;
forceCritical = False; zeta = 5.68 / (2 * Sqrt[mass * k0]);
{naturalFrequency, r, cc, zeta, dampedFrequency, tdd, timeConstant} =
 convertToStandardValues[k0, mass, forcingFrequencyInRadian, forceCritical, zeta];

u0 = 1.; v0 = 1.; f0 = 1.; forcingFrequency = forcingFrequencyInRadian / (2 * Pi);
tscale = 200; responsePlotType = "full solution"; gTick += del,

testCase == 8,
k0 = 1; mass = 10;
forcingFrequencyInRadian = 0 ; forceCritical = False; zeta = .7 / (2 * Sqrt[mass * k0]);
{naturalFrequency, r, cc, zeta, dampedFrequency, tdd, timeConstant} =
 convertToStandardValues[k0, mass, forcingFrequencyInRadian, forceCritical, zeta];

u0 = 1; v0 = 0; forcingFrequency = 0; r = 0; tscale = 80; f0 = 0;
phaseDiagramType = "argand"; responsePlotType = "full solution"; gTick += del
];
gTick += del} &],
```

```
              {
               1 → Style["beating phenomenon", 10],
               2 → Style["resonance no damping", 10],
               3 → Style["resonance underdamped (1)", 10],
               32 → Style["resonance underdamped (2)", 10],
               4 → Style["step response underdamped", 10],
               5 → Style["step response critical", 10],
               6 → Style["step response overdamped", 10],
               7 → Style[Row[{"response phase lag by 90", Degree}], 10],
               8 → Style["free underdamped response", 10]
              }, ImageSize -> All]
            }
           }, Alignment → Left,
           Spacings → {.1, .4}
          ]
         }
        }]
       }
      }
     ], ControlPlacement → Left]
   , (* ABOVE IS THE LEFT PANEL*)
   Item[
    Panel[
     Grid[{
       {
        Grid[{
          {Dynamic[
            gTick;
            Show[
             dynamicMagnificationPlot,
             If[f0 == 0 || r == 0, Sequence @@ {},
              Graphics[{Red,
                PointSize[.04],
                Point[{r, If[magnificationFactor > 5, 5, magnificationFactor]}]
               }]
             ]
            ]]}
         }](*1,1*)
        ,
        Grid[{
          {Dynamic[
            gTick;
            Show[
             phasePlot,
             If[f0 == 0 || r == 0, Sequence @@ {},
              Graphics[{Red, PointSize[.04], Point[{r, If[r == 1, -90, 180./Pi ArcTan[1 - r^2, -2 zeta r^2]]}]
               }]
             ]
            ]
           ]
          }
```

```
      }](*1,2*)
    },
    {
    Grid[{
       {
        PopupMenu[Dynamic[responsePlotType, {responsePlotType = #; gTick += del} &],
         {
          "transient" → Style["transient", 10],
          "steady state" → Style["steady state", 10],
          "transient+steady state (separate)" →
           Style["transient+steady state (separate)", 10],
          "full solution" → Style["transient+steady state (combined)", 10],
          "load with response" → Style["load with response", 10]
         }]
       },
       {
        Dynamic[
         gTick;

         makeResponsePlot[responsePlotType, f0, k0, r, v0,
          u0, zeta, naturalFrequency, tscale, transient, steadyState, {206, 206}, t]
        ]
       },
       {
        Grid[{
           {Style["time", 10],
            Manipulator[Dynamic[tscale, {tscale = #; gTick += del} &], {0.1, 500, 0.1}, ImageSize → Tiny],
            Style[Dynamic@padIt2[N[tscale], {4, 1}], 11], Spacer[2], Style["(sec)", 10]
           }
         }, Spacings → {.2, 0}, Alignment → Left]
       }
      }, Spacings → {0, 0}, Alignment → Center, Frame → None
     ](*2,1*)
     ,
     Grid[{
       {Dynamic[gTick; phaseLagPlot]},
       {
        RadioButtonBar[Dynamic[phaseDiagramType, {phaseDiagramType = #} &],
         {"argand" → Style["Argand", 10], "standard" → Style["standard", 10]
         }, Appearance → "Horizontal"]
       }
      }, Spacings → {0, 0}
     ](*2,2*)
    }
   }, Spacings → {0, 0},
   Frame → All, FrameStyle -> Directive[Thickness[.005], LightGray]
  ], Background → White, Alignment → Center, FrameMargins → 0
 ], ControlPlacement → Right],

{{gTick, 0}, None},
{{del, $MachineEpsilon}, None},
{{cc, 0.7}, None},
{{u0, 1.}, None},
{{v0, 1.}, None},
{{f0, 1.}, None},
{{forcingFrequencyInRadian, 0.1}, None},
{{forcingFrequency, .1 / (2 * Pi)}, None},
{{k0, 1.}, None},
{{mass, 10.}, None},
{{tscale, 200.}, None},
{{forceCritical, False}, None},
```

```
{{testCase, 1}, None},
{{specialPlot, 1}, None},
{{responsePlotType, "full solution"}, None},
{{phaseDiagramType, "argand"}, None},

{{zeta, 0.111}, None},
{{naturalFrequency, 0.316}, None},
{{dampedFrequency, 0.314}, None},
{{r, .1}, None}, (*frequency ratio*)
{{tdd, 2 Pi / 0.314}, None},
{{timeConstant, 3.162}, None},

{{magnificationFactor, 0}, None},
{dynamicMagnificationPlot, None},
{phasePlot, None},
{phaseLagPlot, None},
{{setIC, True}, None},
{transient, None},
{steadyState, None},
{sol, None},

SynchronousUpdating → True,
AppearanceElements → "ManipulateMenu",
ControlPlacement → Left,
Alignment → Center,
ImageMargins → 0,
FrameMargins → 1,
ContentSize → {0},
SynchronousInitialization → True,
ContinuousAction → True,
Alignment -> Center,
Paneled → True,
Frame → False,
TrackedSymbols :→ {gTick},
AutorunSequencing → Automatic,

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] &);
  (*--------------------------------------------*)
  padIt1[v_?numeric, f_List] := AccountingForm[Chop[v],
    f, NumberSigns → {"-", "+"}, NumberPadding → {"0", "0"}, SignPadding → True];
  (*--------------------------------------------*)
  padIt2[v_?numeric, f_List] := AccountingForm[Chop[v],
    f, NumberSigns → {"", ""}, NumberPadding → {"0", "0"}, SignPadding → True];
  (*--------------------------------------------*)
  padIt2[v_?numeric, f_Integer] := AccountingForm[Chop[v],
    f, NumberSigns → {"", ""}, NumberPadding → {"0", "0"}, SignPadding → True];
  (*--------------------------------------------*)
  makePhaseDifferencePlot[zeta_?numericPositive,
    f0_?numericPositive, forcingFrequency_?numericPositive,
    naturalFrequency_?numericStrictPositive] := Module[{r, phaseAngle, z, data},
```

```
  If[f0 == 0, r = 0, r = forcingFrequency / naturalFrequency];
  phaseAngle = If[r == 1, Pi / 2, ArcTan[1 - r^2, 2 zeta r]];

  data = Table[{{z, If[f0 == 0, 0, Sin[z - phaseAngle]]}, {z, If[f0 == 0, 0, Sin[z]]}}, {z, 0, 2 Pi, Pi / 10}];

  ListLinePlot[{data[[All, 1]], data[[All, 2]]},
   PlotLabel → Style[Grid[{
       {"displacement phase relative to load"},
       {"red color represents load"},
       {Row[{θ, " = ", padIt1[N[-(180 / Pi) phaseAngle], {4, 1}], Degree}]
       }},
      Spacings → {0, .2}], 11],
   PlotStyle → {{Dashed, Black}, Red}, PlotRange → All, ImagePadding → {{10, 12}, {10, 10}},
   GridLines → Automatic, GridLinesStyle → Directive[Thickness[.001], LightGray],
   AspectRatio → 0.9, Ticks → {{0, Pi / 2, Pi, Pi + 1 / 2 Pi, 2 Pi}, None}
  ]
 ];
(*-------------------------------------------*)
makeArgandDiagram[
  zeta_?numericPositive,
  f0_?numericPositive,
  forcingFrequency_?numericPositive,
  naturalFrequency_?numericStrictPositive,
  maxTime_?numericStrictPositive] := Module[{phaseAngle, r, tipOfForce},

  If[f0 == 0, r = 0, r = forcingFrequency / naturalFrequency];
  phaseAngle = If[r == 1, -(Pi / 2), ArcTan[1 - r^2, -2 zeta r]];
  tipOfForce = {Cos[forcingFrequency maxTime - Pi / 2], Sin[forcingFrequency maxTime - Pi / 2]};
  ListLinePlot[{{{0, 0}}, {{0, 0}}},
   PlotLabel → Style[

     Grid[{
       {"displacement phase relative to load"},
       {"red color represents load"},
       {Row[{θ, " = ", padIt1[N[180 / Pi (phaseAngle)], {4, 1}], Degree}]
       }
      },
      Spacings → {0, .2}
     ], 11],

   AspectRatio → .9,
   ImagePadding → {{10, 12}, {10, 10}},
   GridLines → {Range[-1.2, 1.2, .2], Range[-1.2, 1.2, .2]},
   GridLinesStyle → Directive[LightGray, Thickness[.001]],
   PlotRange → {{-1.2, 1.2}, {-1.2, 1.2}},
   Ticks → None,
   Axes → True,
   Epilog → {
     {Red, Thick, Arrowheads[.1], Arrow[{{0, 0}, tipOfForce}]},
     {Blue, Arrowheads[.1], Arrow[{{0, 0}, {Cos[forcingFrequency maxTime + phaseAngle - Pi / 2],
         Sin[forcingFrequency maxTime + phaseAngle - Pi / 2]}}]},
     {Black, Circle[{0, 0}, 1]},
     If[phaseAngle < -Pi / 16,
      circularArrow[Circle[{0, 0}, 0.3,
        {forcingFrequency maxTime - Pi / 2, forcingFrequency maxTime + phaseAngle - Pi / 2}]],
      Sequence @@ {}]
    }
  ]
 ];
(*-------------------------------------------*)
(*Function below by belisarius from stackoverflow used in the above function to draw an arced arrow*)
circularArrow[s_Circle] := s /. Circle[a_, r_, {start_, end_}] :> (
    {
```



```
        {Blue, s},
        {Blue, Arrow[{# - r / 10^6 {Sin@end, -Cos@end}, #}]}
       } &[a + r {Cos@end, Sin@end}]
    );
(*-------------------------------------------*)
convertToStandardValues[k0_?numericStrictPositive,
  mass_?numericStrictPositive,
  forcingFrequency_?numericPositive,
  isCriticalDamping_?bool,
  zetaOld_?numeric] := Module[{naturalFrequency,
   r, cc, dampedFrequency, tdd, timeConstant, zeta = zetaOld},

  naturalFrequency = Sqrt[k0 / mass];
  r = forcingFrequency / naturalFrequency;
  cc = zeta * (2 mass naturalFrequency);

  If[isCriticalDamping || Abs[zeta - 1] ≤ $MachineEpsilon,
   zeta = 1;
   dampedFrequency = 0;
   tdd = Infinity;
   timeConstant = 1 / naturalFrequency;
   cc = 2 mass naturalFrequency
   ,
   If[zeta - 1 > $MachineEpsilon,
    (*overdamped*)
    dampedFrequency = 0;
    tdd = Infinity;
    timeConstant = 1.0 / (naturalFrequency (zeta - Sqrt[zeta^2 - 1]));
    cc = zeta (2 mass naturalFrequency)
    ,
    (*must be underdamped or zero*)
    dampedFrequency = naturalFrequency Sqrt[1.0 - zeta^2];
    tdd = (2 π) / dampedFrequency;
    timeConstant = 1 / naturalFrequency;
    cc = zeta (2 mass naturalFrequency)]
  ];

  {naturalFrequency, r, cc, zeta, dampedFrequency, tdd, timeConstant}
 ];

(*-------------------------------------------*)
makeGenericPhasePlot[zeta_List, actualR_] := Module[{i, r, data, legend, opt = {Right, Top}},
  r = If[actualR < 2, 2, actualR];

  data = Table[{i,
        If[i == 1 || # == 0, -90, 180. / Pi ArcTan[1 - i^2, -2 # i^2]]}, {i, 0, r, .01}
     ] & /@ zeta;

  legend = Row[{ξ, " = ", Style[Dynamic@padIt2[#, {3, 2}], 11]}] & /@ zeta;

  ListLinePlot[data,
   PlotRange → All,
   PlotLegends → Placed[LineLegend[legend,
      LegendMargins → 1,
      LegendLayout → Function[{x},
        Grid[x, Spacings → {0.1, 0}, Alignment → Left]]], opt
    ],
   GridLines → Automatic,
   GridLinesStyle → Directive[LightGray, Thickness[.001]],
   Frame → True,
   FrameLabel → {{θ, None},
     {Row[{"frequency ratio", Spacer[5], ϖ, " / ", ω}], Style["phase angle vs. frequency ratio", 12]}},
   ImageMargins → 0,
```

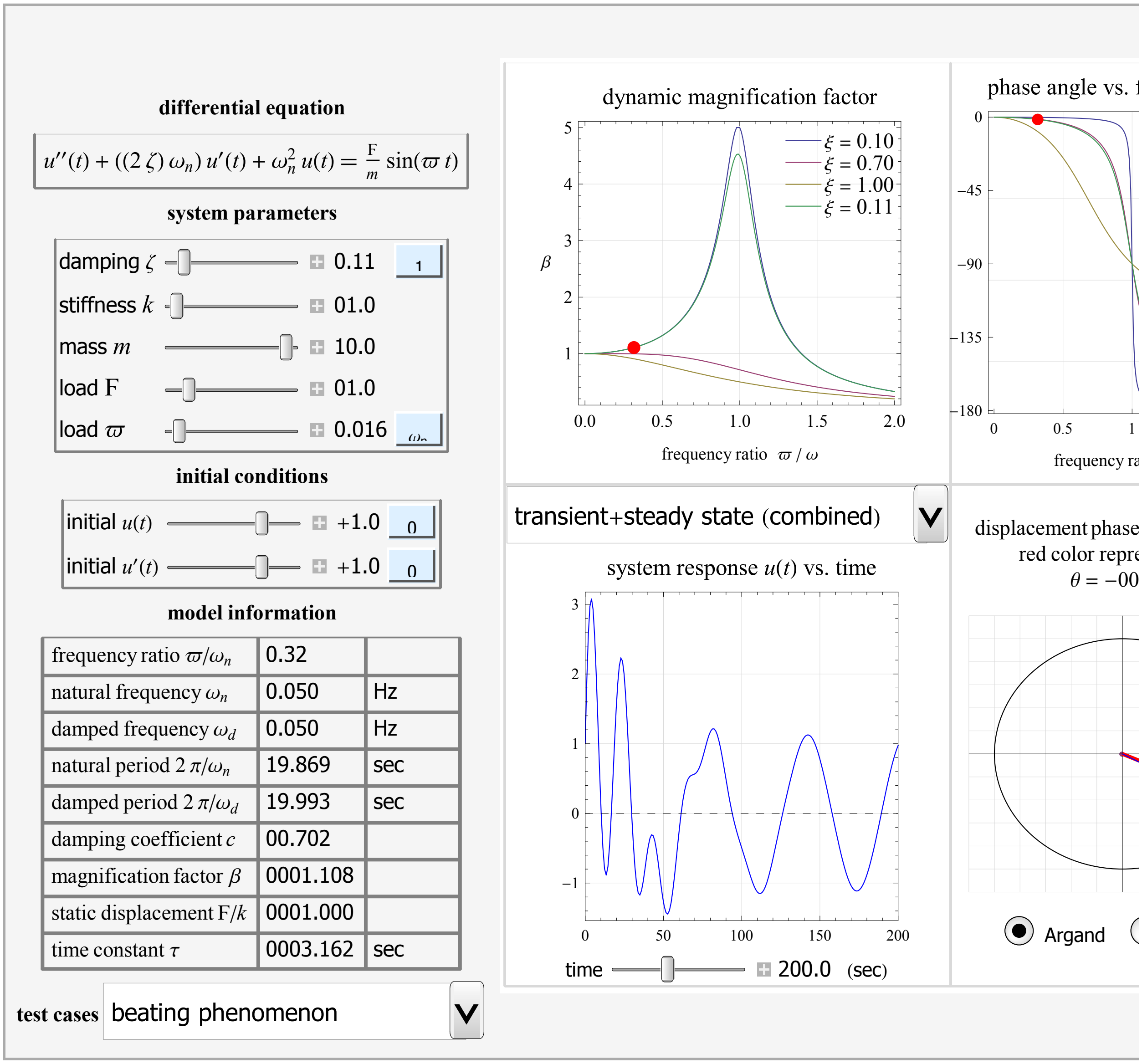
```
    ImagePadding → {{20, 12}, {35, 25}},
    AspectRatio → 1.05,
    RotateLabel → False,
    FrameTicksStyle → 8,
    AxesStyle → {Dashed, Gray},
    FrameTicks → {{{-180, -135, -90, -45, 0}, None},
      {{0, 0.5, 1, 1.5, 2, 2.5, 3}, None}},
    PerformanceGoal → "Speed"
   ]
  ];

(*--------------------------------------------*)
makeGenericDynamicMagnificationFactorPlot[zeta_List, actualR_] := Module[{r, i,
    data, legend, opt = {Right, Top}, z},

   r = If[actualR < 2, 2, actualR];

   data = Table[{i, If[(# == 0 && i == 1), 5, (z = 1 / Sqrt[(1 - i^2)^2 + (2 # i)^2]; If[z > 5, 5, z])]},
        {i, 0, r, .01}] & /@ zeta;

   legend = Row[{ξ, " = ", Style[Dynamic@padIt2[#, {3, 2}], 11]}] & /@ zeta;

   ListLinePlot[data,
    PlotRange → All,
    PlotLegends → Placed[LineLegend[legend,
       LegendMargins → 1,
       LegendLayout → Function[{x}, Grid[x, Spacings → {0.1, 0},
          Alignment → Left]]], opt],
    GridLines → Automatic,
    GridLinesStyle → Directive[LightGray, Thickness[.001]],
    Frame → True,
    FrameLabel → {{β, None},
      {Row[{"frequency ratio", Spacer[5], ϖ, " / ", ω}], Style["dynamic magnification factor", 12]}},
    ImagePadding → {{25, 12}, {35, 25}},
    ImageMargins → 0,
    AspectRatio → 0.88,
    ImageSize → {206, 206},
    RotateLabel → False,
    PerformanceGoal → "Speed"]
  ];
(*--------------------------------------------*)
calculateMagnificationFactor[zeta_?numericPositive,
   f_?numericPositive,
   forcingFrequency_?numericPositive,
   naturalFrequency_?numericStrictPositive] := Module[{r = forcingFrequency / naturalFrequency},
   Which[f == 0, 0,
    zeta == 0, If[r == 1, Infinity, 1 / (1 - r^2)],
    True, 1 / Sqrt[(1 - r^2)^2 + (2 zeta r)^2]
   ]
  ];

(*--------------------------------------------*)
makeResponsePlot[responsePlotType_String, f0_, k_?numericPositive, r_?numericPositive, v0_, u0_, zeta_,
   naturalFrequency_, tscale_, transient_, steadyState_, imageSize_List, t_] := Module[{sol, plotLegend,
    plotStyle, plot, envelop, dampedFrequency, logarithmicDecrement, uAtEndOfCycle, epilog = {}},

   Which[responsePlotType == "transient",
    sol = transient;
    plotStyle = Red,

    responsePlotType == "steady state",
    sol = steadyState;
    plotStyle = Blue,
```

```
  responsePlotType == "transient+steady state (separate)",
  sol = {transient, steadyState};
  plotStyle = {Red, Blue},

  responsePlotType == "full solution",
  sol = transient + steadyState;
  plotStyle = Blue,

  responsePlotType == "load with response",

  sol = {transient + steadyState,
    If[r == 0,
     f0 / k,
     f0 / k Sin[naturalFrequency * r t] (*must be harmonic loading*)
    ]
   };

  plotStyle = {Blue, Red}

 ];
 (*make envelope*)
 If[f0 == 0 && (u0 ≠ 0 || v0 ≠ 0) && responsePlotType ≠ "steady state" && zeta > 0 && zeta < 1,
  (
   plotLegend = None;
   dampedFrequency = naturalFrequency * Sqrt[1 - zeta ^ 2];
   uAtEndOfCycle = (transient + steadyState) /. t → (2 Pi / dampedFrequency);
   logarithmicDecrement = Log[u0 / uAtEndOfCycle];
   epilog =
    Text[Style[Row[{"logrithmic decrement", Spacer[1], padIt2[logarithmicDecrement, {4, 2}]}], 11],
     Scaled[{0.5, 0.9}], {0, 0}];
   envelop = Sqrt[u0 ^ 2 + ((v0 + zeta * naturalFrequency * u0) / dampedFrequency) ^ 2];
   plot = Plot[{envelop Exp[-zeta naturalFrequency t],
       -envelop Exp[-zeta naturalFrequency tt]}, {t, 0, tscale},
     PlotStyle → {{Magenta, Dashed}, {Magenta, Dashed}},
     PlotRange → All,
     PerformanceGoal → "Speed"
    ]
  ),
  (
   plot = {};
   If[f0 > 0,
    (
     If[responsePlotType == "transient+steady state (separate)",
      (
       plotLegend = Placed[LineLegend[
          Style[#, 10] & /@ {"transient", "steady state"},
          LegendMargins → 1,
          LegendLayout → Function[{x}, Grid[x, Spacings → {0.1, 0}, Alignment → Left]]], {Center, Top}]
      ),
      (
       If[responsePlotType == "load with response",
        (
         plotLegend = Placed[LineLegend[Style[#, 10] & /@ {"response", "load"},
            LegendMargins → 1,
            LegendLayout → Function[{x}, Grid[x, Spacings → {0.1, 0}, Alignment → Left]]], {Center, Top}]
        ),
        (
         plotLegend = None
        )
       ]
      )
     ]
    ),
```

```
      (
       plotLegend = None
      )
     ]
    )
   ];

   Show[Plot[Tooltip[Chop@Evaluate@sol, Text[Style[TraditionalForm[Chop@sol], 14]]], {t, 0, tscale},
     PlotRange → {{0, tscale}, All},
     ImagePadding → {{30, 15}, {10, 20}},
     ImageMargins → {{0, 0}, {0, 0}},
     FrameLabel → {
       {None, None},
       {None, Row[{"system response ", Style["u", Italic], "(", Style["t", Italic], ")", " vs. time"}]
       }},
     Frame → True,
     LabelStyle → 12,
     RotateLabel → False,
     GridLines → Automatic,
     GridLinesStyle → Directive[Thickness[.001], LightGray],
     AxesOrigin → {0, 0}, AxesStyle → {Dashed, Gray},
     FrameTicksStyle → 8,
     AspectRatio → 1.05,
     ImageSize → imageSize,
     PlotStyle → plotStyle,
     PlotLegends → plotLegend,
     PerformanceGoal → "Speed",
     Exclusions → None,
     Epilog → epilog,
     Evaluated → True
    ],
    plot
   ]
  ];

 (*--------------------------------------------*)
 oneDegreeOfFreedomSolution[u0_?numeric, v0_?numeric, k_?numericStrictPositive, zeta_?numericPositive,
   f0_?numericPositive, forcingFrquency_?numericPositive, w_?numericStrictPositive, t_] :=
  Module[{wd, z1, z2, a, b, r, phaseAngle, p1, p2, steadyState, transient},

   r = forcingFrquency / w;

   If[f0 == 0,
    steadyState = 0;
    transient = Which[
      zeta == 0, u0 Cos[w t] + v0 / w Cos[w t],

      zeta < 1,
      wd = w Sqrt[1 - zeta^2];
      Exp[-zeta w t ] (u0 Cos[wd t] + (v0 + u0 zeta w) / wd Sin[wd t]),

      zeta == 1,
      (u0 + (v0 + u0 w) t) E^(-w t),

      True,
      p1 = -w zeta + w Sqrt[zeta^2 - 1];
      p2 = -w zeta - w Sqrt[zeta^2 - 1];
      a = (v0 - u0 p2) / (2 w Sqrt[zeta^2 - 1]);
      b = (-v0 + u0 p1) / (2 w Sqrt[zeta^2 - 1]);
      Exp[p1 t] + b Exp[p2 t]
     ],
    (*F is not zero now*)
    {steadyState, transient} = Which[
      zeta == 0, (*undamped*)
```

```
If[forcingFrquency == 0, (*constant force*)
 {f0 / k, (u0 - f0 / k) Cos[w t] + v0 / w Sin[w t]}
 ,
 If[Abs[r - 1] ≤ 0.01, (*resonance*)
  {-(f0 / k) (forcingFrquency t) / 2 Cos[forcingFrquency t],
   u0 Cos[forcingFrquency t] + v0 / forcingFrquency Sin[forcingFrquency t]
  },
  {(f0 / k) 1 / (1 - r^2) Sin[forcingFrquency t],
   u0 Cos[w t] + (v0 / w - ((f0 / k) r / (1 - r^2))) Sin[w t]}]
],

zeta < 1,
phaseAngle = If[r == 1, Pi / 2, ArcTan[1 - r^2, 2 zeta r]];
z1 = Sqrt[(1 - r^2)^2 + (2 zeta r)^2];
wd = w Sqrt[1 - zeta^2];
a = If[forcingFrquency == 0,
  u0 - (f0 / k),
  u0 + (f0 / k) / z1 Sin[phaseAngle]
 ];
b = If[forcingFrquency == 0, (v0 + a zeta w) / wd,
  v0 / wd + (u0 zeta w) / wd + (f0 / k) / (wd z1) (zeta w Sin[phaseAngle] - forcingFrquency Cos[phaseAngle])
 ];

If[forcingFrquency == 0,
 {(f0 / k), Exp[-zeta w t] * (a Cos[wd t] + b Sin[wd t])}
 ,
 {(f0 / k) / z1 Sin[forcingFrquency t - phaseAngle], Exp[-zeta w t] (a Cos[wd t] + b Sin[wd t])}],

(*critical*)
zeta == 1,
phaseAngle = If[r == 1, Pi / 2, ArcTan[1 - r^2, 2 r]];
z1 = Sqrt[(1 - r^2)^2 + (2 r)^2];
a = Which[forcingFrquency == 0,
  u0 - (f0 / k),
  True,
  u0 + (f0 / k) / z1 Sin[phaseAngle]
 ];
b = Which[forcingFrquency == 0,
  v0 + u0 w - (f0 / k) w,
  True,
  v0 + u0 w + (f0 / k) / z1 (w Sin[phaseAngle] - forcingFrquency Cos[phaseAngle])
 ];

If[forcingFrquency == 0,
 {(f0 / k), (a + b t) Exp[-w t]}
 ,
 {(f0 / k) / z1 Sin[forcingFrquency t - phaseAngle], (a + b t) Exp[-w t]}
],

(*overdamped*)True,
phaseAngle = If[r == 1, Pi / 2, ArcTan[1 - r^2, 2 zeta r]];
z1 = (f0 / k) / Sqrt[(1 - r^2)^2 + (2 zeta r)^2];
z2 = Sqrt[zeta^2 - 1];
a = (v0 + u0 w zeta + u0 w z2 +
    z2 (w (zeta + z2) Sin[phaseAngle] - forcingFrquency Cos[phaseAngle])) / (2 w z2);
b = -((v0 + u0 w zeta - u0 w z2 + z2 (w (zeta - z2) Sin[phaseAngle] - forcingFrquency Cos[phaseAngle])) /
    (2 w z2));

If[forcingFrquency == 0,
 p1 = -w zeta + w Sqrt[zeta^2 - 1];
 p2 = -w zeta - w Sqrt[zeta^2 - 1];
 b = ((f0 / k) p1 - u0 p1 + v0) / (p2 - p1); a = u0 - (f0 / k) - b;
 {
  (f0 / k),
```

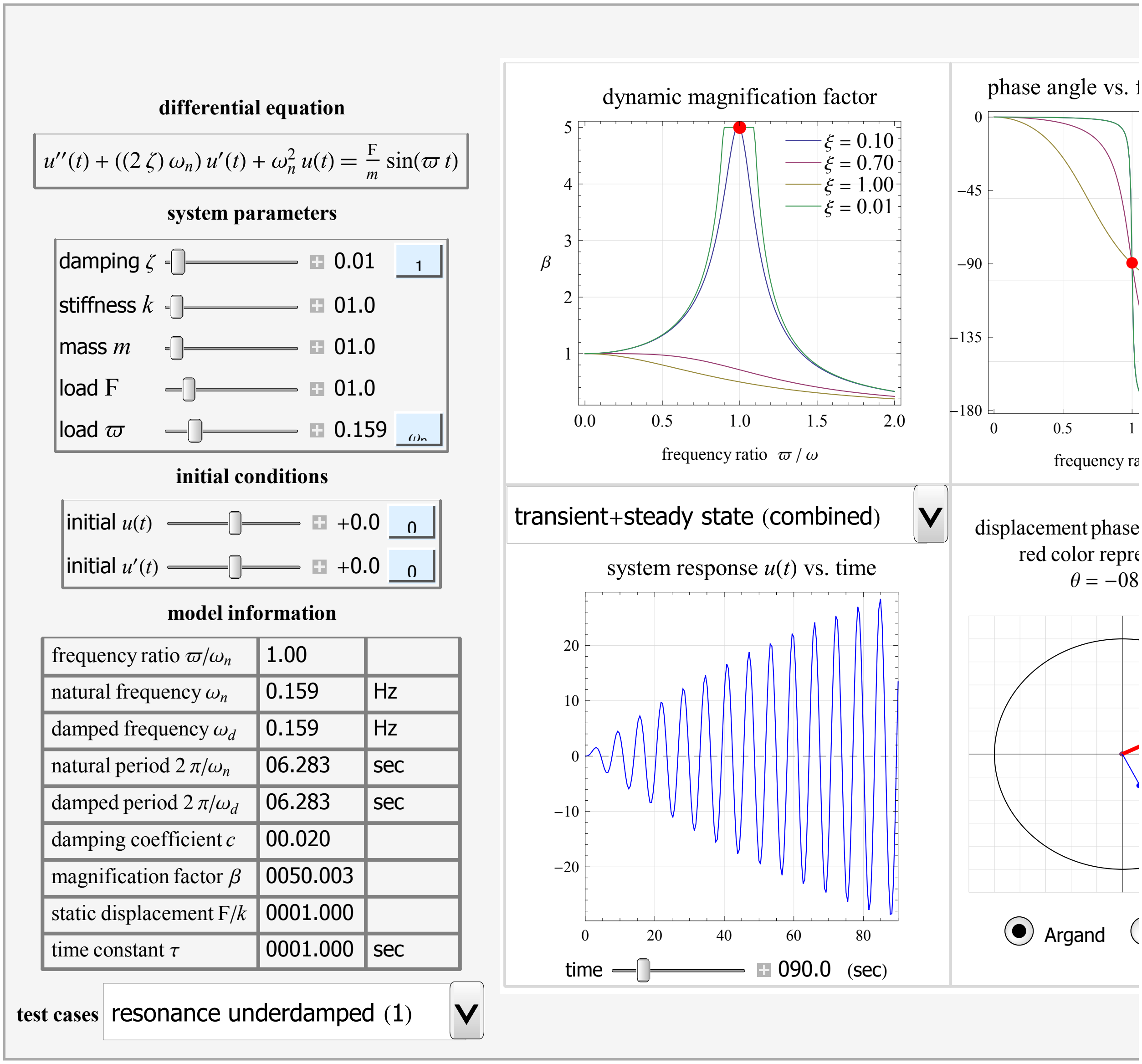
```
                a Exp[p1 t] + b Exp[p2 t]
               }
               ,
               {z1 Sin[forcingFrquency t - phaseAngle],
                a Exp[(-zeta + z2) w t] + b Exp[(-zeta - z2) w t]
               }]
             ]
           ];
           {transient, steadyState}
         ];
       }
    ]
```

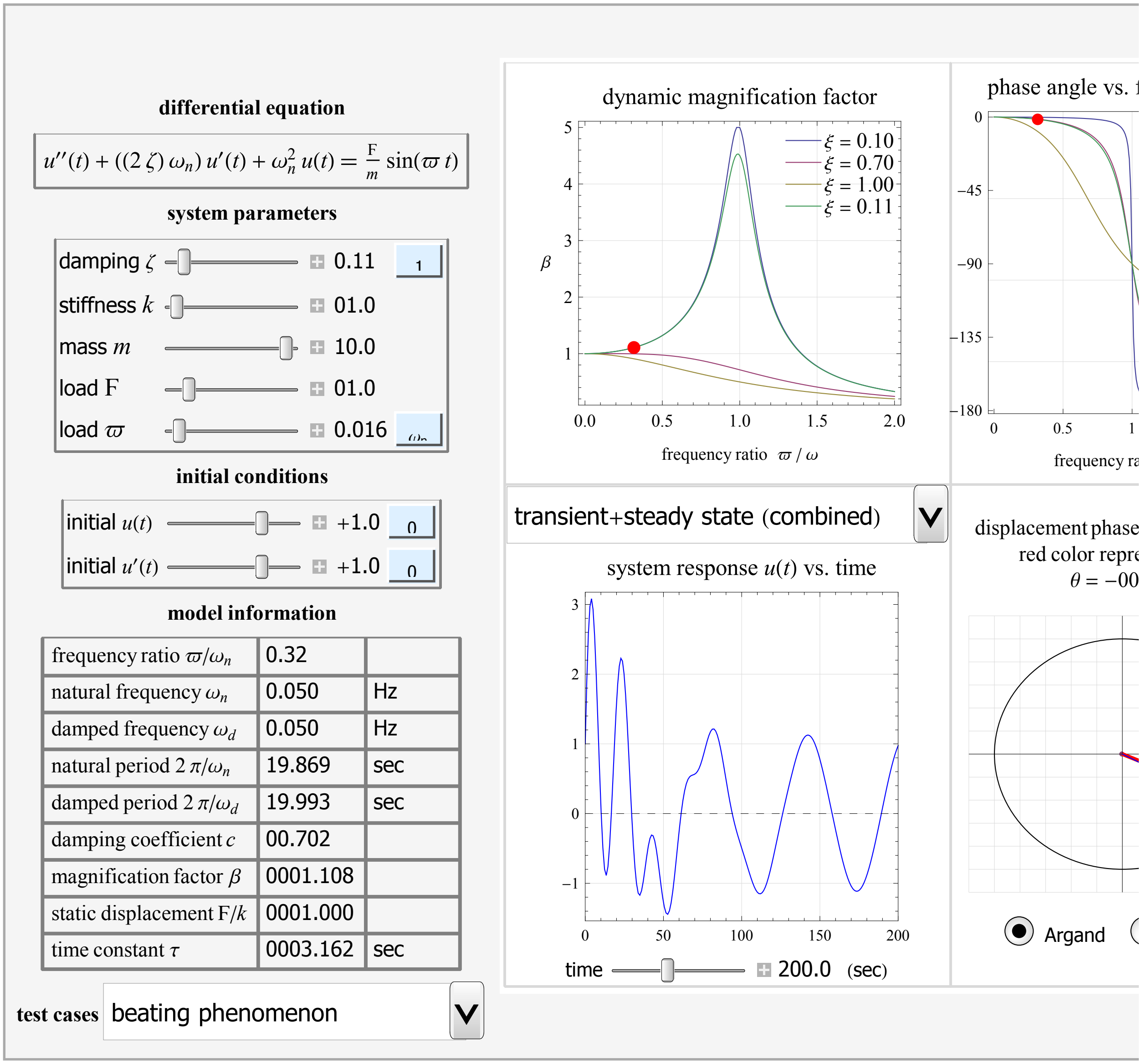


Caption

This Demonstration gives a complete analysis of a second-order system with harmonic loading. The system's differential equation is $m u'' + c u' + k u = f(t)$, where $f(t) = F \sin(\varpi t)$, m is the mass of the system, c is the damping coefficient, k is the stiffness, F is the magnitude of the force, and ϖ is the force frequency. The response $u(t)$ is plotted as a function of time for the underdamped, critically damped, and overdamped cases. This Demonstration displays the transient response (the homogeneous part of the total solution), the steady state response (the particular part of the total solution), and the total response, which is the combination of the two. You can see the analytical solution for each case by moving the mouse over the response curve. Separate displays are given for the dynamic magnification factor and the phase of the response relative to the force. A number of pre-configured test cases can be chosen, to illustrate several important cases of system responses under different loading conditions.

Thumbnail





Details

(optional)

The equation of motion of a second-order linear system of mass m with harmonic applied loading is given by the differential equation $m\,u'' + c\,u' + k\,u = F\sin(\varpi\,t)$. There are 12 different analytical solutions depending on whether damping or loading is present and, if so, whether the system is underdamped, critically damped or overdamped.

The solution for each of the 12 cases was derived analytically and shown in the Demonstration, subject to the user's choice. Following are definitions of the relevant parameters. All units are in SI.

The damping ratio is $\xi = \frac{c}{c_r}$, where c is the damping coefficient, such that $c_r = 2\,\omega\,m$ represents critical damping. The natural underdamped frequency is given by $\omega = \sqrt{\frac{k}{m}}$, where k is the stiffness and m is the mass. The damped frequency of the system, defined for $\xi < 1$, is given by $\omega_d = \omega\sqrt{1-\xi^2}$. The frequency ratio is $r = \frac{\varpi}{\omega}$, where ϖ is the forcing frequency. The dynamic magnification factor β is the ratio of the steady state response to the static response. The static response is given by $\frac{F}{k}$, where F is the force magnitude. The time constant is $\tau = \frac{1}{\xi\,\omega}$ and the damped period of oscillation is $T_d = \frac{2\pi}{\omega_d}$.

When the system is undamped and the load is harmonic, resonance occurs when $r = 1$ or $\varpi = \omega$. When the system is underdamped and the load is again harmonic, practical resonance occurs when $\varpi = \omega\sqrt{1-2\,\xi^2}$ and the corresponding maximum magnification factor is

$\beta = \frac{1}{2\xi\sqrt{1-\xi^2}}$. You can force the loading frequency to be equal to the natural frequency by clicking the button located to the right of the slider used to input the loading frequency. The forcing frequency ϖ is expressed in Hz, but converted to radians per second internally.
This Demonstration also shows plots of the phase of the response $u(t)$ relative to loading $\sin(\varpi t)$. The phase of the response lags behind loading by an angle $\theta = -\tan^{-1}\frac{2\xi r}{1-r^2}$, which is plotted in the complex plane on an Argand diagram. The phase angle ranges from $0°$ to $-180°$.
When the loading frequency ϖ is set to zero, only F is used as the force $f(t)$. This allows a constant force loading, F. For example, by setting $\varpi = 0$ and $F = 1$, a step response is obtained. To make the loading zero, the slider F is set equal to zero.
The Demonstration contains a number of pre-configured test cases to illustrate different loading conditions, such as beating phenomenon, resonance, practical response, impulse response, and step responses for different damping values.
References

[1] M. Paz and W. Leigh, *Structural Dynamics: Theory and Computation*, 5th ed., Boston: Kluwer Academic Publishers, 2004.

[2] W. T. Thomson, *Theory of Vibration with Applications*, Englewood Cliffs, NJ: Prentice-Hall, 1972.

[3] R. W. Clough and J. Penzien, *Dynamics of Structures*, New York: McGraw-Hill, 1975.

[4] R. K. Vierck, *Vibration Analysis*, Scranton, PA: International Textbook Company, 1967.

[5] A. A. Shabana, *Theory of Vibration*, Vol. 1, New York: Springer-Verlag, 1991.

[6] B. Morrill, *Mechanical Vibrations*, New York: Ronald Press, 1957.

Control Suggestions (optional)

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

Search Terms (optional)

beats
vibration
resonance
steady state
transient state
structural dynamics
damped motion
Tooltip
second-order system

Related Links (optional)

Damped Simple Harmonic Motion
Critically Damped Simple Harmonic Motion
Underdamped Simple Harmonic Motion

Authoring Information

Contributed by: Nasser M. Abbasi

