Free Response in a Second-Order System

Initialization Code (optional)

Manipulate

```
Manipulate[
 (
  If [forceCritical == True, b = Round[N[2 m Sqrt[k / m]], 0.1]];
  makePlot[m, b, k, y0, yder0, maxt, currentTime, autoYscale, addGridLines, forceCritical]
 ),

 {{m, 45, "M (kg)"}, 1, 70, 1, Appearance → "Labeled", ImageSize → Tiny},
 {{b, 14, "damping N/m/s"}, 6, 100, 0.1, Appearance → "Labeled", ImageSize → Tiny},
 {{forceCritical, False, "critical ξ"}, {True, False}, ImageSize → Tiny},
 {{k, 24, "stiffness N/m"}, 1, 100, 1, Appearance → "Labeled", ImageSize → Tiny},
 {{maxt, 50, "simulation time (sec)"}, 0, 100, 0.1, Appearance → "Labeled", ImageSize → Tiny},

 Delimiter,

 {{y0, .3, "y(0) m"}, -0.5, 0.5, 0.1, Appearance → "Labeled", ImageSize → Tiny},
 {{yder0, 0, "y'(0) m/s"}, -0.5, 1, 0.1, Appearance → "Labeled", ImageSize → Tiny},

 {{currentTime, 100, "start simulation"}, 0, Dynamic[maxt], 1, ControlType → Trigger,
  AnimationRepetitions → Infinity, DisplayAllSteps → True, AnimationRate → 4, ImageSize → Tiny},

 Delimiter,

 Grid[{
   {
    Grid[{
      {"auto y-axis scale", Spacer[1], Control[{{autoYscale, True, ""}, {True, False}, ImageSize → Tiny}]},
      {"gridlines", Spacer[1], Control[{{addGridLines, True, ""}, {True, False}, ImageSize → Tiny}]}
     }, Spacings → 0, Alignment → Left
    ],
    "",
    Grid[{
      {"test cases"},
      {PopupMenu[Dynamic[testCase, {testCase = #;
            Which[testCase == 1,
             m = 45; b = 14; forceCritical = False;
             k = 24; maxt = 30; upperLimit = 1.5; y0 = 0.5; yder0 = 0.3; currentTime = 0,
             testCase == 2,
             m = 70; b = 6.7; forceCritical = False;
             k = 100; maxt = 100; upperLimit = 1.5; y0 = 0.5; yder0 = 0.3; currentTime = 0,
             testCase == 3,
             m = 70; b = 6.7; forceCritical = False;
             k = 100; maxt = 100; upperLimit = 1.5; y0 = 0; yder0 = 1; currentTime = 0
            ];
           } &],
        {
         1 → Style["case 1", 10],
         2 → Style["case 2", 10],
```

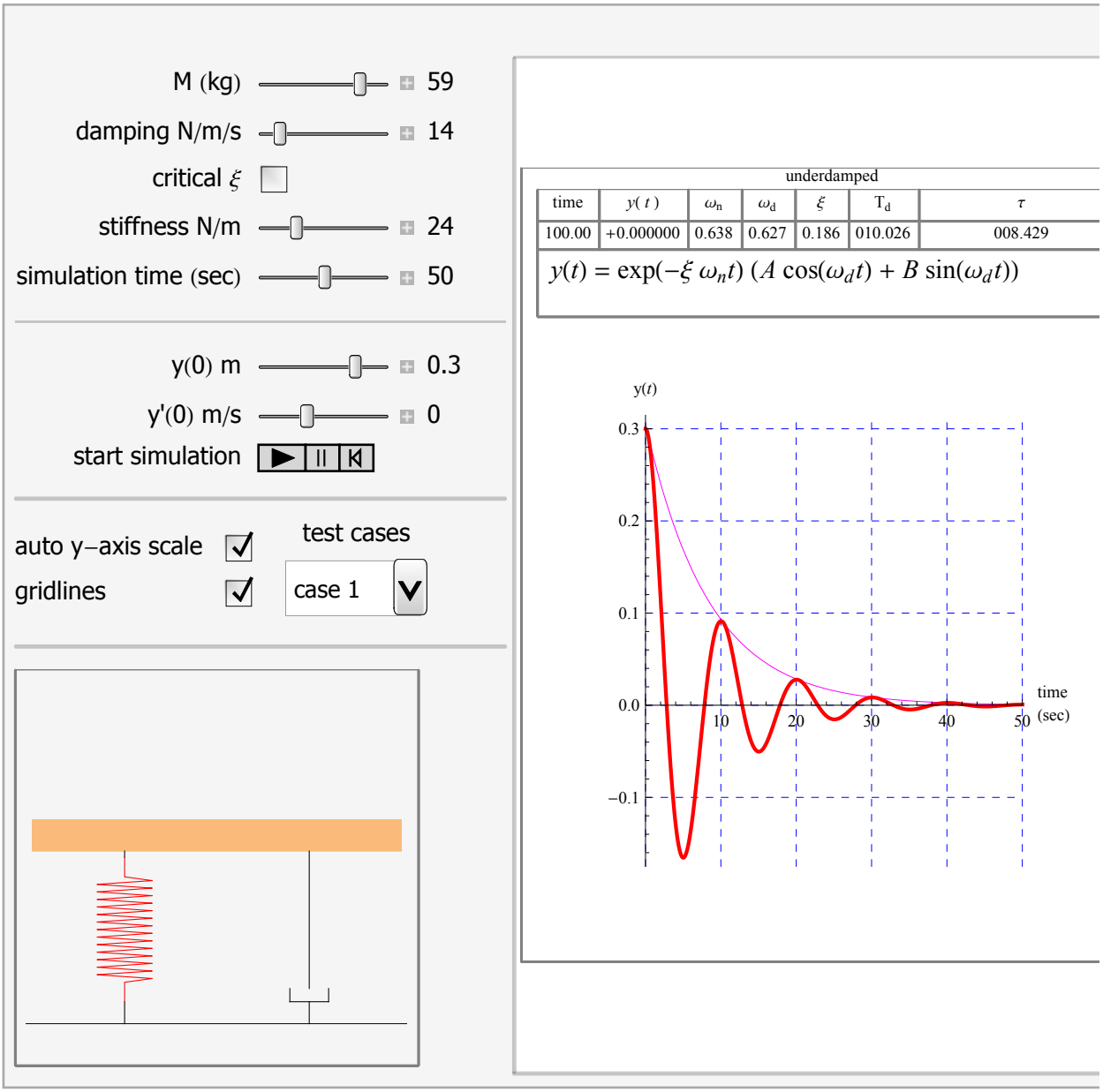
```
        3 → Style["case 3", 10]
       }, ImageSize -> All]
     }
    }, Alignment → Center, Spacings → {.1, .4}
   ]
  }
 }, Alignment → Center, Spacings → {.5, 0}],

Delimiter,

Grid[
 {{Dynamic[makeDynamic[m, b, k, y0, yder0, 100, currentTime, forceCritical, upperLimit]]}},
 Alignment → Center, Frame → True, FrameStyle -> Directive[Thickness[.001], Gray]],

{{testCase, 1}, None},
{{upperLimit, 3}, None},
Alignment → Center,
ImageMargins → 2, (*important*)
FrameMargins → 1,
Alignment -> Center,
Paneled → True,
Frame → False,
AutorunSequencing → Automatic,

ControlPlacement → Left,
SynchronousUpdating → True,
ContinuousAction → True,
SynchronousInitialization → True,
TrackedSymbols :→
 {y0, currentTime, m, b, k, maxt, yder0, autoYscale, forceCritical, impulseResponse, addGridLines},
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] &);
  (*-------------------------------------------*)
  (* 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[Chop[v] , f, NumberSigns → {"", ""}, NumberPadding → {"0", "0"}, SignPadding → True];
  padIt2[v_?numeric, f_Integer] := AccountingForm[Chop[v] , f, NumberSigns → {"", ""},
    NumberPadding → {"0", "0"}, SignPadding → True];
  (*-------------------------------------------*)
  (*--------------------*)
  (*                    *)
  (*--------------------*)
  makePlot[m_, b_, k_, y0_, yder0_, maxt_, currentTime_, autoYscale_, addGridLines_, isCritical_] :=
   Module[{ξ, ωn = Sqrt[k / m], ωd, t, type, y, sng, timeToPlot, Td, τ, t1, t7, t8, asString, plotDouble},
```

```
ξ = b / (2. m ωn);
{y, ωd, Td, τ, asString} = getSolution[t, ξ, y0, yder0, ωn, isCritical];
sng = If[Chop[N@(y /. t → currentTime)] ≤ 0, "-", "+"];

t1 = If[isCritical, "critically damped",
  If[ξ < 1, "underdamped", "overdamped"]
 ];

timeToPlot = currentTime;
If[currentTime ≤ 10^-6, timeToPlot = 0.01];

If[ξ < 1,
 {
  t7 =
   Row[{PaddedForm[Td, {6, 3}, NumberSigns → {"", ""}, NumberPadding → {"0", "0"}]}];
  t8 =
   Row[{PaddedForm[(Chop[N@τ, 10^-5]), {6, 3}, NumberSigns → {"", ""}, NumberPadding → {"0", "0"}]}]
 },
 {
  t7 = Row[{PaddedForm[0, {6, 3}, NumberSigns → {"", ""}, NumberPadding → {"0", "0"}]}];
  t8 = Row[{PaddedForm[0, {6, 3}, NumberSigns → {"", ""}, NumberPadding → {"0", "0"}]}]
 }
];

plotDouble = If[ Not[isCritical] && ξ < 1 && Abs[yder0] < $MachineEpsilon, True, False];

Text@Grid[{
   {t1},
   {
    Grid[{
       {
        "time",
        Row[{Style["y", Italic], "( ", Style["t", Italic], " )"}],
        "ωn",
        "ωd",
        "ξ",
        "Td",
        "τ"
       },
       {padIt2[currentTime, {5, 2}],
        Row[{sng, PaddedForm[(Chop[N@y /. t → currentTime, 10^-5]), {6,
             6}, NumberSigns → {"", ""}, NumberPadding → {"0", "0"}]}],
        NumberForm[N@ωn, {6, 3}],
        If[ ξ < 1,  NumberForm[N@ωd, {6, 3}], NumberForm[0, {6, 3}] ],
        Row[{NumberForm[N@ξ, {6, 3}]}],
        t7,
        t8
       },
       {Pane[
         Style[Text[asString], If[Or[ξ < 1, isCritical] , 14, 10]], ImageSize → {300, 30}], SpanFromLeft}
     }, Spacings → {.8, .7}, Alignment → Center, Frame → All, FrameStyle ->
      Directive[Thickness[.001], Gray]]
   },
   {

    Plot[Evaluate@If[ plotDouble, {y0 Exp[-ξ ωn t] (Sqrt[1/(1 - ξ^2)]), y}, y], {t, 0, timeToPlot},

     PlotRange → {{0, maxt}, If[autoYscale, All, {-8, 8}]},
     Frame → False,
     ImageSize → {290, 300},
     ImagePadding → {{42, 40}, {30, 30}},
     ImageMargins → 20,
```

```
        GridLines → If[addGridLines, Automatic, None],
        GridLinesStyle → Directive[Blue, Thickness[.0005], Dashed],
        AspectRatio → 1.2,
        PlotStyle →
         If[ plotDouble, {Directive[Thin, Magenta], Directive[Red, Thick]}, Directive[Red, Thick]],
        AxesLabel → {Text@Column[{"time", "(sec)"}], Text@Row[{"y(", Style["t", Italic], ")"}]}
       ]
      }
     }, Spacings → {0, .1}, Alignment → Center,
     Frame → True, FrameStyle -> Directive[Thickness[.001], Gray]]

 ];
(*--------------------*)
(*                    *)
(*--------------------*)
makeDynamic[m_, b_, k_, y0_, yder0_, maxt_, currentTime_, isCritical_, upperLimit_] :=
 Module[{ξ, ωn = Sqrt[k / m], ωd, t, y, asString, Td, τ},

  ξ = b / (2 m ωn);
  {y, ωd, Td, τ, asString} = getSolution[t, ξ, y0, yder0, ωn, isCritical];

  makeSystem[y /. t → currentTime, upperLimit]
 ];
(*--------------------*)
(*                    *)
(*--------------------*)
makeSystem[yy_, upperLimit_] := Module[{a1 = 4, a2 = 1, a3 = 7, a4, a5, a6, a7, y, annot, sng},

  a4 = 0.1 a3;
  a7 = 0.2 a2;
  a5 = 0.6 a4;
  a6 = a5;
  y = 3 * (yy - a2 / 2);
  sng = If[y ≤ 0, "- ", "+ "];

  annot = Row[
    {sng, PaddedForm[(Chop[N@yy, 10^-5]), {6, 6}, NumberSigns → {"", ""}, NumberPadding → {"0", "0"}]}];
  Graphics[
   {
    {Opacity[.5], Orange, Rectangle[{-a1, y}, {a1, y + a2}] },
    {Black, Line[{{-a1 / 2, y - a7}, {-a1 / 2, y}}] },
    {Black, Line[{{a1 / 2, y}, {a1 / 2, -a3 + a4 + a5}}] },
    {Black, Line[{{a1 / 2, -a3}, {a1 / 2, -a3 + a4}}] },
    {Black, Line[{{a1 / 2 - a6, -a3 + a4}, {a1 / 2 + a6, -a3 + a4}}] },
    {Black, Line[{{a1 / 2 - a6, -a3 + a4}, {a1 / 2 - a6, -a3 + a4 + a5}}] },
    {Black, Line[{{a1 / 2 + a6, -a3 + a4}, {a1 / 2 + a6, -a3 + a4 + a5}}] },
    {Black, Line[{{-a1 / 2, -a3}, {-a1 / 2, -a3 + a4}}] },
    {Black, Thin, Line[{ {-3 a1, -a3}, {3 a1, -a3}}] },
    (*{Black, Dashed,Line[{{-a1,0},{ a1 ,0}}]},*)
    {Black, Style[Text[annot, {-2 a1 , yy}, {0, 0}] , 11] },
    {Red, Line[Rugo[-a1 / 2, y - a7, -a1 / 2, -a3 + a4]]}
   },
   AspectRatio → .85,
   PlotRange → {{-a1, a1}, {-7, upperLimit}},
   Axes → False,
   AxesOrigin → {0, 0},
   ImageMargins → 2,
   ImagePadding → 2, ImageSize → {200, 200}
  ]
 ];
(*--------------------*)
```

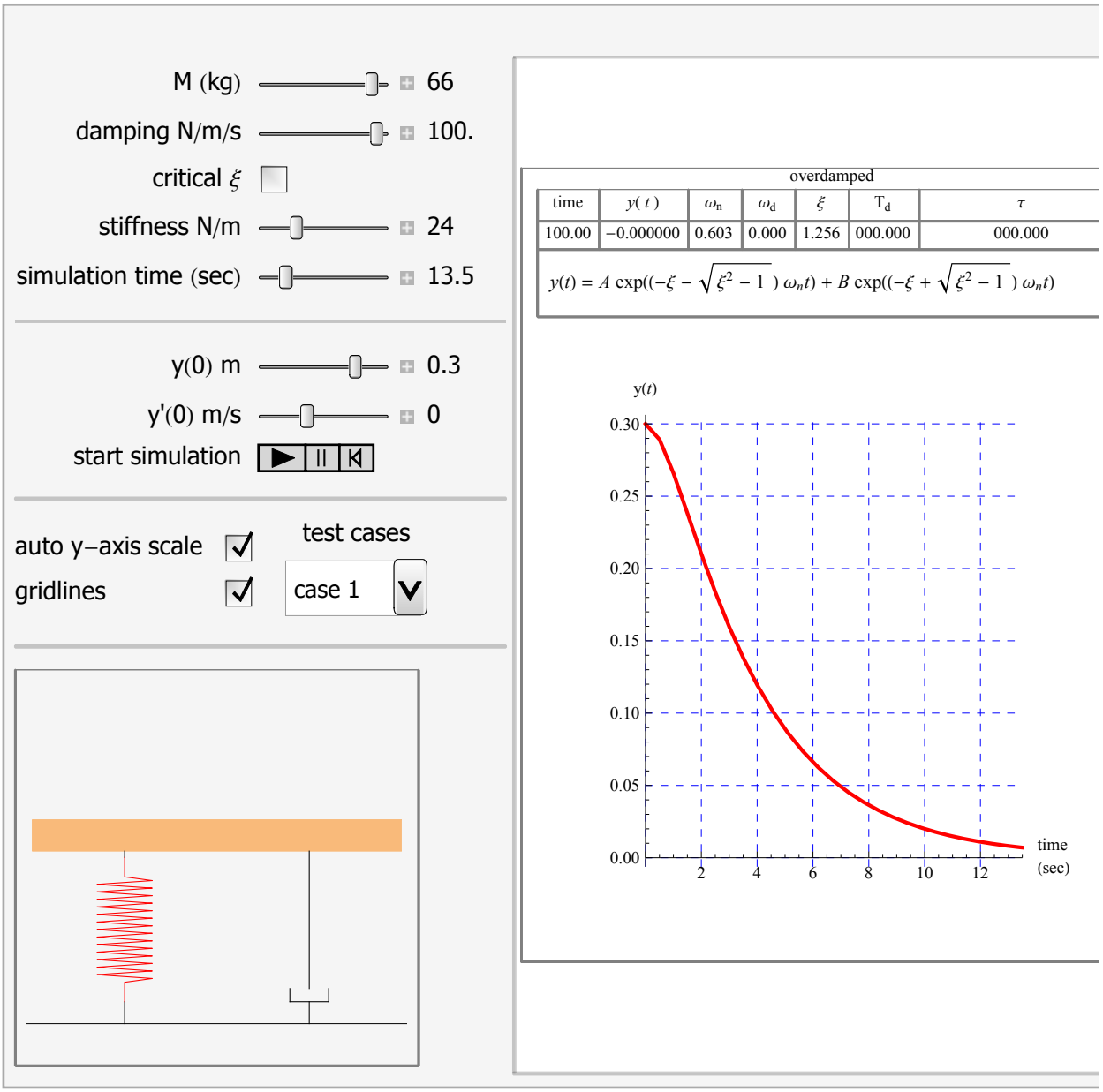
```
(*                    *)
(*--------------------*)
getSolution[t_, ξ_, y0_, yder0_, ωn_, isCritical_] := Module[{ωd = -1, A, B, y, Td = -1, τ = -1, asString},
  If[isCritical,
   {
    A = y0;
    B = yder0 + A ξ ωn;
    y = (A + B t) Exp[-ξ ωn t];

    asString =
     Text@Row[{Style["y", Italic], "(", Style["t", Italic], ") = (", Style["A", Italic], " + ", Style[
         "B", Italic], " ", Style["t", Italic], ") exp(-ξ ", ω_Style["n",Italic], Style["t", Italic], ")"}]
   },
   {
    If[ξ > 1,
     {
      A = -(yder0 + y0 ξ ωn - y0 Sqrt[-1 + ξ^2] ωn) / (2 Sqrt[-1 + ξ^2] ωn);
      B = y0/2 + (y0 ξ)/(2 Sqrt[-1 + ξ^2]) + yder0/(2 Sqrt[-1 + ξ^2] ωn);
      y = A Exp[(-ξ - Sqrt[ξ^2 - 1]) ωn t] + B Exp[(-ξ + Sqrt[ξ^2 - 1]) ωn t];

      asString = Text@Row[{Style["y", Italic], "(", Style["t", Italic], ") = ",
          Style["A", Italic], " exp((-ξ - Sqrt[ξ^2 - 1]) ", ω_Style["n",Italic], Style["t", Italic], ") + ",
          Style["B", Italic], " exp((-ξ + Sqrt[ξ^2 - 1]) ", ω_Style["n",Italic], Style["t", Italic], ")"}]
     },
     {
      A = y0;
      ωd = ωn Sqrt[1. - ξ^2];
      Td = 2 Pi/ωd;

      τ = 1/(ξ ωn);
      B = (yder0 + A ξ ωn)/ωd;
      y = Exp[-ξ ωn t] (A Cos[ωd t] + B Sin[ωd t]);

      asString = Text@Row[{Style["y", Italic], "(", Style["t", Italic], ") = exp(-ξ ", ω_Style["n",Italic],
          Style["t", Italic], ") (", Style["A", Italic], " cos(", ω_Style["d",Italic], Style["t", Italic],
          ") + ", Style["B", Italic], " sin(", ω_Style["d",Italic], Style["t", Italic], "))"}]
     }
    ]
   }
  ];

  {y, ωd, Td, τ, asString}
 ];
(*--------------------*)
```

```
(*                    *)
(*--------------------*)
(*Thanks to Árpád Kósa for this function below which draws the spring found at WRI demo site*)
Rugo[xkezd_, ykezd_, xveg_, yveg_] := {
   hx = xveg - xkezd; hy = yveg - ykezd; szel = 0.6; veghossz = 0.5;
   hossz = Sqrt[hx^2 + hy^2] ;
   dh = (hossz - 2 * veghossz) / 30;
   {xkezd, ykezd}} ~Join~ {{xkezd + hx * (dh + veghossz) / hossz, ykezd + hy * (dh + veghossz) / hossz}} ~
  Join~ Table[If[OddQ[i], {xkezd + hx * (i * dh + veghossz) / hossz + hy * szel / hossz,
     ykezd + hy * (i * dh + veghossz) / hossz - hx * szel / hossz},
    {xkezd + hx * (i * dh + veghossz) / hossz - hy * szel / hossz,
     ykezd + hy * (i * dh + veghossz) / hossz + hx * szel / hossz}], {i, 2, 28}] ~Join~
  {{xkezd + hx * (29 * dh + veghossz) / hossz, ykezd + hy * (29 * dh + veghossz) / hossz}} ~Join~ {{xveg, yveg}}
)
]
```



Caption

This Demonstration can be used to study the free response of a second-order linear system. You can vary the system parameters (mass, damping, and stiffness) and simulate the response. The analytical solution is displayed at the top of the plot for the cases of underdamping, critically damping, and overdamp. In addition, a standard physical model of mass-spring-damper is run at the same time as the response plot is updated. You can set the initial conditions for initial position and speed using the sliders. Setting the initial conditions to $y(0) = 0$ and $y'(0) = 1$ makes the response the same as the impulse response.



Details (optional)

The underdamped response of a second-order system is given by $y(t) = \exp(-\xi\,\omega_n\,t)\,(A\cos(\omega_d\,t) + B\sin(\omega_d\,t))$. The critically damped system has the response $y(t) = (A + B\,t)\exp(-\xi\,\omega_n\,t)$ and the overdamped system has the response $y(t) = A\exp\left(\left(-\xi + \sqrt{\xi^2 - 1}\right)\omega_n\,t\right) + B\exp\left(\left(-\xi + \sqrt{\xi^2 - 1}\right)\omega_n\,t\right)$. In all of the above, A and B can be found from the initial conditions, ω_n is the natural frequency in $\mathrm{rad/sec}$, ω_d is the damped natural frequency in $\mathrm{rad/sec}$, and ξ is the damping coefficient.

For the underdamped case, the damped period of oscillation is given by $T_d = \frac{2\pi}{\omega_d}$ and the time constant is given by $\tau = \frac{1}{\xi\,\omega_n}$; both are in seconds. All units displayed are in SI.

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)

linear system
second order system
stiffness
damping
mass-spring-damper

Related Links (optional)

Authoring Information

Contributed by: Nasser M. Abbasi