From the Continuous Time Fourier Transform to the Discrete Time Fourier Transform by Sampling

Initialization Code (optional)

Manipulate

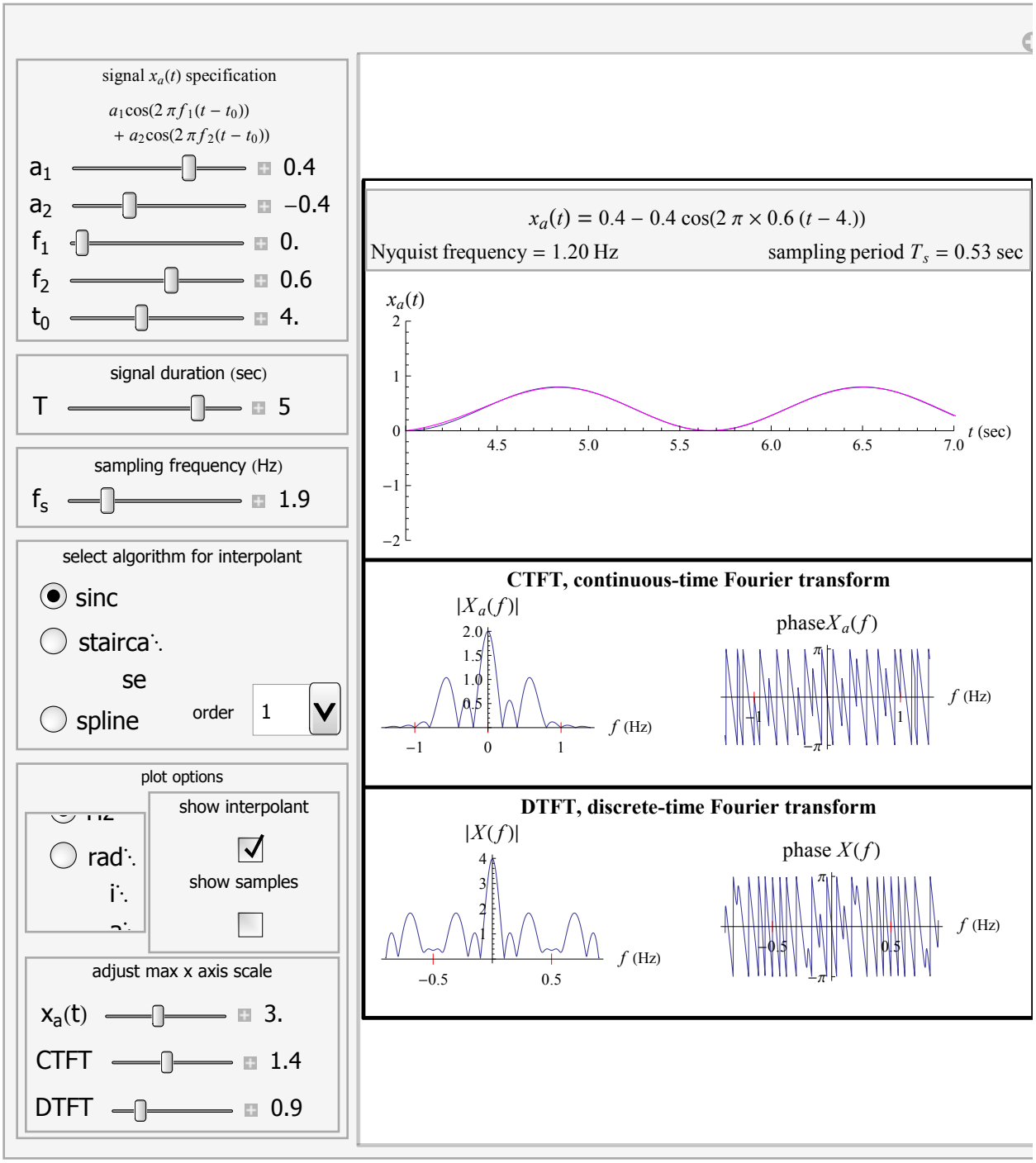
```
Manipulate[

 process[Fs, {f1, f2}, {a1, a2}, T, t0, showSinc = 2, showSamples = 2,
  unitsInHz = 1, maxCTFTscale, maxDTFTscale, maxXscale, interpolant, splineOrder],

 Panel[Labeled[Grid[{
     {Text@Style[Column[{
          Row[{Style["a", Italic]1 , "cos(", 2 π,
            Style["f", Italic]1, "(", Style["t", Italic] - Style["t", Italic]0, "))"}],
          Row[{" + ", Style["a", Italic]2 , "cos(", 2 π, Style["f", Italic]2, "(",
            Style["t", Italic] - Style["t", Italic]0, "))"}]
         }], 10]},
     {Control[{{a1, N[9/10], " a1"}, -1., 1, 1/10, ImageSize → Small, Appearance → "Labeled" }]},
     {Control[{{a2, N[8/10], " a2"}, -1., 1, 1/10, ImageSize → Small, Appearance → "Labeled" }]},
     {Control[{{f1, N[9/10], " f1"}, 0., 1, 1/10, ImageSize → Small, Appearance → "Labeled" }]},
     {Control[{{f2, N[2/10], " f2"}, 0., 1, 1/10, ImageSize → Small, Appearance → "Labeled" }]},
     {Control[{{t0, 0., " t0"}, 0., 10, 1/10, ImageSize → Small, Appearance → "Labeled" }]}
    }, Spacings → {0, 0}], Text@
    Style[Row[{"signal ", Style["x", Italic]Style["a",Italic] , "(", Style["t", Italic], ") specification"}], 10],
   {{Top, Center}}], FrameMargins → 1, ImageSize → 190],

 Panel[Labeled[
   Control[{{T, 6, " T"}, 1, 6, 1, ImageSize → Small, Appearance → "Labeled" }],
   Style["signal duration (sec)", 10], {{Top, Center}}, Spacings → {0, 0}],
  FrameMargins → 3, ImageSize → 190],

 Panel[Labeled[
   Control[{{Fs, 4.5, " fs"}, .1, 10, 1/10, ImageSize → Small, Appearance → "Labeled"}],
   Style["sampling frequency (Hz)", 10], {{Top, Center}}, Spacings → {0, 0}],
  FrameMargins → 3, ImageSize → 190],

 Panel[Labeled[Grid[{
     {
      Control[{{interpolant, 2, ""}, {1 → "sinc", 2 → "staircase", 3 → "spline"},
        ControlType -> RadioButtonBar, Appearance → "Vertical"}]
      ,
      Control[{ {splineOrder, 0, Style["order", 10]},
        {0 → Style["0", 9],
         1 → Style["1", 9],
         2 → Style["2", 9],
         3 → Style["3", 9],
         4 → Style["4", 9]
        },
        Enabled → Dynamic@TrueQ[interpolant == 3], ControlType → PopupMenu, ImageSize → Tiny}]
     }
```

```
    }, Spacings → {1, 1}, BaselinePosition → Bottom, Alignment → Bottom],
   Style["select algorithm for interpolant", 10], {{Top, Center}}],
  Alignment → {Center, Center}, FrameMargins → 1, ImageSize → 190],

 Panel[Labeled[Grid[{
     {Panel[Labeled[
        Control[{{unitsInHz, 2, ""}, {1 → "Hz", 2 → "radial"}, ControlType -> RadioButtonBar,
          Appearance → "Vertical"}], Style["x axis units", 10], {{Top, Center}}],
       Alignment → {Center, Center}, FrameMargins → 1, ImageSize → {70, 70}],

      Panel[Grid[{{
          Labeled[Control[{{showSinc, 2, ""}, {1, 2}, ControlType → Checkbox}],
           Style["show interpolant", 10], {{Top, Center}}], SpanFromLeft},
         {Labeled[Control[{{showSamples, 1, ""}, {1, 2}, ControlType → Checkbox}],
           Style["show samples", 10], {{Top, Center}}], SpanFromLeft}
        }, Spacings → {0, 0}], Alignment → {Center, Center}, FrameMargins → 1, ImageSize → 110],
      SpanFromLeft},

     {Panel[Labeled[Column[{
          Control[{{maxXscale, 7, "  xa(t)"}, .1, 7, 1/10, ImageSize → Tiny, Appearance → "Labeled" }],
          Control[{{maxCTFTscale, 2, "  CTFT"}, .1, 3, 1/10, ImageSize → Tiny, Appearance → "Labeled" }],
          Control[{{maxDTFTscale, .5, "  DTFT"}, .5, 3, 1/10, ImageSize → Tiny, Appearance → "Labeled" }]},
         Alignment → Center
        ], Style["adjust max x axis scale", 10], {{Top, Center}}
       ], Alignment → {Center, Center}, FrameMargins → 0, ImageMargins → 0, ImageSize → 180], SpanFromLeft}},
    Spacings → {0, 0}
   ], (*Grid[...]*)
   Style["plot options", 9], {{Top, Center}}, Spacings → {0, 0}], (*Labeled*)
  Alignment → {Center, Center}, FrameMargins → 0, ImageMargins → 0, ImageSize → 190], (*Panel*)

 ControlPlacement → Left,
 FrameMargins → 0,
 ImageMargins → 0,
 ContinuousAction → False,
 SynchronousUpdating → False,
 AutorunSequencing → {1, 2, 3},
 Initialization :→
  {
   (* The following 3 functions are used by plotting to control ticks*)
   xticksForDTFT[unitsInHz_, tickLen_][min_, max_] :=
    Module[{}, If[unitsInHz, {{-4, -4, tickLen, Red}, {-3, -3, tickLen, Red}, {-1.5, -1.5, tickLen, Red},
       {-.5, -.5, tickLen, Red}, {.5, .5, tickLen, Red}, {1.5, 1.5, tickLen, Red}, {3, 3, tickLen, Red},
       {4, 4, tickLen, Red}}, {{-3, -6 Pi, tickLen, Red}, {-1.5, -3 Pi, tickLen, Red},
       {-1/2, -Pi, tickLen, Red}, {1/2, Pi, tickLen, Red}, {1.5, 3 Pi, tickLen, Red}, {3, 6 Pi, tickLen, Red}}
     ]
    ];

   ticksForCTFTinRadian[tickLen_][min_, max_] := Module[{},
     Table[{i, i*2 Pi, .05, Red}, {i, Ceiling[min], Floor[max], 1}]];

   ticksForCTFTinHz[tickLen_][min_, max_] := Module[{},
     Table[{i, i, tickLen, Red}, {i, Ceiling[min], Floor[max], 1}]];

   (* This is the sinc interpolation formula *)
   (* T: Sampling period *)
   (* samples: list of samples, sequence of numbers *)
   (* t: the instance of time to evaluate interpolant at *)
   sincInterpolate[t_, T_, samples_] := Module[{k},
     Sum[samples[[k]]*Sinc[Pi/T (t - (k - 1) T)], {k, 1, Length[samples]}]
    ];

   generateSignalExpression[fUser_, a_, t0_] := Module[{m, s, xa, xa2, xa1, c},
```

```
  m = Abs[a[[1]]];
  s = Text@If[fUser[[1]] == 0, ToString[m],
     Row[{m, " cos(2 π x ", ToString[fUser[[1]]], " (", Style["t", Italic], " - ", ToString[t0], "))"}]];

  If[m ≠ 0, If[a[[1]] < 0, xa1 = Text@Row[{"- ", s}], xa1 = s]];

  m = Abs[a[[2]]];
  s = Text@If[fUser[[2]] == 0, ToString[m],
     Row[{m, " cos(2 π x ", ToString[fUser[[2]]], " (", Style["t", Italic], " - ", ToString[t0] , "))"}]];

  If[m ≠ 0, If[a[[2]] < 0, xa2 = Text@Row[{" - ", s}], xa2 = Text@Row[{" + ", s}]]];

  Which[
   a[[1]] == 0 && a[[2]] == 0, xa = Text["0"],
   a[[1]] == 0, xa = xa2,
   a[[2]] == 0, xa = xa1,
   True, xa = Text@Row[{xa1, xa2}]
  ];

  StringReplace[ToString[xa, FormatType → TraditionalForm],
   c : LetterCharacter ~~ "$" ~~ DigitCharacter .. :> c]
 ];

(* Main Manipulate expression *)
process[Fs_, fUser_, a_, duration_, t0_, showSinc_, showSamples_,
  unitsInHz_, maxCTFTscale_, maxDTFTscale_, maxXscale_, interpolant_, splineOrder_] :=
 Module[{nSamples, data, xa, ctft, dtft, nHarmonics = 2, xaString, sincNumberOfPoints = 10, sincDeltaTime,
   xaReconstruct, pSinc, panel, disks, lines, dtftTerm, tickLen = .05, from, to, ticksFontSize = 9,
   aspectRatio = .49, imageSize = 186, spectraPlotOptions, k, f, t, i, n, Ts = 1 / Fs, p1, p2, p3, p5, p6, x},

  xaString = generateSignalExpression[fUser, a, t0];
  xa = Sum[
    a[[k]] Cos[2 Pi fUser[[k]] (t - t0)] (UnitStep[t - t0] - UnitStep[t - t0 - duration]), {k, 1, nHarmonics}];
  nSamples = Floor[duration / Ts] + 1;

  (*for speed, I enter the DTFT term directly*)
  dtftTerm = N[a[[1]] Cos[2 fUser[[1]] π (n Ts)] + a[[2]] Cos[2 fUser[[2]] π (n Ts)]];
  dtft = Sum[dtftTerm * Exp[-I 2 Pi f n], {n, 0, nSamples}] * Exp[-I 2 Pi f Floor[t0 / Ts]];

  (*for speed, I enter the CTFT term directly*)
  ctft = Chop[
     1/(4 π) i (a[[1]] (e^(-2 i π (-fUser[[1]] duration+f (duration+t0))) / (f - fUser[[1]]) +
          e^(-2 i π (fUser[[1]] duration+f (duration+t0))) / (f + fUser[[1]]) -
          2 e^(-2 i f π t0) f / (f^2 - fUser[[1]]^2)) +
        a[[2]] (e^(-2 i π (-fUser[[2]] duration+f (duration+t0))) / (f - fUser[[2]]) +
          e^(-2 i π (fUser[[2]] duration+f (duration+t0))) / (f + fUser[[2]]) -
          2 e^(-2 i f π t0) f / (f^2 - fUser[[2]]^2)))];

  from = t0;
  to = t0 + maxXscale;
  If[to > t0 + duration, to = t0 + duration];
  data = Table[{t0 + (i - 1) * Ts, dtftTerm /. n → (i - 1)}, {i, 1, nSamples}];

  (* This puts little dots the sample location *)
  disks =
   Table[{PointSize[.014], Red, Opacity[1], Point[{data[[i, 1]], data[[i, 2]]}]}, {i, 1, nSamples}];

  (* This draws the samples lines *)
  lines = Table[Line[{{data[[i, 1]], 0}, {data[[i, 1]], data[[i, 2]]}}], {i, 1, nSamples}];

  p1 = Plot[xa, {t, from, to}, AxesOrigin → Automatic,
    Ticks → {Automatic, Automatic},
    ImagePadding → {{24, 45}, {5, 5}},
```

```
  PlotRange → {{from, to}, {-2, 2}},
  ImageMargins → 0,
  ImageSize → 380,
  AspectRatio → .40,
  AxesLabel → {Style[Row[{Style["t", Italic], " (sec)"}], 10],
    Style[Row[{Style["x", Italic]Style["a",Italic], "(", Style["t", Italic], ")"}], 12]},
  PlotLabel → If[Max[fUser] * 2 > Fs, Style["Warning: signal is undersampled!", Red]],
  TicksStyle -> Directive[ticksFontSize],
  Epilog → {If[showSamples, {{Dashed, Red, disks}, lines}, Text[" "]]}
 ];

spectraPlotOptions =
 {PlotRange → All, TicksStyle -> Directive[ticksFontSize], ImageSize → imageSize, AxesOrigin → {0, 0},
  AspectRatio → aspectRatio, Exclusions → None};

p2 = Plot[ComplexExpand[Abs[ctft]], {f, -maxCTFTscale, maxCTFTscale}, Evaluate[spectraPlotOptions],
  ImagePadding → {{10, 55}, {25, 20}},
  Ticks → {If[unitsInHz, ticksForCTFTinHz[tickLen], ticksForCTFTinRadian[tickLen]], Automatic},
  AxesLabel → {Row[{If[unitsInHz, Style["f", 10, Italic], Style["Ω", 10]],
      Style[If[unitsInHz, " (Hz)", " (rad/sec)"], 9]}],
    If[unitsInHz, Style[Row[{"|", Style["X", Italic]Style["a",Italic], "(", Style["f", Italic], ")|"}],
      12], Style["|Xa(Ω)|", 12]]}
 ];

p3 = Plot[ComplexExpand[Arg[ctft]], {f, -maxCTFTscale, maxCTFTscale}, Evaluate[spectraPlotOptions],
  ImagePadding → {{10, 55}, {5, 20}},
  Ticks → {If[unitsInHz, ticksForCTFTinHz[tickLen], ticksForCTFTinRadian[tickLen]], {-Pi, 0, Pi}},
  AxesLabel → {Row[{If[unitsInHz, Style["f", Italic, 10], Style["Ω", 10]],
      Style[If[unitsInHz, " (Hz)", " (rad/sec)"], 9]}],
    If[unitsInHz, Style[Row[{"phase", Style["X", Italic]Style["a",Italic], "(", Style["f", Italic], ")"}],
      12], Style[Row[{"phase ", Style["X", Italic]Style["a",Italic], "(Ω)"}], 12]]}
 ];

Which[interpolant == 1 || interpolant == 2,
 {
  Which[interpolant == 1, (*sinc*)
   {
    sincDeltaTime = Ts / (sincNumberOfPoints - 1);
    xaReconstruct = Table[{t + t0,
       sincInterpolate[t, Ts, data[[All, 2]] ]}, {t, 0, duration, sincDeltaTime}
     ];
   },
   interpolant == 2, (*stairacse*)
   xaReconstruct = Table[{
       {(t0 + (i - 1) * Ts) - Ts / 2, data[[i, 2]]},
       {(t0 + (i - 1) * Ts) + Ts / 2, data[[i, 2]]},
       {(t0 + (i - 1) * Ts) + Ts / 2, If[i == nSamples, 0, data[[i + 1, 2]]]}},
      {i, 1, nSamples}];
  ];
  pSinc = ListPlot[xaReconstruct,
    Joined → True, PlotRange → All, PlotStyle → {Magenta}, ImageMargins → 5, ImageSize → imageSize];
 },
 True, (*spline*)
 {
  If[nSamples > 1, (*can't do spline on one sample*)
    pSinc = Plot[Interpolation[data, InterpolationOrder → splineOrder][x],
      {x, data[[1, 1]], data[[-1, 1]]}, ImageMargins → 5,
      ImageSize → imageSize, PlotStyle → {Magenta}], pSinc = ListPlot[{0, 0}]];
 }
];

p5 = Plot[Abs[dtft], {f, -maxDTFTscale, maxDTFTscale}, Evaluate[spectraPlotOptions],
```

```
  ImagePadding → {{10, 50}, {25, 20}},
  Ticks → {xticksForDTFT[unitsInHz, tickLen], Automatic},
  AxesLabel → {Row[{If[unitsInHz, Style["f", 10, Italic], Style["ω", 10]],
       Style[If[unitsInHz, " (Hz)", " (rad)"], 9]}],
    If[unitsInHz, Style[Row[{"|", Style["X", Italic], "(", Style["f", Italic], ")|"}], 12],
     Style[Row[{"|", Style["X", Italic], "(ω)|"}], 12]]}
 ];

p6 = Plot[Arg[dtft], {f, -maxDTFTscale, maxDTFTscale}, Evaluate[spectraPlotOptions],
  Ticks → {xticksForDTFT[unitsInHz, tickLen], {-Pi, Pi}},
  ImagePadding → {{10, 50}, {5, 20}},
  AxesLabel → {Row[{If[unitsInHz, Style["f", Italic, 10], Style["ω", 10]],
       Style[If[unitsInHz, " (Hz)", " (rad)"], 9]}],
    If[unitsInHz, Style[Row[{"phase ", Style["X", Italic], "(", Style["f", Italic], ")"}], 12],
     Style[Row[{"phase ", Style["X", Italic], "(", Style["ω", Italic], ")"}], 12]]}
 ];

panel = Panel[Grid[{
    {Null, SpanFromLeft},
    {Text@Row[{Style["x", Italic, 12]Style["a",Italic],
        "(", Style["t", Italic, 12], ") = ", Style[xaString, 12]}], SpanFromLeft},
    {Null, SpanFromLeft}, {Text@Style[Row[{"Nyquist frequency = ",
         NumberForm[Max[fUser]*2, {3, 2}], " Hz"}], 12], Text@Style[Row[{"sampling period ",
         Style["T", Italic]Style["s",Italic], " = ", NumberForm[Ts, {3, 2}], " sec"}], 12]}
   }, Frame → None, Alignment → Center, Spacings → {6, 0}], FrameMargins → 1, ImageMargins → 0];

Grid[
 {
  { Labeled[ If[showSinc, Show[{p1, pSinc}], p1],
    panel, {{Top, Center}}, Spacings → {0, 1}], SpanFromLeft},
  {Labeled[Grid[{{p2, p3}}], Text@Style["CTFT, continuous-time Fourier transform", 12, Bold],
    {{Top, Center}}, Spacings → {0, 0}], SpanFromLeft},
  {Labeled[Grid[{{p5, p6}}], Text@Style["DTFT, discrete-time Fourier transform", 12, Bold],
    {{Top, Center}}, Spacings → {0, 0}], SpanFromLeft}
 },
 Alignment → Center,
 Spacings → {0, 1},
 Frame → All
]]]
```



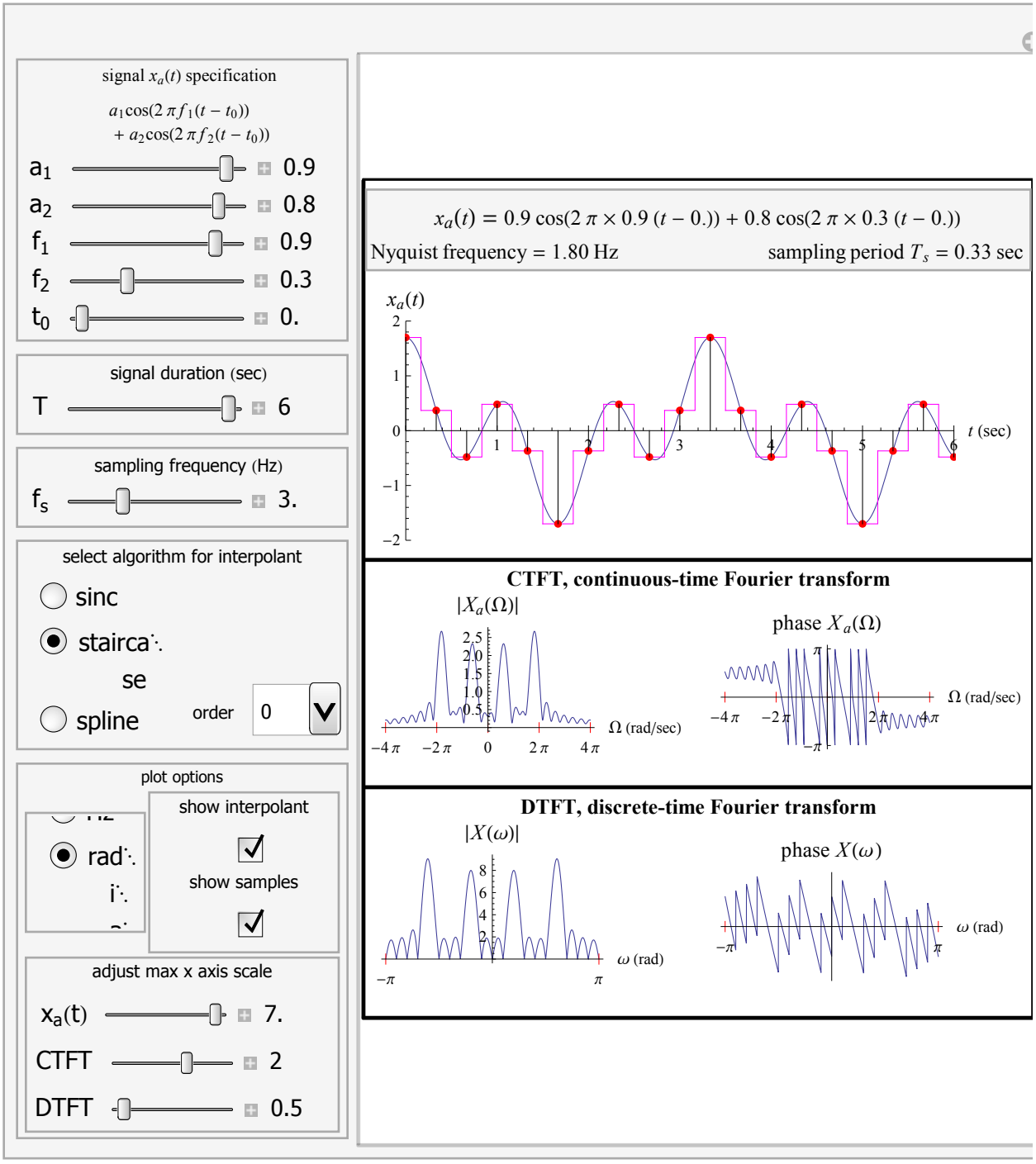
Caption

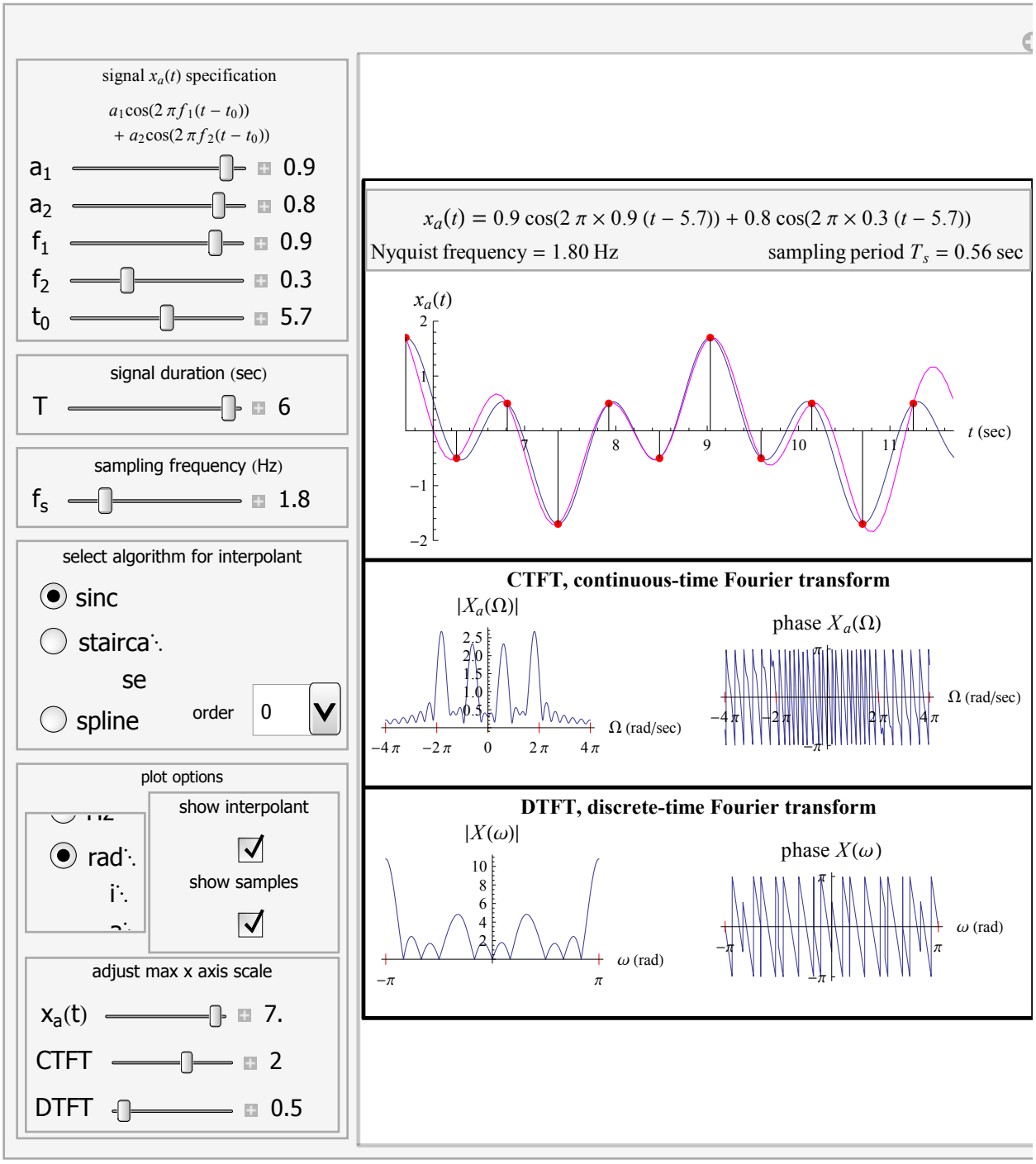
The main purpose of this Demonstration is to illustrate the relation between the continuous-time Fourier transform (CTFT) of a continuous time signal $x_a(t)$ and the discrete time Fourier transform (DTFT) of the discrete signal $x(n)$ generated from $x_a(t)$ by sampling. This Demonstration also illustrates the use of different algorithms to reconstruct the signal $x_a(t)$ from the signal $x(n)$.

The time domain signal $x_a(t)$ is made of two harmonics. You can vary the amplitude and the frequency of each harmonic as well as the sampling frequency, the duration of $x_a(t)$, and the delay time. The spectrum of $x_a(t)$ and of $x(n)$ are also displayed (the magnitude and the phase spectra). A number of plotting options are available to help analyze the results generated.

The following important relation between the CTFT and the DTFT is observed: The CTFT is an aperiodic and continuous function. The DTFT is also a continuous function, but it is a periodic function with a period of 2π. The maximum frequency present in the DTFT is π in radians (or $\frac{1}{2}$ in cycles). In addition, there is a scaling effect: the magnitude spectrum of $x(n)$ is $\frac{1}{T}$ of the magnitude spectrum of $x_a(t)$, where T is the sampling period. The units of the DTFT are radians, while the units of the CTFT are in radians per second.

Thumbnail





Details (optional)

The CTFT of a continuous time signal $x_a(t)$ is defined as $X(\Omega) = \int_{-\infty}^{\infty} x_a(t)\exp(-I\,\Omega\,t)\,dt$, where Ω is in radians per second. The DTFT of a discrete signal $x(n)$ is defined as $X(\omega) = \sum_{n=-\infty}^{\infty} x(n)\exp(-I\,\omega\,n)$, where ω is in radians. In *Mathematica*, the built-in function `FourierTransform` implements the CTFT and the function `FourierSequenceTransform` implements the DTFT.

In this Demonstration, a signal $x_a(t)$ made up of two harmonics is used to analyze the process of sampling and generating the DTFT. The signal $x_a(t)$ is defined as $x_a(t) = a_1\cos(2\pi f_1(t-t_0)) + a_2\cos(2\pi f_2(t-t_0))$, where f_i is the frequency of each harmonic in units of cycles per second (Hz), a_i is the amplitude of each harmonic in units appropriate to the nature of the signal (such as volts), and t_0 is the

amount of delay in units of seconds. The signal $x_a(t)$ is of limited duration; you can vary the duration. Since $x_a(t)$ is time-limited, its spectrum will not be band-limited. In practice, this is handled by passing the signal through an antialiasing filter before sampling it. In this Demonstration this filter has not been implemented. Hence some aliasing will be present even if the sampling frequency is more than the Nyquist frequency. For the purposes of this Demonstration, it was not necessary to implement an antialiasing filter before sampling.

To reconstruct the time domain signal from the discrete time signal, one of the following algorithms can be selected: sinc interpolation, staircase interpolation, or spline interpolation (of order up to four). It was found that to obtain good reconstruction using the staircase and spline algorithms, the sampling frequency needs to be much higher than the Nyquist frequency.

You can vary the x axis plot range to see more details in the plot. Other plotting options are available to allow the change of units of the spectrum from Hz to radians and to be able to superimpose the samples and the reconstructed signal on top of the original signal $x_a(t)$ in order to better see the effect of sampling.

To make the signal $x_a(t)$ become one harmonic only, you can set the amplitude of the other harmonic to zero. To make the signal $x_a(t)$ become just a constant (DC signal), you can set the frequency f_i of both harmonics to zero.

Control Suggestions (optional)

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

Search Terms (optional)

FourierTransform
FourierSequenceTransform
continuous time Fourier transform
discrete time Fourier transform
CTFT
DTFT
sinc interpolation
staircase
spline
Nyquist
sampling
aliasing
band limited
time limited

Related Links (optional)

FourierTransform
FourierSequenceTransform
Sampling Theorem
Fourier Transform

Authoring Information

Contributed by: Nasser M. Abbasi