MathModelica® System Designer Professional

Frequency Analysis of Simulation Data

1 Abstract

This notebook is an example of how *Mathematica* can be utilized for frequency analysis. First we will define a Modelica model of a weak axes, similar to what is described in the Getting Started examples of the model editor, then we will develop a simple *Mathematica* program to perform fourier analysis.

2 Initialization

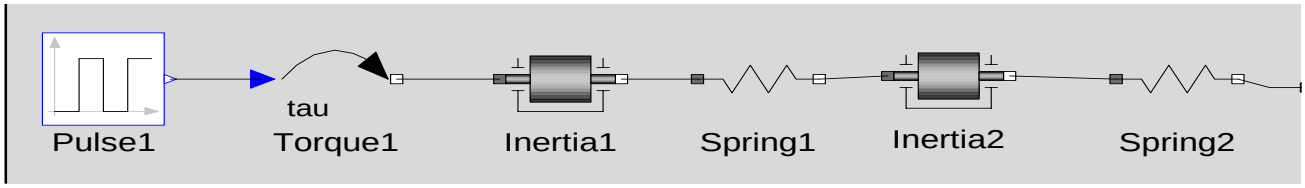
We begin by loading *MathModelica system Designer*. This is done by evaluating the following command:

```
Needs["MathModelica`"];
```

3 Frequency Analysis

The aim of this example is to take a model and show how *Mathematica* can be used to analyze simulation results. In this case we will do a frequency analysis on a weak axis model.

3.1 The Model



Consider a weak axis excited by a torque pulse train. The axis is modeled by three segments joined by two torsion springs. The model can easily be created in the model editor (see the Getting Started examples for detailed explanation on how to do this). When the model has been composed it can be downloaded to the notebook by using the PrintDefinition command. As this model is allready available in the Getting Started package we can start download it directly

```
PrintDefinition[GettingStarted.Professional.WeakAxis];
```

```
model WeakAxis
  annotation(Diagram(coordinateSystem(extent={{-148.5,-105},{
  Modelica.Blocks.Sources.Pulse pulse1(width=1,period=200) ar
  Modelica.Mechanics.Rotational.Torque torque1 annotation(Pla
  Modelica.Mechanics.Rotational.Inertia inertia3 annotation(P
  Modelica.Mechanics.Rotational.Spring spring2(c=1) annotatic
  Modelica.Mechanics.Rotational.Inertia inertia2 annotation(P
  Modelica.Mechanics.Rotational.Inertia inertia1 annotation(P
  Modelica.Mechanics.Rotational.Spring spring1(c=0.7) annotat

equation
  connect(inertia1.flange_b,spring1.flange_a) annotation(Line
  connect(spring2.flange_b,inertia3.flange_a) annotation(Line
  connect(pulse1.y,torque1.tau) annotation(Line(visible=true,
  connect(spring1.flange_b,inertia2.flange_a) annotation(Line
  connect(torque1.flange_b,inertia1.flange_a) annotation(Line
  connect(inertia2.flange_b,spring2.flange_a) annotation(Line
end WeakAxis;
```

It is possible to simulate the model GettingStarted.Professional.WeakAxis directly,but in this case we choose to evaluate the cell above to make the model available on the top level.

3.2 Simulation

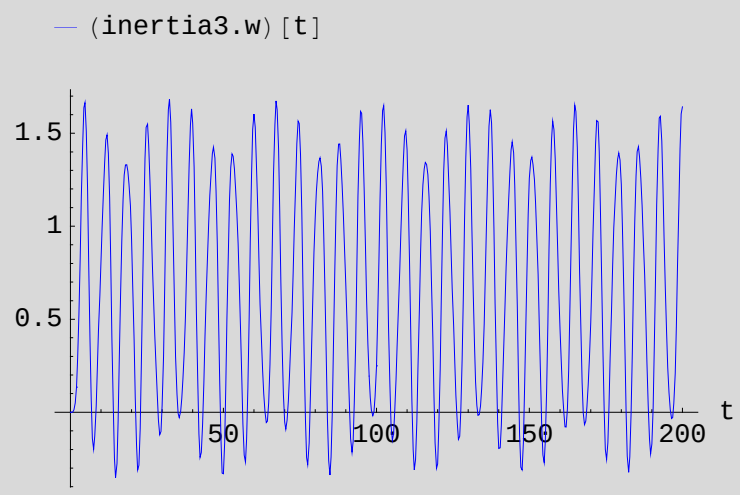
When the model has been created and evaluated it can be simulated. We simulate the model for 200 seconds using the simulate command.

```
Simulate[WeakAxis, {t, 0, 200}]

<SimulationData:  :  : {0., 200.} :
  509 data points : 2 events : 59 variables>
{inertia1.a, inertia1.flange_a.ϕ, inertia1.flange_a.τ,
   inertia1.flange_b.ϕ, inertia1.flange_b.τ, inertia1.J, inertia1.ϕ,
   inertia1.w, inertia2.a, inertia2.flange_a.ϕ, inertia2.flange_a.τ,
   inertia2.flange_b.ϕ, inertia2.flange_b.τ, inertia2.J, inertia2.ϕ,
   inertia2.w, inertia3.a, inertia3.flange_a.ϕ, inertia3.flange_a.τ,
   inertia3.flange_b.ϕ, inertia3.flange_b.τ, inertia3.J,
   inertia3.ϕ, inertia3.w, pulse1.amplitude, pulse1.offset,
   pulse1.period, pulse1.startTime, pulse1.T0, pulse1.T_width,
   pulse1.width, pulse1.y, spring1.c, spring1.flange_a.ϕ,
   spring1.flange_a.τ, spring1.flange_b.ϕ, spring1.flange_b.τ,
   spring1.phi_rel, spring1.phi_rel0, spring1.τ, spring2.c,
   spring2.flange_a.ϕ, spring2.flange_a.τ, spring2.flange_b.ϕ,
   spring2.flange_b.τ, spring2.phi_rel, spring2.phi_rel0, spring2.τ,
   torque1.bearing.ϕ, torque1.bearing.τ, torque1.flange_b.ϕ,
   torque1.flange_b.τ, torque1.τ, (inertia1.ϕ)′, (inertia1.w)′,
   (inertia2.ϕ)′, (inertia2.w)′, (inertia3.ϕ)′, (inertia3.w)′}
```

A SimulationData object is returned. This object contains all model parameters and variables, and the results can be plotted using the PlotSimulation command. In this case we plot the angular velocity of the right most axis-segment, inertia3.w.

```
PlotSimulation[{inertia3.w[t]}, {t, 0, 200}];
```



3.3 Analysis

We will use the built-in Fourier function to perform a frequency analysis. As this function takes a list rather then an interpolating function as argument we sample inertia3.w with a sample frequency 4Hz and store the result in a variable called data1..

```
data1 = Table[inertia3.w[t], {t, 0, 200, .25}];
```

Then we remove the mean from data1, storing the result in data2.

```
mean = Apply[Plus, data1] / Length[data1];
```

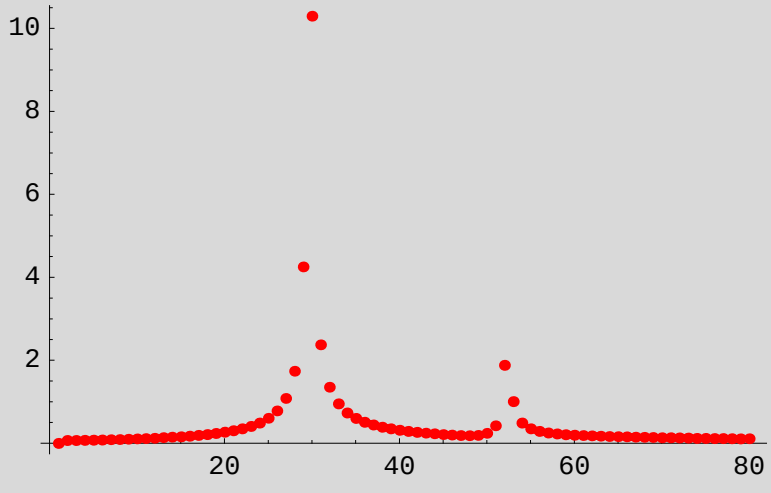
```
data2 = data1 - mean;
```

With his we can compute the absolute values of the discrete Fourier transform.

```
fdata1 = Abs[Fourier[data2]];
```

We can then plot the 80 first points of data.

```
ListPlot[fdata1[[Range[80]]],
  PlotRange -> All, PlotStyle → {Red, PointSize[0.015]}];
```



The result shows two clear peaks at data point 30 and 52 respectively.

```
{fdata1[[30]], fdata1[[52]]}

{10.2943, 1.87824}
```

3.4 Defining a Fourier plot function

To make it easier to do the same analysis on other models we will define a *Mathematica* function for this. As the obtained result showed the peaks for respective data point rather then frequency we will also define the function, FourierPlot, such as it scales the axes such that amplitude of trigonometric components are plotted against frequency (Hz).

Before we define the function we load the color and graphics package

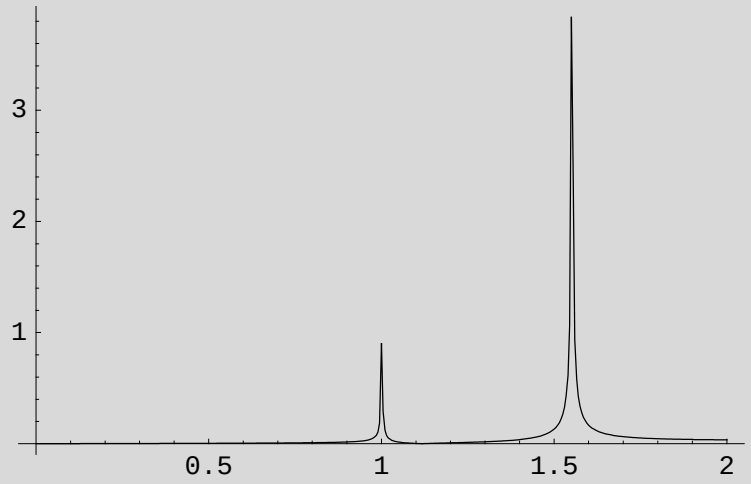
```
Needs["Graphics`Colors`"];
Needs["Graphics`Graphics`"];
```

Now we define a function that takes a simulation variable as input, samples the signal from tmin to tmax with the sampling frequency $\frac{1}{T}$, and plots the result as a function of frequency.

```
FourierPlot[signal_, {t_, tmin_, tmax_}, T_, options___] :=
 Module[{data1, n, mean, data2, fdata1, fdata2, f},
  data1 = Table[signal, {t, tmin, tmax, T}];
  n = Length[data1];
  mean = Apply[Plus, data1] / n;
  data2 = data1 - mean;
  fdata1 = 2 / Sqrt[n] Abs[Fourier[data2]];
  fdata2 = Drop[fdata1, -Round[n / 2.]];
  f = Range[0, 1 / (2 * T), (1 / (2 T) - 0) / Round[n / 2.]];
  ListPlot[Transpose[{f, fdata2}], options];
 ]
```

We make a simple example to verify the function.

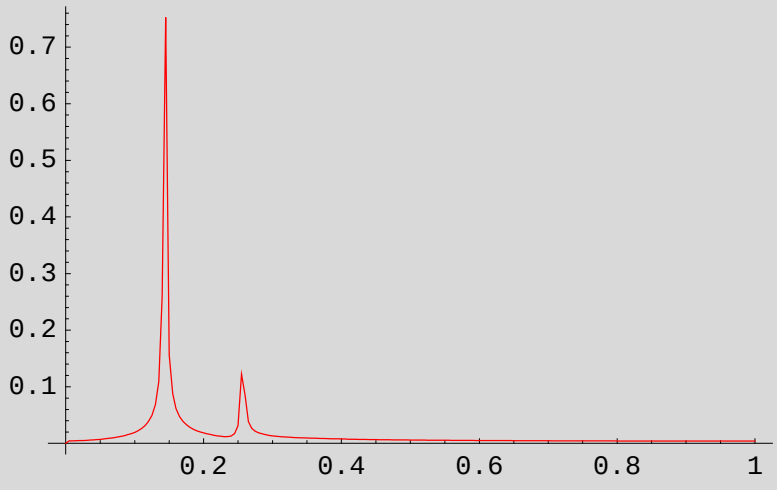
```
FourierPlot[Sin[2 π t] + 5 Sin[2 π 1.55 t],
 {t, 0, 200}, 0.25, PlotRange → All, PlotJoined → True]
```



3.5 Using FourierPlot

We take the same variable as earlier, namely inertia3.w, and sample it with a sample frequency of $\frac{1}{0.5}$ Hz to be sure that we fulfill the sampling theorem, and use the newly defined function to plot the result.

```
FourierPlot[inertia3.w[t], {t, 0, 200}, 0.5,
 PlotRange → All, PlotJoined → True, PlotStyle → Red]
```



We see that the frequency tops are at 0.14 Hz and 0.26 Hz respectively.

For this simple example it can actually be shown that the frequencies of the eigenmodes of the system is given by the imaginary parts of the eigenvalues of the following matrix (c_1 and c_2 are the spring constants).

$$\frac{1}{2\pi}\,\text{Eigenvalues}\left[\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -c_1 & 0 & -c_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ -c_1 & 0 & -c_1 - c_2 & 0 & -c_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -c_2 & 0 & -c_2 & 0 \end{pmatrix} /.\ \{c_1 \to 0.7,\ c_2 \to 1\}\right] //\ \text{Chop}$$

$$\{0.256077\,i,\ -0.256077\,i,\ 0.143344\,i,\ -0.143344\,i,\ 0,\ 0\}$$

Which fits very well with the peaks in the diagram above.