Example of Model Reconstruction Using the Package

Author : Tomasz Konopka
Date : January 2011

The objective of this notebook is to demonstrate model reconstruction using the package.

The input to the analysis is a set of two time-series. In this notebook, these are imported from two text files.

The objective of the analysis is to determine what equations/model structures are consistent with the time-series data.

Loading data

It is nice to clear variables before starting a new computation to avoid conflicts

In[1]:=
```
ClearAll["Global`*"];
```

Load the package. Sometimes the location of the package must be added to the path in order for Mathematica to find it

In[2]:=
```
$Path = Prepend[$Path, "/home/tomasz/MyP_Bruxelles/Autoosc/v0.1"];
Needs["Autoosc`"];
```

To check if the package is loaded properly, execute the following command. If all is ok, it should display some text

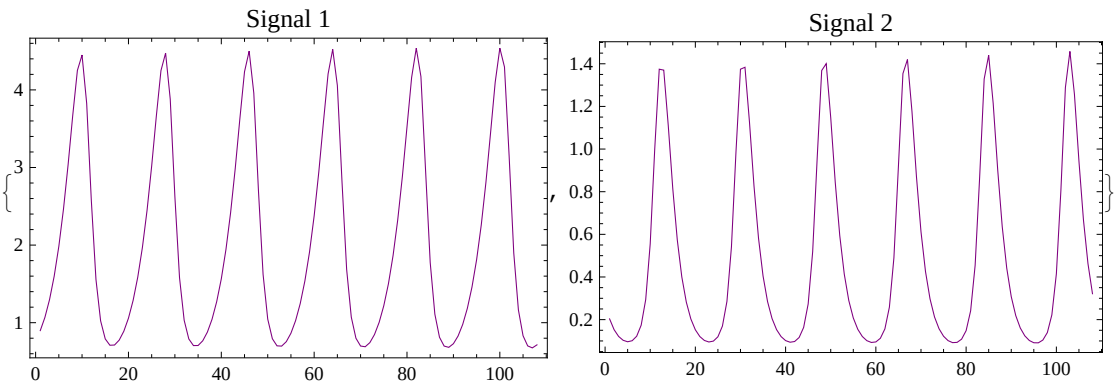
In[4]:=
```
AOAbout[]
```

```
Package Autoosc
Version: 0.1
Author: Tomasz Konopka
```

Read data from disk and show it

In[5]:=
```
LoadedSignals = {0, 0};
LoadedSignals[[1]] = ReadList["/home/tomasz/MyP_Bruxelles/Autoosc/exampledata1.dat"];
LoadedSignals[[2]] = ReadList["/home/tomasz/MyP_Bruxelles/Autoosc/exampledata2.dat"];
{AOPlotModel[LoadedSignals[[1]], 250, "Signal 1"],
 AOPlotModel[LoadedSignals[[2]], 250, "Signal 2"]}
```

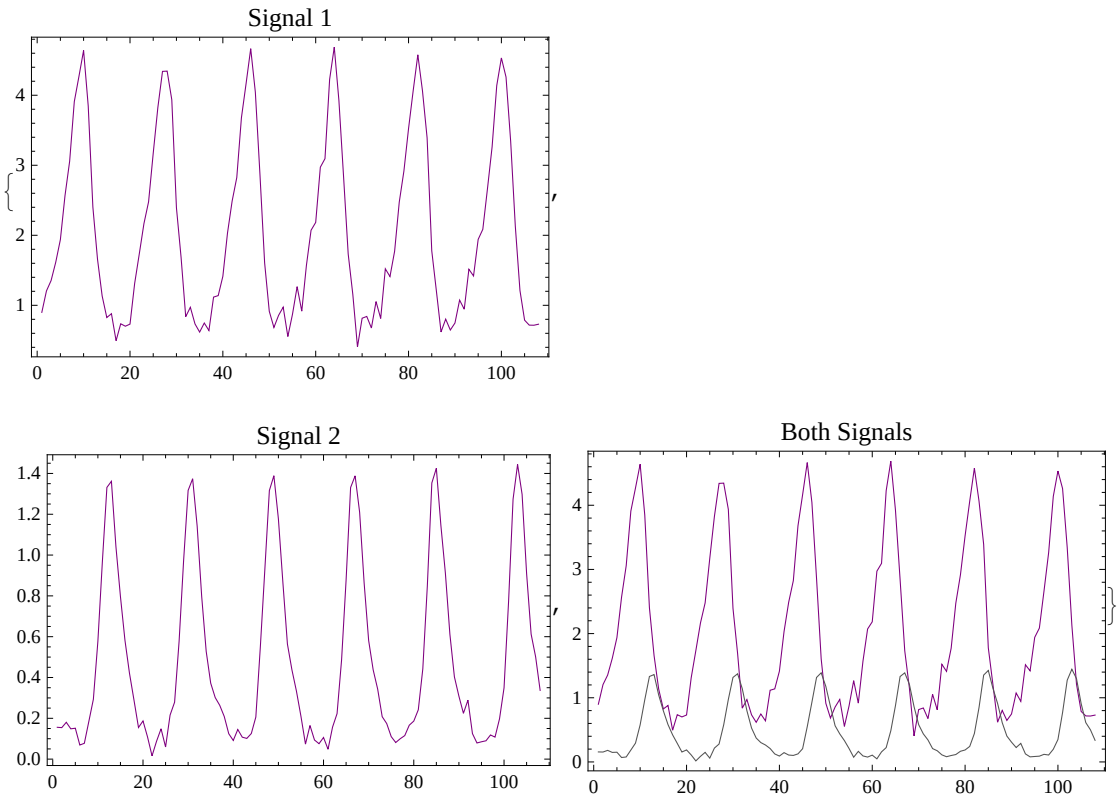
Out[7]=



Add some noise if desired

In[8]:=
```
Signals = {0, 0};
Signals[[1]] = AOCorruptSeries[LoadedSignals[[1]], 8 / 100];
Signals[[2]] = AOCorruptSeries[LoadedSignals[[2]], 8 / 100];
{AOPlotModel[Signals[[1]], 250, "Signal 1"], AOPlotModel[Signals[[2]], 250, "Signal 2"],
 AOPlotModel[{Signals[[1]], Signals[[2]]}, 250, "Both Signals"]}
```

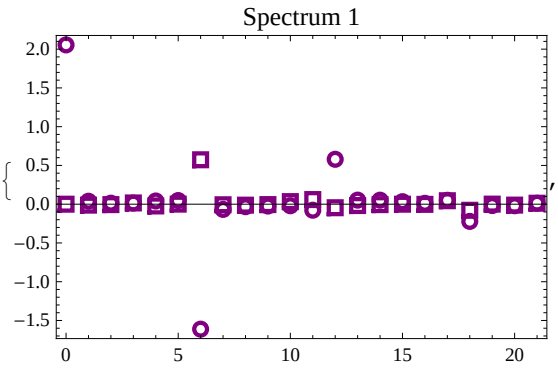
Out[11]=

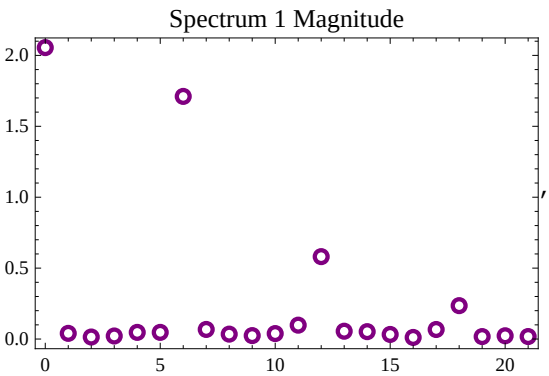


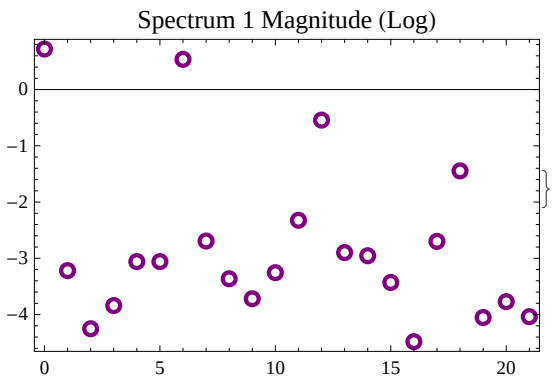
Compute Spectra and show them

In[12]:=
```
Spectra = {0, 0};
Spectra[[1]] = AOComputeSpec[Signals[[1]]];
Spectra[[2]] = AOComputeSpec[Signals[[2]]];
{AOPlotModelSpec[Spectra[[1]], 22, 250, "Spectrum 1"],
 AOPlotModelSpecAbs[Spectra[[1]], 22, 250, False, "Spectrum 1 Magnitude"],
 AOPlotModelSpecAbs[Spectra[[1]], 22, 250, True, "Spectrum 1 Magnitude (Log)"]}
{AOPlotModelSpec[Spectra[[1]], 22, 250, "Spectrum 2"],
 AOPlotModelSpecAbs[Spectra[[2]], 22, 250, False, "Spectrum 2 Magnitude"],
 AOPlotModelSpecAbs[Spectra[[2]], 22, 250, True, "Spectrum 2 Magnitude (Log)"]}
```

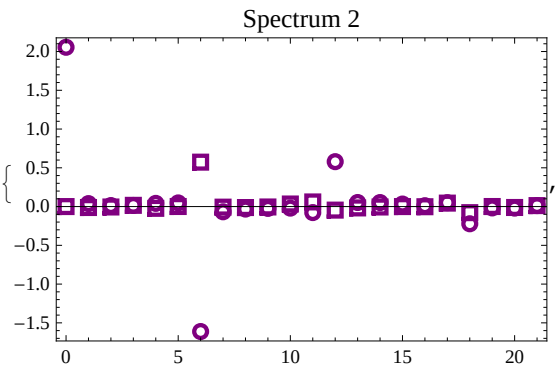
Out[15]=

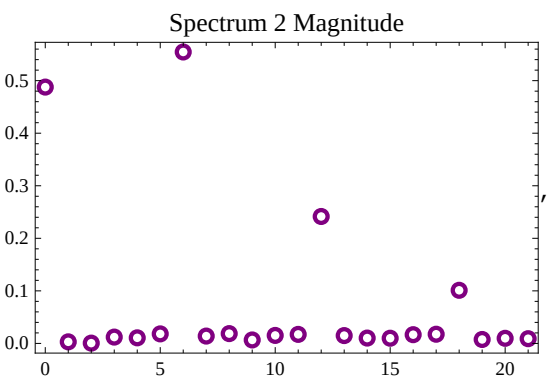

Spectrum 1

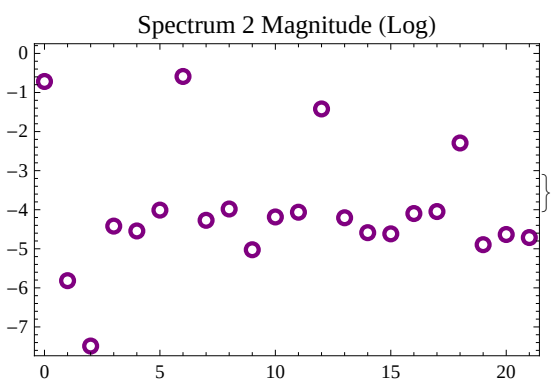

Spectrum 1 Magnitude


Spectrum 1 Magnitude (Log)

Out[16]=


Spectrum 2


Spectrum 2 Magnitude


Spectrum 2 Magnitude (Log)

Preliminaries

Set a variable numcycles, setting the spacing between peaks in the spectra. In this example it is 6

In[17]:=
```
numcycles = 6;
```

Get suggestions for depths. The results here will change differ here for every time a noisy signal is generated

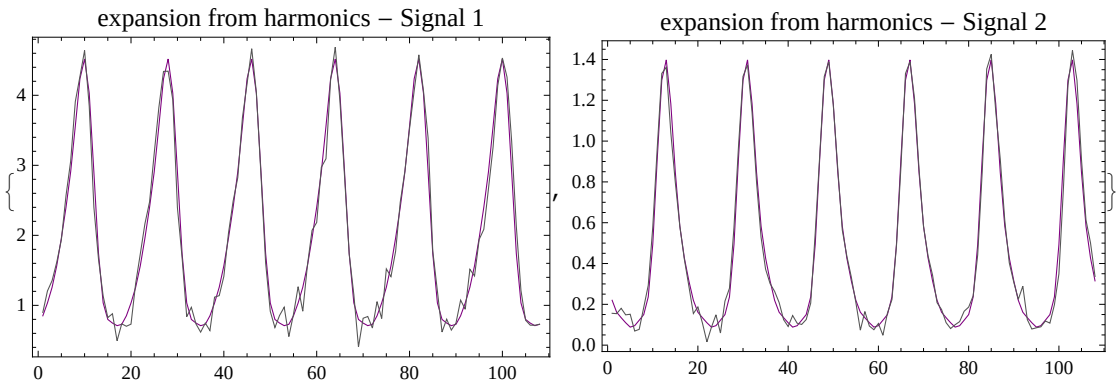
In[18]:=
```
Do[
  Print["Signal ", i, " - suggested depth: ",
    AOSuggestDepth[Spectra[[i]], numcycles, "normal", 1]];
  ,
  {i,
   1,
   2}];
```

```
Signal 1 - suggested depth: 3

Signal 2 - suggested depth: 2
```

The procedures only use the spectral peak heights. Here see how information in those peaks compare with the full signal.

In[19]:=
```
{AOPlotModel[{AOShowAnsatz[Spectra[[1]], numcycles, 4], Signals[[1]]},
  250, "expansion from harmonics - Signal 1"],
 AOPlotModel[{AOShowAnsatz[Spectra[[2]], numcycles, 4], Signals[[2]]},
  250, "expansion from harmonics - Signal 2"]}
```

Out[19]=



Based on the above, decide how many harmonics to use in the procedure below

In[20]:=
```
numharmonics = 4;
```

Set the type of models to analyze. Here chose all models with maximal order equal to 1

In[21]:=
```
AOBuildEqList[1];
Print["Number of model structures considered will be ", Length[AOGetEqList[]]];
```

```
Number of model structures considered will be 60
```

Set tolerance levels. The arrays can be of different lengths from the specified depth

In[23]:=
```
AOGetLevels[]
GoDepth = 2;
ToleranceArray = {1, 1, 1};
RIToleranceArray = {0.5, 1, 2};
parameterranges = {{pp1, -6000, 6000}, {pp2, 0, 300}, {qq1, -6000, 6000}, {qq2, 0, 300}};
AOSetLevels[GoDepth, parameterranges, ToleranceArray, RIToleranceArray]
AOGetLevels[]
```

```
depth: 3
Tolerances: {1, 1, 1}
ReImTolerances: {1, 1, 1}
Parameter ranges: {{-1 000 000, 1 000 000}, {0, 1 000 000}, {-1 000 000, 1 000 000}, {0, 1 000 000}}

depth: 2
Tolerances: {1, 1, 1}
ReImTolerances: {0.5, 1, 2}
Parameter ranges: {{-6000, 6000}, {0, 300}, {-6000, 6000}, {0, 300}}
```

Model reconstruction

■ Using the frequency domain

Compute parameters for one model structure.

```
In[30]:= try1 = AOComputeParameters[Spectra[[1]], Spectra[[2]],
            numcycles, numharmonics, {{3, 0, 1}, {6, 1, 0}}, 2, "S", False]
```

$$\text{Out[30]}= \left\{\frac{qq1\, X[t]}{qq2 + X[t]} + pp1\, XD[t] - X'[t],\ \{\{3, 0, 1\}, \{6, 1, 0\}\},\right.$$
$$\{\{1, \{qq1 \to -0.456113 - 0.0289149\, i,\ pp1 \to 0.22046 + 0.0153435\, i,\ qq2 \to -0.222146 - 0.0295642\, i\},$$
$$\{qq1 \to -0.122859 - 0.00936135\, i,$$
$$pp1 \to 0.0580619 + 0.00541862\, i,\ qq2 \to -0.829389 - 0.0391829\, i\}\}\},$$
$$\left.\{2, \{\{qq1 \to -0.719203 - 0.0966277\, i,\ pp1 \to 0.24087 + 0.0885875\, i,\ qq2 \to -0.648559 + 0.57186\, i\},\right.$$
$$\left.\left.\{qq1 \to -0.544159 + 0.388618\, i,\ pp1 \to 0.16734 - 0.143722\, i,\ qq2 \to -0.870645 - 0.310317\, i\}\}\}\right\}$$

The result shows the equation tested, the equation code, and the computed parameter values.
Interpret the result, i.e. apply the parameter and tolerance criteria to suggest whether the model is correct or not

```
In[31]:= AOInterpretParameters[try1[[3]]]
```

```
Out[31]= {False, {}}
```

The result shows whether the parameter values are consistent within the defined tolerance levels. If yes, one set of parameter values is also given.
Alternatively, compute parameters for all model structures. This can take some time.

```
In[32]:= parametersall = {0, 0};
         parametersall[[1]] = AOComputeParametersAllModels[
            Spectra[[1]], Spectra[[2]], numcycles, numharmonics, 2, "A", False];
         parametersall[[2]] = AOComputeParametersAllModels[Spectra[[2]],
            Spectra[[1]], numcycles, numharmonics, 2, "A", False];
```

Since the results from parameter calculations are large, it is usually only interesting to view them one line at a time

```
In[35]:= parametersall[[1, 3]]
```

$$\text{Out[35]}= \left\{pp1 + \frac{qq1\, XD[t]}{qq2 + X[t]} - X'[t],\ \{\{1, 0, 0\}, \{7, 1, 1\}\},\right.$$
$$\{\{1, \{pp1 \to -0.00370169 + 0.0014668\, i,\ qq1 \to -0.047773 - 0.0056316\, i,$$
$$qq2 \to -1.04681 - 0.0495957\, i\}\}\},\ \{2,$$
$$\left.\{\{pp1 \to -0.202238 + 0.0705467\, i,\ qq1 \to -0.202084 + 0.0619082\, i,\ qq2 \to -1.5264 + 0.187874\, i\}\}\}\right\}$$

Interpret all the model structures in a row

In[36]:=
```
interpretations = {0, 0};
interpretations[[1]] = AOInterpretParametersList[parametersall[[1]]];
interpretations[[2]] = AOInterpretParametersList[parametersall[[2]]];
```

The results after interpreations are often small, so it is sometimes interesting to list them all at once.
Get the number of remaining models

In[39]:=
```
Print["Number of remaining models for signal 2 driven by signal 1: ",
  Length[interpretations[[1]]]];
Print["Number of remaining models for signal 1 driven by signal 2: ",
  Length[interpretations[[2]]]];
```

```
Number of remaining models for signal 2 driven by signal 1: 1

Number of remaining models for signal 1 driven by signal 2: 2
```

In[41]:=

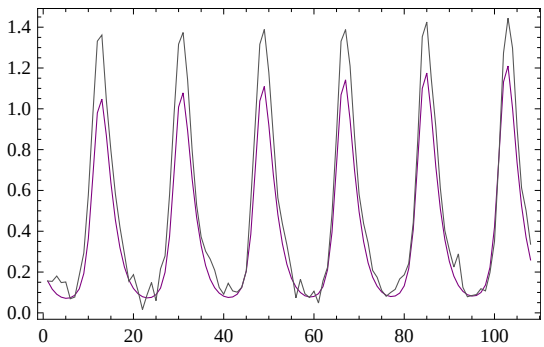
■ Using the time domain

The time domain can be used to check that the suggested model equations do really produce signals similar to those input

Can show plots of reconstructed curves for all of the model structures shortlisted in the frequency domain
Here, the plots show signal 2 being driven by signal 1. The items before the plots show the equation code and the equation itself.

In[42]:=
```
Do[
  temp = interpretations[[1, i]];
  temp2 = AORecreateSignal[Spectra[[1]],
    Signals[[2, 1]], numcycles, numharmonics, temp[[3]], temp[[2]], 0];
  Print[temp[[3]], "   ", temp[[1]], "   ", AOPlotModel[{temp2, Signals[[2]]}, 250, ""]];
  , {i, 1, Length[interpretations[[1]]]}];
```

```
{{2, 1, 0}, {4, 1, 1}}    pp1 X[t] + qq1 X[t] XD[t] - X'[t]
```



Such reconstructions can be used for model selection. First compute all the errors

In[43]:=
```
interpretationsWerr = {0, 0};
interpretationsWerr[[1]] = AOComputeTimeDomainErrList[
   Spectra[[1]], Signals[[2]], numcycles, numharmonics, interpretations[[1]], 0];
interpretationsWerr[[2]] = AOComputeTimeDomainErrList[Spectra[[2]],
   Signals[[1]], numcycles, numharmonics, interpretations[[2]], 0];
```

Then apply interpretation criteria. Here the threshold is set as a multiple of the signal offset

In[46]:=
```
timeinterpretations = {0, 0};
timeinterpretations[[1]] =
  AOInterpretTimeDomainErrList[interpretationsWerr[[1]], 0.04 * Re[Spectra[[2, 1]]]];
timeinterpretations[[2]] = AOInterpretTimeDomainErrList[
   interpretationsWerr[[2]], 0.04 * Re[Spectra[[1, 1]]]];
```

Again print the numbers of remaining models

In[49]:=
```
Print["After time domain analysis, number of remaining models
    for signal 2 driven by signal 1: ", Length[timeinterpretations[[1]]]];
Print["After time domain analysis, number of remaining models for signal
    1 driven by signal 2: ", Length[timeinterpretations[[2]]]];
```

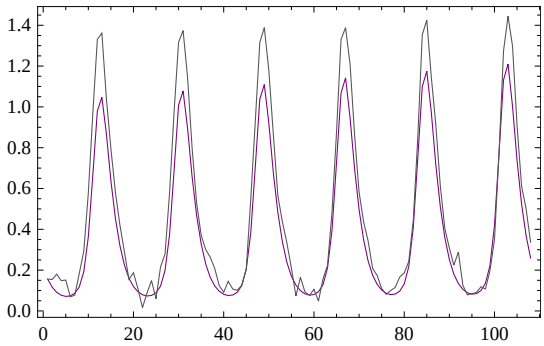
```
After time domain analysis, number of remaining models for signal 2 driven by signal 1: 1

After time domain analysis, number of remaining models for signal 1 driven by signal 2: 2
```

Finally, can look at the reduced set of models and their time domain reconstructions
Here, this is done for the two cases, 1 driving 2, and 2 driving 1

In[51]:=
```
Print["Models where signal 1 drives signal 2"];
Do[
  temp = timeinterpretations[[1, i]];
  temp2 = AORecreateSignal[Spectra[[1]],
    Signals[[2, 1]], numcycles, numharmonics, temp[[3]], temp[[2]], 0];
  Print[temp[[3]], "   ", temp[[1]], "   ", AOPlotModel[{temp2, Signals[[2]]}, 250, ""]];
  , {i, 1, Length[timeinterpretations[[1]]]}];
Print["\nModels where signal 2 drives signal 1"];
Do[
  temp = timeinterpretations[[2, i]];
  temp2 = AORecreateSignal[Spectra[[2]],
    Signals[[1, 1]], numcycles, numharmonics, temp[[3]], temp[[2]], 0];
  Print[temp[[3]], "  ", temp[[1]], "   ", AOPlotModel[{temp2, Signals[[1]]}, 250, ""]];
  , {i, 1, Length[timeinterpretations[[2]]]}];
```

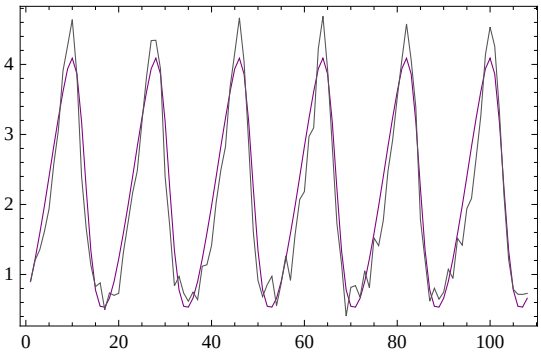
Models where signal 1 drives signal 2

```
{{2, 1, 0}, {4, 1, 1}}    pp1 X[t] + qq1 X[t] XD[t] - X'[t]
```

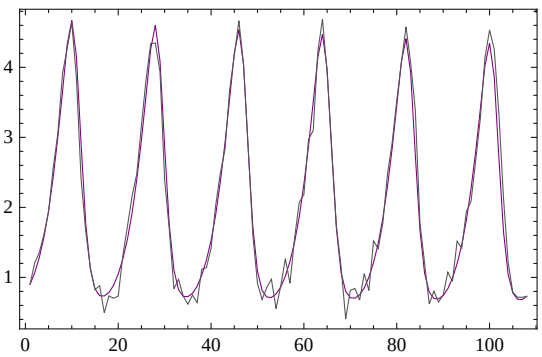


Models where signal 2 drives signal 1

```
{{1, 0, 0}, {3, 0, 1}}    pp1 + qq1 XD[t] - X'[t]
```




```
{{2, 1, 0}, {4, 1, 1}}    pp1 X[t] + qq1 X[t] XD[t] - X'[t]
```



Conclusion

That's all!

The number of model equations listed at the end is usually small. They represent educated guesses for what mechanisms might be responsible for generating the data signals.

The equations originally used to generate the curves in the imported signals were actually the Lotka - Volterra equations. The correct equation code for both the relation between signal 1 driving signal 2 and for signal 2 driving signal 1 is therefore {{2,1,0},{4,1,1}}. Are these codes among the short list?

When noise levels are low and tolerance levels are set at reasonable values, the Lotka-Volterra equations should appear on the short list. They may even be the only solutions left after all the model selection steps.

When noise levels are high, when tolerance levels are set too strictly, or both, the short list might be empty or show model structures that do not include the Lotka-Volterra equations.

In[55]:=