`Autoosc` Package Documentation

Package Name:	`Autoosc` for Mathematica®
Version:	0.1
Date:	January 26, 2011
Author:	Tomasz Konopka
Licence:	GPL 3

Contents

Introduction

The package is written for Mathematica®. Its purpose is to help analysis involved with the following research question:

What are the equations that may be responsible for generating a given set of time-series signals?

When the set of equations considered as candidate descriptions of the signals is finite, the question is well defined and can be answered by testing each equation in turn.

The equations supported by the package are of the form

$$0 = -\dot{\varphi}(t) + W_1(p_1, p_2; \varphi(t)^{r_1}, \psi(t)^{s_1}) + W_1(q_1, q_2; \varphi(t)^{r_2}, \psi(t)^{s_2}) \tag{1}$$

where φ and ψ are some signal, $p_{1,2}$ and $q_{1,2}$ are real parameters, and $r_{1,2}$ and $s_{1,2}$ are integer interaction orders. Each of the terms W_1 and W_2 can be of one of the forms given in Table 1 with $k_{1,2}$ appropriately replaced by $p_{1,2}$ or $q_{1,2}$.

Term	Expression	Term	Expression	Term	Expression
1	k_1	5	$\frac{k_1}{k_2+\varphi^r}$	9	$\frac{k_1}{k_2+\psi^s}$
2	$k_1\varphi^r$	6	$\frac{k_1\varphi^r}{k_2+\varphi^r}$	10	$\frac{k_1\varphi^r}{k_2+\psi^s}$
3	$k_1\psi^s$	7	$\frac{k_1\psi^s}{k_2+\varphi^r}$	11	$\frac{k_1\psi^s}{k_2+\psi^s}$
4	$k_1\varphi^r\psi^s$	8	$\frac{k_1\varphi^r\psi^s}{k_2+\varphi^r}$	12	$\frac{k_1\varphi^r\psi^s}{k_2+\psi^s}$

Table 1: **Taxonomy of terms to appear in model equations.** k_1 and k_2 are free parameters (when two terms are added together, their parameters are distinct, e.g. p_1, p_2, for the first term and q_1, q_2 for the second term). The integers r, s set the order of the interaction.

Global Variables

Some of the output from the package functions involve variables in addition to numerical values. In order for this type of output (created within the package functions) to be easily accessible to the user, the scope of these variables is set to global. The variable names used in this way in the package are:

```
pp1, pp2, qq1, qq2, t, X[t], XD[t]
```

The first four variables stand for the parameters appearing in the model equations. `t` is the time variables. The last two variables stand for the primary and driving signal, respectively.

The package functions assume that these variables do not have values assigned to them. This condition must be satisfied through the use of the package to ensure correct functionality.

Package Functions

This section describes the Mathematica package `Autoosc` used for most of the presented analysis. Example of using the package are provided in separate notebooks.

`AOAbout[]`

Purpose:

to give some information about the package

Arguments:

none

Output:

a printed message containing the version and date of the package code.

`AOBuildEqList[maxorder]`

Purpose:

to recreate a list of model structures

Arguments:

`maxorder` an integer representing the maximal power of the signals in the model equations

Output:

none. The new equation list can be retrieved using `AOGetEqList`

`AOComputeParameters[ser1, ser2, numcyc, numharm, modeldef, depth, numer, toquiet]`

Purpose:

to compute one or more sets of values for parameters appearing in a model equation.

Arguments:

`ser1`	the spectrum of the driving signal ψ
`ser2`	the spectrum of the signal ϕ
`numcyc`	the number of complete wavelengths in the signal time series, or the interval between peaks in the spectra
`numharm`	number of harmonics to include in the signal mode expansion
`modeldef`	a list of numbers giving the code of the model equation. An example is $\{\{3,0,1\},\{8,2,1\}\}$, which uses an equation in which the first term is of type 3 and orders $r = 0$ (irrelevant for term of type 3) and $s = 1$, and the second term is of type 8 and orders $r = 2$ and $s = 1$.
`depth`	number of parameter sets to compute. Larger numbers imply computing using higher harmonics.
`numer`	determines how parameters are solved for. Option "S" and "N" force the parameter solving function to use `Solve` and `NSolve`, respectively. Anything else, for example "A", implies letting the function automatically choose between the two methods given the model equation. The running time and accuracy of the results can depend on which option is selected.
`toquiet`	a boolean instructing whether to display error/warning messages generated during the parameter solving calcualtions. Setting `True` suppresses the warning messages. Setting `True` still give error messages if these occur in other parts of the input.

Output:

a list which contains the model equation, the model code representation (for reference), and the parameter values.

`AOComputeParametersAllModels[ser1, ser2, numcyc, numharm, depth, numer, toquiet]`

Purpose:

to batch compute the parameters associated with many model structures

Arguments:

`ser1`	spectrum of the driving signal ψ
`ser2`	spectrum of the signal φ
`numcyc`	number of complete cycles in the signal time-series, or spacing between peaks in their spectra
`numharm`	number of harmonics to use in signal expansion
`depth`	number of parameter sets to compute for each model structure
`numer`	determines how parameters are solved for. The options "N" and "S" forces use of `NSolve` and `Solve`, respectively. Other input, for example "A", allow the function to automatically select one of these methods based on the type of model considered.
hs`toquiet`	a boolean instructing whether to display error/warning messages generated during the parameter solving calculations. Setting `True` suppresses the warning messages.

Output:

a long list of data. Each item in the list carries information about one model structure and is formatted like described for `AOComputeParameters`.

Note:

This function uses a little bit of optimization. It is therefore slightly faster than calling `AOComputeParameters` for a pair of signal for each model individually

---

`AOComputeSpec[ser1]`

Purpose:

to compute the spectrum of a time series. Note: the function essentially uses the built-in `Fourier[···]` functionality in Mathematica. However, it also rescales some values outputed by that command. The other functions in the package assume all spectra have been computed with this function.

Arguments:

`ser1`	a list of real numbers representing the time series.

Output:

a list of complex numbers representing the spectrum of the input.

`AOComputeTimeDomainErrList[ser1, ser2, numcyc, numharm, modellist, tstep]`

Purpose:

to compute the error between a reconstructed signal and its original for a list of proposed model structures

Arguments:

`ser1`	the spectrum of the driving signal
`ser2`	the time-series of the original signal
`numcyc`	number of the cycles in the signals, or the spacing between peaks in the spectra
`numharm`	number of harmonics to use in signal reconstruction
`modellist`	a list of models for which to reconstruct the time-series and compute errors. The list should be of the format outputed by `AOInterpretParametersList`
`tstep`	a small real number used as a time-step in model reconstruction

Output:

A list of the same length as `modellist` containing the same information, plus the value of the computed error appended at the end of each model's entry.

`AOCorruptSeries[ser, level]`

Purpose:

to add normal noise to a time series

Arguments:

`ser`	a list representing a time series
`level`	a real number representing a noise level

Output:

a noisy time series of the same length as the original. The noise added is sampled from a normal distribution centered around the mean of the signal with standard deviation `level` times that of the mean. The outputed time series is positive semi-definite.

`AOCorruptSpec[ser, level]`

Purpose:

to add normal noise to a spectrum

Arguments:

`ser`	a list representing a spectrum
`level`	a real number representing a noise level

Output:

a noisy spectrum of the same length as the original. The noise is added to each component of the spectrum individually. For each item, it is sampled from a normal distribution centered around that items magnitude with standard deviation `level` times that of that magnitude.

`AODelaySpec[ser, tau]`

Purpose:

to produce a spectrum of a time-delayed signal

Arguments:

`ser`	a spectrum
`tau`	a number representing a time delay. It can also be a variable

Output:

a list of numbers representing the spectrum of a time-delayed signal. It is computed purely in the frequency-domain.

`AOGetEqList[]`

Purpose:

to check what types of model structures are currently set to be tested

Arguments:

none

Output:

a list of model structures. Each item in the list is a definition of a model structure in the form $\{\{t_1, r_1, s_1\}, \{t_2, r_2, s_2\}\}$. The numbers $t_{1,2}$ refer to the term code, all of which are listed in Table 1. The other numbers $r_{1,2}$ and $s_{1,2}$ denote the orders of each interaction.

`AOGetLevels[]`

Purpose:

to verify what criteria are currently set for parameter interpretation

Arguments:

none

Output:

a printed list of criteria

`AOGetRecreateSignalJump`

Purpose:

to retrieve the current value of the jump parameter used in signal reconstruction

Arguments:

none

Output:

an integer representing the currently set jump parameter

`AOInterpretParameters[prevres]`

Purpose:

to apply the parameter interpretation criteria determined with `AOSetLevels` to sets of parameter values.

Arguments:

`prevres`	a list or parameter sets. The first item should be a list, the first item of which is ignored (can be e.g. an integer representing the depth level) and the second item of which is a list of parameters. The second item is another such list, and so on. The format is that of one of the items outputed by `AOComputeParameters`.

Output:

A list of two items. The first item is a true or false value stating whether one of the inputed parameter sets is consistent according to the criteria. The second item is that consistent parameter set.

AOInterpretParametersList[prevres]

Purpose:

to interpret parameters for many models at once

Arguments:

prevres	the output from the function AOComputeParametersAllModels, or another similarly formatted data structure

Output:

a list of models consistent with the selection criteria. Each item of the list is another list containing information about one model. The first item in the inner list is the model equation. The second is the set of parameter values that is consistent with the selection criteria. The third and fourth are the model code and the list of all parameter sets.

AOInterpretTimeDomainErrList[prevres, thresh]

Purpose:

to interpret and invalidate models based on the time domain reconstruction criteria

Arguments:

prevres	a list of data in the format outputed from AOComputeTimeDomainErrList
thresh	a threshold value. Models with errors above the threshold are rejected. Good thresholds depend on the signal, e.g. the mean value of the signal

Output:

A shorter list of the same format as prevres containing only those model structures passing the criteria.

AOModelSubSample[ser, pos1, pos2, ssz]

Purpose:

to produce a short time series from a long time series

Arguments:

ser	a long time series
pos1	the index of the starting point
pos2	the index of the ending point
ssz	a small number

Output:

a list representing a time series. The new time series is subsampled from ser one in the interval (pos1, pos2) and is approximately ssz times shorter than ser.

`AOPlotModel[ser, is, title]`

Purpose:

to generate a plot of a time series with some standardized settings

Arguments:

`ser`	if this is a list of numbers, it is used as the signal to plot. If it is a list of two lists, two signals are plotted, the first in purple and the other in gray.
`is`	the image size of the plot
`title`	a string used for the plot title

Output:

a graphics object

`AOPlotModelSpec[spec, speclen, is, title]`

Purpose:

to produce a plot of a signal spectrum

Arguments:

`spec`	a spectrum to be plotted
`speclen`	the number of modes to show in the plot
`is`	the image size
`title`	a string to be used for the plot title

Output:

a graphics object of a spectrum. Coefficients for cosine components appear as circles. Coefficients for sine components appear as squares.

`AOPlotModelSpecAbs[spec, speclen, is, tolog, title]`

Purpose:

to produce a plot of the aboslute value of a signal spectrum

Arguments:

`spec`	If this is one list of numbers, it is used as the spectrum to be plotted. If it is a list of two lists, each of these is used as a spectrum. The first is plotted in purple, the other in gray.
`speclen`	the number of modes to show in the plot
`is`	the image size
`tolog`	a boolean determining whether to show data on a logarithmic scale
`title`	a string to be used for the plot title

Output:

A graphics object of the spectrum.

`AORecreateSignal[ser1, ic, numcyc, numharm, modeldef, params, tstep]`

Purpose:

to build a signal φ using an initial condition, model equation, and a driving signal

Arguments:

`ser1`	the spectrum of the driving signal
`ic`	the initial condition for the signal to be rebuilt
`numcyc`	number of complete wavelengths in the driving signal, or the spacing between peak heights in its spectrum
`numharm`	number of harmonics of the driving signal to use in reconstruction
`modeldef`	a list of numbers representing the model equation
`params`	a list of parameter values to substitute into the model equation
`tstep`	a real number to use as the time step in the rebuilding process. If this is set to zero, the signal recreation is done using Mathematica's `NDSolve` function. Otherwise, a manual solver is used. See note below about how this parameter affects performance.

Output:

a list of numbers representing a signal time series

Note:

The running time of `AORecreateSignal` depends on the length of the signal and on the desired accuracy of the reconstruction. The latter can be controlled in a number of ways.

If the parameter `tstep` is set to zero, the reconstruction occurs using Mathematica's NDSolve. This is normally the fastest and most accurate method.

If the parameter `tstep` is set above zero, the reconstruction occurs using a manual ODE solver. The performance of this depends on `tstep`, with smaller values yielding better but slower results. Speed can also be regulated through a "jump" parameter using `AOSetRecreateSignalJump`. During the manual reconstruction, it is necessary to compute the value of the driving signal at many time points, which requires calling computationally expensive sine and cosine functions. Setting the "jump" parameter to values greater than one allows this function to approximate sections of the driving signal using piecewise linear segments. Setting the jump parameter to values larger than one often results in speedup withtout undue loss of accuracy.

`AOSetLevels[dep, ranges, tole, reimtole]`

Purpose:

to set options and tolerance levels used in parameter interpretation

Arguments:

`depth`	an integer determining how many parameter sets to use
`ranges`	a list of allowed ranges for the parameters. Each element of the list should be another list of type {`parametername, minvalue, maxvalue`}
`tole`	a list of tolerances for each depth level. The first item of the list is ignored. The second item determines by how much parameters computed at level 2 can differ from those at level 1 and still be acceptable. Tolerances of 0.5, 1, 2 mean they can differ by 50%, 100%, 200%, etc. Tolerances are set independently for each depth level.
`reimtole`	a list of tolerances to imaginary parameter values for each depth level

Output:

none.

`AOSetEqList[newlist]`

Purpose:

to custom set a list of model structures to be tested

Arguments:

`newlist`	a list of model structures. Each item in the list should be a code for a model equations. The format should be the same as the output from `AOGetEqList` after the list is generated automatically using `AOBuildEqList`

Output:

none. The new equation list can be retrieved using `AOGetEqList`

`AOSetRecreateSignalJump[jump]`

Purpose:

to set the jump parameter used in signal reconstruction

Arguments:

`jump`	an integer greater or equal to 1. The role of the parameter is discussed under `AORecreateSignal`

Output:

none. The jump parameter can be retrieved using `AOGetRecreateSignalJump`

`AOShowAnsatz[ser1, numcyc, numharm]`

Purpose:

to construct a time series of the ansatz used in parameter reconstruction.

Arguments:

`ser1`	a signal spectrum
`numcyc`	number of cycles in signal time-series, or spacing between peaks in the spectrum
`numharm`	number of harmonics to include in the signal ansatz

Output:

a list of numbers representing an approximated signal time series

Note:

Neither this function or the list of numbers it outputs is used in the parameter computations. It does, however, give a time-domain representation of what information from the signal is being used in parameter identification.

`AOShowTerms[pp1not, pp2not, ordnot, signalX, signalXD]`

Purpose:

to list the types of terms and their codes

Arguments:

`pp1not`	a variable name to be used as parameter 1
`pp2not`	a variable name to be used as parameter 2
`ordnot`	a variable name to be used as an order
`signalX`	a variable name to be used as the signal X[t]
`signalXD`	a variable name to be used as the driving signal XD[t]

Output:

a list containing term codes and their expressions

`AOSuggestDepth[ser1, numcyc, noisetype, thr]`

Purpose:

to estimate how many parameter sets should be computed from the signal

Arguments:

`ser1`	a spectrum
`numcyc`	the number of complete cycles in the signal time series. Equivalently, the spacing between peaks in the spectrum
`noisetype`	the type of noise profile to test against. This should be set to "normal" since other noise types are not implemented.
`thr`	a threshold value

Output:

an integer greater or equal to zero. The output plus one represents the number of peaks whose heights are larger than the threshold times the mean noise level.

Examples

Two notebooks are provided with examples.

`ModelReconstructionExample.nb` is an example of how to use the package for model reconstruction. The calculations in this notebook load two data signals, and aim to determine the equation structures.

`ModelReconstructionTricks.nb` contains examples of some tricks that can be done with the package functions. This includes automating testing for only a few hand-picked model structures, and testing of model structures where a signal has an explicit time delay.