GPoM : 4 Visualization of the outputs

Sylvain Mangiarotti & Mireille Huc

2020-02-18

Model vizualization

The GPoM package enables to explore large numbers of models in polynomial ODEs form in order to identify which model can reproduce the observed behaviour. The present vignette aims to explain how to retrieve the information obtained from the `gPoMo` function.

In practice, the `gPoMo` function will generate a list of variables hereafter called `outputGPoM`:

```
data("Ross76")
# time vector
tin <- Ross76[seq(1, 3000, by = 8), 1]
# single time series
data <- Ross76[seq(1, 3000, by = 8), 3]
# global modelling
# results are put in list outputGPoM
outputGPoM <- gPoMo(data, tin=tin, dMax = 2, nS=c(3), show = 0, method = 'rk4',
                    nPmin = 3, nPmax = 12, IstepMin = 400, IstepMax = 401)
```

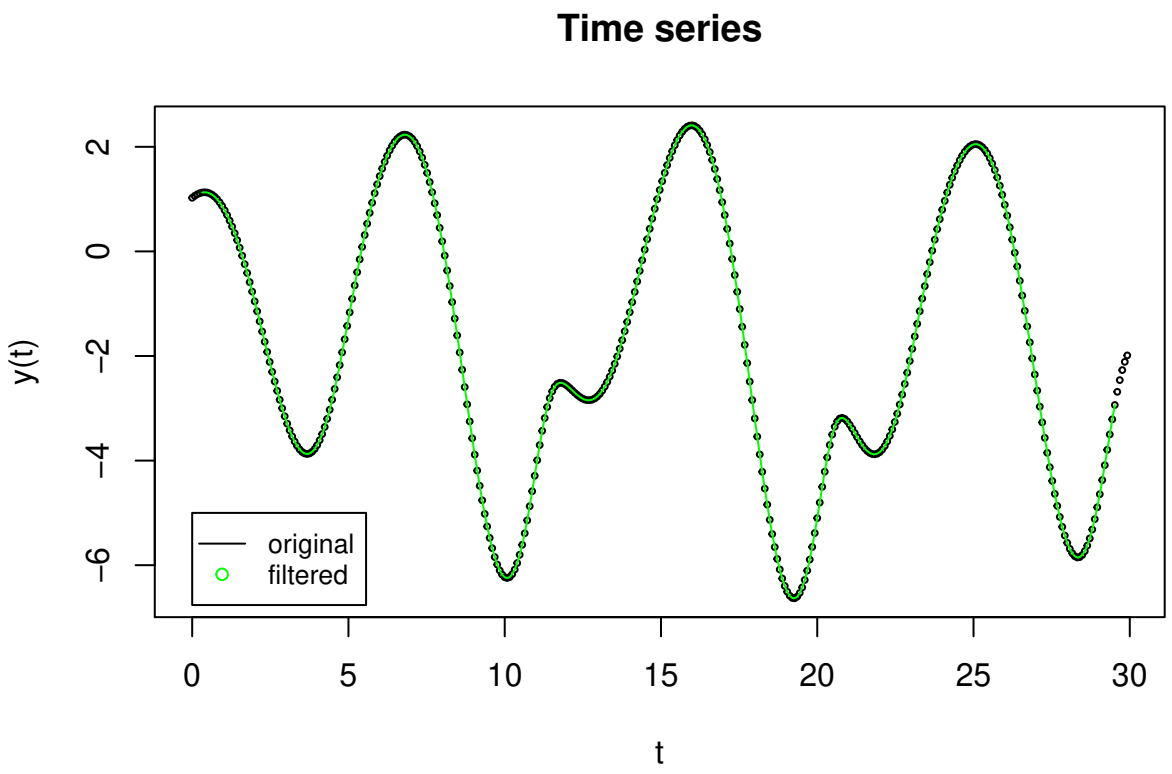
```
## ### For Istep = 400 (max: 401), models to test: 10 / 10
## ### Number of unclassified models: 3 / 10
```

To test the potential of a given model to reproduce the observed dynamics, the numerical integrability of each model is tested and the resulting behaviour may either diverge, converge to a trivial attractor (fixed points, periodic cycles) or to a chaotic attractor.

The aim of the present vignette is to show how to have a rapid overview of the models explored or obtained with the `gPoMo` function.

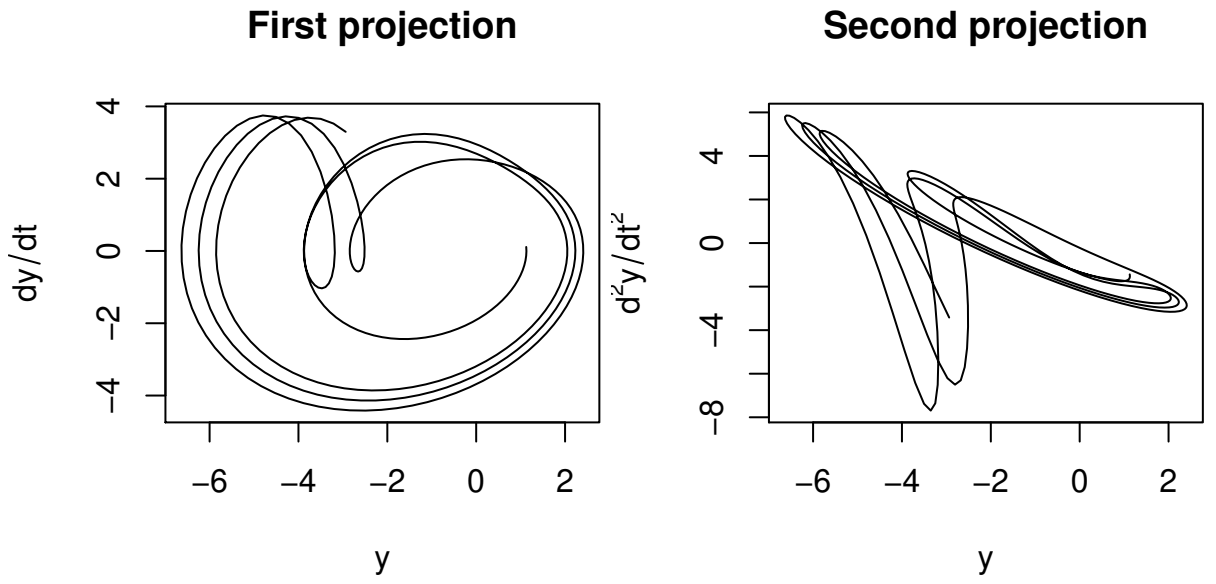
In order to keep a memory of the analyzed signal, the data used as an input are kept in `$tin` (for the input time) and `$inputdata` (for the input time series). The filtered version obtained when computing the derivatives is put in `$tfilt` (the vector time of the filtered time series which length is lower due to some lost at the boundaries) and in `$filtdata`, a matrix with the filtered time series (first column) and its successive derivatives (second and next columns).

```
# plot data and models time series
plot(outputGPoM$tin, outputGPoM$inputdata,
     xlab='t', ylab='y(t)', main = 'Time series', cex=0.4)
lines(outputGPoM$tfiltdata, outputGPoM$filtdata[,1], type='l', col='green')
legend(0,-5, c("original", "filtered"), col=c('black', 'green'),
       lty=c(1,NA), pch=c(NA,1), cex = 0.8)
```



The original differential phase portrait can thus be plotted as follows:

```
# plot data and models time series
plot(outputGPoM$filtdata[,1], outputGPoM$filtdata[,2], xlab='y',
     ylab= expression(dy/dt), main = 'First projection', type = 'l', cex = 0.4)
plot(outputGPoM$filtdata[,1], outputGPoM$filtdata[,3], xlab='y',
     ylab= expression(d^2*y/dt^2), main = 'Second projection', type = 'l', cex = 0.4)
```



Information about the tested models are kept in the other variables: `$okMod`, `$stockoutreg` and `$models`.

Variable `$okMod` provides information about the obtained numerical integration of all the tested models: values are chosen equal to zero when the integrated trajectory diverged during the numerical integration, equal to 1 when the model could not be categorized (meaning that the model is still under categorizing test).

Since we are mainly interested in non-diverging models, these are expected to be detected in the `$okMod` parameter such as: `$okMod != 0`. Note that diverging models may be identified as non-diverging if the numerical integration was not performed on a sufficiently long duration. The number of `$okMod` identified as non-diverging (based on the chosen maximum integration steps `IStepMax`) is obtained by:

```
# How many models could not be identified as non-diverging
sum(outputGPoM$okMod)
```

```
## [1] 3
```

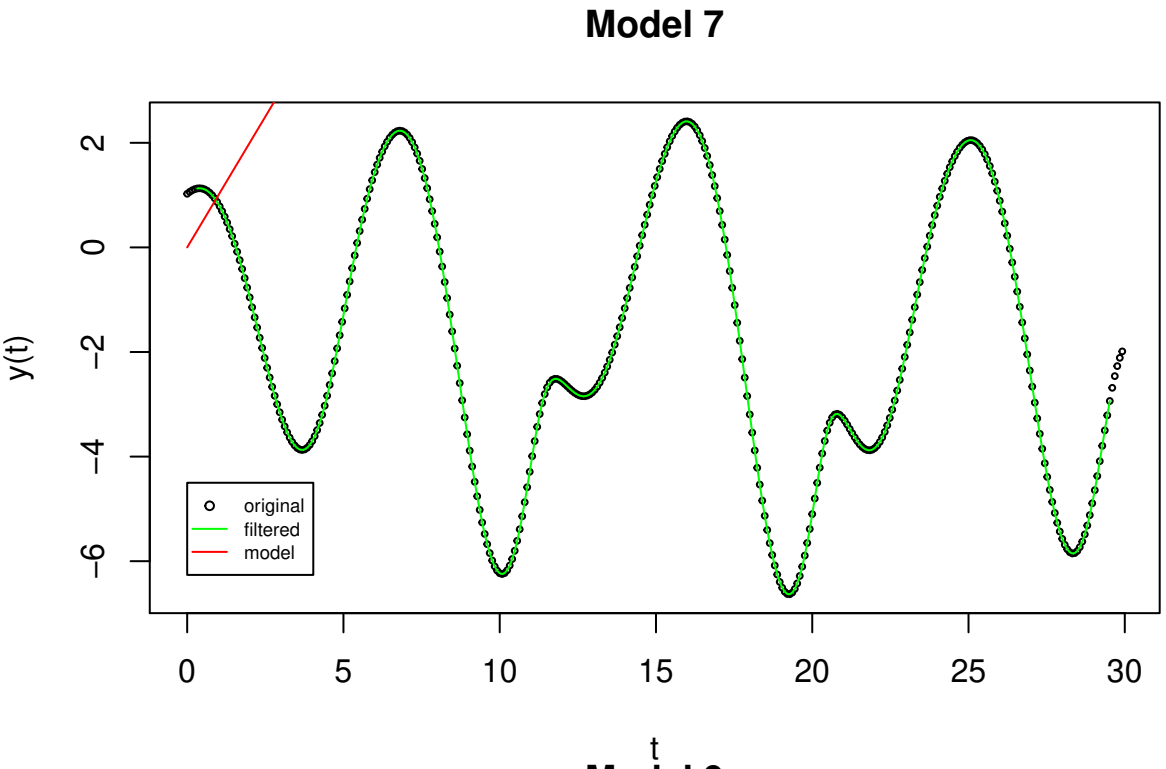
The reference numbers of the uncategorized models can be retrieved as follows:

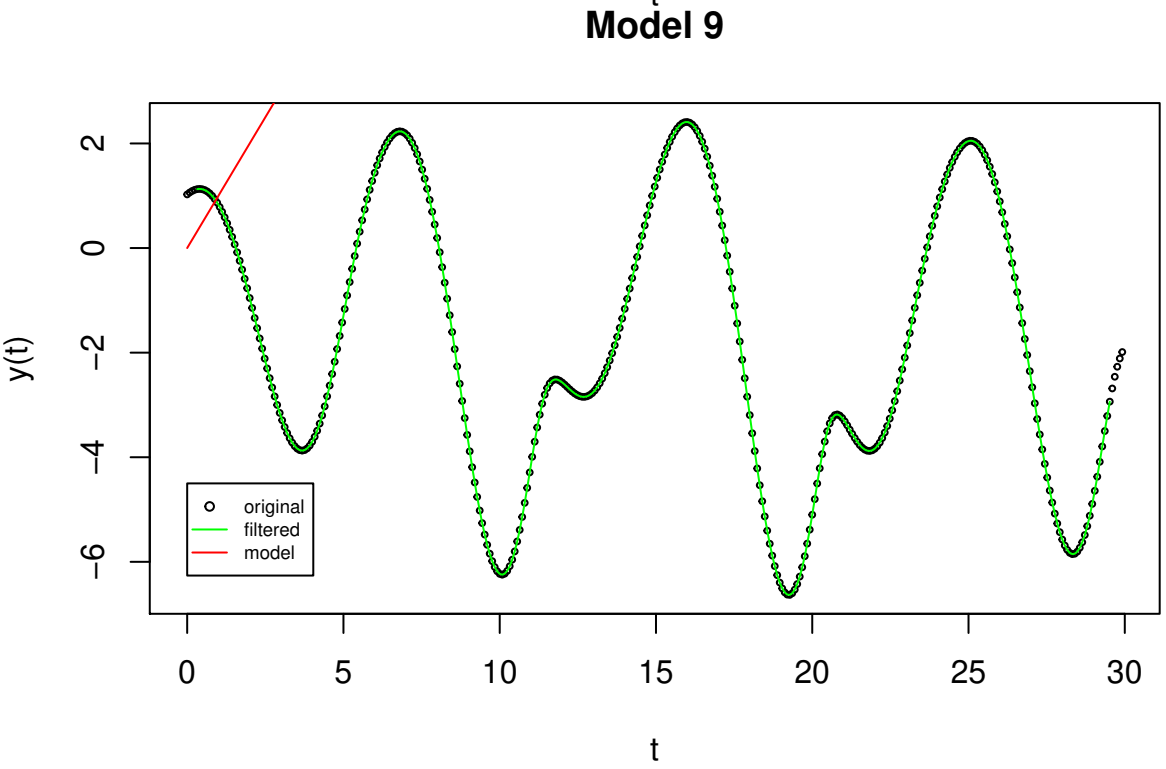
```
# what is the reference number of these models?
which(outputGPoM$okMod == 1)
```

```
## [1]  7  9 10
```

The numerical integration of the tested models is kept in variable `$stockoutreg`. For example, it can be plotted for models #7 and #9:

```
# plot data and models time series
plot(outputGPoM$tin, outputGPoM$inputdata,
     xlab='t', ylab='y(t)', cex = 0.4, main = 'Model 7')
lines(outputGPoM$tfiltdata, outputGPoM$filtdata[,1], type='l', col='green')
lines(outputGPoM$tout, outputGPoM$stockoutreg$model7[,1], type='l', col='red')
legend(0,-4.5, c("original", "filtered", "model"), col=c('black', 'green', 'red'),
       lty=c(NA,1,1), pch=c(1, NA, NA), cex = 0.6)
#
plot(outputGPoM$tin, outputGPoM$inputdata,
     xlab='t', ylab='y(t)', cex = 0.4, main = 'Model 9')
lines(outputGPoM$tfiltdata, outputGPoM$filtdata[,1], type='l', col='green')
lines(outputGPoM$tout, outputGPoM$stockoutreg$model9[,1], type='l', col='red')
legend(0,-4.5, c("original", "filtered", "model"), col=c('black', 'green', 'red'),
       lty=c(NA,1,1), pch=c(1, NA, NA), cex = 0.6)
```





In this example, only one iteration of the model test was performed (since `IstepMin` $*2 <$ `IstepMax`). As a consequence, the same initial conditions are used for both filtered data and model reconstruction:

```
head(outputGPoM$filtdata[1:3], 1)
```

```
## [1] 1.126608
```

```
head(outputGPoM$stockoutreg$model7, 1)
```

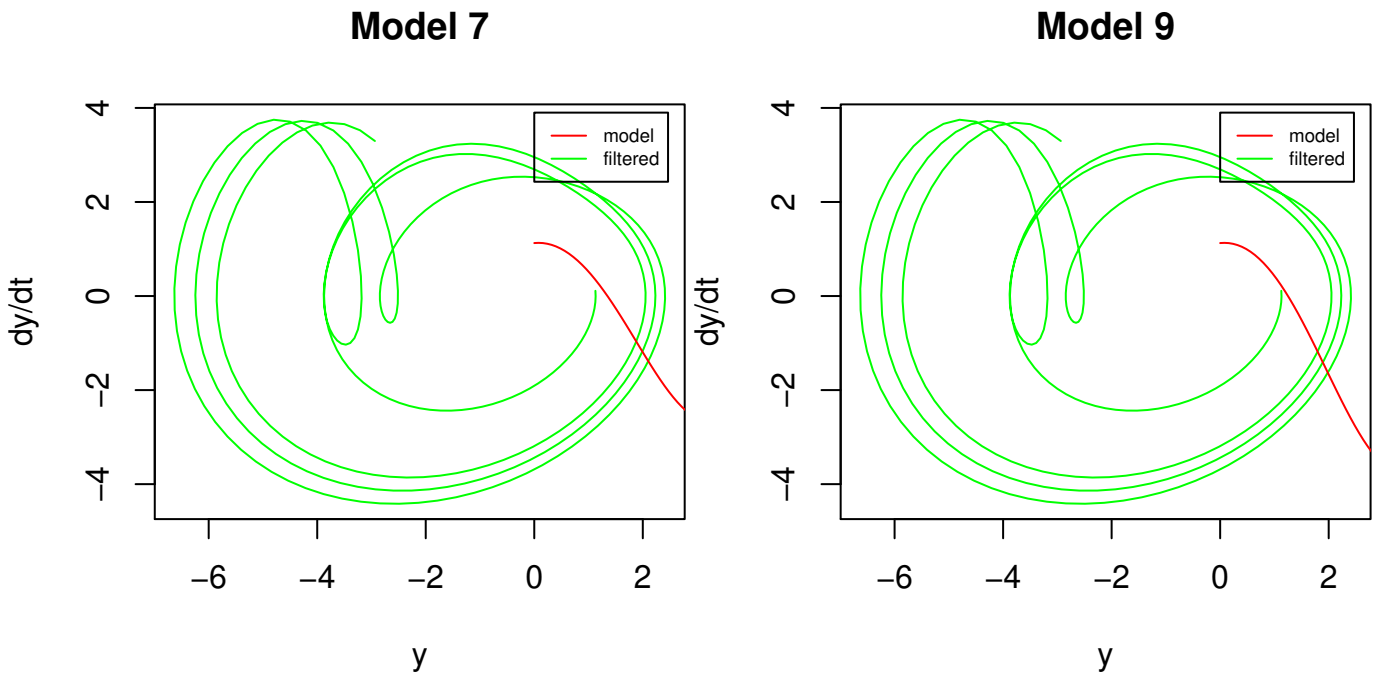
```
##      time        1         2         3
## [1,]    0 1.126608 0.1110817 -1.438434
```

(note that `$filtdata` has one more derivative which was necessary to identify the model parameters).

Original and models phase portraits can also be plotted for comparison as follows:

```
# plot data and models phase portraits
plot(outputGPoM$filtdata[,1], outputGPoM$filtdata[,2], type='l', col='green',
     xlab='y', ylab='dy/dt', main = 'Model 7')
lines(outputGPoM$stockoutreg$model7[,1], outputGPoM$stockoutreg$model7[,2],
      type='l', col='red')
legend(0,3.9,c("model", "filtered"), col=c('red', 'green'), lty=c(1,1), cex=0.6)

#
plot(outputGPoM$filtdata[,1], outputGPoM$filtdata[,2], type='l', col='green',
     xlab='y', ylab='dy/dt', main = 'Model 9')
lines(outputGPoM$stockoutreg$model9[,1], outputGPoM$stockoutreg$model9[,2],
      type='l', col='red')
legend(0,3.9,c("model", "filtered"), col=c('red', 'green'), lty=c(1,1), cex=0.6)
```



Finally, models coefficients are all kept in the sublist `$models`. These are separated in `$models$mToTest#` where `#` should be replaced by the model number (the list of the models that were tested with the `gPoMo` algorithm) and `$models$model#` (the list of uncategorized models). In both cases, models coefficients are in a matrix of dimension `pMax * nVar` with `pMax` the maximal number of parameter (see vignette `1 Conventions`) and `nVar` the number of variables (also called model dimension). The corresponding equations can be edited directly using the `visuEq` function as:

```
nVar <- dim(outputGPoM$models$model7)[2]
pMax <- dim(outputGPoM$models$model7)[1]
dMax <- p2dMax(nVar,pMaxKnown = pMax)
# See equations
```

```
visuEq(outputGPoM$models$model7, approx = 2)
```

```
## dX1/dt = 1 X2
##
## dX2/dt = 1 X3
##
## dX3/dt = -2.086 X3  + 0.647 X2 X3  -0.522 X2^2  -2.747 X1  -0.29 X1 X3  + 0.848 X1 X2  -0.375 X1^2
```

```
#
visuEq(outputGPoM$models$model9, approx = TRUE)
```

```
## dX1/dt = 1 X2
##
## dX2/dt = 1 X3
##
## dX3/dt = -1.6  -3.1 X3  + 0.8 X2  + 0.9 X2 X3  -0.5 X2^2  -3.6 X1  -0.5 X1 X3  + 1.1 X1 X2  -0.5 X1^2
```

By default, a large number of digits is provided for each parameter. To make its reading easier, this number can be controlled by using the optional parameter `approx = TRUE`, the minimum number of digit will be kept (the same for all the parameters) but keeping all the terms in the polynomial. The number of digits to add to the minimum number of digits required can also be chosen manually.

The number of terms of each model can be obtained as follows:

```
# for model #7
sum(outputGPoM$models$model7 != 0)
```

```
## [1] 9
```

```
# for model #9
sum(outputGPoM$models$model9 != 0)
```

```
## [1] 11
```

Since these models have a canonical structure, it may also be interesting to know the number of terms in the last equation (which polynomial defines the model), such as:

```
# for model #7
colSums(outputGPoM$models$model7 != 0)
```

```
## mToTest
##       1       1       7
```

```
# for model #9
colSums(outputGPoM$models$model9 != 0)
```

```
## mToTest
##       1       1       9
```

Finally, to have an overview of the overall outputs of `gPoMo` at a glance for all the models, the following command can be used:

```
######################
# automatic ##########
######################
visuOutGP(outputGPoM)
```

Next step

An overview of all the main elements of the global modelling technique and GPoM tools were presented in the previous vignettes. An illustration of model validation will now be presented in the last vignette `5 Predictability` based on the model forecasting performances.