GPoM : 3 Modelling

Sylvain Mangiarotti & Mireille Huc

2018-07-26

Global Modelling

This document gives an illustration on how to apply the global modelling technique in various contexts. The aim of the `GPoM` package is to obtain models of Ordinary Differential Equations (ODEs) of polynomial form from time series (either single or multiple). Such type of modelling requires a careful time series preprocessing. Such data preprocessing was sketched in the vignette `2 PreProcessing`. In the present vignette, it is assumed that the studied time series has already been carefully preprocessed, so that the modelling tools can now be applied without anymore preprocessing.

Single time series modelling

The chaotic system introduced by Rössler in 1976[1] is well adapted to test and illustrate the performances of the approach to model nonlinear dynamics. Simulations of this chaotic system are available in the GPoM package and can be directly loaded as follows:

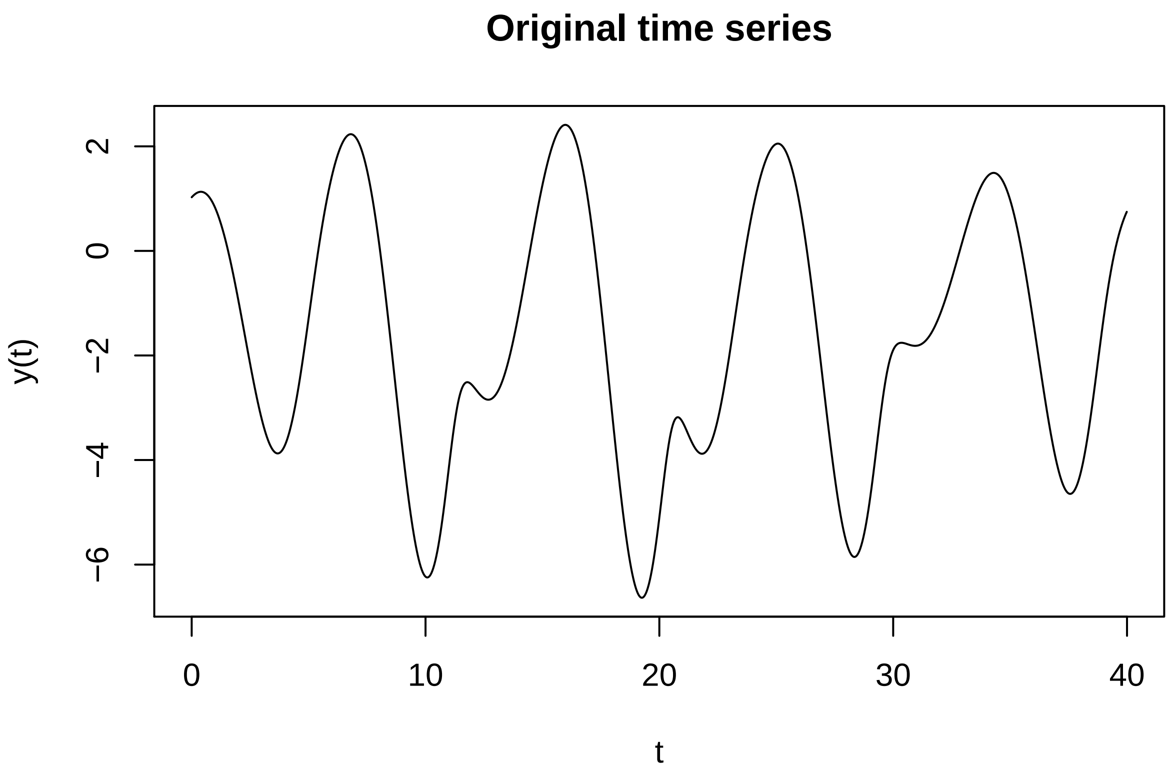
```
# load data
data("Ross76")
```

(note that the GPoM package can easily be used to produce such chaotic data set, see vignette `1 Conventions`).

The loaded variable `Ross76` is a matrix. The time vector is in the first column and the time series $x(t)$, $y(t)$ and $z(t)$ of the three variables are in the next columns two to four. The global modelling technique from one single time series is introduced in the present section. The variable y will be considered here:

```
# time vector
tin <- Ross76[,1]
# single time series
data <- Ross76[,3]
# plot
plot(tin, data, type='l', xlab = 't', ylab = 'y(t)', main = 'Original time series')
```

[1]O. Rössler, An Equation for Continuous Chaos, *Physics Letters*, **57A**(5), 1976, 397-398.



Starting from single time series, the following canonical form of equations can be expected[2]:

$dX_1/dt = X_2$

$dX_2/dt = X_3$

...

$dX_n/dt = P(X_1, X_2, ..., X_n)$

where the modeled variable X_1 will correspond to the observed variable y, and X_2 to X_n are its successive derivatives dy/dt and dy^2/dt^2. To apply the global modelling technique to the time series presented above, it is necessary to choose a trial model dimension `nVar` (corresponding to n in the previous equation) and a maximal polynomial degree `dMax`. To restrict the model search to a smaller ensemble of models, the minimum and maximum numbers of parameters for the model can be chosen manually using `nPmin` and `nPmax`. `IstepMax` corresponds to the maximum number of iterations to be used for the numerical integration.

```
# model dimension
nVar = 3
# maximum polynomial degree
dMax = 2
```

```
# generalized Polynomial modelling
out1 <- gPoMo(data, tin = tin, dMax = dMax, nS = nVar, show = 0,
              nPmin = 9, nPmax = 11, verbose = FALSE, IstepMax = 3000)
```

Note that, to follow the search process in real time, parameter `show` should be set on (`show = 1`). In practice, the model size (that is, its number of parameters which is bounded by `nPmin` and `nPmax`) is generally unknwown. It may thus be necessary to extend its range (be default `nPmin = 1` and `nPmax = 14`). However, in practice, `nPmax` cannot be chosen too large in order to keep the computation time reasonable.

With the parameterization chosen here, two models are obtained:

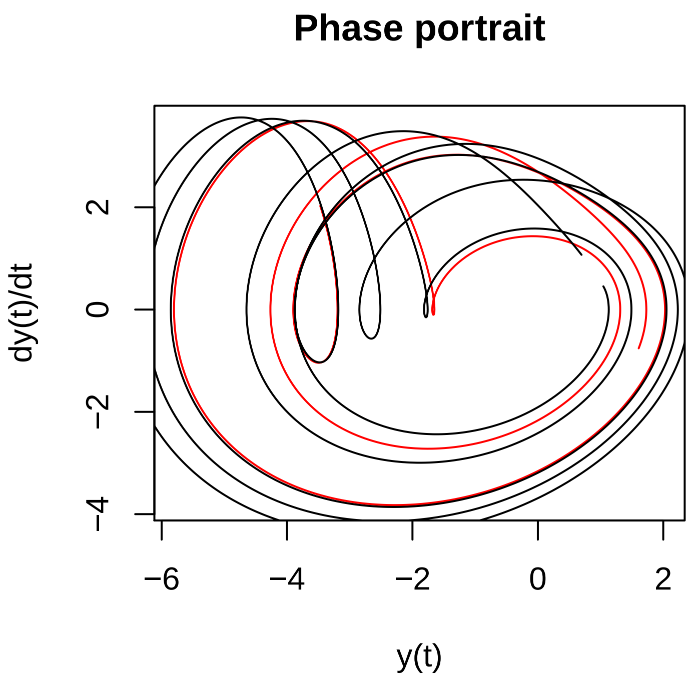
[2]G. Gouesbet & C. Letellier, 1994. Global vector-field reconstruction by using a multivariate polynomial L2 approximation on nets. *Physical Review E*, **49**(6), 4955-4972.

```
which(out1$okMod == 1)
```

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

The model #3 shows a quite good performance to reproduce the original phase portrait:

```
# obtained model #3
plot(out1$stockoutreg$model3[,1], out1$stockoutreg$model3[,2], type='l', col = 'red',
     xlab = 'y(t)', ylab = 'dy(t)/dt', main = 'Phase portrait')
# original phase portrait
lines(out1$filtdata[,1], out1$filtdata[,2])
legend(3,2,c("model", "original"), col=c('red', 'black'), lty=c(1,1), cex=0.6)
```



The equations of the obtained model can be edited as follows (more precision in the digits can be obtained choosing a larger value of `approx`):

```
# obtained model #3
visuEq(nVar, dMax, out1$models$model3, approx = 1)
```

```
## dX1/dt = X2
##
## dX2/dt = X3
##
## dX3/dt = -2  -3.5 X3  + 1.1 X2  + X2 X3  -0.5 X2^2  -4 X1  -0.5 X1 X3  + 1.3 X1 X2  -0.5 X1^2
```

Various methods may be used to validate such a model. In practice, note that validation may depend on the objective of the model (characterization, forecasting, etc.). One way to validate the model can be done by considering the forecasting performances of the model. A code dedicated to such a type of validation is presented in vignette `5 Predictability`.

Detection of causal couplings and retro-modelling

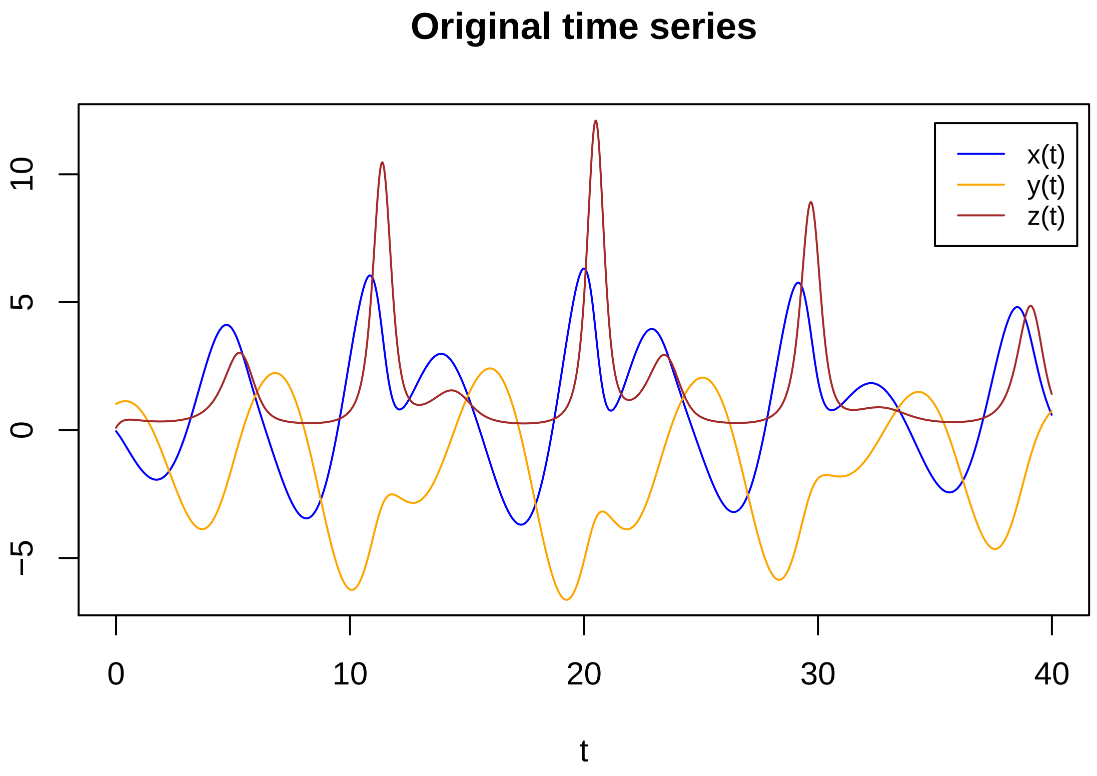
One interest of the global modelling technique is (1) to enable the detection of nonlinear dynamical couplings between observed variables, and potentially even (2) to obtain an analytical formulation of this couplings in order (3) to help for the inference of the causal relations.

Here again, the Rössler-1976 chaotic system is used to illustrate the purpose. The time series of the three variables x, y and z are used this time.

```
# multiple (three) time series
data <- Ross76[,2:4]
```

The three time series are as follows:

```
# x(t)
plot(tin, data[,1], ylim = c(-6.5,12), type='l', col = 'blue',
     xlab = 't', ylab = '', main = 'Original time series')
# y(t)
lines(tin, data[,2], type = 'l', col = 'orange')
# z(t)
lines(tin, data[,3], type = 'l', col = 'brown')
legend(35,12,c("x(t)", "y(t)", "z(t)"), col=c('blue', 'orange', 'brown'), lty=1, cex=0.8)
```



A set of three equations is expected, one for each variable:

$dx/dt = P_x(x, y, z)$

$dy/dt = P_y(x, y, z)$

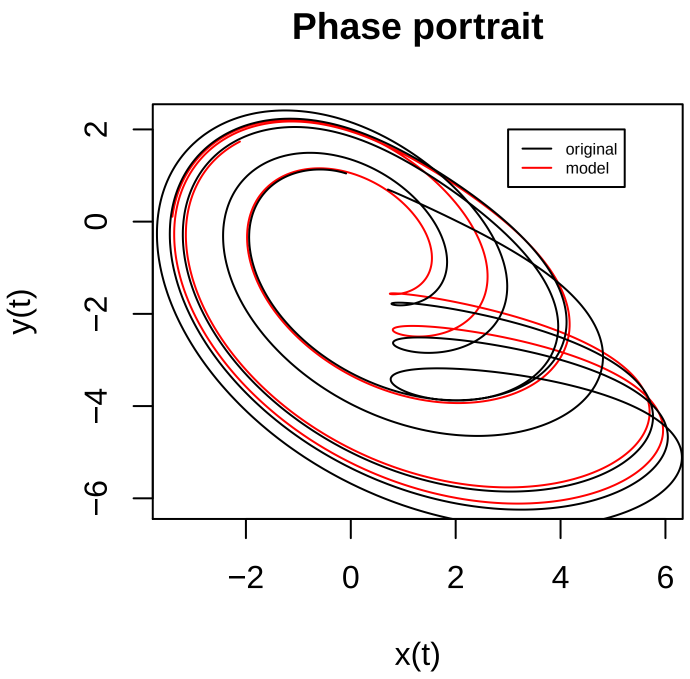
$dz/dt = P_z(x, y, z)$

To define such a structure, the number `nS` of equations for each series must be defined as a vector such as: `nS = c(1,1,1)`. (note that to discard information about the current number of models under test, the optional parameter `verbose` can be set to `verbose = FALSE`).

```
# generalized Polynomial modelling
out2 <- gPoMo(data, tin = tin, dMax = 2, nS = c(1,1,1),  show = 0,
              IstepMin = 10, IstepMax = 3000, nPmin = 7, nPmax = 8)
```

The simplest obtained model able to reproduce the observed dynamics is the model #5:

```r
# obtained model #5
plot(out2$stockoutreg$model5[,1], out2$stockoutreg$model5[,2],
     type='l', col = 'red',
     xlab = 'x(t)', ylab = 'y(t)', main = 'Phase portrait')
# original phase portrait
lines(out2$filtdata[,1], out2$filtdata[,2])
legend(3,2,c("original", "model"), col=c('black', 'red'), lty=1, cex=0.5)
```



The equations of the obtained model are given by:

```r
visuEq(3, 2, out2$models$model5, approx = 4)
```

```
## dX1/dt = -0.99924 X3  -0.9995 X2
##
## dX2/dt = 0.51985 X2  + 0.99971 X1
##
## dX3/dt = 1.99568  -3.99102 X3  + 0.99775 X1 X3
```

The original Rössler system is thus retrieved. The ability of the approach to retrieve original equations was also tested to numerous other dynamical systems (see vignette `7 Retro-modelling`).

A practical application of the global modelling technique under real world conditions was done for modelling the plague epidemic of Bombay by the end of 19th century and its coupling to the epizootics among black and brown rats[3]. This analysis enabled not only the detection of the couplings but also to obtain an algebraic formulation of this coupling. An interpretation could be proposed for all the terms of the obtained model.

Generalized global modelling and polynomial a priori structure

The GPoM package enables a generalized formulation of the global modelling technique combining the canonical formulation – one single observed variables and its derivatives – and the multivariate formulation

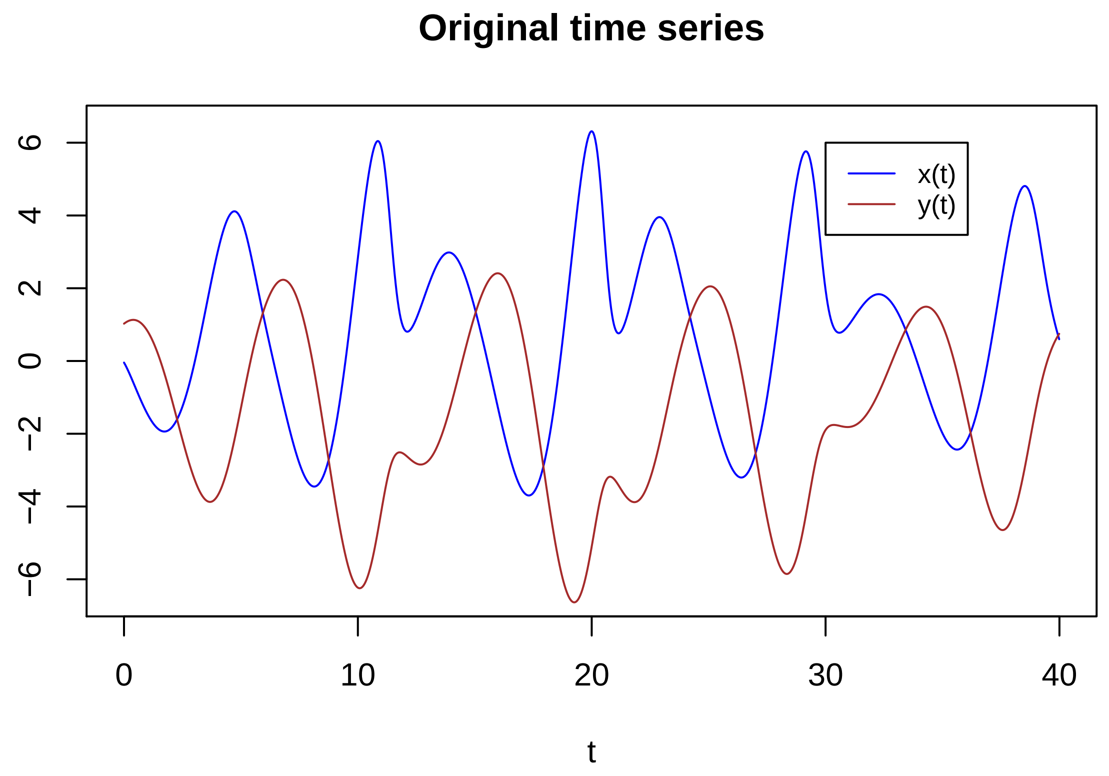
[3]S. Mangiarotti, 2015. Low dimensional chaotic models for the plague epidemic in Bombay (1896–1911). *Chaos, Solitons & Fractals*, **81A**, 184–196.

– several observed variables considered together. Examples are given in the two previous sections. The two variables x and y of the Rössler-1976 chaotic system are considered here to illustrate the purpose.

```r
# time vector
tin <- Ross76[,1]
# multiple (two) time series
data <- Ross76[,2:3]
```

The two time series are as follows:

```r
# x(t)
plot(tin, data[,1], ylim = c(-6.5,6.5),
     type='l', col = 'blue',
     xlab = 't', ylab = '', main = 'Original time series')
# y(t)
lines(tin, data[,2], type = 'l', col = 'brown')
legend(30,6,c("x(t)", "y(t)"), col=c('blue', 'brown'), lty=1, cex=0.8)
```



Obviously, the correlation between the two time series is not very high since close to 0.5 in magnitude (Note that alternative examples with very low levels of correlations will also be considered in vignette 7 `Retro-modelling`):

```r
# correlation between Rössler-x and Rössler-y
cor(data[,1], data[,2])
```

```
## [1] -0.502108
```

Though, their dynamics are linked in a causal way since dynamically coupled as expressed by the original (fully deterministic) system:

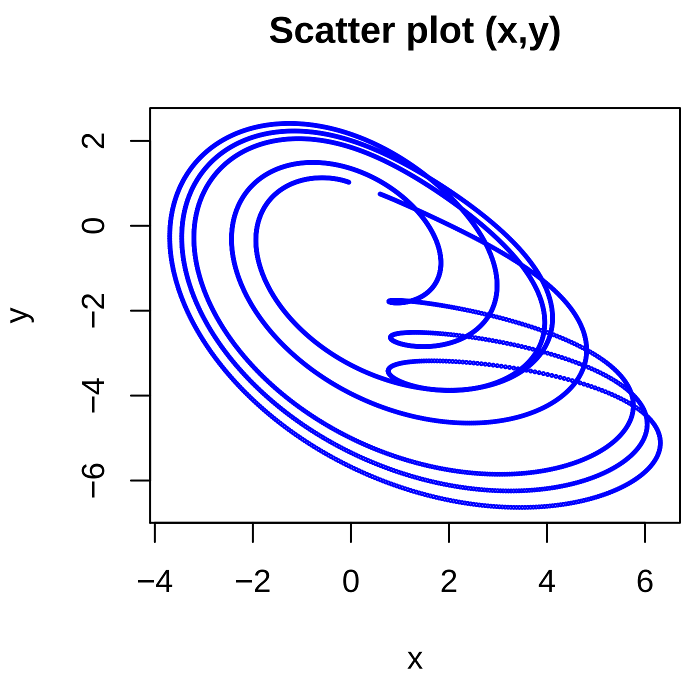
$dx/dt = -y - z$

$dy/dt = x + ay$

$dz/dt = b + z(x - c)$.

Because x and y are coupled to the third variable z (assumed not to be observed in the present example), and because of the presence of a nonlinear term (xz in the last equation), the relation between x and y can only be poorly detected using a simple statistical technique such as the correlation. This difficulty directly results from the complex coevolution between these two variables as illustrated by the following scatter plot (x,y):

```r
plot(data[,1], data[,2],
     type='p', cex = 0.2, col = 'blue',
     xlab = 'x', ylab = 'y', main = 'Scatter plot (x,y)')
```



In its principle, the global modelling technique is well adapted to tackle the detection of nonlinear couplings.

As the Rössler-1976 system is three-dimensional, a three-dimensional system can be expected, that is, such as `sum(nS) = 3` (in practice, this dimension is generally unknown and a trial dimension can be used). Two formulations may thus be expected, either based on variables (X_1, X_2, Y_1):

$dX_1/dt = X_2$

$dX_2/dt = P_X(X_1, X_2, Y_1)$

$dY_1/dt = P_Y(X_1, X_2, Y_1)$,

which general structure will be defined by `nS = c(2,1)` (two equations of canonical form will be generated from the observed time series $x(t)$, a single one from $y(t)$);

or based on variables (X_1, Y_1, Y_2):

$dX_1/dt = P_X(X_1, Y_1, Y_2)$

$dY_1/dt = Y_2$

$dY_2/dt = P_Y(X_1, Y_1, Y_2)$

which general structure will be defined by `nS = c(1,2)` (one equation generated from $x(t)$, two ones of canonical form from $y(t)$).

For a tutorial purpose, we will assume here that some *a priori* information is available about the model sructure telling us that the former formulation based on variables X_1, X_2 and Y_1 is more suited and that specific polynomial terms should be excluded (e.g. Y_1^2 and X_2Y_1 should be excluded from polynomial P_X, and only the terms Y_1, X_1 and X_1Y_1 should be accepted in polynomial P_Y). It is then possible to provide a template for the model formulation allowing some terms and forbidding others. This template structure can be provided as an input of the `gPoMo` function through the optional parameter `EqS`. In the present case, the following structure will be used as a template:

```
# model template:
EqS <- matrix(1, ncol = 3, nrow = 10)
EqS[,1] <- c(0,0,0,1,0,0,0,0,0,0)
EqS[,2] <- c(1,1,0,1,0,1,1,1,1,1)
EqS[,3] <- c(0,1,0,0,0,0,1,1,0,0)
visuEq(3, 2, EqS, substit = c('X1','X2','Y1'))
```

```
## dX1/dt = X2
##
## dX2/dt = 1  + Y1  + X2  + X2^2  + X1  + X1 Y1  + X1 X2  + X1^2
##
## dY1/dt = Y1  + X1  + X1 Y1
```

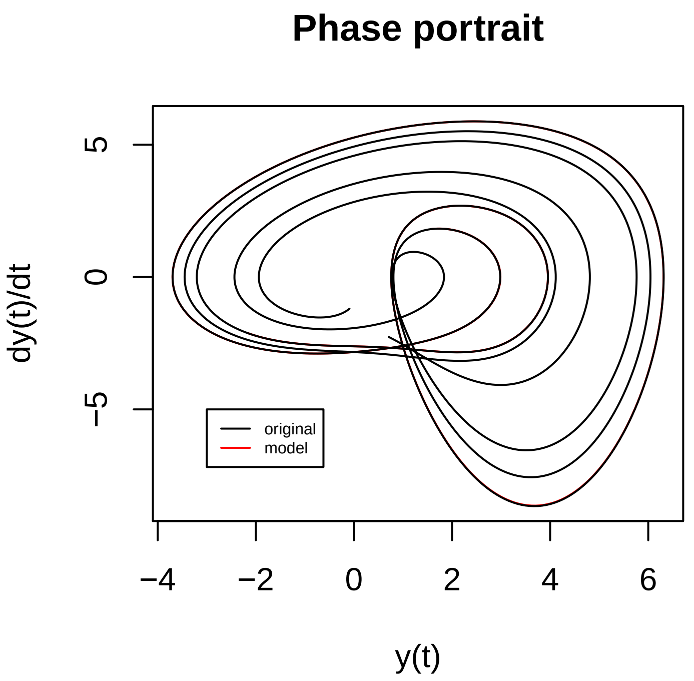
Each term of this template can be used by the global model if set to 1 and cannot if set to 0. In other words, terms that are not included in this template cannot be added, but terms that are included may be removed.

The search for a Generalized Global Model will be launched as follows:

```
# generalized Polynomial modelling
out3 <- gPoMo(data, tin = tin, dMax = 2, nS=c(2,1), EqS = EqS,
              show = 0, verbose = FALSE,
              IstepMin = 10, IstepMax = 2000, nPmin = 9, nPmax = 11)
```

The simplest obtained model able to reproduce the observed dynamics is found to be the model #2:

```
# obtained model #2
plot(out3$stockoutreg$model2[,1], out3$stockoutreg$model2[,2],
     type='l', col = 'red',
     xlab = 'X1', ylab = 'X2', main = 'Phase portrait')
# original phase portrait
lines(out3$filtdata[,1], out3$filtdata[,2])
legend(-3,-5,c("original", "model"), col=c('black', 'red'), lty=1, cex=0.5)
```



The equations of the obtained model are the following :

```r
visuEq(3, 2, out3$models$model2, approx = 4, substit = c('X1','X2','Y1'))
```

```
## dX1/dt = X2
##
## dX2/dt = -1.99472  -4.51179 Y1  -3.99202 X2  -0.99974 X1  + 0.99828 X1 Y1  + 0.99783 X1 X2
##
## dY1/dt = 0.51985 Y1  + 0.99971 X1
```

When no information is available about the polynomial structure, the model search can be launched without using the option `EqS`. In this situation, all the terms are considered as possible terms (all the terms are set to 1 in `EqS`). When there is no reason to choose one formulation rather than the other, then all the formulations may be tested one by one. In the present case, models can actually be obtained with both sets of variables (X_1,Y_1,Y_2) and (X_1,X_2,Y_1)).

A practical application of the generalized global modelling technique could be performed from observational data under real world conditions for the modelling of the West-Africa epidemic of Ebola Virus Desease[4].

Blind separation and modelling of two independant sets of equations

Finally, since the approach enables – in its principle – to detect nonlinear couplings, it may also be used to detect separated sets of equations. Two separated systems are considered here: the Rössler-1976 system already presented upper and the Sprott-K system[5].

```r
# load data
data("TSallMod_nVar3_dMax2")
# multiple (six) time series
tin <- TSallMod_nVar3_dMax2$SprK$reconstr[1:400,1]
TSRo76 <- TSallMod_nVar3_dMax2$R76$reconstr[,2:4]
```

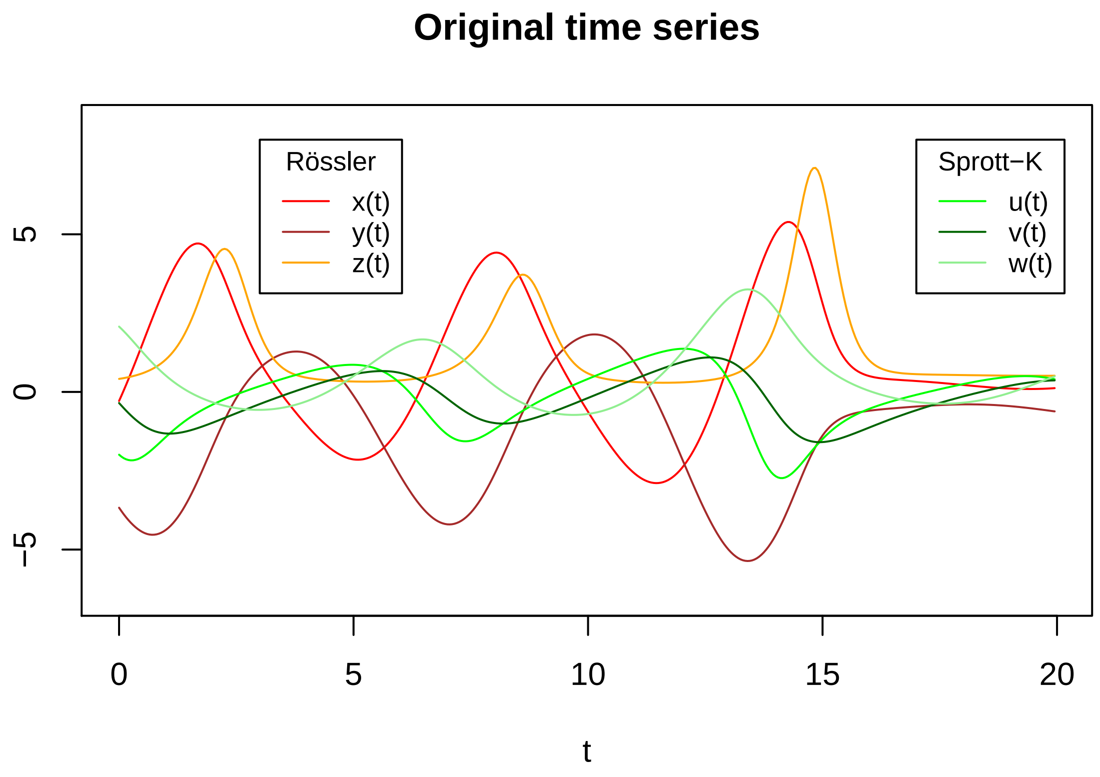
[4] S. Mangiarotti, M. Peyre & M. Huc, 2016. A chaotic model for the epidemic of Ebola virus disease in West Africa (2013–2016). *Chaos*, **26**, 113112.

[5] J.C. Sprott, 1994. Some simple chaotic flows. *Physical Review E*, **50**(2), 647-650

```
TSSprK <- TSallMod_nVar3_dMax2$SprK$reconstr[,2:4]
data <- cbind(TSRo76,TSSprK)[1:400,]
```

Six time series are thus considered in this illustration, three for each system:

```
# For the Rössler-1976 system
# x(t)
plot(tin, data[,1], ylim = c(-6.5,8.5), type='l', col = 'red',
     xlab = 't', ylab = '', main = 'Original time series')
# y(t)
lines(tin, data[,2], type = 'l', col = 'brown')
# z(t)
lines(tin, data[,3], type = 'l', col = 'orange')
# For the Sprott-K system
# u(t)
lines(tin, data[,4], type = 'l', col = 'green')
# v(t)
lines(tin, data[,5], type = 'l', col = 'darkgreen')
# w(t)
lines(tin, data[,6], type = 'l', col = 'lightgreen')
legend(3, 8,title = 'Rössler', c("x(t)", "y(t)", "z(t)"),
       col=c('red', 'brown', 'orange'), lty=1, cex=0.8)
legend(17, 8,title='Sprott-K', c("u(t)", "v(t)", "w(t)"),
       col=c('green', 'darkgreen', 'lightgreen'), lty=1, cex=0.8)
```



Since one equation is expected from each variable, vector `nS` is defined as `nS = c(1,1,1,1,1,1)`. To speed up the computation, the fourth order Runge-Kutta numerical integration `method = 'rk4'` is preferred[6].

[6]package `deSolve` is used for this purpose.

```r
# generalized Polynomial modelling
out4 <- gPoMo(data, dtFixe = 1/20, dMax = 2, nS = c(1,1,1,1,1,1),
              show = 0, method = 'rk4',
              IstepMin = 2, IstepMax = 3,
              nPmin = 13, nPmax = 13)
```

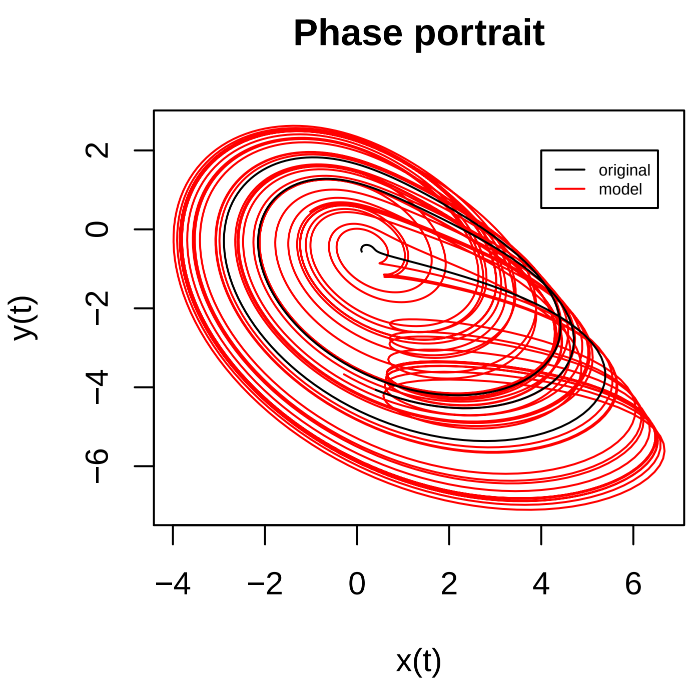
Due to the huge number of models to be tested, numerical integration is sketched here on a quite short duration and the model search is directly focused on models of 13-parameter size. A larger range of model sizes and a longer numerical integration can be applied to check the ability of the technique to retrieve the original models. This can easily be done, except that a much longer running time will be required.

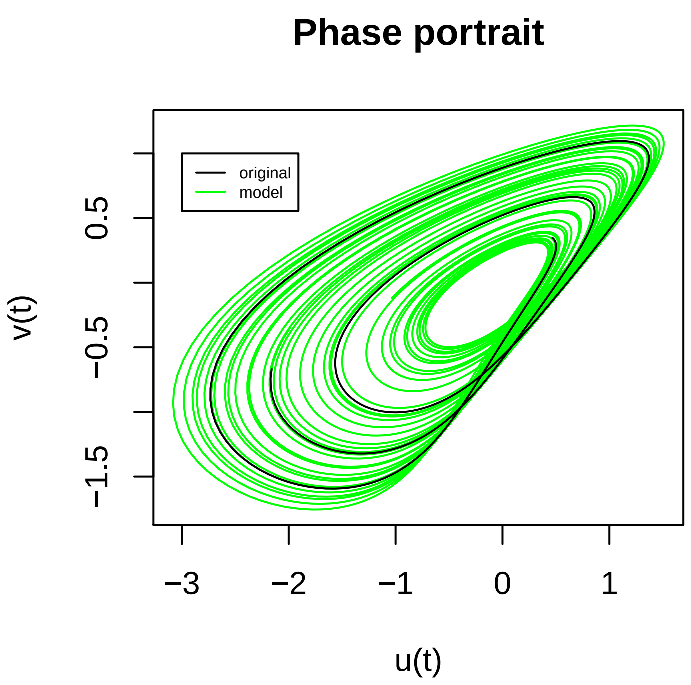
However, to distinguish between integrable and non integrable models (in a numerical sense), it is necessary to consider the numerical integration of large lengths. With longer integration durations (e.g. from `IstepMin = 10` to `IstepMax = 10000`), many of the models will be rejected or lead to trivial solutions (fixed points, periodic cycles)

In the present case, only one obtained set of equations (model #347) able to reproduce the original phase portraits of the two original systems (Rössler-1976 and Sprott-K) is found:

```r
KL <- out4$models$model347
v0 <- as.numeric(head(data,1))
outNumi <- numicano(nVar = 6, dMax = 2, Istep=5000, onestep=1/20, KL=KL, v0=v0)
# obtained model #347
plot(outNumi$reconstr[,2], outNumi$reconstr[,3],
     type='l', col = 'red',
     xlab = 'x(t)', ylab = 'y(t)', main = 'Phase portrait')
# original phase portrait
lines(out4$filtdata[,1], out4$filtdata[,2])
legend(4,2,c("original", "model"), col=c('black', 'red'), lty=1, cex=0.5)

# original phase portrait
plot(outNumi$reconstr[,5], outNumi$reconstr[,6],
     type='l', col = 'green',
     xlab = 'u(t)', ylab = 'v(t)', main = 'Phase portrait')
# original phase portrait
lines(out4$filtdata[,4], out4$filtdata[,5])
legend(-3,1,c("original", "model"), col=c('black', 'green'), lty=1, cex=0.5)
```





The estimated model reads

```
visuEq(6, 2, out4$models$model347, approx = 2,
       substit = c("x", "y", "z", "u", "v", "w"))
```

```
## dx/dt = -0.988 z  -0.992 y
##
## dy/dt = 0.517 y  + 0.994 x
##
## dz/dt = 1.931  -3.862 z  + 0.966 x z
##
## du/dt = -0.99 w  + 0.982 u v
##
## dv/dt = -0.992 v  + 0.993 u
##
## dw/dt = 0.298 w  + 0.994 u
```

The global modelling technique enables to distinguish two independant sets of equations: one for the Rössler-1976 system (equations one to three), the other one for the Sprott-K system (equations four to six). Moreover, the obtained equations are identical in their structure and very close in their parameterization to the original equations from which the observed time series were derived. This latter point is interesting since it shows that – in the present case at least – not only the couplings, but also each of the nonlinear terms of the equations can be retrieved. The approach may thus be used as a retro-modelling technique in order to interpret the model terms. This problem will be investigated in more details in vignette `7 Retromodelling`.

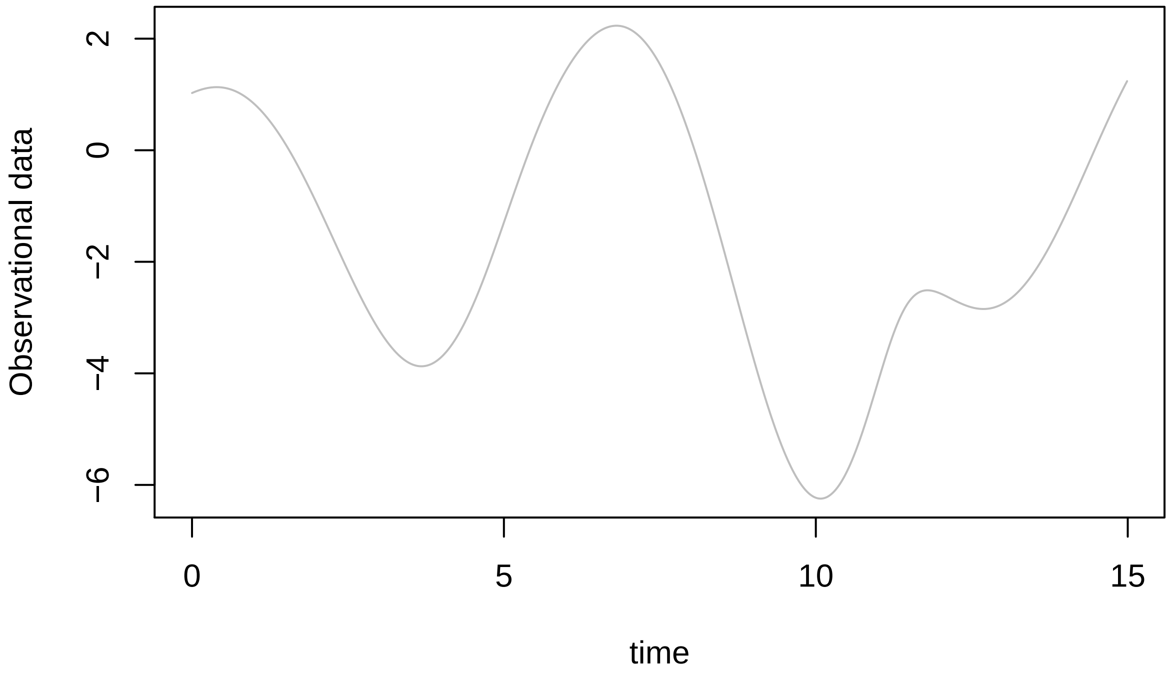
Time series with gaps

When measurements are gathered under real environmental conditions, the time series may have gaps, or may be contaminated by temporary perturbations which should be removed. The `GPoM` package can be applied to such types of data sets[7].

[7]S. Mangiarotti, F. Le Jean, M. Huc, C. Letellier, 2016. Global modelling of aggregated and associated chaotic dynamics. *Chaos, Solitons & Fractals*, **83**, 82-96

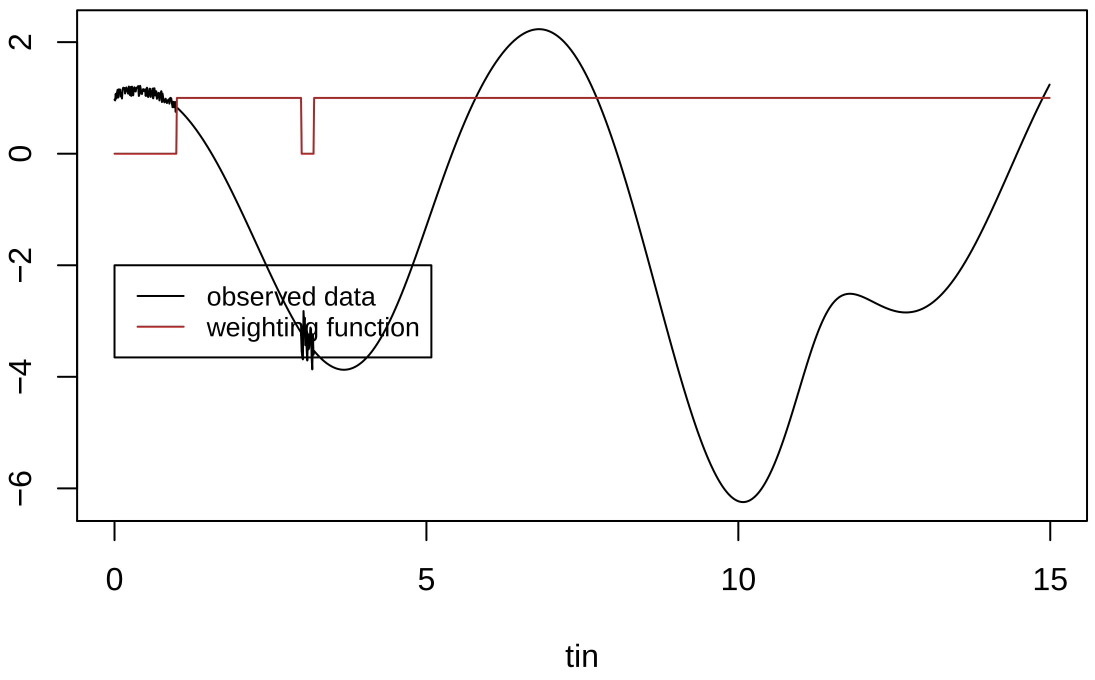
An example of this kind of situation is given in the present section. The Rössler system is used again in this example:

```
# load data
data("Ross76")
# time vector
tin <- Ross76[1:1500,1]
# single time series
series0 <- series <- Ross76[1:1500,3]
# plot
plot(tin, series0, type = 'l', col = 'gray', xlab = 'time', ylab = 'Observational data')
```



To illustrate our purpose, some noise is added to the original time series and a weighting function is defined to indicate on what part of the signal the analysis should focus. This function is equal to 1 when the time series is free of noise, and equal to 0 when it is noise-contaminated.

```
# some noise is added
series[1:100] <- series[1:100] + 0.1 * runif(1:100, min = -1, max = 1)
series[301:320] <- series[301:320] + 0.5 * runif(1:20, min = -1, max = 1)
plot(tin, series, ylab='', type = 'l', col = 'black')
#
# weighting function
W <- tin * 0 + 1
W[1:100] <- 0  # the first fourty values will be discarded
W[301:320] <- 0  # twenty other values will be discarded too
lines(tin, W, type = 'l', col = 'brown')
legend(0, -2,c("observed data", "weighting function"),
       col=c('black', 'brown'), lty=1, cex=0.8)
```



The global modelling is then applied:

```
# Weighted time series
GPout1 <- gPoMo(data = series, tin = tin, weight = W,
                dMax = 2, nS = 3, winL = 9, show = 1,
                IstepMin = 10, IstepMax = 6000,
                nPmin = 5, nPmax = 11, method = 'rk4')
```

```
## ### For Istep = 10 (max: 6000), models to test: 7 / 7
## ### For Istep = 20 (max: 6000), models to test: 7 / 7. Runtime: ~ 0h 0min 0.18s
## ### For Istep = 40 (max: 6000), models to test: 7 / 7. Runtime: ~ 0h 0min 0.19s
## ### For Istep = 80 (max: 6000), models to test: 7 / 7. Runtime: ~ 0h 0min 0.38s
## ### For Istep = 160 (max: 6000), models to test: 7 / 7. Runtime: ~ 0h 0min 0.74s
## ### For Istep = 320 (max: 6000), models to test: 7 / 7. Runtime: ~ 0h 0min 1.47s
## ### For Istep = 640 (max: 6000), models to test: 6 / 7. Runtime: ~ 0h 0min 2.6s
## ### For Istep = 1280 (max: 6000), models to test: 6 / 7. Runtime: ~ 0h 0min 4.93s
## ### For Istep = 2560 (max: 6000), models to test: 4 / 7. Runtime: ~ 0h 0min 6.6s
## ### For Istep = 5120 (max: 6000), models to test: 3 / 7. Runtime: ~ 0h 0min 9.94s
## ### Number of unclassified models: 3 / 7
```

The following equations are retrieved (the option `weight` indicates when the observations provides in `series` should be considered in the analysis, or not):

```
visuEq(3, 2, GPout1$models$model7, approx = 2)
```

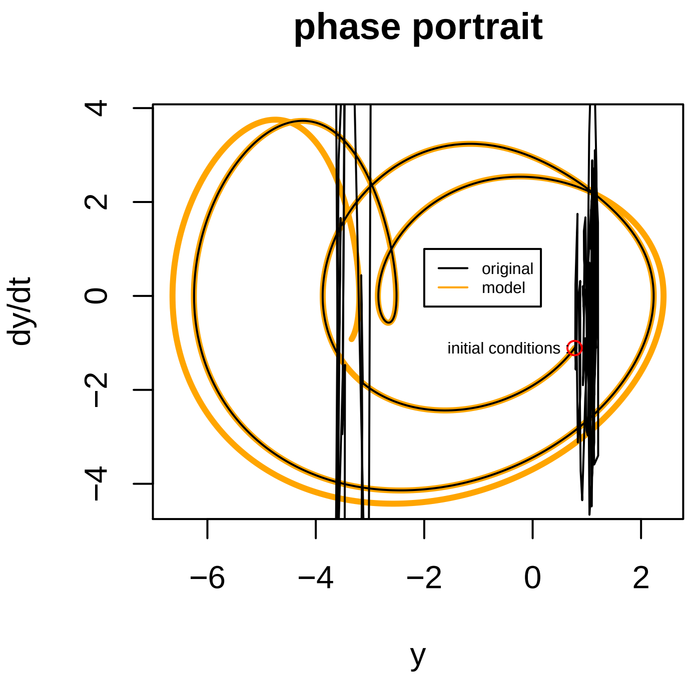
```
## dX1/dt = X2
##
## dX2/dt = X3
##
## dX3/dt = -1.993 -3.473 X3 + 1.076 X2 + 0.998 X2 X3 -0.519 X2^2 -3.993 X1 -0.519 X1 X3 + 1.268 X1 X2 -0.
```

This set of equations can then be integrated numerically. Integration will be launched from the first state for which weight is equal to 1.

```r
# first weight which value not equal to zero:
i1 = which(GPout1$Wfiltdata== 1)[1]
v0 <-  GPout1$filtdata[i1,1:3]
rcstr <- numicano(nVar=3, dMax=2, Istep=5000, onestep=1/250,
                  KL = GPout1$models$model7,
                  v0=v0, method="rk4")
```

The best obtained model is #7 that efectively reproduces the dynamic of the original system:

```r
plot(rcstr$reconstr[,2], rcstr$reconstr[,3], type='l', lwd = 3,
     main='phase portrait', xlab='y', ylab = 'dy/dt', col='orange')
# original data:
lines(GPout1$filtdata[,1], GPout1$filtdata[,2], type='l',
      main='phase portrait', col='black')
# initial condition
lines(v0[1], v0[2], type = 'p', col = 'red')
legend(-2,1,c("original", "model"), col=c('black', 'orange'), lty=1, cex=0.5)
legend(-2.1,-0.7,"initial conditions : ", cex = 0.5,  bty="n")
```



Note that model #3 also produces a good approximation of the original system.

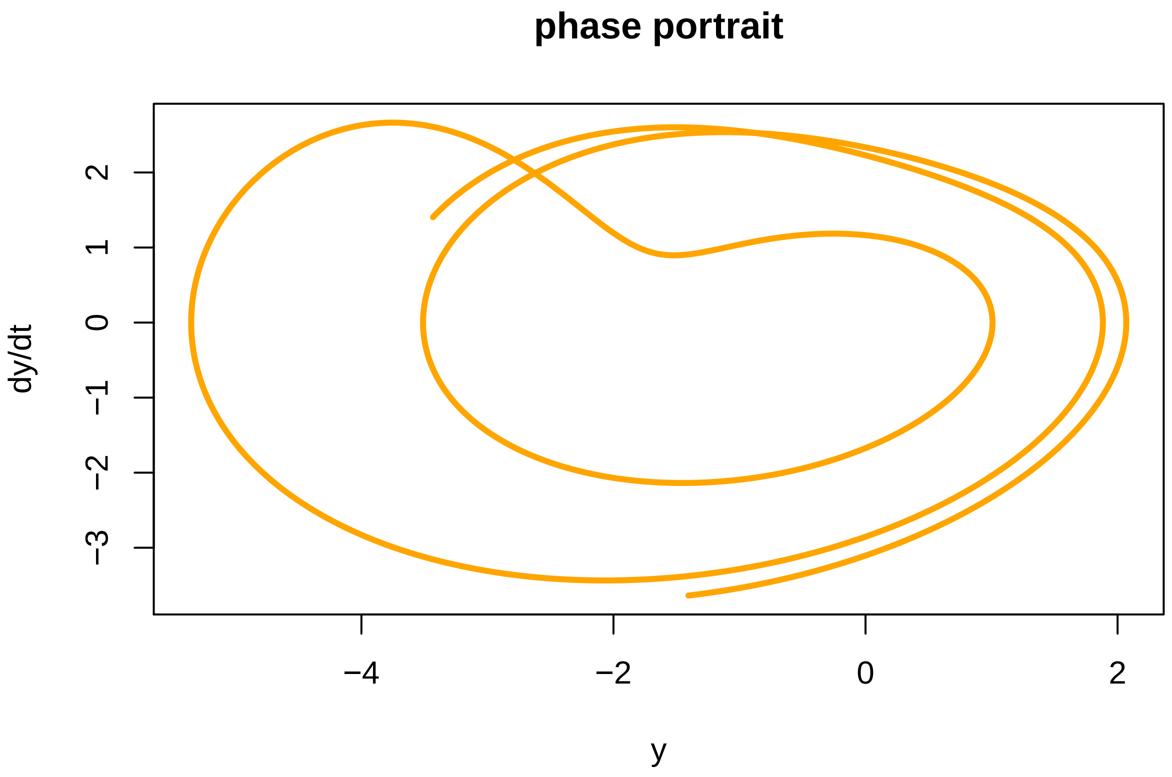
```r
visuEq(3, 2, GPout1$models$model3, approx = 2)
```

```
## dX1/dt = X2
##
## dX2/dt = X3
##
## dX3/dt = -1.587 X3  + 0.784 X2 X3  -0.52 X2^2  -1.612 X1  + 0.911 X1 X2
```

This model converges to a periodic attractor but with a slight modification of its parameterization, a chaotic solutions can be obtained.

```r
# first weight which value not equal to zero:
i1 = which(GPout1$Wfiltdata== 1)[1]
v0 <-  GPout1$filtdata[i1,1:3]
```

```r
KL3bis <- KL3 <- GPout1$models$model7
KL3bis[2,3] <- KL3[2,3] * 1.1
rcstr <- numicano(nVar = 3, dMax = 2,
                  Istep = 40000, onestep = 1/250, KL = KL3bis,
                  v0 = v0, method = "rk4")
plot(rcstr$reconstr[35000:40000,2], rcstr$reconstr[35000:40000,3],
     main='phase portrait', xlab='y', ylab = 'dy/dt',
     type='l', lwd = 3, col='orange')
```



Time series in associassion

Equivalent records of the same dynamical behavior may have been performed by several sensors, concurently or not and at same or different locations. It may then be useful, and simetimes necessary, to consider these time series simultaneously in the modelling process. The `GPoM` package can also be applied to such type of data sets[8].

A set of four time series of different time legnth, all derived from the Rössler system, are made available to show how to apply the global modelling technique to such a set of time series.

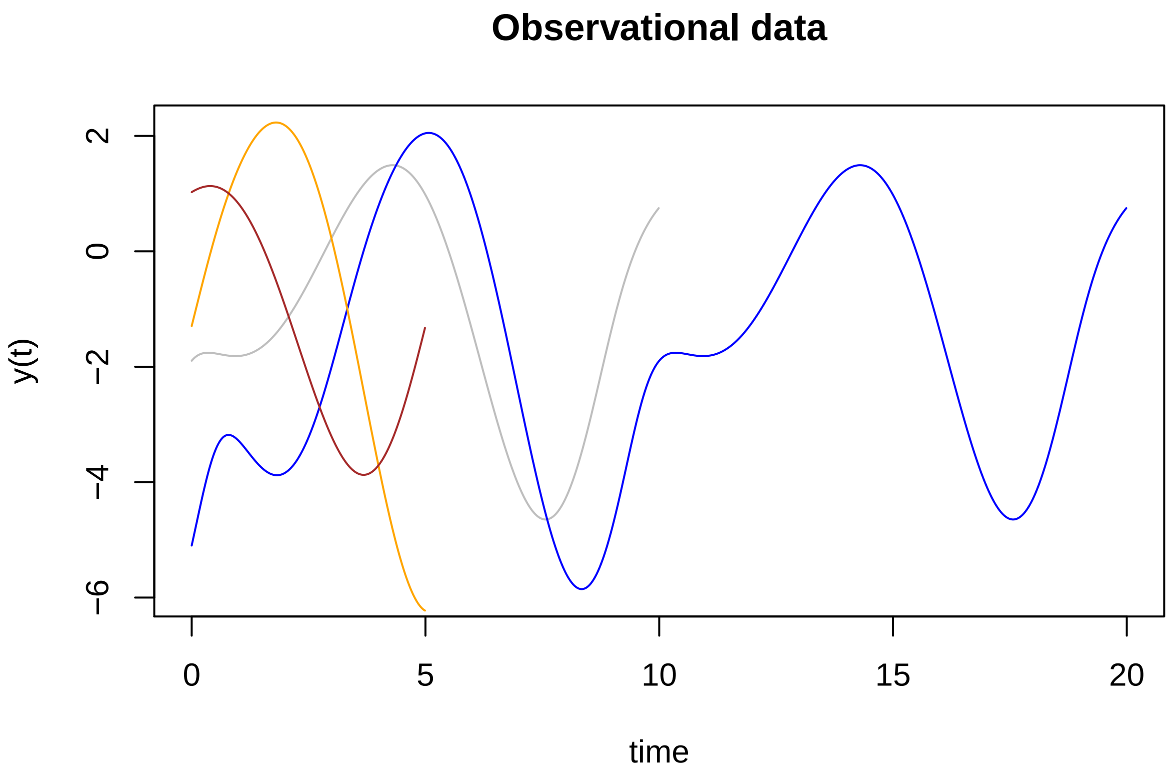
```r
# load data
data("severalTS")
```

```
## Warning in data("severalTS"): data set 'severalTS' not found
```

```r
# plot
plot(svrlTS$data1$TS[,1] - svrlTS$data1$TS[1,1], svrlTS$data1$TS[,2],
     type = 'l', col = 'gray',
     xlab = 'time', ylab = 'y(t)', main = 'Observational data',
```

[8]Ibid

```r
     xlim = c(0, 20), ylim = c(-6, 2.2))
lines(svrlTS$data2$TS[,1] - svrlTS$data2$TS[1,1], svrlTS$data2$TS[,2],
      type = 'l', col = 'blue')
lines(svrlTS$data3$TS[,1] - svrlTS$data3$TS[1,1], svrlTS$data3$TS[,2],
      type = 'l', col = 'orange')
lines(svrlTS$data4$TS[,1] - svrlTS$data4$TS[1,1], svrlTS$data4$TS[,2],
      type = 'l', col = 'brown')
```

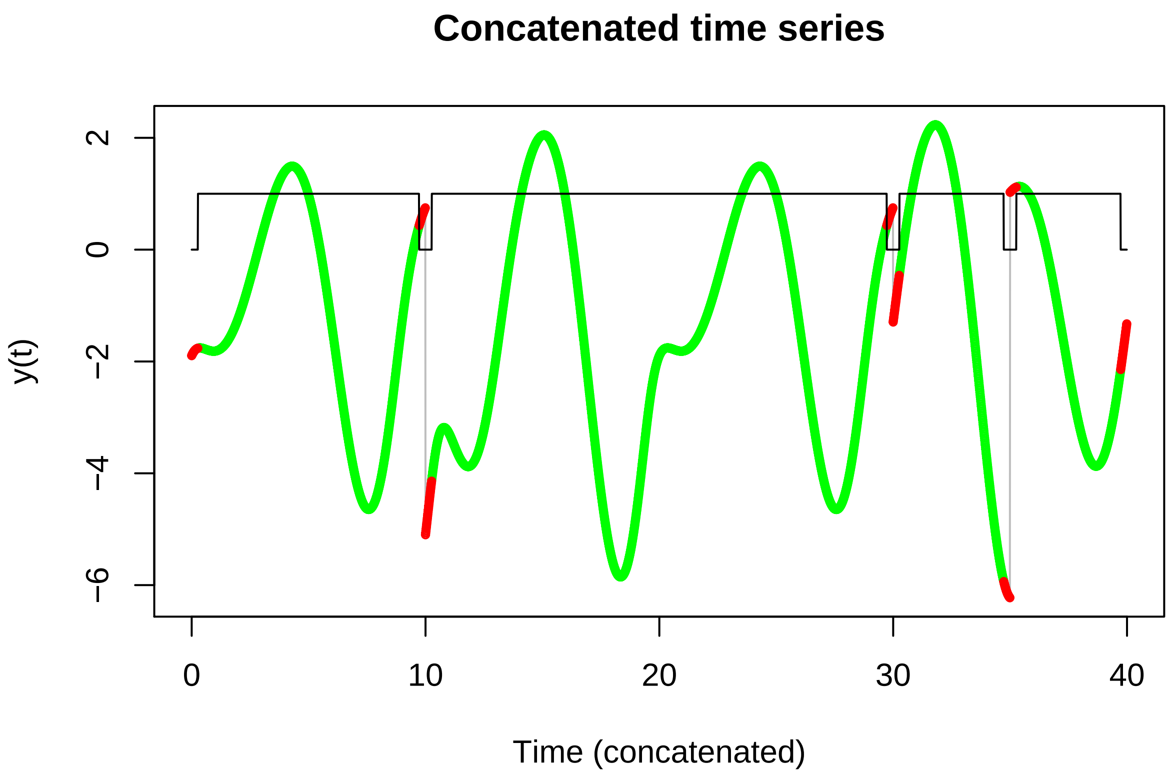


The function `concat` concatenates the set of separated time series into a single time series, and provides a corresponding weight vector `W` that takes the bundary conditions between successive time series into account, in order to avoid any discontinuous effect.

```r
# Concatenate the data set into a single time series
winL = 55
concaTS <- concat(svrlTS, winL = winL)
```

A large value is chosen for parameter `winL` just to highlight its effect on the next plots. The gray line enables to clearly distinguish the discontinuities at the connexion of originally separated time series. Data points rejected at the boudary are plotted as red dots and kept ones as green dots. The corresponding weighted vector is plotted as a black line.

```r
# Plot the concatenated time series
plot(concaTS$sglTS$TS[,1], concaTS$sglTS$TS[,2],
     main = 'Concatenated time series',
     xlab = 'Time (concatenated)', ylab = 'y(t)',
     type = 'l', col = 'gray')
lines(concaTS$sglTS$TS[concaTS$sglTS$W == 1,1],
      concaTS$sglTS$TS[concaTS$sglTS$W == 1,2], type = 'p', col = 'green', cex = 0.5)
lines(concaTS$sglTS$TS[concaTS$sglTS$W == 0,1],
      concaTS$sglTS$TS[concaTS$sglTS$W == 0,2], type = 'p', col = 'red', cex = 0.5)
lines(concaTS$sglTS$TS[,1], concaTS$sglTS$W, type = 'l')
```



The output of function `concat` can directly be used for global modelling:

```r
GPout2 <- gPoMo(data = concaTS$sglTS$TS[,2], tin = concaTS$sglTS$TS[,1],
                dMax = 2, nS = 3, winL = winL, weight = concaTS$sglTS$W, show = 1,
                IstepMin = 10, IstepMax = 12000, nPmin = 6, nPmax = 12, method = 'rk4')
```

```r
GPout2 <- gPoMo(data = concaTS$sglTS$TS[,2], tin = concaTS$sglTS$TS[,1],
                dMax = 2, nS = 3, winL = winL, weight = concaTS$sglTS$W, show = 0,
                IstepMin = 10, IstepMax = 12000, nPmin = 6, nPmax = 12, method = 'rk4')
```

The dynamic of the original system is effectively reproduced by sereral models.

Output visualization and global models validation

The outputs of the `gPoMo` function are gathered in one single list. The next vignette `4 VisualizeOutputs` aims to show how the results are organized in this list. Examples of model validation are then presented in the vignette `5 Predictability`.