GPoM : 1 Conventions

Sylvain Mangiarotti & Mireille Huc

2020-02-18

The Generalized Global Polynomial Modelling (GPoM) package allows a generic formulation of any Ordinary Differential Equations (ODEs) in polynomial form. The aim of the present vignette 1 Conventions is to introduce briefly the way to describe a set of polynomial ODEs with GPoM and to show how to perform its numerical integration1.

Conventions used to describe a polynomial

The polynomial description is based on a convention defined by the function \texttt{regOrd} that provides the order of the polynomial terms. This convention depends on the model dimension (that is the number \texttt{nVar} of state variables), and on the maximum polynomial degree used for the formulation (defined by the parameter \texttt{dMax}). This order can be visualized using the \texttt{polabs} function. For instance, for \texttt{nVar = 3} and \texttt{dMax = 2}, the convention used to formulate a polynomial will be:

\begin{verbatim}
nVar = 3
dMax = 2
polabs(nVar = nVar, dMax = dMax)
\end{verbatim}

\begin{verbatim}
## [1] "ct" "X3" "X3^2" "X2" "X2 X3" "X2^2" "X1" "X1 X3"
## [9] "X1 X2" "X1^2"
\end{verbatim}

This formulation has \texttt{pMax = 10} terms:

\begin{verbatim}
pMax = d2pmax(nVar, dMax)
\end{verbatim}

Based on this convention, one single ordinary differential equation (ODE) in polynomial form with \texttt{nVar} variables and of maximum polynomial degree \texttt{dMax} can be formulated as one single vector using the convention given by \texttt{regOrd(nVar, dMax)}. As an example, the equation :

\[ \frac{dX_1}{dt} = 1 + 2X_1 - 3X_1X_3 + 4X_2^2 \]

has the three variables \( X_1, X_2 \) and \( X_3 \) (it thus requires at least \texttt{nVar = 3}) and is of maximum polynomial degree two due to terms \( X_1X_3 \) and \( X_2^2 \) (it thus requires at least \texttt{dMax = 2}). Following the convention defined by \texttt{polabs(nVar = 3, dMax = 2)}, it will require the definition of the following vector of parameters:

\begin{verbatim}
param <- c(1, 0, 0, 0, 4, 2, -3, 0, 0)
\end{verbatim}

Indeed:

\begin{verbatim}
nVar = 3
dMax = 2
cbind(param, polabs(nVar, dMax))
\end{verbatim}

\begin{verbatim}
## [1,] "1" "ct"
\end{verbatim}

The same convention will be used for any other equation. Note that, by default, the notation used for the variables is $X_1$, $X_2$, etc. However, to facilitate the analysis, alternative notations may also be used using the optional parameter `Xnote`:

```
poLabs(3, 2, Xnote = 'y')
```

or for a full choice of the notation:

```
poLabs(3, 2, Xnote = c('x', 'W', 'y'))
```

### Definition of a set of polynomial ODE

A set of $N$ equations will require the definition of $N$ parameter vectors and will thus be represented by a matrix of `pMax` lines by `nVar` columns. For example, the Rössler system\(^2\) is defined by a set of three equations

\[
\begin{align*}
\frac{dx}{dt} &= -y - z \\
\frac{dy}{dt} &= x + ay \\
\frac{dz}{dt} &= b + z(x - c).
\end{align*}
\]

For ($a = 0.52$, $b = 2$, $c = 4$), this system can be described by three vectors (one for each equation)

```r
# parameters
a = 0.52
b = 2
c = 4

# equations
Eq1 <- c(0, -1, 0, -1, 0, 0, 0, 0, 0)
Eq2 <- c(0, 0, 0, a, 0, 0, 1, 0, 0)
Eq3 <- c(b, -c, 0, 0, 0, 0, 1, 0, 0)
```

The model formulation is obtained by concatenating the vectors of the three equations into one single matrix $K$ containing all the coefficients of the model:

```
K = cbind(Eq1, Eq2, Eq3)
```

The corresponding model equations can be edited in a mathematical form using the function `visuEq()`

```
visuEq(K)
```

```
## dX1/dt = -1 X3 -1 X2
```
## $\frac{dx_2}{dt} = 0.52 \, x_2 + 1 \, x_1$

## $\frac{dx_3}{dt} = 2 - 4 \, x_3 + 1 \, x_1 \, x_3$

By default, the notation used in `visuEq` for the variables is `X` with an indicative number, such as `X1`, `X2`, etc. Alternative notation can be used to edit the equations with the `visuEq` function using the optional parameter `substit`. For `substit = 1`, single letters are automatically chosen such as

```r
visuEq(K, substit = 1)
```

## $\frac{dx}{dt} = -1 \, z - 1 \, y$

## $\frac{dy}{dt} = 0.52 \, y + 1 \, x$

## $\frac{dz}{dt} = 2 - 4 \, z + 1 \, x \, z$

The notation can be defined also manually, such as:

```r
visuEq(K, substit = c("U", "V", "W"))
```

## $\frac{dU}{dt} = -1 \, W - 1 \, V$

## $\frac{dV}{dt} = 0.52 \, V + 1 \, U$

## $\frac{dW}{dt} = 2 - 4 \, W + 1 \, U \, W$

### Numerical integration

The numerical integration of the model defined by matrix `K` can be done using the `numicano` function. It requires the use of the external package `deSolve`. The following parameters are required as input:

```r
# The initial conditions of the system variables
v0 <- c(-0.6, 0.6, 0.4)
# the model formulation K (see former section)
# the number of integration steps `Istep`
Istep <- 5000
# the time step length `onestep`
onestep = 1/50
# the model dimension `nVar`
nVar = 3
# the maximum polynomial degree `dMax`
dMax = 2
```

The numerical integration is launched as follows:

```r
outNumi <- numicano(nVar, dMax, Istep = Istep, onestep = onestep, KL = K, v0 = v0)
```

The output of the function `numicano` is a list that contains (1) a memory `KL` of the model parameters

```r
outNumi$KL
```

from which `nVar` and `dMax` (required to reformulate the equations) can be retrieved

```r
# nVar
dim(outNumi$K)[2]
# dMax
pMax <- dim(outNumi$K)[1]
p2dMax(nVar, pMaxKnown = pMax)
```
and (2) the simulations $reconstr$. This matrix has $nVar + 1$ columns. The first one is the time, the other ones correspond to the variables of the system ($X_1, X_2, X_3, ...$) (or ($x, y, z$) to keep the formulation used previously in the text).

Note that all the other input parameters used in numicano can be retrieved from the outputs:

```r
# initial conditions
head(outNumi$reconstr, 1)[2:(nVar+1)]
# time step
diff(outNumi$reconstr[1:2,1])
# number of integration time step
dim(outNumi$reconstr)[1]
```

The simulated time series can be plotted as follows:

```r
plot(outNumi$reconstr[,1], outNumi$reconstr[,2], type='l',
     main='time series', xlab='t', ylab = 'x(t)')
```

![Time series](image)

and the plot of the phase portrait as well:

```r
plot(outNumi$reconstr[,2], outNumi$reconstr[,3], type='l',
     main='phase portrait', xlab='x(t)', ylab = 'y(t)')
```
Next steps

The aim of the GPoM package is to retrieve ODEs from time series using global modelling. Such type of modelling may require a careful data preprocessing. Simple examples of preprocessing will be given in the next vignette 2 Preprocessing. Examples for applying global modelling to time series will then be presented in vignette 3 Modelling.