RiDMC: an R package for the numerical analysis of dynamical systems

Antonio, Fabio Di Narzo\textsuperscript{1} Marji Lines\textsuperscript{2}

\textsuperscript{1}Università degli studi di Bologna
\textsuperscript{2}Università degli studi di Udine

UseR! 2008, Dortmund 12-08-2008

Dynamical Systems

- Dynamical systems theory is an interdisciplinary field, with major contributions coming from mathematics and physics but also many other fields like population studies and meteorology.
- A dynamical system is a mathematical model which formalizes the ‘rules’ describing the time dependence of a point’s position in its ambient space.
- The point symbolizes a state of the system, and is usually represented as a \textit{d}-variate real vector.
- Examples of dynamical systems include the description of the swinging of a clock pendulum, the flow of water in a pipe, the number of fish each spring in a lake, the daily rainfall in a city, etc.
RiDMC: the story

- iDMC (the interactive Dynamical Model Calculator) is a stand-alone Java application -with GUI- from which the C library idmclib originated as a spin-off (http://idmc.googlecode.com)

- idmclib is a standard-C library which relies on the LUA library for model code interpretation and on the Gnu Scientific Library (GSL) for computational tasks and random number generation. The idmclib is small, self-sufficient, and documented. License: GPL-v2 (http://idmclib.googlecode.com)

- RiDMC is a self-contained R package which internally uses the idmclib C library for core numerical analyses, and exploits R power for delivering a more complete, interactive and flexible environment to the final user for the numerical analysis of dynamical systems

RiDMC workflow

What is the typical workflow with RiDMC?

- write down the model in the LUA language, save it in a plain text file
- load the model as an R object
- perform analyses by using one or more model methods
- plot resulting objects
Writing models

- Models are specified in the interpreted LUA language
- The language is very easy to learn, and many models are already given as examples

Hénon map

\[
\begin{align*}
    x_{t+1} &= a - x_t^2 + b y_t \\
    y_{t+1} &= x_t
\end{align*}
\]

name = `Henon`
type = `D`
parameters = `{`a`, `b`}`
variables = `{`x`, `y`}`
function f(a, b, x, y)
    x1 = a - x^2 + b * y
    y1 = x
    return x1, y1
end

Analyzing a model

- Package design is object oriented, and all major analysis functions have been written as (S3) Model methods
- To date, the following methods are available:

<table>
<thead>
<tr>
<th>function</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trajectory, TrajectoryList</td>
<td>Model trajectories</td>
</tr>
<tr>
<td>Basin, BasinMulti</td>
<td>Basins of attraction</td>
</tr>
<tr>
<td>Bifurcation</td>
<td>Bifurcation diagram</td>
</tr>
<tr>
<td>LyapunovExponents</td>
<td>Lyapunov exponents</td>
</tr>
<tr>
<td>cycles</td>
<td>Periodic Cycles</td>
</tr>
</tbody>
</table>

- Each method returns an object which can be directly plotted by the usual plot method
A first, basic explorative analysis of a dynamical system involves the visual inspection of model trajectories. Trajectories can be plotted vs time axis or represented in the system state space, where time dimension is lost, but other model features can be appreciated. With RiDMC one can easily compute and plot trajectories for both discrete and continuous time dynamical systems.
Trajectories (II)

```
> m <- Model(`henon.lua```
> tr <- Trajectory(m, par, var, time, transient)
> tr
```

= iDMC model discrete trajectory =
model: Henon
parameter values: 1.42 0.3
starting point: 0 0
transient length: 10000
time span: 1000

Trajectories (III)

```
plot(tr)
```

![Graph showing the trajectory of a Henon model](image)
A key aspect of a dynamical system is its limit behaviour, i.e. the system's state as time tends to infinity.

As we have already seen, this can be approximated by using the Trajectory method and exploiting the transient option.

Even more useful in this respect can be the TrajectoryList method, which shows multiple trajectories in the same plot, by allowing for variations in starting points and/or parameter values.
Attractors

\[
\text{par} \leftarrow \text{c}(a = 1.4, b = 0.3)
\]

\[
\text{var} \leftarrow \text{list}(\text{c}(x = -1, y = -1), \text{c}(x = 1, y = 1))
\]

\[
\text{trL} \leftarrow \text{TrajectoryList}(m, n=20, \text{par}, \text{var}, \text{time}=50)
\]

\[
\text{plot(trL)}
\]

Basins of attraction

\[
\text{bs} \leftarrow \text{Basin}(m, \text{par}, \text{xlim}, \text{ylim}, \text{transient}, \text{iterations})
\]
One of the more interesting possibilities of nonlinear systems is sensitive dependence on initial conditions.

The Lyapunov Exponent (LE) measures the average rate of divergence in time of two nearby trajectories:

\[ |\delta x_t| \approx e^{\lambda t} |\delta x_0| \]

Positive values of \( \lambda \) indicate SDIC and suggest chaotic attractors.

Computing the value of \( \lambda \) can be very hard to do analytically, but numerical approximations can be obtained with RiDiMC.
Sensitive Dependence on Initial Conditions

```r
par <- c(a = 1.4, b = 0.3)
x0 <- c(x = 0, y = 0)
var <- list(x0, x0 + 0.001)
trL <- TrajectoryList(m, n = 2, par, var, time = 30)
```

Lyapunov exponents (II)

```r
ly <- LyapunovExponents(m, par, var, time, par.min, par.max)
ly
```

=iDMC Lyapunov exponents diagram=
Model: Henon
Starting point: x = 0.5, y = 1
Parameter values: a = 1.4, b = 0.3
Varying par.: a
Varying par. range: [ 0.3, 1.4 ]
MLE range: [ -0.5975, 0.4279 ]
Lyapunov exponents (III)

Note

RiDMC isn’t just for toy models...
Current status

- The idmclib C API is quite stable. Currently working on documentation and distribution system
- RiDMC core computing functions are stable too
- The plotting functions (grid-based) may change in the future
- Extract raw data and write your custom plotting functions if you want forward-compatibility of your code!

Perspectives

- fix bugs
- stabilize plotting functions
- add more analysis routines
The end.