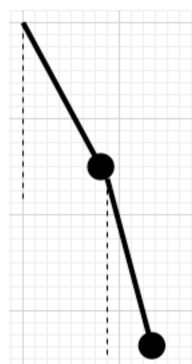
Hamiltonian Chaos: The Double Pendulum

Simulation, Poincare Plot, Sonification

SE Computerorientierte Physik, SS 2006

Katharina Vogt



Contents:

The tutorial for this project is taken from the Book:
An Introduction to Computer Simulation Methods,
2nd Edition,
Harvey Gould, Jan Tobochnik, Addison Wesley, 1996.

The physical system

A pendulum (consisting of a weight of mass m and a massless cord of length l) is hanged on a solid ceiling; on this first a second, identical pendulum is fixed. The system has two degrees of freedom: it may only move in a plane.
Energy is conserved (there is no friction), thus it is a hamiltonian system, which may be described by its canonical momenta, p_1 and p_2 respectively, and its canonical coordinates (the angles),q_1 and q_2.

$$hamilton = \frac{1}{2ml^2}\frac{p_1^2+2p_2^2-2p_1p_2cos(q_1-q_2)}{1+sin(q_1-q_2)^2} + mgl[3-2cosq_1-cosq_2]$$

Depending on the initial conditions, the system behaves regularly (as done in the simulation file *Doppelpendel-regulaer.rtf*) or shows chaotic evolution (as in file *Doppelpendel-life.rtf*).

The numerical model

This physical system is modelled using the forth-order Runge-Kutta Method. This is an iterative method for solving differential equations numerically, the error of the solution being of the order of the step size to the power of 5 (h^5). Originally, this technique was developed around 1900 by the German mathematicians C. Runge and M.W. Kutta.

For a scalar or a function $y\prime = g(t, y)$, the RK4-approximation is given by

$$y_{n+1} = y_n + \frac{h}{6}[k_1 + 2k_2 + 2k_3 + k_4].$$

Each value is determined by the previous value and the increment h times an estimated slope, which is a weighted average of slopes. The coefficients used are given by:

$$k_1 = g(x_n, y_n)$$
$$k_2 = g(x_n + \tfrac{h}{2}, y_n + \tfrac{k_1}{2})$$
$$k_3 = g(x_n + \tfrac{h}{2}, y_n + \tfrac{k_2}{2})$$
$$k_4 = g(x_n + h, y_n + k_3)$$

The double pendulum is described by the following equations:
The vector y of the generalized coordinates is $y = (q_1, p_1, q_2, p_2)$. Its derivative $y\prime = g(y)$ gives the following tremendous expression (where $Q = q_1 - q_2$).

$$g(y) = y\prime = \begin{pmatrix} q_1\prime \\ p_1\prime \\ q_2\prime \\ p_2\prime \end{pmatrix} =$$

$$= \frac{1}{ml^2} \cdot \begin{pmatrix} \frac{p_1 - p_2 cosQ}{1+sin^2Q} \\ -\frac{1}{2ml^2} \cdot [2p_1p_2 \cdot \frac{sinQ}{1+sin^2Q} - \frac{p_1^2+2p_2^2-2p_1p_2cosQ}{(1+sin^2Q)^2} \cdot 2sinQcosQ] - mglsinq_2 \\ \frac{2p_2 - p_1 cosQ}{1+sin^2Q} \\ -\frac{1}{2ml^2} \cdot [2p_1p_2 \cdot \frac{sinQ}{1+sin^2Q} - \frac{p_1^2+2p_2^2-2p_1p_2cosQ}{(1+sin^2Q)^2} \cdot 2sinQcosQ] - 2mglsinq_1 \end{pmatrix}$$

The algorithm is organized as follows:

1. declaration and setting of variables:
   generalized coordinates, hamilton and physical variables, begin and ending time $(t_0, t_{max})$and step size h, control variables (z counting the jumps and the energy testing variable *htest*, see below), RK4-coefficients, initial conditions: starting angle of the first pendulum q_0 and energy h_0 (given to the second pendulum as momentum)

2. definition of functions:
   hamilton-energy function and $g(q_1\prime, p_1\prime, q_2\prime, p_2\prime)$

3. setting and posting of the initial conditions for the simulation run

4. testing, if the initial conditions make sense (that is if the function $f(q_0, e_0)$ is a real number)

5. creating the array a for the storage of the results and running the RK4-algorithm

6. test during each run:

   - if the system is out of reasonable physical borders (i.e. the energy is more than doubled or halved) the simulation is stopped; this happens for extreme initial conditions.
   - If the angles q_1 or $q_2 > 2 \cdot \pi$), it is set back (q_1 and q_2 modulo 2π. The number of these jumps is stored, but was usually neglectable in relation to the size of the simulation.

7. storing of a time-like index, the four generalized coordinates and the overall energy

Analyzing methods

Graphical analysis

A first glimpse of the data could be done by plotting the generalized coordinates (each separately) and the overall energy H. This gives an idea about the periodicity and the conservation of the energy over time.
In figure 1 the overall energy for a simulation run (DP-01) is shown. [1]

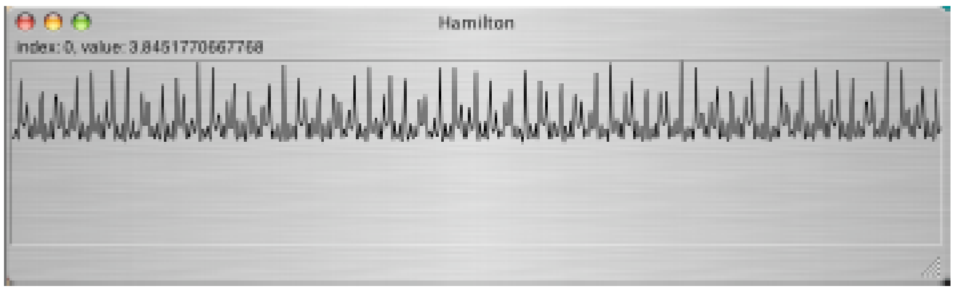


Figure 1: Hamilton of a simulation run. The energy is oscillating, which is possibly due to rounding errors, but stays within 50 and 200 per cent of the initial energy.

A quick idea of the interaction between the two pendulums could be gained by comparing either the angles or the momenta with each other. The calculated difference between the angles are shown in figure 2. For reasons of conventional paper size, and the huge amount of data, only about half of the data is displayed.

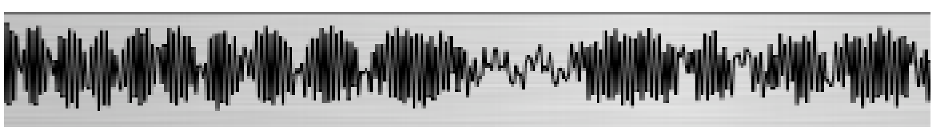


Figure 2: Difference between the two angles q_1 and q_2 at a range of calculation steps in the simulation run for the Poincare plot. See the next section for details.

The evolving system may be pictured in a phase space, giving a so-called Poincare-plot (for the SC3-code refer to *Doppelpendel-visual.rtf*). In this visualization method, one plane of the (in this case) 4-dimensional phase space is drawn; each plotted point in this plane indicates *where* a periodical, quasi-periodical or chaotical trajectory returns to this plane.

[1] The plotting function in SuperCollider3 is more determined for interactive use than for display of stored data: there is no axis labelling, but a click on the picture gives the current value. These values and the axis are explained in the text.

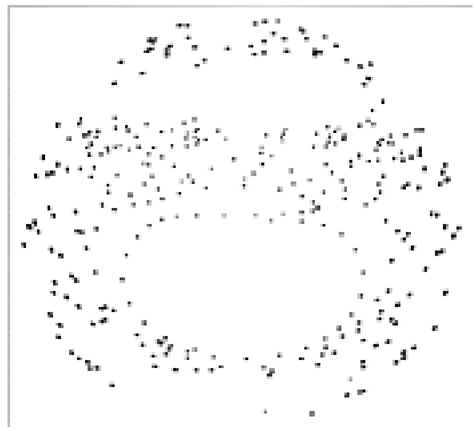


Figure 3: Poincare plot of the simulation run "DP10" (see appendix for details). On the abscissa, the angle q_1 is drawn, and on the ordinate the momentum p_1. For the plot, the program was slightly modified, in order to spare calculation time: artificial jumps were built in, whenever the angles exceeded 2π. Then, p_2 was set to zero and p_1 accordingly calculated out of the energy function ($p_1 = f(q_1, q_2, h_0)$)). This helped starting again at different states of the phase plot, an provide a more complete picture.

For this project, the Poincare-plot was drawn using q_1 as abscissa and p_1 as ordinate axis. Points were plotted whenever $q_2 = 0$ and $p_2 > 0$.

Sonification

Additionally to this graphical analysis, several approaches to *sonify* the results have been attempted in file *Doppelpendel-Sonifikationen.rtf*:

Poincare-Task

In a first attempt, *temporal* information of the periodicity of the system was used[2]: whenever the second pendulum went through the perpendicular position and moved to the right, the point in time was stored, and later sonified as a rhythm of pitched beeps. (This is the same data as used for the Poincare-plot.) A regular, periodic movement would lead to a steady beat, whereas a chaotic movement gives no chance to find such a beat. Slight chances would indicate a quasi-periodic movement, and may easily be detected. This behaviour may also indicate a small overall energy in the system.
This method has been called *"Poincare-task"* in this sonification.

[2] The indices of the numerical calculation are proportional to the physical time t. The index equals $i = t/h$, h being the step size of the RK4-algorithm.

In the file, DP-01-poincare.mp3, a regular movement may be observed. The two pendulums follow the same rhythm until the end.

Hints for listening to the rhythm-based audio files:

- *Notes played shortly after one another, and perceived as a overdrive in the channels are obviously a result of the filtering of data: in the code, the following condition was used to store the time-indices:*
  `if ( a[i][1].round(0.01) == 0.`
  If the pendulum moved rather slow, several indices would be stored one after the other. This is not a huge problem, as the sound is very different from a normal beep, and does not take much longer than a normal time step.
- *In the Poincare-Task, the beginning of a loop is marked with a sine wave. In the next sonification (Task-Zero-Angles), the files start with 2 second introductional sounds (which are slightly detuned) in order to assure synchronization of the two channels.*

Task of zero angles

A second sonification of periodicity uses information on the angles of the two pendulums. Each point of time where a pendulum went through the perpendicular position was stored. These rhythms are played simultaneously out at two channels (i.e. left and right channel of headphones). It is a bit harder to follow the rhythm in this task than in the Poincare-Task, but periodic, quasi-periodic and chaotic movement may still be distinguished.

In the example DP-01-zeroangles.mp3, the same pattern persists over time (right – left – right - right). It is not that easy to follow, as the playback is rather slow because the sound would override too easily because the data filtering has the same problems as mentioned above.

Modulation Task

A completely different approach treats all the information about the two pendulums, without filtering them in the first place. This information is used to modulate two sine waves of different frequencies (angles) and amplitudes (canonical moments), going out on two different channels (as above). In this example, the association link between the model and the mapping is very direct, and the swinging of the two pendulums can be "seen" as patterns. (The brain interprets two equal (or very similar) but phase-delayed sine waves as one circular movement.)

This is most easily perceived in a very slow playback, where the velocity (a parameter of the sonification) was set to 30 per cent of its "natural" value. Compare the example DP-02-AmpAndFreq–vel-0.3.mp3.

Now the speed of the modulation may be changed dramatically, in order to provide a completely different sound. At 50 times of the original velocity, the examples are very short (the clicks are artefacts that mark the loop) but give spectral information. The changing of the timbre of these frequency clusters changes with a changing system and should be very noisy for the chaotic case. The example DP-02-AmpAndFreq–vel-0.3.mp3 may be compared with a different simulation run, DP-01-AmpAndFreq–vel-50.mp3, whose overtones sound completely different. For comparison the files at velocity 1 (the original) and 2 (the doubled speed) are provided as well.

For a more complete analysis of the double pendulum and the sonification tools more simulation examples have to be run and compared. In the above examples (except of the different simulation for the Poincare plot) the initial conditions led to a regular or quasi-perdiodic movement.

Appendix

Data

Data was stored using archives, and from the original array a (containing an index, q_1, p_1, q_2, p_2 and the overall energy) data was extracted in various ways in the file *doppelpendel-datenextract.rtf*.
The initial conditions of the simulation runs used in the examples above are the following:

- **DP10 = double pendulum 10**
  - starting angle of $q_1 = 0.1$
  - starting energy $H_0 = 10$
  - this leads to a starting momentum $p_2 = 3.146$
  - 400.000 time steps were taken into account for the analysis (then the algorithm became unstable)
  - there were no jumps during the calculation

- **DP-01**

  - starting angles of $q_1 = 0.71$
  - starting energy $H_0 = 4.9$
  - starting momentum was $p_1 = 0.501$
  - a had 1000001 entries
  - there were no jumps during the calculation

- **DP-02**
  - starting angles of $q_1 = -0.58$
  - starting energy $H_0 = 3.8$
  - starting momentum was $p_1 = 0.912$
  - a had 1000001 entries
  - there were no jumps during the calculation

Resources used for this document

- Lecture notes of *"Computerorientierte Physik", SS 2006, Prof. Chr. Gattringer.*
- Wikipedia: http://en.wikipedia.org
- The programming language used is *"SuperCollider3"*, a programming language and engine for real time audio synthesis, originally written by James McCartney and now under the free GNU license. It is available for download at: http://www.audiosynth.com.

I would like to thank Julian Rohrhuber and Till Bovermann for contributing to the data extracting, visualization and sonification of this project.