Experiment 315: The Double Pendulum

Aim

To look at some aspects of the surprising motion of the double pendulum, a system which is still not completely understood, even theoretically.

Note: A familiarity with PYTHON would be useful but is not absolutely necessary.

References:

1. Richter and Scholz, "Chaos in Classical Mechanics", in " Stochastic Phenomena and Chaotic Behaviour in Complex Systems" Springer-Verlag (1983).
2. Baker and Gollub, " Chaotic Dynamics, an Introduction" Cambridge University Press (1990) [General library call number 003.B16].
3. Wolf *et al*, "Determining Lyapunov Exponents from a Time Series" Physica 16D, 285 (1985) [can be accessed online by looking for the Physica D periodical in the Voyager Library catalog].
4. Blackburn and Smith, "Driven Pendulum for Studying Chaos" Rev. Sci. Instr. 60, 422 (1989)

Ref. 1 touches on some of the incredible richness of the double pendulum system. Ref. 2 has a good section on Lyapunov exponents (from page 120 onwards). In addition, listing 11 of Appendix B of Ref. 2 contains a program in TRUEBASIC to calculate the Lyapunov exponents for the simple driven damped pendulum in Experiment 314. We have translated this program to TURBOC and used it to calculate the exponents for the double pendulum. Ref. 3 describes a method to extract Lyapunov exponents from an experimental time series. Ref. 4 describes the instrumentation of Experiment 314. We have used essentially the same technology to measure the angles.

Apparatus

The double pendulum is made from aluminium with a hardened steel pivot and jewel bearings to minimise friction. The system parameters are:

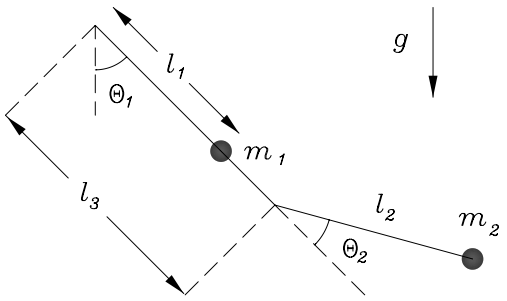

Figure 1: Schematic diagram of the apparatus.

m_1	=	$315.50\,\mathrm{g}$	l_1	=	$25.5\,\mathrm{mm}$
m_2	=	$76.53\,\mathrm{g}$	l_2	=	$23.5\,\mathrm{mm}$
I_1	=	$9.07 \times 10^{-4}\,\mathrm{kg\ m^2}$	l_3	=	$57\,\mathrm{mm}$
I_2	=	$6.90 \times 10^{-5}\,\mathrm{kg\ m^2}$	g	=	$9.799403\,\mathrm{m\ s^{-2}}$

m_1 and m_2 are the masses of the upper and lower pendula respectively. l_1 and l_2 are the displacements of the centre of mass from their respective pivot points (see Fig. 1). They were measured by balancing each pendulum individually on a length of thread. l_3 is the distance between the two pivot points. I_1 and I_2 are the moments of inertia of the two pendula. These were measured dynamically, by timing small angle swings for each pendulum in turn. The period of each pendulum is given by

$$T_i = 2\pi\sqrt{\frac{I_i}{m_i l_i g}}\,.$$

Each of the angles θ_1, θ_2 is measured using an HP model HEDS-6100 code wheel and an HP model HEDS-9000 encoder module. The encoder is a C-shaped module with a single lensed LED on one side of a 1.8 mm gap, and four photo-detectors on the other. The code wheel, a 5 cm diameter metal foil disk, rotates between the LED and the detectors. Slots are cut into the disk in such a way that when two detectors are illuminated, the adjacent pair are in darkness. There are two channels of digital output, consisting of two square waves 90° out of phase, which either move forwards or backwards depending on which way the pendulum is rotating. Thus the angle is measured to $360°/4000 = 0.09°$ accuracy, and direction is sensed as well.

For angle θ_1, the output of the encoder is fed directly to an interface attached to a PC via a USB connection.

To measure angle θ_2 we did not want to use wires (which would twist) or brushes (which would introduce extra friction) to couple the information out, so an alternative had to be found. θ_2 is measured with a encoder mounted on the upper pendulum in which the two output channels are amplitude modulated up to 8 and 10 MHz using two ceramic resonators and a 7400 NAND gate. The encoder gets its power supply through voltage applied directly to the frame of the pendula and transferred to the upper pendulum through the axis of rotation. The output of the NAND gate drives a small transmitting coil, which is inductively coupled to a receiving coil mounted on the frame. The signal from this is fed via a coaxial cable to the box housing the USB interface, where it is amplified using an NE592 broadband video amplifier. Two more of these amplifiers separate out the 8 MHz and 10 MHz components using the same type of ceramic resonators; then rectification and low-pass filtering stages and a TTL-compatible comparator (LM339) reconstructs the square wave signals. It is then transferred to the PC in the same way as θ_1.

Theory

The Lagrangian for an undamped double pendulum free to move in a plane is given by

$$L = K - V \tag{1}$$

where the kinetic energy K and the potential energy V are

$$\begin{aligned}
K &= k_1\dot{\theta}_1^2 + k_2\left(\dot{\theta}_1 + \dot{\theta}_2\right)^2 + k_3\dot{\theta}_1\left(\dot{\theta}_1 + \dot{\theta}_2\right)\cos\theta_2 && (2)\\
V &= \nu_1\left(1 - \cos\theta_1\right) + \nu_2\left(1 - \cos(\theta_1 + \theta_2)\right) && (3)\\
\text{and}\quad k_1 &= \frac{1}{2}\left(I_1 + m_2 l_3^2\right) && (4)\\
k_2 &= \frac{1}{2}I_2 && (5)\\
k_3 &= m_2 l_2 l_3 && (6)\\
\nu_1 &= \left(m_1 l_1 + m_2 l_3\right) g && (7)\\
\nu_2 &= m_2 l_2 g && (8)
\end{aligned}$$

The momenta conjugate to θ_1 and θ_2 are

$$\begin{aligned}
P_1 &= \frac{\partial L}{\partial\dot{\theta}_1} = 2\dot{\theta}_1\left(k_1 + k_2 + k_3\cos\theta_2\right) + \dot{\theta}_2\left(2k_2 + k_3\cos\theta_2\right) && (9)\\
P_2 &= \frac{\partial L}{\partial\dot{\theta}_2} = \dot{\theta}_1\left(2k_2 + k_3\cos\theta_2\right) + 2\dot{\theta}_2 k_2 && (10)
\end{aligned}$$

P_1 is just the total angular momentum of the system about the upper pivot point, while P_2 does not have such a simple interpretation. From these we may write down the Euler-Lagrange equations, which with some rearrangement are

$$\ddot{\theta}_1 = \frac{a_1 \sin(\theta_1+\theta_2) - a_2 \sin\theta_1 + a_3 \sin\theta_2}{4k_1k_2 - k_3^2\cos^2\theta_2} \tag{11}$$

$$\ddot{\theta}_2 = \frac{b_1 \sin(\theta_1+\theta_2) - b_2 \sin\theta_1 + b_3 \sin\theta_2}{k_3^2\cos^2\theta_2 - 4k_1k_2} \tag{12}$$

where

$$\begin{aligned} a_1 &= k_3\nu_2\cos\theta_2 & \qquad b_1 &= 2k_1\nu_2 + k_3\nu_2\cos\theta_2 \\ a_2 &= 2k_2\nu_1 & \qquad b_2 &= 2k_2\nu_1 + k_3\nu_1\cos\theta_2 \\ a_3 &= 2k_2k_3\left(\dot{\theta}_1+\dot{\theta}_2\right)^2 + k_3^2\dot{\theta}_1^2\cos\theta_2 & \qquad b_3 &= \left(2k_2k_3 + k_3^2\cos\theta_2\right)\left(\dot{\theta}_1+\dot{\theta}_2\right)^2 \\ & & & \quad + \left(2k_1k_3 + k_3^2\cos\theta_2\right)\dot{\theta}_1^2 \end{aligned}$$

The variables $\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2$ (the two angles and corresponding angular velocities) define a four-dimensional space called "phase space". At any time the system has a unique value of $\theta_1, \theta_2, \dot{\theta}_1$ and $\dot{\theta}_2$ and so is described by a point in phase space. As time proceeds, this point will wander around phase space. If the phase space has more than two dimensions and the equations of motion are non-linear, chaos can arise. This is when two nearby points in phase space tend to diverge exponentially in time. In other words, if there is a small error in the measurement of the starting conditions, it is quickly magnified into a huge error, so that predictive power is lost; as you might guess, the weather is a chaotic system. Our system has four dimensions, and the equation are highly non-linear (all the trigonometrical functions on the right hand side), so we expect some fairly weird motion.

We may also use the Hamiltonian formulation, in which case we find

$$H = \sum_{i=1,2} P_i\dot{\theta}_i - L \tag{13}$$

$$= \frac{P_1^2k_2 + P_2^2\left(k_1+k_2+k_3\cos\theta_2\right) - P_1P_2\left(2k_2+k_3\cos\theta_2\right)}{4k_1k_2 - k_3^2\cos^2\theta_2} + \nu_1\left(1-\cos\theta_1\right) + \nu_2\left(1-\cos(\theta_1+\theta_2)\right) \tag{14}$$

With the idealisation of no damping, we have

$$H = K + V = E \tag{15}$$

where the total energy E is conserved. Hamilton's equations of motion are then given by

$$\dot{\theta}_i = \frac{\partial H}{\partial P_i} \tag{16}$$

$$\dot{P}_i = -\frac{\partial H}{\partial \theta_i} \tag{17}$$

and the four variables in our phase space are then $(\theta_1, \theta_2, P_1, P_2)$.

Small angle motion

For small angles θ_1, θ_2 we can linearise equations (11) and (12) to obtain

$$\begin{pmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{pmatrix} = \frac{1}{d}\begin{pmatrix} c_1 & c_2 \\ c_3 & c_4 \end{pmatrix}\begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} \tag{18}$$

$$\text{where} \quad c_1 = k_3\nu_2 - 2k_2\nu_1 \tag{19}$$

$$c_2 = k_3\nu_2 \tag{20}$$

$$c_3 = 2k_2\nu_1 + k_3\nu_1 - 2k_1\nu_2 - k_3\nu_2 \tag{21}$$

$$c_4 = -k_3\nu_2 - 2k_1\nu_2 \tag{22}$$

$$d = 4k_1k_2 - k_3^2 \tag{23}$$

Because the equations are now linear, the motion is regular, and easily analysed. There are two normal modes (see Experiment 312 "Coupled Pendulums" for a discussion of normal modes) and their angular frequencies are given by:

$$\omega_{1,2}^2 = \frac{k_1\nu_2 + k_2\nu_1 \pm \sqrt{(k_1\nu_2 + k_2\nu_1)^2 - \nu_1\nu_2(4k_1k_2 - k_3^2)}}{4k_1k_2 - k_3^2} \tag{24}$$

The corresponding eigenvectors have

$$\frac{\theta_2}{\theta_1} = \frac{k_2\nu_1 - k_1\nu_2 - k_3\nu_2 \mp \sqrt{(k_1\nu_2 + k_2\nu_1)^2 - \nu_1\nu_2(4k_1k_2 - k_3^2)}}{k_3\nu_2} \tag{25}$$

Zero gravity motion

If we tilt the pendulum on its side there is no restoring torque due to gravity, so that $g = 0$ as far as the pendulum is concerned. From equations (14), (15), (16) and (17) we can show that the total angular momentum P_1 is conserved, and that

$$\dot{\theta}_2 = \pm\sqrt{\frac{4E(k_1 + k_2 + k_3\cos\theta_2) - P_1^2}{4k_1k_2 - k_3^2\cos^2\theta_2}} \tag{26}$$

We may deduce from equation (26) that for a given energy E, the maximum value of P_1 is

$$P_1^{\max} = \sqrt{4E(k_1 + k_2 + k_3)} \tag{27}$$

which occurs when $\theta_2 = 0$ (i.e. the pendulum is stretched out). We can also see that $\dot{\theta}_2$ cannot be zero if $|P_1| < P_1^{\text{sep}}$, where

$$P_1^{\text{sep}} = \sqrt{4E(k_1 + k_2 - k_3)} \tag{28}$$

This means that the lower pendulum rotates ($\dot{\theta}_2$ never changes sign). However for $|P_1| > P_1^{\text{sep}}$, the lower pendulum oscillates with $|\theta_2| < \pi$.

Large angle motion

For large angles of swing the motion of the pendulum is chaotic. A definition of chaos is that nearby trajectories in phase space tend to diverge exponentially in time; that is, if we have two nearby starting points $\theta_i^{\text{I}}(0)$ and $\theta_i^{\text{II}}(0)$, then as we let the system evolve,

$$|\theta_i^{\text{II}}(t) - \theta_i^{\text{I}}(t)| \sim \exp(\lambda t) \tag{29}$$

The average rate λ of divergence or convergence is called a Lyapunov exponent. In n-dimensional phase space there are n Lyapunov exponents $\lambda_{i,\ldots n}$, which describe what happens to a small sphere of initial points (see page 120 of Ref. 2, in the Laboratory). The sphere is stretched into an ellipsoid, the n Lyapunov exponents describing the contraction or expansion of the ellipsoid along its principal axes. The ellipse also wanders around phase space and rotates.

If at least one Lyapunov exponent is positive, the system is chaotic; even if the starting conditions are known very precisely, the small error is rapidly magnified in time, and predictive power is lost.

The function `pendlyap` calculates the Lyapunov exponents for the double pendulum sitting on the bench. It does so by numerically integrating both the non-linear equations (11) and (12) for the given start conditions, and those same equations linearized around the large angle motion. This is done by calculating the Jacobian of equations (11) and (12) and is used to study how nearby trajectories converge or diverge in the four dimensions of phase space (i.e., for four orthonormal displacement vectors). Those four vectors essentially describe how a small sphere of initial points around the main trajectory evolve. A Gram-Schmidt orthonormalisation is carried out on the displacement vectors every few time steps, otherwise they ultimately all align along the fastest diverging direction (see Ref. 3 for a description of the algorithm and how it works). We will discuss more about this function below.

Procedure

Note that you are encouraged to look at the source code of the following PYTHON functions referenced in this pamphlet. All these functions are found in the `phylab.py` file.

Small angle motion

(1) Using the given system parameters and equations (24) and (25), calculate the two normal mode frequencies and the corresponding ratio θ_2/θ_1 for the eigenvectors.

(2) Switch on the computer and ensure the DC plug pack powering the black box that interfaces the pendulum to the USB port of the PC is also turned on.

(3) Open PYTHON (Spyder).

(4) Next, start the acquisition software `Expt315-Double Pendulum` from the Start Menu. It is located in `All programs` → `Physics` → `AdvLab`. We will be switching back and forth between the acquisition software (to acquire data) and PYTHON (to analyze and plot the data).

(5) Click on button "Poll Devices". If device is found, the "Select Device" drop down box is enabled.

(6) Look in "Select Device" drop down box for "exp315" and click on it.

(7) Click on "Open Device" button. If successful, this button is disabled and "Close Device" button is enabled. When you are done, make sure you click "Close Device" before closing down the program.

(8) Ensure pendulums are stationary before clicking on "Clear" and "Reset". This sets the starting angles of the encoder to zero for both pendulum. This is necessary, e.g., after a pendulum has done several complete round-trips, in which case the angle output by the encoder is a multiple of 2π rather than 0.

(9) Set the "Sampling Period" and "Time to Sample" to the values you want. You can start with a sample time of 1 ms, and a 5-second run (i.e. 5000 data points).

(10) Notice in the "Save Data To" edit box where the data will be saved, by default in your user profile directory, in `'C:/Users/.../DoublePendulum/DoublePendulum.txt'` where `...` is your UPI. If you wish, click on "Browse" to change that location or filename. Note that you will need to specify the location of your data file in the `acquire` PYTHON routine discussed below.

(11) Set pendulums in motion.

(12) Click on "Go". This button is disabled and "Stop" button will be enabled.

(13) Wait for data to stop rolling in. "Stop" button becomes disabled. Go to `File` → `Save` to save the data in the specified directory. You may also go to `File` → `Setup...` to tell the programme to perform this step automatically.

(14) Now switch to PYTHON and type

```
directory =         # location of your saved data
show_fig = 1        # 1 = plots data, 0 = no plot shown

t, p1, p2, sampling_period, myfig = acquire(directory,show_fig)
```

to load the acquired data.

The `acquire` function stores the data in the variables `t`, `p1` and `p2`, where `t` is in seconds and `p1` and `p2` are θ_1 and θ_2 respectively. The sampling period (in seconds) is also extracted from your data file. If you want Python to also plot the graph shown the acquisition software display, set `show_fig = 1`, else `show_fig = 0`. You are now ready to do some real work.

(15) Before acquiring any data, make sure to first stop the pendulums, and then click "Reset", as explained in (8) above.

(16) Set the start angles $\theta_1(0)$ and $\theta_2(0)$ so as to excite one of the normal modes [i.e. make $(\theta_1(0), \theta_2(0))$ an eigenvector]. Remember to keep both angles reasonably small ($< 30°$).

(17) Release the pendulum and click on "Go" simultaneously. Load the data into PYTHON and check that you have excited a normal mode (how can you tell?).

(18) In PYTHON, type `T = period(t,pn)` to estimate the period T of the motion from the zero-crossing times for either `pn = p1` or `p2`. Do your values of `T` make sense?

(19) You will now be using the following code to compute the power spectra of `p1` and `p2`. To this end, make sure you understand what it does.

Recall that the spectral density of a signal is the square modulus of its Fourier transform. Below, we first define a frequency vector `f` compatible with the sampling theorem. Note that the `fftshift` function shifts the zero-frequency component to the center of the spectrum by swapping the two half of the vector returned by `fft`. This is made necessary because the vector returned by the `fft` function first packs the positive frequencies, starting from 0, and then the negative frequencies.

```
import numpy as np
import matplotlib.pyplot as plt

n = len(t)

dt = np.mean(np.diff(t))
f = np.arange(-n/2,n/2,1)/(n*dt)

FTp1 = np.fft.fftshift(np.fft.fft(p1))  # Pendulum 1
spec = np.abs(FTp1)**2

plt.figure()
plt.semilogy(f,spec)
plt.axis([-10, 10, 0, 1e3*max(spec)])
plt.xlabel('Frequency [Hz]')
plt.ylabel('Spectral density [log]')
plt.show()
```

Are the peaks where you would expect? You may find that they are not very nicely resolved. How can you improve the spectral resolution? Note that other methods like the Burg Maximum Entropy algorithm might make a better job than `fft` at estimating the power spectrum of short time series but using `fft` is the simplest solution.

(20) Repeat (15) – (19) for the other normal mode, and for an arbitrary small angle start condition.

Zero gravity motion

(21) Calculate $P_1^{\text{sep}}/\sqrt{E}$ using equation (28).

(22) **Carefully** tilt the pendulum through about 90°, so that the plane of motion is parallel to the bench top. Ensure that the pendulum can swing freely through 360°. Now make fine adjustments until the pendulum experiences no restoring torque for any value of θ_1, i.e. gravity has no effect.

(23) Use the acquisition software and select a sample time of 50 ms, and a run of 60 seconds (1200 data points). Hold the upper pendulum and carefully set the lower pendulum rotating, then set the upper pendulum rotating as well. If $|P_1| < P_1^{\text{sep}}$, the lower pendulum will keep rotating, but if $|P_1| > P_1^{\text{sep}}$ the lower pendulum will stop rotating and start oscillating about the fully stretched position ($\theta_2 = 0$).

Due to the lower pendulum's greater damping, its rotation will slow down at a quicker rate than that of the upper pendulum. If initially $|P_1| < P_1^{\text{sep}}$, then as the two pendulums rotate and slow down, a transition in the lower pendulum's motion will occur when $|P_1|$ becomes greater than P_1^{sep}. Once this has been observed, carefully stop the pendulum by letting the upper pendulum gently brush against your hand, and then repeat this several times during the 60 second run.

(24) Once you are confident with this procedure and you have saved meaningful data, reload these into PYTHON.

(25) Type

```
theta = 90
show_fig = 1

P1, E, myfig = mom(t,p1,p2,sampling_period,theta,show_fig)
```

to calculate the momentum P_1 [Eq. (9)] and energy E [Eq. (15)] and to get a graph of θ_2 and $|P_1|/\sqrt{E}$. Print it out and use it to estimate $|P_1^{\text{sep}}|/\sqrt{E}$. Compare this with the theoretical value.

Large angle motion

(26) We will now use the acquisition software with a sample time of 1 ms and a run of 5 seconds (5000 data points) and `acquire` to get several runs of data. Make sure to stop the pendulums and click on "Reset" between each run. Each time, take the initial conditions as close as possible to

$$\theta_1(0) = -110^\circ \qquad \dot{\theta}_1(0) = 0$$
$$\theta_2(0) = +110^\circ \qquad \dot{\theta}_2(0) = 0$$

and the lower pendulum should just hang. Make sure you click "Go" just before you release the pendulums so as to clearly get the start of the motion.

(27) Use `acquire` to load the data and type

```
threshold = 1

p1cut,p2cut, myfig = pendcut(t,p1,p2,threshold)
```

to cut off the zero-velocity portion at the beginning. Try initially a cut-off angular velocity of `threshold = 1` or 1 event/ms, i.e. $1/4000 \times 2\pi$ rad s^{-1}. You can adjust this for example to values of up to 5 events/ms. This pads the data vectors out with zeros (so don't worry about the abrupt jump to zero at the end).

(28) Carefully return the pendulum to the vertical plane. If the data looks acceptable, type into PYTHON lines 1 – 10:

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  # set up 5 columns to store 5 runs of data, with 500 rows
5  p1array = np.zeros((len(p1cut),5))
6  p2array = np.zeros((len(p2cut),5))
7
8  # add data to p1array and p2array
9  p1array[:,0] = p1cut
10 p2array[:,0] = p2cut
11
12 plt.figure()
13 plt.plot(t, p2array[:,0], t, p2array[:,1])
14 plt.show()
```

(29) Repeat (26) – (27) four more times with as near as possible to the same start conditions. Remember to recycle lines 9 – 10 above:

```
# run 2:
p1array[:,1] = p1cut
p2array[:,1] = p2cut

# run 3:
p1array[:,2] = p1cut
p2array[:,2] = p2cut
```

and so on, because you want each column of `p1array` and `p2array` to contain one run of data.

(30) Type lines 12 – 14 to display the θ_2 motion for the first two runs on the same graph. Note that the trajectories initially almost overlay one another, but quickly diverge.

(31) Plot the log of the difference between two runs as a function of time by adding

```
plt.figure()
plt.plot(t, np.log(np.abs(p2array[:,1]-p2array[:,0])))
plt.grid()
```

before the `plt.show()` line.

Note the oscillations in the difference, and the overall divergence. Make a crude estimate of the dominant Lyapunov exponent (the average rate of divergence) by getting a printout and fitting a straight line to the plot by eye. We have

$$|\theta_2^{\mathrm{II}}(t) - \theta_2^{\mathrm{I}}(t)| \sim \exp(\lambda_1 t)$$

so λ_1 is simply the slope of the graph. Note that the estimate of λ_1 depends on how much you allow the two trajectories to diverge. There is a play-off between two factors: because λ_1 is the *average* rate of divergence, we should average over a long time; but as it is the divergence rate of *nearby* trajectories, we should not in theory allow them to get too far apart.

(32) Now get PYTHON to repeat this process for every pair of trajectories by using `findlyap`. This function uses an array `maxdiffv` which represents a range of values for the maximum angular difference between any two trajectories.

For each angle `maxdiffv[i]`, the function only uses data from time $t = 0$ up to the time that two trajectories have diverged by that angle. If this maximum angular difference is too small, we won't have averaged over a long enough time to get a good estimate of the Lyapunov exponent. If it is too big, we won't be measuring the divergence rate of "nearby" trajectories any more. If we are lucky, there will be an intermediate range of angles where our estimate of the Lyapunov exponent is independent of the maximum angular difference, so we might hope we have a good measurement of the exponent. Try

```
maxdiffv = np.linspace(10,360,10)

findlyap(p2array, maxdiffv)
```

for a start. The output is a plot of `lylv`, the dominant Lyapunov exponent, versus `maxdiffv`, the maximum difference between trajectories. If the plot levels off at some value for the Lyapunov exponent, we have found a reasonably uniform divergence rate over a range of difference angles which we can compare with a numerically calculated value.

(33) Compare the value of the Lyapunov exponent measured with the numerically calculated value by typing

```
tstep = 50*0.001             # seconds
inicond = np.array([-110, 110, 0, 0])*np.pi/180
maxiter = 1000

pendlyap(tstep,inicond,maxiter)
```

This may take a few tens of seconds. Observe the Lyapunov exponents converging over this time. Start with the initial parameters `inicond` and a time step of 50 ms between re-orthonormalisation, with `maxiter = 1000` steps.

List of Equipment

1. Pendulum assembly
2. Standalone PC with double pendulum software and PYTHON.
3. A set of custom PYTHON functions in `phylab.py`: `acquire`, `period`, `mom`, `pendcut`, `findlyap`, and `pendlyap`.

R. B. Levien
S. M. Tan
December 1993
Updated by S. Coen and H. Oudenhoven, September 2010
Updated for Python by R. Au-Yeung, 30 January 2015