LectNotebook14

March 8, 2021

0.1 Introduction

In this lecture, we will briefly discuss the calculation of the error of some "experimental" measurements and look at comparing different theoretical predictions to experimental data. We will also briefly revise some Monte Carlo methods.

0.2 The simple pendulum

Figure 1 shows a schematic diagram of a pendulum of length l with a bob of mass m. The bob is at an angle θ, and an arc-length s, from its equilibrium position. The combination of the gravitiational force, mg, and the tension in the supporting (massless) rod, T_s, ensure that the pendulum moves along the arc indicated in the figure. The restoring force, F, is the force that acts towards the pendulum's equilibrium position.

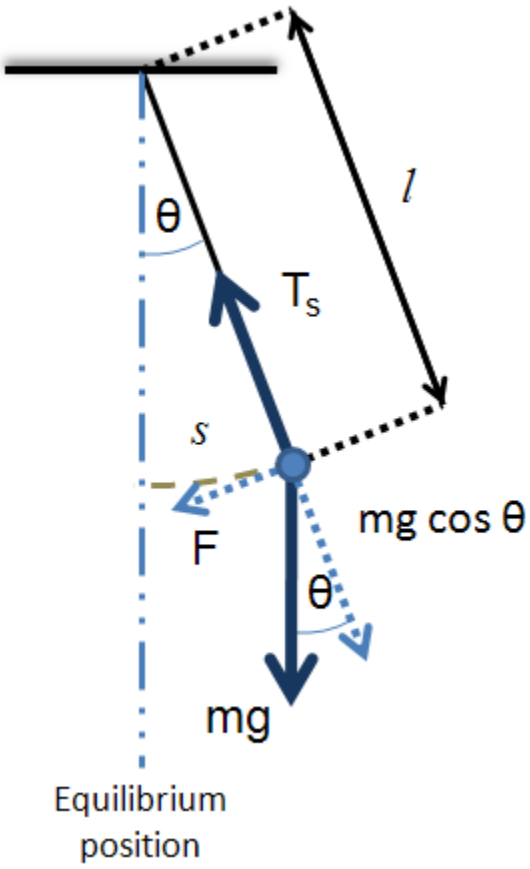


Figure 1: *Simple pendulum illustrating the forces acting on the bob and the definitions of the arc s and angle θ.*

The force F is given by:

$$F = -mg\sin\theta.$$

Note that F acts in the negative s direction in Figure 1.

Newton's second law states that $F = ma$, where $a = \frac{d^2s}{dt^2} = \ddot{s}$. Since $s = l\theta$, we have:

$$m\ddot{s} = ml\frac{d^2\theta}{dt^2} = ml\ddot{\theta}.$$

Equating this to the restoring force gives $-mg\sin\theta = ml\ddot{\theta}$. Hence:

$$\ddot{\theta} = -\frac{g}{l}\theta.$$

The acceleration of the bob is proportional to its displacement from equilibrium and directed towards its equilibrium position. This is an example of simple harmonic motion (SHM), a phenomenon that is encountered in many areas of physics. The motion of the bob can be described by the equation $\theta = \theta_0\sin(\omega t + \phi)$. Here, ω is the angular frequency of the bob's motion, θ_0 the amplitude and the value of ϕ (the phase constant) is determined by the *initial conditions* (the position and speed of the pendulum at $t = 0$).

The angular frequency is given by:

$$\omega = \sqrt{\frac{g}{l}},$$

and this in turn is related to the period of the pendulum's oscillations by:

$$T = \frac{2\pi}{\omega} \tag{1}$$

$$= 2\pi\sqrt{\frac{l}{g}}. \tag{2}$$

0.3 The pendulum at large amplitude

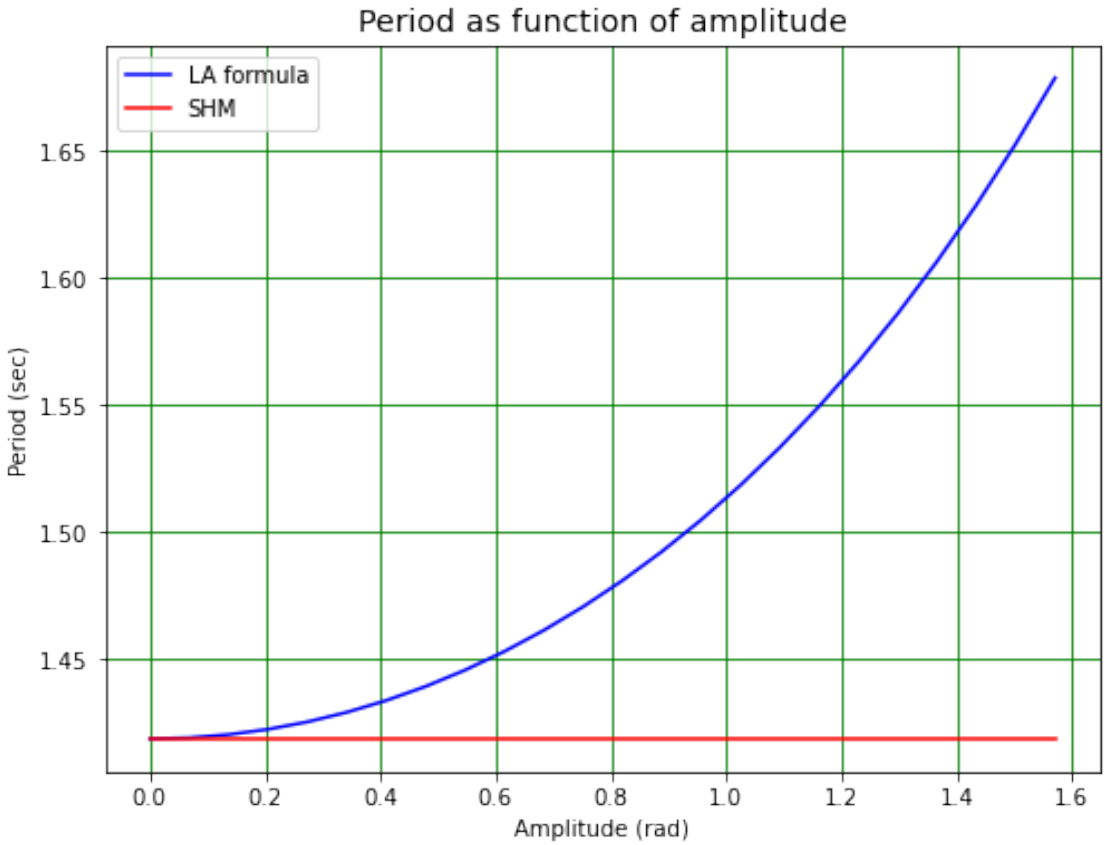
If large ampitude oscillations are considered, the pendulum is much more difficult to describe mathematically than is the case for small amplitude oscillations. In fact, there is no algebraic expression for T that is valid at large angles, and measurements can deviate noticeably from the predictions of the above simple harmonic motion (SHM) formula. This paper, by Lima and Arun, gives more accurate (but still approximate) results for the period at large amplitude. Their formula (referred to as the LA formula in the following) is:

$$T = -2\pi\sqrt{\frac{l}{g}}\frac{\log\left[\cos\frac{\theta_0}{2}\right]}{1-\cos\frac{\theta_0}{2}}.$$

Here, θ_0, the maximum displacement of the pendulum, is assumed to occur at time $t = 0$.

The SHM and LA expectations for the period of a pendulum of length 0.5 m are plotted below as a function of amplitude.

[8]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
#
def periodLima(L, thetaZero):
    #
    T = -2*np.pi*np.sqrt(L/g)*np.log(np.cos(thetaZero/2))/(1 - np.cos(thetaZero/
 ↪2))
    #
    return T
#
g = 9.81 # m/s**2
L = 0.5 # m
periodSHM = 2*np.pi*np.sqrt(L/g)
thetaBot = 0.0001
thetaTop = np.pi/2
nTheta = 24
thetaArr = np.linspace(thetaBot, thetaTop, nTheta)
periodArr = periodLima(L, thetaArr)
#
plt.figure(figsize = (8, 6))
plt.title('Period as function of amplitude', fontsize = 14)
plt.xlabel('Amplitude (rad)')
plt.ylabel('Period (sec)')
plt.plot(thetaArr, periodArr, label = "LA formula", color = 'b')
plt.plot(thetaArr, periodSHM*np.ones(nTheta), label = "SHM", color = 'r')
plt.grid(color = 'g')
plt.legend()
plt.show()
```



0.4 Large angle pendulum experimental study

We will look at data taken using a pendulum of length $l = 0.5$ m (see below for how the "measurements" were really made!). The data are a set of measurements of the period for a range of angular amplitudes. We will investigate whether a significant deviation from the "simple" form for T can be identified. The data are available as three files (which you can look at using MS excel):

- *thetaTable.csv* contains the values of the amplitude of the pendulum's oscillations in radians. This is just a single column of data.
- *periodTableSmall.csv* contains a table formed of 10 measurements of the period for each of the amplitudes in *thetaTable.csv*. The rows in *periodTableSmall.csv* correspond to the rows in *thetaTable.csv*. E.g. the third row (counting from zero!) contains the measured periods for the amplitude 0.394 radians.
- *periodTableLarge.csv* is the same as *periodTableSmall.csv*, except that there are 1000 measurements for each amplitude of the pendulum.

[2]:
```
!"C:\Program Files\Microsoft Office\root\Office16\EXCEL.EXE" thetaTable.csv
```

[3]:
```
!"C:\Program Files\Microsoft Office\root\Office16\EXCEL.EXE" periodTableSmall.
↪csv
```

[20]:
```
!"C:\Program Files\Microsoft Office\root\Office16\EXCEL.EXE" periodTableLarge.
↪csv
```

0.4.1 Low precision experiment

We will read in *thetaTable.csv* and *periodTableSmall.csv* in the following cell, then check that they have the expected numbers of rows and columns (using the `array.shape` syntax, which gives us access to the length of 1D arrays, the number of rows and columns for 2D arrays, etc.).

[4]:
```
thetaArr = np.loadtxt("thetaTable.csv", delimiter = ',')
periodArrSm = np.loadtxt("periodTableSmall.csv", delimiter = ',')
np.set_printoptions(precision = 3)
print(" ")
print("Shape of thetaArr =",thetaArr.shape)
print("Shape of periodArrSm =",periodArrSm.shape)
#
nTheta = thetaArr.shape[0]
nMeas = periodArrSm.shape[1]
print("nTheta =",nTheta,"and nMeas =",nMeas)
print(" ")
print("thetaArr =\n",thetaArr)
print(" ")
print("periodArr =\n",periodArrSm)
```

```
Shape of thetaArr = (16,)
Shape of periodArrSm = (16, 10)
nTheta = 16 and nMeas = 10

thetaArr =
 [0.1   0.198 0.296 0.394 0.492 0.59  0.688 0.786 0.884 0.982 1.081 1.179
 1.277 1.375 1.473 1.571]

periodArr =
 [[0.74  1.948 1.428 1.047 0.267 0.459 1.975 1.21  2.307 0.988]
 [0.245 1.052 1.342 0.865 1.266 1.068 1.198 2.143 2.007 1.532]
 [1.712 0.596 1.846 1.955 1.822 2.506 1.284 1.697 1.031 2.482]
 [1.814 1.699 1.983 1.732 1.197 1.097 1.029 1.918 1.248 1.145]
 [0.775 1.223 2.328 1.367 1.417 1.923 1.981 1.642 0.584 1.784]
 [1.175 0.909 1.561 1.183 1.462 1.965 2.016 1.817 1.524 1.964]
 [0.461 1.367 1.832 2.975 0.809 0.876 0.481 2.472 0.503 0.636]
 [1.323 2.343 1.59  0.603 1.438 1.878 0.221 2.295 2.265 1.842]
 [1.081 1.543 1.445 2.417 1.957 1.505 0.372 1.892 1.057 0.679]
 [1.584 1.162 2.027 1.687 1.059 1.396 1.569 1.545 1.846 2.014]
 [0.711 1.379 1.104 1.65  1.109 2.22  2.685 1.18  0.31  0.693]
 [2.005 2.217 2.207 2.148 1.354 1.833 1.446 0.971 1.828 1.522]
 [1.806 1.464 1.105 2.42  1.752 2.312 2.291 1.425 1.938 1.259]
 [2.017 2.035 0.789 1.668 1.344 2.083 2.442 0.876 1.244 1.148]]
```

```
[1.044 1.821 2.309 1.446 0.974 2.423 1.694 1.828 0.961 1.478]
[1.773 1.334 1.3   1.856 1.635 1.284 1.168 1.393 2.157 1.796]]
```

We now calculate the mean value of the period for each angle:

$$\overline{T} = \frac{1}{N}\sum_{n=1}^{N} T_n.$$

The *standard error* of the mean is given by:

$$\delta\overline{T} = \frac{1}{\sqrt{N}}\sqrt{\sum_{n=1}^{N}\frac{(T_n - \overline{T})^2}{N-1}}.$$

In this formula, the factor

$$s = \sqrt{\sum_{n=1}^{N}\frac{(T_n - \overline{T})^2}{N-1}}$$

describes the spread of T values we obtain from our experiment. It is the sample standard deviation of the T distribution. (While the precision with which we know the average of this distribution increases as we make more measurements by the factor $1/\sqrt{N}$, the value of s is a property of the apparatus and experimental procedure we are using; it doesn't change - within errors - with the number of measurements we make. We can only decrease s by using better apparatus or improving our experimental procedure.)

[5]:
```
#
# Calculate means for each amplitude
meanPeriods = np.mean(periodArrSm, axis = 1)
#
# Calculate sample standard deviation using numpy function, hence get standard␣
 ↪error of mean.
errPeriods = np.std(periodArrSm, axis = 1, ddof = 1)/np.sqrt(nMeas)
print(" ")
print("Number\t Amplitude (rad)\t Mean period (s)")
for n in range(0, nTheta):
    print(f"{n:d}\t {thetaArr[n]:.2f}\t\t\t {meanPeriods[n]:.2f} +-␣
 ↪{errPeriods[n]:.2f}")
```

```
Number   Amplitude (rad)         Mean period (s)
0        0.10                    1.24 +- 0.21
1        0.20                    1.27 +- 0.17
2        0.30                    1.69 +- 0.19
3        0.39                    1.49 +- 0.12
4        0.49                    1.50 +- 0.17
5        0.59                    1.56 +- 0.12
```

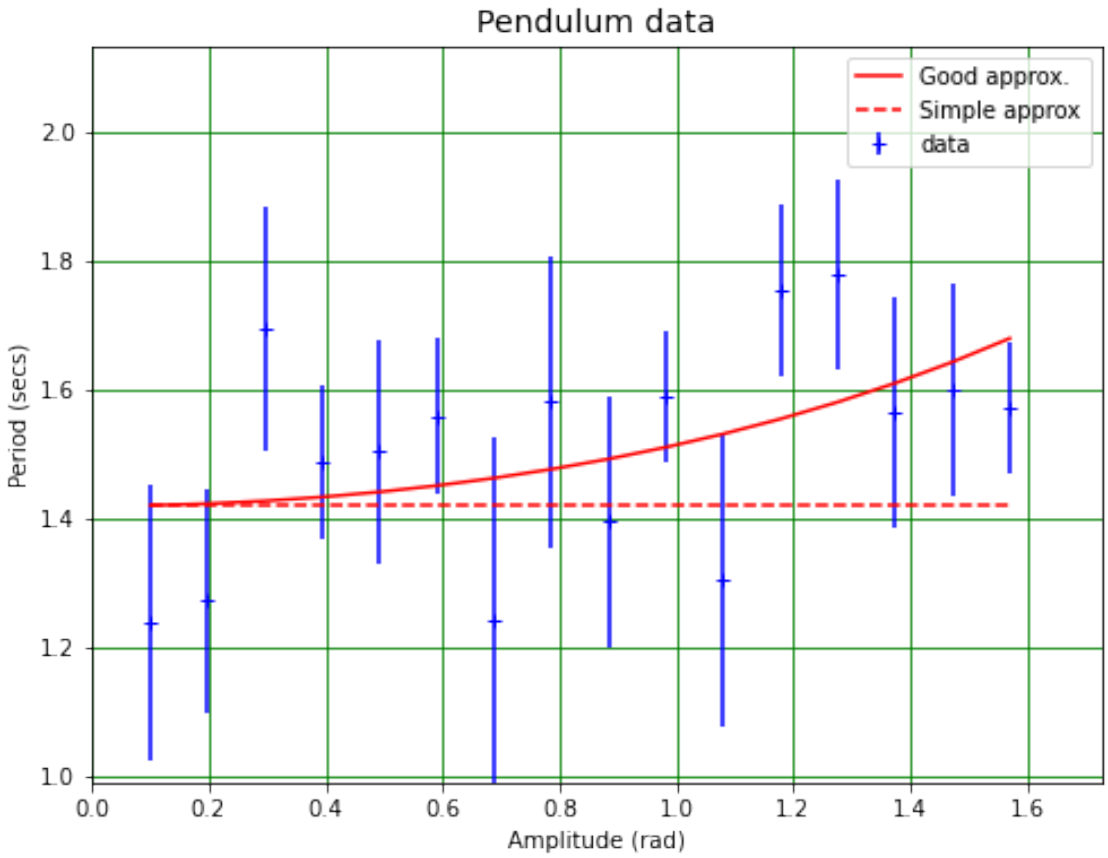
```
6       0.69                    1.24 +- 0.28
7       0.79                    1.58 +- 0.23
8       0.88                    1.39 +- 0.19
9       0.98                    1.59 +- 0.10
10      1.08                    1.30 +- 0.23
11      1.18                    1.75 +- 0.13
12      1.28                    1.78 +- 0.15
13      1.37                    1.56 +- 0.18
14      1.47                    1.60 +- 0.16
15      1.57                    1.57 +- 0.10
```

We now plot our measurements and compare them with the theoretical SHM and Lima curves.

[6]:
```
#
# Expected period and chi2 for SHM
ySHM = 2*np.pi*np.sqrt(L/g)*np.ones(nTheta)
chi2SHM = (ySHM - meanPeriods)**2/errPeriods**2
chi2sumSHM = np.sum(chi2SHM)
print(" ")
print(f"Chi2 for simple approximation = {chi2sumSHM:.2f}, chi2/NDF =␣
 ↪{chi2sumSHM/nTheta:.2f}")
#
# Expected period and chii2 for Lima formula
chi2 = (periodLima(L, thetaArr) - meanPeriods)**2/errPeriods**2
chi2sum = np.sum(chi2)
print(" ")
print(f"Chi2 for Lima/Arun approximation = {chi2sum:.2f}, chi2/NDF = {chi2sum/
 ↪nTheta:.2f}")
#
print(" ")
plt.figure(figsize = (8, 6))
plt.title('Pendulum data', fontsize = 14)
plt.xlabel('Amplitude (rad)')
plt.ylabel('Period (secs)')
plt.errorbar(thetaArr, meanPeriods, yerr = errPeriods,
             linestyle = '', marker = '+', color = 'b', label = "data")
plt.plot(thetaArr, periodLima(L, thetaArr), color = 'r', label = "Good approx.")
plt.plot(thetaArr, ySHM, color = 'r', linestyle = '--', label = "Simple approx")
plt.xlim(0.0, 1.1*thetaArr[nTheta - 1])
plt.ylim(0.8*np.min(meanPeriods), 1.2*np.max(meanPeriods))
plt.legend()
plt.grid(color = 'g')
```

```
Chi2 for simple approximation = 25.82, chi2/NDF = 1.61

Chi2 for Lima/Arun approximation = 12.62, chi2/NDF = 0.79
```



Does the difference between the two χ^2 values calculated above allow us to state that the pendulum is not following SHM? In this case, no, as acceptable χ^2/NDF values lie in the range 0.2 to 5.0 (some people use 0.25 to 4) so both of the above values are OK: we cannot say the pendulum is not obeying the SHM prediction.

0.4.2 Increase precision

Repeat the above calculation using the large dataset in the file *periodTableLarge.csv*. Can we now conclude that there is a deviation from SHM at large amplitudes?

[7]:
```
periodArrLg = np.loadtxt("periodTableLarge.csv", delimiter = ',')
print(" ")
print("Shape of periodArrLg =",periodArrLg.shape)
#
nMeas = periodArrLg.shape[1]
print("nTheta =",nTheta,"and nMeas =",nMeas)
#
meanPeriods = np.mean(periodArrLg, axis = 1)
errPeriods = np.std(periodArrLg, axis = 1, ddof = 1)/np.sqrt(nMeas)
print(" ")
```

```
print("Number\t Amplitude (rad)\t Mean period (s)")
for n in range(0, nTheta):
    print(f"{n:d}\t {thetaArr[n]:.2f}\t\t\t {meanPeriods[n]:.2f} +-␣
 ↪{errPeriods[n]:.2f}")#
g = 9.81
ySHM = 2*np.pi*np.sqrt(L/g)*np.ones(nTheta)
chi2SHM = (ySHM - meanPeriods)**2/errPeriods**2
chi2sumSHM = np.sum(chi2SHM)
print(" ")
print(f"Chi2 for simple approximation = {chi2sumSHM:.2f}, chi2/NDF =␣
 ↪{chi2sumSHM/nTheta:.2f}")
#
chi2full = (periodLima(L, thetaArr) - meanPeriods)**2/errPeriods**2
chi2sumFull = np.sum(chi2full)
print(" ")
print(f"Chi2 for Lima/Arun approximation = {chi2sumFull:.2f}, chi2/NDF =␣
 ↪{chi2sumFull/nTheta:.2f}")
#
print(" ")
plt.figure(figsize = (8, 6))
plt.title('Pendulum data', fontsize = 14)
plt.xlabel('Amplitude (rad)')
plt.ylabel('Period (secs)')
plt.errorbar(thetaArr, meanPeriods, yerr = errPeriods,
             linestyle = '', marker = '+', color = 'b', label = "data")
plt.plot(thetaArr, periodLima(L, thetaArr), color = 'r', label = "Good approx.")
plt.plot(thetaArr, ySHM, color = 'r', linestyle = '--', label = "Simple approx")
plt.xlim(0.0, 1.1*thetaArr[nTheta - 1])
plt.ylim(0.8*np.min(meanPeriods), 1.1*np.max(meanPeriods))
plt.legend()
plt.grid(color = 'g')
plt.savefig("periodDataLargeNew.png")
```

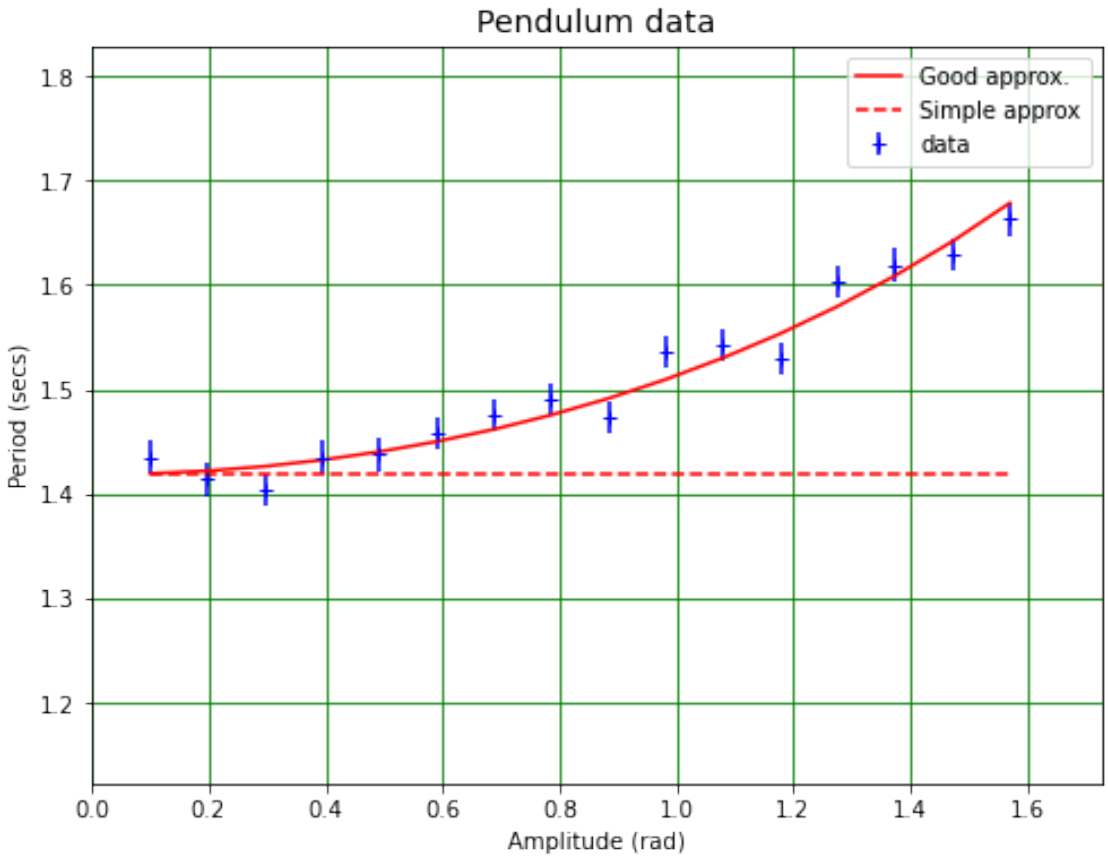
```
Shape of periodArrLg = (16, 1000)
nTheta = 16 and nMeas = 1000

Number   Amplitude (rad)         Mean period (s)
0        0.10                    1.44 +- 0.02
1        0.20                    1.41 +- 0.02
2        0.30                    1.40 +- 0.02
3        0.39                    1.43 +- 0.02
4        0.49                    1.44 +- 0.02
5        0.59                    1.46 +- 0.02
6        0.69                    1.48 +- 0.02
7        0.79                    1.49 +- 0.02
8        0.88                    1.47 +- 0.02
```

```
9        0.98                    1.54 +- 0.02
10       1.08                    1.54 +- 0.02
11       1.18                    1.53 +- 0.02
12       1.28                    1.60 +- 0.02
13       1.37                    1.62 +- 0.02
14       1.47                    1.63 +- 0.02
15       1.57                    1.66 +- 0.02

Chi2 for simple approximation = 930.34, chi2/NDF = 58.15

Chi2 for Lima/Arun approximation = 16.58, chi2/NDF = 1.04
```



The SHM formula results in a χ^2/NDF which is outside the range 0.2 to 5.0, so this formula does not describe the data. The χ^2/NDF value for the Lima/Arum formula is inside the acceptable range, so this does describe the data.

These results demonstrate the importance of precision in physics experiments!

0.5 Monte Carlo revisited

The "experimental data" used above wasn't actually a set of measurements of the motion of a pendulum. (It is actually quite difficult to make a good enough pendulum and measure its behaviour for large amplitudes!) It was generated using the code in the cell below.

[12]:
```
Debug = False
#
# Range of amplitudes (angles) considered
thetaBot = 0.1
thetaTop = np.pi/2
nTheta = 16
thetaArr = np.linspace(thetaBot, thetaTop, nTheta) # rad
#
# Pendulum length
L = 0.5 # m
#
# Assumed measurement precision
periodSigma = 0.5 # seconds
#
nMeas = 10
periodTabSm = np.zeros((nTheta, nMeas))
#
periodLimaValues = np.zeros(nTheta)
#
for n in range(0, nTheta):
    periodLimaValues[n] = periodLima(L, thetaArr[n])
    periodTabSm[n, 0:nMeas] = np.random.normal(periodLimaValues[n], scale =␣
 ↪periodSigma, size = nMeas) # seconds
if Debug: print(periodTab)
#
np.savetxt("thetaTable.csv", thetaArr, delimiter = ',', )
np.savetxt("periodTableSmall.csv", periodTabSm, delimiter = ',')
#
nMeas = 1000
periodTabLg = np.zeros((nTheta, nMeas))
#
for n in range(0, nTheta):
    periodTabLg[n, 0:nMeas] = np.random.normal(periodLimaValues[n],␣
 ↪periodSigma, nMeas) # seconds
if Debug: print(periodTab)
#
np.savetxt("periodTableLarge.csv", periodTabLg, delimiter = ',')
```

0.6 Summary

I hope that revising calculating means and their standard errors, fitting technoques (last week) and the calculation and use of χ^2 values is useful for your practical work!