Ojas Bardiya
UID: 505145284
CEE/MAE 20
07/18/20

# 1. Pendulum

## 1.1. Introduction

We analyze how the different kinematic quantities – Angular position, Angular Speed and acceleration change with time for a pendulum by treating its motion using 2 separate differential equations. In addition, we also determine the change in energy per unit mass of the pendulum with time. We use the Forward Euler and Semi-implicit Euler methods

We use the in-built MATLAB value for $\pi$ and consider the value of g = 9.81. Regarding the pendulum, we take the initial position as $\theta = \pi/3$ and the length L = 1.

We analyze these quantities for a time period of t = 20s with a time-step of dt = 0.005s.

## 1.2. Model and Theory

Forward Euler Method –

$\omega_{k+1} = -g/L*\sin(\theta_k)*dt + \omega_k$

$\theta_{k+1} = \omega_k*dt + \theta_k$

$\alpha_{k+1} = (\omega_{k+1} - \omega_k)/dt$

Semi-Implicit Euler Method –

$\omega_{k+1} = -g/L*\sin(\theta_k)*dt + \omega_k$

$\theta_{k+1} = \omega_{k+1}*dt + \theta_k$

$\alpha_{k+1} = (\omega_{k+1} - \omega_k)/dt$

Calculating Energy per unit mass –

$E_k = g*L(1 - \cos(\theta_k)) + 0.5*(L*\omega_k)^2$

where –

$\omega$ = Angular speed

$\theta$ = Angular position

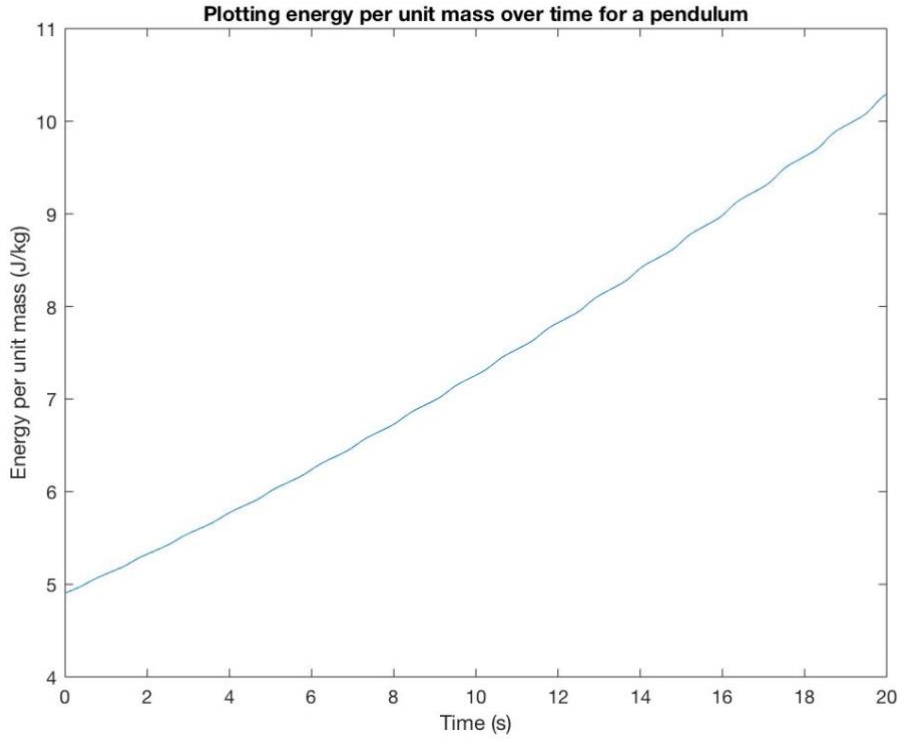
$\alpha$ = acceleration

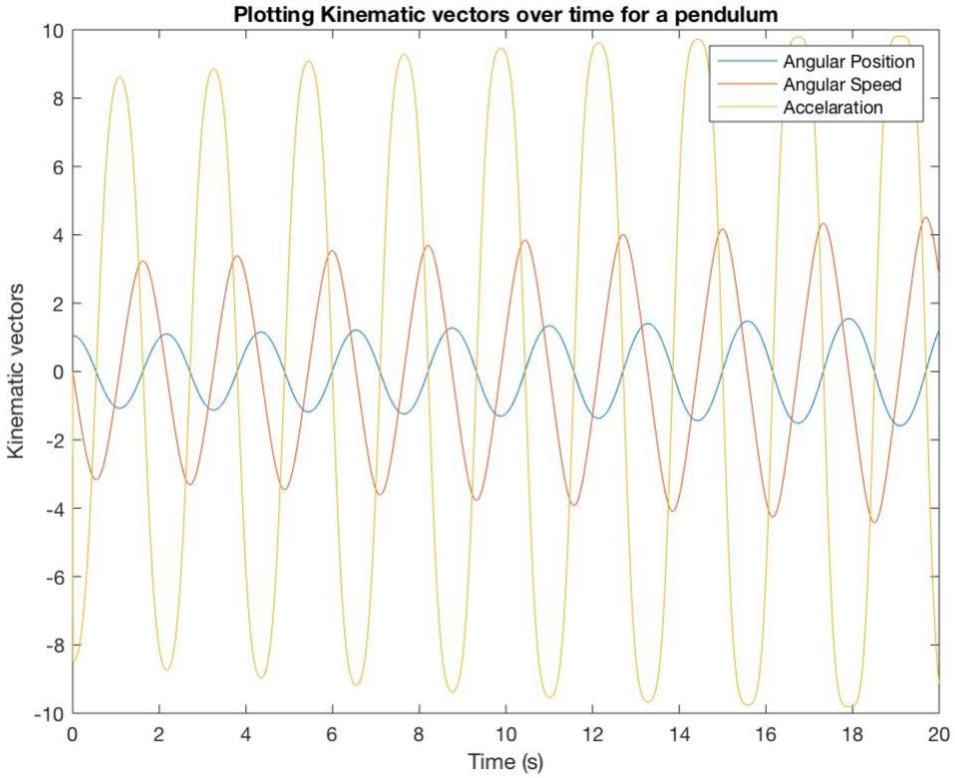
## 1.3. Methods and Pseudocode

*Set up the initial values of the differential equations*
*Set the time-step value*
*Initialize the arrays for angular position, angular speed and acceleration*
*Determine whether to use the Forward or Semi-implicit Euler method*
*Loop through the time period in the given time-steps*
    *If Forward Euler, use the corresponding discretized equations to update the necessary arrays*
    *If semi-implicit Euler, use the corresponding discretized equations to update the necessary arrays*
*Initialize an array and determine its values for Energy per unit mass using the arrays previously filled*
*Plot the different values accordingly*

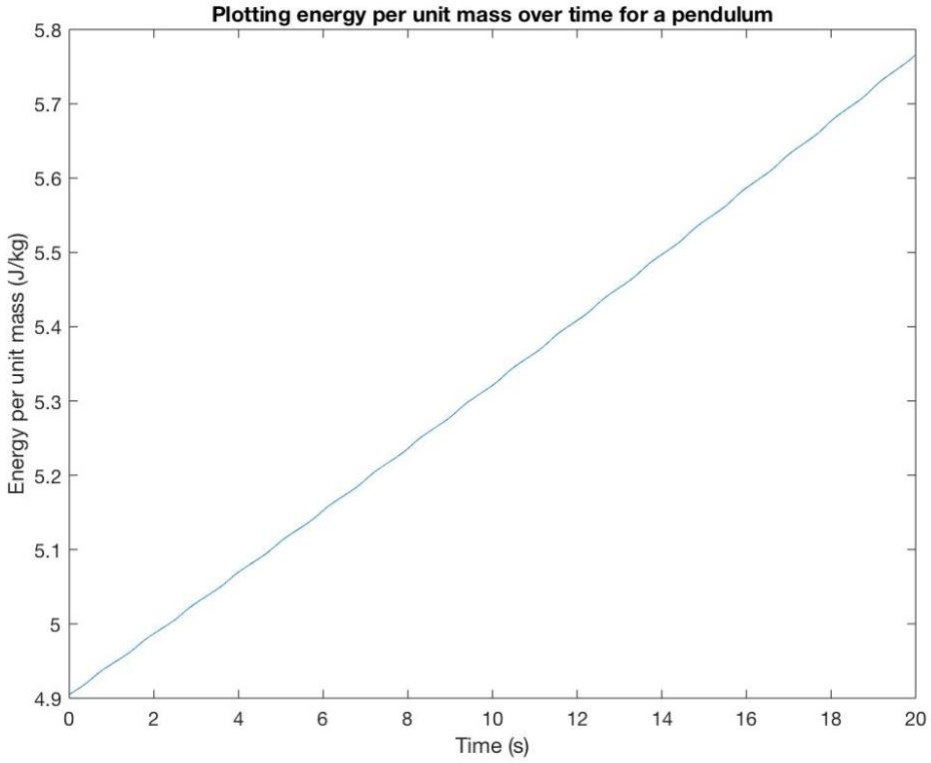
### 1.4. Calculations

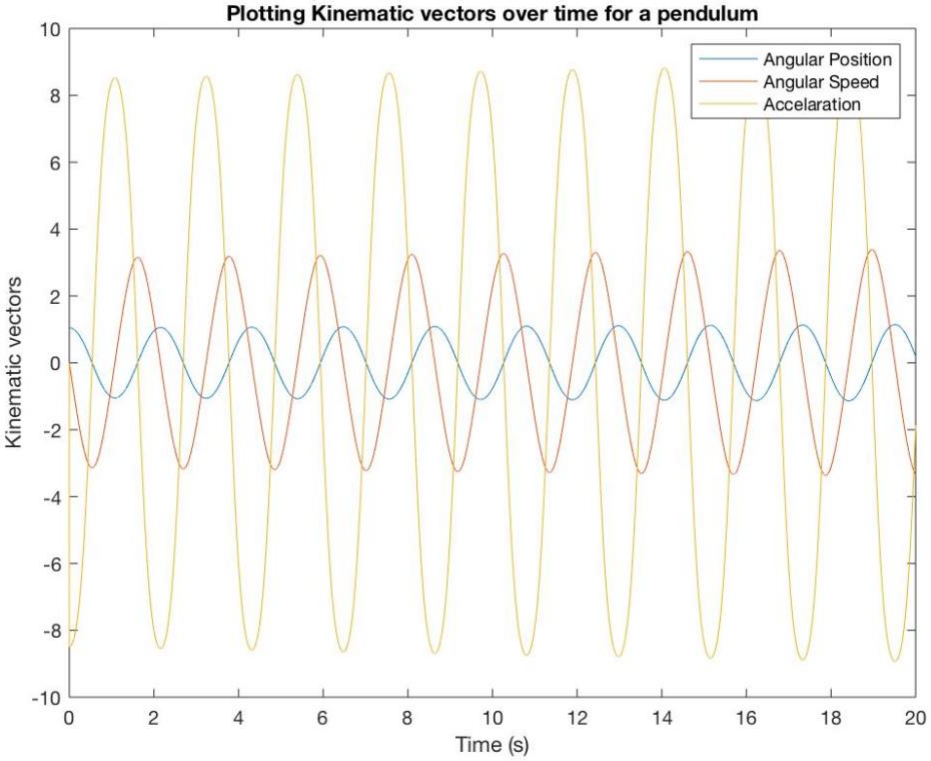
For Forward Euler, we observe that the system g**ains** energy over time, leading to the values of all kinematic quantities to oscillate over a larger range as the time increase.



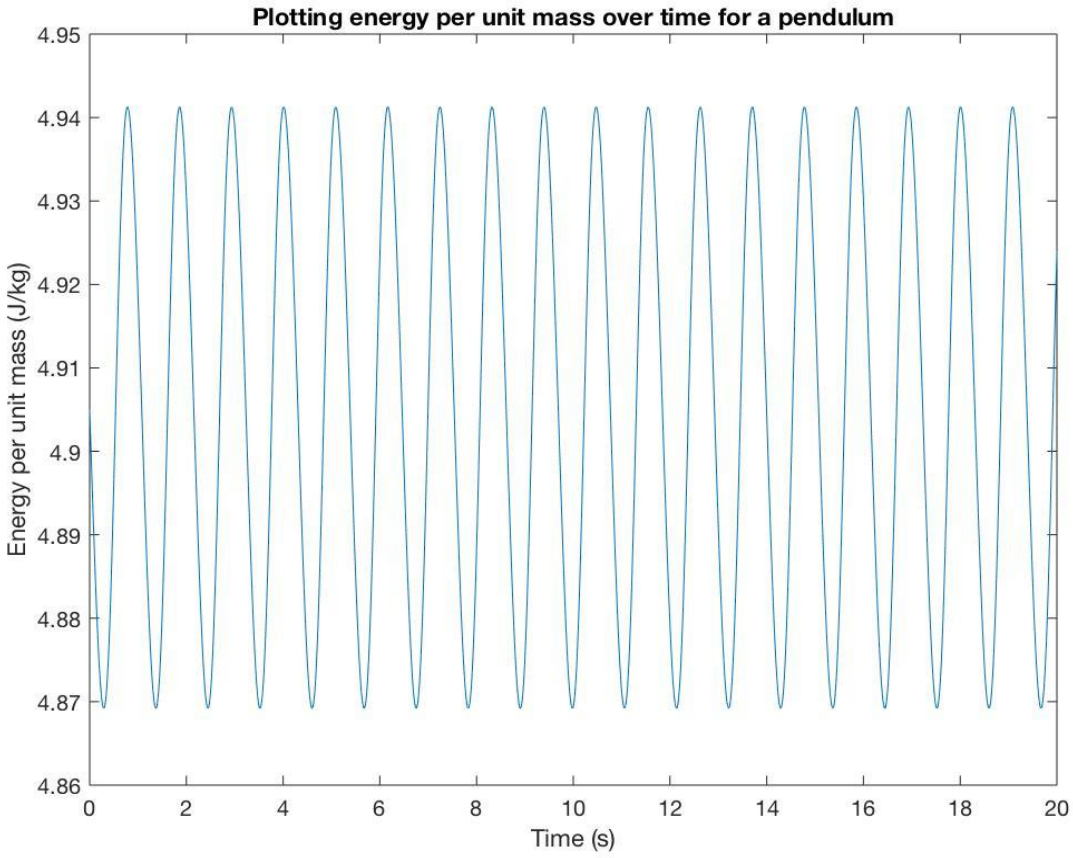



If we use a smaller time-step, say dt = 0.001s we see that the system **still gains energy but at a much lower rate**, causing the kinematic quantities to oscillate over a larger range but which grows much more slowly in comparison to a larger time-step.


Plotting energy per unit mass over time for a pendulum
Energy per unit mass (J/kg)
Time (s)
5.8
5.7
5.6
5.5
5.4
5.3
5.2
5.1
5
4.9
0
2
4
6
8
10
12
14
16
18
20


Plotting Kinematic vectors over time for a pendulum
Kinematic vectors
Time (s)
Angular Position
Angular Speed
Accelaration
10
8
6
4
2
0
-2
-4
-6
-8
-10
0
2
4
6
8
10
12
14
16
18
20

The semi-implicit method conserves energy to a far greater accuracy than the Forward Euler Method. As we see in the graph below, the total energy is confined to a certain range which decreases with the decrease in the value of the time-step.




In this case the energy is confined to a value roughly between 4.87-4.94, which is more or less constant. We can improve the accuracy by decreasing the value of the time-step

### 1.5. Conclusion

We can infer that the semi-implicit Euler method does a far better job of analyzing the change in the kinematic quantities and energy of a pendulum over a period of time than the forward Euler method.

## 2. DNA Analysis

### 2.1. Introduction

We analyze a chromosome stored in the form of a 1-D array and determine the number of protein-coding segments in it. We use the built-in mean, max and min functions in MATLAB to output the desired values.

## 2.2. Method and Theory

The four bases of DNA are
A – Adenine (1)
C – Cytosine (2)
G – Guanine (3)
T – Thymine (4)

A start codon consists of the bases ATG in that order. This can be represented as [ 1 4 3] in the form of a numeric array.
Similarly, the 3 types of stop codons are TAA, TAG, and TGA which are represented as [ 4 1 1], [ 4 1 3] and [ 4 3 1] in the form of a numeric array respectively. We determine the length of a protein-coding segment using this approach.

## 2.3. Pseudocode

*Load the file*
*Set the count for all the stop codons to zero*
*Initialize the start point as 0 (false)*
*Create an array to hold the value of the segment lengths*
*Set the current index of the array to 0*
*Loop through the bases of the chromosome using a step of 3*
    *Check if the start is zero*
        *If so, set the start as the current iteration*
    *Else*
        *Check if any of the stop codons are encountered*
            *If so, increment the count of the corresponding stop codon by 1*
            *Determine the segment length*
            *Increment the current index by 1*
            *Set the value stored in the current index to the segment length*
            *Reset the start point to zero*
*Output the desired values*

Note: The segment length is determined using –
segment_length = k (current iteration value) – start + 3

## 2.4. Calculations

We get the following result –

Total protein coding segments: 4339
Average length: 87.69
Maximum length: 1797
Minimum length: 6

The percentage of DNA directly used in the protein coding process is **30.14 %** which is determined using the formula –

Percentage = $\frac{\text{sum(total length of protein coding segments)}}{\text{total number of bases}} * 100$

The most frequently used stop codon is TGA with a total of **1897** occurrences and the least frequent is TAG with a total of **980** occurrences.

We could extend our algorithm to account for a single mutation in a codon by using logical arrays. If 2 out of 3 bases match for a given stop codon, we can create a series of conditional statements that decide when to exit the current part of the script.

### 2.5. Conclusion

We observe that roughly only 1/3$^{rd}$ of the DNA is directly used in the protein-coding process and the stop codons occur in varying frequencies which could be a result of mutations over time.