Simulating a four bar linkage

A four bar linkage is a simple mechanism formed by four bars connected at the their ends by pins. These connections are such that a closed chain is formed by the bars. Given the simplicity of this mechanism it is an ideal candidate to perform a computer simulation. We chose to embark on such a project. To accomplish this task it was separated into four steps:

1. An analysis of the geometry of the mechanism.
2. Creating code that interprets the geometry and animates the mechanism.
3. An analysis of the forces and accelerations felt by the bars.
4. Incorporating the dynamical analysis into the animation.

After these steps were completed, different bar lengths and masses were experimented with to understand their influence on the motion of the mechanism. These conclusions and the details of each step are presented bellow.

Step 1. **Analysis of the geometry:**
For the sake of simplicity each bar was considered to be a line and the pins that connected them were assumed to not occupy any space. This meant that the entire geometry of the mechanism could be described in terms of the lengths of the idealized bars. However, the pins allowed the bars to rotate and some angles needed to be defined to account for this motion. We chose these angles θ and α as the ones measured between the horizontal and two of the bars.

We continued by placing the bars on a Cartesian plane and choose seemingly advantageous coordinates for them. The bar defined as ***A*** was held vertical with end at the origin. The bar ***r*** was placed with one end at the origin and allowed to rotate. Bar ***R*** was placed with one end at the top of bar **A** and allowed to rotate there. Bar ***C*** was attached to the free ends of bar ***r*** and ***R***. These bars, which could be conveniently described as vectors, are shown in Figure 1.

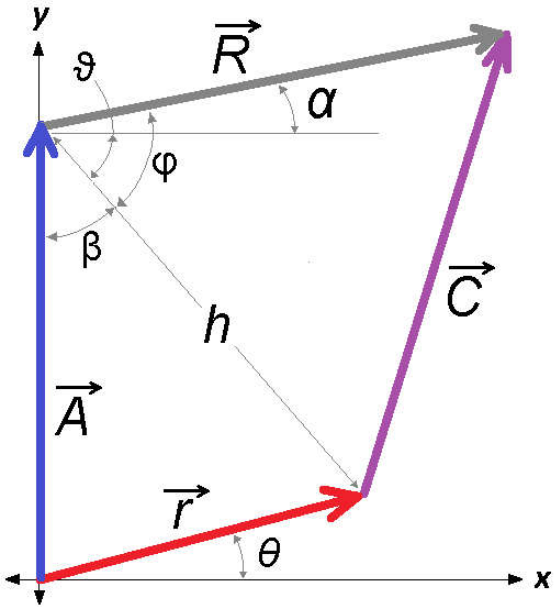

Figure 1. Vector diagram of bars

From this diagram the following length relationships were established:

$$h^2 = r^2 + A^2 - 2rA\sin(\theta)$$

$$C^2 = h^2 + R^2 - 2hR\cos(\varphi)$$

The angles were related through these equalities:

$$\tan(\beta) = \frac{\cos(\theta) r}{A - r\sin(\theta)}$$

$$\varphi = \alpha + \vartheta$$

$$\vartheta = \frac{\pi}{2} + \beta$$

Hence the angle of bar ***R*** could be defined as:

$$\alpha = atan\left(\frac{\cos(\theta) r}{A - r\sin(\theta)}\right) - asin\left(\frac{R^2 - C^2 + h^2}{2Rh}\right)$$

As *α* could be defined explicitly as a function of *θ* it meant the position and orientation of every bar in the mechanism could be described in terms of one variable. That is to say, the entire system possessed only one degree of freedom.

Step 2. **Interpreting the geometry and animating the mechanism:**
Python was chosen as the programming language to write the code in. This was advantageous for two reasons: it has packages with functions that can solve *systems* of differential equations (which would be needed to solve any equations of motion) and it has packages dedicated to animation. This was of great aid as it meant that preexisting functions could be used and combined to generate a desired animation. Of these, the *matplotlib* package included functions capable of generating plots and on these plots manipulating objects called *patches* through the function *FuncAnimation*.

The *patch* objects were a list of a basic shapes that could have simple parameters such as their size, position and orientation modified through functions associated with their class. Hence so long as these angles and coordinates were known they could easily be assigned to a corresponding *patch* and updated when commanded to.

In addition to this, *FuncAnimation* required that another function, the *animate* function, be define by the user. This custom function had to receive an index value and use it to modify objects or parameters that could be returned to *FuncAnimation,* which would in term use these outputs to update a plot at know time intervals (hence generating a motion picture).

With this information the code was structured as follows:

- We defined *patches* of shapes that could be used represent the bars and pins of the four bar linkage. Circles were used to represent the pins while thick lines were used to represent the bars.
- A function was defined to calculate the angle *α* as a function of the input angle *θ.*

- Coordinates were defined for the position the pins as functions of θ, and these were in term used to define vectors for each of the bars, which were themselves used to draw the line *patches*.
- As every point or coordinate used by the *patches* was dependent on the input angle θ, the *animate* function was defined to use the index value to update θ and use it to update the parameters of the *patches*. *Animate* then returned these updated objects.
- *FuncAnimate* then used this definition of the *Animate* function to generate an animation of the mechanism.

Step 3. **Analysis of the forces and moments on the bars:**
As the mechanism had many joints it was soon realized that direct application of Newton's second law to each bar would prove troublesome. Each pin generated two reaction forces thus each bar felt four unknown forces as they were attached to two pins. Additionally each bar had three accelerations that were also not known. A system of equations had arisen and it was too complex to be solved by hand without error. To circumvent this impediment a different approach was taken.

Instead of focusing on every force in the system, only those that could cause motion were necessary to understand how the system moved. That is to say, only forces that induced a *displacement* of the bars were relevant. Said again, this meant that only forces that did *work* on a system needed to be considered. This train of thought was expanded upon by exploring *D'Alembert's principle* and his inclusion of *inertial forces* and their effects on the *virtual* work of a system. This method is best suited when the motions of bodies are known precisely, which was exactly our case as the position of every bar was known through θ.

For the sake of simplicity we assumed that only bar ***r*** and ***R*** had mass. We then considered a small imaginary displacement of the mechanism and the inertial moments acting on each bar. We also included an external torque *T* acting on bar ***r***. This is seen in Figure 2. Note that both bar ***r*** and ***R*** undergo rotation about a fixed axis, as such only their angle needs to be considered to completely describe their motion.

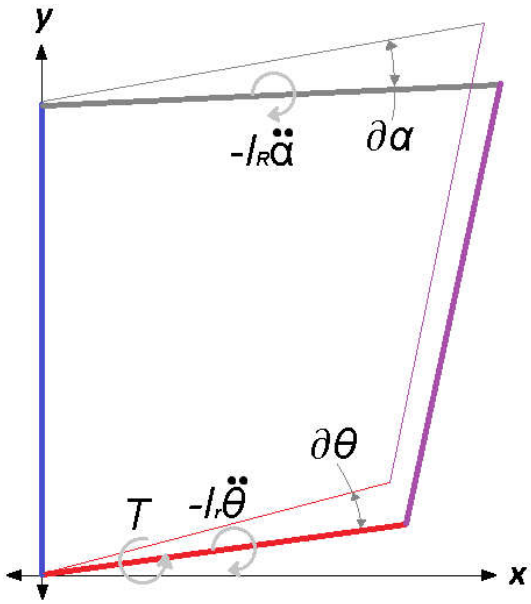

Figure 2. Moments and virtual displacements of the mechanism.

Given that the sum of the virtual works is zero, we can write:

$$\partial\alpha(-I_R\ddot{\alpha})+\partial\theta(-I_r\ddot{\theta})+\partial\theta(T)=0$$

We note that α is a function of θ as such:

$$\frac{\partial\alpha}{\partial\theta}=\alpha^{[1]} \quad \rightarrow \quad \partial\alpha=\partial\theta\,\alpha^{[1]}$$

Substituting: $\partial\theta\,\alpha^{[1]}(-I_R\ddot{\alpha})+\partial\theta(-I_r\ddot{\theta})+\partial\theta(T)=0$

By the chain rule we have that:

$$\ddot{\alpha}=\frac{\partial^2\alpha}{\partial t^2}=\frac{\partial^2\alpha}{\partial\theta^2}\left(\frac{\partial\theta}{\partial t}\right)^2+\frac{\partial\alpha}{\partial\theta}\frac{\partial^2\theta}{\partial t^2} \quad \rightarrow \quad \ddot{\alpha}=\alpha^{[2]}(\dot{\theta})^2+\alpha^{[1]}\ddot{\theta}$$

Substituting once again and simplifying:

$$-\partial\theta\left(I_R\alpha^{[1]}(\alpha^{[2]}(\dot{\theta})^2+\alpha^{[1]}\ddot{\theta})\right)-\partial\theta(I_r\ddot{\theta})+\partial(\theta T)=0$$

$$-\left(I_R(\alpha^{[1]})^2+I_r\right)\ddot{\theta}-I_R\alpha^{[1]}\alpha^{[2]}(\dot{\theta})^2+T=0$$

$$\ddot{\theta}=\frac{\left(T-I_R\alpha^{[1]}\alpha^{[2]}(\dot{\theta})^2\right)}{\left(I_R(\alpha^{[1]})^2+I_r\right)}$$

With an explicit definition of the angular acceleration of bar ***r***, it was now possible to find a numerical solution for θ with respect to time.

Step 4. **Incorporating the dynamical analysis into the animation:**
The equation defining $\ddot{\theta}$ included the derivatives of α with respect to θ. There were two choices to calculate these functions: Either perform the derivation by hand and find an exact solution, or *approximate* the derivatives using *finite differences*.

By inspecting the definition of α one could see that its first derivative would involve the sum of inverse trigonometric functions whose inputs are quotients of products. This would undoubtedly lead to a complicated result and the second derivative would only be more complicated. Attempting to perform these computations by hand would be very error prone unless done with great care. By comparison, a finite difference could make use the already coded function $\alpha(\theta)$ and the results, while only approximations, were likely to be more accurate than a wrongly performed hand computation. These derivatives were then approximated as:

$$\alpha^{[1]}\approx\frac{\alpha(\theta+d\theta)-\alpha(\theta-d\theta)}{2d\theta}$$

$$\alpha^{[2]}\approx\frac{\alpha(\theta+d\theta)-2\alpha(\theta)+\alpha(\theta-d\theta)}{d\theta^2}$$

Where $d\theta$ was an arbitrary small value. In our case taken as 10^{-5} .

Now all that was required was to find a solution for θ with respect to time. To do this, the derivative of a state vector was coded as a function whose inputs were the state vector itself and a parameter proportional to time. The required calculations were performed over an arbitrary time interval that was subdivided into an arbitrary number of steps. The computations themselves were done with the function *ideint()* included in the package *numpy*.

The output of the solver was a 2D array with the values of θ and the time parameter. It was only a simple task now to access the values of θ using the index from the *animate* function. We were careful to make sure the number of frames in the animation were equal to the number of time steps used to calculate the angle. This ensured that each frame could access an existing value and that the program did not attempt to draw more frames than there existed data.

At this point the program was complete.

Results and conclusions:

We chose to perform a numerical experiment by comparing the results of different configurations of bar lengths and moments of inertia. For every test, bar ***r*** (which we will refer to as the "crank") was given the same initial angular velocity. Each configuration is shown in Table 1.

Configuration	Light Crank-Short Bars	Heavy Crank-Short bars	Light Crank-Long bars
Moment of Inertia			
r	1.0	10.0	1.0
R	10.0	1.0	10.0
Bar length			
r	0.5	0.5	0.5
R	0.8	0.8	0.8
C	0.8	0.8	3.1
A	0.7	0.7	3.0

Table 1. Tested configurations of the four bar linkage.

A light crank combined with short bars lead to very jerky motion. As the crank rotated it abruptly and drastically changed angular velocity. This can be interpreted as the kinetic energy stored in the oscillating bar ***R*** being transferred to the lighter crank, which necessarily had to spin faster to keep the total energy of the system constant. This is shown in Figure 3 and Figure 4.

Conversely a heavy crank with short bars lead to the crank rotating smoother. The angular velocity still had abrupt changes but these were less than with a light crank. This is sensible as the crank stored most of the kinetic energy of the system and bar ***R***, due to its low mass, only stored a small amount. See Figure 5 and Figure 6 for this second result.

Finally, the third configuration with a light crank and long bars had a similar motion to a heavy crank. In this case, the motion was smoother and more symmetric than a light crank with short bars. This meant that both the geometry and the mass distribution affected the motion of the bars. Figure 7 and Figure 8 show this third result.

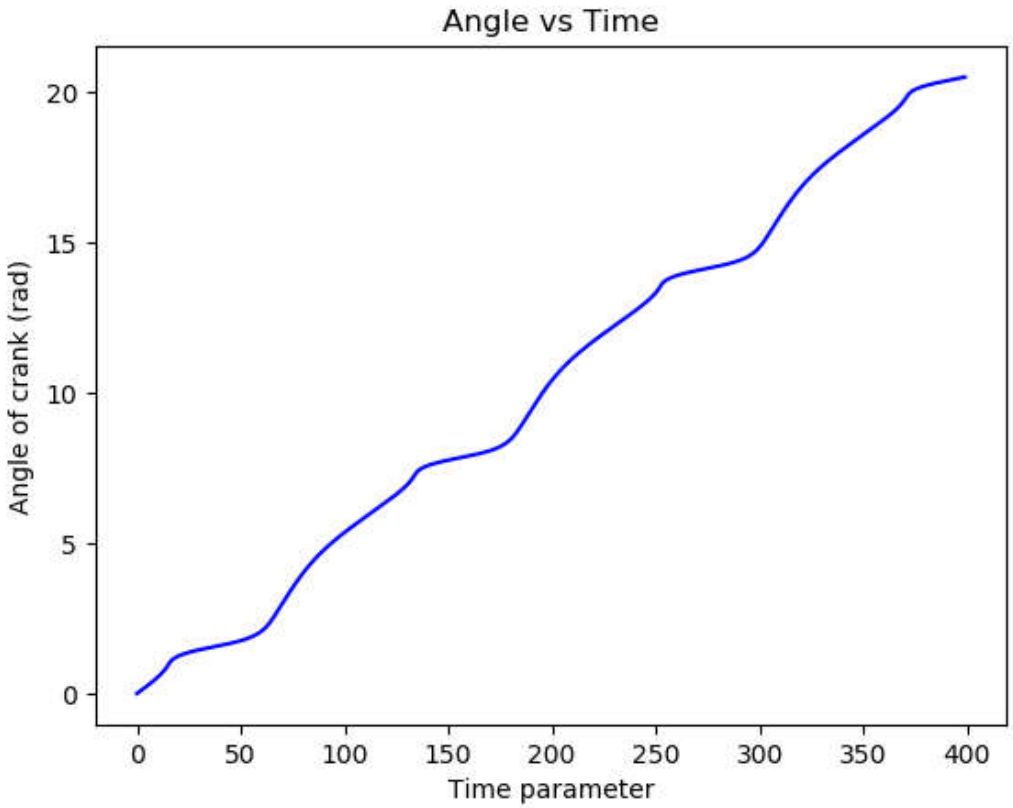

Figure 3. Angle of Light crank with short bars.

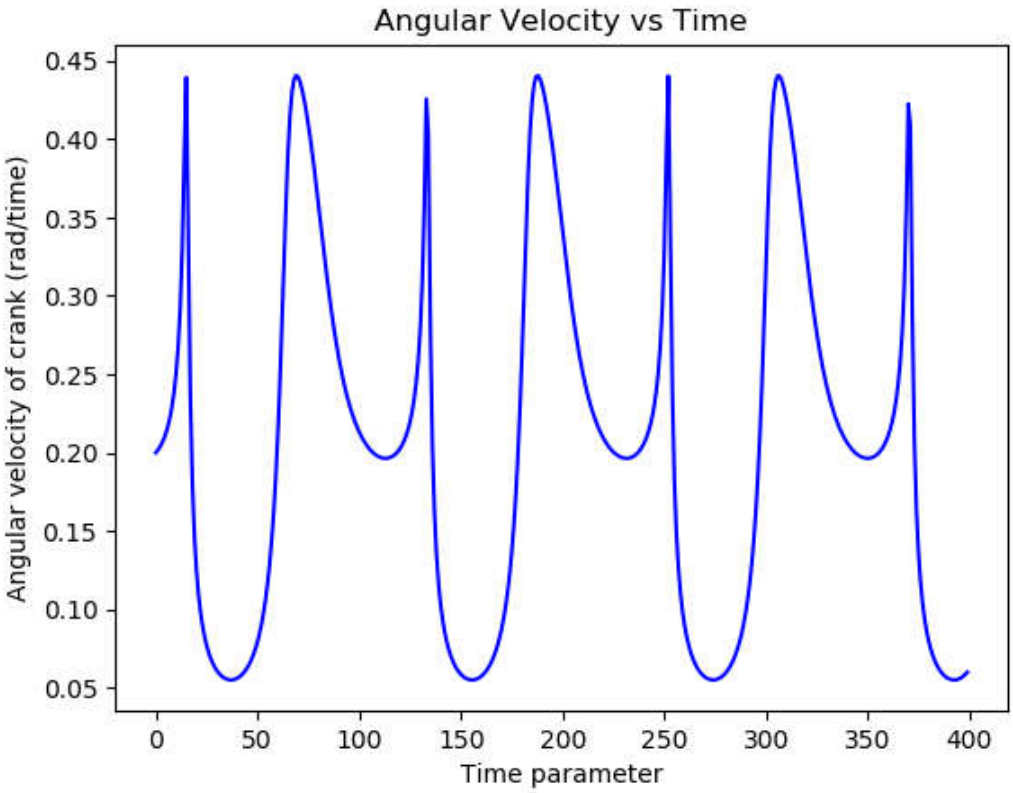

Figure 4. Angular velocity of light crank with short bars.

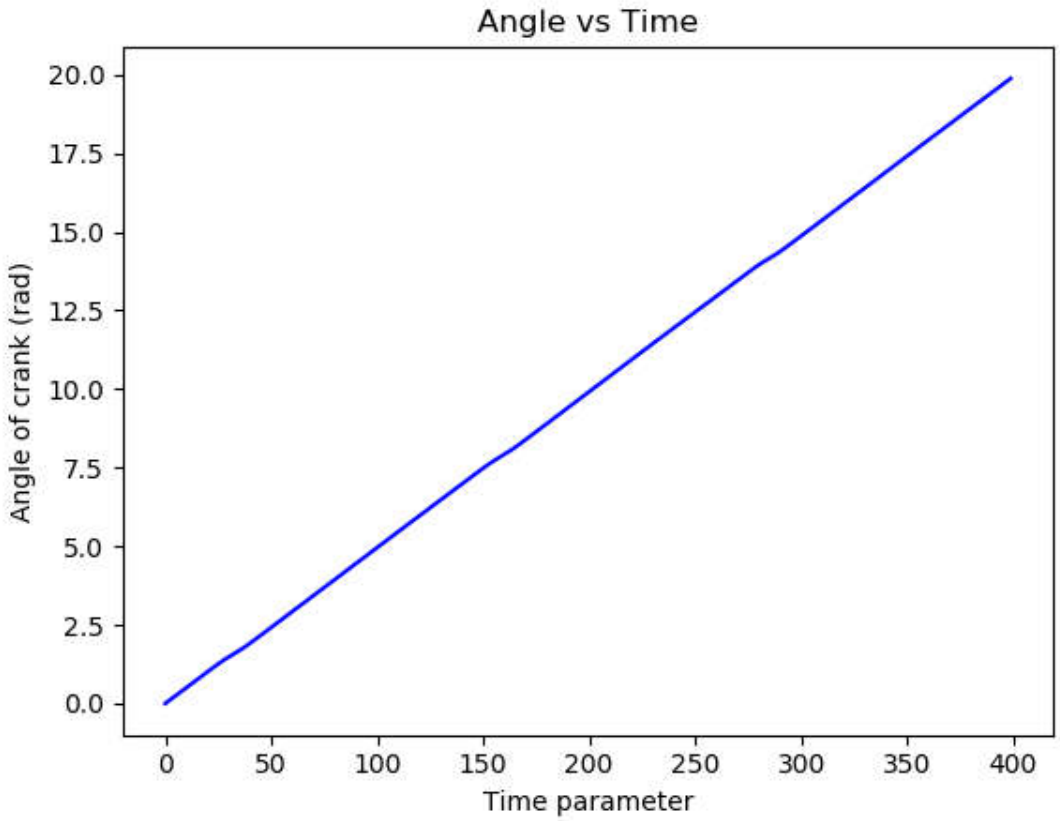

Figure 5. Angle of heavy crank with short bars.

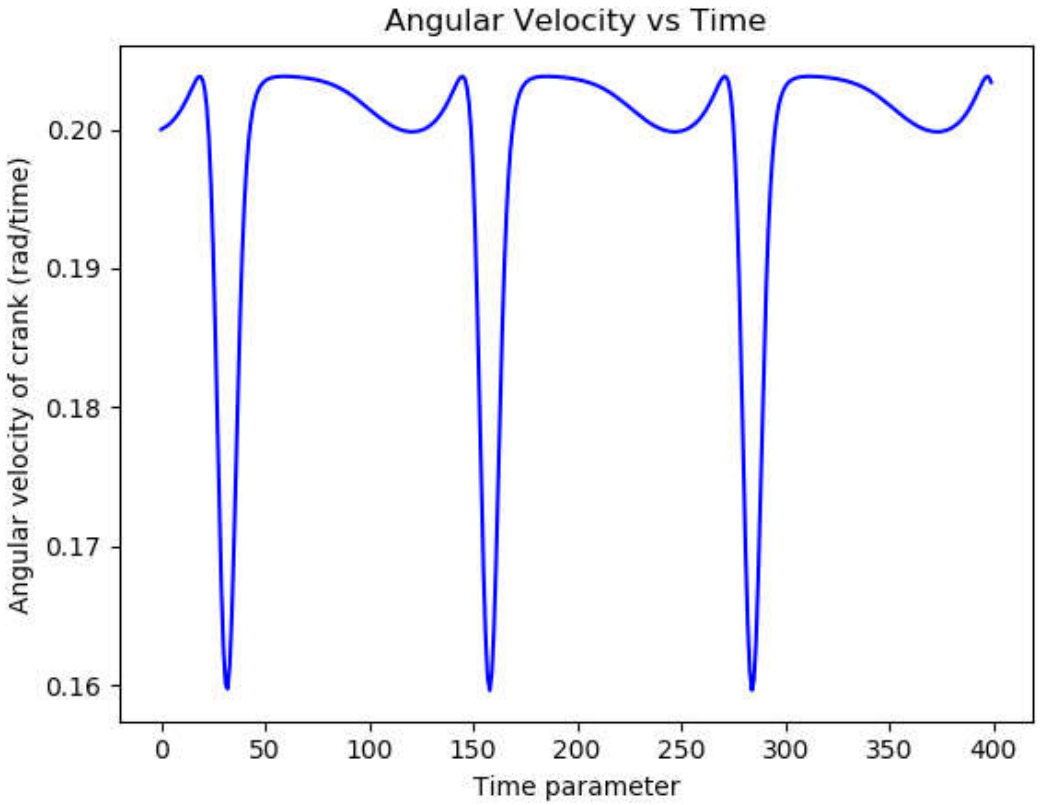

Figure 6. Angular velocity of heavy crank with short bars.

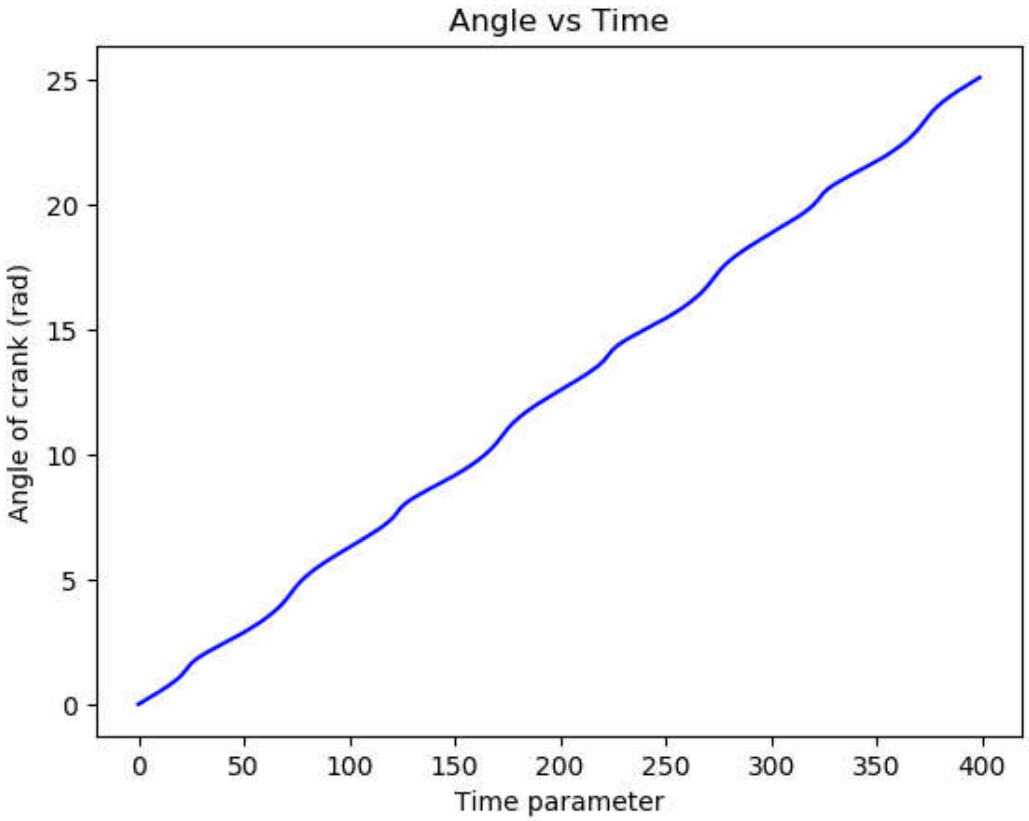

Figure 7. Angle of light crank and long bars.

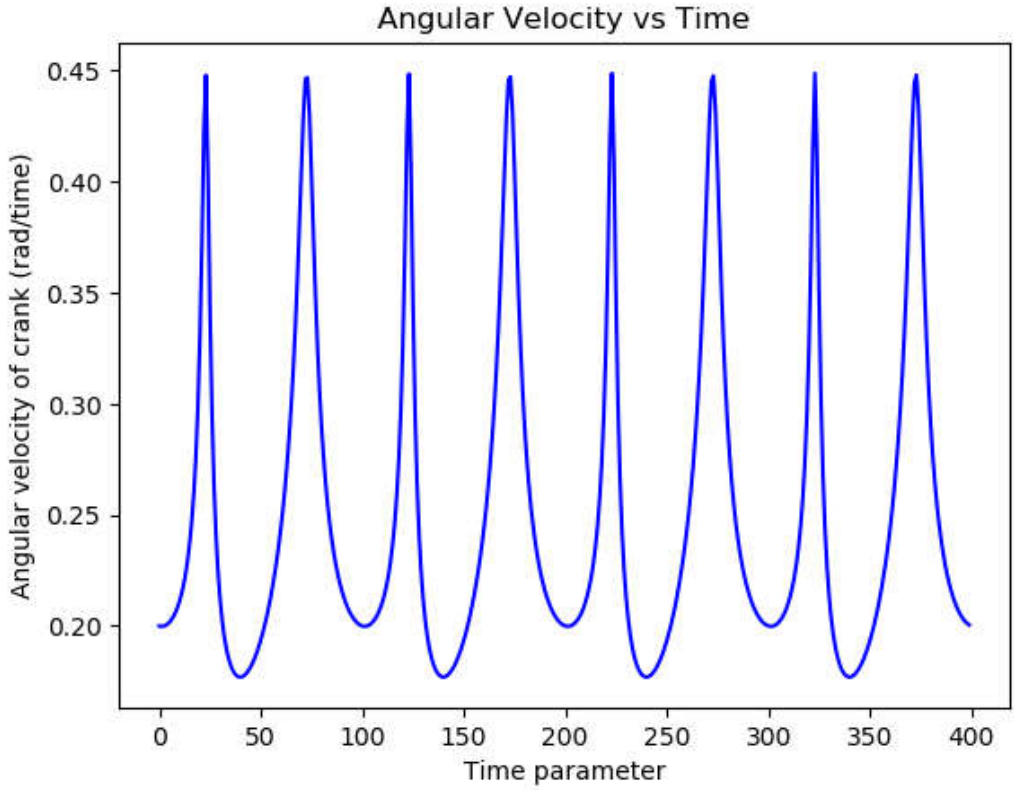

Figure 8. Angular velocity of light crank and long bars.