**KINEMATIC SILMULATOR**

**IMPLEMENTING THE CONCEPTS OF KINEMATICS TO VISUALIZE AND SIMUALTE LINKAGES AND ANALYZE ITS MOTION USING PYTHON**

Implementation of python to simulate closed-loop kinematic linkage mechanism and analyze its motion parameters

**1. INTRODUCTION**

Kinematic is the study of how objects move without focusing much on the forces acting on them. It is a branch of mechanics that is most important to design any mechanism or machine components that undergo a series of motions relative to each other. Although creating a mechanism involves many equations involved during it, it becomes much more complex if the movement of one part affects the other. During this situation, the equations get combined and should be solved simultaneously to get the design parameters. This paper mainly deals with designing a closed-loop kinematic chain mechanism that involves designing kinematic linkage parameters that create the required motion. For example, a simple slider-cranker mechanism is designed to effectively convert linear motion to circular motion as in the case of IC engines or circular motion to reciprocating motion as in the case of reciprocating pump.

Coming to the designing part of these mechanisms, there are many types of mechanisms such as:

* Four bar mechanism
  + Crank-rocker mechanism
  + Drag link mechanism
  + Double crank mechanism
  + Double rocker mechanism
* Slider cranker mechanism
  + Rotary engine – I inversion of slider-crank mechanism
  + Whitworth quick return motion mechanism–I inversion of slider-crank mechanism
  + Oscillating cylinder engine–II inversion of slider-crank mechanism (connecting rod fixed)
  + Pendulum pump or bull engine–III inversion of slider-crank mechanism (slider fixed)
  + Scotch –Yoke mechanism
  + Elliptical trammel

These are some of the real-world applications of mechanisms used to achieve the required motion and speed. Designing these mechanisms required careful calculation and visualization of these mechanisms using some helpful tool, and there are much advanced software available to do the same, but this work revolves around how to make similar tool using python and customizing it based on the user using this tool

This paper deals with the kinematic linkage problems and about its visualization. The concepts used here are well available in any mechanics of machinery text books or related to it. This is a solver with simulator which enables the user to use it to visualize the motion and its kinematics with good accuracy.

This simulator has been divided into backend and frontend part where in each part is explained below along with the simulator image.

In the backend part the major calculation are done by using concepts of kinematics to solve position, velocity,acceleration of the linkages in motion.

* The equations used to solve the problems are derived by considering a single closed loop, where there can be maximum of 2 unknowns for each loop. This idea made it feasible to cover all the situation to solve the problem and simulate it.
* The equation uses vector addition properties where the unknow values of vectors can be anywhere along the links of a closed loop, but by solving the equation considering the unknows are of first or first two links greatly simplifies the problem.
* Due to this reason all the unknown values are considered to be of either first or second link denoted by subscript 1 or 2 for the respective variables.
* The program first calculates all the position variables first then use them to calculate speed related variables and use all of the results to calculate the acceleration variables.
* Considering all of the above points, the concepts used and the equations implemented are discussed below considering every condition.

**POSITION ANALYSIS**

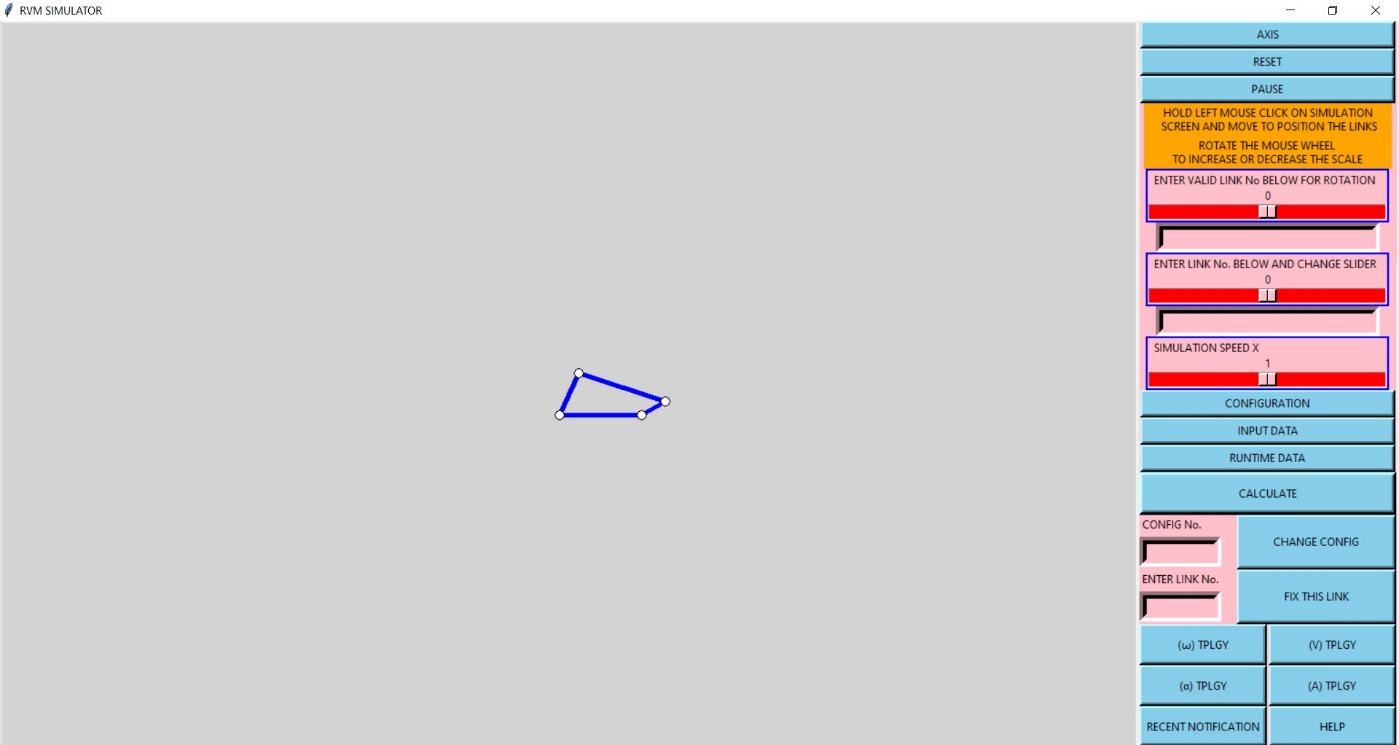
**NOTE: FOR THE GIVEN CONDITION, THERE CAN BE MORE THAN ONE SOLUTION CORRESPONDING TO DIFFERENT CONFIGURATION. ALL THOSE CONDITIONS ARE COVERED IN THE SIMULATOR.**

**VELOCITY ANALYSIS**

**ACCELERATION ANALYSIS**

* After all of the above three analysis all the values for the given configuration is obtained. After this the points undergo series of algorithm to be converted into plottable form on the screen.
* First step is where the links get there proper order from head to tail fashion. This is where the **configuration**  option is useful, where the algorithm correctly organizes the links with their length and angle along with their neighbouring links from head to tail manner.
* All this above mentioned algorithm takes place for only **one loop**. To solve multiple loops those algorithms are run multiple times keeping the previously solved values in memory and comparing them to avoid same answer. To get all those configurations simulated at once is not possible due to which there as an option to enter the configuration number that will be simulated.
* These are the main backend calculations that majorly take care of all the simulations. The remaining options for the simulations are taken care by the frontend section, where all the runtime manipulations are done.

**FEATURES OF THE SIMULATOR**

****

**FIRST OVERVIEW**

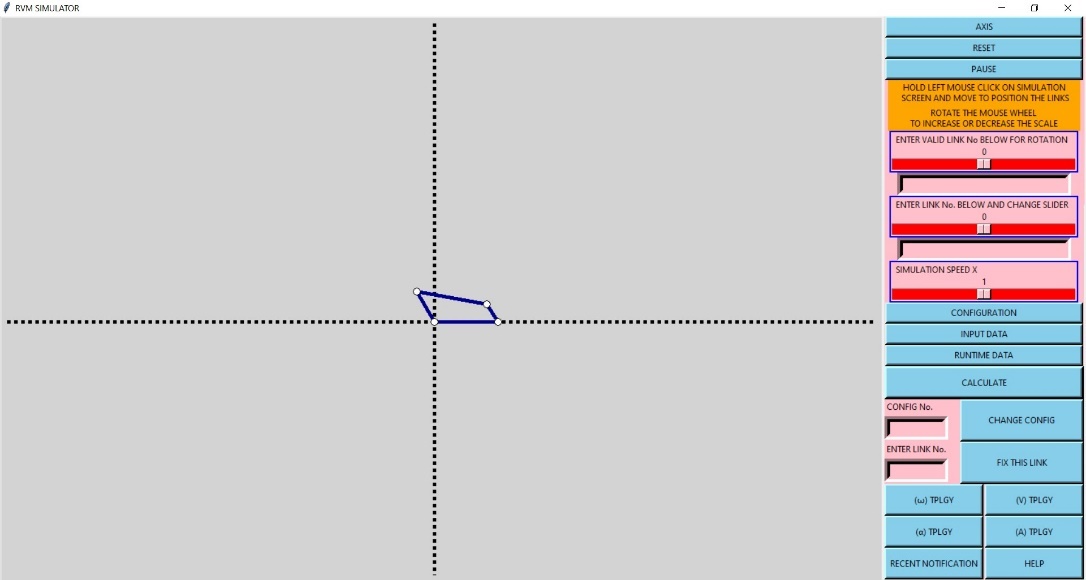
**This section mainly focuses on the frontend visualization of the simulator.**

* This simulator is made from python using modules such as numpy, math, cmath, sympy, tkinter, threading, time, etc.,
* This simulator simulates in the form of frames where in each frame all the backend calculations to find appropriate values and all the frontend manipulations for the linkages takes place and shows it on the screen, and every command is picked up in the successive frames and all the functions and algorithms are adjusted based on the command and again undergoes calculation.
* All the widgets used are placed in the layout which is split up on the screen and the additional windows, widgets, options that occur on a button press or by a command gets popped up on top of the current screen to avoid scarcity of space due to which the main widgets appears properly on the main display screen.

**Different types of features of the simulator :**

**AXIS**

* places axis approximately to the center of the simulation screen for reference.



**RESET**

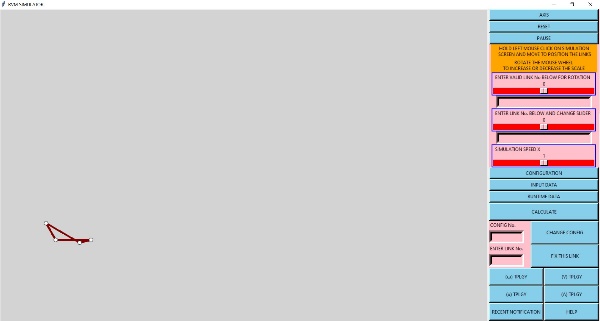
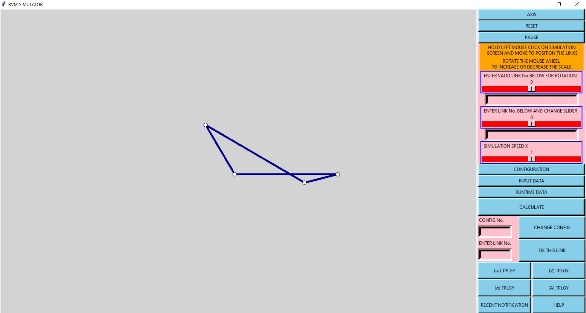
* reset all values back to the values found successfully just after calculation.
* This also make sure all the conditions are set back to initial condition.
* This option also changes color of the linkages to make sure it’s a simulation reset.

**PAUSE/PLAY**

* Stop or continue the simulation
* It basically achieves this by disabling the angle increment for each frame. Due to which we can still visualize other changes by stopping the simulation since the code is still running in backend.

**To move the links anywhere on the screen we need to click the left mouse button and drag it on the simulations screen.**

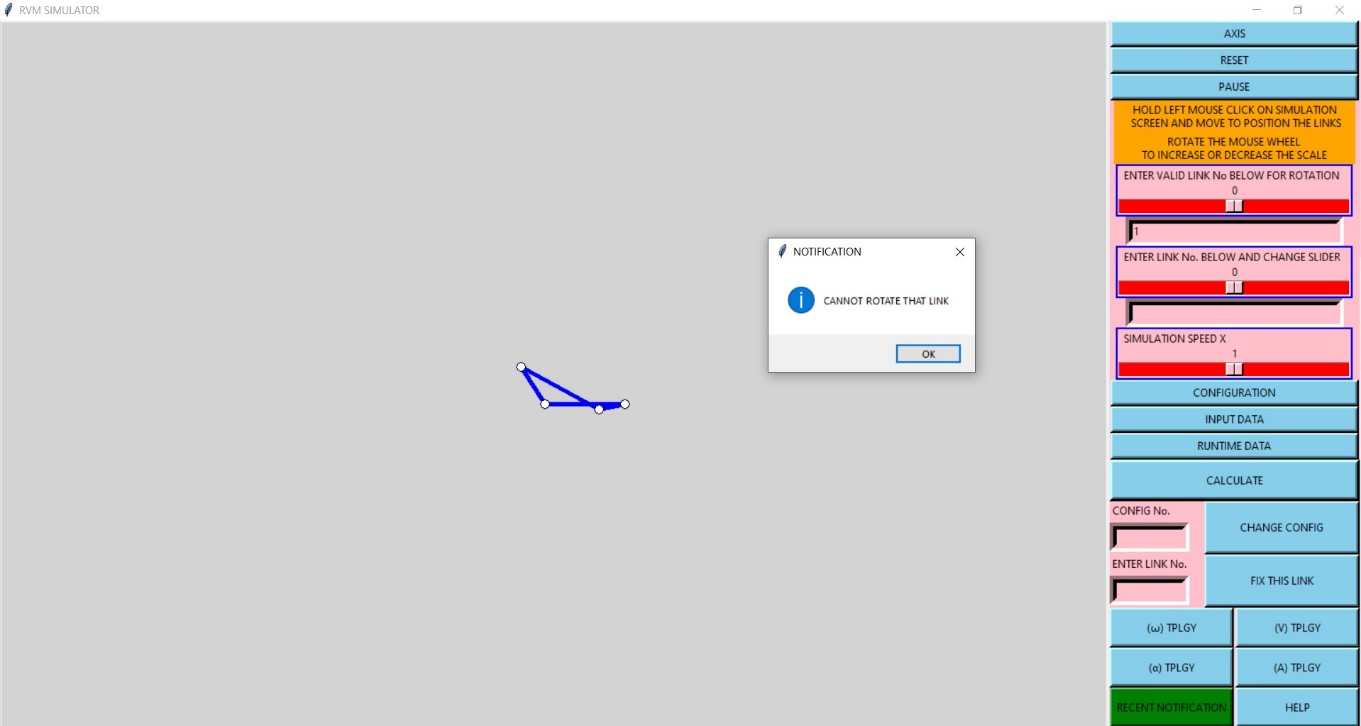
**To scale up size of links or scale down, rotate the mouse wheel up or down.**

First one is the effect of dragging and second image is the effect of scaling.

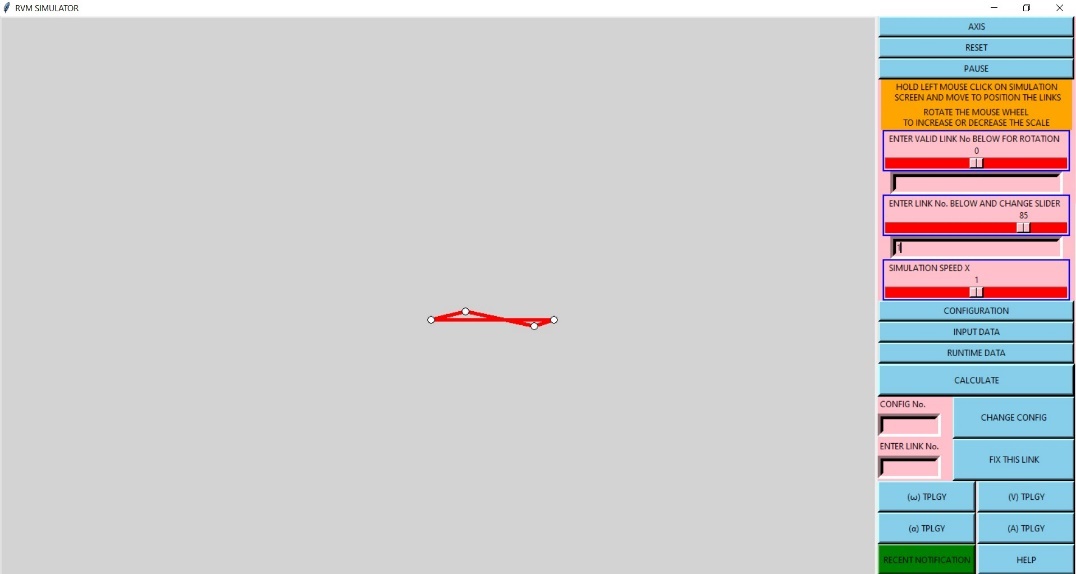
**MANUAL ROTATION**

This feature helps in visualizing simulation by manually rotating the links by entering the link number to be rotated. But if the entered link number to be rotated is of the non-rotatable links, then the notification button goes green and shows an error stating it cannot rotate that link.



**MANUAL RESIZING**

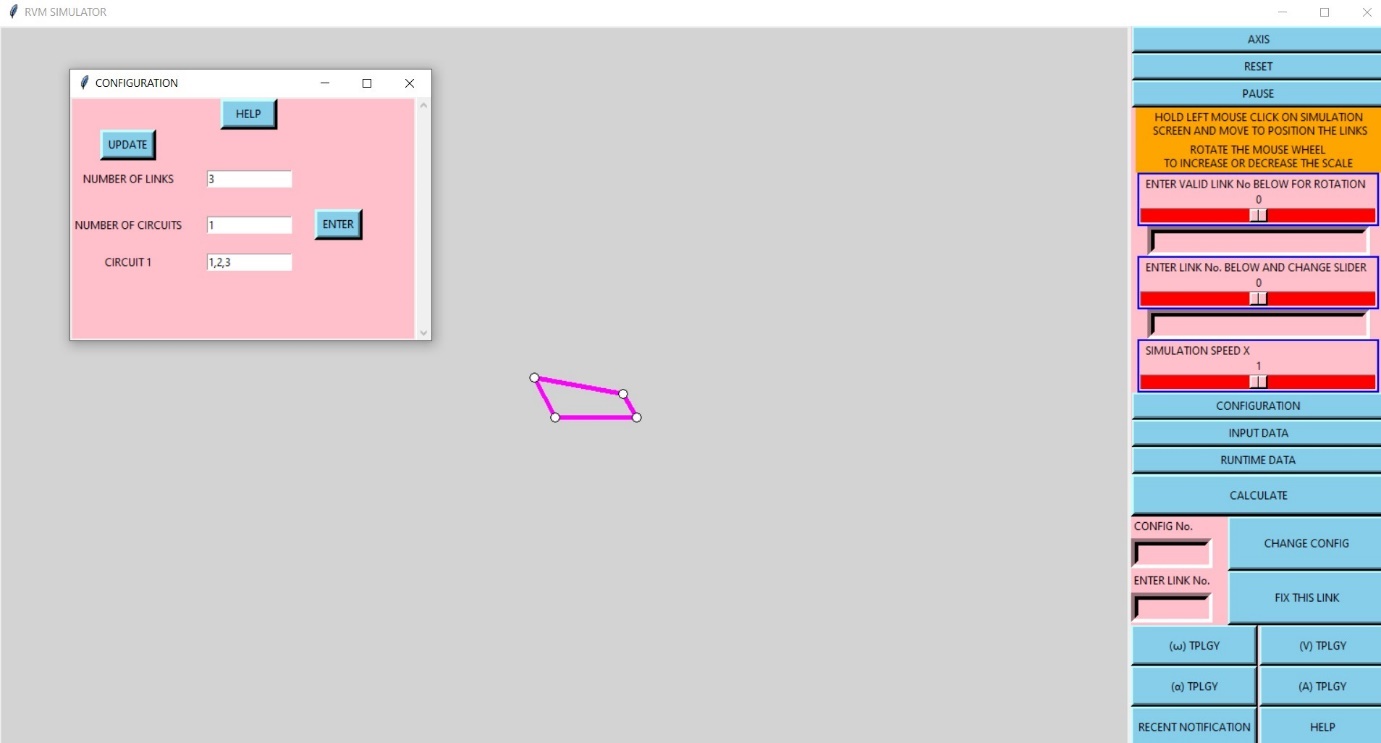
* This feature helps in visualizing simulation by manually resizing the links by entering the link number to be resized.



**SIMULATION SPEED**

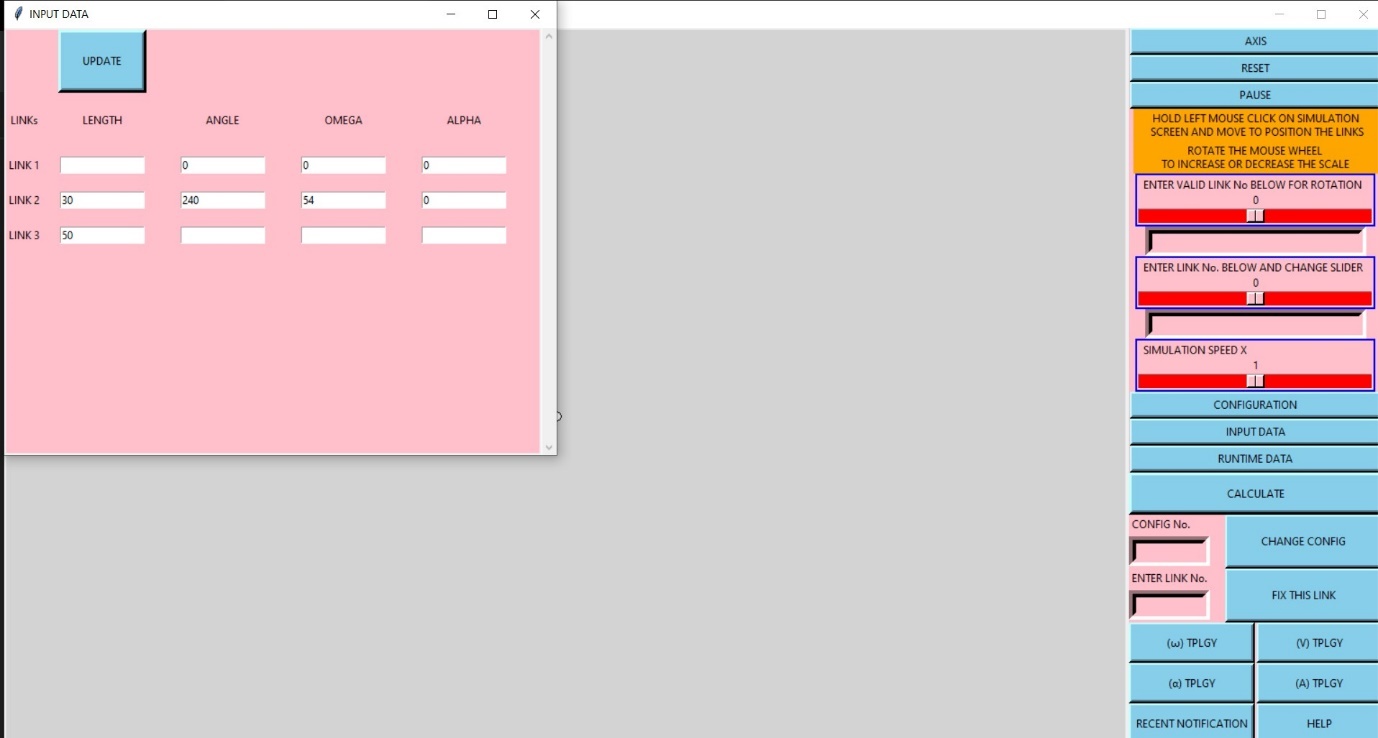
* This feature helps to manually visualize speed of simulation in runtime by either increasing the speed of simulation or slowing it down.

**CONFIGURATION**

* This feature changes the number of links, their order and number of loops.
* This is mainly responsible for making sure the links are placed head to tail in proper order desired by the user.
* This feature mainly asks for number of links and the configuration.
* Ex., if we need a slider cranker mechanism then there are 3 links placed in the order 1,2,3. Where each number in **1,2,3** is links number and their details such as r,, has to entered accordingly in the **INPUT DATA** feature.
* Even after entering configuration data the previous configuration is still getting executed until we hit **CALCULATE**  and there is answer obtained without any errors. Else the previous configuration will continue to simulate

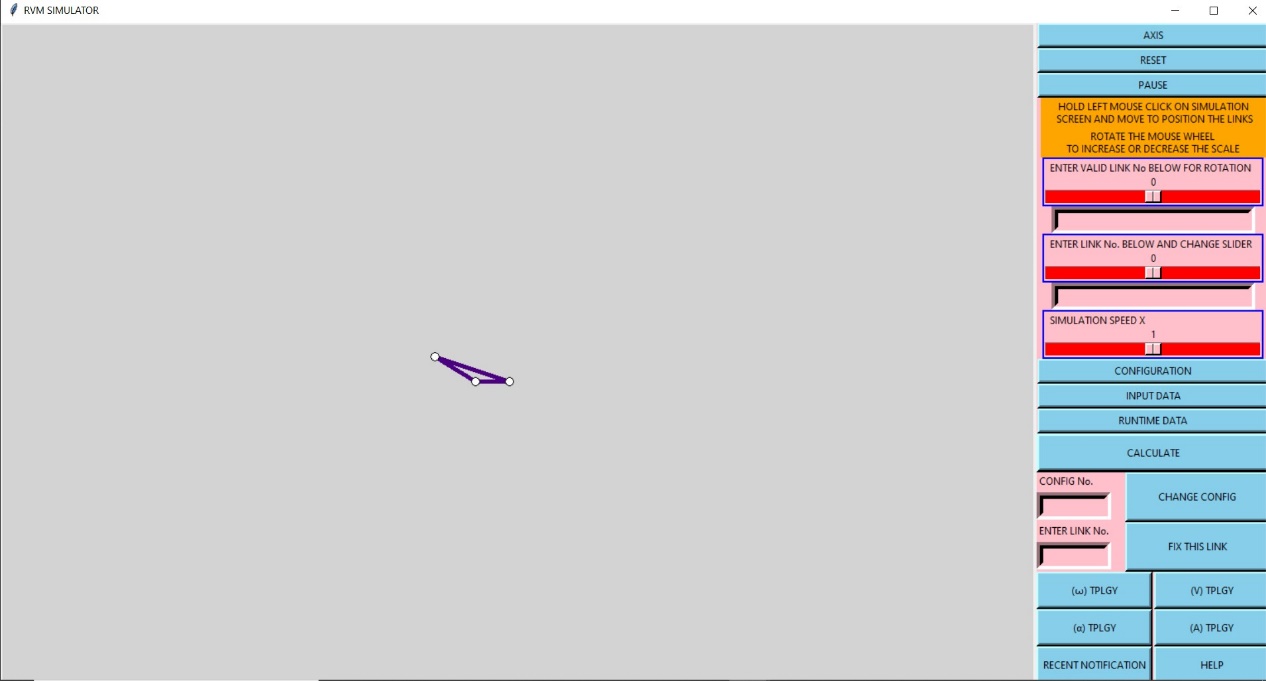
**INPUT DATA**

* This feature mainly stores the simulation data for the linkages and the simulator takes in values from here evertime it’s been reset, because this is the initial condition for the current configuration and every time reset command is entered then the simulator comes back to this condition.



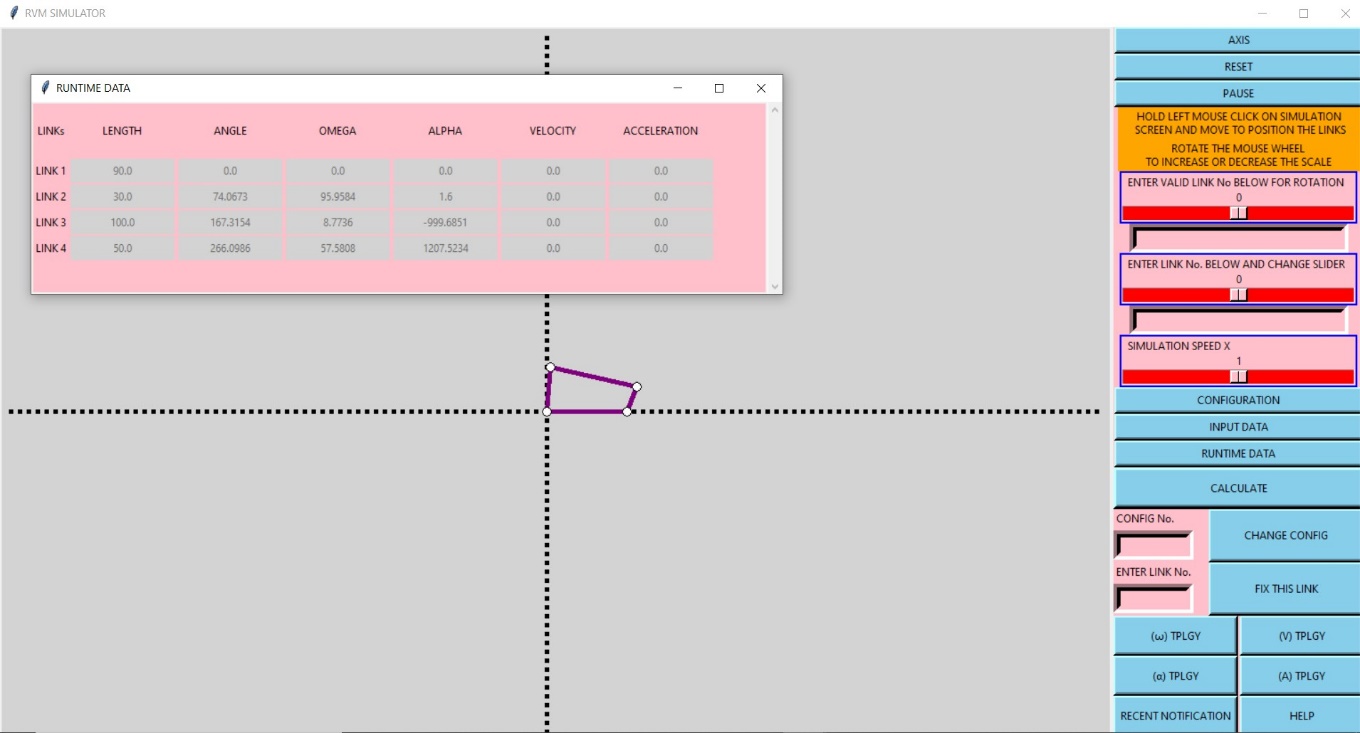
**CALCULATION**

* If all the above process are done without any error then after entering **CALCULATE,** the result would be updated as shown.



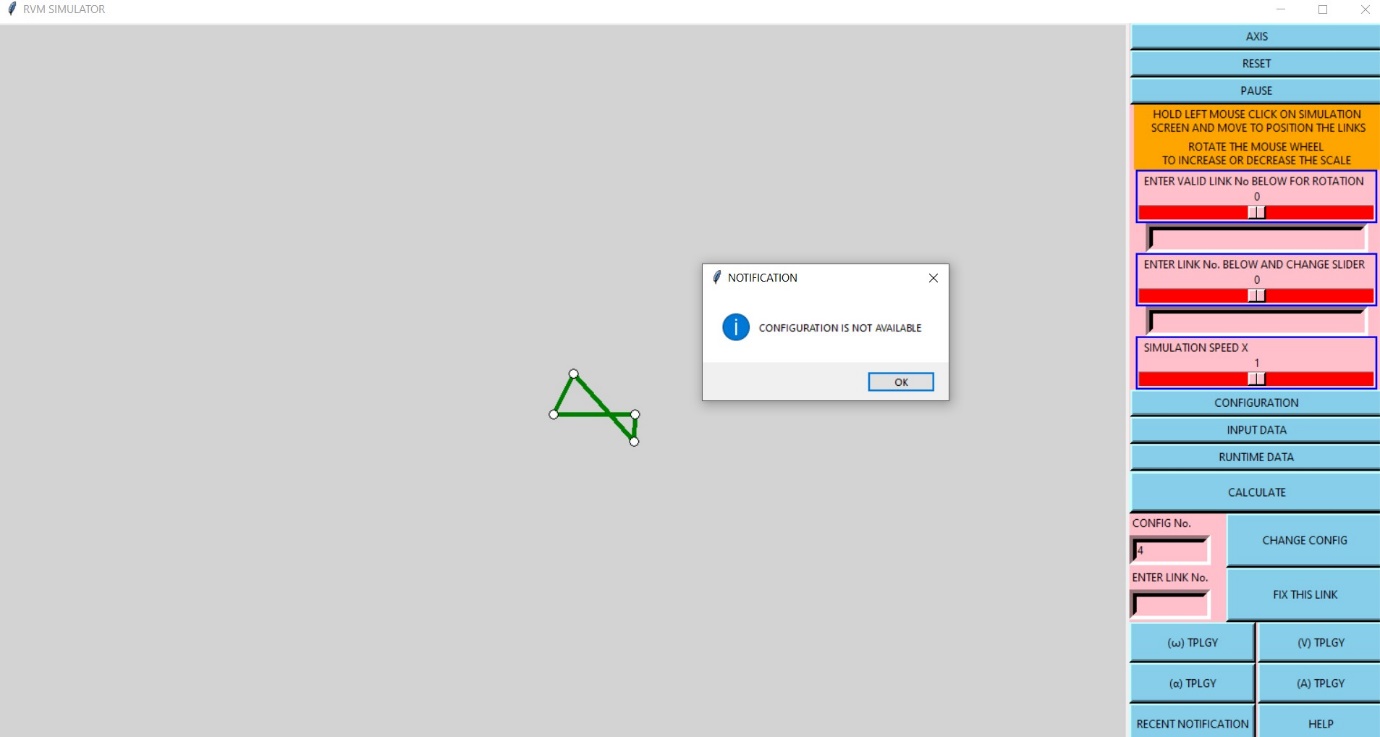
**RUNTIME DATA**

* This feature enables us to see all the values for all the links changing continuously in runtime.



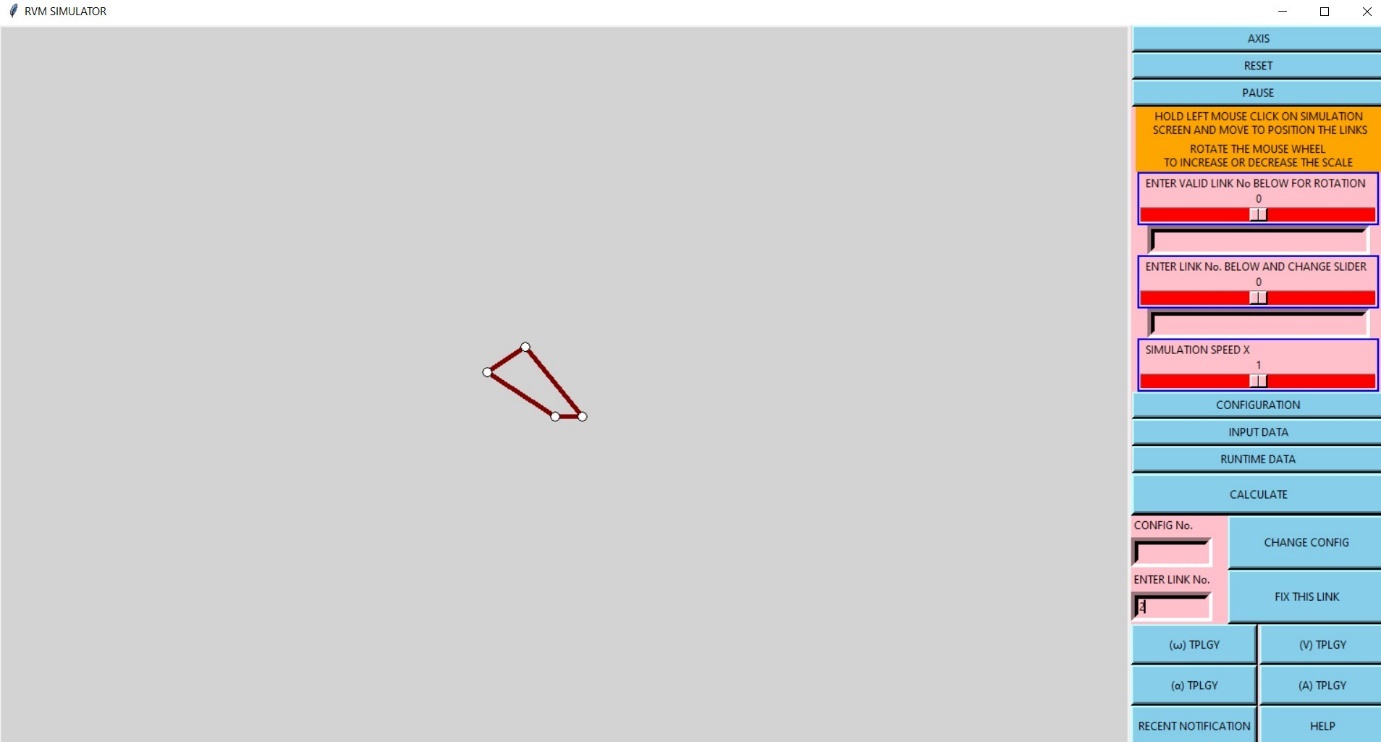
**CHANGE CONFIG**

* This feature takes in the configuration number and after entering **CHANGE CONFIG** then this will show the configuration entered.
* Initially knowing the number of configurations is difficult, due to which user can enter the integer number from 1 and if there doesn’t exist any configuration for the entered configuration number then it shows an erro

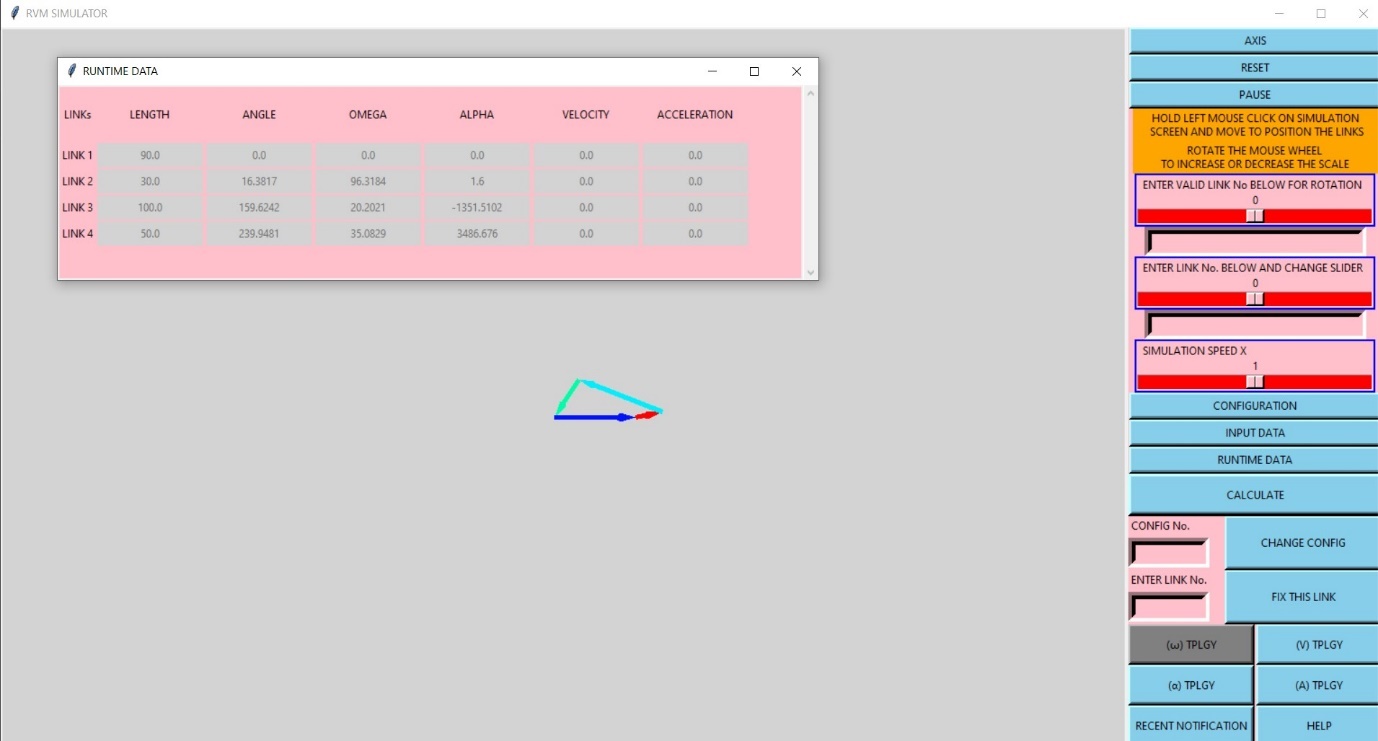


**FIX THIS LINK**

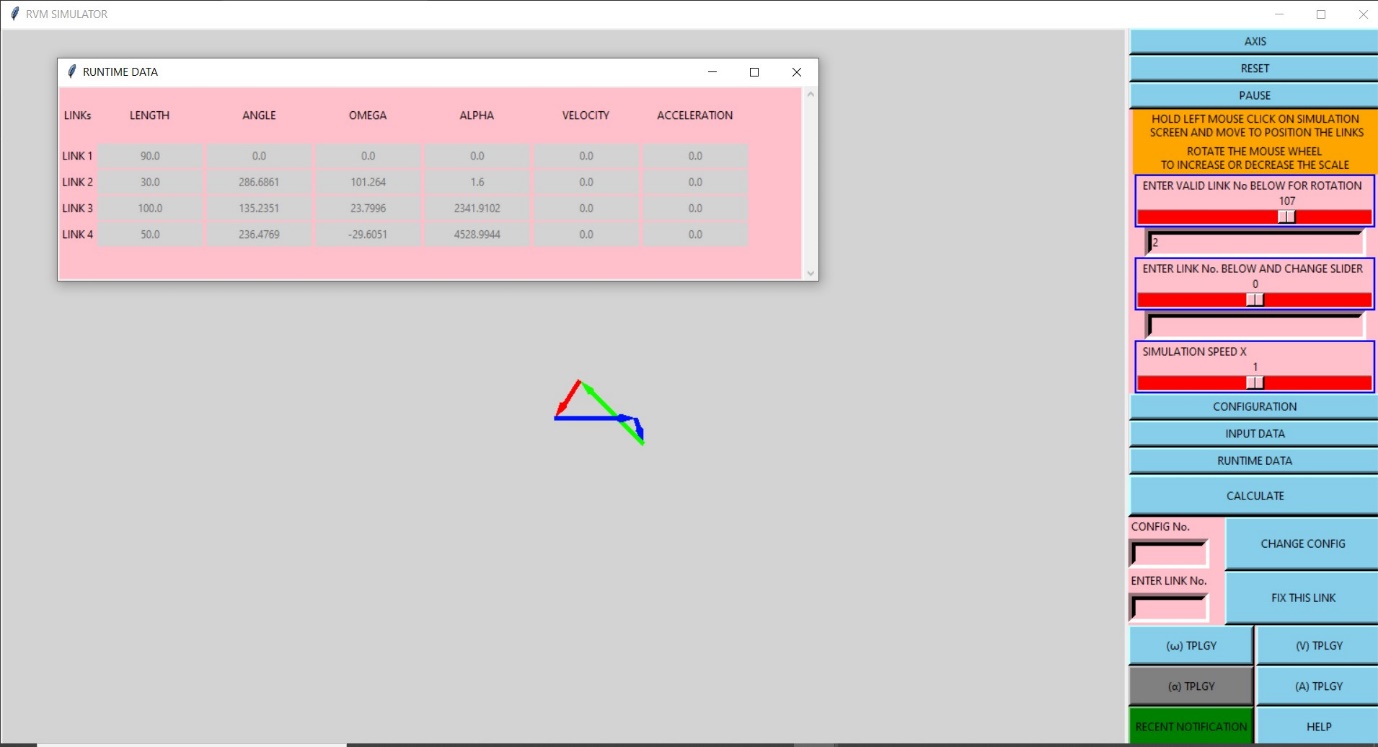
* This feature basically changes the ground link to different link among other links. The link number that has to be fixed should to be entered in this feature.
* This basically rotates all the links by the same amount as the current angle of the chosen link to be fixed. Since all the links gets rotated by this amount about its pivot point (point where the link start), its equivalent to setting the chosen link as ground link that stays horizontal unless manually rotated.



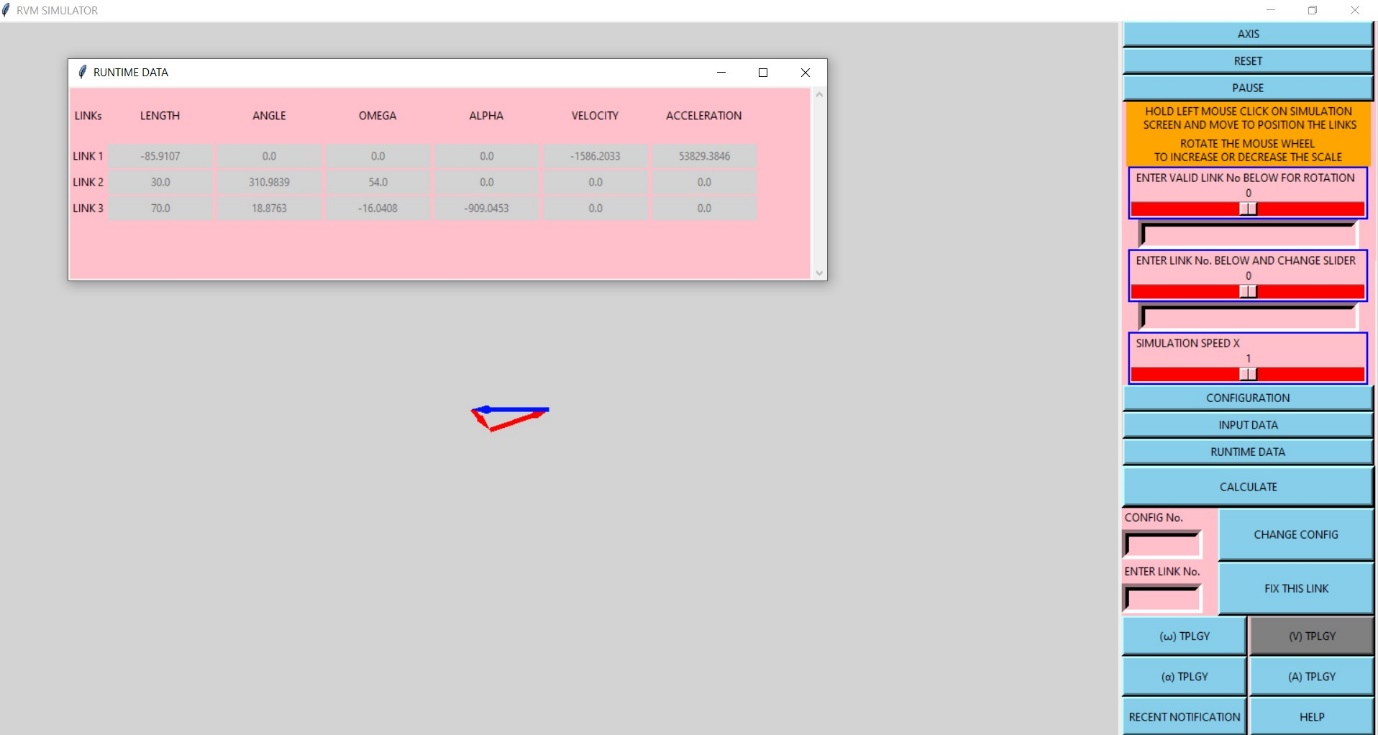
* Here the 2nd link that is shortest gets fixed, due to which the simulation appears to be tilted pr rotated by some amount, giving the visualization of the 2nd link being fixed or grounded.
* This feature enables user to visually feel the value in runtime during simulation that given color contour from red being highest value to blue being lowest value.
* The results of this color contour can be verified by comparing it with runtime data.



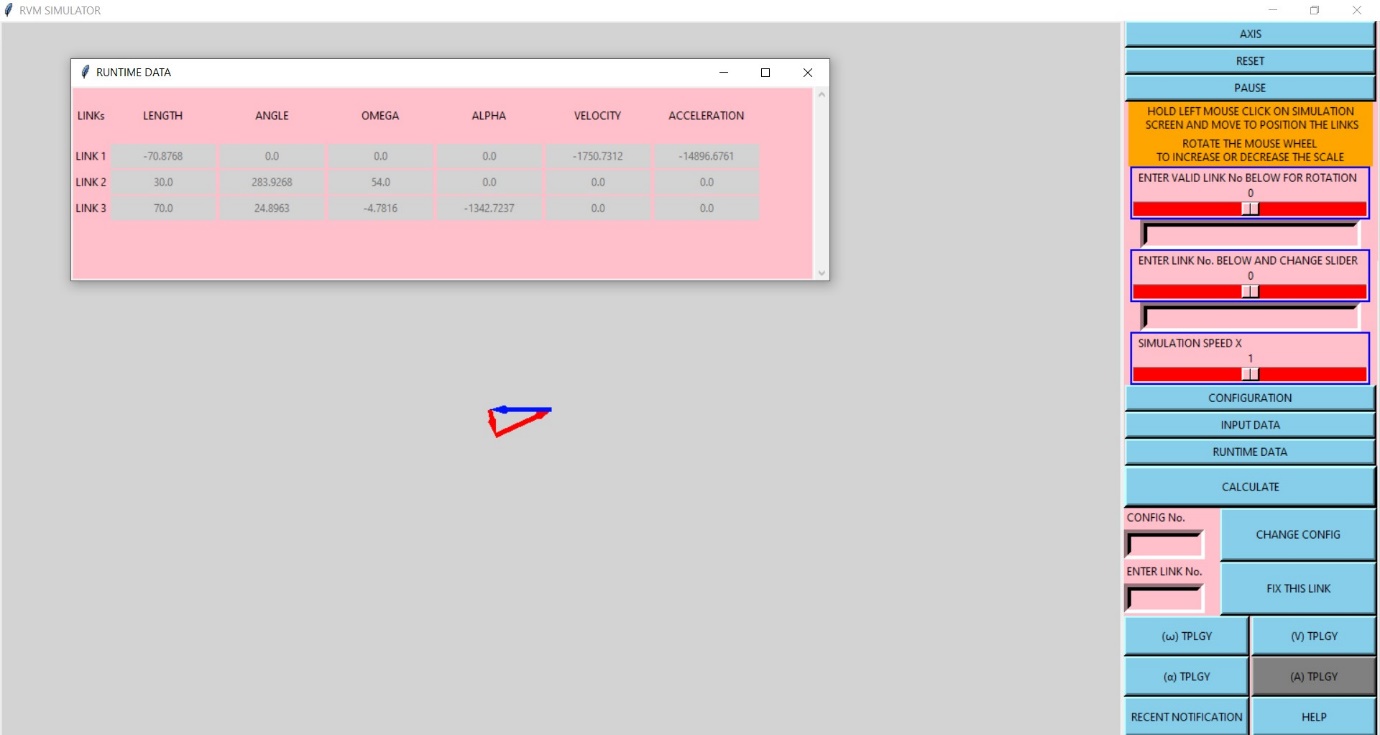
* As seen in the above image, values are in the order 2>4>3>1, where these numbers indicate link number.
* Following the above order, the highest values is link 2 that is the red color and least value is link 1 which is dark blue. All the intermediate values have intermediate color in red to blue spectrum.
* **NOTE THAT THE GROUND LINK ALSO HAS SOME COLOR SINCE THE COLOR CONTOUR ARE ALL RELATIVE DUE TO WHICH IF THERE IS A LINK WITH NEGATIVE , THEN THAT WILL SHOW MORE OF BLUE SHADE THAN THE GROUND LINK.**
* All the above rules apply same to **.**
* Where  **V**  and **A**  are velocity and acceleration of the respective link.



* Since velocity and acceleration isn’t visible in rigid links, slider cranker is used below since the slider length changes who’s first and second derivative gives **V**(velocity)and **A**(acceleration) .

****

**VELOCITY TOPOLOGY**

****

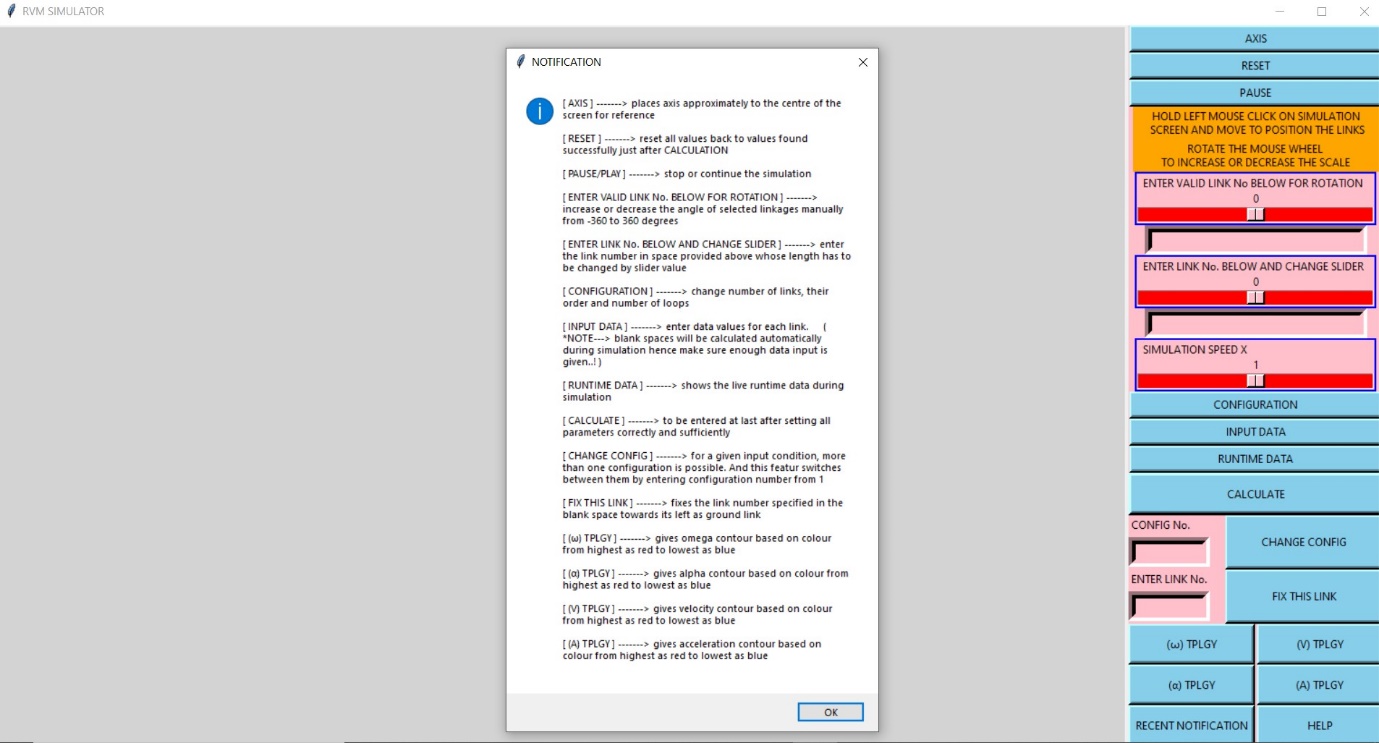
**ACCELERATION TOPOLOGY**

**RECENT NOTIFICATION**

* This feature stores the recent notification occurred in the simulation which helps the user to review the notification again.

**HELP**

* This feature helps the user to understand about the simulator and use it effectively.



**CONCLUSION**

This simulator is developed to help the user to understand and visualize about the kinematic linkage problems and also solve the problem provided the initial conditions are given appropriately.