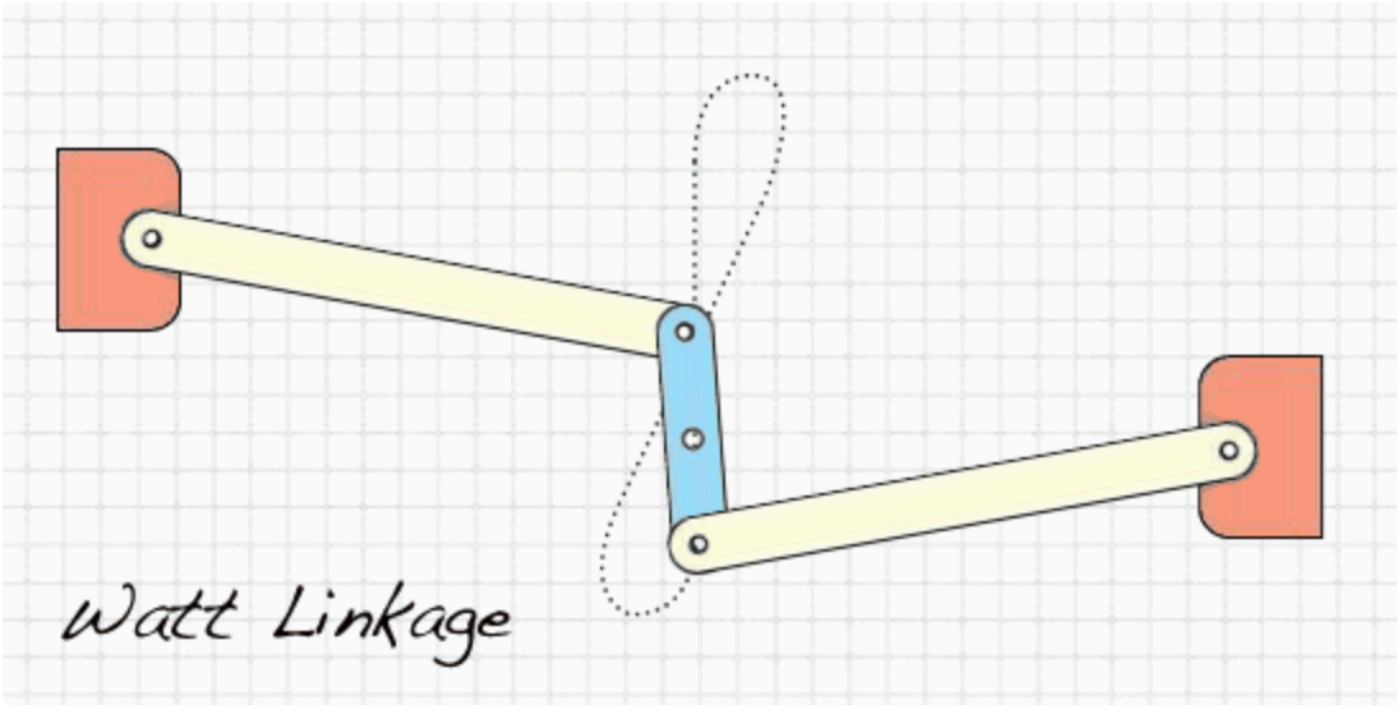
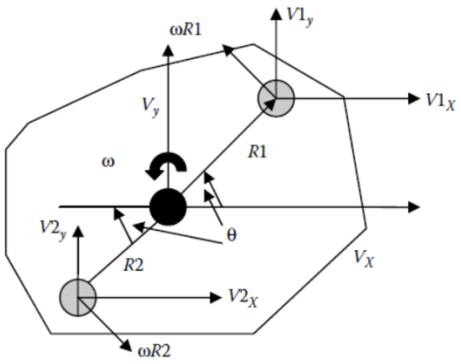
Project report for Mechatronics Programming for Real Time Systems



First the derivation of the Bond Graph:

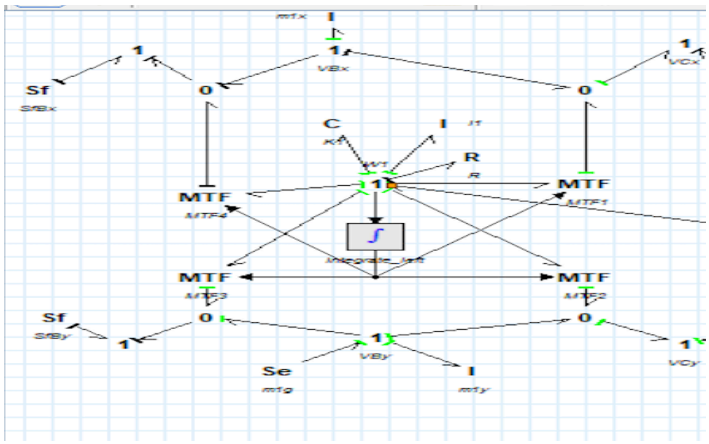
As shown in the figure, our model is a 3 bar mechanisms which is widely used in the car mechanisms, Watt's linkage mechanism is used in the rear axle of some car suspensions as an improvement over the Panhard rod. This mechanism is used to prevent relative sideways motion between the axle and body of the car. It approximates a vertical straight line motion more closely and does so while locating the center of the axle rather than toward one side of the vehicle, as commonly used when fitting a Panhard rod. These 3 bar mechanisms are modeled by the same way as the generic model for any 3 bar mechanisms:



But, our model is a little bit twisted as they must be referenced to the same frame, but the frame is fixed for each link the model will be different a little bit in the signs (cos and sines).

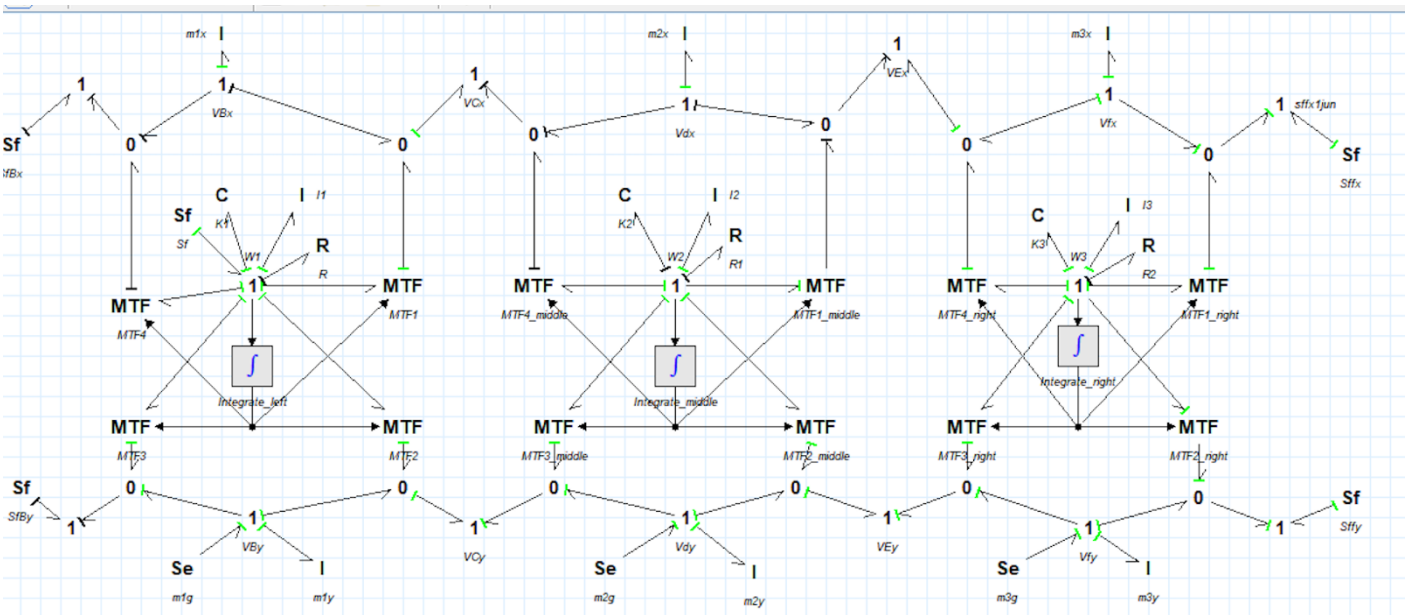
So, after searching for the right model, we have figured out by observing the model if we used the same model for the 3 links but with a global fixed frame and signs will be changed automatically as the linkes of the model exceeds 90 degrees, 180 degrees, etc, so the model will be right for all of them.

So the model for 1 link only:



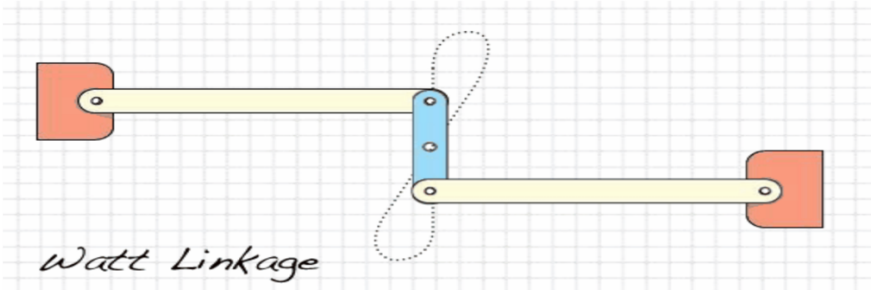
We can observe that the sources of flow on the left are equal to zero (as they are fixed joints), and also we can observe that every 2 links share a common points which are C point and E point, as the model developed earlier in Milestone 1.

So by connecting the 3 links together and applying the fixed joints constraints the full model will be like this:



Every 1 juntion represents a certain point's velocity in a certain direction (i.e x or y direction).

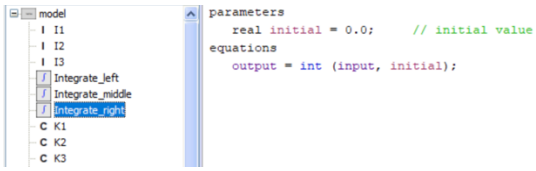
And by checking the real applications of the watt links mechanism, it seems that its trajectory is that the (omega and theta of the right link is always oppositve that the left link) and the theta and omega for the middle link is very small and its middle point is just a translation motion (until the end), you can check the 3D simulation video.

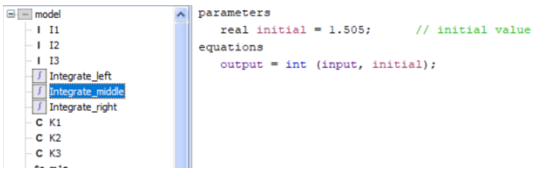


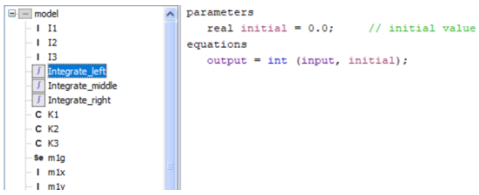
So, we assumed that our input is to the omega of the left link and we are already know that there is a certain trajectory that it can move to, we assumed this input to be a sine wave from 0.4 radians to -0.4 radians to avoid any breaking of the linkage and track the output of the model.

```
parameters
   real flow = 1;
variables
   real effort;
equations
   p.f = flow * sin(6*time);
   effort = p.e;
```

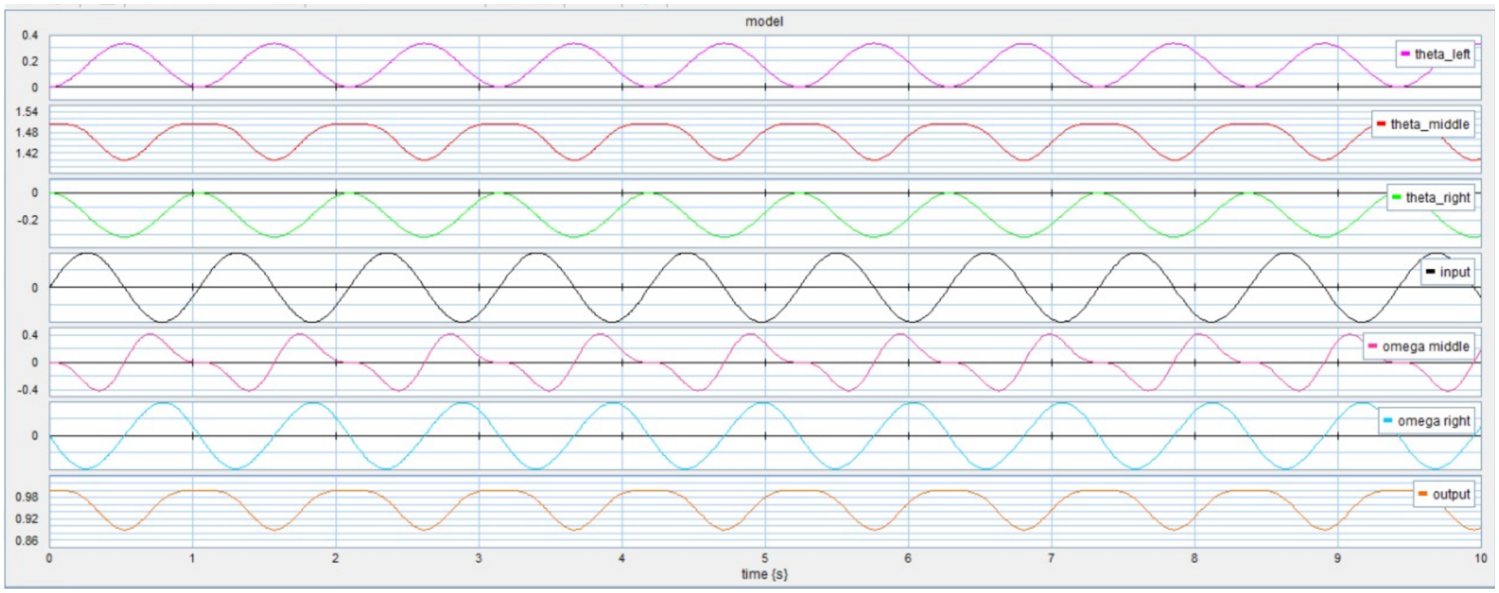
The intial conditions as shown in the image is: link 1 is of angle 0 and link 2 has angle 90 degrees (1.57 radians) and link 3 has angle 0 or – 180 degrees (-3.4).







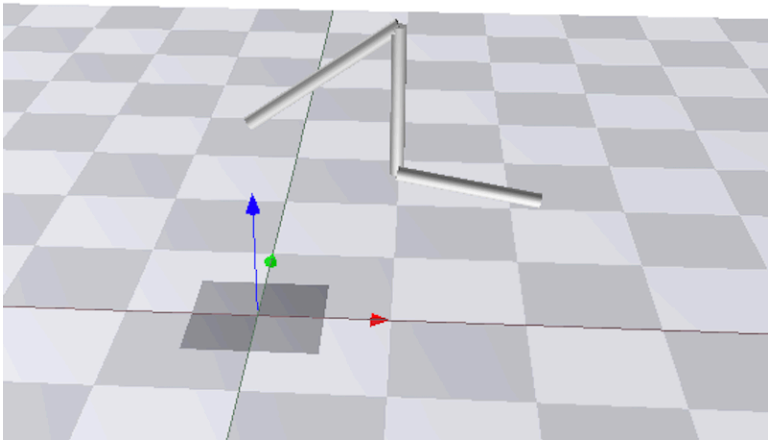
We found this output for thetas and omegas.



It’s as we expected exactly: the theta left is oscillating from 0 to 0.4 and the theta right is oscillating between 0 and -0.4 and the theta middle is oscillating between 1.5 to 1.48 (which is approximately constant and equals to 90)

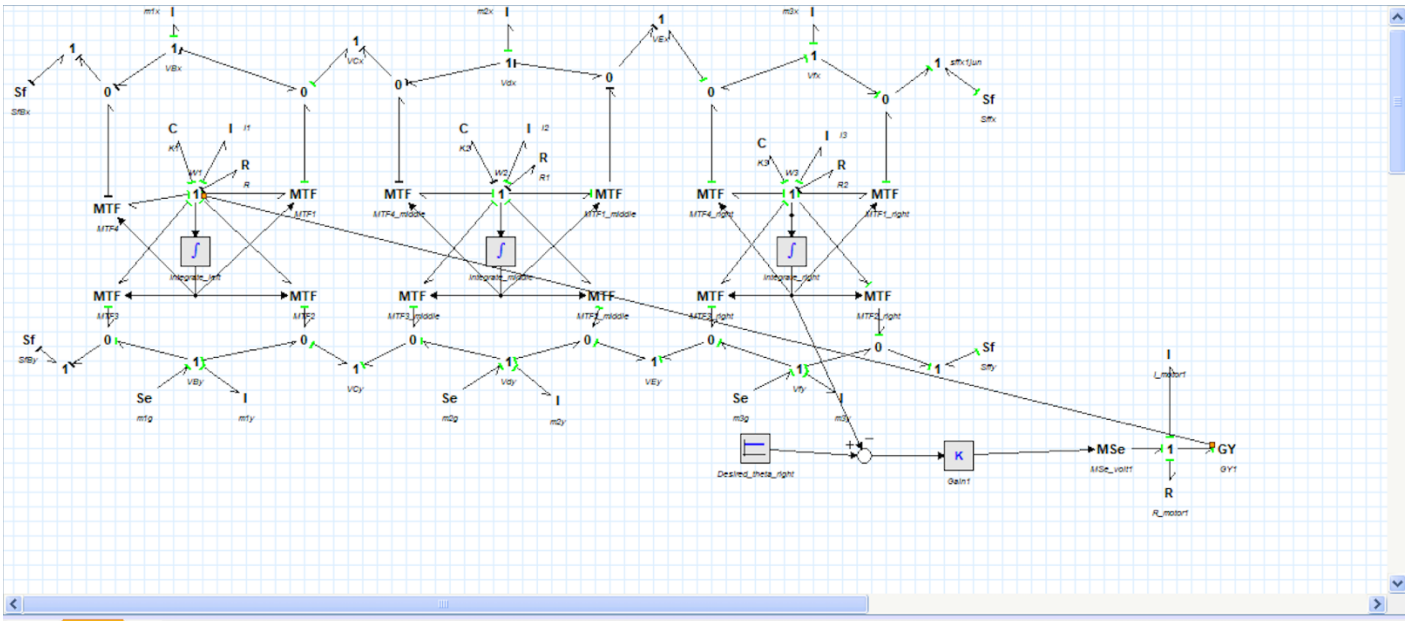
And Also, it is very obvious that the omega left is always opposite to the omega right (their summation equal to zero, and omega middle is the same with a smaller magnitude.

And Also, we animated the 3D animation of our project and it acted as expected and here is a hence of our Watt’s Straight line mechanism, check attachement for full video,:



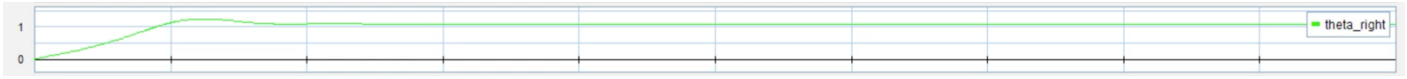
Control by P controller:

We tested the control for our model to check its feasibility to control our linkage, and in our control mode we assumed that the input is a motor torque to the right link to control the angle of the left link with a P control. We are putting a constant block as desired theta right and calculate the error and multiply it by -1, as theta right is inversely propotional to theta left and make this as an input to the voltage of the motor and then supplying torque in the other direction to the left link as shown in the Next Figure:



And We observed the results to be like this:

We are checking that the omegas are stabilizing, and for checking the theta right (desired to be controlled)

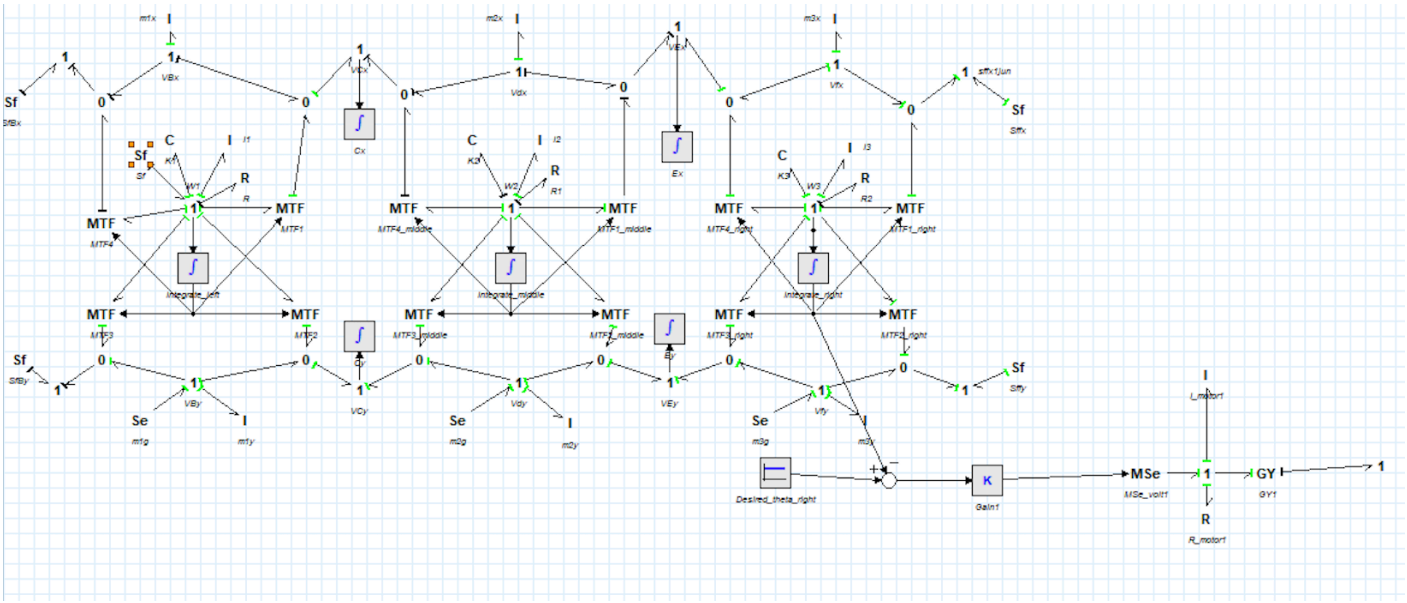


Theta right is stabilized at 1 which is the given value with a small steady state error of 0.1:

Theta desired:

```
parameters
   real C = 0.9;      // output value
equations
   output = C;
```

The full model for animation response, closed loop response, and open loop response, parameters, MTFs functions are attached as well. Here is the full and last model:



Thanks for patience☺