# **Weekly Report – W1 Spring 2023**

## **Problem & Task**

1. Update about our own simulation package “Falling SRA V2”:

(1). Since it’s hard to figure out whether the inertia matrix in the governing equations is always positive definite by its analytical solutions of the eigenvalues, according to the suggestions of the advisor, I decided to add another function in the simulation package to record the eigenvalues for each simulation time step so that we would be able to check if there are any negative values after the simulation is done;

(2). ODE function setting up: Last week I was stuck by the last step, to set up the ODE functions according to the expressions using MATLAB Symbolic Toolbox, there seemed to be a lot of limitations to rewrite those equations with symbolic variables into the format that MATLAB could solve ODEs, so this week I will try at least 3 different methods referenced from the resources in MATLAB Community;

2. Update about the TMTDyn package

(1). Figure out the reason behind for some of the special input settings, there will be some errors for animation recording.

## **Solution**

1. Falling SRA V2

#### (1). ODE setting up

All the matrices and expressions were derived using symbolic variables in MATLAB, when transferring them into the format that the ode functions in MATLAB could accept, there are a lot of problems however, which have been listed below:

* ode functions in MATLAB cannot deal with symbolic variables directly, which means that we have to replace the symbolic variables like and by q(1) and q(2) , apparently the class for the symbolic variable expression and state array q is different;
* If we insist on symbolic variables, according to the documentation of MATLAB, it seemed that we could only use dsolve function rather than ode to solve the EOMs, which is definitely not what we want.

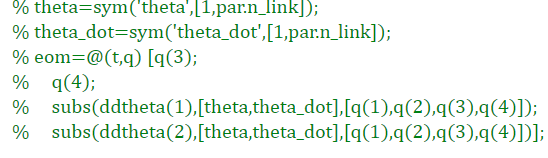
To begin with, our simplified EOMs will be

where according to the notes in last week’s report, for brevity, we can rewrite it into a more compact format as follows,

Thus due to the calculation rule in MATLAB, we have

Since the expressions for and are too complex, even simplified after using Mathematica, so their specific expressions are shown in the Appendix. If we set the symbolic variables in the way of ode solver, they should be

The first method I used was to directly replace the symbolic variables, theta1, theta2, theta\_dot1 and theta\_dot2 by q(1), q(2), q(3) and q(4) respectively, but it failed because the sym class cannot be transferred to double.



**Fig. W1-1** The example to use directly replacing method to get the ode function

The second method I used was to set up a symbolic variable, theta(t), however, this method is only applicable for dsolve function, I haven’t found any supported documentation in ode function. The third one is to transfer the expression of and in to characters, then use replace function to replace the symbolic variables, theta1, theta2, theta\_dot1 and theta\_dot2 by character ‘q(1)’, ‘q(2)’, ‘q(3)’ and ‘q(4)’, then change the class from string to symbolic again, but we still cannot directly use the expression to forge the ode function as the array of q should be double. So temporarily the only way I could think out is to type the name of the array storing the expressions in the command window of MATLAB, then copy and paste them into the positions of q(3) and q(4) in the ode function, then the setting up had been finished. Except for this, I haven’t found out any other better methods that could establish the ode functions automatically.

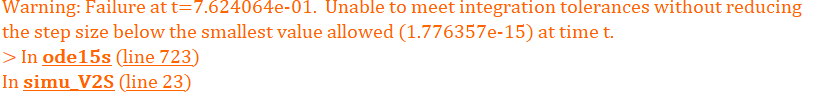
#### (2). Simulation results

To start the simulation, the initial conditions and other parameters were set as follows,

**Table W1-1.** The parameter setting to trigger the simulation

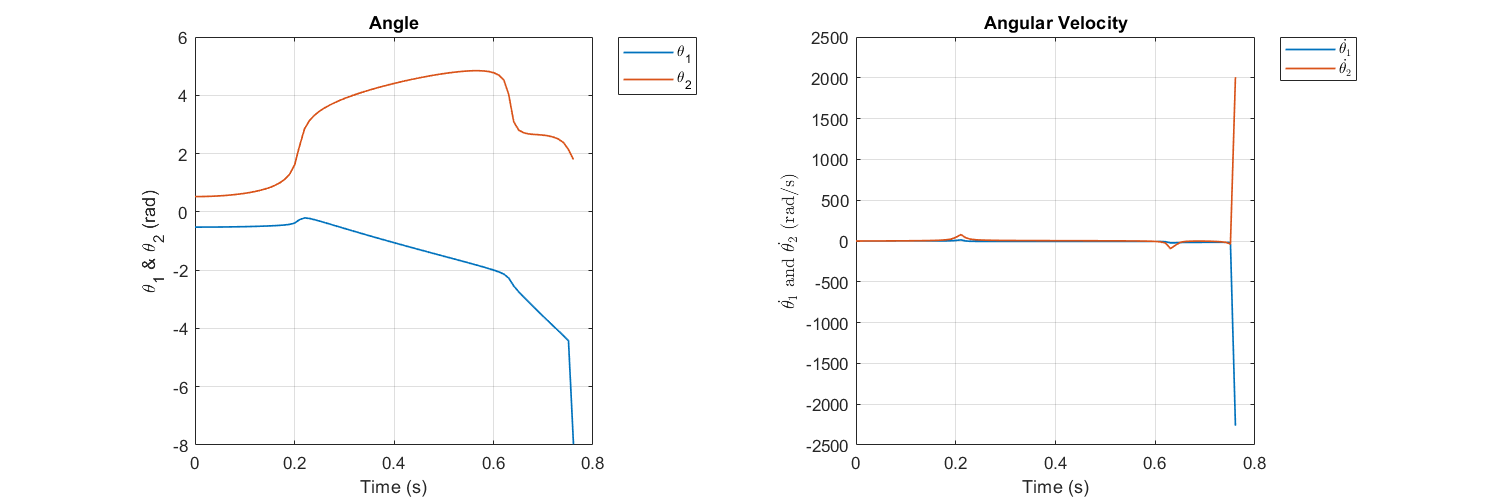
|  |  |
| --- | --- |
| **Parameter** | **Setting** |
| Initial conditions (IC) |  |
| Time span | Simulation time: 0 ~ 10 s  Time step: 1000 |
| Tolerance | Relative:  Absolute: |

Rather than the Simulink model I have set up previously, this time the simulation solved by ode function can run successfully without any errors, the only limitation was that there are a few lines of warnings in the command window which can be seen in the figure below,



**Fig. W1-2** The pop-up warning for the simulation with the settings in Table W1-1

And the simulation results are shown in the figure as follows, the figure on the left is the angle change, and the right one is the angular velocity; the results for the angular velocities could vary quite a lot with the increasing of the time step, the trend of angle change remained almost the same for different time step settings.



**Fig. W1-3** The simulation results for angle and angular velocity change

Moreover, the simulation time didn’t comply with the time span we have set for triggering the simulation, the simulation aborted at around 0.8 s, which is exactly as the warning told us, so according to the feedback in command window, I have tried to change the relative and absolute tolerances, the simulation still stopped at 0.7624064 s. After that I looked up some resources in MATLAB Community, somebody said it might be due to the singularity problems of the matrices, but I have recorded the eigenvalues of the inertia matrix for each time step, they are all positive values, I haven’t figured out the reason behind yet, maybe it’s just about the parameter settings.

1. Animation frame problem for TMTDyn package

The animation frame error emerged for two additional tests last week, one is 1 time Young’s Modulus, 1 time density, 2 segments with 40 N external force exerted on the SRA tip in the x direction; another one is 1 time Young’s Modulus, 1 time density, 5 segments with 5 N external force exerted on the SRA tip. So this week’s work is to firstly duplicate the tests with issues again, then try to figure out any possible solutions to deal with it; if not, maybe we can block the video record function first, then run the simulation again without blocking, because the video recording process will not be terminated by the error, however, this will affect the final result plot, when we block the animation, the result figure might show up again, which is the backup plan.

#### (1). Trouble shooting for frame issue

Because the frame error only appeared twice in all the tests scheduled, and the 5-segment one takes so long to obtain the result, so we just need to duplicate the one with 2 segments and 40 N input, however, the error didn’t emerge for no reasons. Then I tried from 50 N to 100 N, the error was gone, and I could guarantee that all the simulation conditions are exactly the same when I did test for 40 N last week. Then I began to think about this problem in the two directions below.

##### 1). Find some useful resource online to see how others cope with this issue

As said by the experts in MATLAB Community, the getframe function will keep the first figure with its size, if the size changes in a for loop, it will cause some error, the best and simplest solution is to use hold on command and plot all data inside, however, it would not explain our case, because I haven’t amended any parameters concerning animation, which I will just take as a reference.

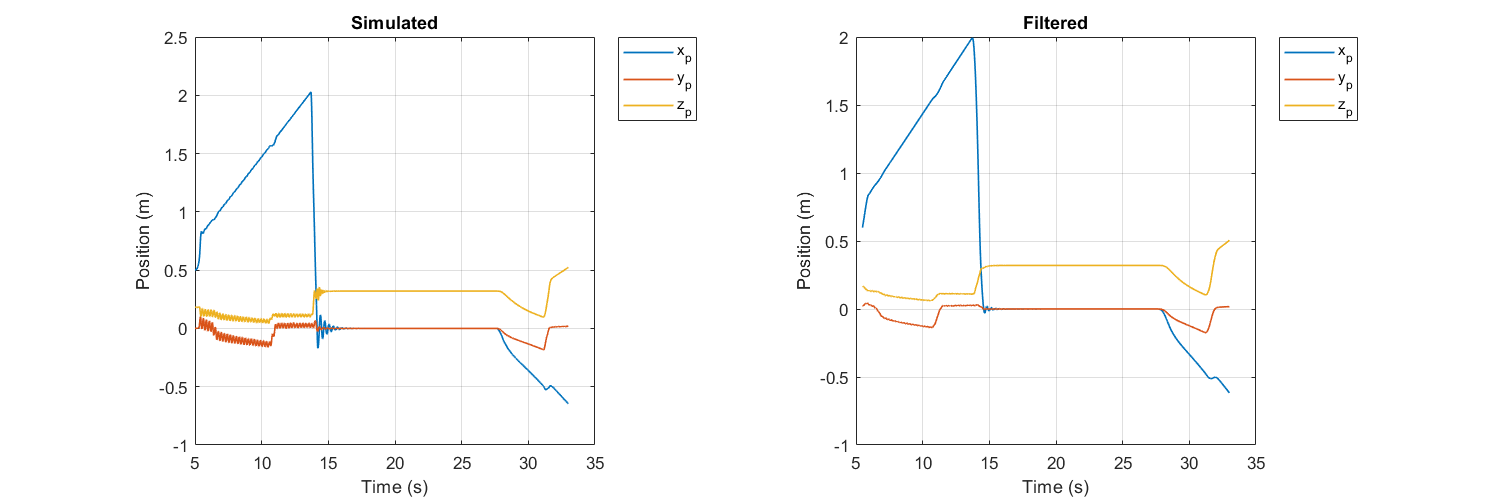
##### 2). Explore whether the frame will be affected by RAM dispatched

I recalled the whole process before and after, the only difference seems to be last week when I did the simulation, another exe procedure was running at the same time, which accounts for a lot of memory, I guess if the RAM dispatchment will affect the frame. According to what I have looked up, it said the memory leak problem is more likely due to MATLAB itself rather than getframe function, to figure out the real reason, maybe we have to clarify how MATLAB allocates memory when dealing with different types of dataset, such as a struct array, which shouldn’t be our focus. Thus I also made a backup plan as follows.

#### (2). Backup plan for this issue

This additional test is relatively easy to achieve, and it was designed to test if the animation process error will affect the final result plot. Usually the procedure will be terminated by a unexpected error showing up in the command window, and the subsequent codes will not be executed, that’s the reason why the final position plot of the SRA tip will not be displayed. Last week we have confirmed that even we have been notified the code had some bugs due to some special input settings, the video still can be record; so this week we are going to see if the lines of code for recording the animation video is taken out, will the final results show up again?

As shown in the figure below, our desired output comes back again, thus a rough conclusion can be made that there is no conflict between video recording and final result plot, if we find it hard to fix the frame problem, we can use this backup plan to save time.



**Fig. W1-4** The final result for animation process with issue when the relative animation codes were taken out

## **Plan**

1. Discuss about the potential reason that would cause the simulation result of the “Falling SRA V2” and try to resolve it by setting different parameters first;
2. Discuss about if there are any other potential issues that remained in the TMTDyn package;
3. Maybe I need to get familiar with the SRA hardware in our lab as soon as possible.