EML 6351 Simulation Project 4

Implement a RISE-based modular controller

- ## Simulate 4 different adaptive update law

  To implemente 4 different adaptive update law, we have to moddify $\dot{\hat{\theta}}$ while $\hat{\theta}$ and $\dot{\hat{\theta}}$ satisfy the below conditions:

  $$\hat{\theta}(t) = f_1(t) + \Phi(q, \dot{q}, e_1, e_2, t)$$

  where: $\|f_1(t)\| \leq \gamma_1$, $\|\Phi\| \leq \rho_1(\|\bar{e}\|)\,\|\bar{e}\|$;
  And $\bar{e} = \left[e_1^t, e_2^t\right]$, $\left\|\dot{f_1}(t)\right\| \leq \gamma_2 + \gamma_3\,\|e_1\| + \gamma_4\,\|e_2\| + \gamma_5\,\|r\|$.

  $$\dot{\hat{\theta}}(t) = g_1(t) + \Omega(q, \dot{q}, e_1, e_2, r, t)$$

  where: $\|g_1\| \leq \delta_1$, $\|\Omega\| \leq \rho_2(\|Z\|)\,\|Z\|$;
  And $Z = [e_1^t, e_2^t, r^t]$, $\|\dot{g_1}(t)\| \leq \delta_2 + \delta_3\,\|e_1\| + \delta_4\,\|e_2\| + \delta_5\,\|r\|$.

  All the simulations are implemented with MATLAB, the ***.m** file can be found in the **src** folder.

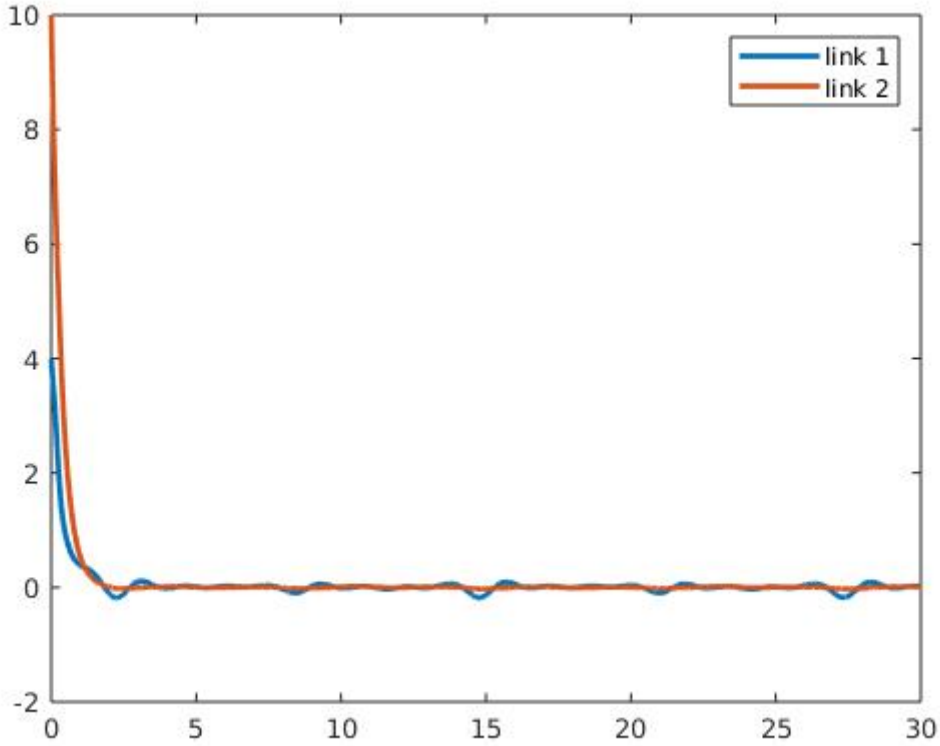
- ## Discussion
- ### (A) Simulation Section
  - #### Adaptive Update Law with $\dot{\hat{\theta}} = cos(t)$

    1. Control gains.

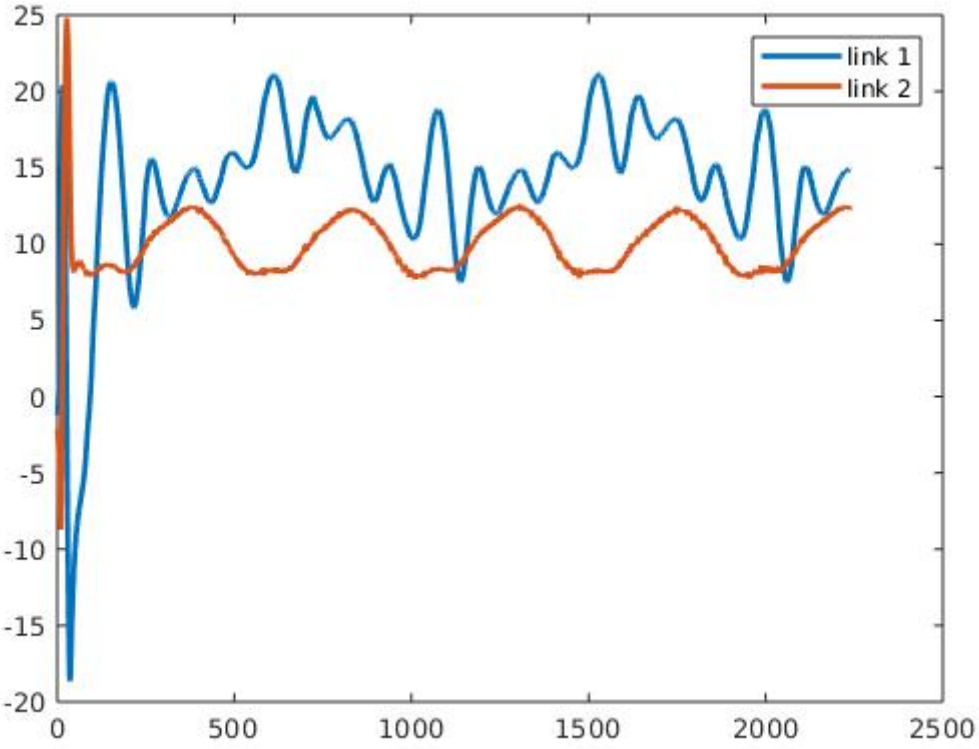
    In this case, I simply use sinusoidal as $\dot{\hat{\theta}}$.

    Control gains: ks = 10, $\alpha_1 = 3$, $\alpha_2 = 4$, $\beta = 1$.

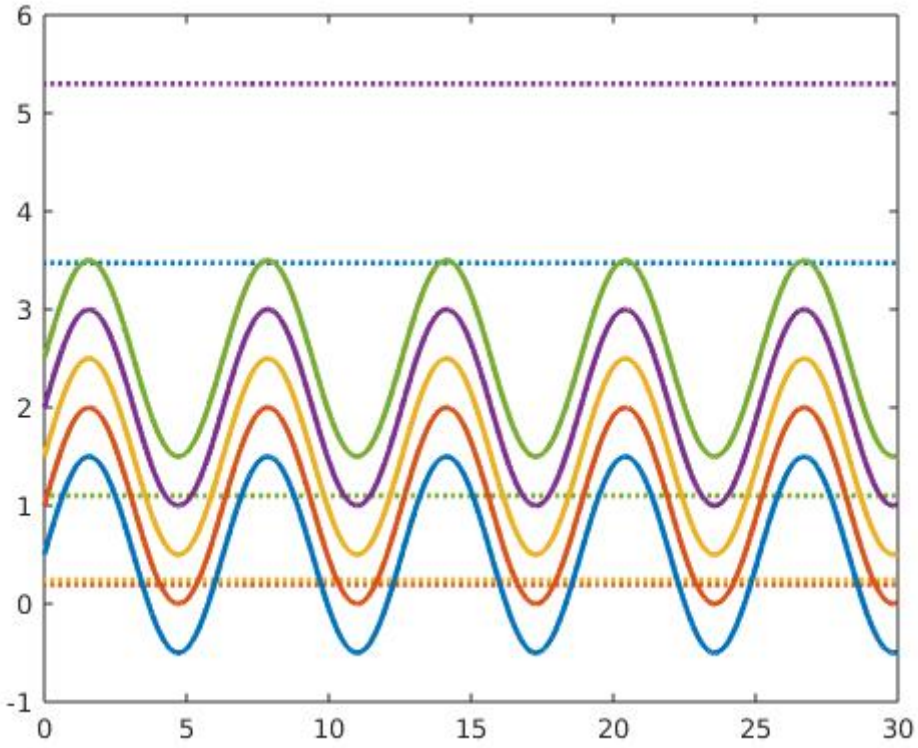
    2. Tracking error plot for each link.



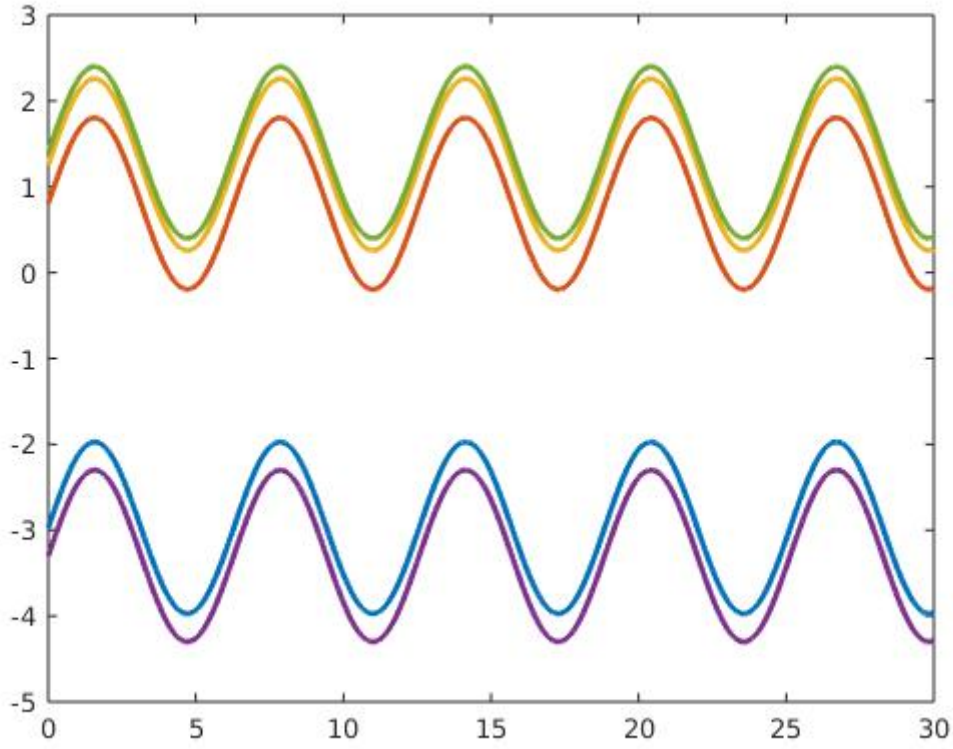
3. Control input plot for each link.



4. Plot of the adaptive estimates.



5. Plot of the parameter estimate errors.



- **Adaptive Update Law with** $\dot{\hat{\theta}} = \gamma \cdot \dot{Y}_d \cdot r$

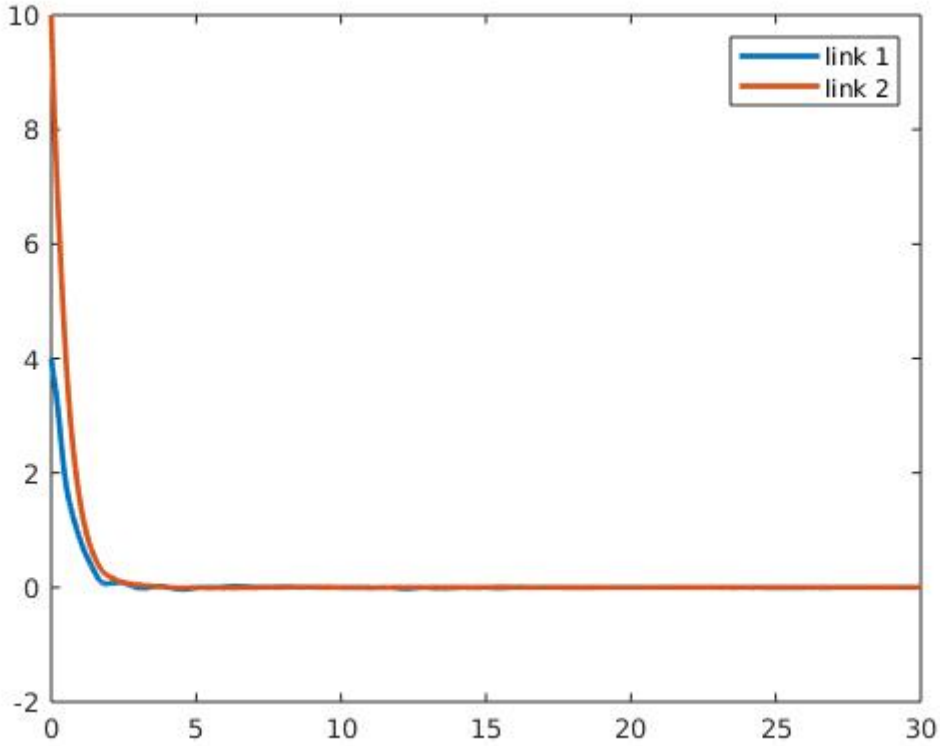
  1. Control gains.

  In this case, I try to add some regulation terms to $\dot{\hat{\theta}}$ just like the standard adaptive update law.

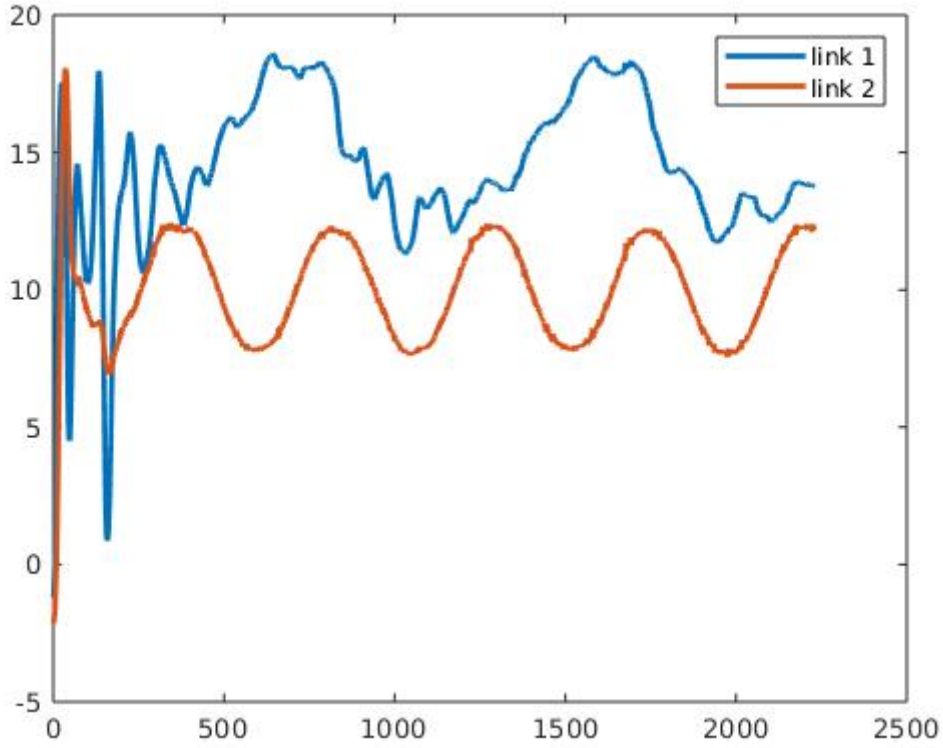
Control gains: ks = 10, $\alpha_1 = 2$, $\alpha_2 = 3$, $\beta = 1$;

$$\gamma = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

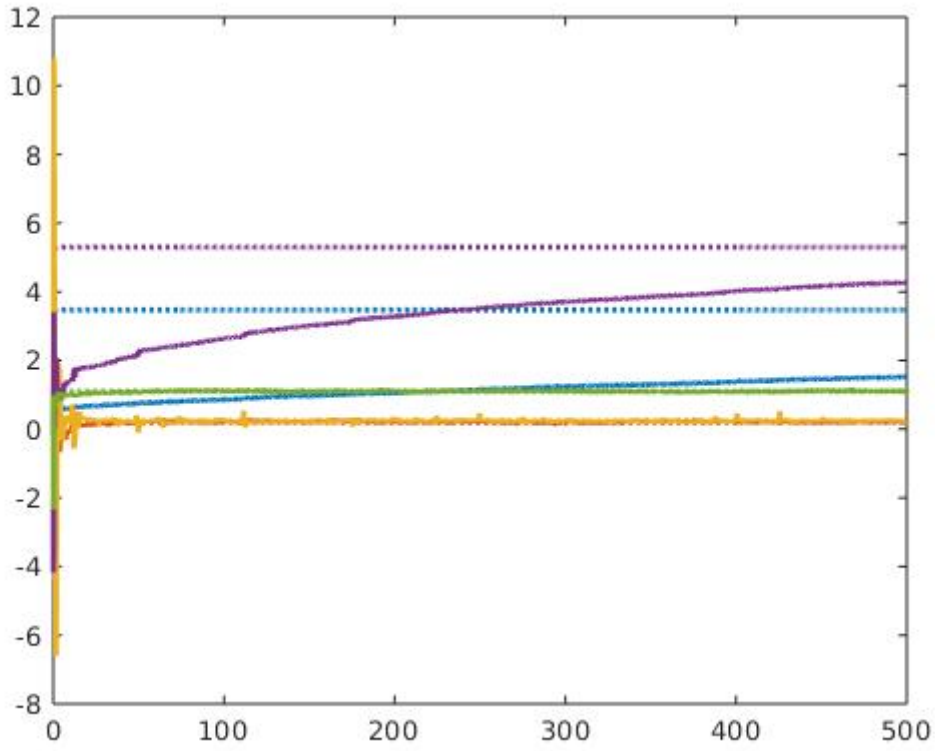
2. Tracking error plot for each link.



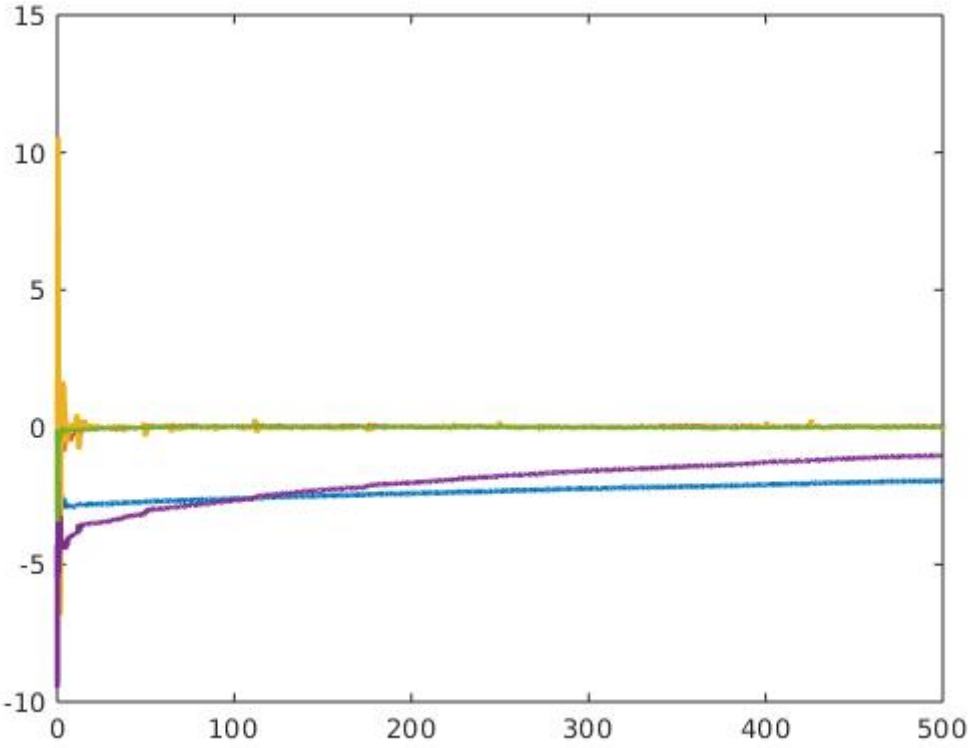
3. Control input plot for each link.



4. Plot of the adaptive estimates.



5. Plot of the parameter estimate errors.



- **Adaptive Update Law with** $\dot{\hat{\theta}} = \gamma \cdot \dot{Y}_d \cdot (e_1 + e_2 + r)$

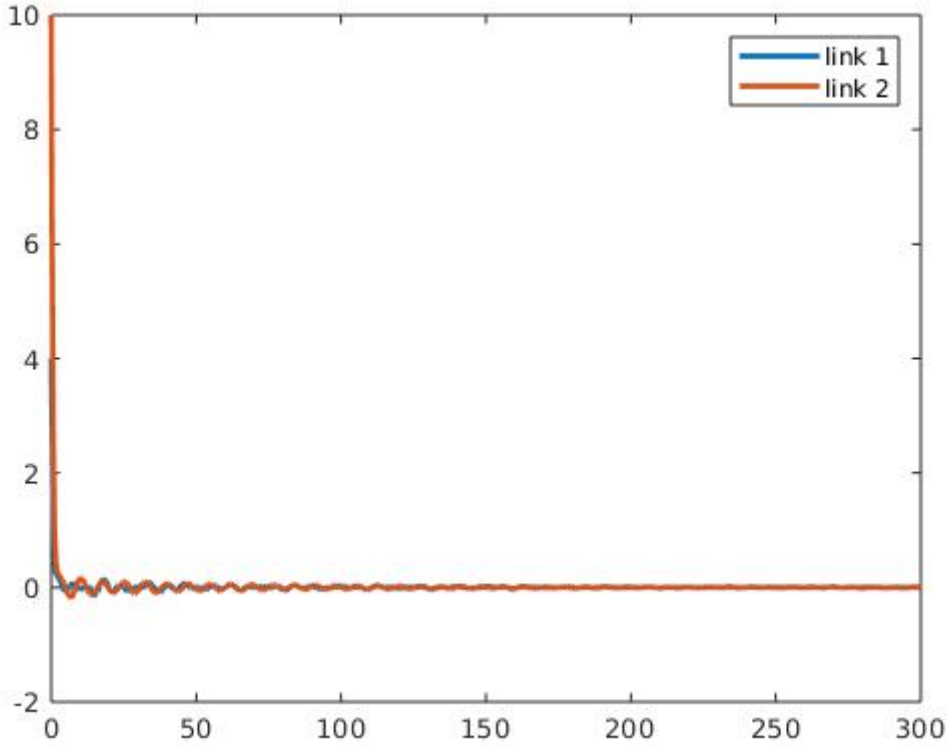
  1. Control gains.

  In this case, I put all the error terms into the adaptive law.

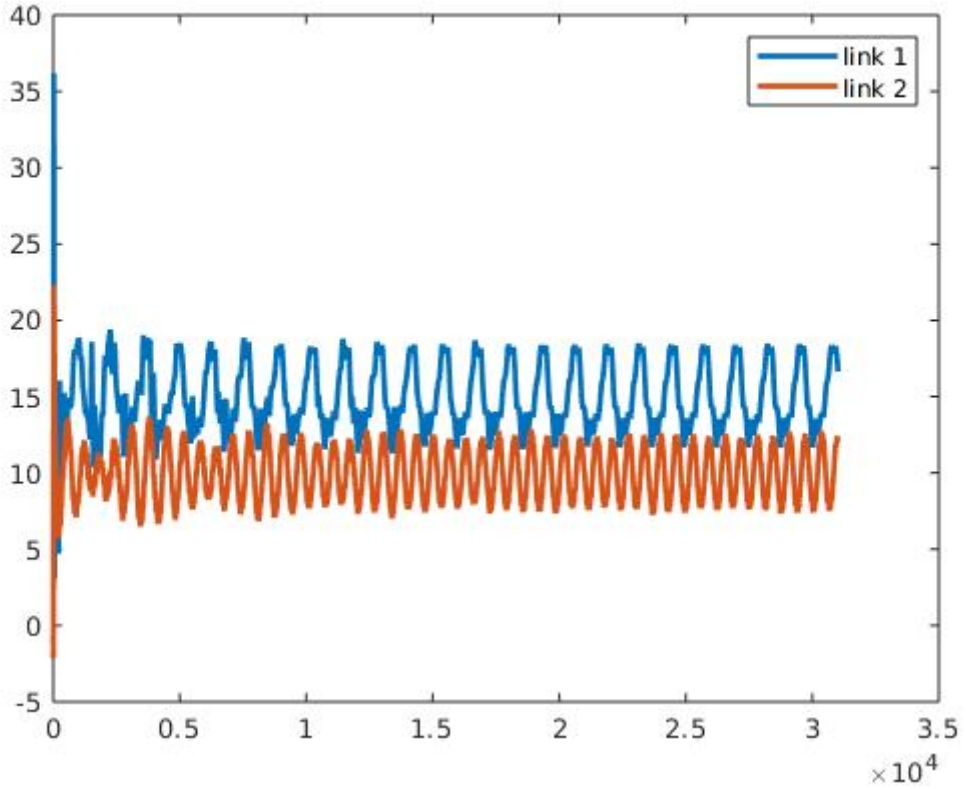
  Control gains: ks = 10, $\alpha_1 = 2$, $\alpha_2 = 3$, $\beta = 1$;

  $$\gamma = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

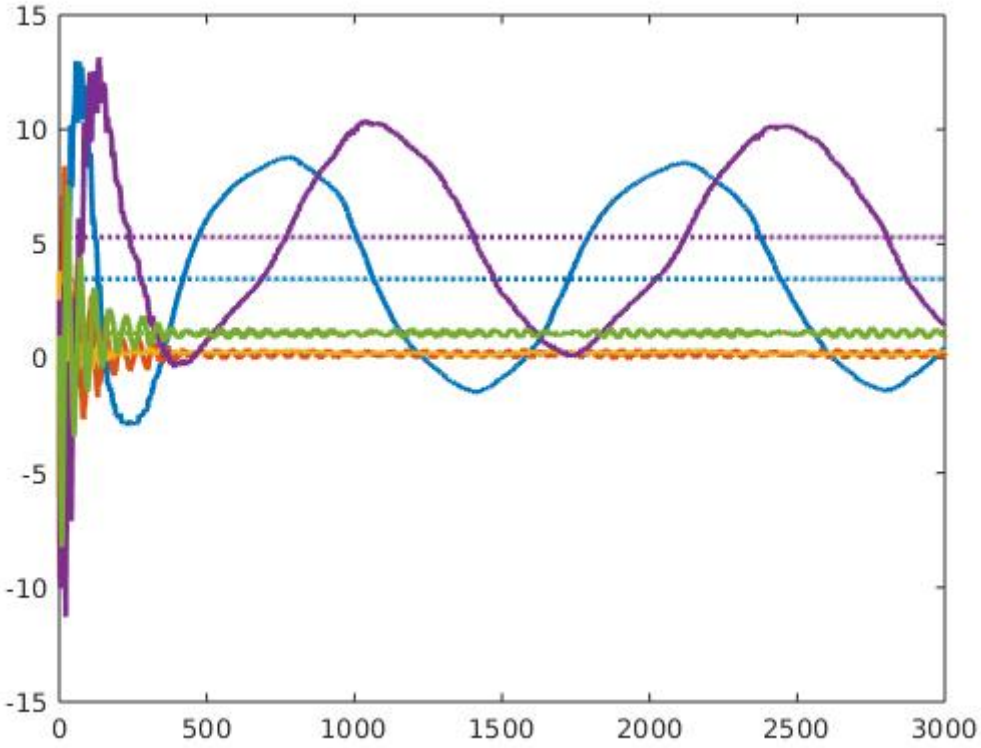
  2. Tracking error plot for each link.



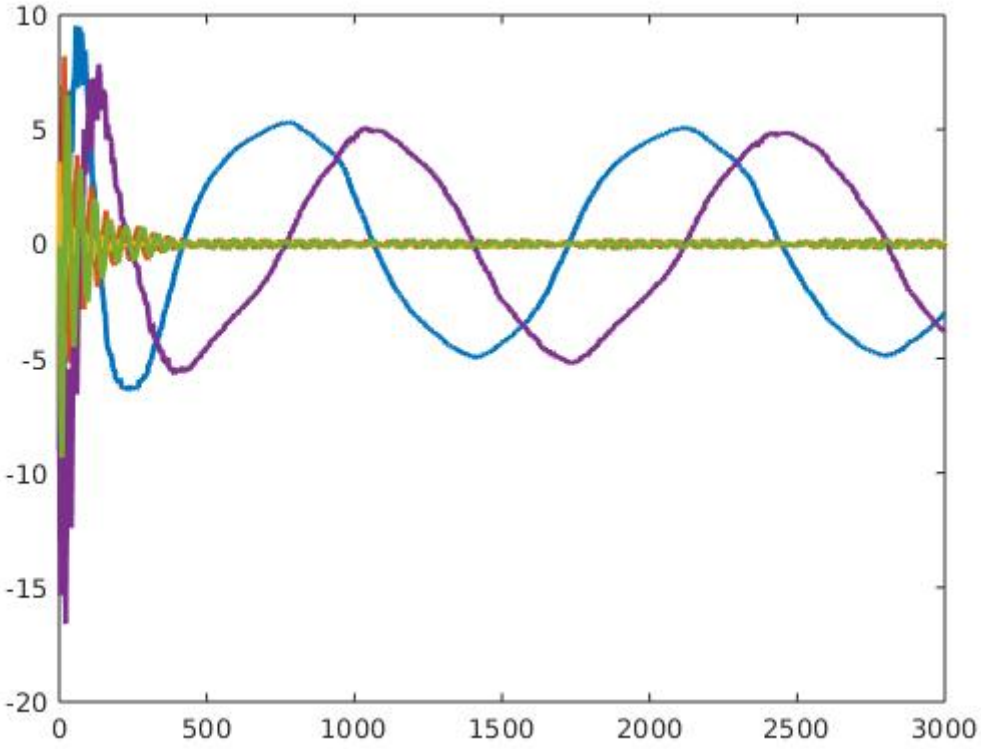
3. Control input plot for each link.



4. Plot of the adaptive estimates.



5. Plot of the parameter estimate errors.



- **Adaptive Update Law with** $\dot{\hat{\theta}} = \gamma \cdot \dot{Y}_d \cdot r + \gamma \cdot \dot{Y}_{df} \cdot r$

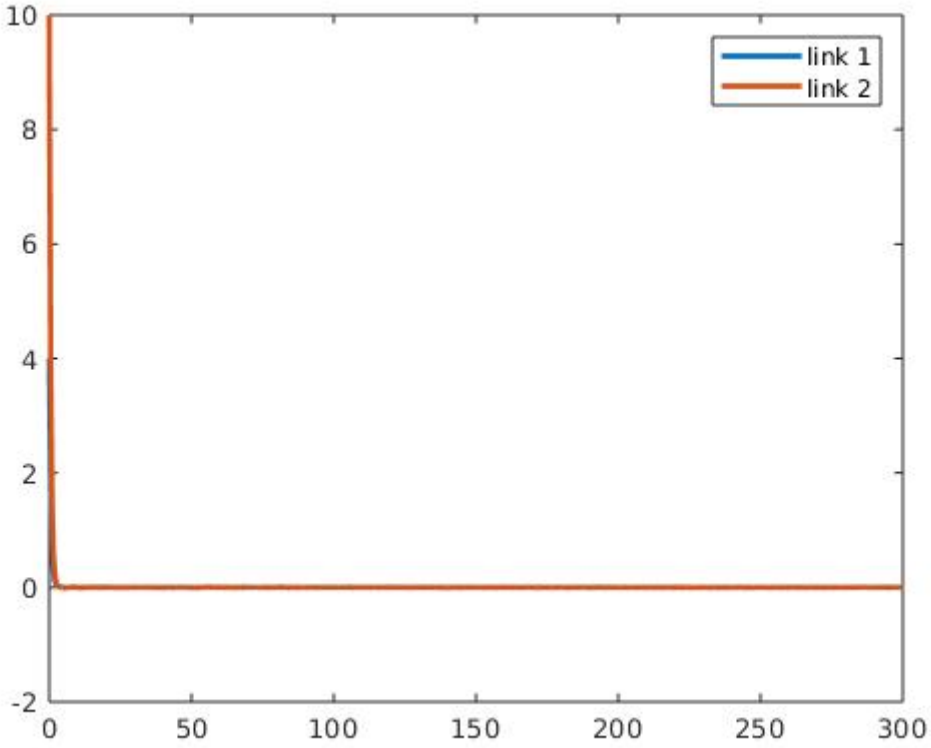
1. Control gains.

In this case, I utilize $\dot{Y}_{df}$, and see how it work.

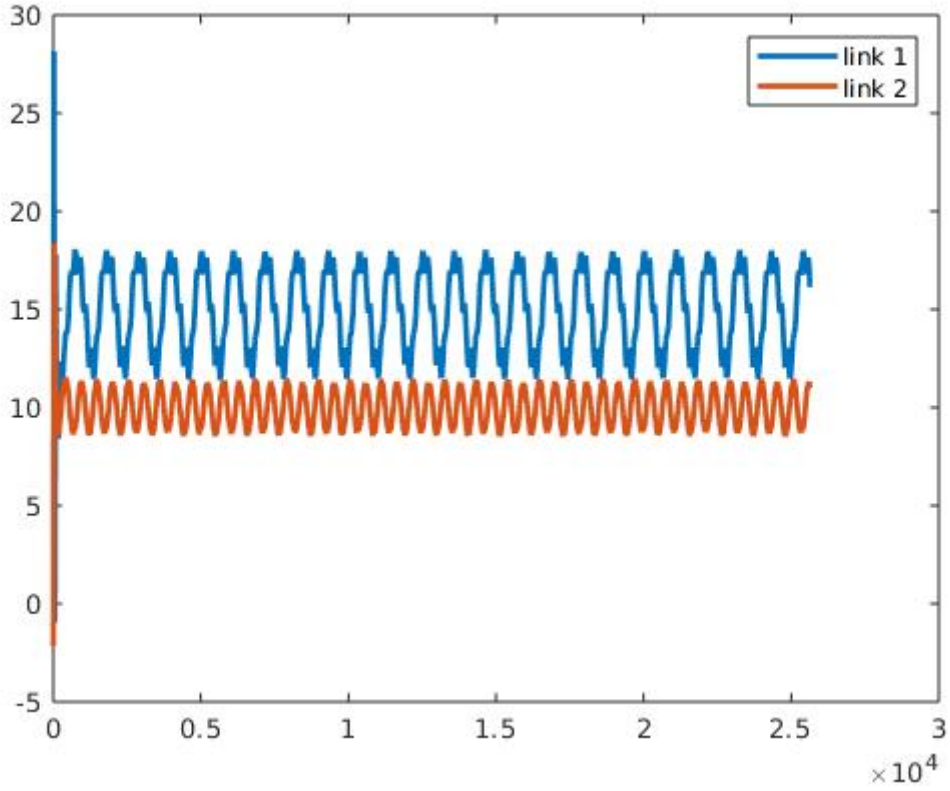
Control gains: ks = 10, $\alpha_1 = 2$, $\alpha_2 = 3$, $\beta = 1$;

$$\gamma = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

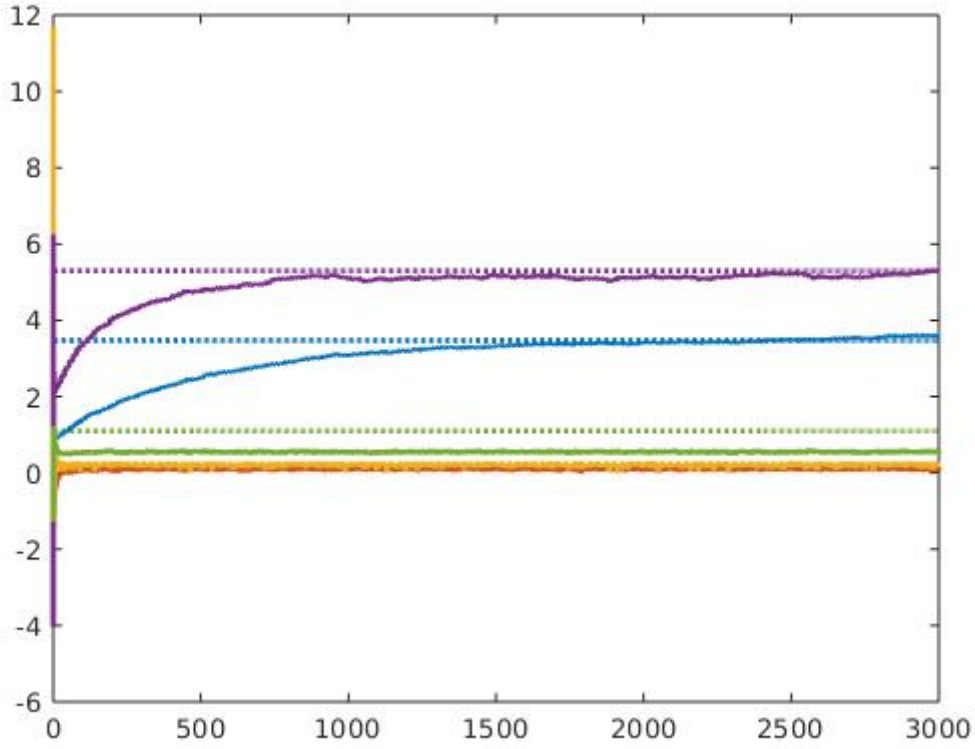
2. Tracking error plot for each link.



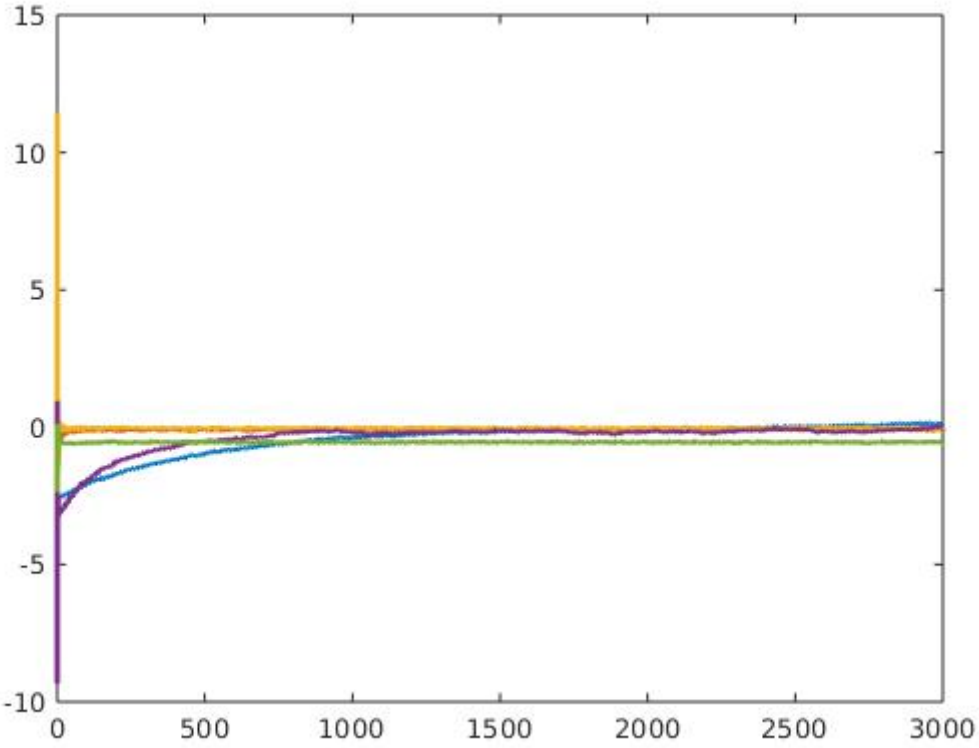
3. Control input plot for each link.



4. Plot of the adaptive estimates.



5. Plot of the parameter estimate errors.



- **(B) Discussion section**

  - **Differences in tuning the control gains/adaptations**

    As we are implementing a RISE-based modular controller, there could be many kinds of parameters can be twiddled in different kind of adaptation law. For my 4 implementation, there are mainly 5 gains/adaptations I can tune: α_1 , α_2 , β , k_s and γ .

    Take the standard adaptive update law for example:

    k_s is a term to provide $-k_s \cdot r^2$ in the lyapunov analysis, and thus k_s should be big enough to predominate other r^2 which may come from Young's inequality and etc. As we increse k_s , the tracking error will converge faster, however, the input will increase and estimates error would converge much slower. I think that's a trade-off.

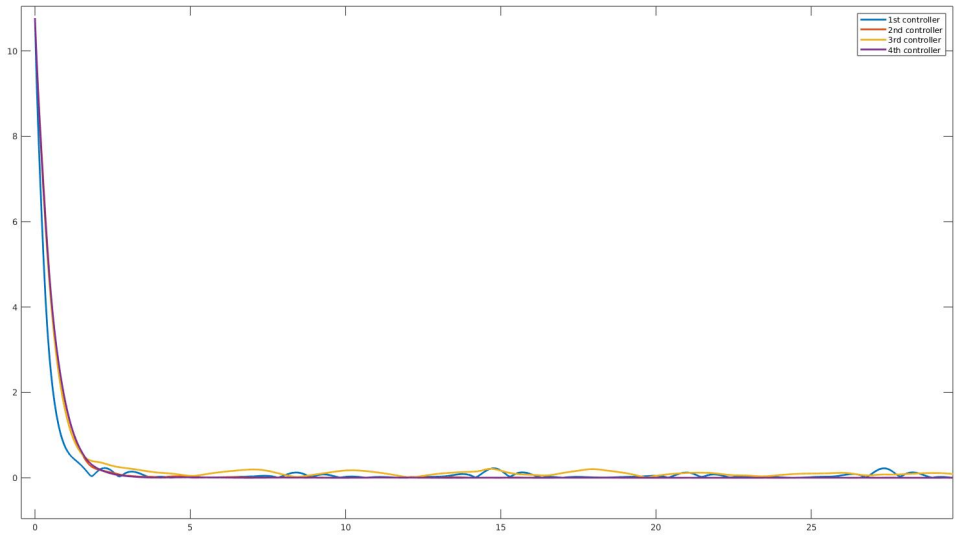
    α_1 and α_2 contribute the $-\alpha_1 e_1^2 - \alpha_2 e_2^2$ in the lyapunov analysis. When incresing α_1 and α_2 , we have to increse k_s too, otherwise our simulation could break down. With bigger α_1 and α_2 , we will have quicker convergence of both estimates error and tracking error. However, the input would increase.

    Twiddling β can actually increase the speed of estimate error convergence. In my implementation, I found when β is small, then the estimate error would converge faster.

    In my implementation, γ is a matrix. It's a learning rate hyperparameter. When we modify the specific element in this matrix, we can increse or decrease the corresponding parameter's estimation estimates' changing rates.

  - **Performance of the tracking error for each controller**

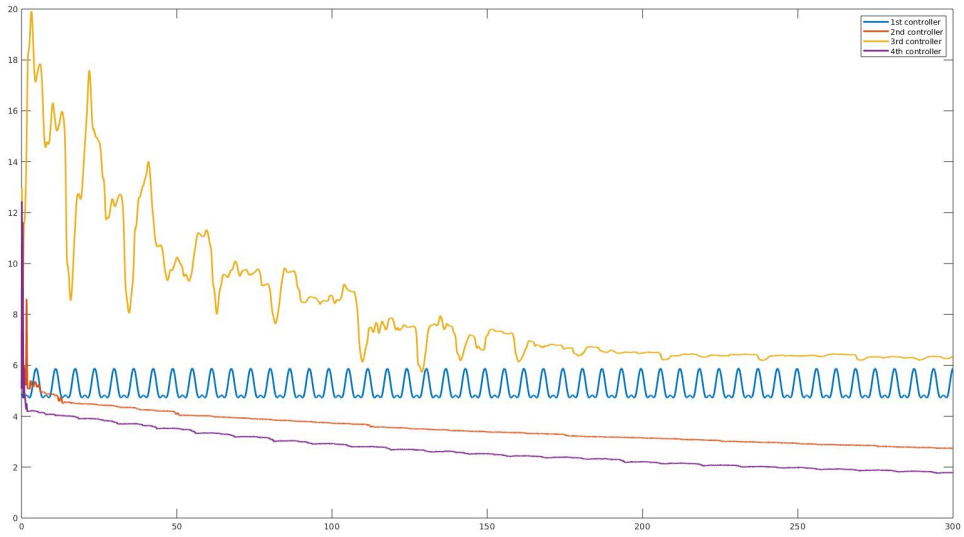
I compared the L2-norm of each controller's tracking error, $\|e\|$, as shown below:



As I expected, all the contoller has good performance on the tracking error as RISE-based modular controller design is designed to achieve asymptotic tracking after all. In fact, the ones with DCAL and only r in it has better performance, the one uses random sinusoidal as adaptive update law has slightly worse performance (blue), and the one uses not only r but also e_1 and e_2 has the worst performance (yellow).

- **Performance of the adaptation for each case**

I compared the L2-norm of each controller's parameter estimation error, $\left\|\tilde{\theta}\right\|$, as shown below:



Among my 4 controllers, the one uses composite adaptive update structure has the best performance. However, I have not used prediction error in this contoller. The one use tradional gradient update law has slightly worse performance, but if we put all the error terms along with the DCAL term, the performance is worst.

From the result of adaptation and tracking error. I can see that in RISE-based modular controller, the adaptation and tracking error performance have positive correlation. The better adaptation performance we have, the better tracking performance our controller can achieve.

I think the modular method give us the possibility to twiddle our controller like PID controller, and the way we modify the adaptation law can make a huge difference in performance. The RISE-based method also get us the idea to put r in the $\dot{\hat{\theta}}$.

Things can be improved: I think I can compare my controllers to the composite adaptive controller with RISE-term in **Project 2**, and see if I can design a better controller using the modular method.