EML 6351 Simulation Project 1

- **Introcuction**

In this project, we need to design our own traditional adaptive controller (with gradient adaptive update law), and 2 composite adaptive controllers (with gradient adaptive update law and least squares adaptive update law) for a two-link rigid revolute robot manipulator, to make sure the actual angular position of each joints can keep track of the desired angular position.
The robot manipulator model is as below:

$$\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix} = \begin{bmatrix} p_1+2p_3c_2 & p_2+p_3c_2 \\ p_2+p_3c_2 & p_2\end{bmatrix}\begin{bmatrix}\ddot{q}_1\\ \ddot{q}_2\end{bmatrix} + \begin{bmatrix}-p_3s_2\dot{q}_2 & -p_3s_2(\dot{q}_1+\dot{q}_2)\\ p_3s_2\dot{q}_1 & 0\end{bmatrix}\begin{bmatrix}\dot{q}_1\\ \dot{q}_2\end{bmatrix} + \begin{bmatrix}f_{d1} & 0\\ 0 & f_{d2}\end{bmatrix}\begin{bmatrix}\dot{q}_1\\ \dot{q}_2\end{bmatrix}$$

where:

$$q_{d1} = cos(0.5t)$$
$$q_{d2} = 2cos(t)$$

- **Derive the filter torque $\tau_f(t)$, the filtered regression matrix $Y_f(t)$ and the filtered desired regression matrix $Y_{df}(t)$.**

1. Filtered torque $\tau_f(t)$.

τ is known and designed by us, and $\tau_f(t)$ is f convolved with τ. f designed as $f = \beta e^{-\beta t}$. After Laplacian transform, we will have $\dot{\tau}_f = \beta * \tau - \beta * \tau_f$.

2. Filtered regression matrix $Y_f(t)$.

First, we have $Y_f\theta = f * m(q)\ddot{q} + f * N(q,\dot{q})$, and after some mathematics processes, we can derive them into a form as below:

$$Y_f\theta = \Omega + \beta m(q(t))\dot{q}(t) - \beta e^{-\beta t} m(q(0))\dot{q}(0) + W$$

with $\dot{\Omega} = -\beta\Omega - \beta^2 m(q(t))\dot{q}(t)$, and $\dot{W}(t) = -\beta W(t) + \beta(-\dot{m}(q(t))\dot{q}(t) + N(q(t),\dot{q}(t))$.
After initialize Ω and W, we can get $Y_f\theta$, and factor out the theta, we can have Y_f.

3. filtered desired regression matrix $Y_{df}(t)$.

We can get $Y_{df}(t)$ as follow:

$$Y_{df} = f * Y_d$$
$$\dot{Y_{df}} = \beta * Y_d - \beta * Y_{df}$$

Where Y_d is derived beforehand:

$Y_{d11} = \ddot{q_{d1}}$
$Y_{d12} = \ddot{q_{d2}}$
$Y_{d13} = 2cos(d_2)\ddot{q_{d1}} + cos(d_2)\ddot{q_{d2}} - sin(d_2)(\dot{q_{d1}} + \dot{q_{d2}})(\dot{q_{d2}})$
$Y_{d14} = \dot{q_{d1}}$
$Y_{d15} = 0$
$Y_{d21} = 0$
$Y_{d22} = \ddot{q_{d1}} + \ddot{q_{d2}}$
$Y_{d23} = cos(q_{d2})\ddot{q_{d1}} + sin(q_{d2})\dot{q_{d1}}^2$
$Y_{d24} = 0$
$Y_{d25} = \dot{q_{d2}}$
$Y_d = \begin{bmatrix} Y_{d11} & Y_{d12} & Y_{d13} & Y_{d14} & Y_{d15} \\ Y_{d21} & Y_{d22} & Y_{d23} & Y_{d24} & Y_{d25} \end{bmatrix}$
β is designed by us, so we can intergrate $\dot{Y_{df}}$ with initilization to get $Y_{df}(t)$.

- **Derive and simulate the controllers**

(a) Traditional adaptive controller with gredient adaptive update law.

The code is included in 'traditional.m', and to implement it, just hit run button or implement the traditional function in command window.

(b) Composite adaptive controller with gredient adaptive update law.

The code is included in 'torquefiltermethod.m', and to implement it, just hit run button or implement the torquefiltermethod function in command window.

(c) Composite adaptive controller with least squares adaptive update law.

The code is included in 'leastsquaremethod.m', and to implement it, just hit run button or implement the leastsquaremethod function in command window.

(d) turn in the code All the codes are in src file and can be implemented by themselves in Matlab (no dependent needed).

- **Typed report**

(a) Dynamics model.

The dynamics model is:

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} p_1 + 2p_3c_2 & p_2 + p_3c_2 \\ p_2 + p_3c_2 & p_2 \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} -p_3s_2\dot{q}_2 & -p_3s_2(\dot{q}_1 + \dot{q}_2) \\ p_3s_2\dot{q}_1 & 0 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} + \begin{bmatrix} f_{d1} & 0 \\ 0 & f_{d2} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}$$

where $p_1, p_2, p_3, f_{d1}, f_{d2}$ are unknown positive scalar constans, and $s_2 = sin(q_2), c_2 = cos(q_2)$, and:
$q_{d1} = cos(0.5t)$
$q_{d2} = 2cos(t)$

(b) Problem definition and open-loop error system development.

The project's objective is to let q keep track of q_d.

(c) Control design (including adaptive update law) and closed-loop error system development

This dynamics model's states are up to sescond order derivative, so we not only design e as $e = q - q_d$, we also should have a second term r as $r = \alpha * e + \dot{e}$, and when r goes to 0, we can also have e goes to 0. With r, we can have $\ddot{q}$ in $\dot{V}$, thus put τ into lyapunov analysis. For adapative update law, we should have a term $\tilde{\theta}$ in V , so when can at least bound $\tilde{\theta}$ after our lyapunov analysis.

(d) Stability analysis of each controller.

For all the 3 controllers, we have the same Lyapunov function:

$$V = \frac{1}{2}r^t m r + \frac{1}{2}e^t e + \frac{1}{2}\tilde{\theta}^t \gamma^{-1}\tilde{\theta}$$

- For the traditional controllers, we design:

$$\dot{\hat{\theta}} = -\gamma * Y_d^t * r$$
$$\tau = -K * r + Y_d * \hat{\theta}$$

With doing so, we can finally have:

$$\dot{V} = r^t N - \alpha e^2 - k r^2$$

where $N = \rho \left\| Z \right\|$ and ρ is a function of Z with $Z = [r; e]$, specifically we can bound the parameters in this function by Mean Value Theorem. With tuning α and k , we can have:

$$\dot{V} \leqslant -\varphi \left\| Z \right\|^2$$

In the end, we can prove that rr and e go to 0 with Lemma.

- For Composite adaptive controller with gredient adaptive update law, we design:

$$\dot{\hat{\theta}} = -\gamma * Y_d^t * r + \gamma * Y_{df}^t * \epsilon$$
$$\epsilon = \tau_f - Y_{df} * \hat{\theta}$$

After deriving $\dot{V}$, We can have a similar form like the traditional method, but we will have one more term as follow in the $\dot{V}$ and that is $-(1 - \frac{\beta * C}{2}) \left\| Y_{df}\hat{\theta} \right\|^2$, and after tuning we have make sure this term is nagative, thus we can have the same result like the tradition method, but we can now also have $Y_{df}\hat{\theta}$ is of L2 bound.

- For Composite adaptive controller with least squares adaptive update law, we design:

$$\dot{\hat{\theta}} = -P * Y_d^t * r + P * Y_{df}^t * \epsilon$$
$$\epsilon = \tau_f - Y_{df} * \hat{\theta}$$

$$\dot{P} = -P * Y_{df}^{t} * Y_{df} * P$$

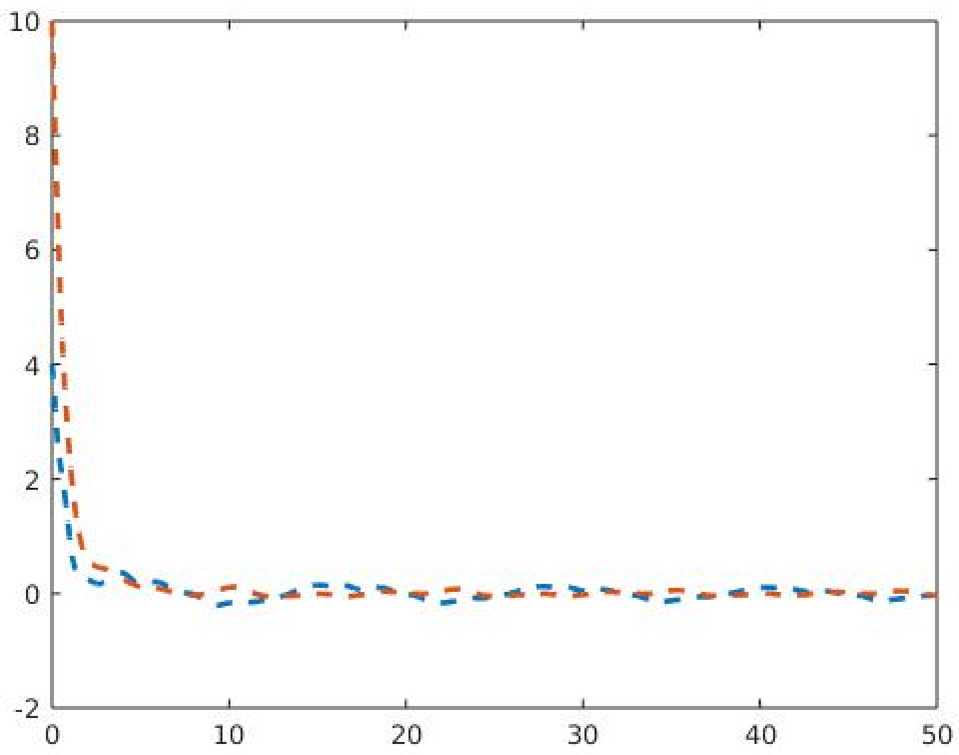
where we initialize P as a postive definite matrix.

After deriving $\dot{V}$, we now have the very similar form of the composite adaptive controller with gredient adaptive update law method, and the 1 in $-(1 - \frac{\beta * C}{2}) \left\| Y_{df}\hat{\theta} \right\|^2$ term will become $\frac{1}{2}$. So we will have the same conclusion as the last one.

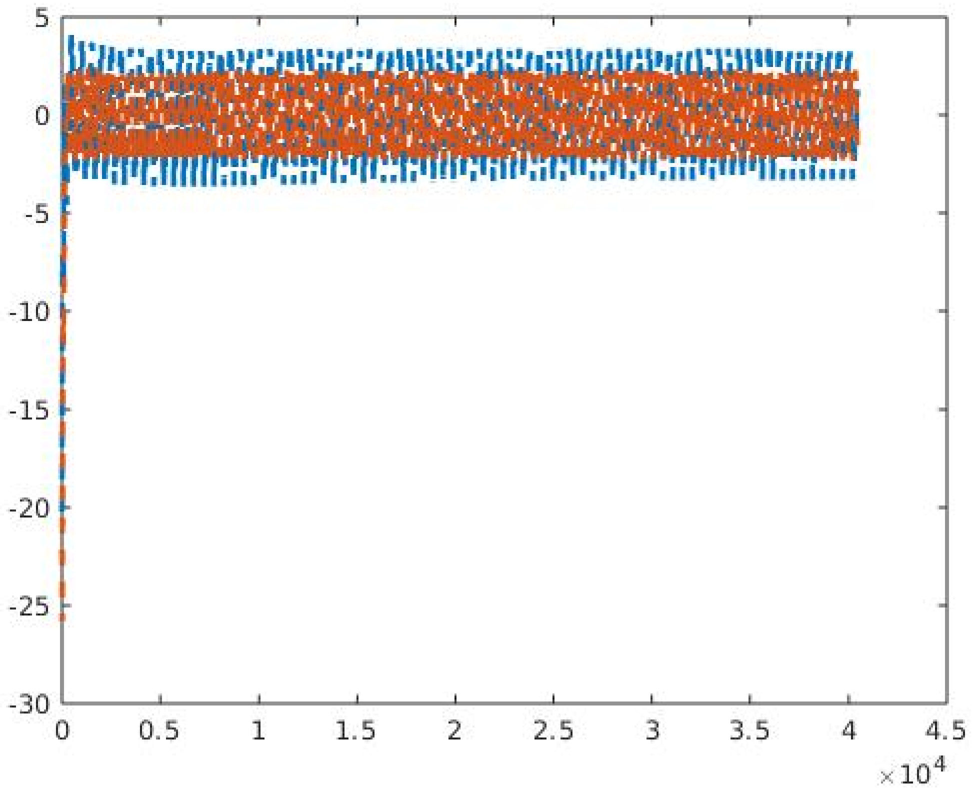
(e) Stability analysis of each controller.

For the traditional controllers:

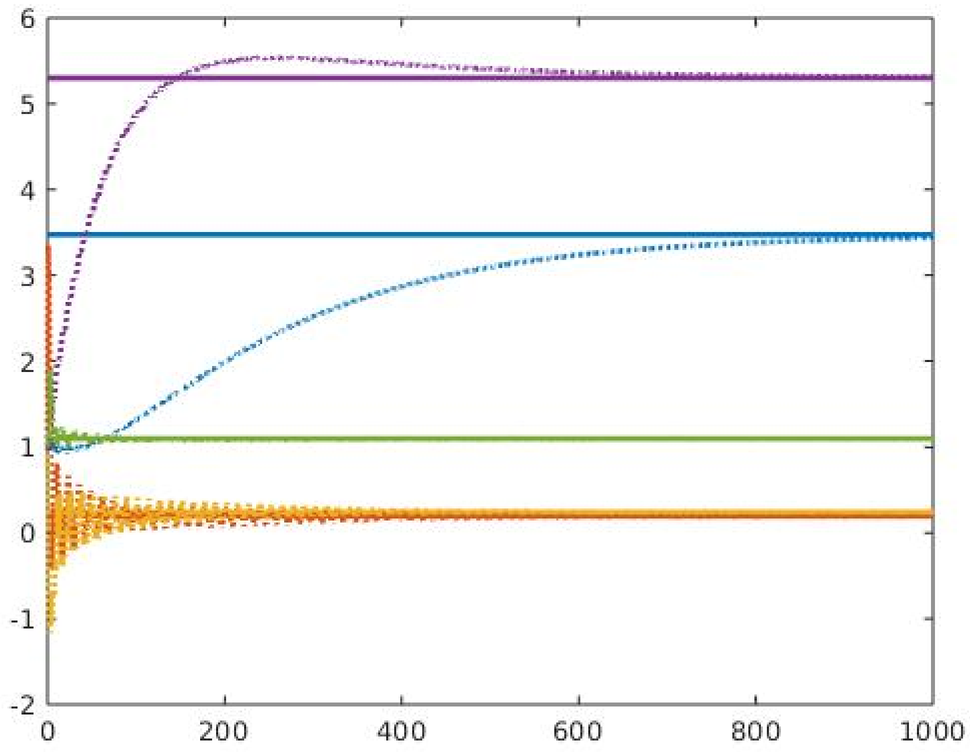
- Control gain and their value:
  My control gain K = 5, α = 1.5, and
  and
  $$\gamma = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$
- Tracking error plot for each link:
  For e:



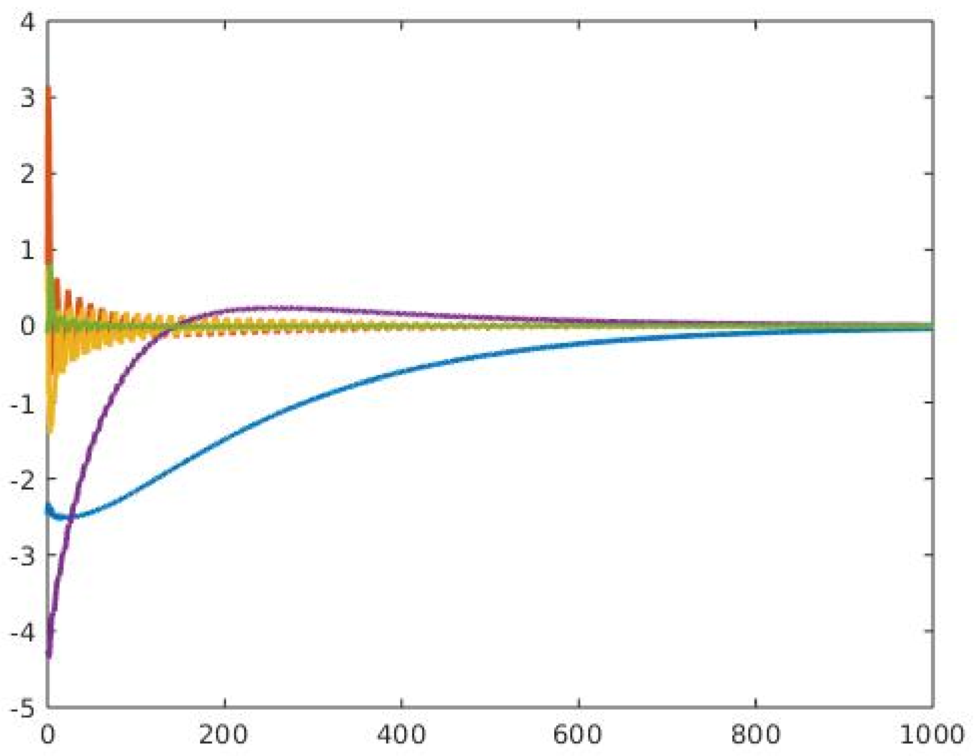
- Control input plot



- Plot of adaptive estimates



- Plot of the parameter estimate errors($\hat{\theta}$)

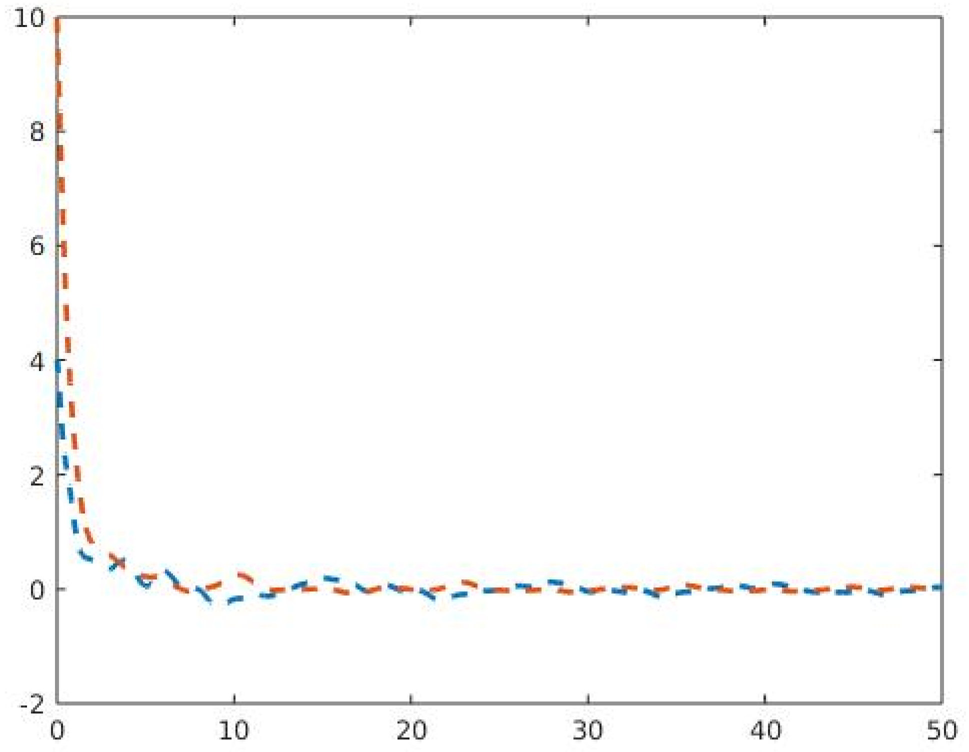


For the Composite adaptive controller with gredient adaptive update law:

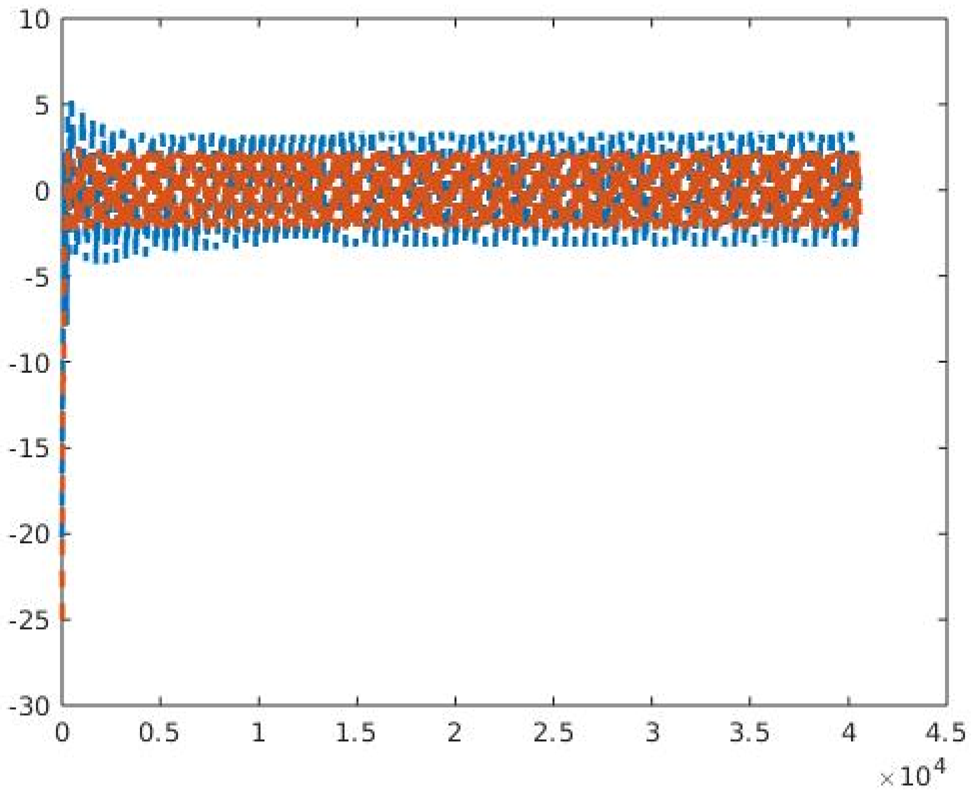
- Control gain and their value:
  As for comparison, I use the same control gain as K = 5, α = 1.5, β = 1, and

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

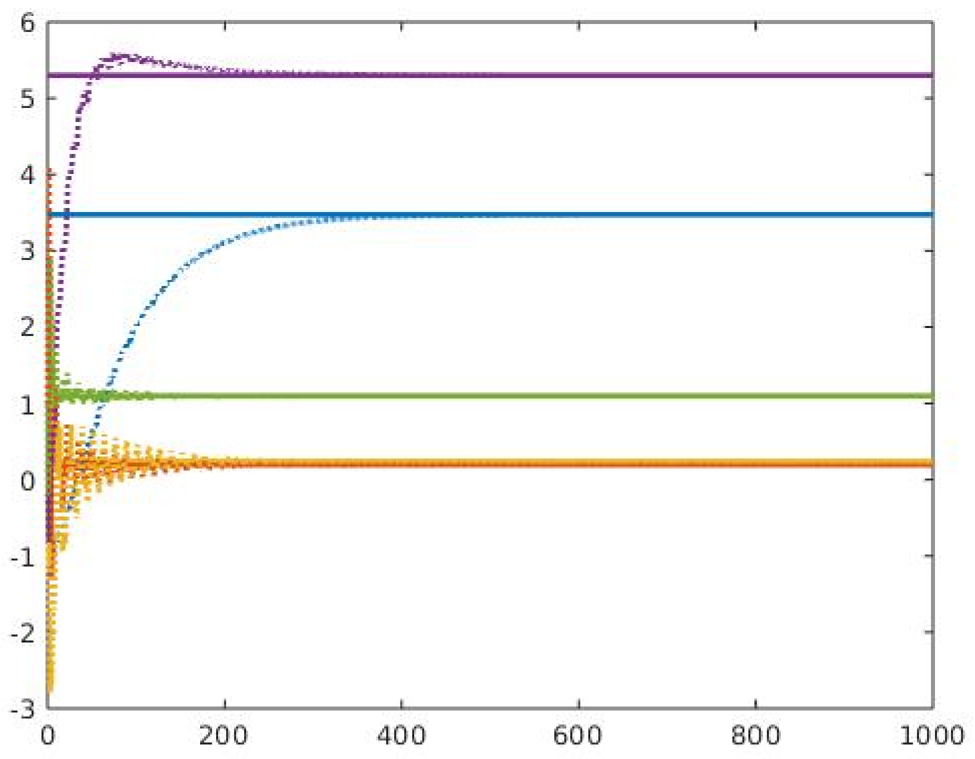
- Tracking error plot for each link:
  For e:



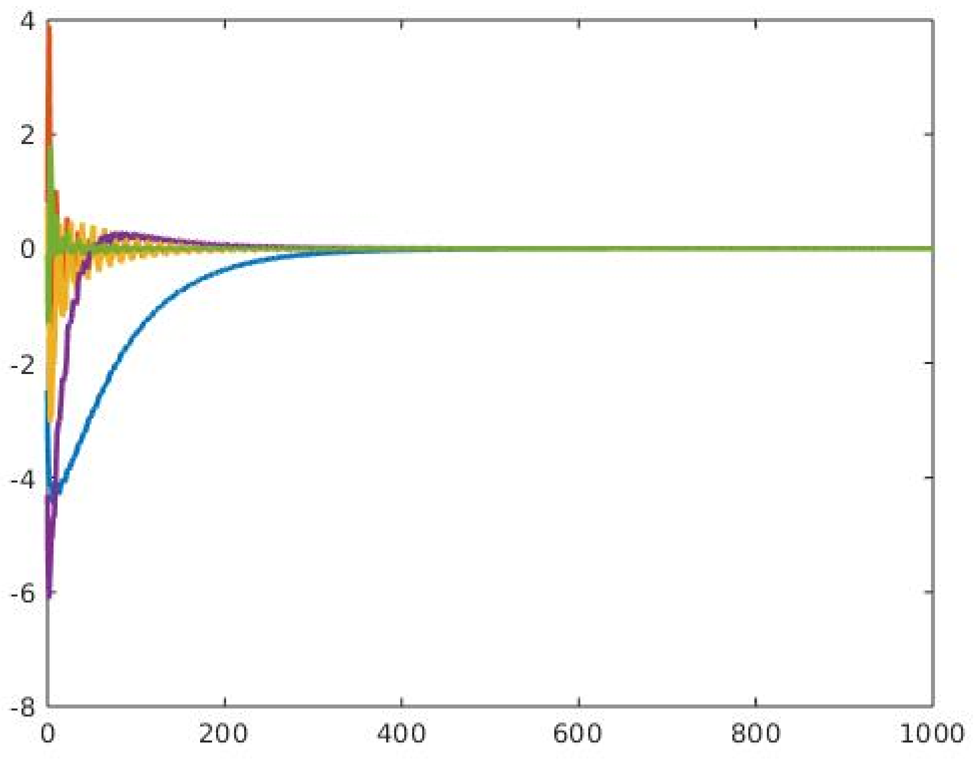
- Control input plot



- Plot of adaptive estimates



- Plot of the parameter estimate errors($\hat{\theta}$)



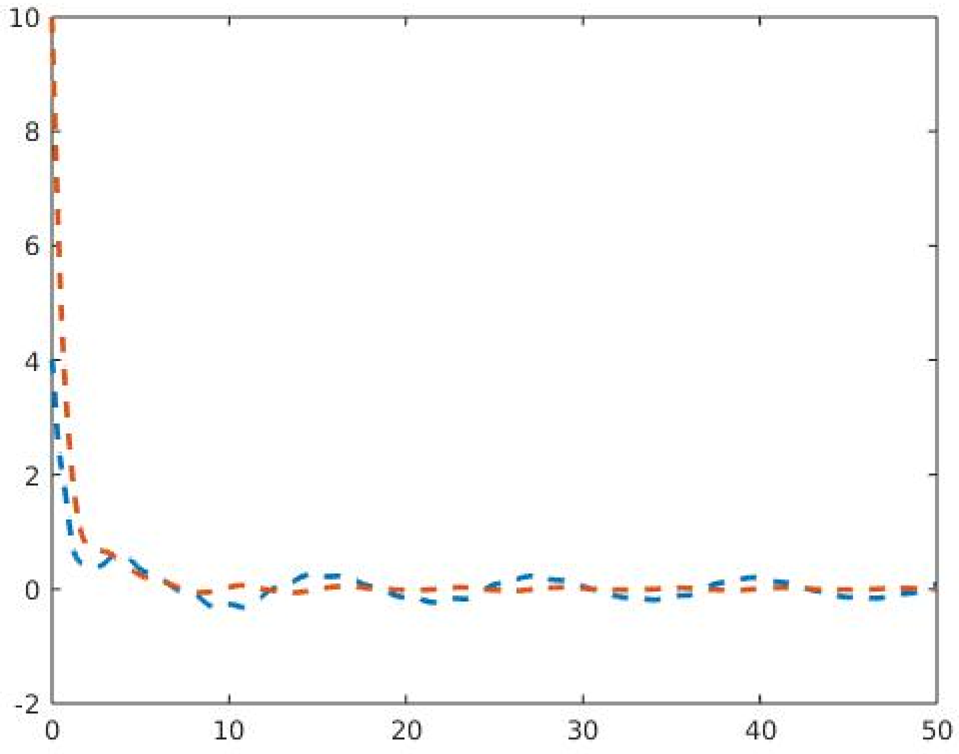
For the Composite adaptive controller with least squares adaptive update law:

- Control gain and their value:
  As for comparison, I use the same control gain as K = 5, α = 1.5, β = 1, and

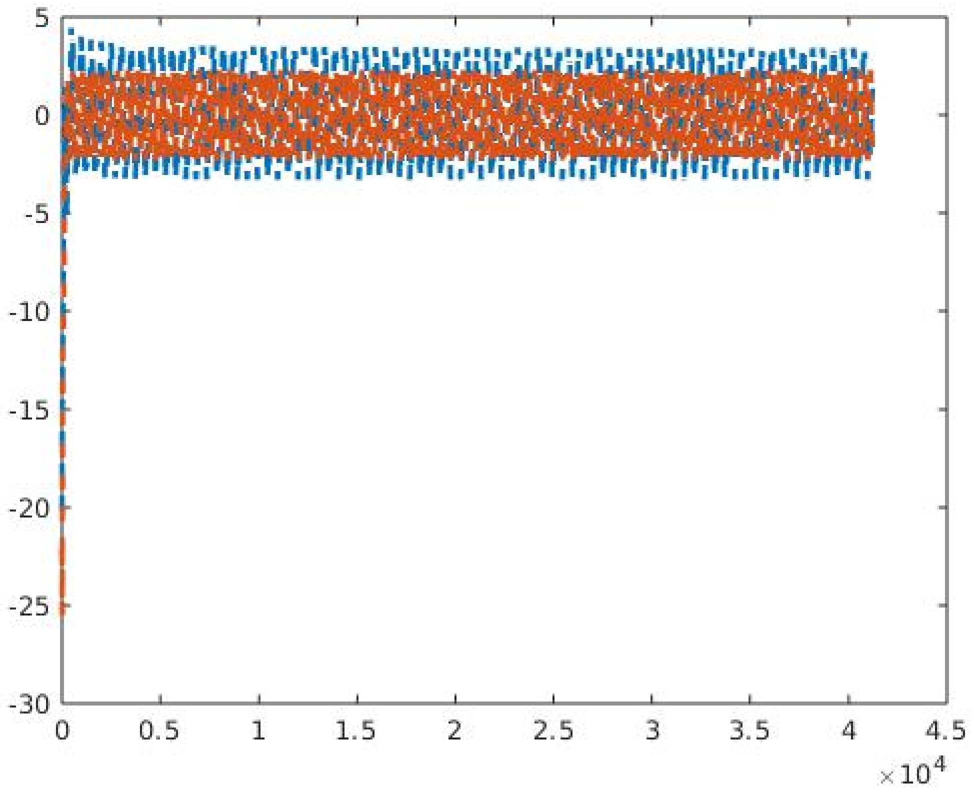
$$P0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

- Tracking error plot for each link:

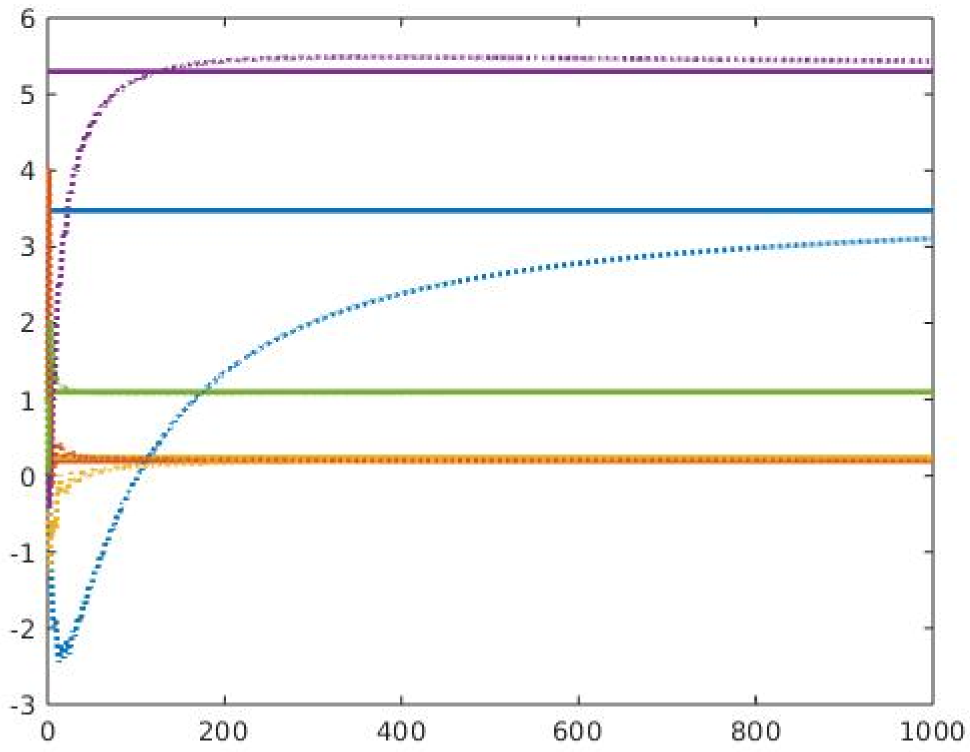
  For e:



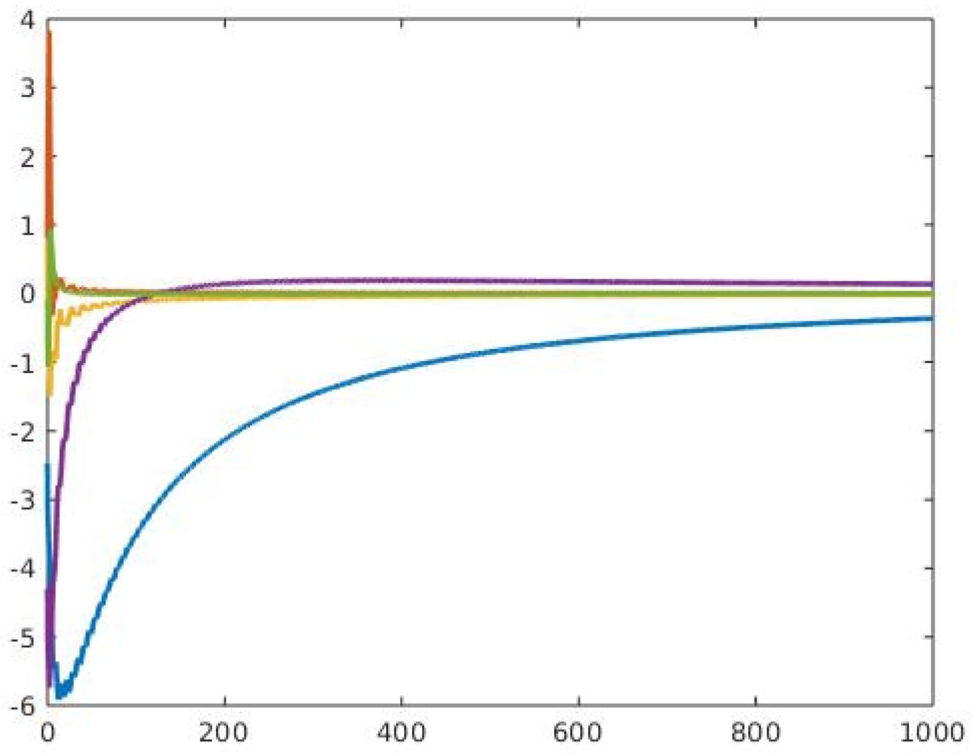
- Control input plot



- Plot of adaptive estimates



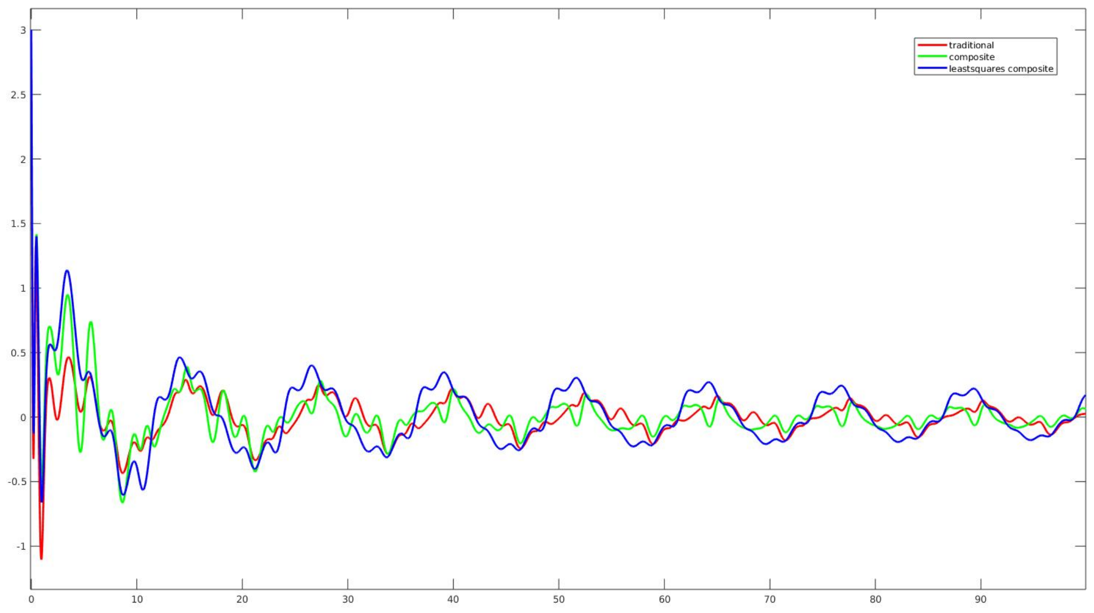
- Plot of the parameter estimate errors($\hat{\theta}$)



(f) Discution section:

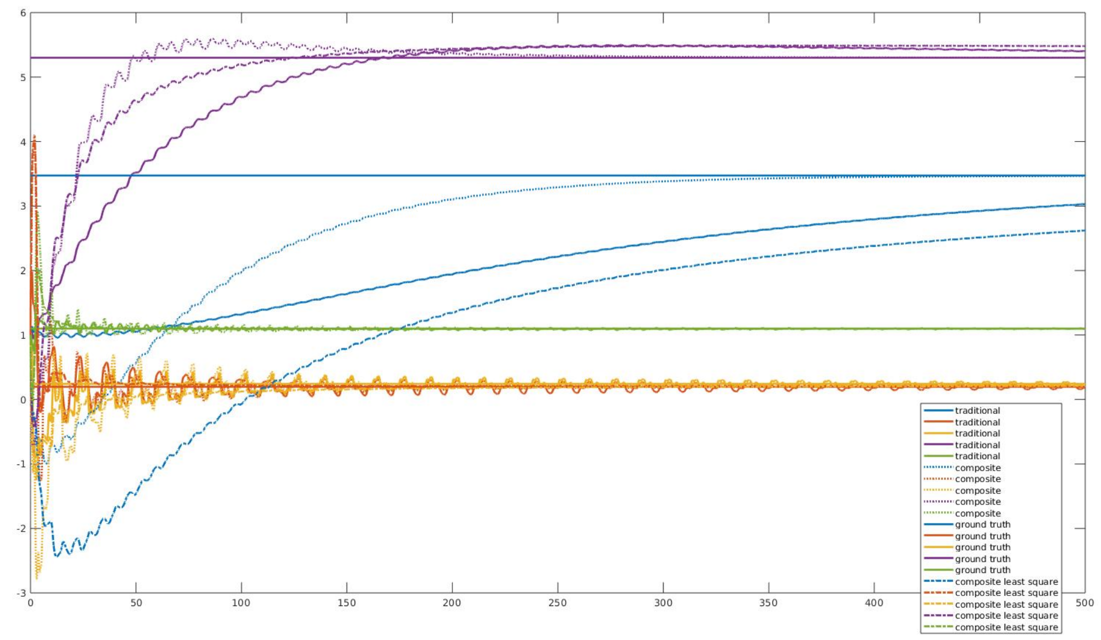
1. Differences in tunning the control gains/adaptation gains.
When increasing control gain K, we can clearly notice that the tracking error would converge faster, but it slow down the convergence of parameter estimate errors, and when increasing adaptation gains α , tracking error would converge faster, and it may slow down the parameter estimate errors, I can see from my experiment that it suppress the overshoot of parameter estimate errors. When incresing the eigenvalue of the γ , it increase the convergence speed of both the tracking error and the parameter estimate errors, but I think it would increse the input value.

2. Performance of the tracking error for each controller.
I extract the first joint's tracking error from the 3 controller:



Of the same control gain, I can see the traditional controller and the Composite adaptive controller with gredient adaptive update law have the similar performance, and the the Composite adaptive controller with least squares adaptive update law is not as good as the first two. However, overall their performance are not quite different.

3. Performance of the adaptation for each case.



For adaptation, the two composite controller both outperform the traditional one, but I can't see which one of these two can perform better that the other one. I think it's because they have different adaptation parameter as one using γ, and the other one has P0 which is initialized by us.