Adaptive Control of a Two-Link Revolute Chain Under Gravity

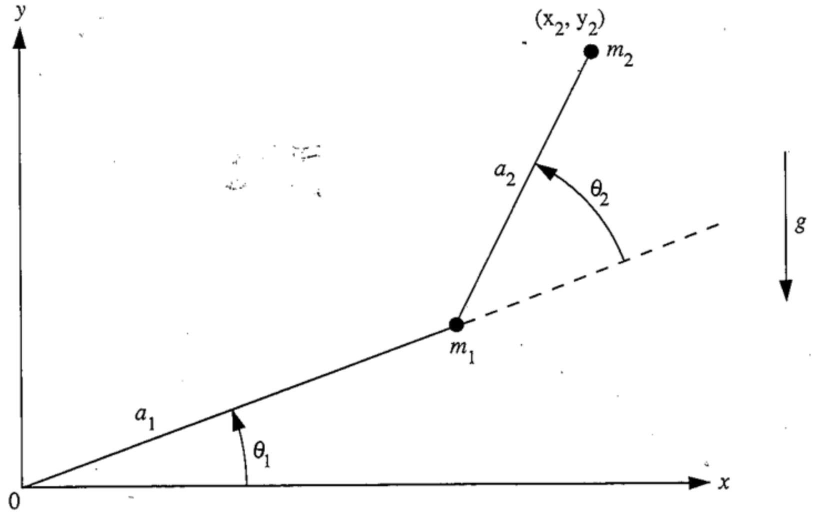
Composite and Concurrent Methods

Jason Nezvadovitz

Dynamics

Model

The system is modeled as two rigid links with lumped-parameter masses at their endpoints, as shown in the figure below. The derivation of the dynamics can be found in a number of introductory mechanics and robotics texts.



Following the derivation outlined in “Robot Manipulator Control Theory and Practice” by Frank Lewis, we arrive at the following dynamics.

$$\begin{bmatrix} (m_1 + m_2)\, a_1^2 + m_2 a_2^2 + 2m_2 a_1 a_2 \cos\theta_2 & m_2 a_2^2 + m_2 a_1 a_2 \cos\theta_2 \\ m_2 a_2^2 + m_2 a_1 a_2 \cos\theta_2 & m_2 a_2^2 \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix}$$

$$+ \begin{bmatrix} -m_2 a_1 a_2 (2\dot{\theta}_1 \dot{\theta}_2 + \dot{\theta}_2^2) \sin\theta_2 \\ m_2 a_1 a_2 \dot{\theta}_1^2 \sin\theta_2 \end{bmatrix} + \begin{bmatrix} (m_1 + m_2)\, g a_1 \cos\theta_1 + m_2 g a_2 \cos(\theta_1 + \theta_2) \\ m_2 g a_2 \cos(\theta_1 + \theta_2) \end{bmatrix}$$

$$= \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}$$

where τ is the motor torque at each joint. If we define a position state vector,

$$q \coloneqq \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

and control input vector,

$$u := \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}$$

then these dynamics can be expressed concisely as a typical Euler-Lagrange system,

$$M(q)\ddot{q} + V(q, \dot{q}) + G(q) = u$$

We will consider every system parameter (m_1, m_2, a_1, a_2, and g) to be unknown / immeasurable. With this limitation, it becomes reasonable to use such a simple dynamic model in the first place: a distributed mass model for this system can be reduced to an equivalent lumped mass model, so treating all of the lumped mass parameters as unknown provides the realism of not being able to easily compute the lumped mass equivalent of a real arm.

Additionally, we notice that the entire left-hand-side of the dynamics is linear in the unknown parameters. We use a symbolic manipulation tool like MatLab to expand out the dynamics into its two scalar equations and sift through the terms seeking out the minimum number of unique nonlinear combinations of the parameters (weights). Defining the vector of these weights as,

$$\Theta := \begin{bmatrix} (m_1 + m_2)ga_1 \\ (m_1 + m_2)a_1^2 \\ m_2 g a_2 \\ m_2 a_1 a_2 \\ m_2 a_2^2 \end{bmatrix}$$

allows us to express the dynamics as,

$$Y_u \Theta = u$$

where

$$Y_u := \begin{bmatrix} \cos(\theta_1) & \ddot{\theta}_1 & \cos(\theta_1 + \theta_2) & \left(2\ddot{\theta}_1 + \ddot{\theta}_2\right)\cos(\theta_2) - (\dot{\theta}_2^2 + 2\dot{\theta}_1\dot{\theta}_2)\sin(\theta_2) & \ddot{\theta}_1 + \ddot{\theta}_2 \\ 0 & 0 & \cos(\theta_1 + \theta_2) & \ddot{\theta}_1\cos(\theta_2) + \dot{\theta}_1^2\sin(\theta_2) & \ddot{\theta}_1 + \ddot{\theta}_2 \end{bmatrix}$$

This matrix is a function of the state, but also of the state derivative, which is assumed to not be directly measurable (reasonable for a robotic arm that only has typical joint encoders).

<u>Error System</u>

Our objective is to design the control input u such that q asymptotically tracks some desired trajectory q_d. Thus we define the following tracking error,

$$e := q_d - q$$

We then define a filtered error signal with a tunable gain (diagonal 2x2 matrix) α,

$$r \coloneqq \dot{e} + \alpha e$$

so that our objective is,

$$\lim_{t \to \infty} r = 0$$

which implies that e and $\dot{e}$ go to zero as well. The time derivative of r is,

$$\dot{r} = \ddot{e} + \alpha \dot{e} = \ddot{q}_d - \ddot{q} + \alpha \dot{e}$$

After multiplying through by M,

$$M\dot{r} = M\ddot{q}_d - M\ddot{q} + M\alpha\dot{e}$$

we can introduce the dynamics through the $M\ddot{q}$ term,

$$M\dot{r} = M\ddot{q}_d + M\alpha\dot{e} + V + G - u$$

Using the same weight vector Θ as defined previously, we can express this equation as,

$$M\dot{r} = Y\Theta - u$$

where Y (different from Y_u) is a function of only the measurable state,

$$Y \coloneqq \begin{bmatrix} \cos(\theta_1) & \ddot{\theta}_{1_d} + \alpha_1\dot{e}_1 & \cos(\theta_1 + \theta_2) & (2\ddot{\theta}_{1_d} + \ddot{\theta}_{2_d} + 2\alpha_1\dot{e}_1 + \alpha_2\dot{e}_2)\cos(\theta_2) - \dot{\theta}_2(2\dot{\theta}_1 + \dot{\theta}_2)\sin(\theta_2) & \ddot{\theta}_{1_d} + \ddot{\theta}_{2_d} + \alpha_1\dot{e}_1 + \alpha_2\dot{e}_2 \\ 0 & 0 & \cos(\theta_1 + \theta_2) & \left(\ddot{\theta}_{1_d} + \alpha_1\dot{e}_1\right)\cos(\theta_2) + \dot{\theta}_1^2\sin(\theta_2) & \ddot{\theta}_{1_d} + \ddot{\theta}_{2_d} + \alpha_1\dot{e}_1 + \alpha_2\dot{e}_2 \end{bmatrix}$$

Finally, we define an adaptive estimation error,

$$\widetilde{\Theta} \coloneqq \Theta - \widehat{\Theta}$$

$$\dot{\widetilde{\Theta}} = -\dot{\widehat{\Theta}}$$

where $\widehat{\Theta}$ is an estimate of Θ. The open loop error system is then,

$$M\dot{r} = Y\widetilde{\Theta} + Y\widehat{\Theta} - u$$

Controller

Effort Design

Consider the candidate Lyapunov function,

$$V \coloneqq \frac{1}{2} r^T M r + \frac{1}{2} \tilde{\Theta}^T \Gamma^{-1} \tilde{\Theta}$$

for some positive definite, 5x5 gain matrix Γ which may be a function of time. V is positive definite because both Γ^{-1} and M are positive definite. The time derivative is then,

$$\dot{V} = \frac{1}{2} \dot{r}^T M r + \frac{1}{2} r^T M \dot{r} + \frac{1}{2} \dot{\tilde{\Theta}}^T \Gamma^{-1} \tilde{\Theta} + \frac{1}{2} \tilde{\Theta}^T \Gamma^{-1} \dot{\tilde{\Theta}} + \frac{1}{2} \tilde{\Theta}^T \frac{d}{dt} (\Gamma^{-1}) \tilde{\Theta}$$

We can make use of the properties of the transpose and the matrix identity,

$$\frac{d}{dt} (\Gamma^{-1}) \equiv -\Gamma^{-1} \dot{\Gamma} \Gamma^{-1}$$

to further simplify the derivative to,

$$\dot{V} = r^T M \dot{r} + \tilde{\Theta}^T \Gamma^{-1} \dot{\tilde{\Theta}} - \frac{1}{2} \tilde{\Theta}^T \Gamma^{-1} \dot{\Gamma} \Gamma^{-1} \tilde{\Theta}$$

and finally substitute the dynamics in to find,

$$\dot{V} = r^T \left(Y \tilde{\Theta} + Y \hat{\Theta} - u \right) - \tilde{\Theta}^T \Gamma^{-1} \dot{\hat{\Theta}} - \frac{1}{2} \tilde{\Theta}^T \Gamma^{-1} \dot{\Gamma} \Gamma^{-1} \tilde{\Theta}$$

For Lyapunov stability of the error r, we seek to choose u so that $\dot{V}$ is at least negative semidefinite. Recognizing the need for feedback on r and feedforward on YΘ, we design u as,

$$u \coloneqq kr + Y \hat{\Theta}$$

Then the Lyapunov derivative becomes,

$$\dot{V} = r^T \left(Y \tilde{\Theta} + Y \hat{\Theta} - kr - Y \hat{\Theta} \right) - \tilde{\Theta}^T \Gamma^{-1} \dot{\hat{\Theta}} - \frac{1}{2} \tilde{\Theta}^T \Gamma^{-1} \dot{\Gamma} \Gamma^{-1} \tilde{\Theta}$$

$$= r^T \left(Y \tilde{\Theta} - kr \right) - \tilde{\Theta}^T \Gamma^{-1} \dot{\hat{\Theta}} - \frac{1}{2} \tilde{\Theta}^T \Gamma^{-1} \dot{\Gamma} \Gamma^{-1} \tilde{\Theta}$$

<u>Adaptation Design</u>

The typical method is to let Γ be a tunable but constant gain matrix and to let the adaptation law minimize squared tracking error by gradient descent, i.e. the tracking gradient method,

$$\dot{\widehat{\Theta}} \coloneqq \Gamma Y^T r, \quad \text{for } constant\ \Gamma$$

so that,

$$\dot{V} = r^T\big(Y\widetilde{\Theta} - kr\big) - \widetilde{\Theta}^T\Gamma^{-1}\Gamma Y^T r$$
$$= -r^T kr$$

By Barbalat's Lemma, this result yields global asymptotic tracking, but it does not yield any reliable system identification without persistence of excitation. At best, the adaptation serves to eliminate steady state error without introducing overshoot, which we will consider to be benchmark performance. This method was examined more closely in the predecessor to this document. Here, we will examine the stability of four more advanced adaptation laws.

<u>The Composite Gradient Method</u>

Similar to the tracking gradient method, Γ is held constant, but now the gradient descent cost function is extended to also contain the square of a "prediction" error, ε, representing the difference between the control input performed and the expected system response,

$$\epsilon \coloneqq u - Y_u\widehat{\Theta}$$
$$= Y_u\Theta - Y_u\widehat{\Theta}$$

Thus for some diagonal gain matrix k_u, and noting that k_u=0 reduces this law to exactly the tracking gradient method, we choose,

$$\dot{\widehat{\Theta}} \coloneqq \Gamma Y^T r + \Gamma \mathrm{k_u} Y_u^T \epsilon, \quad \text{for } constant\ \Gamma$$

For this system, however, even though Y_u contains no unknown parameters, it contains the assumed unmeasurable state derivative. Thus, a filtered prediction error is used instead, which we will see has the same usefulness as the regular prediction error. Let,

$$\epsilon_f \coloneqq u_f - Y_{u_f}\widehat{\Theta}$$

where, for any quantity h, some tunable gain β, and the natural base e,

$$h_f \coloneqq \beta e^{-\beta t} * h$$

Obviously, u_f can be computed by integrating the following differential equation,

$$\dot{u}_f = \beta(u - u_f)$$

Additionally, now the computation of Y_{uf} can be manipulated to only require measurable states (this is the purpose of the filtering). First we define,

$$\dot{h}_1 \coloneqq \frac{d}{dt}(M\dot{q}) = M\ddot{q} + \dot{M}\dot{q}$$

$$\rightarrow \quad h_1 = M\dot{q}$$

$$h_2 \coloneqq -\dot{M}\dot{q} + V + G$$

Thus we can express Y_u as a sum of terms involving the state derivative and those that don't,

$$Y_u\Theta = M\ddot{q} + V + G$$

$$= \dot{h}_1 + h_2$$

Next we note that, for any functions f and h, the identity,

$$f * \dot{h} = \dot{f} * h + f(0)h - h(0)f$$

holds. Thus if

$$f \coloneqq \beta e^{-\beta t}$$

$$\rightarrow \quad \dot{f} = -\beta^2 e^{-\beta t}$$

then,

$$Y_{u_f}\Theta = \left(\dot{f} * h_1 + f(0)h_1 - h_1(0)f\right) + f * h_2$$

$$= -\beta^2 e^{-\beta t} * M\dot{q} + \beta M\dot{q} - M(0)\dot{q}(0)\beta e^{-\beta t} + \beta e^{-\beta t} * \left(-\dot{M}\dot{q} + V + G\right)$$

$$= \beta\left(Y_{u_{f_1}} + Y_{u_{f_2}}\right)\Theta$$

where,

$$\dot{Y}_{u_{f_1}}\Theta = V + G - \left(\beta M + \dot{M}\right)\dot{q} - \beta Y_{u_{f_1}}\Theta$$

$$\rightarrow \quad \dot{Y}_{u_{f_1}} = \begin{bmatrix} \cos(\theta_1) & -\beta\dot{\theta}_1 & \cos(\theta_1+\theta_2) & -\beta(2\dot{\theta}_1+\dot{\theta}_2)\cos(\theta_2) & -\beta(\dot{\theta}_1+\dot{\theta}_2) \\ 0 & 0 & \cos(\theta_1+\theta_2) & \dot{\theta}_1\left((\dot{\theta}_1+\dot{\theta}_2)\sin(\theta_2)-\beta\cos(\theta_2)\right) & -\beta(\dot{\theta}_1+\dot{\theta}_2) \end{bmatrix} - \beta Y_{u_{f_1}}$$

and

$$Y_{u_{f_2}}\Theta = M\dot{q} - M(0)\dot{q}(0)e^{-\beta t}$$

$$\rightarrow \quad Y_{u_{f_2}} = \begin{bmatrix} 0 & \dot{\theta}_1 & 0 & (2\dot{\theta}_1+\dot{\theta}_2)\cos(\theta_2) & \dot{\theta}_1+\dot{\theta}_2 \\ 0 & 0 & 0 & \dot{\theta}_1\cos(\theta_2) & \dot{\theta}_1+\dot{\theta}_2 \end{bmatrix} - e^{-\beta t}\begin{bmatrix} 0 & \dot{\theta}_1 & 0 & (2\dot{\theta}_1+\dot{\theta}_2)\cos(\theta_2) & \dot{\theta}_1+\dot{\theta}_2 \\ 0 & 0 & 0 & \dot{\theta}_1\cos(\theta_2) & \dot{\theta}_1+\dot{\theta}_2 \end{bmatrix}(0)$$

With Y_{uf} and u_f now computable, we can define the adaptation law as,

$$\dot{\hat{\Theta}} \coloneqq \Gamma Y^T r + \Gamma \mathrm{k_u} Y_{u_f}^T \epsilon_f, \quad \text{for } constant\ \Gamma$$

so the Lyapunov derivative becomes,

$$\dot{V} = r^T\left(Y\widetilde{\Theta} - kr\right) - \widetilde{\Theta}^T\Gamma^{-1}\left(\Gamma Y^T r + \Gamma \mathrm{k_u} Y_{u_f}^T \epsilon_f\right)$$

$$= r^T Y\widetilde{\Theta} - r^T kr - \widetilde{\Theta}^T Y^T r - \widetilde{\Theta}^T k_u Y_{u_f}^T \epsilon_f$$

$$= -r^T kr - \widetilde{\Theta}^T k_u Y_{u_f}^T \epsilon_f$$

$$= -r^T kr - \widetilde{\Theta}^T k_u Y_{u_f}^T \left(\mathrm{Y_{uf}}\Theta - Y_{u_f}\widehat{\Theta}\right)$$

$$= -r^T kr - \widetilde{\Theta}^T k_u Y_{u_f}^T \mathrm{Y_{uf}}\widetilde{\Theta}$$

Integrating this result reveals,

$$r \in L_\infty \cap L_2$$

$$Y_{u_f}\widetilde{\Theta} \in L_\infty \cap L_2$$

And thus by Barbalat's lemma,

$$\lim_{t\to\infty} r = 0$$

$$\lim_{t\to\infty} Y_{u_f}\widetilde{\Theta} = 0$$

This Lyapunov result is no more exciting than the previously described benchmark result obtained with just tracking gradient adaptation, because the second limit says nothing definitive about the fate of $\widetilde{\Theta}$. However, we can now derive a condition under which this result yields convergence of $\widetilde{\Theta}$ itself to zero, but that is less restrictive than persistence of excitation. Examining the estimation error derivative,

$$\dot{\tilde{\Theta}} = -\dot{\hat{\Theta}}$$

$$= -\Gamma Y^T r - \Gamma \mathrm{k_u} Y_{u_f}^T Y_{u_f} \tilde{\Theta}$$

We note that from the Lyapunov analysis, the first term goes to zero as t goes to infinity. Thus,

$$\lim_{t\to\infty} \dot{\tilde{\Theta}} = -\Gamma \mathrm{k_u} Y_{u_f}^T Y_{u_f} \tilde{\Theta}$$

and so at large time t, $\tilde{\Theta}$ evolves according to the following differential equation,

$$\dot{\tilde{\Theta}} = -\Gamma \mathrm{k_u} Y_{u_f}^T Y_{u_f} \tilde{\Theta}$$

which has a solution,

$$\tilde{\Theta}(t) = \tilde{\Theta}(0) e^{-\int_0^t \Gamma \mathrm{k_u} Y_{u_f}^T Y_{u_f} dt}$$

Thus if,

$$\lim_{t\to\infty} \int_0^t \Gamma \mathrm{k_u} Y_{u_f}^T Y_{u_f} dt = \infty$$

then,

$$\lim_{t\to\infty} \tilde{\Theta} = 0$$

This is known as the infinite integral condition, and it is basically a metric on the sufficiency of the learning so far. That is, exploration of the state space helps the estimation error go to zero if it adds to the magnitude of the integral.

The Composite Least-Squares Method

This method is almost identical to the composite gradient method, but now the gain Γ is made to be a function of the state (and thus varies in time). Using typical optimization techniques, Γ is chosen so that it minimizes the square of the prediction error if the system is linear. The development of this optimal Γ will not be presented here, but the result will. That is, we will use the optimal linear Γ on this system *heuristically*. Let Γ evolve from,

$$\dot{\Gamma} \coloneqq -\Gamma \mathrm{k_u} {Y_{u_f}}^T Y_{u_f} \Gamma$$

and just like with the composite gradient method, let,

$$\dot{\hat{\Theta}} := \Gamma Y^T r + \Gamma \mathrm{k_u} Y_{u_f}^T \epsilon_f$$

Thus, the Lyapunov derivative becomes,

$$\dot{V} = r^T\left(Y\widetilde{\Theta} - kr\right) - \widetilde{\Theta}^T\Gamma^{-1}\left(\Gamma Y^T r + \Gamma \mathrm{k_u} Y_{u_f}^T \epsilon_f\right) + \frac{1}{2}\widetilde{\Theta}^T\Gamma^{-1}\Gamma \mathrm{k_u} Y_{u_f}^T Y_{u_f}\Gamma\Gamma^{-1}\widetilde{\Theta}$$

$$= r^T\left(Y\widetilde{\Theta} - kr\right) - \widetilde{\Theta}^T Y^T r - \widetilde{\Theta}^T \mathrm{k_u} Y_{u_f}^T \epsilon_f + \frac{1}{2}\widetilde{\Theta}^T k_u Y_{u_f}^T Y_{u_f}\widetilde{\Theta}$$

$$= -r^T kr - \widetilde{\Theta}^T \mathrm{k_u} Y_{u_f}^T \epsilon_f + \frac{1}{2}\widetilde{\Theta}^T k_u Y_{u_f}^T Y_{u_f}\widetilde{\Theta}$$

$$= -r^T kr - \widetilde{\Theta}^T \mathrm{k_u} Y_{u_f}^T \mathrm{Y_{u_f}}\widetilde{\Theta} + \frac{1}{2}\widetilde{\Theta}^T k_u Y_{u_f}^T Y_{u_f}\widetilde{\Theta}$$

$$= -r^T kr - \frac{1}{2}\widetilde{\Theta}^T k_u Y_{u_f}^T Y_{u_f}\widetilde{\Theta}$$

Despite the ½ on the second term, this is the identical stability result as was obtained for the composite gradient method, and thus the same infinite integral condition dictates the convergence of the adaptive estimates. Again, the only difference is that the once tunable parameter Γ now varies with time for least-squares optimality if the system is linear.

The Concurrent Gradient Method

This method is also similar to the composite gradient method, but now instead of using the instantaneous Y_u and u_f values for the prediction error, N pairs of these values from previous times are combined. The list of these previous data points, referred to as the "history stack," is updated selectively by a metric that will be discovered during the stability analysis. Let,

$$\dot{\hat{\Theta}} := \Gamma Y^T r + \Gamma \mathrm{k_u}\sum_{i=1}^{N}\left(Y_{u_{f_i}}^T\left(u_{f_i} - Y_{u_{f_i}}\hat{\Theta}\right)\right), \quad \text{for } constant\ \Gamma$$

$$= \Gamma Y^T r + \Gamma \mathrm{k_u}\sum_{i=1}^{N}\left(Y_{u_{f_i}}^T Y_{u_{f_i}}\right)\widetilde{\Theta}$$

The Lyapunov derivative is then,

$$\dot{V} = r^T\left(Y\widetilde{\Theta} - kr\right) - \widetilde{\Theta}^T\Gamma^{-1}\left(\Gamma Y^T r + \Gamma \mathrm{k_u}\sum_{i=1}^{N}\left(Y_{u_{f_i}}^T Y_{u_{f_i}}\right)\widetilde{\Theta}\right)$$

$$= -r^T kr - \widetilde{\Theta}^T \mathrm{k_u} \sum_{i=1}^{N} \left(Y_{u_{f_i}}^{T} Y_{u_{f_i}} \right) \widetilde{\Theta}$$

In the previous methods' results, the matrix $k_u Y_{u_f}^T Y_{u_f}$ was necessarily positive semidefinite (with a minimum eigenvalue of zero) because $Y_{u_f}^T Y_{u_f}$ (the "square" of a non-square matrix) is always rank deficient. Thus the entire Lyapunov result was negative semidefinite and we had to use Barbalat's lemma and other development to arrive at the infinite integral condition. However, now we have a matrix sum in its place. If the history stack of $Y_{u_{f_i}}^T Y_{u_{f_i}}$ remains only positive semidefinite, then the stability result reduces to that of the previous methods-asymptotic tracking. However, if data can be collected such that the summation of the history stack becomes positive definite, then we can upper bound the second term by the minimum eigenvalue of the history stack sum. That is,

$$if, \qquad \min\left(eig\left(\sum_{i=1}^{N} \left(Y_{u_{f_i}}^{T} Y_{u_{f_i}} \right) \right) \right) \coloneqq \underline{\lambda} > 0$$

$$then, \qquad \dot{V} \leq -r^T kr - \mathrm{k_u} \underline{\lambda} \left\| \widetilde{\Theta} \right\|^2$$

$$\leq -\min\left(k, k_u \underline{\lambda}\right) V$$

Thus, by simple integration we see that V decays exponentially at a rate proportional to $\min\left(k, k_u \underline{\lambda}\right)$ and thus both r and $\widetilde{\Theta}$ go to zero exponentially. This result technically existed for the previous methods, except $\underline{\lambda}$ was always zero. Evidently, to decide on adding a new data pair to the history stack, we examine if it raises the minimum eigenvalue of $\sum_{i=1}^{N} \left(Y_{u_{f_i}}^{T} Y_{u_{f_i}} \right)$. As long as the eigenvalue is anything above zero, the result will begin to converge exponentially, however slowly unless the data is rich enough to bring up the value of $\mathrm{k_u} \underline{\lambda}$. Of course, we could artificially boost $\mathrm{k_u}$, but doing so will severely bias the update law towards the prediction error and thus forfeit robustness. We cannot forget that it is the $\Gamma Y^T r$ term responsible for tracking.

The Integral Concurrent Gradient Method

In the previous concurrent gradient formulation, we made use of the filtered input and regressor matrix. Of course, the filtering method itself was not relevant to the analysis. Its purpose was solely to rewrite the progression of Y_u in terms of measurable states only (and so the lowpass filter $\beta e^{-\beta t}$ was convenient). Another option is to simply use integrals of Y_u and u. Define,

$$\overbrace{u} \coloneqq \int_0^t u \, d\sigma$$

$$\overbrace{Y}_u \coloneqq \int_0^t Y_u \, d\sigma = Y_{u_{f_1}}(\beta = 0) + Y_{u_{f_2}}(\beta = 0)$$

where Y_{uf1} and Y_{uf2} are defined just as they were for the lowpass filtering case, but with β=0. Rewritten here for convenience,

$$\dot{Y}_{u_{f_1}}\Theta = V + G - \dot{M}\dot{q}$$

$$= \begin{bmatrix} \cos(\theta_1) & 0 & \cos(\theta_1+\theta_2) & 0 & 0 \\ 0 & 0 & \cos(\theta_1+\theta_2) & \dot{\theta}_1(\dot{\theta}_1+\dot{\theta}_2)\sin(\theta_2) & 0 \end{bmatrix}\Theta$$

and

$$Y_{u_{f_2}}\Theta = M\dot{q} - M(0)\dot{q}(0)$$

$$= \left(\begin{bmatrix} 0 & \dot{\theta}_1 & 0 & (2\dot{\theta}_1+\dot{\theta}_2)\cos(\theta_2) & \dot{\theta}_1+\dot{\theta}_2 \\ 0 & 0 & 0 & \dot{\theta}_1\cos(\theta_2) & \dot{\theta}_1+\dot{\theta}_2 \end{bmatrix} - \begin{bmatrix} 0 & \dot{\theta}_1 & 0 & (2\dot{\theta}_1+\dot{\theta}_2)\cos(\theta_2) & \dot{\theta}_1+\dot{\theta}_2 \\ 0 & 0 & 0 & \dot{\theta}_1\cos(\theta_2) & \dot{\theta}_1+\dot{\theta}_2 \end{bmatrix}(0)\right)\Theta$$

Arriving at this equation for $\int_0^t Y_u \, d\sigma$ is a simple application of integration by parts, but it can also be deduced from its similarity to the previous convolutional filtering process. Now we can define the adaptation law just as before but using the integration-computed data pairs. That is,

$$\dot{\hat{\Theta}} \coloneqq \Gamma Y^T r + \Gamma \mathrm{k_u} \sum_{i=1}^{N}\left(\overbrace{Y}_{u_i}^T\left(\overbrace{u}_i - \overbrace{Y}_{u_i}\hat{\Theta}\right)\right), \quad \text{for } constant\ \Gamma$$

The stability analysis follows identically to that of the previous section.

Simulation

The four control schemes were implemented on a simulation of the given system. To verify robustness, a small noise disturbance was added to the simulation model, as well as a small amount of damping in the joints, and actuator saturation (max 250 N*m at the first joint and 30 N*m at the second joint). The simulation used the following true parameters:

- m_1 = 5 kg
- m_2 = 3 kg
- a_1 = 1 m
- a_2 = 0.5 m
- g = 9.81 m/s^2

Gains were varied throughout experimentation. The best results achieved are summarized here with their respective gains. In each plot, the dashed lines are the desired trajectories / actual model parameters. A random smooth trajectory was invented for testing the controller, but this same trajectory was used across each controller type.

The errors are not displayed because they are better visualized with how the system tracks the desired trajectories. When you can see when the trajectory was harsh or sharp and when it was not, it is a lot clearer to see how the controller was truly behaving. Small blips in the error convergence do not give the insight as to why they're there. Thus tracking was displayed for this report, not the errors alone.

The adaptation was always initialized to all zeros, and in every case,

- k = diag(100, 100)
- α = diag(1, 1)

The code used to implement the controller and run this simulation was made in Python and can be found at https://github.com/jnez71/planArm/tree/master/two_link

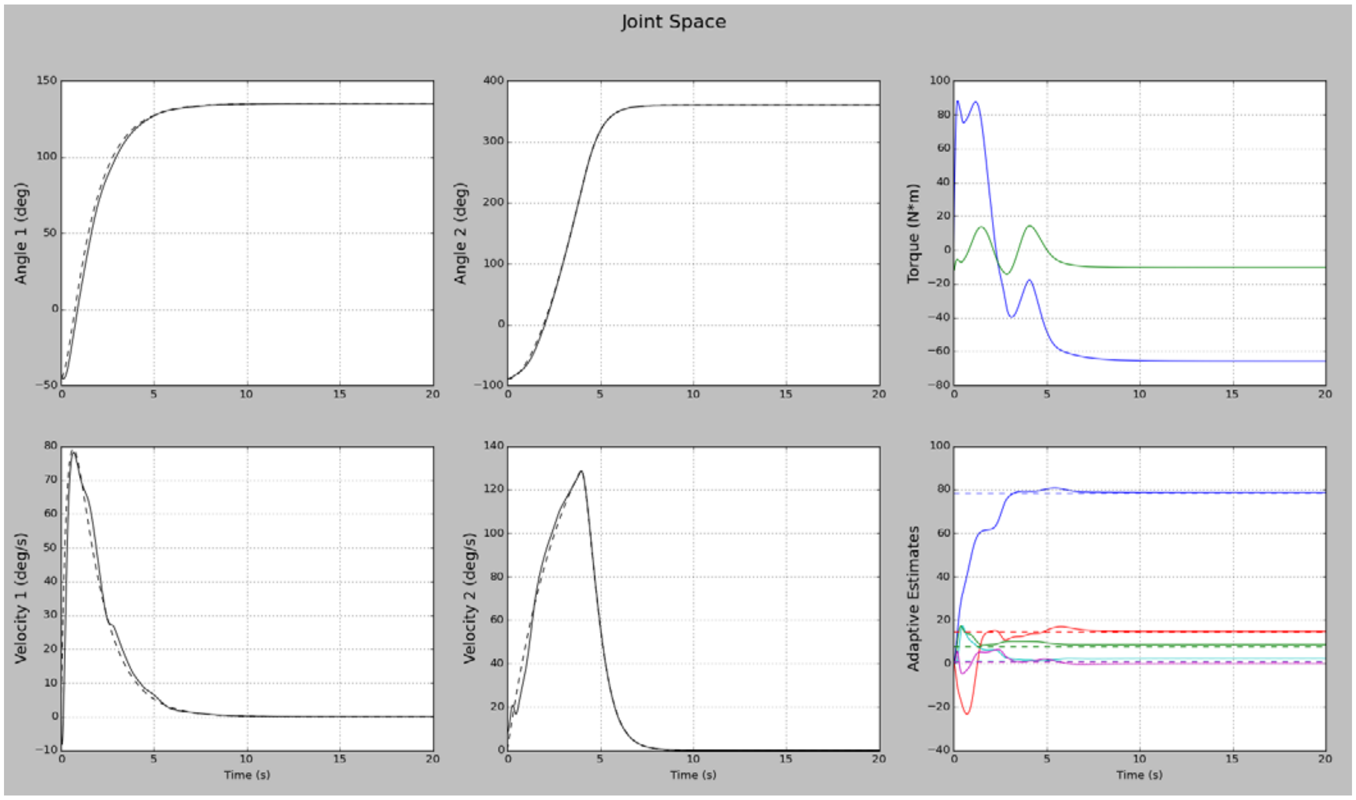
Simply download the files in that folder, and make sure to have Python, numpy, and matplotlib installed. Then run sim_2link.py to view the simulation (or edit the source as you please, it is very well commented).

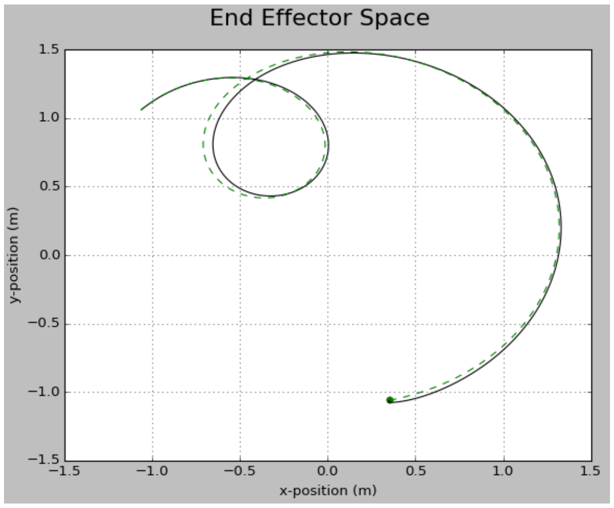
…

The Composite Gradient Method

- Γ = diag(200, 200, 200, 200, 200)
- k_u = diag(2, 2, 2, 2, 2)
- β = 0.2

Final adaptation error ($\widetilde{\Theta}$): [-0.01   -0.64   0   -0.84   0.79]

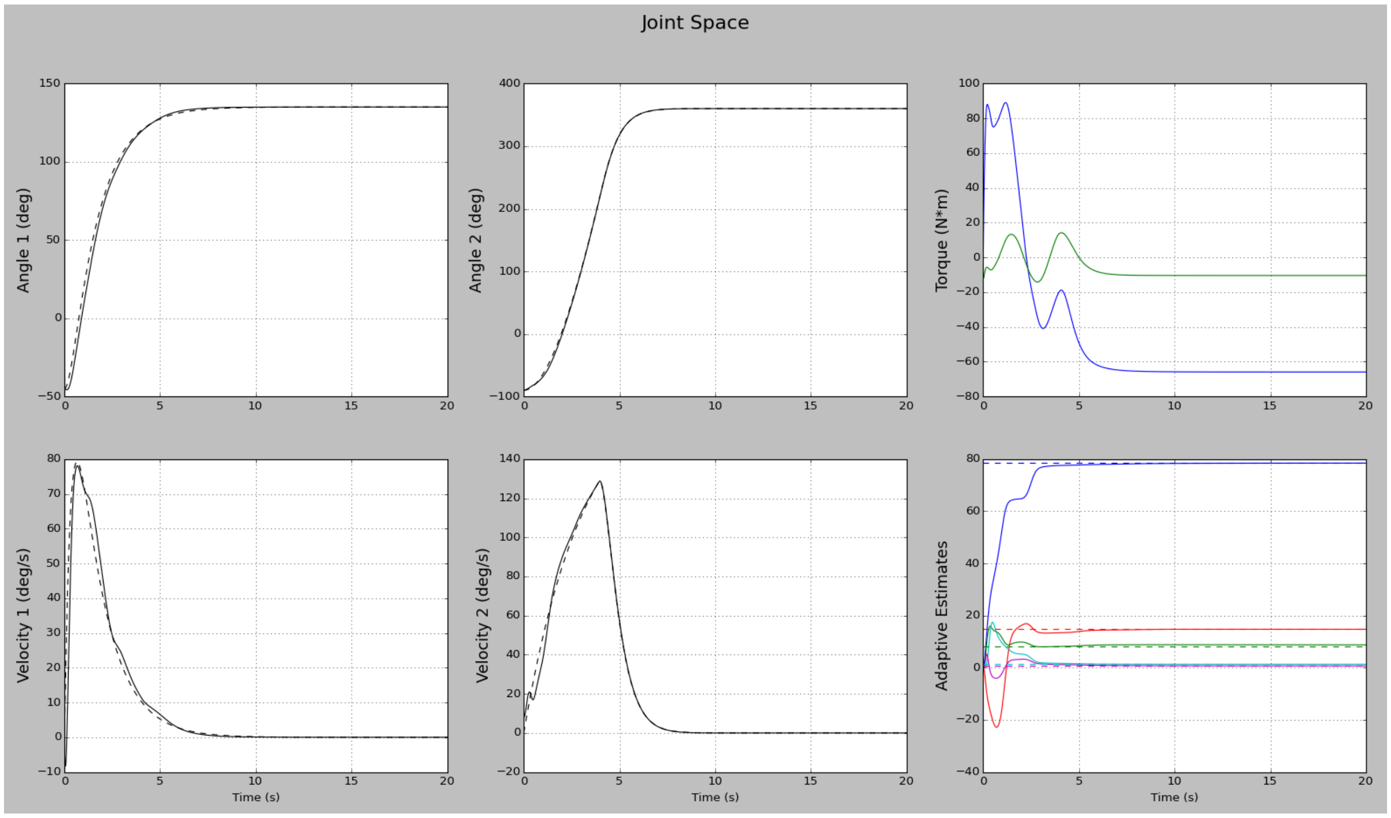


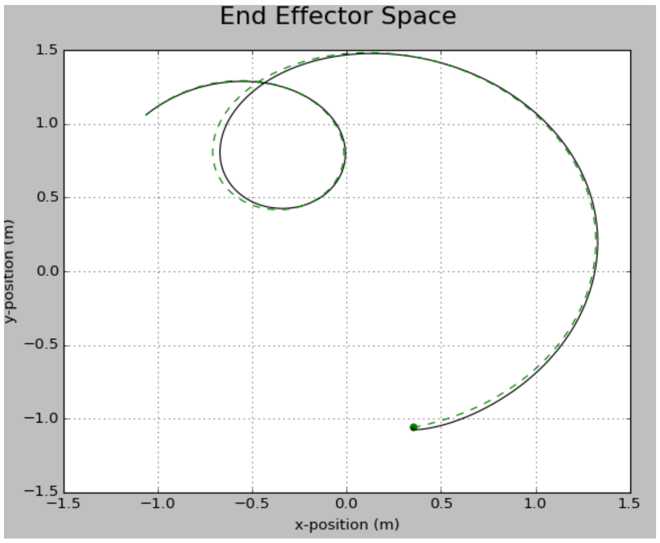


The Composite Least-Squares Method

- Γ found by least squares ODE
- k_u = diag(2, 2, 2, 2, 2)
- β = 0.2

Final adaptation error: [0.02   -0.77   -0.01   0.18   0.05]

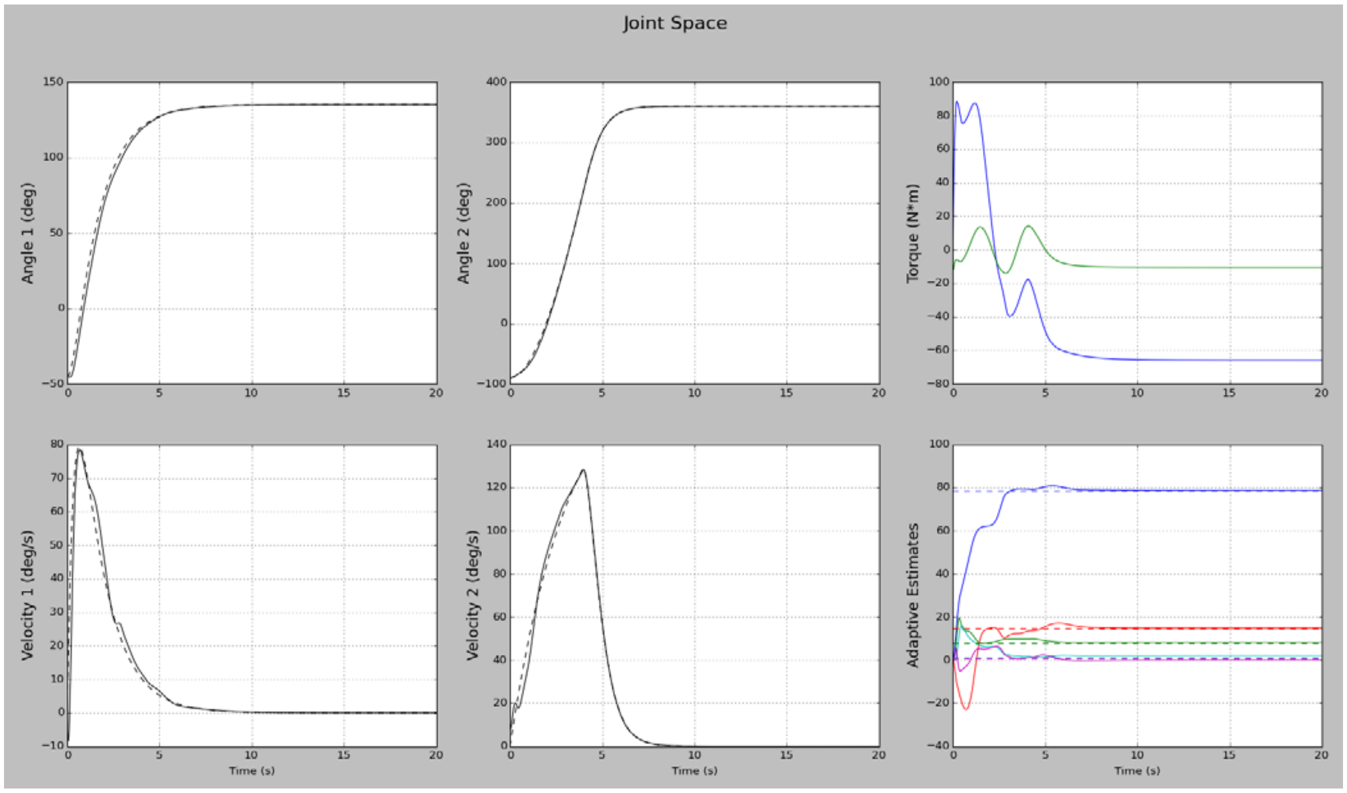


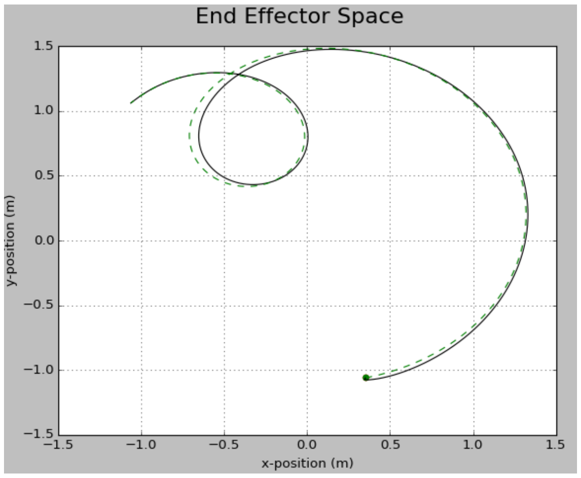


The Concurrent Gradient Method

- Γ = diag(200, 200, 200, 200, 200)
- k_u = diag(2, 2, 2, 2, 2)
- β = 0.2
- N = 3

Final adaptation error: [-0.01 -0.15 0 -0.62 0.64]

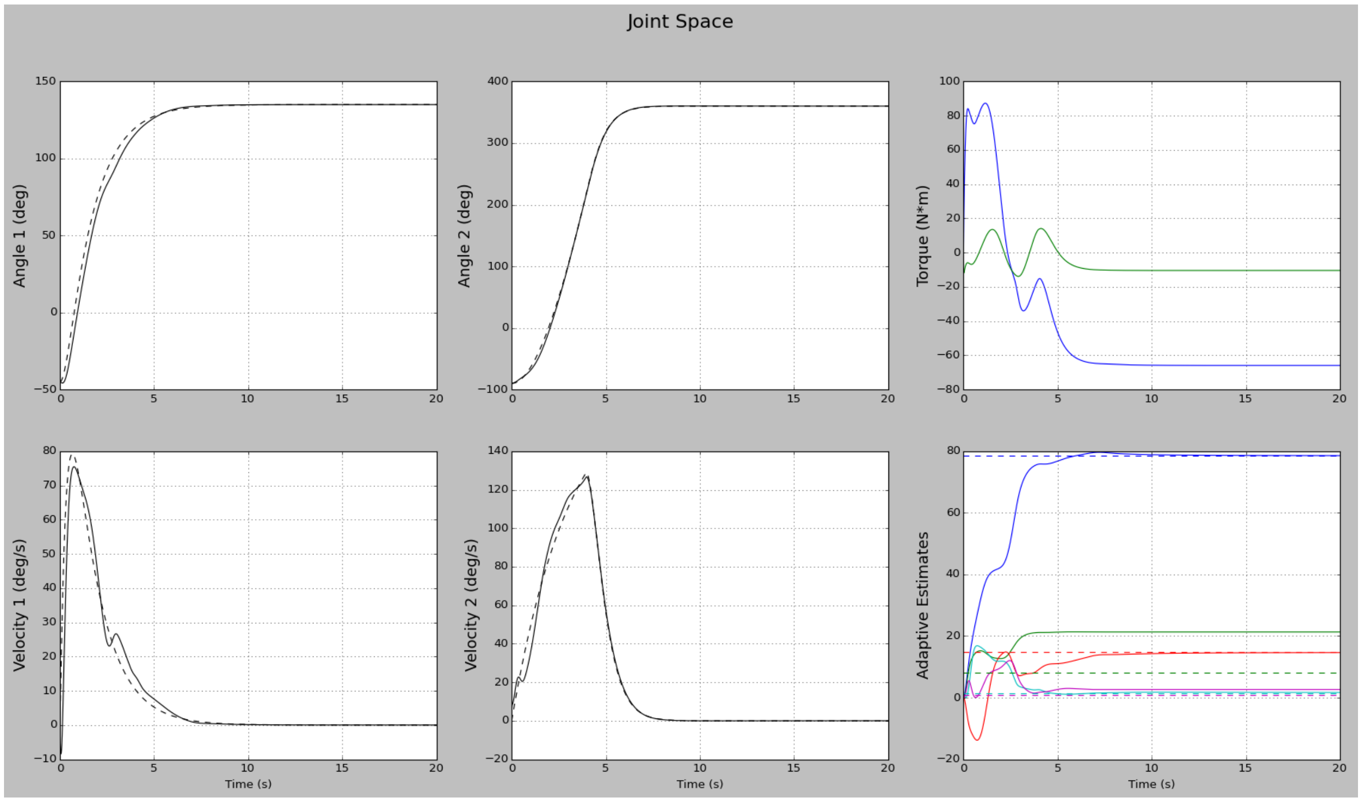


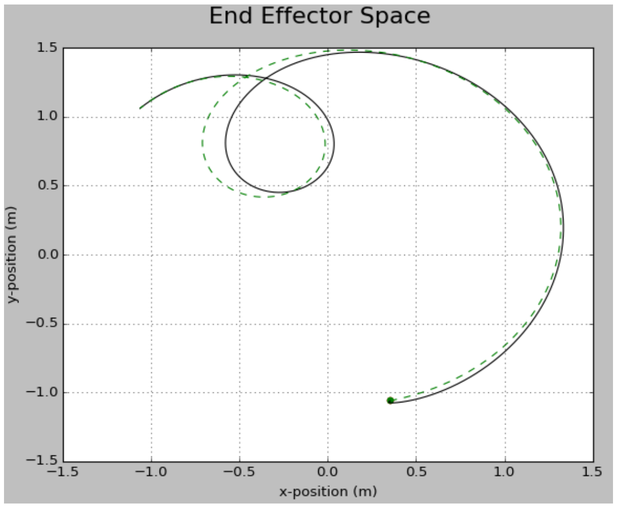


The Integral Concurrent Gradient Method

- Γ = diag(200, 200, 200, 200, 200)
- k_u = diag(2, 2, 2, 2, 2)
- N = 3

Final adaptation error: [-0.07 -13.31 0.08 -0.18 -1.89]





Discussion / Findings

Tuning Gains

- Tuning adaptation gains is somewhat similar to tuning control gains. The primary tradeoff is between rapid convergence of error to zero, and oversensitivity.
- If Γ is too small, the tracking performance of any of the control schemes approaches that of a simple PD controller- poor. Γ too large actually caused instability.
- A Γ roughly twice the size of k worked nicely for each controller type. It caused it to learn quickly enough compared to the tracking speed of the PD term.
- The least squares selection for Γ was the most effective. The decay of Γ as the goal was approached led to smooth convergence both in tracking and adaptation. It appeared to "filter" an otherwise jumpy adaptation process.
- When using the convolutional filter, a $\beta > 1$ caused immediate instability. $\beta > 0.5$ caused erratic but stable performance. β too caused the composite control scheme to approach the performance of the basic tracking-error-only control scheme. β around 0.2 is ideal.
- Increasing k_u sped up adaptation convergence, but worsened tracking. In the end, great tracking could still be achieved (with some added adaptation performance) with $k_u = 2$.
- Increasing the stack size added very little benefit after about N = 5. Past N = 10 and the system would go unstable unless I lowered Γ.

Comparison of Methods - Tracking

- The tracking performance was roughly the same in all cases (pretty darn good).
- Integral concurrent was "the worst" tracking, but not by much.
- Tracking was also this good for the just-tracking-error adaptation control scheme (last project). That is, tracking was still good even as k_u went to 0.

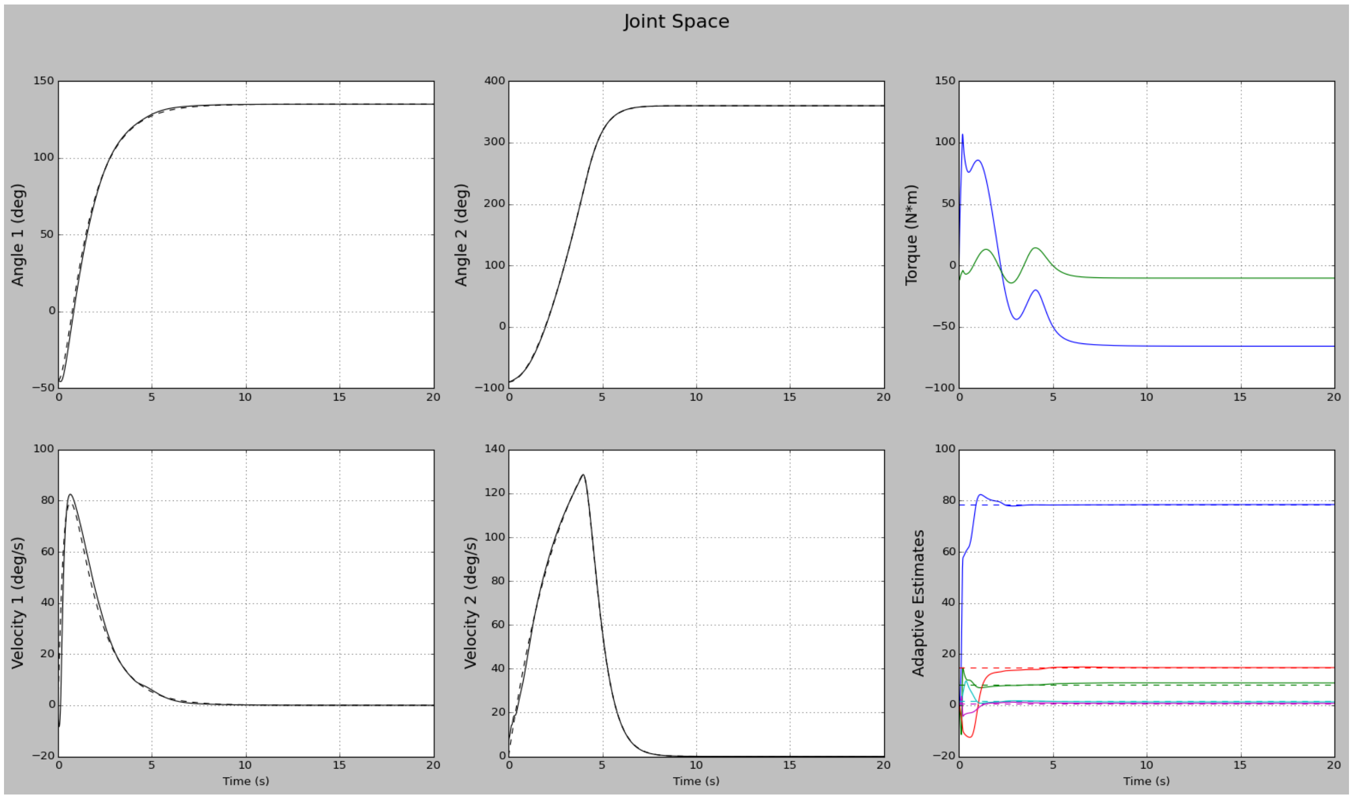
Comparison of Methods - Adaptation

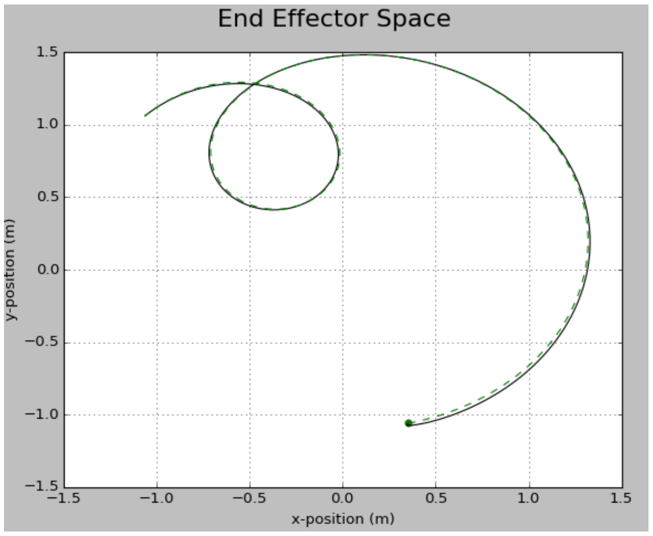
- Concurrent behaved better than composite when using convolutional filtering, however when using integral filtering the performance became poorer.
- Least squares composite provided substantial performance over fixed Γ composite. In general, applying least squares made the otherwise jumpy adaptation smoother.
- Every one of these methods provided substantially better system identification than the just-tracking-error control scheme (last project).
- In the end, the best method I found was actually ***integral concurrent with least squares adaptation***. Here are the results:

The Least Squares Integral Concurrent Gradient Method

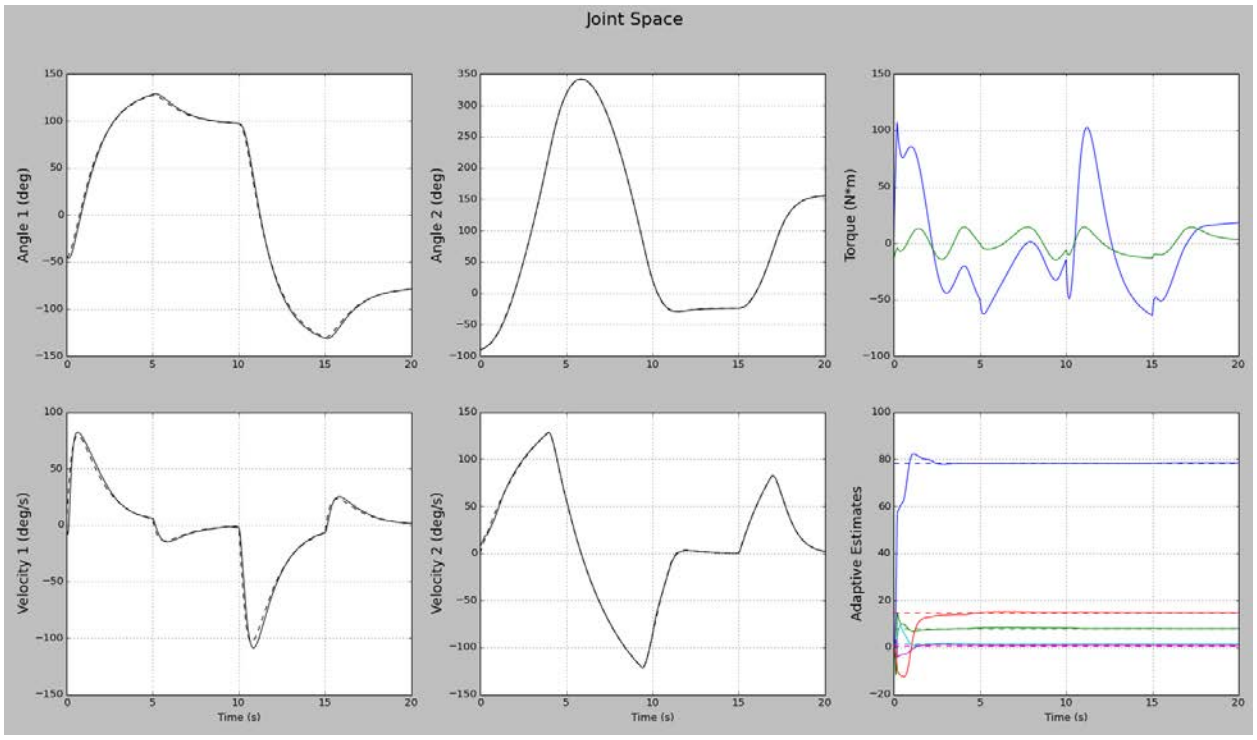
- Γ = diag(200, 200, 200, 200, 200)
- k_u = diag(2, 2, 2, 2, 2)
- N = 5

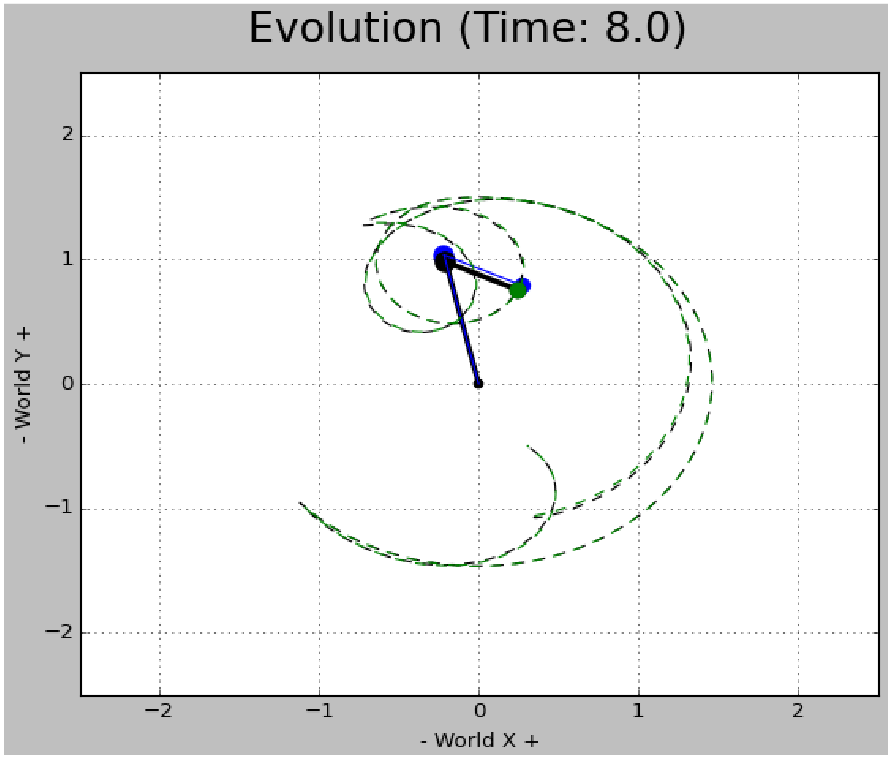
Final adaptation error: [0.02 -0.73 -0.01 0.18 0]





Same thing, but tracking an even wackier trajectory.





(blue is what it “thinks it looks like”… convergence!)