Introduction

The goal of this document is to obtain a linearized state space model that is parameterized in terms of the path arc length s, instead of time, t, as normal. If we begin with a normal state space model:

$$\begin{aligned} \dot{x}_1 &= f_1(\vec{x}, \vec{u}) \\ \dot{x}_2 &= f_2(\vec{x}, \vec{u}) \\ &\vdots \\ \dot{x}_n &= f_n(\vec{x}, \vec{u}) \end{aligned} \tag{1}$$

and then apply the chain rule:

$$\begin{aligned} \frac{d\dot{x}_1}{ds}\frac{ds}{dt} &= f_1(\vec{x}, \vec{u}) \\ &\vdots \\ \frac{d\dot{x}_n}{ds}\frac{ds}{dt} &= f_n(\vec{x}, \vec{u}) \end{aligned} \tag{2}$$

now, if we move the $\frac{ds}{dt}$ term to the other side, we get:

$$\begin{aligned} \frac{d\dot{x}_1}{ds} &= f_1(\vec{x}, \vec{u}) \left(\frac{ds}{dt}\right)^{-1} \\ &\vdots \\ \frac{d\dot{x}_n}{ds} &= f_n(\vec{x}, \vec{u}) \left(\frac{ds}{dt}\right)^{-1} \end{aligned} \tag{3}$$

Writing this in vectorized notation:

$$\frac{d\dot{\vec{x}}}{ds} = \vec{F}(\vec{x}, \vec{u}) \left(\frac{ds}{dt}\right)^{-1} \tag{4}$$

In MATLAB, the function `linmod` will linearize a model for us. What we can do is, define a Simulink block that calculates $\frac{ds}{dt}$ and implement a Simulink model that represents 4. Then, when we run `linmod` on this Simulink model, it will return A, B, and C matrices that are parameterized with respect to the spatial coordinate s. However, there are a few subtle points about this that I think I should write down:

- `linmod` performs a lineraization around a constant operating point, *not* a trajectory of operating points. So it's probably going to be necessary to wrap `linmod` into a for loop that steps through all the state vectors in order of the path arc length.

- In implementing $\frac{ds}{dt}$ in simulink, I don't think that it's sufficient to first calculate s and then take a (non-causal) numerical derivative. I don't think that a numerical derivative will behave as we want it within `linmod`. I believe that `linmod` doesn't "march" or "change" time at all when linearizing, so the value of this approximate derivative will not change at all from the initial condition within linmod. Instead, I think we need to analytically calculate (or analytically approximate) $\frac{ds}{dt}$, which is what the next section is about.

Approximating the Path Speed

So I think that in order to do this rigorously, I need to first define some coordinate systems and variables. There are 3 coordinate frames, the global reference frame $\overline{U}$, the body fixed reference frame, $\overline{B}$, and the Serret-Frenet frame attached to the path, $\overline{S}$. Each of these coordinate frames is comprised of a point, which defines the origin of the coordinate system, along with three unit vectors:

$$\overline{U} = \{U, \vec{i}_{\overline{U}}, \vec{j}_{\overline{U}}, \vec{k}_{\overline{U}}\} \tag{5}$$
$$\overline{B} = \{B, \vec{i}_{\overline{B}}, \vec{j}_{\overline{B}}, \vec{k}_{\overline{B}}\} \tag{6}$$
$$\overline{S} = \{S, \vec{i}_{\overline{S}}, \vec{j}_{\overline{S}}, \vec{k}_{\overline{S}}\}. \tag{7}$$

The Serret-Frenet frame is defined so that the point S, which is the origin of the coordinate system, lies on the path. Thus, the simple version shape of the path relative to the point U is described by:

$$\vec{r}_{S/U}(\phi) = W\cos(\theta)\vec{i}_{\overline{U}} + H\sin(2\theta)\vec{j}_{\overline{U}}. \tag{8}$$

However, in order to get this to start and end at the correct points (and travel in the right direction on the figure 8), we can make the substitution that $\theta \to \left(\frac{3}{2} + 2\phi\right)\pi$. In this case the nondimensional parameter $\phi \in [01]$ results in the type of motion around the path that we want. Specifically $\phi = 0$ starts at the center of the figure 8 and $\phi = 1$ ends up back at the center.

The unit vector $\vec{i}_{\overline{S}}$ is defined to be tangent to the path, pointing in the direction of increasing arc length, s (note that this makes it a function of ϕ). In general, the path arc length s is given by

$$s = \int_{s_i}^{s_f} \|\frac{d}{d\phi}\vec{r}_{S/U}(\phi)\| d\phi. \tag{9}$$

In our case, this integral is *extremely* difficult to solve, and it looks like it actually wasn't solved until about 2006 and it requires a bunch of really high level math. So I don't think that finding an expression for this, and then differentiating with respect to time is going to be possible. Instead, what I'd like to do is the following:

$$\frac{d}{dt}s = {}^{\overline{U}}\vec{v}_{B/U} \cdot \vec{i}_{\overline{s}}, \tag{10}$$

basically, take the component of the velocity in the direction of the tangent vector of the Serret-Frenet frame. So in order to do this, we need to find all the things on the right hand side of 10. The first term is pretty easy, specifically:

$${}^{\overline{U}}\vec{v}_{B/U} = v\cos(\psi)\vec{i}_{\overline{U}} + v\sin(\psi)\vec{i}_{\overline{U}} \tag{11}$$

where v is the speed of our system and ψ is the heading angle, measured as the angle that goes from $\vec{i}_{\overline{U}}$ to $\vec{i}_{\overline{B}}$. The second term in 10 is the part that is slightly mode subtle. Recall that $\vec{i}_{\overline{S}}$ varies with ϕ, so then we have two questions:

1. what is the expression for $\vec{i}_{\overline{S}}$ as a function or ϕ and,

2. how do we find the appropriate ϕ for a specified state of the system (this is the projection operation).

Answering the first question isn't terrible. The unit tangent vector for any generic path described by $\vec{r}(\phi)$ is given by:

$$\frac{\frac{d}{d\phi}\vec{r}(\phi)}{\|\frac{d}{d\phi}\vec{r}(\phi)\|}. \tag{12}$$

Therefore in our case, $\vec{i}_{\overline{S}}$ is given by:

$$\vec{i}_{\overline{S}} = \frac{W\cos(2\pi\phi)}{\sqrt{4H^2\cos^2(4\pi\phi) + W^2\cos^2(2\pi\phi)}}\vec{i}_{\overline{U}} - \frac{2H\cos(4\pi\phi)}{\sqrt{4H^2\cos^2(4\pi\phi) + W^2\cos^2(2\pi\phi)}}\vec{j}_{\overline{U}}. \tag{13}$$

We could solve the second question in two ways, we could take an analytical approach, or a numerical approach. I think that the analytical approach might be possible, but I've worked on it for a while and it's certainly nontrivial. Since that's not the focus of this work, it's probably better to take the numerical approach.

The goal is to find the appropriate value of ϕ so that we know what the correct $\vec{i}_{\overline{S}}$ is to use in equation 10. If we call that value ϕ^*, then it is given by:

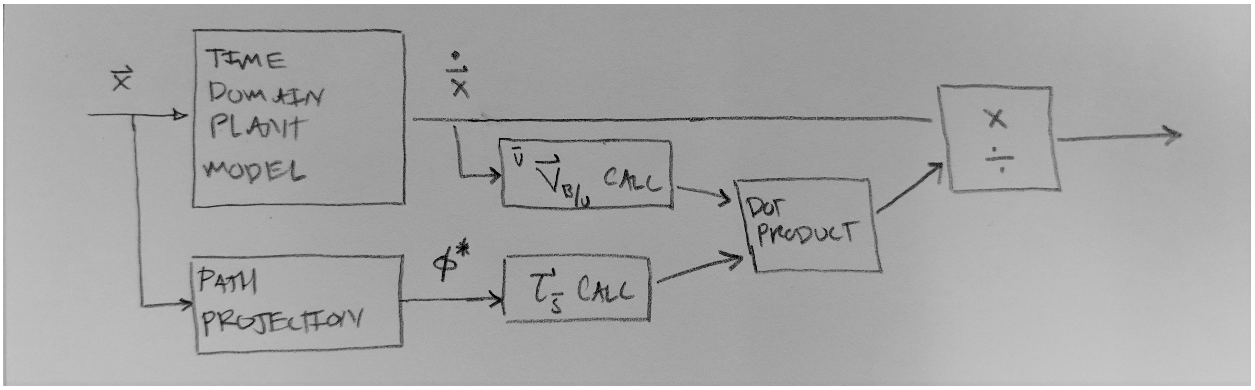
$$\phi^* = \arg\min_{\phi}\|\vec{r}_{B/S}\|, \tag{14}$$

where the position vector $\vec{r}_{B/S}$ relates the position of the body-fixed coordinate system to a point S on the path. Specifically, $\vec{r}_{B/S} = \vec{r}_{B/U} - \vec{r}_{S/U}$ where $\vec{r}_{S/U}$ is a function of ϕ. If we restrict the domain of this problem then I think it becomes pretty easy to solve quickly (thus keeping our simulations from taking forever). Specifically, I've already implemented the following in Simulink:

$$\begin{aligned} \phi^* &= \underset{1}{\mathrm{rem}}\left(\arg\min_{\phi}\|\vec{r}_{B/S}\|\right) \\ &\text{subj.to} : \phi_{k-1} \leq \phi \leq \phi_{k-1} + \Delta\phi \end{aligned} \tag{15}$$

Where ϕ_{k-1} is the value of ϕ found at the previous time step, and $\Delta\phi$ is a user-set parameter. Also, the function $\underset{1}{\mathrm{rem}}$ indicates the mathematical remainder under division by 1 (I don't know if there's a better way to notate this). This just makes sure that we keep ϕ in the appropriate range.

In summary, the block diagram shown in the figure below is what I'm proposing as the representation of equation 4. So I would build this in Simulink and run `linmod` on it with every $\vec{x}(s)$ from the previous iteration as the linearization points to get our path linearization.



$\vec{x}$
TIME DOMAIN PLANT MODEL
$\dot{\vec{x}}$
$^{U}\vec{V}_{B/U}$ CALC
DOT PRODUCT
PATH PROJECTION
ϕ^*
τ_s CALC
×
÷