Symbolic Lagrangian Solver User Guide

Ran Tian

February 27, 2016

1 Mathematical Background

Given a mechanical system with n degrees of freedoms, we can use q and $\dot{q}$ to represent its position and speed of all the degree of freedoms. A proper choice of state variables is then $x = [q, \dot{q}]^T$. The state-space equations of this mechanical system is generally:

$$\dot{x} = f(x, u) \tag{1}$$

Expressing a system in state-space form can be very useful for simulations. To get the state-space expression, we start from the systems Lagragian:

$$L = T(\dot{q}) - V(q) \tag{2}$$

where $T(\dot{q})$ is kinetic energy and $V(q)$ is potential energy. We only consider kinetic energy of the form:

$$T(\dot{q}) = \frac{1}{2}\dot{q}^T M(q)\dot{q} \tag{3}$$

This form of kinetic energy is very general.

The Eular-lagrange equation for the system's Lagrangian is:

$$\frac{d}{dt}(\frac{\partial L}{\partial \dot{q}_i}) - \frac{\partial L}{\partial q_i} = u_i - fric_i, \quad i = 1, 2, ..., n \tag{4}$$

where $fric_i$ is the friction on the ith degree of freedom.

Compute the Eular-Lagrange equation on each degree of freedom and group them together we get this very important form of system dynamics:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + \frac{\partial V(q)}{\partial q} = u - fric \tag{5}$$

where entries of $C(q, \dot{q})$ matrix can be computed by Christoffel Symbols of the first kind:

$$c_{kj} = \sum_{i=1}^{n} c_{ijk}(q)\dot{q} = \sum_{i=1}^{n} \frac{1}{2}\{\frac{\partial M_{kj}}{\partial q_i} + \frac{\partial M_{ki}}{\partial q_j} - \frac{\partial M_{ij}}{\partial q_k}\}\dot{q}_i \tag{6}$$

The standard state-space representation is then readily available:

$$\begin{aligned} \dot{q} &= \dot{q} \\ \ddot{q} &= M^{-1}(q)[-C(q, \dot{q})\dot{q} - \frac{\partial V(q)}{\partial q} + u - fric] \end{aligned} \tag{7}$$

In nonlinear systems, we often need to linearize the state-space equation around a nominal point $(\hat{x}, \hat{u})$. The states and input can be divided into $x = \hat{x} + \tilde{x}$, $u = \hat{u} + \tilde{u}$, where $\tilde{x}$ and $\tilde{u}$ are deviation from the nominal point. The deviations form an approximate linear system:

$$\dot{\tilde{x}} \approx \left.\frac{\partial f(x,u)}{\partial x}\right|_{x=\hat{x}} \tilde{x} + \left.\frac{\partial f(x,u)}{\partial u}\right|_{u=\hat{u}} \tilde{u} \tag{8}$$

Here $f(x, u)$,which has two entries, is exactly the same as (7). Notations are usually simplified by:

$$\dot{\tilde{x}} \approx A(\hat{x})\tilde{x} + B(\hat{u})\tilde{u} \tag{9}$$

Where A is a $2n$ by $2n$ matrix, and the size of B is $2n$ by n.

However, it is not usually possible to get the above expression for a complicated system by hand. Clearly the first entry of $f(x, u)$ is already a linear function, so we only need to linearize the second entry. To make things simple, we set:

$$\phi(q) = \frac{\partial V(q)}{\partial q} \tag{10}$$

$$U = -C(q, \dot{q})\dot{q} - \phi(q) + u - fric \tag{11}$$

So we have:

$$f_2(x, u) = \ddot{q} = M(q)^{-1}U \tag{12}$$

When calculating partial derivative of $f_2(x, u)$, a trick is used to avoid solving partial derivative of M^{-1} directly:

$$\frac{\partial M^{-1}}{\partial x} = -M^{-1}\frac{\partial M}{\partial x}M^{-1} \tag{13}$$

Thus:

$$\begin{aligned} \frac{\partial f_2(x,u)}{\partial x} &= \frac{\partial M^{-1}}{\partial x}U + M^{-1}\frac{\partial U}{\partial x} \\ &= -M^{-1}\frac{\partial M}{\partial x}M^{-1}U + M^{-1}(-\frac{\partial C}{\partial x}\dot{q} - C\frac{\partial \dot{q}}{\partial x} - \frac{\partial \phi}{\partial x} - \frac{\partial fric}{\partial x}) \end{aligned} \tag{14}$$

Notice that (14) is a n by $2n$ matrix. Then A can be written as:

$$A = \begin{bmatrix} Zeros(n) & I(n) \\ \frac{\partial f_2(x,u)}{\partial x} & \end{bmatrix} \tag{15}$$

And B matrix is easy to get by calculating the partial derivative of $f(x, u)$ with respect to u, so we just give the result:

$$B = \begin{bmatrix} Zeros(n) \\ M^{-1} \end{bmatrix} \tag{16}$$

$I(n)$ represents a n by n identity matrix, and $Zeros(n)$ is a n by n zero matrix.

2 Programming with Symbolic Lagrangian Solver (SLS)

The idea of SLS is to solve the above problem in a mixed symbolic/numerical way in Matlab. The symbolic part process symbolically assigned kinetic energy, potential energy and friction, and yields function files for numerical simulation.

SLS package contains three files:

- sym_lagrangian_solver.m
- qddot_fun.m

- linearized_model.m

User needs to specify a system's kinetic energy $T(\dot{q})$, potential energy $V(q)$ and friction $fric(q, \dot{q})$ symbolically in Matlab. The kinetic energy must be able to be transform to the form $T = \frac{1}{2}\dot{q}^T M \dot{q}$, although user doesn't have to write it in this form.

Throw symbolic T, V, $fric$ and x into sym_lagrangian_solver.m as:

sym_lagrangian_solver(T, V, $fric$, x)

The following functions will be written to the current directory in Matlab:

- M_fun.m
- M_jac_fun.m
- C_fun.m
- C_jac_fun.m
- fric_fun.m
- fric_jac_fun.m
- phi_fun.m
- phi_jac_fun.m

Each of this function takes in numerical states ($x = [q, \dot{q}]^T$) and returns the corresponding matrix or array. It is clearly known that not all the matrices or arrays depend on the speed part of states, but we still takes in all the states for simplicity.

We don't compute the state-space equation symbolically in that inverting the M matrix symbolically is unacceptablly expensive when the system is not simple. Throw numerical states ($x = [q, \dot{q}]^T$) and numerical system input u, then qddot_fun.m computes acceleration of all degree of freedoms numerically using the above generated functions. Again by giving numerical states and system input, linearized_model.m returns numerical matrix A and B.