MATLAB Assignment 5

October 23, 2014

Due: October 30, 2014

When working on this assignment you might want to take a look at MATLAB code that was developed by students who took ME451 in previous years. The students back then did not come up with identical solutions. Take a look at their solutions and develop your own.
2010: `http://sbel.wisc.edu/Courses/ME451/2010/SimEngine2D/index.htm`
2011: `http://sbel.wisc.edu/Courses/ME451/2011/SimEngine2D/index.htm`

Turning in your assignment: place all your files in a directory called "lastName_Matlab_05", zip that directory, and upload the resulting file "lastName_Matlab_05.zip" in the appropriate Dropbox Folder at Learn@UW.

Problem 1. Consider the Example 3.3.4 at page 68 of the textbook. Generate a pair of input files sliderCrank.acf and sliderCrank.adm that will be used in a future assignment by your simEngine2D to perform 10 seconds worth of Kinematics Analysis of the mechanism using a step size of $\Delta t = 0.001$ seconds. Specify in the ACF file that the output should be saved at 500 intermediate points. Additionally, consider that a motion is applied to this mechanism as follows: $\phi_1(t) - t^2 = 0$.

Notes:

- Consider three bodies for this model. There is no need to define a "ground" body (body 4 in the example 3.3.4).
- Assume the following inertia properties: for body 1: $m_1 = 2.0$ and $\bar{J}_1 = 0.3$; for body 2: $m_2 = 3$ and $\bar{J}_2 = 0.2$; and for body 3: $m_3 = 5$ and $\bar{J}_3 = 0.5$.
- For the initial configuration of the mechanism at $t = 0$, choose the setup that places the mechanism in an "all stretched out" configuration. That is, using the the dimensions shown in Figure 3.3.6, for body 1: $x_1(t_0) = 2$ and $y_1(t_0) = 0$; for body 2: $x_2(t_0) = 7$ and $y_2(t_0) = 0$; and for body 3: $x_3(t_0) = 10$ and $y_3(t_0) = 0$.
- Please use the following set of constraints to model this mechanism: two absolute constraints for the motion of body 1 with respect to ground (`Constraint-1` and `Constraint-2`, one equation each); one revolute joint between bodies 1 and 2 (`Constraint-3`, note is has two equations); one revolute joint between body 2 and 3 (`Constraint-4`, it has two equations); two absolute constraints to capture the kinematics of body 3 (`Constraint-5` and `Constraint-6`, one equation each - the former is an absolute-y, the latter is an absolute angle); and finally, one absolute angle driving constraint, `Constraint-7`: $\phi_1(t) - t^2 = 0$.

Problem 2. Recall the discussion in class about the five stages we have to go through in order to handle a certain physical joint/constraint:

Stage 1 - understand what it does;

Stage 2 - generate constraint *equations* $\mathbf{\Phi}(\mathbf{q}, t)$ that capture mathematically the behavior of the physical joint;

Stage 3 - generate the Jacobian $\mathbf{\Phi}_{\mathbf{q}_i}$ and $\mathbf{\Phi}_{\mathbf{q}_j}$;

Stage 4 - generate $\nu(\mathbf{q}, t)$, the RHS of the velocity equation;

Stage 5 - generate $\gamma(\dot{\mathbf{q}}, \mathbf{q}, t)$, the RHS of the acceleration equation.

For this problem you'll have to do two things:

(a) Generate a MATLAB function that evaluates $\mathbf{\Phi}(\mathbf{q}_i, t)$, $\mathbf{\Phi}_{\mathbf{q}_i}$, $\nu(\mathbf{q}_i, t)$, and $\gamma(\mathbf{q}_i, \dot{\mathbf{q}}_i, t)$ for the *AbsoluteX* constraint.

A possible MATLAB implementation is provided below. It draws on constraint attributes that you will have parsed from the ADM file which here I assume were stored in a MATLAB data structure called `obj`. **Note** that you don't have to follow the code structure suggested, which may not fit the data structures you have decided upon. Moreover, you are not required to implement this functionality in a single MATLAB function. You may implement four functions to calculate each of these four kinematic quantities separately.

(b) Parse the file sliderCrank.adm and then invoke your function from point (a) above to compute $\mathbf{\Phi}(\mathbf{q}_i, t)$, $\mathbf{\Phi}_{\mathbf{q}_i}$, $\nu(\mathbf{q}_i, t)$, and $\gamma(\mathbf{q}_i, \dot{\mathbf{q}}_i, t)$ associated with `Constraint-1` at the **initial time** $t_0 = 0$. Once you compute these quantities, post this information (for Stage 2 through Stage 5) on the class forum to check against results obtained by your colleagues. When computing γ (Stage 5) assume that all entries in $\dot{\mathbf{q}}_i$ are zero.

Listing 1: Sample MATLAB function for *absoluteX*

```
function [Phi, Phi_q, Nu, Gamma] = absXconstraint(obj, t, qi, qdi, flags)
% absXconstraint - evaluate quantities for an AbsoluteX constraint.
%            The quantities that this function computes are based on the entries
%            in the array 'flags'. If an entry in 'flags' is 0, the corresponding
%            output argument will be empty.

    % Extract information from the specified Cartesian generalized coordinates.
    x = qi(1);  phi = qi(3);
    c = cos(phi); s = sin(phi);

    % Initialize outputs to empty, in case they will not be calculated.
    Phi = [];
    Phi_q = [];
    Nu = [];
    Gamma = [];

    if flags(1)
        % Calculate Phi (Stage 2).
        Phi = x + obj.sI(1)*c - obj.sI(2)*s - obj.fun(t);              % a scalar
    end
    if flags(2)
        % Calculate Jacobian (Stage 3).
        Phi_q = [1,  0,  -obj.sI(1)*s - obj.sI(2)*c];                  % a 1x3 row vector
    end
    if flags(3)
        % Calculate RHS of velocity equation (Stage 4)
        Nu = obj.funD(t);                                              % a scalar
    end
    if flags(4)
        % Calculate RHS of acceleration equation (Stage 5)
        phid = qdi(3);
        Gamma = (obj.sI(1)*c - obj.sI(2)*s) * phid^2 + obj.funDD(t);   % a scalar
    end
end
```

Problem 3. Same as above, but now for an *AbsoluteY* constraint. Use the resulting function to compute $\mathbf{\Phi}(\mathbf{q}_i, t)$, $\mathbf{\Phi}_{\mathbf{q}_i}$, $\nu(\mathbf{q}_i, t)$, and $\gamma(\mathbf{q}_i, \dot{\mathbf{q}}_i, t)$ associated with `Constraint-5` at the **initial time** $t_0 = 0$ and assuming that all entries in $\dot{\mathbf{q}}_i$ are zero. Once you compute these quantities, post this information (for Stage 2 through Stage 5) on the class forum to check against results obtained by your colleagues.

Problem 4. Same as above, but now for an *AbsoluteAngle* constraint. Use the resulting function to compute $\mathbf{\Phi}(\mathbf{q}_i, t)$, $\mathbf{\Phi}_{\mathbf{q}_i}$, $\nu(\mathbf{q}_i, t)$, and $\gamma(\mathbf{q}_i, \dot{\mathbf{q}}_i, t)$ associated with `Constraint-6` and `Constraint-7` at the **initial time** $t_0 = 0$ and assuming that all entries in $\dot{\mathbf{q}}_i$ are zero. Once you compute these quantities, post this information (for Stage 2 through Stage 5) on the class forum to check against results obtained by your colleagues.

Problem 5. Same as above, but now for a *RevoluteJoint* constraint. Use the resulting function to compute $\mathbf{\Phi}(\mathbf{q}_i, \mathbf{q}_j, t)$, $\mathbf{\Phi}_{\mathbf{q}_i}$, $\mathbf{\Phi}_{\mathbf{q}_j}$, $\nu(\mathbf{q}_i, \mathbf{q}_j, t)$, and $\gamma(\mathbf{q}_i, \mathbf{q}_j, \dot{\mathbf{q}}_i, \dot{\mathbf{q}}_j, t)$ associated with `Constraint-3` and `Constraint-4` at the **initial time** $t_0 = 0$ and assuming that all entries in $\dot{\mathbf{q}}_i$ and $\dot{\mathbf{q}}_j$ are zero. Once you compute these quantities, post this information (for Stage 2 through Stage 5) on the class forum to check against results obtained by your colleagues.