EE4323 Industrial Control Systems
Homework Assignment 2

Asst. date: 8 May 2017 – Due date: 12 May 2017

Department of Electrical & Computer Engineering
University of New Brunswick

In this assignment you are going to become more familiar with of the use of MATLAB for the simulation of nonlinear systems. The physical problem is, once again, an electro-mechanical system with a gear train and load, as considered in the first homework assignment. This homework will give you some "hands-on" experience in **writing MATLAB models, numerical integration, and state-event handling**.

1. If you haven't already done so, download the state-event handling simulation routines from my web site, specifically `http://www.ee.unb.ca/jtaylor/HS_software.html`; download the zip file and install the routines in your account. There is a user's manual, which is more informative than my slides, and some sample problems. This site is obsolete-looking, but I think it's still clear...

2. **Read the accompanying manual (tutorial guide) provided with these routines!** Also, study the example m-files implementing mode-based models of simple systems. The m-files "stick.m" and "stick_seh.m" are good models for this assignment.

3. Look at `ode45m.m` - that will give you an idea of a high-powered numerical integrator; I mentioned in class the exotic coefficients and the impenetrable step-size adjustment logic ... then look at `eufix1.m` for the other end of the spectrum, it's just the basic Euler algorithm with standard documentation on the front end and minimal initialization. *We will use* `eufix1, ode45m, ode45` *(the current built-in* MATLAB*/ routine) and* `ode45_101` *in this assignment.*

We will investigate the behavour of a quite simple electro-mechanical system, comprised of a permanent magnet DC motor with a thermal model added to be sure that we aren't going to overheat the motor under the operating regimes studied. In other words, we will check to see if a non-thermal model would have exceeded its "envelope".

Since this is not primarily a modelling exercise you should use the following dynamic equations (raw model):

$$\begin{aligned} e_{in}(t) &= R_A i_A + L_A \dot{i_A} + \alpha\omega_1 \\ J_1\dot{\omega}_1 + B_1\omega_1 &= \alpha i_A + R_1 f_c \\ J_2\dot{\omega}_2 + B_2\omega_2 + B_{2C}\ \mathrm{sign}(\omega_2) &= -\tau_{LE} - R_2 f_c \\ C_{TM}\dot{\theta}_M + (\theta_M - \theta_A)/R_{TM} &= i_A^2 R_A \end{aligned} \tag{1}$$

Everything should be familiar, except for the last (thermal) relation which is similar to an electrical capacitor-resistor model with thermal capacity C_{TM}, "resistive losses" to ambient

temperature via R_{TM}, where θ_M, θ_A are motor and ambient temperatures, respectively, and the input is $i_A^2 R_A$. Finally note that there is an external torque acting on the load, τ_{LE} but none on the motor. There are thus three inputs: $u_1 = e_{in}$ = voltage applied to the motor, $u_2 = \tau_{LE}$ = external load torque, and $u_3 = \theta_A$ = the ambient temperature.

We will explore a number of issues with respect to simulation (behavour of algorithms) as well as nonlinear effects. For the following exercises you should develop nice scripts to execute them; include these in your submission:

1. Manipulate the above raw model equations to obtain the state-vector differential equations given the state $x = [\, i_A \;\; \omega_2 \;\; \theta_M \,]^T$ and input $u = [\, e_{in} \;\; \tau_{LE} \;\; \theta_A \,]^T$, and write the corresponding MATLAB model (m-file). Hint: To proceed, simply solve the second relation for the contact force f_c and substitute into the third ordinary differential equation and use $\omega_1 = N\omega_2$ where $N = R_2/R_1$ is the gear ratio to eliminate ω_1. Here are the parameter values, **ready to insert in your model**:

   ```
   global E_0 Hz Tau_0 T_Amb B_2C %%% make these settable from a script
   % motor parameters, Nachtigal, Table 16.5 p. 663
   J_1 = 0.0035;     % in*oz*s^2/rad
   B_1 = 0.064;      % in*oz*s/rad
   % electrical/mechanical relations
   K_E = 0.1785;     % back emf coefficient, e_m = K_E*omega_m
   K_T = 141.6*K_E; % torque coeffic.; in English units K_T is not = K_E!
   R_A = 8.4;        % Ohms
   L_A = 0.0084;     % H
   % gear-train and load parameters
   J_2 = 0.035;      % in*oz*s^2/rad % 10x motor J
   B_2 = 2.64;       % in*oz*s/rad (viscous)
   N = 8;            % motor/load gear ratio; omega_1 = N omega_2
   % Thermal model parameters
   R_TM = 2.2;       % Kelvin/Watt
   C_TM = 9/R_TM;    % Watt-sec/Kelvin (-> 9 sec time constant - fast!)
   ```

   The above block will comprise 2/3 of your basic MATLAB function! Parameters you want to vary in the following exercises should be made **global** [1]

2. Run your model for 1.2 seconds, with e_{in} defined as a 120 volt step applied at $t = 0.05$ sec. and τ_{LE} defined as a 80 N-m step applied at $t = 0.5$ sec., using `ode45m` to produce a result *similar* to that shown in Fig. 1 (including labels, parameter values etc. to provide full documentation of the run, as shown. For this use either the `text` command or `gtext` command which allows you to place text with a mouse click. Define the text in the form `['E_0 = ',num2str(E_0,3)]` so your are sure the value printed is correct – if you insert the value manually (e.g., `gtext('E_0 = 120')` you are sure to forget to change it sometimes! Time the run by putting the command `timer = clock` before the call and `Tsim1 = etime(clock,timer)` immediately after, and record the number of integration time steps taken:

[1] **Note:** to use globals effectively, you must have a global statement in both your model as well as your script; make them identical, as in the above MATLAB code block. **Do not** assign values to these parameters in your model! Set them only in your script or on the command line.

```
timer = clock;
[t1,x1] = ode45m('mlt',tspan,x0);
Tsim1 = etime(clock,timer), % integration time
Len1 = length(t1), % number of time-points
```

Just paste the above lines into your scripts. **Warning:** some integration algorithms define the simulation time by a vector, `tspan = [t0 tfinal]`, others by two scalars, `t0,tfinal`. Observe that there is an initial "spike" in current which is reduced as the angular velocity reaches steady-state, thus producing significant back emf. Also, you will notice a small increase in current and decrease in angular velocity when the load torque is applied at $t = 0.5$. What other observations about "the physics" do you notice? Hand in a listing of the ordinary differential equation model (your MATLAB function) and the script to produce your plot(s).

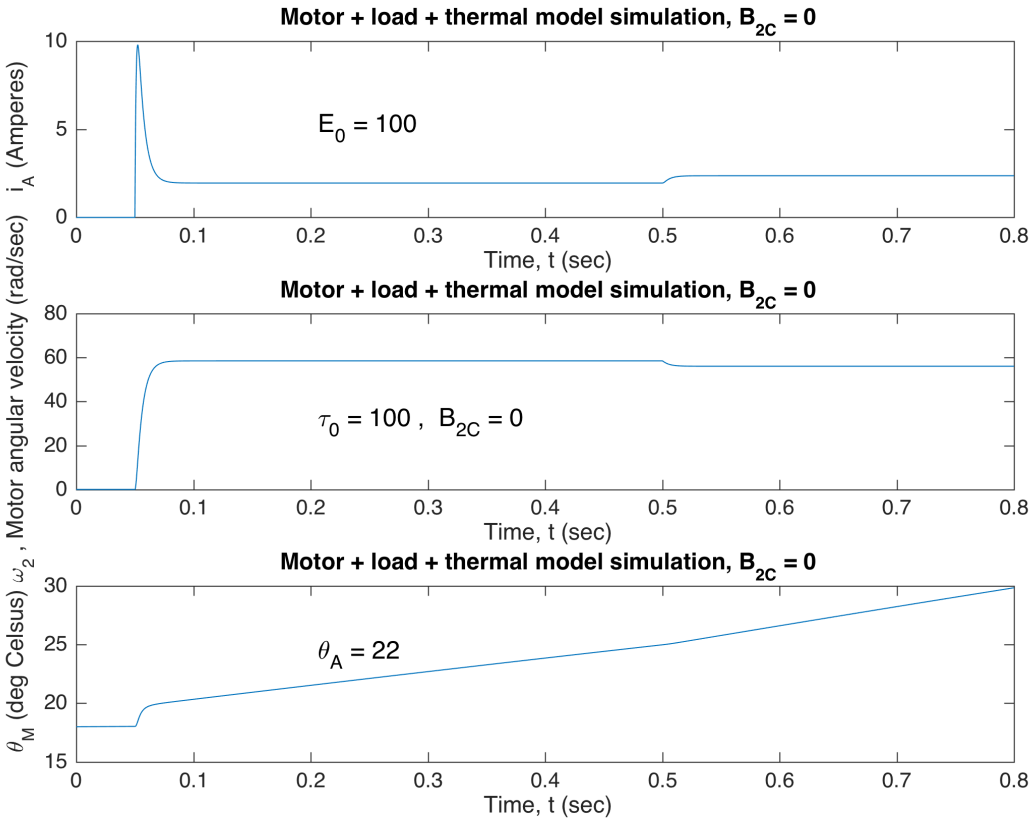

Figure 1: MATLAB plot similar to yours for Q. 2

3. Repeat this check using `eufix1`; determine the value of the integration step h such that the error in peak current (at about $t = 0.051$ sec.) is no more than a few %, then time the run and record the number of time-points as above. (Hint: use the `max` function to obtain the current peak value for each run.)

4. You will notice that the temperature is steadily climbing – will the motor melt?? Find out by running a much longer simulation, with `tfinal = 40`. Comment on this result.

5. Now we will study a scenario where the motor is continually reversing – in other words, when we apply a 5 Hz sine wave of amplitude E_0, starting at $t = 0.05$ sec. Use `ode45m`, set the final time to capture four or five cycles, use zero initial conditions except $\theta = 18^o$ C, set $E_0 = 90$ volts and set B_{2C} to 140 Newtons. Your result should be similar to the plots in Fig. 2. You should observe significant nonlinear behavour in both the angular velocity (the brief "flat spots" where the motor sticks) and the current (sharp change in $\dot{i_A}$ when the motor sticks and thus back emf is zero). Again, collect the run time and number of time points for this case – even for this fairly tame case `ode45m` takes quite a bit of time. To see why, zoom in on one of the regions where the motor is "stuck".

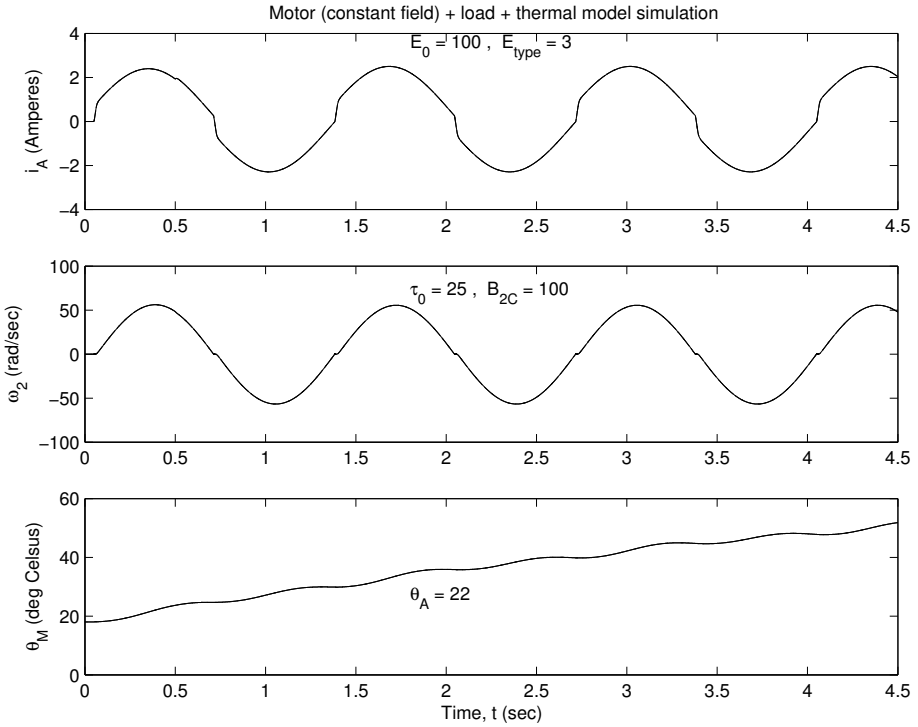

Figure 2: Motor response to a sinusoidal input, $B_{2C} = 100$ N (but different paameter values)

6. Based on the example `sticky / sticky_seh` copy your motor model and add the state-event handling code to handle the Coulomb friction / sticking correctly. Repeat step 5 using the mode-based routine `ode45_101`, compare it's simulation behavour with that of `ode45m` using the same performance measures (simulation time, number of time steps taken). Also, look closely (zoom) to see how the different algorithms deal with the fact that the motor should not move until the static friction (`B_2C`) permits it.

Document and present all the above results systematically and completely, with models, scripts and plots as needed.