SPACE ROBOTICS AND AUTONOMY

MODULE EEEM029

12th October 2018

Coursework

Submission closes on Tuesday, 4th December 2018 at 4pm

Early submissions are encouraged.

Objective: The purpose of this assignment is to develop skills in simulating the kinematics of a robot manipulator and designing a controller using Simulink and Matlab, which are user friendly coding environments dynamically linked to each other through the Matlab Workspace.

Marking Criteria:

The coding coursework carries 20% of the total credits for this module. Remaining 80% of the total credits are for the 2 hr unseen written examination, which includes topics taught by both Dr. S. Fallah and Prof. Y. Gao.

Question: Consider a six-degree of freedom industrial manipulator with revolute joints and spherical wrist such that last three axes intersect. The link parameters are shown in the table below.

Joint i	Joint angle θ_i (°)	Twist angle α_i (°)	Link length a_i (m)	Offset distance d_i (m)
1	θ_1	-90	0	0
2	θ_2	0	0.5	0.25
3	θ_3	90	0	0
4	θ_4	-90	0	1
5	θ_5	90	0	0
6	θ_6	0	0	0.5

Let the composite Denavit-Hartenburg matrix be

$$T_{o6} = \begin{bmatrix} -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 1 \\ 0 & -1 & 0 & 1 \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

1) Develop software coding in Matlab to compute the inverse kinematic solutions for all possible arm configurations so as to reach the given target position and orientation. Name this compilable file as “Inverse.m”. Results for your inverse kinematics solutions, expressed in degrees, should be displayed in the command window when “Inverse.m” file is compiled. [*Note: use four-quadrant inverse tangent function (atan2d(y,x)) in Matlab*]

[4 marks]

2) Validate the accuracy of all the inverse kinematics solutions obtained in Part (i) using forward kinematics. Name this compilable file as “Forward.m” and display the following results in the command window when this file is compiled:

- Homogeneous transformation matrices A_i^{i-1} where i = 1 to 6
- Denavit Hartenburg transformation matrix using forward kinematics

[3 marks]

3) In this part of the assignment, some codes will be written in a Matlab file (.m file extension) and others will be written in Simulink (.slx file extension) using different building blocks.

Note: For designing the PID controller, you need to use only one of the Inverse Kinematics solutions from Part (1). Please use the solution set corresponding to theta1 oriented to the right (positive value of square root) and theta2 in the elbow down configuration (positive value of square root) along with the corresponding values for theta3 to theta6. This means that even though you have multiple solutions from your inverse kinematics code, you will be designing only one PID controller using the recommended inverse kinematics solution.

Two files will be given to you:

- "I_F_matrices.m" that includes the transformation matrices describing the initial and final poses of the manipulator's tip. The corresponding matrices are:

$$Imatrix = \begin{bmatrix} 1 & 0 & 0 & \frac{1}{2} \\ 0 & 1 & 0 & \frac{1}{4} \\ 0 & 0 & 1 & \frac{3}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad Fmatrix = \begin{bmatrix} -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 1 \\ 0 & -1 & 0 & 1 \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- "PID_Controller.slx" with the basic blocks needed to build the controller using the dynamic matrices using the Euler-Lagrange equation of motion. The matrices are D, C and G for the inertia, Coriolis & centrifugal forces and gravity matrices respectively.

(i) After the completion of the inverse and forward kinematics codes in Part 1 and Part 2, use their appropriate contents to create two different (.m) files with the following names:

"InverseKinematic.m"
"ForwardKinematic.m"

- The "InverseKinematic" file is to be a function with "T" (transformation matrix) as input and "theta1, theta2, theta3, theta4, theta5, theta6" as outputs.

  Help: the [n s a] vector are found using the T matrix.

- The "ForwardKinematic" file is to be a function with "theta1, theta2, theta3, theta4, theta5, theta6" as inputs and "X, Y, Z" as outputs.

(ii) Run the file "I_F_matrices.m" and make sure the two variables "Imatrix" and "Fmatrix" are in the Matlab workspace.

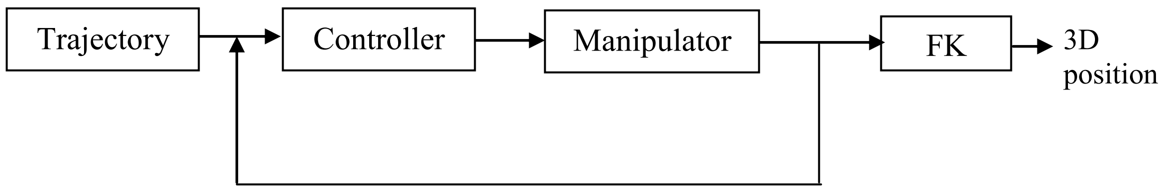
(iii) Open the "PID_Controller.slx" Simulink file. There are four main blocks:

- The "Trajectory" block (blue) includes the trajectory already designed to be tracked by the manipulator. Note that inside this block, there are two inputs called "Imatrix" and "Fmatrix" and you should NOT make any changes to this block given (make sure you run the I_F_matrices m file).

- The "D_matrix" block that describes the inertia matrix of the manipulator.
- The "C_matrix" block that describes the Coriolis and centrifugal forces matrix of the manipulator.
- The "G_matrix" block that describes the gravity matrix of the manipulator.

Important: In Simulink tool bar, go to "Simulation" then "Model configuration parameters". In the window, click on "Data Import/Export", "Additional parameters" at the bottom. In the "Output options" list, choose "Produce specified output only" and type the following in the adjacent space: [0:0.2:10]. This is only to force Simulink to compile the code at the desired specified time steps.

(iv) Using the equation of motion and the PID controller equation given in the coursework, assemble the building blocks by adding the necessary links and gains from the Simulink library in the following closed loop architecture:



FK: ForwardKinematic

Use:

- **Gain:** for the P, I and D terms/matrices of the controller.
- **Integrator**: for the Integral term/matrix of the controller and to integrate the acceleration into a velocity and to angle value.
- **LU Inverse**: for the inverse of a matrix.
- **Product:** for multiplication. Double click on block and choose "Matrix (*)" for matrix multiplication on the list.
- **Sum**: for summation (you can change the sign by double clicking on the block).
- **Demux**: to decouple vectors (such as the theta vector).
- **Scope:** to visualise signals.
- **To workspace**: to send the values of X, Y and Z back to Matlab workspace for plotting.

Important: please do not change the naming conventions given above as it will alter the compilation of the codes. Also do make sure that all files are in the same folder before you run the code.

[8 marks]

4) Plot the joint torques, accelerations, velocities and joint angles using the "scope" blocks in Simulink. Further, plot the 3D trajectory using the instruction "plot3(X,Y,Z)" in a Matlab file with name "ThreeDplot.m". *Note*: the X, Y and Z values will be sent from Simulink to the workspace through the "to workspace" block.

[5 marks]

Coding submission process:

Please upload your coding files to the assignment submission folder via SurreyLearn before the deadline. Coding should be error free.

We recommend you to upload one .zip file that contains all your files listed below:

Inverse.m

Forward.m

I_F_matrices.m

InverseKinematic.m

ForwardKinematic.m

PID_Controller.slx

ThreeDplot.m

You're allowed to submit the .zip file any number of times until the deadline. Please note that SurreyLearn will record the timing of all your submissions and it will retain all your older submissions. We will only use the final submission for evaluation purpose.