MECH 467/541 – Computer Control of Mechatronics Systems
Prof. Y. Altintas
The University of British Columbia, Department of Mechanical Engineering

Project # III: Simulation of Contouring Performance in Coordinated Two-Axis Motion

This project will demonstrate how to design a 2-axis machine controller from beginning to end, and it wraps up the material you learned in MECH 467/541.

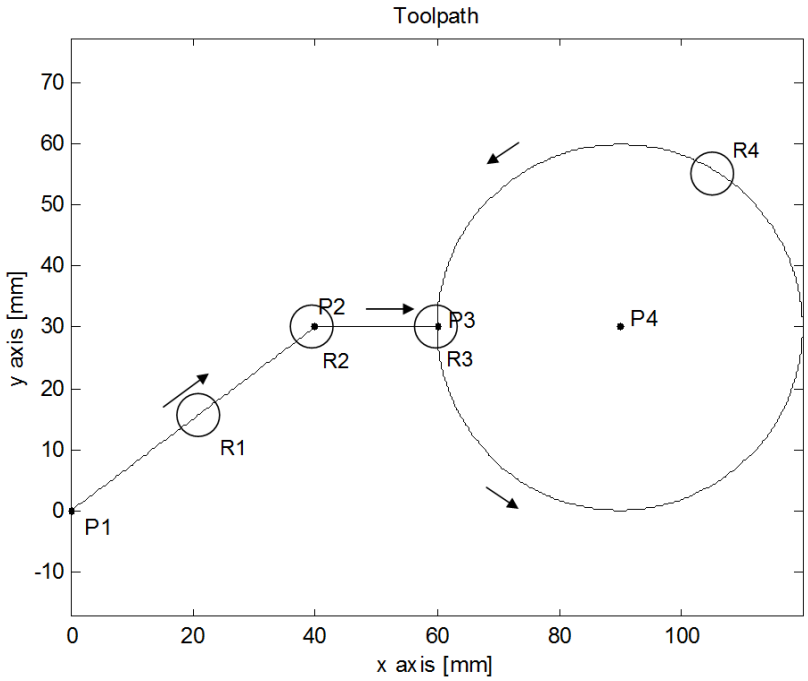

Figure 1: Reference toolpath to be used in contouring performance evaluation.

A sample toolpath is shown in Figure 1, comprising two linear segments and a full circle. The coordinates of the segment end points are P1(0,0) [mm], P2(40,30) [mm], and P3(60,30) [mm]. The center of the circle is P4(90,30) [mm]. Read the entire handout before starting to work on the prelab.

Pre-Lab

Part A – Trajectory Generation

Using trapezoidal feedrate profiling, generate the axis commands as functions in discrete-time $x = x(k \cdot T_i)$, $y = y(k \cdot T_i)$, where $k = 0,1,\ldots$. Use fixed interpolation period as the servo control time interval ($T_i = T_s = 0.1[msec]$). Assume that the desired feedrate value is $F = 200$ [mm/sec] , and the tangential acceleration and deceleration values are $A = -D = 1000$ [mm/sec^2]. The feed profile starts from rest at P1, and also comes to a full stop at the corners P2 and P3. The final feedrate, after performing the counter-clockwise circular interpolation, is also zero. Plot the following:

A.1. The generated toolpath: x-axis position command versus y-axis position command (x_r vs. y_r),
A.2. The displacement (s), feedrate ($\dot{s}$), and tangential acceleration ($\ddot{s}$) profiles as functions of time ($t = kT_i$),
A.3. The axis position ($x_r,\ y_r$), velocity ($\dot{x}_r,\ \dot{y}_r$), and acceleration ($\ddot{x}_r,\ \ddot{y}_r$) commands as functions of time ($t = kT_i$).

Note: Each segment of the toolpath must start and finish at the **EXACT** given coordinates. Otherwise, it results in spikes in the feedrate and acceleration profiles. In order to reach the exact end points, make sure that all of the time periods for acceleration, constant feed, and deceleration periods are exact multiple integers of the interpolation time: $T_j^{'} = N_j T_i = \text{ceil}(\frac{T_j}{T_i}).T_i \leftarrow (j = 1,2,3)$. Then re-evaluate the commanded feedrate and accelerations based on the corrected time periods $T_j^{'}$.

Please make sure that related plots are kept together in an organized engineering report.
(**Note:** Writing MATLAB functions for linear and circular interpolations allows you to quickly interpolate any custom toolpath (as long as it consists of only lines and arcs). Read MATLAB help for "function".)

Part B – Two-Axis Controller Design

In this project, a ball screw XY table will be modeled, and its response will be simulated. The experimentally identified drive dynamics of the XY table are given as:

Axis Parameters	*X* Axis	*Y* Axis
Amplifier Gain (Ka [A/V])	1	1
Motor Torque Constant, (Kt [Nm/A])	0.49	0.49
Encoder Gain, (Ke [mm/rad])	1.59	1.59
Rotational Inertia, (Je [kgm^2])	4.36×10^{-4}	3×10^{-4}
Viscous Friction, (B [Nm/(rad/sec)])	0.0094	0.0091

Table 1: Axis Parameters

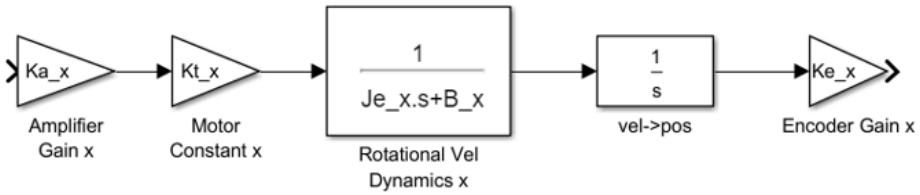

Figure 2: Linear Motor Driven Axis Model (X Axis)

B.1. The design methodology for the Integrator-Lead-Lag controller was introduced in the previous laboratory project. Using MATLAB commands and the given criteria in Table 2, design two different Lead-Lag position controllers (LBW and HBW) for each axis. Then cascade the designed lead-lag controller with the integral action (Use $K_i = \omega_c / 10$). You do not need to obtain the controller parameters manually. (Note: the integral action may slightly alter the PM and gain crossover frequency, but you do not need to make up for this effect in your lead-lag design.)

	Cross Over Frequency ω_c $[Hz]$ and Phase margin (PM [deg])
Case 1 (Low Bandwidth)	X Axis ω_c =20[Hz], PM = 60[deg] Y Axis ω_c =20[Hz], PM = 60[deg]
Case 2 (High Bandwidth)	X Axis ω_c =40[Hz], PM = 60[deg] Y Axis ω_c =40[Hz], PM = 60[deg]
Case 3 (Mismatched Dynamics)	X Axis ω_c =40[Hz], PM = 60[deg] Y Axis ω_c =20[Hz], PM = 60[deg]

Table 2: Lead Lag Design Criteria

B.2. Overlay the Bode plots of the open-loop system of the X axis with the LBW and HBW controllers. On another figure, plot the same graphs for the closed-loop system in the *s* domain.

B.3. Using MATLAB and for both X and Y axes, determine the poles and zeros of the closed loop systems (both continuous (s) and discrete (z) domain), bandwidth, overshoot, and rise time for both cases of LBW and HBW controllers. Tabulate your result. Use zero order hold equivalent of the X-Y drives' transfer functions to obtain their discrete equivalent. You can use SYSD = c2d(SYSC,TS, 'zoh') in your MATLAB code. Cascade the drive's open-loop transfer function in the *z* domain with the discrete equivalent of lead-lag and integral controllers by using Tustin's approximation ($s \approx \frac{2}{T_s}\frac{z-1}{z+1}$). In MATLAB code, you can use c2d command, but without 'zoh' (i.e., SYSD = c2d(SYSC,TS, 'tustin').
(Note: Make sure to use the command `Gcl=feedback(Gol,1)` to construct the closed-loop transfer function in MATLAB.)

Part C – Contouring Performance Simulation

C.1. Implement the discrete position controller cascaded with the continuous plant using a zero-order hold block in MATLAB/Simulink environment as shown in Figure 3. Apply the generated reference trajectories in A.1 to the two-axis position control system. Verify that your trajectories and controllers result in stable contouring of the toolpath. Include the simulated path.

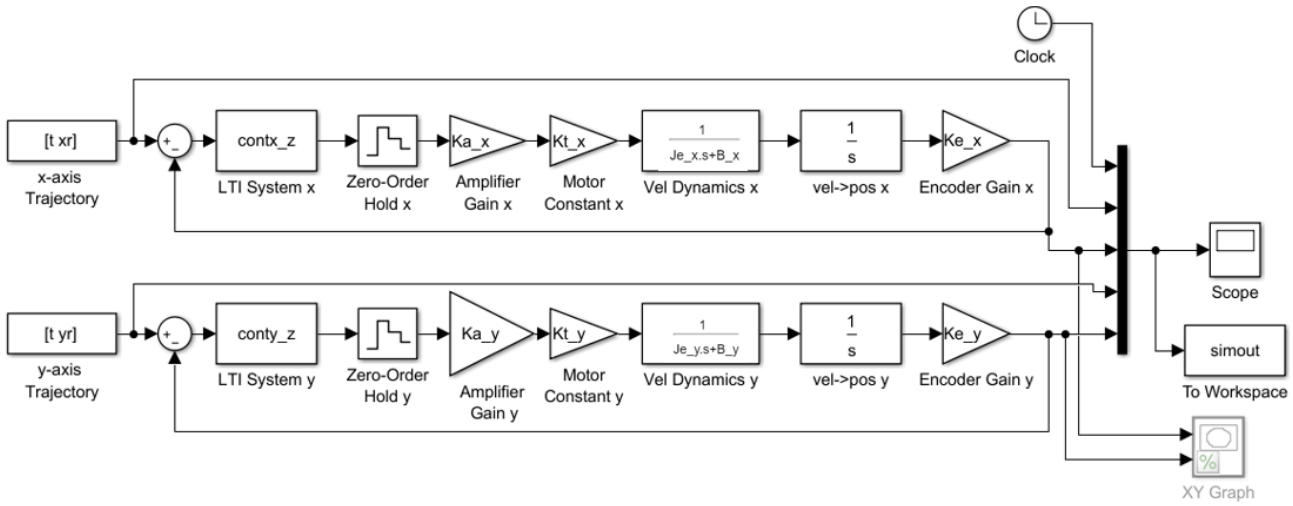

Figure 3: Simulink Model for Trajectory Simulation

Important Notes:

- Please make sure that t, x_r, and y_r have consistent dimensions. Both $[t\ x_r]$ and $[t\ y_r]$ should have dimensions of $[n \times 2]$, otherwise the Simulink model shown above will fail to run.
- In order to avoid extensive manual modifications in Simulink, build your Simulink model in terms of the parameters of the system, and assign them in your script. Also, for the controllers, you can use the general block ***LTI System*** from the ***Control System Toolbox*** in the Simulink library, and directly type in the name of the transfer function, which you will assign in your script.
- Use the ***Sources/From Workspace*** block to read your trajectory data (*t* [sec], x_r [mm], and y_r [mm]) existing in MATLAB's workspace, and use the ***Sinks/To Workspace*** block to import the Simulink outputs back into MATLAB workspace for evaluation. A ***mux*** block will also help to stack simulation arrays together. Once you build your Simulink model, you will not need to open it for simulations. Run it directly from your script using the command ***sim.***

C.2. This section is for the experimental implementation of **your own trajectory on the real XY table**. A picture of the actual ball screw driven XY table is given in Figure 4. The maximum workspace for the XY table is $100[mm] \times 100[mm]$. Define a toolpath using **only** linear and circular interpolation techniques. Please follow these notes in designing your trajectory:

- The geometry can be any shape as long as it contains only lines and arcs (but interpolating the whole path with many small linear segments is not recommended).
- Your designed trajectory should start from the origin (x=0, y=0), and it should be a **connected geometry** (the machine cannot jump from one point to another).
- Interpolate your gometry using **F = 100** [mm/sec] and **A=-D = 250** [mm/s^2] at **Ts=0.1**[msec] sampling time.
- Please consider the direction of the X and Y axes of the machine in your design. The X axis is from right to left, so if you plan to write your name, it will be mirrored!

If t, xr, and yr are the vectors (dimension $[n \times 1]$) of time and input trajectories, store them in a single structure variable as: `traj.t=t; traj.x=xr; traj.y=yr;`
Please use the **exact naming format** provided. Save the structure as a .mat file. For example, `save MyTraj traj` saves the variable `traj` in the MATLAB file MyTraj.mat.

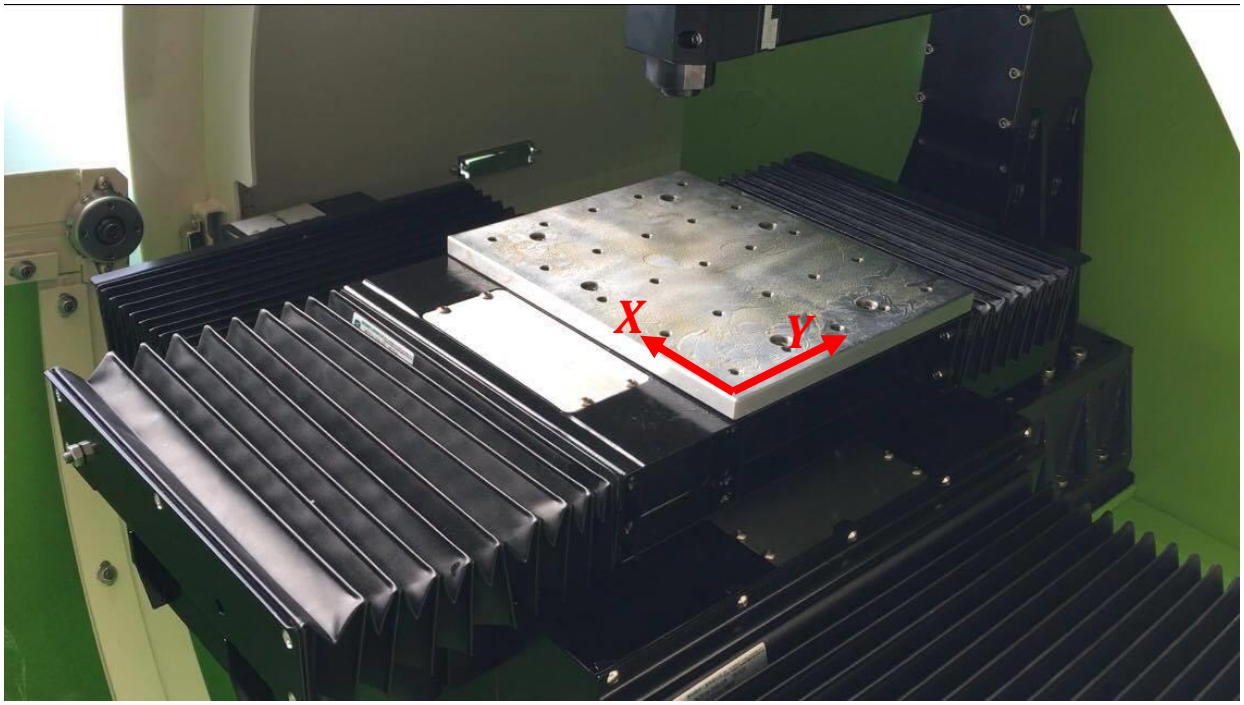

Figure 4: Ball Screw Driven XY Table

Part D – Pole Placement Controller Design (Mandatory only for MECH 541 Students)

Design a pole placement controller for this table. Using the equations given in the lecture notes, provide design calculations for pole placement controllers with a bandwidth of 20Hz and a damping ratio of 0.7 for X and Y axes. Compare simulated bandwidth, rise time, and overshoot against the Case 1 lead-lag controllers designed in part B. For the line circle toolpath, compare the tracking errors of the pole placement controller against the LBW controller (case 1).

Parts E and F can be completed before the lab, but they are not a part of the prelab and are not mandatory for the lab session.

Lab Report - Analysis

For the line-circle trajectory (Figure 1), carry out the following analyses. Merge figures where possible, and keep corresponding plots together. Do not provide plots of the entire toolpath or axis commands since no useful details can be observed. Include zoomed-in sections, as requested in the following questions. You can write the names and draw arrows to mark different overlaid graphs.

Part E – Effect of Bandwidth

E.1. Plot the tracking errors in the x and y axes ($e_x = x_r - x, e_y = y_r - y$) as functions of time for the LBW and HBW controllers. Comment on the effect of bandwidth on tracking error.

E.2. Plot the simulated tool motion (x vs y) for the three cases of control system (part B.1) on top of the reference toolpath (all in one plot). Zoom into the critical regions R1, R2, R3, and R4 shown with circles in Figure 1. Approximately measure the largest value of the predicted contour error, and comment on the effect of bandwidth on contouring accuracy. Note that contouring error is defined for the toolpath (and not axes) and determined based on only the geometry (maximum deviation from the desired path). Unlike tracking error, contouring error is independent of time.
Note: You can use a 2×2 subplot corresponding to R1 to R4. Plot all the required graphs in all subplots. In each one, either manually zoom into the requested region or use the command `axis([])` in your script to directly adjust the plot axis limits.

E.3. For the midpoint of the first line (region R1) and Case 1 controllers, find the values of the tracking errors in the x and y axes from your simulation, and calculate the contouring error **analytically** using these tracking errors. Is it always possible to calculate the contouring error analytically?

E.4. Discussion: In Case 3, the cross-over frequencies of the X and Y axes are assigned differently. Hence, their bandwidths are different, and the two axes no longer have identical position loop dynamics. Considering the simulated toolpaths in part E.2, comment on the contouring performance in the case of ***mismatched*** axis dynamics. Is it desirable to have mismatched dynamics? Why?

Part F – Effect of Maximum Feedrate

F.1. Regenerate the reference trajectory using a desired feedrate of $F = 250$ [mm/sec]. Plot the new displacement (s), feedrate ($\dot{s}$), and tangential acceleration ($\ddot{s}$) profiles, as well as axis position (x_r, y_r), velocity ($\dot{x}_r$, $\dot{y}_r$), and acceleration ($\ddot{x}_r$, $\ddot{y}_r$) commands, all as functions of time (t).

F.2. Simulate the two-axis response using the high feedrate trajectories and LBW controllers (Case 1). Plot the tracking errors vs. time (in both x and y axes) on top of the tracking errors in the case of low feedrate trajectories (part E.1). Comment on the effect of maximum feedrate on tracking errors.

F.3. Overlay the simulated toolpath for the low and high feedrates, zoom into the regions R1 to R4, and comment on the effect of the maximum feedrate on contouring accuracy.

Part G – Experiment vs. Simulation

G.1. For the line circle trajectory and using the measured axis responses from the experiment, plot the experimental tracking errors on top of your simulated ones (using LBW controllers). Also, overlay the simulated and experimental toolpaths, and zoom into the regions R1 to R4. Comment on the reasons for the discrepancy between simulation and experiment.

G.2. For your custom trajectory (part C.2), overlay the simulated and measured toolpaths, and zoom into a few critical points such as corners and sharply curved sections.

Part H – Pole Placement Controller Evaluation (Mandatory only for MECH 541 Students)

Implement the pole placement on the 2-axis machine, run the line circle trajectory, and compare the experimental to simulated responses for your designed pole placement controller.

Report Requirements:

The report must be typed and orderly. Poorly presented reports will not be accepted. Equations may be handwritten into the report, but they must be written very neatly and numbered. Please be as concise as possible. Do not provide unnecessary information. Credit will be given for concise comments that demonstrate a good understanding of the material covered in the lab.
The format of the report is.

1. **Title Page:** Your name, student ID, email, etc.
2. **Abstract*:*** 3-4 lines describing the objective of the laboratory as you understand it.
3. **Introduction:** Describe the role of multi-axis trajectory design in industry and the layout of your report (maximum 5 to 6 lines).
4. **Body:** In this part, you need to present the results and analyses clearly but concisely.
5. **Conclusion:** Explain what you have understood from the lab (5-8 lines).
6. **Appendix:** MATLAB scripts and Simulink block diagram.