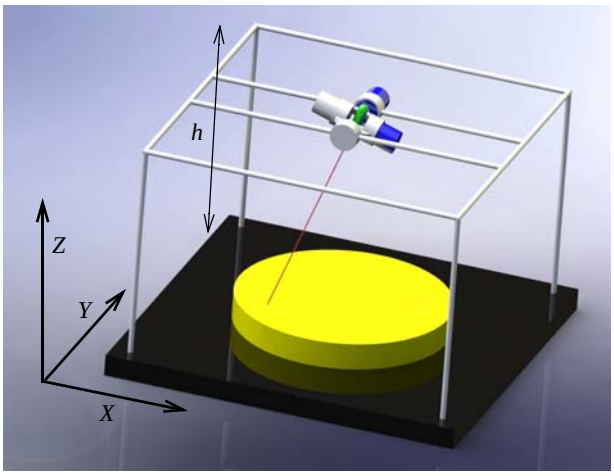
ELEC 341 Project 2017 - Selective Laser Sintering 3D Printer

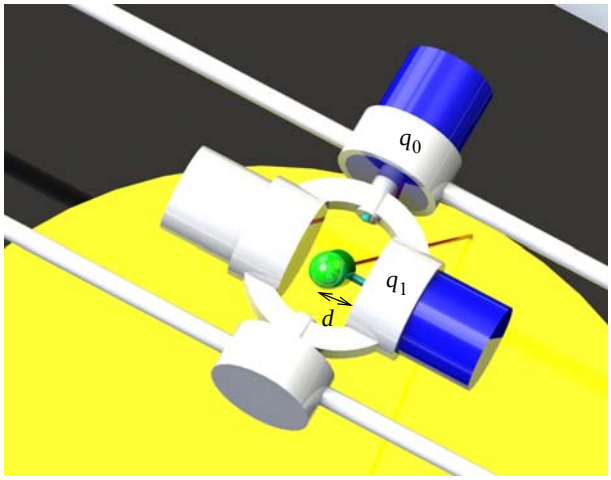
The University of British Columbia

In selective laser sintering (SLS), 3D parts are built by spreading a thin layer of metallic powder over the build plate and welding it into place with a high powered laser. As each layer is formed, the build surface grows upward until the part is complete.



In this SLS system, the laser is guided by a 2-DOF spherical wrist that orients the laser without moving its tip (origin). The q_0 axis orients the laser about the y-axis which sweeps the laser along the x-axis, while the q_1 axis orients the laser about the x-axis which sweeps the laser along the y-axis.

This project has two parts. In PART I, you will model the system. In PART II, you will design a PID controller to minimize the build error of a solid cylindrical part.



In PART I:

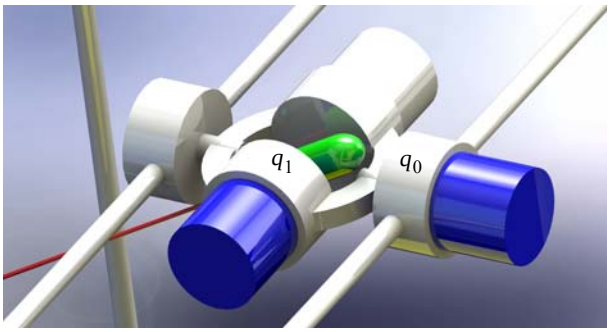
- Develop models to compute the direct and inverse kinematics of the wrist.
- Model the dynamics of the robotic wrist, including all electronics, motors and mechanisms.

In PART II:

- Select the best motor for each joint.
- Design a PID controller that minimizes the build error without burning either motor.
- Modify the time vector to help achieve the desired result.

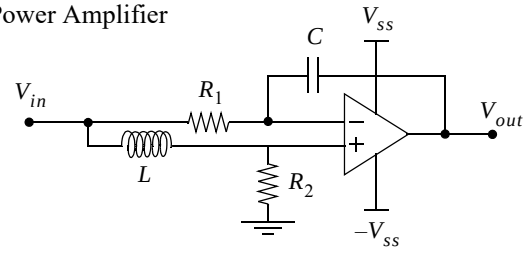
System Description

The q_0 joint is a mechanically commutating DC motor which is connected to the frame of the printer. The q_1 joint is a mechanically commutating DC motor which is connected to a circular arm that is actuated by q_0 on one side and supported by a bearing on the other side. The circular arm has a counterweight that is effectively equal to q_1 to cancel any gravitational effects. The laser is actuated by q_1. Note that the tip of the laser never moves. Only its orientation changes as q_0 and q_1 rotate. This is one benefit of using a spherical wrist configuration. Another benefit is its simple direct and inverse kinematic equations.



Both joints use an identical Power Amplifier circuit (shown) and are direct drive. In other words, they do not use any gearboxes.

Power Amplifier



The part is a solid cylinder with a 100mm radius and 15mm height. It is built in 5mm increments. Normally, SLS print resolution is much finer than this but it is increased in this simulation to reduce run-time.

The origin of the printer is the centre of the build plate and the laser is 200mm above the build plate. Therefore, the x, y and z coordinates of the part range between ±100mm and the laser is located at [0, 0, 200]. The z component corresponds to the height of the part, which increases as each layer is completed. The strength of the laser beam must be computed and corresponds to the distance between the laser and the point on the part that is being sintered.

How to Complete this Project:

Part I: Model the System

- Modify **printer.slx**
  - Design **Direct** & **Inverse Kinematics** (green) blocks
  - Initially, these blocks contain straight connections.
  - Replace these connections with Simulink blocks that calculate the joint values from the cartesian coordinates, and vise versa. Refer to the **Laser Control** block for an example of how the correct length of the laser beam is computed and vectors of Actual joint values (QaLa) and Actual cartesian coordinates (XYZa) are created.
  - When you have done the **Direct Kinematics** correctly, the laser will follow the correct path on the 3D graphic.
  - When you have done the **Inverse Kinematics** correctly, the **XY Desired** and **XY Actual** plots will match.
- Modify **System.m**
  - Use the constants in **CONSTANTS.m** to re-compute the variables set in **DEFAULT.m**. NO MAGIC NUMBERS.
  - Start by converting all values into a consistent set of physical units. For example, voltages should be specified in Volts, not mV. Consult the physical units specified in the **Jnt Controller** (yellow) block.
  - Compute the associated numerators, denominators and gains for each variable. This automatically overwrites the **Jnt Controller** block values when you run the simulation. Include detailed comments to explain your work. THESE COMMENTS ARE CONSIDERED PART OF YOUR REPORT.
  - DO NOT CHANGE THE MOTORS. You must use the default motors in Part 1.
- Check your work
  - Add scope blocks to probe internal signals or modify the Simulink models any way you please during debugging. The block values are all computed in m-files so you can replace the Simulink model with a fresh unaltered copy after you are done. Just save a copy of your **Direct** and **Inverse Kinematics** blocks so you don't lose your work.
  - For example, break the connections around a block, add a step to the input, a scope to the output, and run the model. This will show you the step response. Multiply the numerator of the block by **s** and run it again. This will show you the impulse response. Do these responses make sense? Do they seem to represent what you would expect the physical system to do? If not, check your work.
    - You know what a motor should do when you connect it to a power supply and turn it on. Does your system model behave the same way? How about when you attach a mechanism to it?
  - Evaluate the systems (transfer function, poles/zeros, root locus, etc.). Use Matlab functions to do this. You don't need to compute any of this by hand.
    - help control
    - tf / zpk / feedback / step / rlocus

Part II: Design a Controller

- Modify **System.m** to choose the motors
  - Compute the transfer functions
  - Find the poles & zeros
  - Draw the root locus
  - Use Matlab functions to do this. You don't need to compute any of this by hand.
- Modify **Control.m** to set PID controller gains for each robot axis.
  - Compute the open-loop gain of each joint.
  - Use the techniques you learned in class to choose values for your controllers.
  - The techniques you learned only apply to linear systems. Linearize the system before attempting to apply them.
  - Put back all non-linear elements and adjust your controller gains by hand to optimize their response.

    - Round all PID gain values to 3 SIGNIFICANT FIGURES.
    - You are welcome to use the PID "Tune" button in the PID controller box to produce automatically generated PID values. Compare these to your values.
- Modify **Control.m** to adjust the **Time** vector
    - The default **Time** vector has equal time increments between set points.
    - Modify the time between samples to minimize the "**Max Pos Error**". This vector has many elements and may be difficult to set by hand. Try to develop an analytic Matlab function to set it.
- Verify that you have met the criteria
    - If **Mean I** exceeds the **Maximum Continuous Current** for either motor, you have burned the motor. Note that you may exceed the average during the simulation as long as you have not exceed the average at the end of it (cooling).
    - If **Max I** exceeds the **Stall Current** at any time for either motor, you have burned the motor.
    - If the lamp is green at the end of the simulation, your motors are ok. If it is red, they are burned.
    - Any time your laser exceed the bounds of the part (overshoot), you accumulate build error.
    - Any time your laser is not at the set point when the set-point changes, you accumulate build error.

Hints

- Type "**printer**" from the Matlab window to run the Simulink model file.
- Matlab files are included which assign all motor data sheet values to a variable named "**MotorParam**" with indices that are assigned in **CONSTANTS.m**. See **DEFAULT.m** for an example of how to assign your motors.
- Try setting the **Time** vector so that the time between set-points is dependent on the distance between set-points.
- Although damping (friction) is not explicitly specified in the data sheets, you can figure it out from the other values. If you applied 1V to the motor terminals, how fast would it spin? If not for friction, it would spin infinitely fast!
- Closing the '3D Graphic' window will make the model run faster which is useful when optimizing (tuning) your system. Alternatively, select Simulation/Block Parameters/Sample Time and increase the value to re-draw less frequently.
- Connect a Scope block to any signal in the model to see what its value is when you run the model. Use a MUX to see more than 1 signal in the same scope window and turn on the Legend from the View menu to understand what each colour means. This is useful for general debugging.
- Double-click any of the blocks in the **Jnt Controller** subsystem to see how Matlab variables are used to set the parameters in a Simulink block.
- If you make a typo when setting a variable, Matlab will create a new variable with that name and will not change the variable that you think you changed. Matlab will not know that this is a mistake and will not complain so be careful! Bugs are not always easy to find.
- Any file with its name in CAPITAL LETTERS is one that you should not modify whatsoever.

Technical Specifications:

Amplifier

- Resistance R_1 = 2.2 MΩ
- Resistance R_2 = 5.9 Ω
- Capacitance C = 16 uF
- Capacitance L = 120 mH
- The Source Voltage (Vss) is set such that the maximum motor voltage is not exceeded.

Motor Options

- Maxon A-max 22, Precious Metal Brushes CLL, 5W - Sleeve Bearings (default)
- Maxon A-max 22, Graphite Brushes, 6W - Sleeve Bearings
- You may spec any motor for either joint. Data sheets are provided.

Each motor uses an identical, single-turn, analog position sensor. The sensor is linear and outputs its minimum output voltage at its minimum angular limit, and its maximum output voltage at its maximum angular limit.

Sensor

- Output Voltage = ±5V
- Angular Range = ±180°

The first joint carries the weight of a circular link made from 6061 Aluminum as well as the second motor q_1 and a counter-weight that weighs the same as q_1 but is located on the opposite side of the link to counteract gravity. Estimate the inertia of the arm by a cylindrical tube with the dimensions indicated below. The joint (bearing) opposite q_0 has the same dynamic friction as q_0 as well as a rotary spring with spring constant **K**. The entire joint has the static friction coefficient **uSF** (see comment in **CONSTANTS.m** for an explanation of **uSF** and open the **Static Friction** subsystem block to see how it is implemented in Simulink).

Joint q_0

- Height (h) above build plate = 200mm
- Link Inner Radius = 16 mm
- Link Outer Radius = 22mm
- Link Depth = 5mm
- Density of 6061 Al = 2.7 g/cc
- Motion range = unlimited
- Spring Constant K = 7mNm/rev
- uSF = 700 um

The second joint only carries the weight of the laser which is negligible. The face of the motor is located a distance **d** from the centre of the laser (as shown above). The second joint does not have a centering spring but due to mechanical interference, it has a limited motion range.

Joint q_1

- Link Offset = 9mm (distance d from centre to face of q_1)
- Motion Range q_2 = -60° to +60°

General Information:

- The model was created using Matlab Version **R2017a**.
- You may use any Matlab functions to complete this project. Include all Matlab scripts and output in your report. The control toolbox (help control) is useful for solving transfer functions, drawing a root locus or nyquist plot, etc.
- You are provided with the following files:
  - **printer.slx** is the Simulink model file.
  - **3dPrinter.WRL** is the 3D graphics file.
  - **CONSTANTS.m** in an m-file containing all the numeric specifications and indices in the Maxon motor files.
  - **DEFAULT.m** is an m-file containing default values for all system models.
  - **TRAJECTORY.m** is an m-file that assigns the desired laser position trajectory and set-point time vector.
  - **System.m** is an m-file that you modify to over-write the default values.
  - **Control.m** is an m-file that you modify to overwrite the PID controller gains and time vector.
  - Each of the above Matlab functions are automatically re-run each time you press “play”.
  - A variety of maxon data-sheet m-files (eg. **AMAX22_5W_SB.m**) assign all associated motor parameters that are indicated in the data sheet to the MotorParam vector.

Part 1: Report

Email printer.slx & System.m to the Project TA (DO NOT CC instructor)

- All calculations should have detailed comments describing how they are derived.

Include a PDF of PowerPoint slide deck

- Self-explanatory but simple set of slides that explains all of your work
- Maximum of 10 slides + Title slide with group name, student names & student numbers
- These slides are what you would use if you were going to give a presentation to describe your work
  - No eye charts (microscopic printing)
  - Simple math only
  - Lots of figures with annotations
    - Printout of Direct & Inverse Kinematics blocks + equivalent equation
    - Samples of sub-system step/impulse responses

Part 2: Report

Email Control.m to the Project TA (DO NOT CC instructor)

- Include detailed comments describing how the Time vector is derived.

Include a PDF of PowerPoint slide deck

- Same specifications as above
- Things to include:
  - Strategy & Starting Point
  - Tuning Process - show a few intermediate results
  - Printout of any Root Loci or other evaluations that you did
  - Comparison of your PID gain values to Tune Button PID values

Best Part Prize

Your system must meet the following criteria:

- Simulation / Configuration Parameters / Solver = **ode45 (Dormand-Prince).**
- PID blocks: Derivative divisor (N) = **100.** Refer to the project web page for a description of this parameter.
- PID values may not contain more than 3 significant figures (trailing zeros are ok).
- No burned motors.
- Minimum “Max Pos Error”.

Grading:

- This project is worth 30% of your final grade. 15% for Part 1 and 15% for Part 2.
- All printed reports AND emails must be received by the due date or the report will be considered late.
- LATE reports will be marked PASS/FAIL (PASS = 50%, FAIL = 0%).
- If you do not get a passing grade on this project, ALL OF YOUR QUIZ MARKS WILL COUNT.
- You may work in teams of 1 or 2. Submit 1 report per team. Both team members receive the same grade.
- The team that wins the **Best Part Prize** will receive FULL MARKS (15%) for Part 2.

15% Part 1:

- 25% Direct & Inverse Kinematics
- 50% System Model Values (including physical units)
- 25% Explanation (slide deck)

15% Part 2:

- 50% Starting Point & Tuning Process
- 25% System Performance
- 25% Explanation (slide deck)