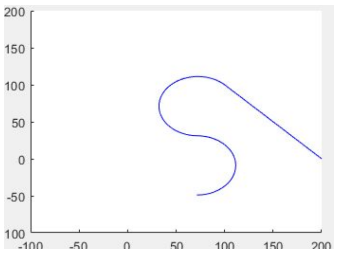
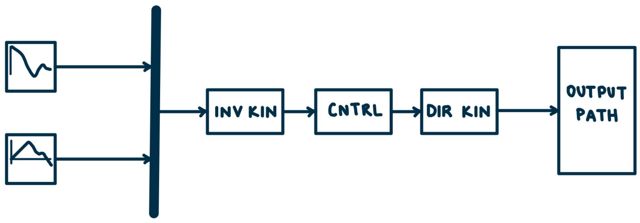
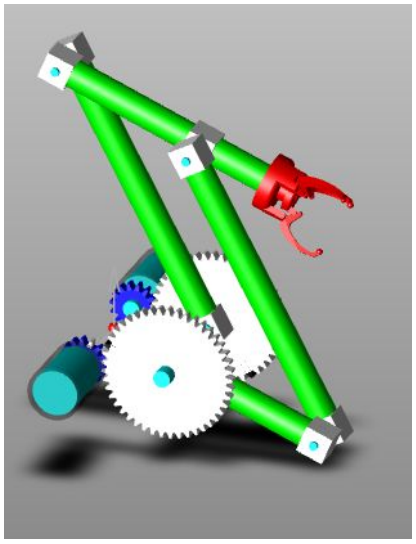
System Overview

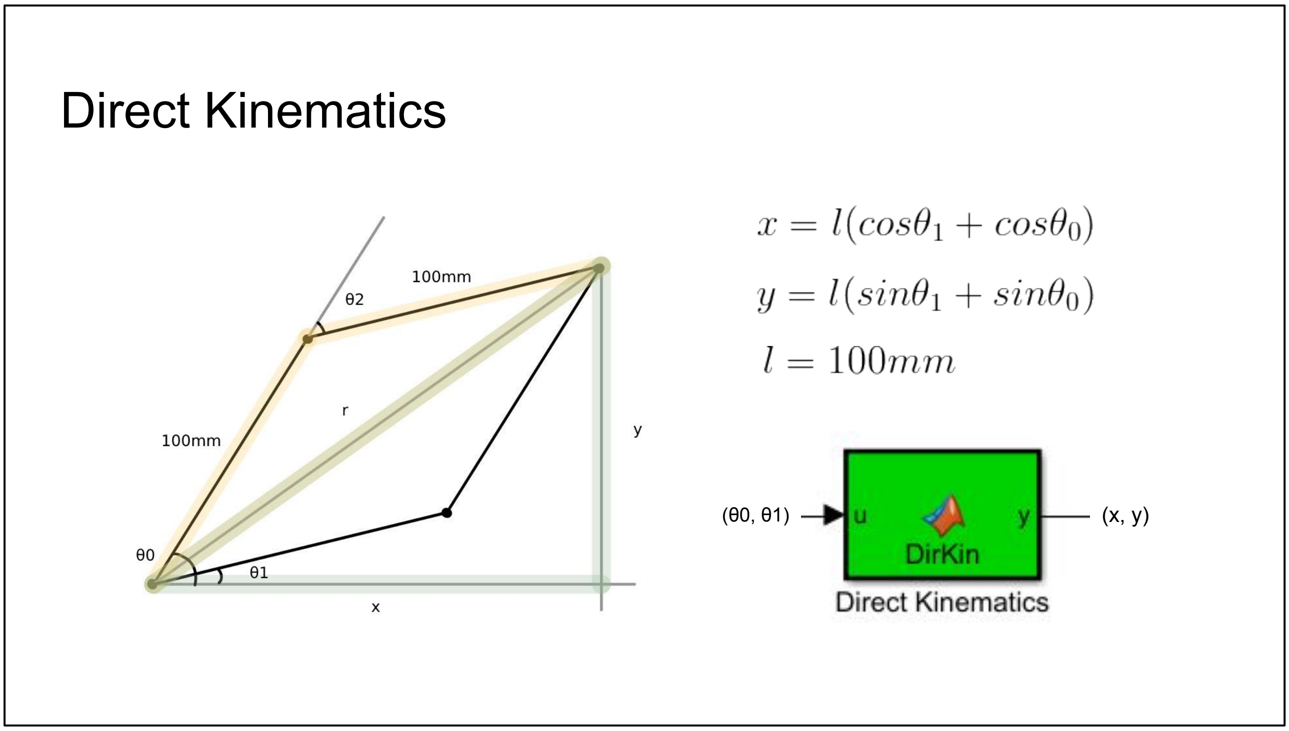


Input path of arm

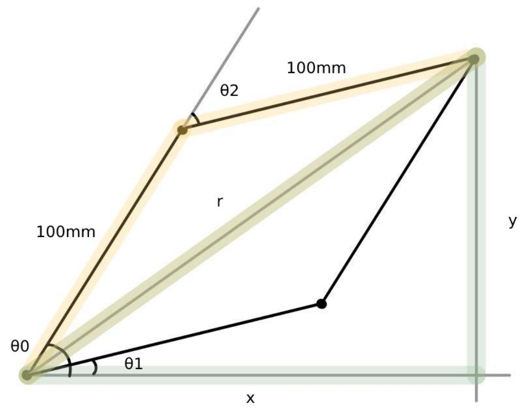




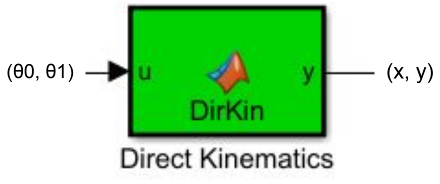
From the block diagram on the left, a vector of x and y coordinates are inputted to the system (the desired position of the end effector of the arm). The coordinates first get passed through an inverse kinematics block which changes the x and y parameters to joint space angles of the two motors (q0, q1) of the robot. The desired angles are then sent into the controller, which from part 1, computes how the robot needs to correct itself (with the use of a PID controller) due to the error from the actual position and desired angle position. Finally, the actual angles of the motors are sent to the direct kinematics block which is responsible for converting them back into the actual x and y coordinates of the end effector and plotted to see the path it takes.



Direct Kinematics



$$x = l(cos\theta_1 + cos\theta_0)$$
$$y = l(sin\theta_1 + sin\theta_0)$$
$$l = 100mm$$



Direct kinematics outputs a position in the (x, y) coordinate for the End Effector from a given the motor angles: u0, u1 as its parameters.
Since there is a 1:3 gear ratio, we must scale our parameters (u0,u1) by -⅓ as we are going from angle of the motor/pinion to the bullgear. (θ0 = u0/-3, θ1 = u1/-3)

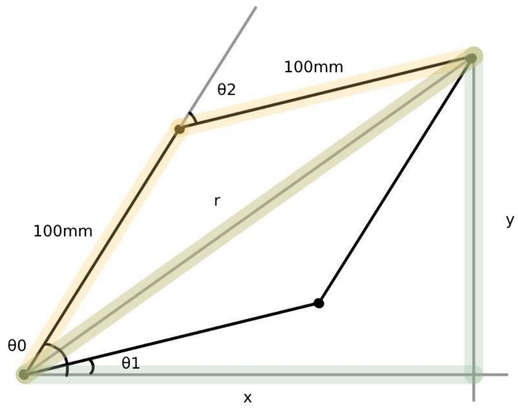
We calculated the x coordinate by obtaining the x components (l*cos(theta)) of each bar and adding them together and did the same for the y coordinate and used l*sin(theta), this is similar to vector addition between the arms to get our output end point/ position.

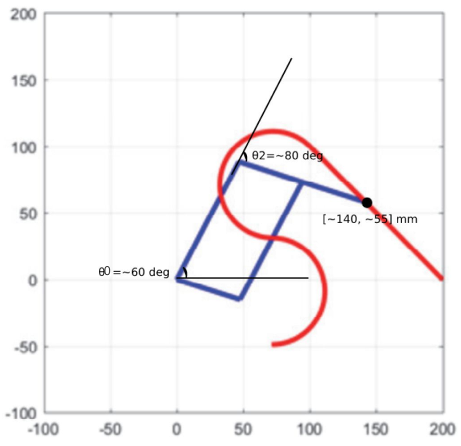
Inverse Kinematics

$$\theta_0 = cos^{-1}(5\sqrt{x^2+y^2}) + tan^{-1}(\frac{y}{x})$$

$$\theta_2 = \pi - 2sin^{-1}(5\sqrt{x^2+y^2})$$

$$\theta_1 = \theta_0 - \theta_2$$





The purpose of the inverse kinematics block is to convert the desired (x, y) coordinates to the desired joint angles - it converts from cartesian coordinates to joint space angles for the robot.

To compute the joint angles θ0 and θ1, we approximated the four-bar linkage into two triangles (the highlighted yellow and green above) and saw that the yellow triangle can be assumed to be isosceles. From there we just used trigonometric identities and we derived the equations above. Due to the 1:3 gear ratio, we need to scale our outputs (θ0, θ1) by -3. The negative scaling is due to the direction of the gear turning with respect to the pinion.

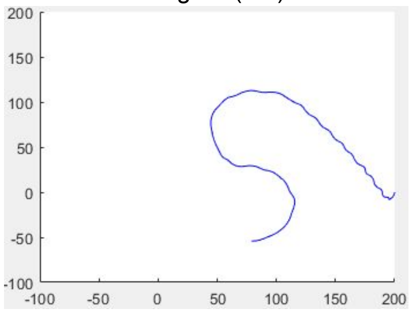
We know our equations are correct because we inputted some test values (taken from the project description) and got that the outputs were 62.7 degrees and 82.5 degrees, which based on the picture is correct.

PID Tuning

	Rise Time	Overshoot	Settle Time	Steady State Error
K_P	↓	↑	↑	↓
K_I	↓	↑	↑	Eliminate
K_D	↑	↓	↓	No effect

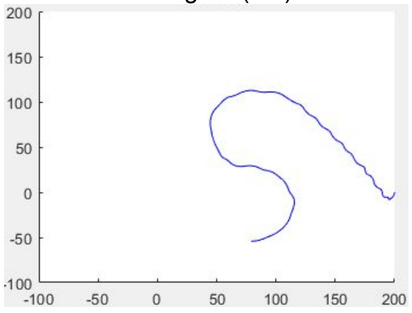
Position - XY Plot

Figure (1.1)



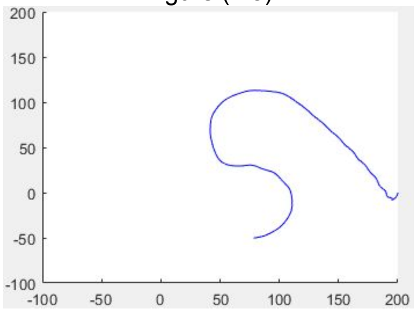
P: 0.085 **I:** 0.00065 **D:** 0.027

Figure (1.2)



P: 0.065 **I:** 0.00095 **D:** 0.0050

Figure (1.3)



P: 0.0502 **I:** 0.00115 **D:** 0.0105

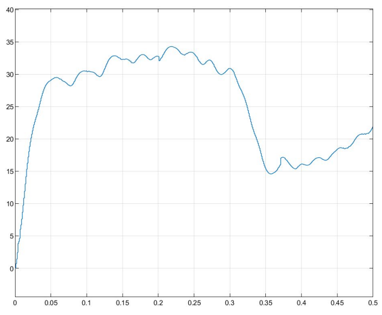
Regarding PID tuning, we started off with our initial values on the right. To get these first values we set the I and D gains to a relatively low number to effectively get rid of their influence on the response. From there the P gain was increased until the system slightly oscillated and the rest of the gains were determined by using intuition from understanding what each of them do, with the help of the above chart as high level reference.

However as it can be seen, the arm's response was still not ideal so we fine tuned the rest. Due to the oscillations, we decreased the P gain (as can be seen in the centre image). This response also showed overshoot and a subtle amount of oscillations at the start of the path.

We reached our final PID gains by further decreasing the P and increasing the D to smooth out the curve. Along with this to get rid of the sharp turn in the middle of the path, when the arm changes direction, the I gain was increased as well. The reason why the arm dips at the start is due to gravity and the system starting, with all the tuning we did we could not fully get rid of the initial dip error.

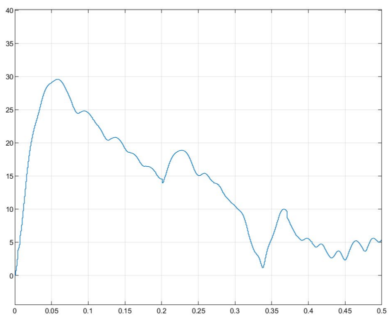
System Performance: Path Error Plot

Figure (2.1)



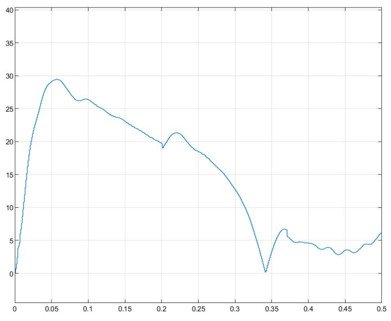
P: 0.085 **I:** 0.00065 **D:** 0.027

Figure (2.2)



P: 0.065 **I:** 0.00095 **D:** 0.0190

Figure (2.3)



P: 0.0502 **I:** 0.00115 **D:** 0.0105

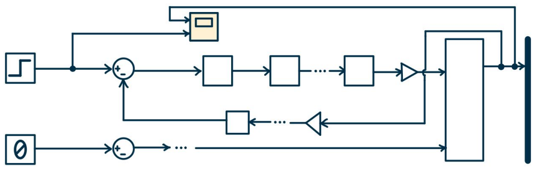
Here are the following images of the path error correlating with PID tuning position plots in the previous slide. The goal of the path error plot is to minimize our path error, this can also be seen by having our error (y axis) be as close to 0 as possible. All 3 figures have large initial error due to the initial dip from gravity.

As seen, our first sample of PID tuning (figure 2.1), we had a fairly poor path error plot. The error average of the plot was very high, therefore, the the PID's chosen for this sample is not ideal.

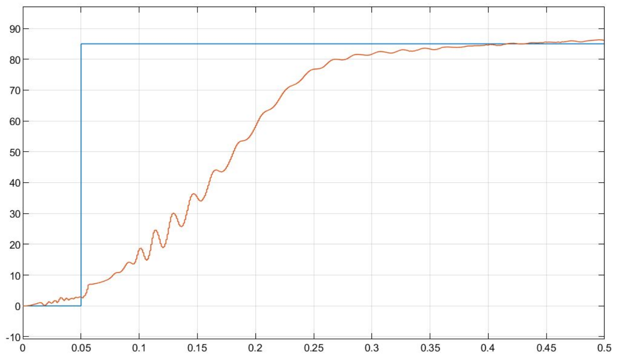
In our second sample of PID tuning (figure 2.2), the chosen PID's allowed our path error avg to be significantly smaller than our previous sample. Comparing to our position plots from our previous slide, there was a lot of "shakiness" in the plot, this translated to a shaky path error plot, meaning we could improve on the P and D tuning to account for "shakiness" (more detailed explanation on page 3).

Our final chosen PID gains (figure 2.3) shows a steadily decreasing path error plot with some increase in error at the end due to the sharp turn. The max error of system was 29.43 mm.

System Performance: Step Response



Modified system to determine step response



Step response

To evaluate the joint space system, the system model was modified. We took out the input MUX and added a step input to one of the motors (q0 in this case) and set an input of 0 to the other motor so it would not interfere. We connected the input step and the output (after the SimulationX coupling) to a scope and the step response on the right was achieved. The input angle was set to 85 degrees as this is a reasonable angle that the motor had to achieved based on the given description (as seen on page 3).

From analyzing this step response we see that the joint space portion of the system has a rise time of roughly 0.215s and has negligible steady state error due to correctly tuning the I gain in the controller. We could decrease the rise time by increasing the P gain however then the system would overshoot and be less stable as it settles to steady state.

Project By:

- Mattias Zurković 75106880
- Karmen Wang 54144183