Outline

This paper aims to outline the math behind a state space control for a 4 jointed arm (although the CAS code for generating the final equation is generalized to N). Within this, there are 3 subgoals: Find the Forward Kinematics of the arm (given the angles of the arm, what is the location of each joint?), find the Inverse Kinematics of the arm (given the desired location of the grabber, what should the angles of the arm be?), and Forward Dynamics (what is the angular acceleration α of each joint?).

Forward Kinematics

Figure 1 shows the name and location of each arm joint in 3D space.

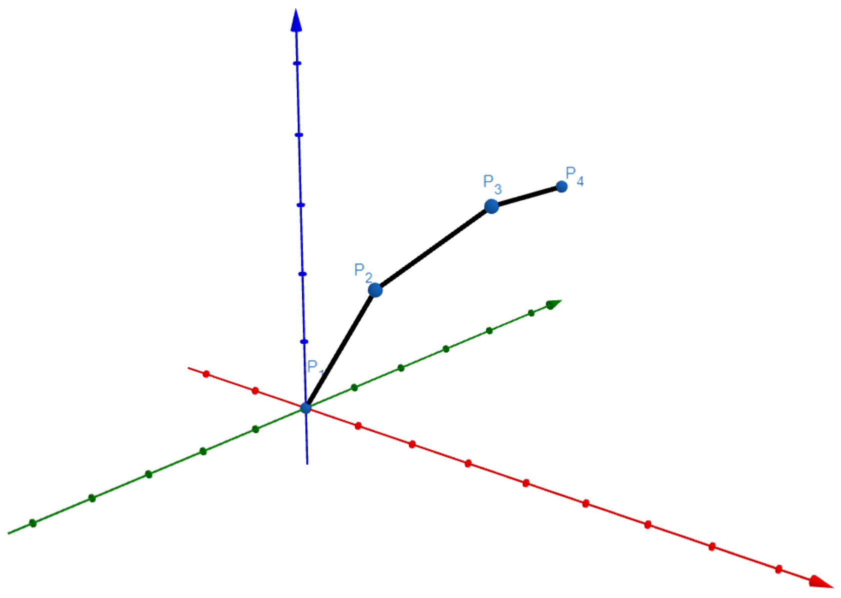

Figure 1: The Arm Joints Locations in 3D

To simplify how to think about this arm, we can split the 3 joints that all rotate in the same plane from the joint that rotates around the Z/vertical axis. So we can do most of our work in the 2D plane, which is where most of the complexity is, then transform back into the 3D plane. This ends up removing a lot of repetition and redundancy within work.

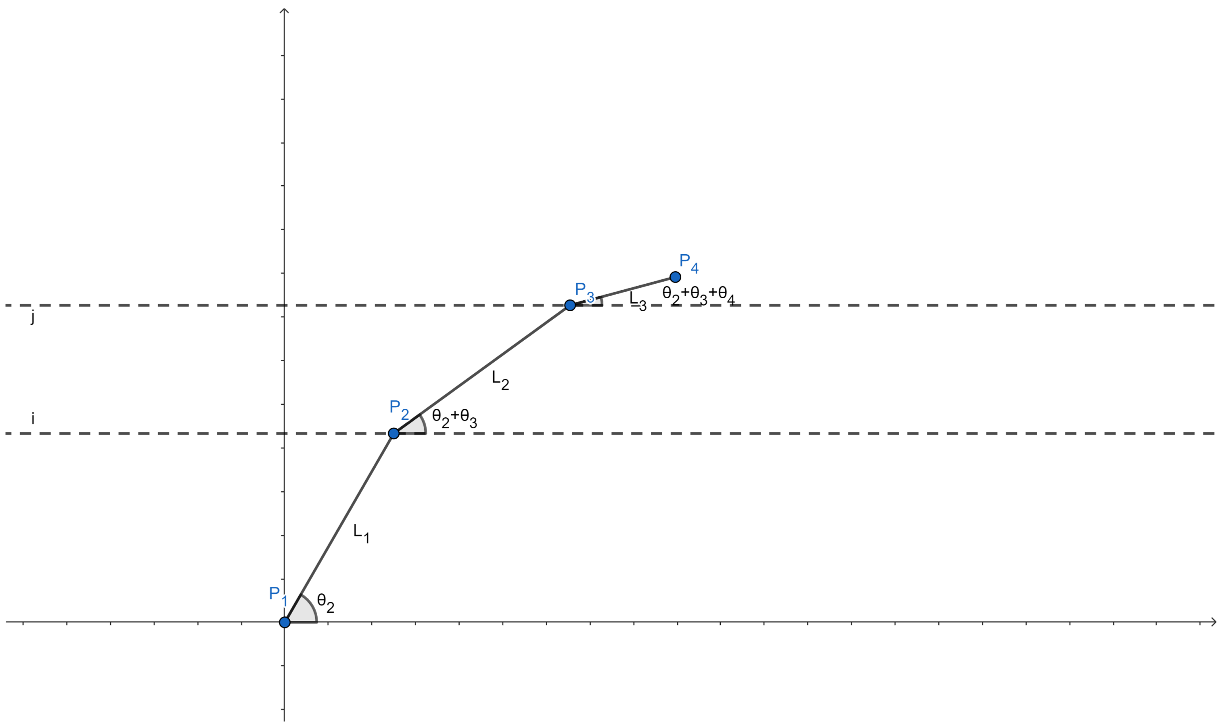

Figure 2: 2D plane of arm joints

From Figure 2, we can see that in this plane:

$$P_1 = (0, 0)$$

$$P_2 = P_1 + (L_1 \sin(\theta_2), L_1 \cos(\theta_2))$$

$$P_3 = P_2 + (L_2 \sin(\theta_2 + \theta_3), L_2 \cos(\theta_2 + \theta_3))$$

$$P_4 = P_3 + (L_3 \sin(\theta_2 + \theta_3 + \theta_4), L_3 \cos(\theta_2 + \theta_3 + \theta_4))$$

This expands to:

$$P_1 = (0, 0)$$

$$P_2 = (L_1 \sin(\theta_2), L_1 \cos(\theta_2))$$

$$P_3 = (L_1 \sin(\theta_2) + L_2 \sin(\theta_2 + \theta_3), L_1 \cos(\theta_2) + L_2 \cos(\theta_2 + \theta_3))$$

$$P_4 = (L_1 \sin(\theta_2) + L_2 \sin(\theta_2 + \theta_3) + L_3 \sin(\theta_2 + \theta_3 + \theta_4), L_1 \cos(\theta_2) + L_2 \cos(\theta_2 + \theta_3) + L_3 \cos(\theta_2 + \theta_3 + \theta_4)).$$

Then to convert the 2D coordinates in the joint plane to 3D real coordinates given the rotation of the rotating base (θ_1), we take

$$f(P) = \left(P_x \sin(\theta_1), P_x \cos(\theta_2), P_y\right).$$

So plugging in our 2D coordinates to the 3D transformation gives us the final 3D coordinates.

Inverse Kinematics

First, we start with a known and proven equation: the equation for the geometrical IK for a 2 jointed arm. The proof is just a lot of geometry and a lot of fiddling with trig identities. The resulting equations are

$$q_2 = -\arccos\left(\frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1 L_2}\right)$$

$$q_1 = \arctan\left(\frac{y}{x}\right) - \arctan\left(\frac{L_2 \sin(q_2)}{L_1 + L_2 \cos(q_3)}\right)$$

for Figure 3:

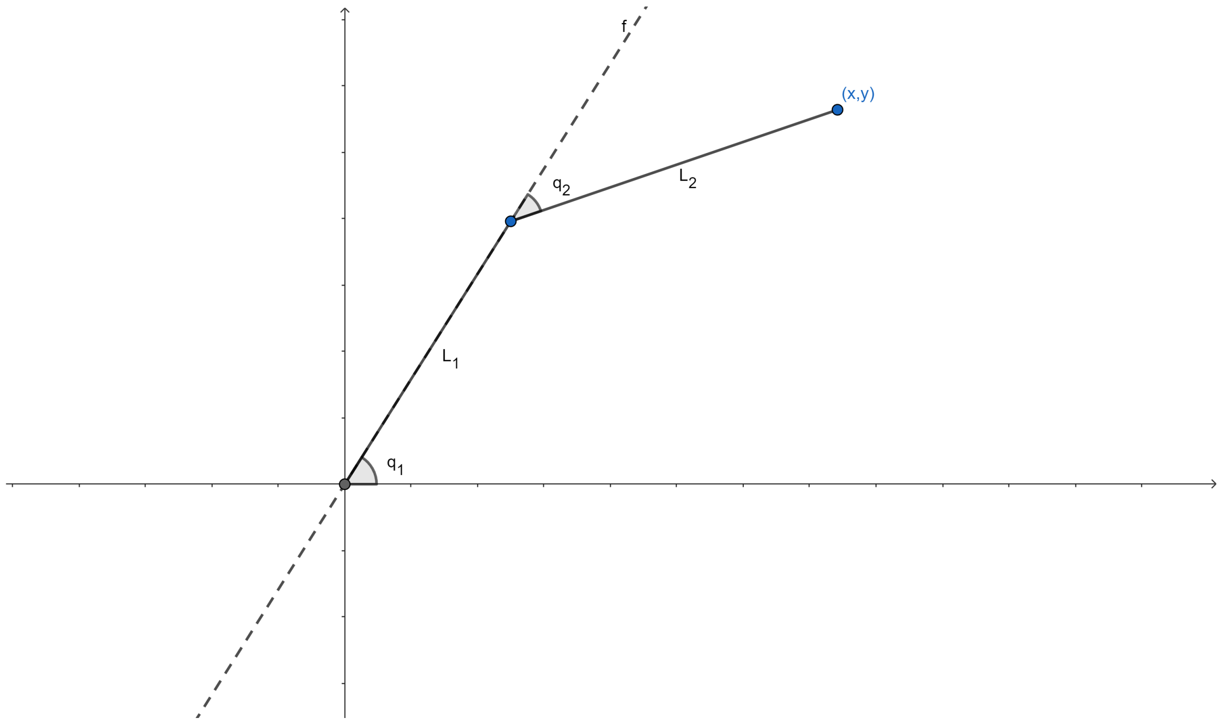

Figure 3: a 2 jointed arm

So to apply this to our 4 axis, we assume we want the manipulator flat. So that means, looking at the 2D plane of the joints, getting the end of the manipulator to point (x, y) means getting the start of the manipulator to point $(x - L_3, y)$. Therefore, looking at the plane of the joints, we run a 2D IK to calculate what θ_2 and θ_3 are necessary to get the start of the manipulator to $(x - L_3, y)$. Then we find θ_4 such that the manipulator is flat, and θ_1 such that the joint plane intersects the desired point. Therefore, the final equations become:

$$\theta_1 = \arctan\left(\frac{x}{y}\right)$$

$$\theta_3 = -\arccos\left(\frac{x^2 + y^2 + z^2 - L_1^2 - L_2^2}{2L_1 L_2}\right)$$

$$\theta_2 = \arctan\left(\frac{z}{\sqrt{x^2 + y^2}}\right) - \arctan\left(\frac{L_2 \sin(q_2)}{L_1 + L_2 \cos(q_3)}\right)$$

$$\theta_4 = -\theta_2 - \theta_3$$

Inertia of a Line Segment

We want to find the inertia for any line segment around a point of rotation R. So the general form will be that the line segment goes from $P_A = (x_1, y_1)$ to $P_B = (x_2, y_2)$, as shown in Figure 4:

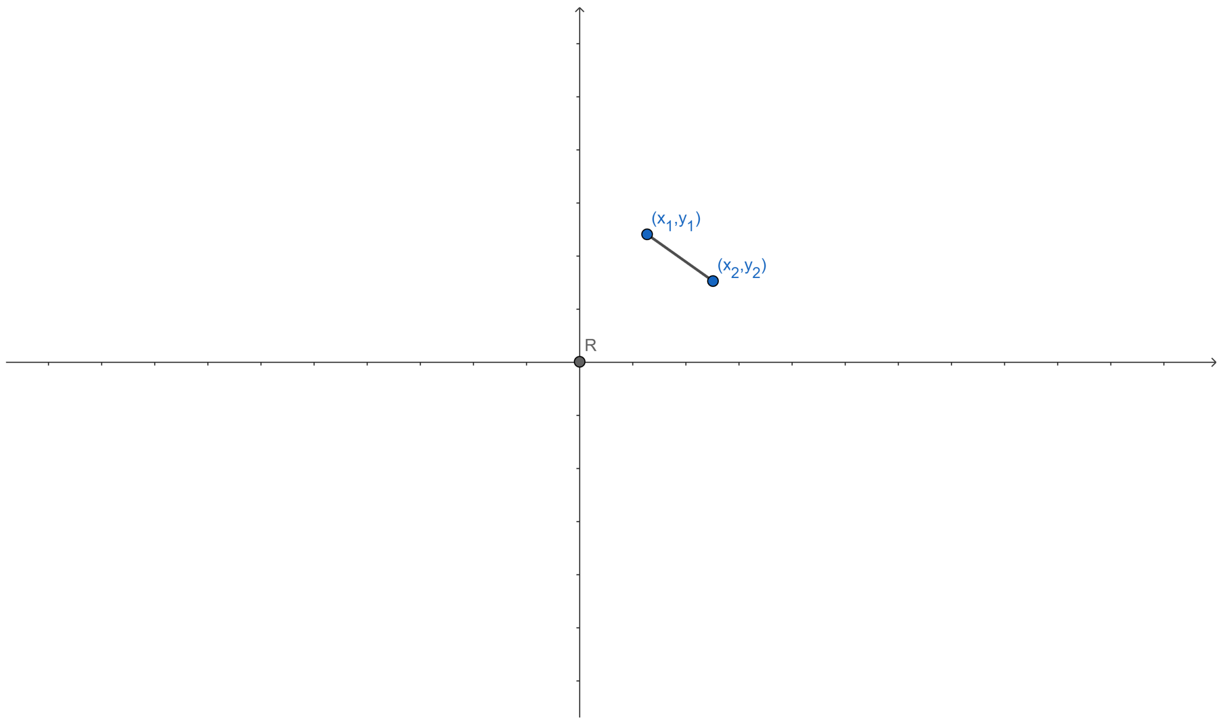

Figure 4: A Line Segment Rotating Around R

We can parametrize this segment over $t = [0, 1]$ with

$$f(t) = \left(x_1 + (x_2 - x_1)t, y_1 + \left(y_2 - y_1\right)t\right).$$

We want to find

$$I(P_A, P_B, m) = \int r^2 dm = m \int_0^1 r^2 dt = m \int_0^1 x^2 + y^2 dt = m \int_0^1 (x_1 + (x_2 - x_1)t)^2 + \left(y_1 + \left(y_2 - y_1\right)t\right)^2 dt$$

$$= \frac{m}{3}\left(x_1^2 + x_1 x_2 + x_2^2 + y_1^2 + y_1 y_2 + y_2^2\right).$$

Potential Energy of The Arm

Potential Energy of Point Masses

The potential as a result of the point masses is

$$\sum mgh = g\left(m_1\left(P_{1_y} - P_{1_y}\right) + m_2\left(P_{2_y} - P_{1_y}\right) + m_3\left(P_{3_y} - P_{1_y}\right) + m_4\left(P_{4_y} - P_{1_y}\right)\right)$$

$$= g\left(m_2 P_{2_y} + m_3 P_{3_y} + m_4 P_{4_y}\right)$$

Potential Energy of Line Segments

Since the mass distribution of a line segment is constant in the vertical frame, and the gravitational potential energy is linear, the gravitational potential energy will be the average of the lower and upper heights of the line segments. Therefore, the potential energy is

$$\sum Mg\frac{h_1 + h_2}{2} = g\left(M_1\frac{P_{1_y} + P_{2_y}}{2} + M_2\frac{P_{2_y} + P_{3_y}}{2} + M_3\frac{P_{3_y} + P_{4_y}}{2}\right)$$

Kinetic Energy of Arm

To calculate the kinetic energy of the arm, we first want to calculate the rotational inertia of the arm around each joint.

J_3

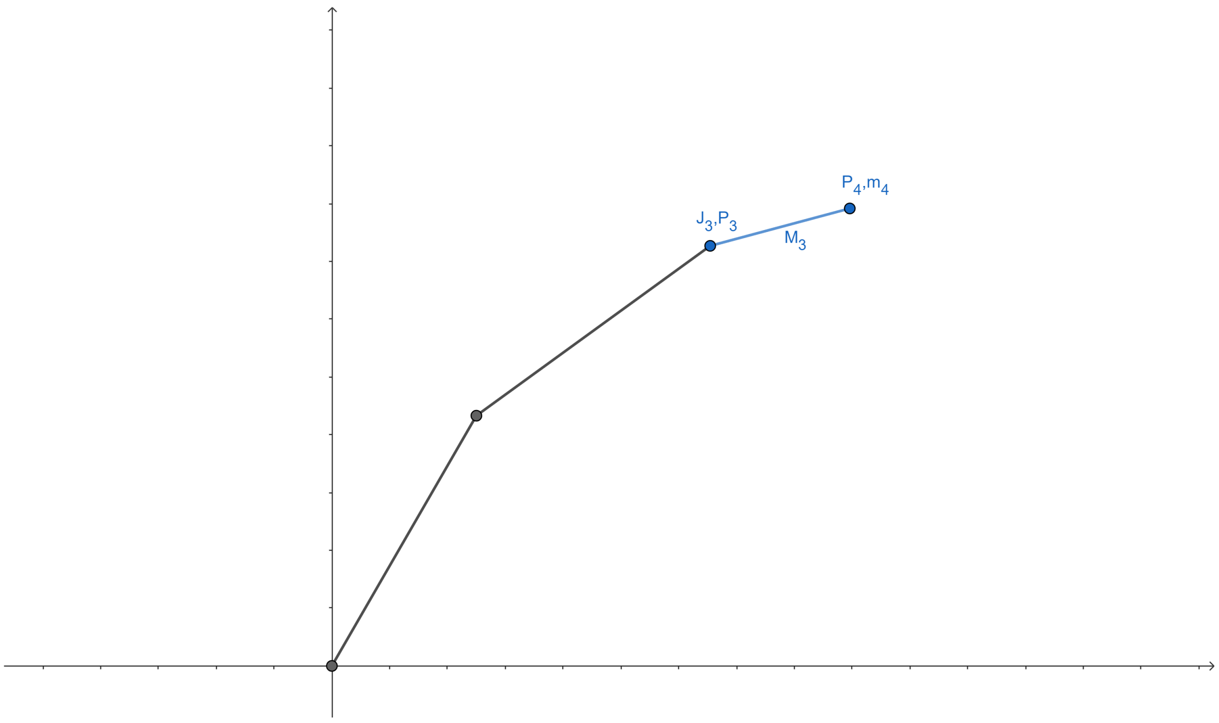

Figure 5: The Arm Focused on J3

Figure 5 shows the useful values for calculating the inertia around J_3. For each of the linear masses, we plug it into our inertia formula from the inertia of a line segment section, and for each point mass, we can just use $I(P, m) = mr^2 = m(x^2 + y^2)$. In this case, we have a total of 1 line segment with inertia $I(P_3 - P_3, P_4 - P_3, M_3)$ and 1 point mass with inertia $I(P_4 - P_3, m_4)$. This expands to total inertia of $m_4 L_3^2 + \frac{1}{3}M_3 L_3^2$. Now, to find the kinetic energy, we plug this into the rotational kinetic energy equation, which is $KE = \frac{1}{2}I\omega^2 = \frac{1}{2}\left(m_4 L_3^2 + \frac{1}{3}M_3 L_3^2\right){\theta_4}'^2$.

J_2

Similarly, for J_2, the inertia will be $I(P_2 - P_2, P_3 - P_2, M_2) + I(P_3 - P_2, P_4 - P_2, M_3)$ for the line segments and $I(P_3 - P_2, m_3) + I(P_4 - P_2, m_4)$ for the point masses. This means the total inertia will be $I(P_2 - P_2, P_3 - P_2, M_2) + I(P_3 - P_2, P_4 - P_2, M_3) + I(P_3 - P_2, m_3) + I(P_4 - P_2, m_4)$ and the kinetic energy will be
$\frac{1}{2}(I(P_2 - P_2, P_3 - P_2, M_2) + I(P_3 - P_2, P_4 - P_2, M_3) + I(P_3 - P_2, m_3) + I(P_4 - P_2, m_4)){\theta_3}'^2$.

J_1

Finally, for J_1 the inertia will be

$$I(P_1 - P_1, P_2 - P_1, M_1) + I(P_2 - P_1, P_3 - P_1, M_2) + I(P_3 - P_1, P_4 - P_1, M_3) +$$
$$I(P_2 - P_1, m_2) + I(P_3 - P_1, m_3) + I(P_4 - P_1, m_4).$$

So the kinetic energy will be

$$\frac{1}{2}(I(P_1 - P_1, P_2 - P_1, M_1) + I(P_2 - P_1, P_3 - P_1, M_2) + I(P_3 - P_1, P_4 - P_1, M_3) +$$
$$I(P_2 - P_1, m_2) + I(P_3 - P_1, m_3) + I(P_4 - P_1, m_4)){\theta_2}'^2.$$

Euler Lagrange Equation

We can calculate the lagrangian using the potential and kinetic energies we just found. The lagrangian is $KE - PE$. To get $\alpha = \theta''$, we need to solve the Euler-Lagrangian equation: $\frac{\partial f}{\partial \theta} = \frac{d}{dt}\frac{\partial f}{\partial \theta'}$. So if we do this for each of the joints, we get a system of equations that we can solve. By solving this system of equations for $\theta'' = [{\theta_1}'', {\theta_2}'', {\theta_3}'', {\theta_4}'']$, we know the evolution of the system without motors. To account for the motor torque, we just need to add the term $\theta'' += \frac{\tau}{I}$. Finally, we have the full equation for θ'', which is now our simulation of the arm.

State Space

To make an LQR controller to find the optimal path in this simulation, we need to calculate the A and B matrices as input to the WPIlib LQR function. For this, we need to establish state and input vectors. Our state will be $x = [\theta, \theta']\}$, while our input will be $u = \tau$.
$A = \text{jacobian}\left(\frac{d}{dt}x, x\right) = \text{jacobian}([\theta', \theta'']], x)$, while $B = \text{jacobian}([\theta', \theta''], u)$. We can plug in our solution for θ'' as a function of θ, θ', and τ to get A and B in terms of x and u or θ, θ', and τ.

Fixing State Space

Now we finally have the state space calculated. We still run into an issue of circular reasoning: τ is an input into the LQR controller. However, the entire purpose of our state space model and the LQR controller is to calculate the optimal τ to get to the desired location. In an affine system, the τ would cancel out in the calculation of A and B, but I have shown in informal proof that this is not possible for our 4-joint arm. So, as a non-affine system, we are left with circular reasoning. A solution is to cache the τ from the last step of calling the LQR controller and use it as an approximate τ to input to the next LQR.

Precomputing Values To Improve Performance

One issue is that the final resulting equations for A and B are extremely slow. Even with some optimizations, approximations, as well as precomputing $\theta' = 0$ for our desired state, it still ends up as a 47kb C++ of almost pure trig functions. This, combined with the slow computation of the LQR controller, means it takes significant computational power. We can precompute the gain matrix for different states to reduce this computational power. The gain matrices lookup table size for an 8-dimensional space grows at n^8,, which is horribly bad. So that a future project may implement an algorithm, for example, looking at some of the mesh algorithms used in CAD programs for efficiently storing the pre-computed matrices.

Code Links

All of this was calculated on a CAS and tested in a GUI that I made in Julia. All of the related code can be found here:

https://github.com/AlistairKeiller/RobotIK