**Introduction**

*[Title drops and cool background visual of robot in action. I should start talking as soon as the video begins.]*

Hi! My name is Matthew Zhong, and this is a presentation for a C++ robotics simulator I wrote in the past five days.

Let’s first begin by discussing what my simulator can do. My simulator can analyze for any kinematic chain the:

*[Have each bullet on the list appear on the screen in a bulleted list fashion as I mention each point.]*

1. Forward kinematics,
2. Forward differential kinematics,
3. Inverse differential kinematics,
4. End-effector and joint reaction statics

Let’s take a look at each of these features, one by one.

**Forward Kinematics**

*[Title drops. Video is paused]*

We begin by analyzing the forward kinematics of a six-joint manipulator.

*[Title fades.]*

We can tweak each joint position to observe how the end effector changes.

*[Zoom into table that says “Th:, Alp:, A: D:”]*

To see how the DH parameters change, take a look at the table at the top.

*[Zoom out back to full screen]*

Here’s what happens when we tweak:

*[Align the video so that I say “Joint X” when X joint begins to move.]*

1. Joint 1 (prismatic)
2. Joint 2 (prismatic)
3. Joint 3 (prismatic)
4. Joint 4 (revolute)
5. Joint 5 (revolute)
6. Joint 6 (prismatic)

Notice how the end effector changes intuitively as each joint is manipulated.

**Forward Differential Kinematics**

*[Title drops. Video is paused.]*

We next analyze the forward differential kinematics of our manipulator.

*[Show finite difference approximation and Jacobian multiplication from slides]*

Internally, this program uses a finite-difference representation of a Jacobian matrix to compute the end effector velocities, governed through the following formula.

*[Wait 1 second after the previous sentence is finished, then fade away both pictures. The video should begin around this point, maybe even a little before.]*

Consider what happens when we manipulate the velocities of each joint. Notice how the end-effector velocity changes as we manipulate:

*[Align the video so that when I change Q’X, the Q’X values begin to move.]*

1. Q’1 (prismatic)
2. Q’2 (prismatic)
3. Q’3 (prismatic)
4. Q’4 (revolute)
5. Q’5 (revolute)
6. Q’6 (prismatic)

*[Align the video so that I start moving the joint positions when I start the next sentence.]*

Additionally, notice how changing joint positions also changes the end effector velocity as appropriate.

**Inverse Differential Kinematics**

*[Title drops. Video is paused.]*

In my simulation, deriving the inverse differential kinematics only considers the linear end effector Jacobian. I wanted to add angles, but the simulation never worked properly with them, sadly.

*[Title disappears.]*

Because I’m working with so many joints, and because the Jacobian matrix I derive has more columns than it has rows, I’m bound to have some free variables, as determined through my simulation.

*[Highlight top-left inverse table]*

By specifying additional constraints to these variables, I can derive the required joint velocities for the other joints, as I’m showing in this example.

*[Align so that I begin manipulating the Q values in the middle of the previous sentence. Manipulate each of Q with LSHIFT+{4,5,6}+LMB]*

As expected, adjusting the required end-effector linear velocity imposes its own set of joint velocity changes.

*[Align so that I begin manipulating the velocity in the middle of the previous sentence. Manipulate each end velocity with LCTRL+{1,2,3}+LMB]*

Finally, notice how changing each manipulator orientation also changes the velocities as expected.

*[Align so that I begin changing the joint positions as the previous sentence begins.]*

**Statics**

*[Title drops. Video is paused.]*

This simulation supports analysis of a force and moment applied to the end effector.

*[Title disappears.]*

We can see that, when we transform this couple to the base frame, while the force moves directly, the moment changes based on the end effector position.

*[Show solo joint and base frame. Use LSHIFT + RMB / MMB, then use LCTRL + RMB / MMB]*

Finally, using the inverse of the Jacobian as I have derived before, we can compute the joint reactions for each joint as appropriate.

*[Show all joint moment reactions and move the force and moment vectors around]*