The Kinematics and Dynamics of a Humanoid Robot Arm

for the FIRA Robot Games

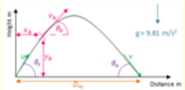
Kinematics

Kinematics is the study of motion with relation to just the time variant properties of a system, without regard to the forces which are required to produce that motion.

For the robot arm, it is the analytical relationship between the joint positions and the end-effector position and orientation.

The Ball's Trajectory

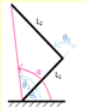
After the ball is thrown it can be modelled as a particle acting under gravity, using projectile trajectory maths and the equations of motion.



The Ball's Trajectory

This finds the throwing conditions in Cartesian coordinates.

Inverse Kinematics



Inverse kinematics finds the joint angles and positions required to place the end effector in the desired throwing position (x,y).

The Geometric Inverse Kinematics Solution:

Using the Cosine Law:	Using the Sine Law: $\frac{\sin B}{b} = \frac{\sin C}{c}$
$c^2 = a^2 + b^2 - 2ab\cos C$	
$\theta_2 = \arccos\left(\frac{x^2+y^2-l_1^{\,2}-l_2^{\,2}}{2l_1l_2}\right)$	$\theta_1 = \arctan\left(\frac{y}{x}\right) - \arcsin\left(\frac{l_2\sin(\theta_2)}{\sqrt{x^2+y^2}}\right)$

Forward Kinematics

Forward kinematics calculates the linear position and orientation of the end-effector from the joint positions. It relates the angular positions to the Cartesian positions.

$$(x,y) = ((l_1\cos\theta_1 + l_2\cos(\theta_1+\theta_2)),(l_1\sin\theta_1 + l_2\sin(\theta_1+\theta_2)))$$

The joint's angular speed from the Cartesian speed is found using the Jacobian mathematical method.

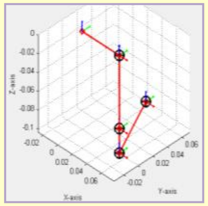
The Jacobian:	The Jacobian relationship: $\dot{x} = J\dot{q}$
$J(q) = \begin{bmatrix} \frac{\partial h_1(q)}{\partial q_1} & \frac{\partial h_1(q)}{\partial q_2} \\ \frac{\partial h_2(q)}{\partial q_1} & \frac{\partial h_2(q)}{\partial q_2} \end{bmatrix}$	$\begin{bmatrix}\dot{\theta}_1 \\ \dot{\theta}_2\end{bmatrix} = [J]^{-1}\begin{bmatrix}\dot{x} \\ \dot{y}\end{bmatrix}$

By: Claire Tobin

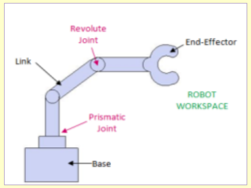
Supervisor: Dr. Guido Herrmann

The Humanoid Robot Arm Holding the Ball

Human Arm



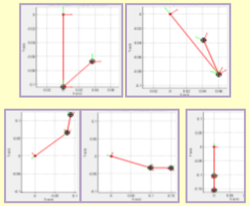
Matlab Representation of 4DOF Arm Model



The Main Features of a Robot

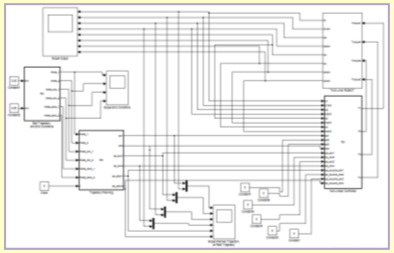
Robot Arm Controller Design

- Kinematic and dynamic relationships were found for the arm
- A trajectory was created
- A Feedback Linearization Nonlinear Controller was designed
- A control system model was put together in Matlab using Simulink

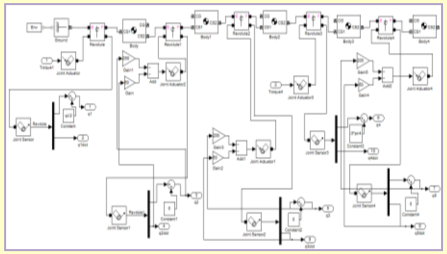


Snapshots of 2DOF Arm Model's Movement

CAD Representation of 4DOF Arm Model



Matlab Model and Controller for 4DOF Arm System



Simulink Model of 4DOF Arm

Dynamics

Dynamics is the relationship between the motion of an object and the motion's causes, where mass and thus forces, as well as time variant properties are studied.

The dynamics of a robotic system are made up of inertia effects and the work the system is doing.

The Generic Dynamic Equation:

Inertia/Mass x Acceleration + Corriolis/Centrifugal Effects + Gravity Effects = Driving Force/Torque

$$M(q)\ddot{q} + V(q,\dot{q})\dot{q} + G(q) = \tau, \quad q \in R^{n_q}$$

- q is the vector of the robot's joint variables
- τ is the actuator torque/force
- $M(q)$ is the mass and inertia matrix
- $V(q,\dot{q})$ is the Corriolis and centripetal vector
- $G(q)$ is the gravity vector

The dynamic matrices were derived using the Lagrange-Euler method.

Trajectory Planning

Trajectory Planning is a method of making a manipulator move from one position to another in a smooth and controlled manner, by giving each joint a smooth function of time to follow. The trajectory is described by several via points through which a polynomial is fitted.

Nonlinear Controllers

The robot arm needs a controller, to relate the desired trajectory to the motors' torques. A controller is effectively the dynamical relationship between the system's output and its actuation.

A practical mechanical system always has nonlinear dynamics due to the damping and friction effects of the natural environment. A nonlinear controller will linearize the dynamic equation.

Joint-Based Feedback Linearization Control

$$\tau_q = V_m(q,\dot{q})\dot{q} - G(q) + M\ddot{q}_d(t) + MK_d(\dot{q}_d(t) - \dot{q}(t)) + MK_p(q_d(t) - q(t))$$

Cartesian Feedback Linearization Control

$$\tau_x = \left(V_m(q,\dot{q}) - M(q)J^{-1}(q)\dot{J}(q)\right)\dot{q} + G(q) + M(q)J^{-1}(q)\left(\ddot{x}_d + K_p(x_d - x) + K_d(\dot{x}_d - \dot{x})\right)$$

Third Year Research Project 2010/2011

University of Bristol

Department of Mechanical Engineering