Motion and Force Analysis of a 6-DoF Goliath-class Engels Robot Arm

Jimmy Chen, *Member, IEEE,* Leonid Shuster, *Fellow, YoRHa*

***Abstract*—Nier: Automata is an action role-playing video game published by Square Enix for the PlayStation 4 video game console. In this video game, all of the characters are robots: the robots come in different sizes and shapes, and serve different functions. This project addresses the Goliath-class Engels. Engels has humanoid dimensions and stands over a hundred meters tall. The aim of this project is to use MATLAB to model the specific arm swinging motion of Engels as seen in the video game, and analyze the resulting kinematics and forces produced.**

***Index Terms*—Role-playing, PlayStation 4, Goliath-class Engels, MATLAB.**

I. Introduction

IN the Robotics Manipulation course at the University of California, Santa Cruz, we were assigned a project that requires us to explore robotics manipulation using knowledge we obtained from the course, as well as online sources. Since we both play video games as a hobby, we decided to use video games as a medium for analyzing an aspect of robotics manipulation. We chose to look for a robot from the video game Nier: Automata. Nier: Automata is an action role-playing video game published by Square Enix for the PlayStation 4 video game console. In this video game, all of the characters are robots: the robots come in different sizes and shapes, and serve different functions. For this project we analyze the Goliath-class machine named Engels, seen in Figure 1. Engels is a humanoid robot that stands over a hundred meters tall with two 6 Degrees-of-Freedom (DoF) robotic arms.

The aim of this project is to use MATLAB to model the specific arm swinging motion of Engels as seen in Nier: Automata, and analyze the kinematics and forces during this swing. While MATLAB has its own robotics toolboxes, we decided to program the arm motion and relevant kinematic functions from scratch. This allows us to see how robotic arms move and function at a fundamental level. This paper is structured as follows. Section II presents related works. Section III shows the methodology used. Section IV shows simulation results. Section V concludes and discusses future work.

II. Related Work

The dynamics of robotic arms have been widely studied throughout academia and industry. Besides understanding how each part of the robotic arm relates to one another, researchers also need to know how manipulators interact with their manipulated objects and surroundings in order to achieve desired tasks.

Fig. 1: Engels

Iqbal et al. [1] proposes a model for a 6 DoF robotic arm that allows the control for a manipulator to achieve any reachable position and orientation of unstructured environments. They utilize a forward kinematic model using Denavit Hartenberg (DH) parameters for position placements. After finding the desired position and orientation of the robot end-effector, inverse kinematics derived from the model is used to acquire corresponding joint angles. The forward kinematic model is verified through the Robotics Toolbox of MATLAB, and the inverse kinematic model is implemented on a real robotic arm. Iqbal et al. [1] show that their developed model allows the end-effector to reach desired coordinates within a precision of ± 0.5cm. The deviation is caused by variations in precision in the 6 DoF robotic arms. The approach can be used to solve related kinematic problems in similar robotic manipulator types.

Senoo et al. [2] proposes a kinetic chain approach for achieving high-speed dynamic manipulation for ball throwing of a 3 DoF robotic arm through different types of experiments. Their first strategy is the "kinetic chain", a control method based on observations in human throwing motion. This strategy produces two types of base functions derived from approximate dynamics and produces efficient motion. The next strategy uses the analysis of contact states during a fast swing for release control. The release method uses features that allow the apparent force generated by high-speed motion to maintain robust control of the ball direction. They then describe the experimental setup and experimental results. Their results show that high-speed throwing motion was achieved, and the system can be used for further investigating detailed robustness evaluation of ball control.

J. Chen and L. Shuster are with the Department of Electrical and Computer Engineering, University of California Santa Cruz, Santa Cruz, CA, 95064 USA.

Manuscript received April 19, 2019; revised August 26, 2020.

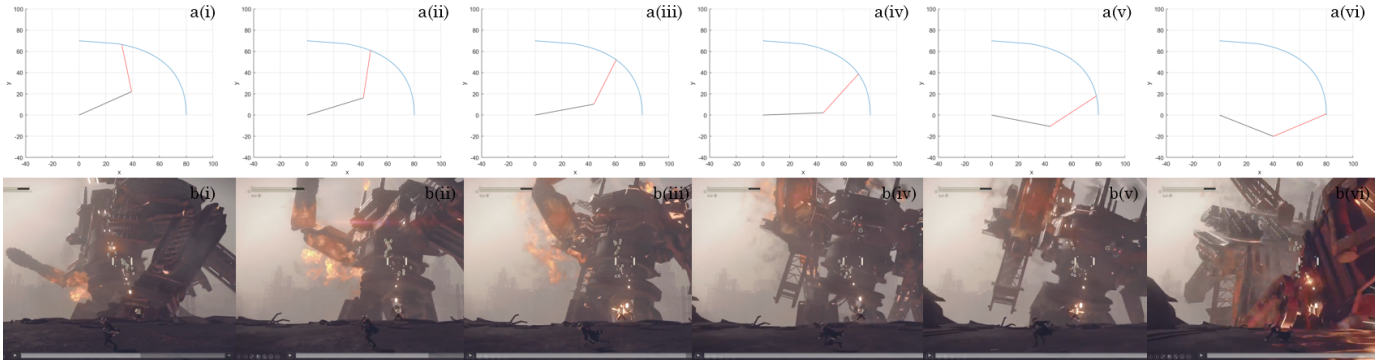


Fig. 2: Comparison between the simulated motion and the analyzed swing motion. Top row, a(i) - a(vi) shows the MATLAB simulation in 6 time stamps. Bottom row, b(i) - b(vi), shows the swing motion in 6 time stamps

The throwing motion studied by Senoo et al. [2] is similar to the swinging motion of Engels that we analyze. We will be simulating this motion and calculating its kinematic properties in MATLAB.

III. Method Overview

The arm swing motion is depicted in Figure 2: the end-effector moves from above Engels' head to the platform. This motion can be modelled by a planar arm in 3-D space. To reach our goal of simulating the arm swing and calculating the kinematics and forces, we programmed and calculated the different facets of robotic arm motion before combining the results. We first make assumptions about the dimensions of Engels' arm, and then describe the methods we used: (1) Implementing forward kinematics. (2) Implementing inverse kinematics. (3) Deriving the Jacobian matrix. (4) Deriving equations for the force on the end-effector.

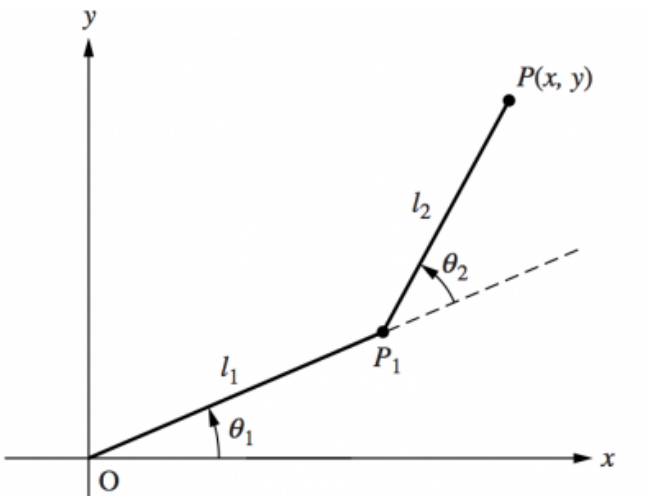


Fig. 3: Simplified schematic of Engels' robotic arm used in MATLAB

A. Assumptions

We have made assumptions on the following attributes of Engels based on various online sources. The overall height of Engels is approximately 110 meters. Based on this we estimated the overall arm length to be 90 meters. Link 1, the upper arm, is 45 meters. Link 2, the forearm, is 45 meters. The end-effector does not have a wrist joint. Since we are modelling the arm as a standalone robotic arm, the base of the robot was set as the shoulder joint, and there is only motion in 2-D space along the x-axis and y-axis with a rotation along the z-axis. The diagram for this model is shown in Figure 3.

TABLE I: DH parameter of our system

i	α_{i-1}	a_{i-1}	θ_i	d_i
1	0	L_1	θ_1	0
2	0	L_2	θ_2	0

B. Implementing Forward Kinematics

Forward kinematics is the process of relating a point's position and orientation to another, and is especially applicable when trying to find the position of the end-effector. In order to do so, the angles and lengths of the joints must be known and used as DH parameters. For our robot, we based our DH parameters on the diagram shown in Figure 3 and our assumptions, and produced the DH parameter chart seen in Table 1.

$$ {}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Fig. 4: General transformation matrix

Then, using the standard transformation matrix shown in Figure 4 and the DH parameters chart, we filled out the transformation matrices that would allow us to go from one coordinate frame to another. Multiplying the transformation matrices all the way through to the end-effector allowed us to find the position of the end-effector at any moment as shown:

$$X_{ee} = L_2 c_{12} + L_1 c_1$$

$$Y_{ee} = L_2 s_{12} + L_1 s_1$$

C. Implementing Inverse Kinematics

Inverse kinematics is the opposite of forward kinematics: knowing the position of the end-effector, find the angles of the joints. There are several ways to approach this, but we used the geometrical approach. Using the same diagram from Figure 3, we solved for the following angles in our system at any moment:

$$\theta_2 = acos(\frac{X_{ee}^2 + Y_{ee}^2 - L_1^2 - L_2^2}{2L_1L_2})$$

$$\theta_1 = atan2(\frac{X_{ee}}{Y_{ee}}) - atan2(\frac{L_2 s_2}{L_1 + L_2 c_2})$$

D. Deriving the Jacobian Matrix

The Jacobian matrix is primarily used as a matrix of first-order partial derivatives of a vector-valued function, and with our system the Jacobian matrix would relate the derivatives of the end-effector's position (end-effector velocity) with the derivatives of the joint angles (angle velocities) as shown here:

$${}^0v = {}^0 J(\Theta)\dot{\Theta}$$

To do so, we started with the position of the end-effector that we derived from forward kinematics. Then, taking the derivative, we were left with the following equations relating change in position with change in angles:

$$\dot{X_{ee}} = V_{X_{ee}} = -L_1\dot{\theta_1}s_1 - L_2(\dot{\theta_1} + \dot{\theta_2})s_{12}$$

$$\dot{Y_{ee}} = V_{Y_{ee}} = -L_1\dot{\theta_1}c_1 - L_2(\dot{\theta_1} + \dot{\theta_2})c_{12}$$

Finally, factoring out the change in angles, the Jacobian matrix was found, and could be used to solve for either the change in position or change in angles knowing one or the other:

$$\begin{pmatrix} \dot{X_{ee}} \\ \dot{Y_{ee}} \end{pmatrix} = \begin{bmatrix} -L_1 s_1 - L_2 s_{12} & -L_2 s_{12} \\ L_1 c_1 - L_2 c_{12} & -L_2 c_{12} \end{bmatrix} \begin{bmatrix} \dot{\theta_1} \\ \dot{\theta_2} \end{bmatrix}$$

$$J_{ee} = \begin{bmatrix} -L_1 s_1 - L_2 s_{12} & -L_2 s_{12} \\ L_1 c_1 - L_2 c_{12} & -L_2 c_{12} \end{bmatrix}$$

E. Deriving the Forces on the End-Effector

In order to find the forces on the end-effector, using the Newton-Euler equations, we derived the equations for the torques on all the joints as shown:

$$\tau_1 = m_2 l_2^2(\ddot{\theta}_1 + \ddot{\theta}_2) + m_2 l_1 l_2 c_2(2\ddot{\theta}_1 + \ddot{\theta}_2) + (m_1 + m_2)l_1^2\ddot{\theta}_1 - m_2 l_1 l_2 s_2 \dot{\theta}_2^2$$
$$-2m_2 l_1 l_2 s_2 \dot{\theta}_1\dot{\theta}_2 + m_2 l_2 g c_{12} + (m_1 + m_2)l_1 g c_1,$$
$$\tau_2 = m_2 l_1 l_2 c_2 \ddot{\theta}_1 + m_2 l_1 l_2 s_2 \dot{\theta}_1^2 + m_2 l_2 g c_{12} + m_2 l_2^2(\ddot{\theta}_1 + \ddot{\theta}_2).$$

Knowing the torque, the same Jacobian used earlier to relate the change in position with the change in angles could be used to relate the torque and forces on the end-effector by using the following equation:

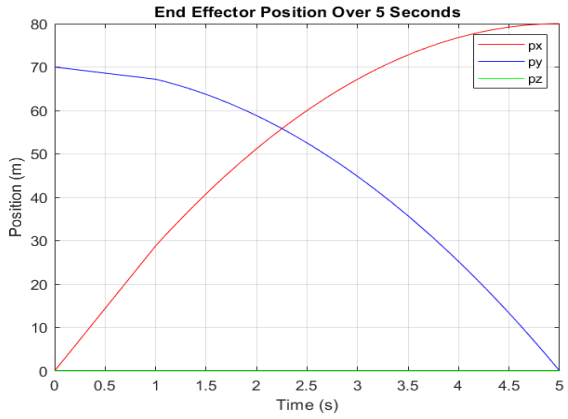
$$\tau = J^T \mathcal{F}$$

Then, by manipulating this equation, the force could be determined from the torque and Jacobian matrix:

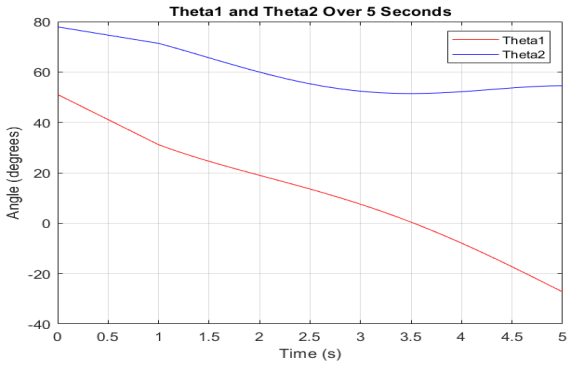
$$F = J^{T^{-1}}\tau$$

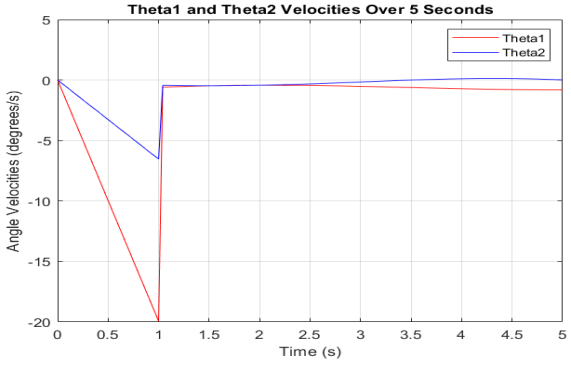
IV. Simulating Arm Movement and Results

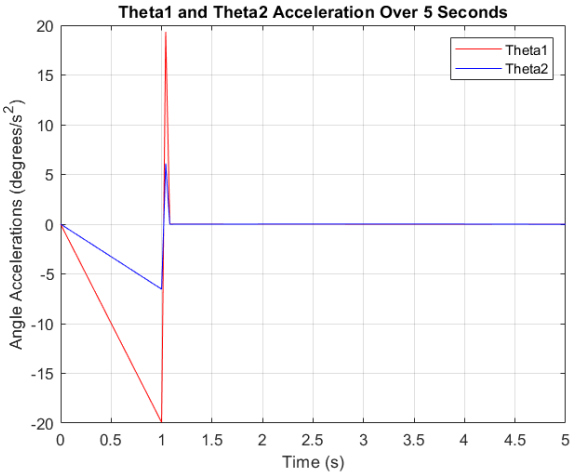
After implementing the previously mentioned equations into MATLAB, we generated a predetermined path for the end-effector modeling that in Figure 2 over five seconds with 100 time samples:



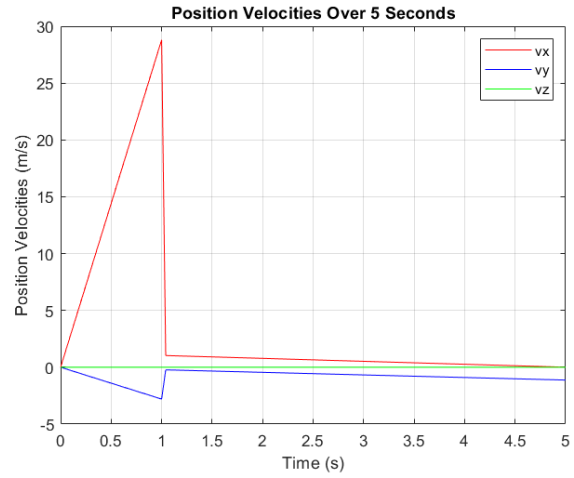
We then utilized our equations and methods to track various characteristics of the robot arm over time. Knowing the position of the end-effector, we first used inverse kinematics to determine the angles of the joints at every end-effector position, as well as the angle velocities and accelerations:



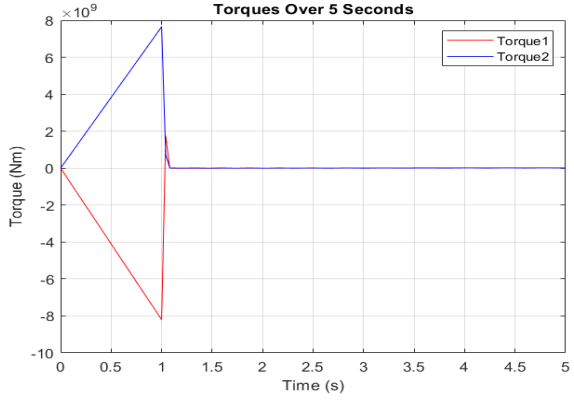




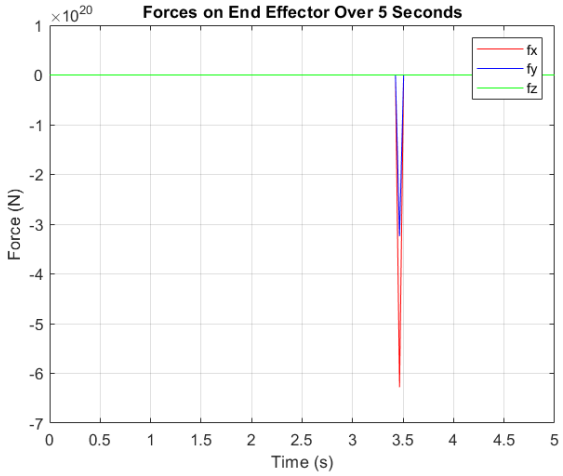
Then, after determining the DH parameters for our system, we used forward kinematics to go from the first joint to the second joint and find its position which we would need in order to draw out our robot arm. Next, we used the Jacobian we derived from before with the current velocities the angles were at, and used them to find the velocity of the end-effector:



Next, with the respective mass, link length, angle velocity, and angle acceleration of each joint, we solved for the torque on every joint by using the torque equations derived from before:



With the torques, we were finally able to solve for the forces on the end-effector by using our derived force equation:



From a single arm swing from Engels, we tracked various components including the end-effector position, velocity, and acceleration, as well as the joint angles, velocities, and accelerations. Most importantly, we were able to use all this to track the torques along its path, and determine what the force of impact will be at any given moment and find out how much damage a single arm swing can produce.

V. Conclusion

In this project, we used fundamental robotics manipulation concepts to analyze various characteristics of a single arm swing from Engels. After deriving the equations for forward kinematics, inverse kinematics, the Jacobian matrix, and torques and forces, we produced a MATLAB simulation that utilized these methods to show the end-effector's position, velocity, and acceleration, the joints' angles, velocities, and acceleration, the torques on the system, and the forces on the end-effector. In the future we plan to design a more accurate simulation of Engels' arm swing by adding additional arm links to represent the movement of Engels' body as it rotates before it swings.

References

[1] Iqbal, Jamshed, R. Ul Islam, and Hamza Khan. "Modeling and analysis of a 6 DOF robotic arm manipulator." Canadian Journal on Electrical and Electronics Engineering 3.6 (2012): 300-306.

[2] Senoo, Taku, Akio Namiki, and Masatoshi Ishikawa. "High-speed throwing motion based on kinetic chain approach." 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2008.

[3] McKerrow, Phillip John, and Phillip McKerrow. Introduction to robotics. Vol. 3. Sydney: Addison-Wesley, 1991.

[4] Corke, Peter. Robotics, vision and control: fundamental algorithms in MATLAB® second, completely revised. Vol. 118. Springer, 2017.