Rotation Project : Simulating Rigid Body Dynamics with Contact and Friction

Manas Bhargava
manas.bhargava@ist.ac.at
Institute of Science and Technology, Austria

Supervised by - Prof. Bernd Bickel
bernd.bickel@ist.ac.at
Institute of Science and Technology, Austria

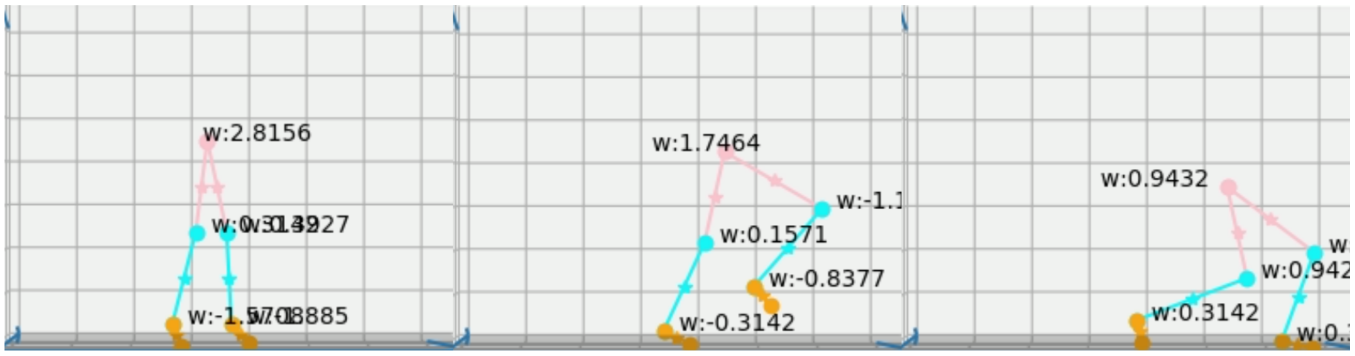

Figure 1: Humanoid walking on the ground after estimating the torque trajectories using our inverse dynamics method.

1 INTRODUCTION

1.1 Motivation

In present times, robots have become an integral part of the human world. They are capable of performing tasks of extreme difficulty with very high precision and in environment which are difficult for humans to access. We can now see robots working in almost every domain of life - from solving rubik's cube quickly to helping us as a house cleaner and from being an integral part of industrial assembly line to explorers out in Mars searching for new life forms. But controlling these robots to perform a certain motion trajectory is a non-trivial task. Consider the problem of humanoid walking, it is very difficult to estimate the torque values at different time steps that should be applied by the motors to make the robot walk. It requires significant human labour and time to get the required torque trajectories. Doing this experiment on a real robot might even damage it if the torque trajectories are not administered carefully. Therefore, having a virtual forward dynamics simulator which imitates the real world by including both contact and friction is helpful. It allows the researcher to test out different torque trajectories without harming the robot. However, trying out different torque values to obtain the correct motion takes a lot of human effort and time, even to generate a simple humanoid walking motion. This problem makes it very difficult to generalize such control method over different robot configuration and motion cycles. Thus, there is a need of an inverse dynamics based control system. It allows one to estimate the torque profile that is needed to get a given motion profile while including both contact and friction.

1.2 Overview of what we did

We formulated the problem of forward dynamics and inverse dynamics including both contact and friction mathematically and developed a rigid-body simulator in 3d to do both these tasks. We created our own dynamics system instead of using commercially available softwares as a black box. This enabled us to learn in detail about the technical difficulties related to implementing rigid body dynamics involving contact and friction. We showcased the motion of humanoid and 4-legged robot using the inverse dynamics formulation by following a particular trajectory while including contact and friction.
This report is organised as follows : first, we introduce the readers to robotics. We then discuss the implementation of both forward dynamics and inverse dynamic methods. Later, we discuss the results that we obtained using these methods. At last, we discuss the failure cases of using inverse dynamics method as a controller for robots and the possible extension to our approach that can help in alleviating the failure cases.

2 A BRIEF INTRODUCTION TO ROBOTICS

This section discusses what do we mean by robots and the existing methods that are used to control the robots.

2.1 What are robots?

Robots are basically any autonomous system of rigid bodies with a central control system which is programmed by the user. The rigid bodies are made up of system of links where every link contains a series of joints and a rod connecting these joints. Figure 2 shows the rigid body system for a 4-legged robot. Every joint have their own local coordinate system which specifies in which direction they allow the rod to rotate or translate. These joints in the rigid body system can be of two types - revolute joints and prismatic joints. Revolute Joints allow the link to rotate about the local z-axis of the joint. While prismatic joints allow the rod to translate about the local z-axis of the joints. The way these joints and their axes are aligned is given by the *Denavit–Hartenberg* parameters (also called DH parameters) [Hartenberg and Denavit 1964]. In DH parametrization, an i^{th} joint is represented by 4-parameters -

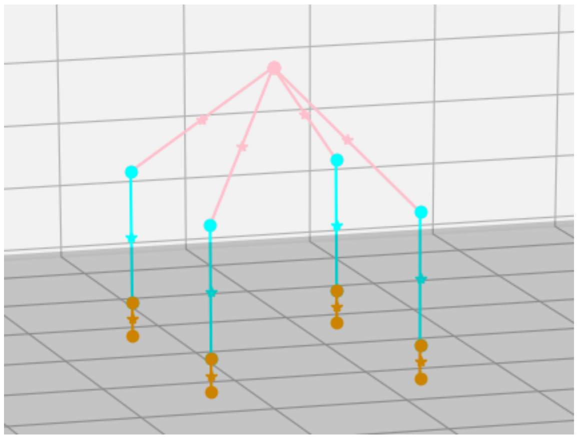

Figure 2: A rigid body system consisting of 4 set of links where each link consist of 3 actuated revolute joint. We also have two 2 prismatic joints attached at the root of the rigid body which are not-actuated and are moved by the influence of normal and friction forces. Thus in total we have 12 actuated joints and 2 non-actuated joints.

- a: length of the link
- θ: the angle with which rotates the previous x-axis (x_{i-1}) is rotated with respect to previous z-axis (z_{i-1}) to obtain current x-axis (x_i)
- α: the angle with which rotates the previous z-axis (z_{i-1}) is rotated with respect to current x-axis (x_i) to obtain current z-axis (z_i)
- d: displacement of the rod in z axis direction.

Amongst the 4 parameters, 1 of them represent the variable parameter which can be actuated by the joint with the help of the motors present in the robot for each joint. For prismatic joint this variable parameter is d and for revolute joint the variable parameter is θ. The Figure 3 describes the usage of DH parameters for 2-link system. Thus for each joint i we can define the transformation matrix (A_i) that takes us from joint i to joint i+1 with appropriate axis transformation and is defined as follows -

$$A_i = Rot_{z,\theta_i} Trans_{z,d} Trans_{z,a} Rot_{x,\alpha_i} =$$

$$\begin{bmatrix} c_\theta & -s_\theta & 0 & 0 \\ s_\theta & c_\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot$$

$$\begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_\alpha & -s_\alpha & 0 \\ 0 & s_\alpha & c_\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_\theta & -s_\theta c_\alpha & s_\theta s_\alpha & a c_\theta \\ s_\theta & c_\theta c_\alpha & -c_\theta s_\alpha & a s_\theta \\ 0 & s_\alpha & c_\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Knowing the A_i transformation matrix for each joint at a given time leads to articulation of our rigid body system to a specific pose. [Mirtich 1996] provides a very good introduction on how the rigid body are simulated based on the DH parameterization system.

2.2 How to control robots?

The methods that are widely used to control robots falls under two major categories - Open Loop Method and Closed Loop Method.

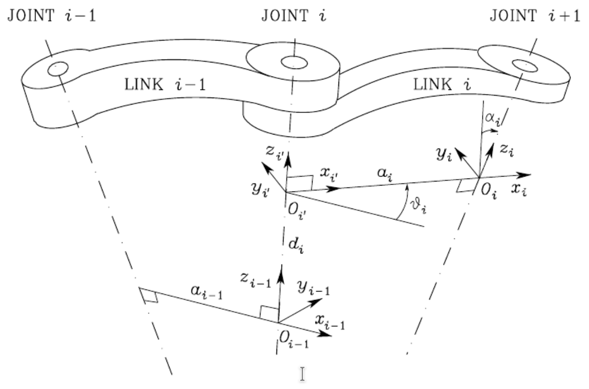

Figure 3: The figure shows the different DH parameters for a joint and how it helps in determining the local axis for the joints.

Open loop methods allows the user to specify the torque values for the entire task. These control methods are very fast as they just ask the robot to follow the instruction blindly without taking into account any feedback from the environment in which they are functioning. Thus, these control methods are usually found in systems where the main aim is to do tasks at a faster pace; like the robots that are used in industrial assembly line or when you are trying to break the record of fastest rubik's cube solving.

On the other hand, Closed loop method have a feedback mechanism that controls the robot's action being performed. They usually have a number of sensors attached to them and these sensors give back feedback about the environment which are then used to make informed decisions. Such control methods are usually found in robots where the possibility of uncertain events is taken into account while performing a certain task. Careful interaction with the environment without causing harm to either environment or the robot is more important than the speed with which the task is done. Some examples are driver less cars, exploring robots in space, etc.

The way we have define this control methods, one can think of forward dynamics being very similar to the open-loop method where we define some torque trajectory which we just apply to simulate our robot and generate the motion profile. On the other hand, inverse dynamics can be thought of as a closed control method, where we have some given trajectory that we need to follow and we calculate the torque value that should be applied by the motors of the robot to follow that trajectory. In the following sections, we discuss both these methods in detail.

3 FORWARD-DYNAMICS

Forward dynamics refers to the method of moving the robots under the influence of time-varying torque which is applied by the motors. As a result of these torques the robots performs certain action while following the laws of Physics. Tuning the torque value of the different motors to make the robot follow a particular trajectory can be extremely hard task and requires either trial and error before obtaining the right torque trajectory or some learning based methods like reinforcement learning. Note that humans also have to practice quite a lot (tune the muscle torque trajectories) before they can walk without falling.

3.1 Simulating the robot

We first write the laws of physics using the Lagrangian mechanics. Each of the different free parameter that can be actuated by the motors are represented by : q = $\left[q_0, q_1, q_2, ...q_{n-1}\right]$ This representation is also called *generalized coordinate representation or minimal coordinate representation* as it uses only the minimum number of parameters that are necessary to represent the rigid body system. Thus from now on, all the terms that we define are written in the generalized coordinate system. We define Lagrangian (L) as follows -

$L = K.E. - P.E.\ where,$

$K.E. = \frac{1}{2}\dot{q}^T \sum_{i=0}^{n} \left[m_i J_{v_i}(q)^T J_{v_i}(q) + J_{w_i}(q)^T R_i(q) I_i R_i(q)^T J_{w_i}(q)\right] \dot{q},$

$and\ \ P.E. = \sum_{i=0}^{n} g^T r_{c_i} m_i\ .$

The equations of motion are then obtained by writing and simplifying the *Euler-Lagrange* equation given by -

$$\frac{\partial}{\partial t}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = \tau,$$

which when simplifies yield's -

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau. \quad (1)$$

where, **M(q)** represents the Mass matrix of the system for some pose configuration q at time t, **C(q,q̇)q̇** represents the coriolis and centrifugal forces that act on the rigid body and **g(q)** represents the force due to gravity acting on the body and τ represents the internal force/torque applied by the motors to actuate the motors. Please refer to [Liu and Jain 2012] for further details on deriving this equation from first principles.
The equation (1) is a non-linear second order differential equation representing the motion of rigid body system in time. Analytical solutions for such system are difficult to obtain. Thus we rely on numerical integration schemes to solve this differential equation and obtain q as a function of time for each joint. We experimented with different integration schemes - implicit euler schemes using fixed point iteration, explicit euler and semi-implicit euler. Each of them has its own pros and cons. Implicit euler damps the total energy of the system while explicit euler incurs error over time and have the tendency to blow up. But at the end, we ended up using a variant of explicit euler for this task. Coming up with a good higher order integration scheme for this task is taken into consideration for future work. Figure 4 showcase the motion of double-pendulum system using our forward dynamics simulation.

3.2 Introducing Contact and Friction

We now introduce the contact and friction forces in our equation (1) to accommodate these phenomenons in our forward dynamics as well. These forces are applied on the rigid body usually at the end effector. Thus we use the jacobian matrix at that point to transfer the forces from cartesian coordinate system into generalized coordinate system. We represent the jacobian matrix corresponding to normal

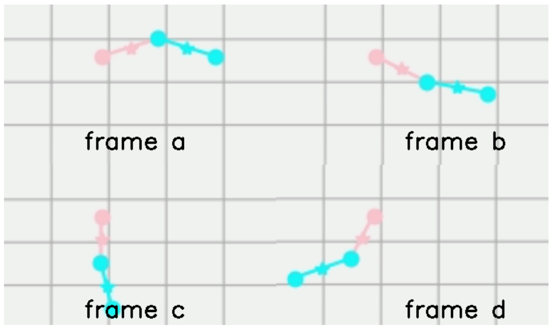

Figure 4: The figure shows motion of 2d-pendulum using forward dynamics. The system was able to perpetuate for a longer duration in time without blowing up the energy.

force (f_n) with N and jacobian matrix corresponding to friction force (f_d) with B as derived in [Tan et al. 2003]. After incorporating these forces we have the modified dynamics equation given by -

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau + Nf_n + Bf_d. \quad (2)$$

3.3 Solving Linear Complementarity Problem

Introduction of these new variables f_n and f_d has complicated our problem as we don't know their values beforehand. Their values are constrained with the current motion of the rigid body system and thus imposes some constraint on the motions. There are 3 constraints that are introduced namely - normal force constraint, friction force constraint and coulomb's friction cone constraint. [Stewart and Trinkle 2000] and [Tan et al. 2003] does a very good job in introducing these constraints and explaining them. Using the contact and friction constraints it can be shown that the equation (3) represents the linear complementarity problem (LCP) [Tan et al. 2003].

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \Delta t N^T M^{-1} N & \Delta t N^T M^{-1} B & 0 \\ \Delta t B^T M^{-1} N & \Delta t B^T M^{-1} B & E \\ \mu & -E^T & 0 \end{bmatrix} \begin{bmatrix} fn \\ fd \\ \lambda \end{bmatrix} + \begin{bmatrix} N^T M^{-1} \tau^* \\ B^T M^{-1} \tau^* \\ 0 \end{bmatrix} \quad (3)$$

where τ^* is a constant value for that time step "n" and is given by -

$$\tau^* = M\dot{q}^n - \Delta t(C(q^n, \dot{q}^n) + g(q^n) - \tau^n) \quad (4)$$

Here the equation (3) is of the form $w = Az + q$. The LCP problem consists in finding w > 0, z > 0, such that w = Az + q and $w^T z = 0$. Linear Complementarity problems occurs in various different fields and are solved by using **LEMKE**'s algorithm [Lloyd 2005]. Thus, solving this LCP gives us the value of f_n and f_d which we can put in equation (2) and solve it using numerical integration method like forward euler to simulate the motion of our rigid body system.

4 INVERSE-DYNAMICS

We define our second control method of inverse dynamics in this section. In this method, we try to solve for torque values as a function of time given a motion profile that the robot is asked to follow. Our formulation closely follows the work done [Zapolsky and Drumwright 2017] but the final matrix obtained is significantly different. We define a new matrix P which helps in choosing which all variables are actually actuated by the motor and which variables are free to move under the influence of contact and friction forces. For the case of humanoids we have 6 revolute joints which are actuated by the motors attached to them and 2 prismatic joints which follow are only influenced by the force applied by normal reaction and friction. Thus, we can rewrite our equation (2) as follows -

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = P^T\tau + Nf_n + Bf_d, \tag{5}$$

$$Pv^+ = \dot{q}_{des}. \tag{6}$$

where v^+ represents the value of $\dot{q}$ at time t + Δt and $\dot{q}_{des}$ represents the desired trajectory for the joints that can be actuated. We again use the normal and friction constraints and obtain another Linear Complementarity problem given by.

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \Delta t N^T M^{-1} X1 & \Delta t N^T M^{-1} X2 & 0 \\ \Delta t B^T M^{-1} X1 & \Delta t B^T M^{-1} X2 & E \\ \mu & -E^T & 0 \end{bmatrix} \begin{bmatrix} fn \\ fd \\ \lambda \end{bmatrix} + \begin{bmatrix} N^T M^{-1} \tau_1^* \\ B^T M^{-1} \tau_2^* \\ 0 \end{bmatrix} \tag{7}$$

where : $X1 = N - P^T(PM^{-1}P^T)^{-1}PM^{-1}N$,
$X2 = B - P^T(PM^{-1}P^T)^{-1}PM^{-1}B$,
$\tau_1^* = N^T M^{-1}\tau^* + N^T M^{-1}P^T(PM^{-1}P^T)^{-1}(\dot{q}_{des} - PM^{-1}\tau^*)$
$\tau_2^* = B^T M^{-1}\tau^* + B^T M^{-1}P^T(PM^{-1}P^T)^{-1}(\dot{q}_{des} - PM^{-1}\tau^*)$

We can solve this equation using LEMKE's algorithm to get the values of f_n and f_d which we put in equation (5) and as before solve it using our numerical integrator to perform the desired motion of rigid body system such that it follows a particular trajectory. Figure 1 showcase the different poses of humanoid walking on the ground involving both friction and contact forces simulated under inverse-dynamics based formulation.

5 RESULTS

We created our own forward dynamics and inverse dynamics simulator in 3D which included both contact and friction. Python and basic python libraries were used for creating these simulators. The only additional implementation that we borrowed was the LEMKE's solver by AndyLamperski. We showcased our methods by testing them on different robot configurations including double link pendulum, 2-legged-humanoid and 4-legged-robot.

6 FUTURE WORK

In this section, we discuss the further research directions that can be pursued to improve on our approach and alleviate the limitations that we faced. The figure 5 displays the limitations of our method leading to robot falling on the ground.

- Better numerical integration method can be use to simulate the rigid body system. Currently, we applied the simple variant of explicit euler method to simulate the robot forward in time whose accuracy is of first order. Obtaining higher order - runge-kutta methods would help us in obtaining better approximates to our solution of both forward dynamics and inverse dynamics.
- Implementing a better robust method to solve the LCP problem. Currently, LEMKE's algorithm is the state of the art for solving LCP problem but it assumes that the A matrix that we obtain in (3) and (7) to be a P-matrix for it to give out a solution. This is not the case for the matrix that we have in our formulation of both forward dynamics and inverse dynamics, which leads to non-availability of solutions in some time steps. Thus having a robust LCP solver that can work with Non-P matrices would be extremely helpful.
- It would be interesting to test the functionality of inverse dynamics as a controller by testing the learnt torque trajectories on real robots. With this we will be able to realize the reality gap between the actual robots and the simulation results.

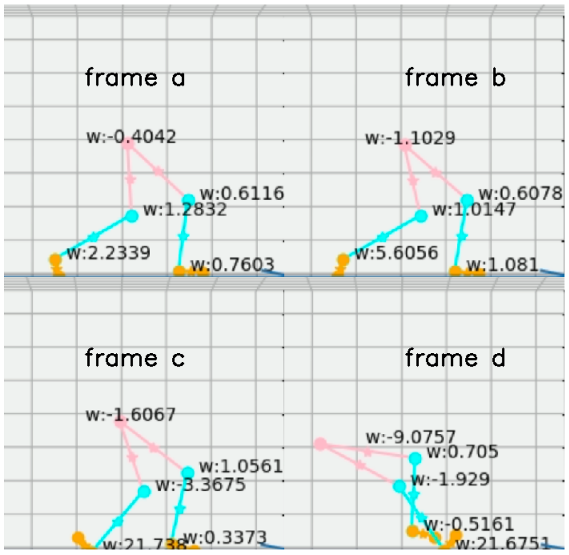

Figure 5: The figure shows failure cases of the human walking because of cumulation of errors over a period of time when the learnt torque trajectories from inverse dynamics are applied to forward dynamics simulation.

REFERENCES

Richard S. (Richard Scheunemann) Hartenberg and Jacques Denavit. 1964. *Kinematic synthesis of linkages.* New York : McGraw-Hill. Includes index.

C. Karen Liu and Sumit Jain. 2012. A Quick Tutorial on Multibody Dynamics.

John E Lloyd. 2005. Fast implementation of Lemke's algorithm for rigid body contact simulation. In *Proceedings of the 2005 ieee international conference on robotics and automation.* IEEE, 4538–4543.

Brian Vincent Mirtich. 1996. *Impulse-based dynamic simulation of rigid body systems.* University of California, Berkeley.

David Stewart and Jeffrey C Trinkle. 2000. An implicit time-stepping scheme for rigid body dynamics with coulomb friction. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, Vol. 1. IEEE, 162–169.

Jie Tan, Kristin Siu, and C Karen Liu. 2003. Contact Handling for Articulated Rigid Bodies Using LCP. (2003).

Samuel Zapolsky and Evan Drumwright. 2017. Inverse dynamics with rigid contact and friction. *Autonomous Robots* 41, 4 (2017), 831–863.