Fall 2019 ME751 Final Project Report

University of Wisconsin-Madison

Simulation of Robotiq Linkage-Based Robotic Gripper Interacting with a Cylindrical Handle Using Chrono

Mike Hagenow

December 14, 2019

**Abstract**

Abstract placeholder.

All code and build instructions are provided in the following repository: <https://github.com/mhagenow01/751finalproject>

Contents

[1. Problem statement 4](#_Toc531620937)

[2. Solution description 4](#_Toc531620938)

[3. Overview of results. Demonstration of your project 4](#_Toc531620939)

[4. Deliverables: 4](#_Toc531620940)

[5. Conclusions and Future Work 5](#_Toc531620941)

[References 5](#_Toc531620942)

# General information

1. Home Department: Mechanical Engineering
2. Current status: PhD student
3. Mike Hagenow (Team Leader)
4. I release the ME751 Final Project code as open source and under a BSD3 license for unfettered use of it by any interested party. Note: Some of the meshes are from another GitHub repository, <https://github.com/a-price/robotiq_arg85_description>, which is also under a BSD license.

# Problem statement

We have developed a method for recognizing directions of slip when a user performs a demonstration using instrumented tongs. Our intent is then for the robot to perform a similar action. A major challenge is that slip is hard to control as it involves modulating forces to allow prescribed levels of motion. Grippers also commonly have only unilateral force sensing and control capabilities. Given the lack of available sensing, I think this is a great opportunity to use simulation to better understand the interaction of the gripper with the object to be able to think about control strategy without having to explicitly instrument the gripper further.

This implementation uses Project Chrono in order to develop a friction model and simulations involving a common linkage-driven robot gripper with a cylindrical handle (e.g. toaster oven, fridge). It is my hope that running simulations will give us insight as to how we should dictate our gripper force control loop to properly interact with the object.

# Solution description

Implementing this project required building skills using Chrono, doing kinematic modeling for systems with several bodies and geometric constraints, as well as implementing various friction models and exporting data. In order to achieve these goals, the project was broken up into four main milestones:

* Milestone 1: Complete a subset of the Chrono tutorials to get familiar with the interface
* Milestone 2: Model the kinematics of the gripper and perform a kinematic analysis that demonstrates the functionality of the linkage-based design of the gripper
* Milestone 3 – Set of a friction/contact problem against a rotating cylinder utilizing both the penalty and complementarity methods.
* Milestone 4 – Extract simulation data and compare with some experimental results to determine whether a grip-force based controls strategy for slip is feasible.

All of the code is structured to be run independently. Many of the files have redefinitions of the gripper geometry, but with various tweaks for the desired task or friction model.

# Overview of results. Demonstration of your project

## Chrono Tutorials

To gain competency with the Chrono platform, I first had to go through a few of the tutorials to understand the basic structure of how the simulations are constructed. Below is a list of the tutorials and demo files that I worked with and the main takeaways as far as code structure and functionality:

**Installation guides (Core Chrono Module + Linking to Chrono)** - these instructions and tutorials were well designed. Chrono makes good use of CMake and is configured to make linking libraries a very straightforward process.

**demo\_CH\_buildsystem** – This tutorial gives a good overview for creating ridid bodies, adding markers, and constraining them together with links. The slider crank also provides an example of a constant angular velocity motor, which is what I used for early testing.

**demo\_CH\_functions** – This tutorial was helpful for creating custom functions, which I used for the driving constraints, both for the kinematics of the gripper earlier on and to define a function to turn the cylinder. There are also some good prebuilt function classes for things like sine waves, but I actually thought just creating the ChFunction instantiation was easy enough!

**demo\_CH\_stream** – This was interesting to look over as far as filestreaming with Chrono. In my project, I actually ended up just using the std library filestreaming, but the Chrono functionality seems much more advanced and would be useful for more involved projects.

**demo\_IRR\_crank/my\_example** – These were used to get some basic idea of how to construct the irrlicht visualization.

**Reference manual** - In addition to sample code, I found it very useful to look over the reference manual. A few examples of sections I reviewed include ChFunctions, Collision Shapes, Motors, and Simulation System.

## Gripper Kinematic Simulation

The first main milestone of the project was to model the kinematics of the gripper in Chrono. This turned out to be a bit more challenging than originally anticipated. Since the gripper is often used in commercially available robots, there is a ROS package that had some of the meshes and dimensions for the bodies and joints. In ROS, these are expressed in an URDF (universal robot definition framework) file, so I needed to transcribe the geometry to the Chrono format. The major challenge was that not all of the geometry was defined. In ROS, only serial chain robots are supported. Since this gripper is a linkage-based parallel mechanism, the way it is displayed in ROS is a bit of a visual hack. As a result, there were no dimensions for the redundant joint that is attached to the gripper finger from the outside of the gripper.

For proper linkage kinematics, proper dimensioning is crucial as small errors can greatly affect the range of motion of the mechanism. The process was mostly trial and error to try and best determine geometry that lead to similar kinematics as are observed on the actual gripper.

When developing the kinematic demonstration, which opens and closes the gripper using a custom function imposed on the most proximal joint to the cuff, small errors in the geometry cause the redundant joints to build up error and try to rip apart the system. I also ran into problems where I needed to make sure I had to play with my integration step size to get proper simulations, likely because of this same error.

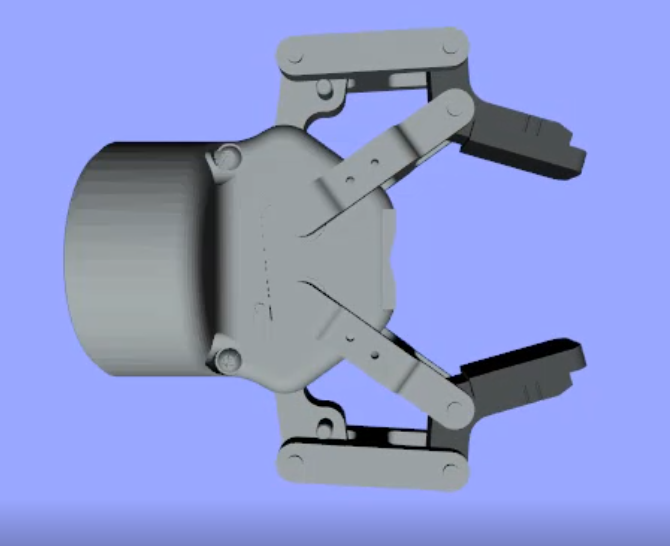


Figure 1: Gripper Kinematic Model in Chrono

Overall, I was successful in the goal to replicate some kinematics and could provide a demonstration where the gripper opens and closes in Chrono.

## Friction Testing

Before doing the cylinder grasp analysis, I created a friction testing playground for the complementarity method. The goal of this playground was to work on the collision geometry and to experiment with the contact properties.

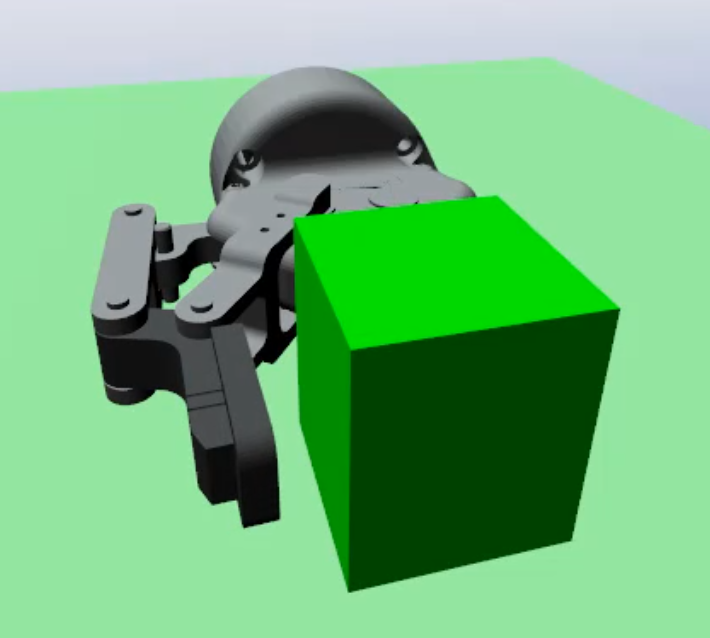


Figure 2: Gripper Pushing Block in Chrono

Since the visualization was created using stl meshes for the pieces of the gripper, these still needed to be collision geometry in order for the gripper to interact with other objects in the environment. I experimented with adding basic primitive collision shapes (in this case, all boxes) to various links of the gripper geometry. In the simulation, I didn’t add any collision properties to the interior link as it was difficult to get the orientation correct. As a result, you can see during the simulation that the block briefly penetrates the inner finger. In the future, I will try and get the collision models refined so that the gripper can undergo more complex interactions with the environment. In this case, only the collision with the gripper pads was needed for the cylinder grasping. This playground was also used for testing the contact and friction properties. Overall, it provided a good troubleshooting platform to find issues with the gripper model and friction.

## Gripper Cylinder Complementarity and Penalty Methods

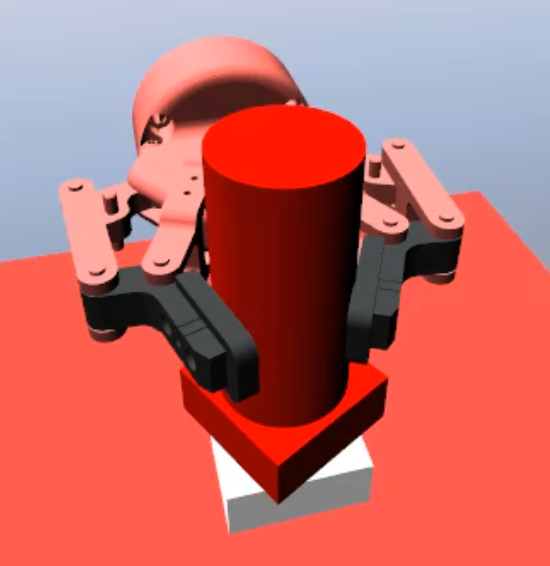


Figure 3: Cylinder Grasp in Chrono (Complementarity)

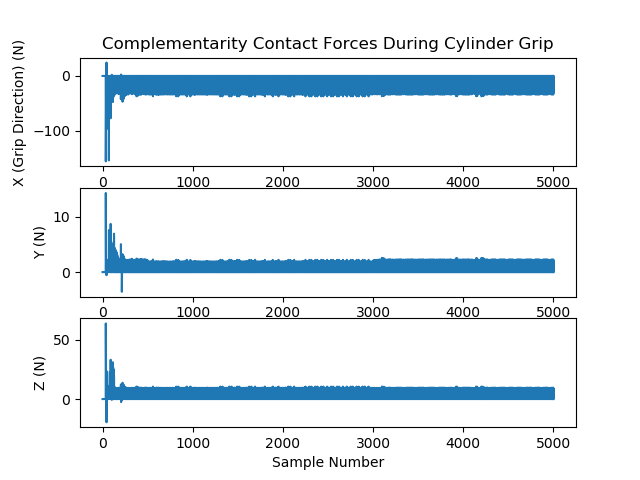
Once friction had been explored, two different simulations were developed: one that uses the complementarity method and one that uses a penalty-based method. As far as implementation in Chrono, switching between the methods was relatively trivial. I ran into some issues getting the contact to function properly and the cause ended up being that I was using old functions that did not appear to have the intended effect in the newest version of Chrono.

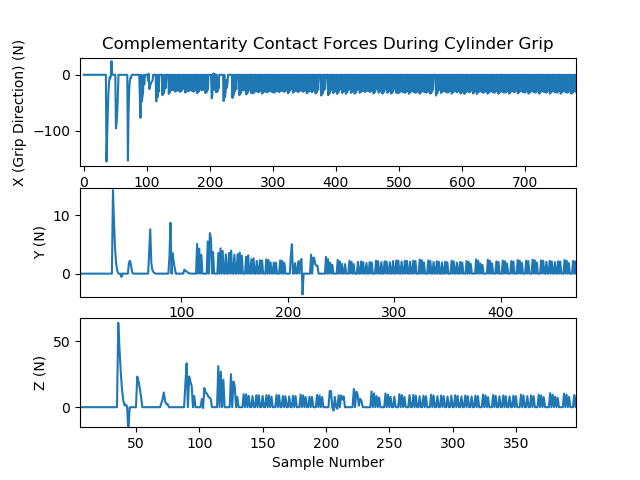
In order to create the gripping effect, I built a cylinder and stand using the ChEasy rigid body classes in Chrono. The largest challenge in building the cylinder grasp was determining the thickness and position of the cylinder. Compared to a purely parallel gripper, the linkage based gripper has two options for grasping: friction and form-fit grasping. If the object is located near the end of the the gripper fingertips, the design will naturally lead to a friction-type grasp [1]. For this project, I chose to do more of a form-fit grasp, which is more similar to how a person would naturally grasp a handle. The gripping motion is produced by a constant-torque motor on the outer-knuckle link. The cylinder is driven by a custom function that starts stationary and then turns at a fixed rate (to simulate grabbing and then slipping). In the penalty method, there is intentionally a fairly large amount of penetration in order to highlight the effect of this method. Overall, this section produced the intended outcome: a set of simulations employing the two friction models to simulate interaction between the gripper and a handle.

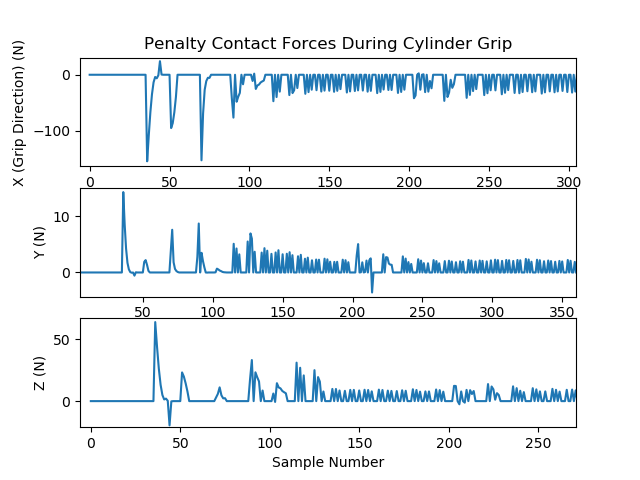
## Analysis and Potential Controls Strategy

Placeholder text

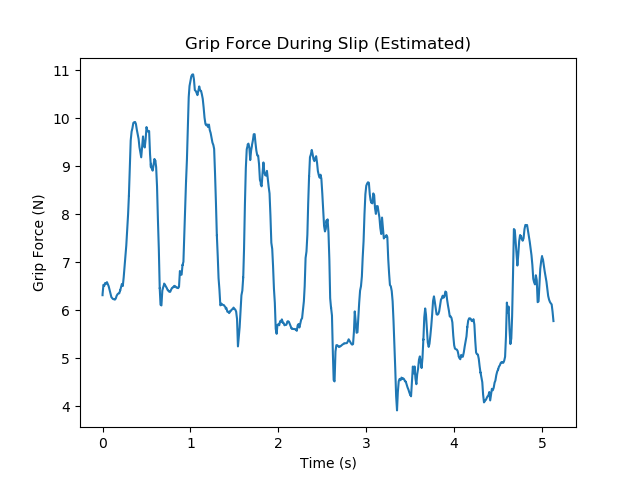
Friction vs form fit grasp







Explain here what you obtained, explained why the results are good/bad. This is the place where you talk about the outcomes of your Final Project effort. It is not the end of the world if your code doesn’t work as anticipated. Explain here how far you have made it.



Most often, you have a comparison against sequential code, perhaps via a scaling analysis. Make sure you include plots and/or tables to show your results.

# Deliverables:

## Files included in repository

Below is a list of the code that was created to support this project:

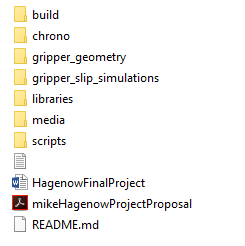
* /gripper\_slip\_simulations/gripper\_kinematics.cpp
* /gripper\_slip\_simulations/gripper\_friction\_test\_comp.cpp
* /gripper\_slip\_simulations/gripper\_complementarity\_friction.cpp
* /gripper\_slip\_simulations/gripper\_penalty\_friction.cpp
* /gripper\_slip\_simulations/gripper\_ complementarity\_friction\_plot\_generator.cpp
* /gripper\_slip\_simulations/gripper\_penalty\_friction\_plot\_generator.cpp
* /scripts/plot\_results.py
* /scripts/run\_complementarity.bash
* /scripts/run\_penalty.bash

## Building the files in the repository

The project builds against the Chrono C++ API. To build the gripper simulation files, you will need to follow the steps outlined here: <http://api.projectchrono.org/development/tutorial_install_project.html>

Additionally, since the project uses some relative paths, please follow the below steps to build successfully:

* Chrono should be placed in the root directory of the github repository (i.e., at the same level as the gripper\_slip\_simulations directory)
* Chrono is configured to build all files into the 'build' directory in the root
* gripper\_slip\_simulations contains a CMakeLists that can also be used through CMake to build the gripper simulation files. I did all testing with Visual Studio 2017. It was tested one two separate machines. When I tested, the executables were built to the build/Release/ directory (provided you compile with Release).

[](https://github.com/mhagenow01/751finalproject/blob/master/media/example_file_structure.PNG?raw=true)

## Running the Repository

Once the files have been build, this repository contains 4 exe files that can be run alone that include irrlicht visualization (*gripper\_kinematics*, *gripper\_friction\_test\_comp*, *gripper\_complementarity\_friction*, *gripper\_penalty\_friction*).

Note: If you do not want to build the files, I have also included a video in the media folder (*gripper\_simulation\_videos.mp4*) that shows all four of the simulations.

In order to create the plots in the report, there are bash scripts in the /scripts/ directory that allow you to run the penalty and complementarity methods and plot the contact forces from one of the gripper fingers: *run\_complementarity.bash*, *run\_penalty.bash*

These scripts will run the Chrono simulation, save the data to a file, and then plot the data in python. It is important these be run from the scripts directory as they rely on relative paths.

They also require the following python packages: numpy, matplotlib. I tested using Python 3.7.0 on windows and using the Git Bash terminal (Note: I had to run *alias python='winpty python.exe'* for the system to recognize the python command).

# Conclusions and Future Work

Simulations where the slip is in the “whats it called” zone

Using the closed loop to simulate missing gripper functionality

# References

[1] Robotiq 2F-85 Gripper Manual. <https://assets.robotiq.com/website-assets/support_documents/document/2F-85_2F-140_Instruction_Manual_e-Series_PDF_20190206.pdf>