# Project 4: Grasping

Junhua (Michael) Ma
junhua@berkeley.edu

Antony Zhao
ayzhao7761@berkeley.edu

***Abstract*—In this project, we developed and evaluated grasping techniques for an Allegro multi-fingered hand mounted on a Sawyer robotic arm, using the MuJoCo physics simulator. Our approach combines an inverse kinematics (IK) solver to compute joint configurations that achieve specified fingertip positions and orientations, with a grasp synthesis algorithm designed to generate force-closure grasps. The system demonstrates robust and effective grasping performance in simulation across a variety of object geometries. Additionally, we explored extensions such as grasping objects of different shapes and executing placement tasks into containers.**

## I. Methods

### A. Levenberg-Marquardt Inverse Kinematics

The goal of inverse kinematics is to determine joint configurations $q$ that achieve a desired end-effector position and orientation. In this project, we use the well-known Levenberg-Marquardt method used in practical robotics applications described by equation

$$q_{t+1} = q_t + ((J^T J + \lambda I)^{-1} J^T (x_d - x_t) \tag{1}$$

where $q_t$ is the joint state at time $t$, $J$ is the manipulator Jacobian, $\lambda$ is the damping coefficient, $x_d$ is the desired position, and $x_t$ is the current position. While the different between $q_t$ and $q_d$ is above a certain threshold, we repeatedly apply the equation to obtain updated joint state. We also clip $q_t$ at each step to ensure that the joint limits are obeyed.

To adapt the Levenberg–Marquardt algorithm for multiple fingers with desired positions and orientations, we compute both the position and orientation error for each fingertip. Correspondingly, we obtain the Jacobians for position and orientation with respect to the joint configuration. By stacking the individual position and orientation Jacobians into a single combined Jacobian matrix, and similarly concatenating the error vectors, we construct a unified system. This allows us to apply the standard Levenberg–Marquardt update to solve for the joint configuration q .

### B. Grasp Synthesis

The grasp synthesis algorithm looks to obtain grasping joint states for the Allegro hand that also ensures force closure, which means the hand can resist arbitrary external wrenches on the object to be a good grasp.

Initially, we use inverse kinematics to bring the palm slightly above the ball facing downward towards the ball, which sets the stage for synthesizing grasp. The grasp synthesis algorithm overall aims to minimize a joint state objective function. Before the fingers form contact with the ball, the joint state objective function is simply the sum of squared distances between the ball and each fingertip. After contacts are formed, the joint state objective function first obtain the position and normal vectors of contact between each finger and the ball. Then a discrete friction cone is computed for each contact normal, from which the graspability matrix G is computed.

Using G, we aim to achieve force closure by examining the necessary and sufficient conditions for force closure. For necessary condition, which examines if the origin of the wrench space lies inside the convex hull of the contact wrenches, we solve a simplified optimization problem formulated as

$$Q^+(\mathbf{G}) = \min_{\boldsymbol{\alpha} \geq \mathbf{0}} \|\mathbf{G}\boldsymbol{\alpha}\|_2$$

We define the necessary condition to be satisfied if the resulting $Q^+$ value is below a certain threshold. If the necessary condition is satisfied, we check the sufficient condition, which examines if the origin lies in a strict interior of the wrench space modeled as a ball with radius $r$, with a optimization problem formulated as

$$Q^-(\mathbf{G}) = \max_{k=1,\dots,K} (\min -r)$$

subject to:

$$r\mathbf{q}_k = \sum_{i=1}^{N} \alpha_i \mathbf{g}_i, \quad \sum_{i=1}^{N} \alpha_i = 1, \quad \alpha_i \geq 0, \quad r \geq 0$$

For $Q^- < 0$, the force closure should be guaranteed, while $Q^- = 0$ implies a marginal grasp that may or may not lead to successful grasp.

### C. Extensions

To further test and validate the synthesized grasps, we examine two extensions.

The first extension is to grasp and place the ball into a container. With a good grasp, this task should be trivial as the ball should remain in hand during transit until the hand release the grasp and drop the ball at the desired location.

The second extension grasp a cube instead of a ball. We defined a cube with similar properties like mass and size as the ball, and replaced the ball in simulation with the cube. For simplicity, we used the same distance function to compute the distance between each fingertip and the surface, which is a very rough estimation, but from the simulation we still obtained successful grasp. Additionally, we also obtained success with placing the cube into a container.

## II. Experimental Results

### A. Inverse Kinematics Solver

For a given position and orientation for each of the 4 fingers as well as the palm, we can obtain joint state $q$ with inverse kinematics, with an example result shown in Fig. 1.

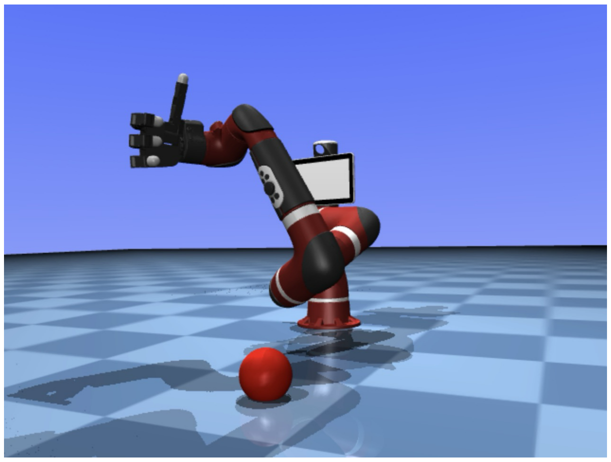

Fig. 1. Example joint state from IK solver given target finger and palm positions and orientations.

To grasp the ball with inverse kinematics, we generated 4 grasp points for each finger along the equator of the ball that are 50 degrees apart, and simply used the same downward orientation as target orientation for each fingertip. The result obtained is shown Fig. 2, which shows poor performance of the IK solver to generate good grasp without good target orientations. Generally, we tested a few target positions and orientations, and got high error and tend to fail to converge even over 10k steps.

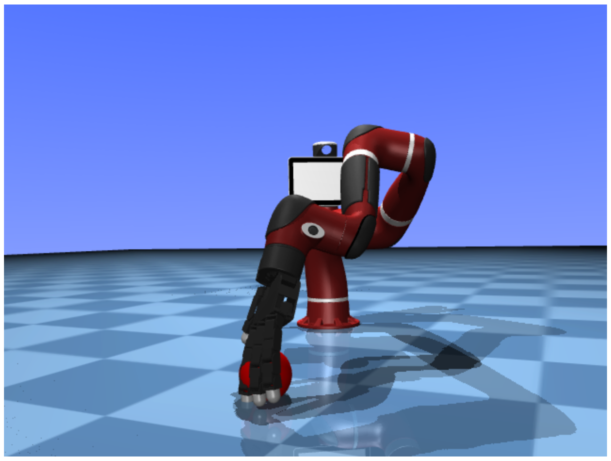

Fig. 2. Result of grasping with IK solver given target positions along the equator and downward orientation.

Additionally, we used the same inverse kinematics setup to prepare for grasp synthesis by bringing the palm down above the ball. We simply set desired palm position to be over the ball position by a certain offset and desired palm orientation to be downward, and result is shown in Fig. 3. In this case, for a more simple and exact target configuration, the IK solver is very effective.

TABLE I
GRASP SYNTHESIS HEIGHT COMPARISON

| **Starting Height** | $Q^+$ | $Q^-$ | **Success** |
|---|---|---|---|
| 0.12 | N/A | N/A | No |
| 0.13 | 7.2e-6 | -0.006 | Yes |
| 0.14 | 7.6e-6 | -0.007 | Yes |
| 0.16 | 9.27e-7 | -0.001 | Yes |
| 0.18 | N/A | N/A | No |

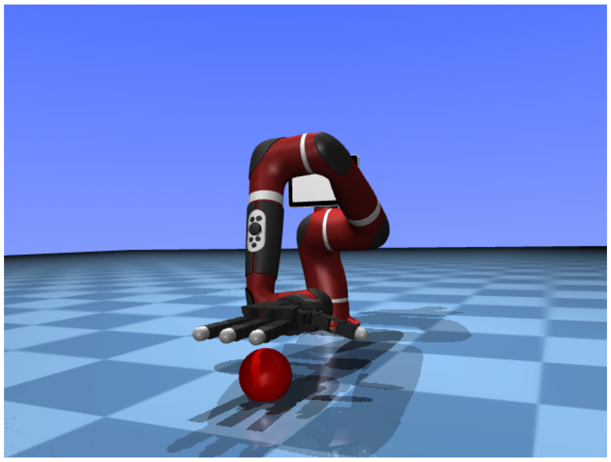

Fig. 3. Result of using IK solver to lower the hand for grasp synthesis.

### B. Grasp Synthesis

Overall, grasp synthesis algorithm generates good grasping joint states that leads to successful grasps during simulation. An example grasp generated through grasp synthesis is shown in Fig. 4, which looks much better than the grasp from IK solver. A typical loss curve obtained during grasp synthesis is shown in Fig. 5, which gradually converges.

Additionally, we experimented with different starting heights of the palm above the ball, and the results are summarized in Table. 1. Generally, we found that successful grasp synthesis is highly dependent on the starting configurations of the hand. Of course, if the hand is too far away, the fingers can never reach the ball. However, even small height adjustments can lead to failures where the sufficient conditions optimization solver may fail. In terms of the values of $Q^+$ and $Q^-$, generally we are able to obtain $Q^+$ values low enough to meet the necessary condition, and more negative $Q^-$ leads to better grasp, as $Q^-$ very close to 0 leads to grasps closer to being marginal. From the simulation we confirmed that having $Q^- < 0$ leads to good grasps with the ball firmly in hand without any sliding.

### C. Extensions

We obtained similarly successful results for some extension tasks, with transferring ball to a container shown in Fig. 6,

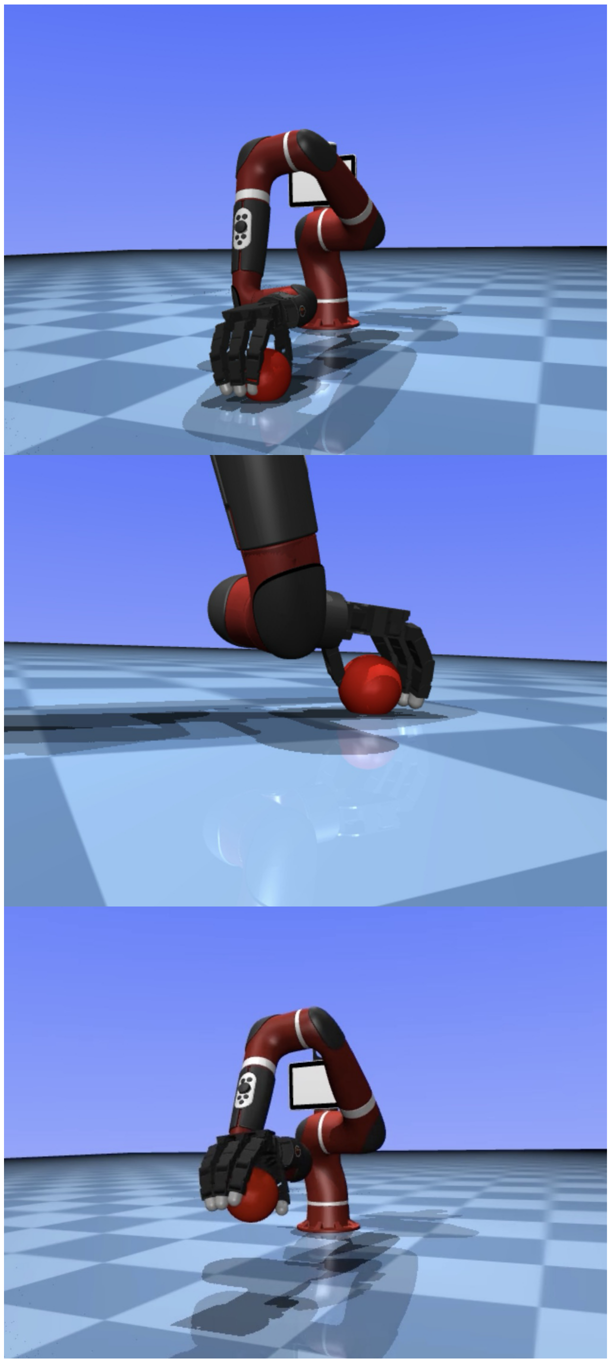
Fig. 4. Result of grasp synthesis. **Top:** front view of grasp. **Middle:** side view of grasp. **Bottom:** front view of grasp lifting the ball.

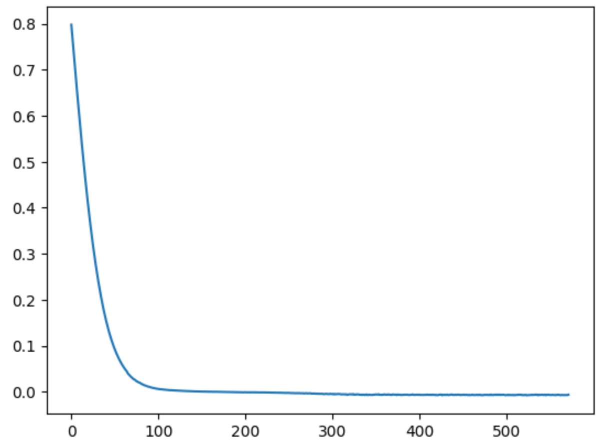

Fig. 5. Typical loss curve obtained during grasp synthesis. Vertical axis is loss, horizontal axis is iteration.

picking up a cube shown in Fig. 7, and transferring cube to a container shown in Fig. 8. The grasp is generalizable to similar shapes like cubes even without an accurate finger-to-surface distance function. A potential real world use case of grasping is demonstrated by grasping object and transferring them to a designated location.

## III. Discussion

Overall, the IK solver and grasp synthesis complemented each other well. The IK solver excels at producing larger joint movements, enabling coordinated motion of both the arm and hand, though this often comes at the cost of precision and guaranteed force closure. In contrast, the grasp synthesis method focuses on hand-specific joints and is more effective for generating precise, force-closed grasps. The synergy is especially handy for transferring object to container tasks, where IK is utilized to direct the hand to the target position, and force closure ensures the object doesn't slip out of the hand during movement. Moreover, the grasp synthesis is pretty effective overall as it can also generalize to object of different shapes, although its generality towards other object attributes like mass, size, and so on can be explore further in future works.

However, there are also a few aspects about this project that are challenging or confusing. One aspect is the use of sufficient condition to ensure force closure in synthesized grasp. From our testing, even without the use of the sufficient condition and simply exiting grasp synthesis upon contact, the resulting grasp can still demonstrate all the tasks successfully. Therefore, we are unable to see the effect of sufficient conditions very well. The other aspect is the overall difficulty with implementing sufficient condition, as the solver frequently results in no solutions even with small changes to the starting palm configuration. Since the sufficient condition function relies on the graspability matrix, which requires the correct contact positions, normals, and friction cones, it's hard to debug since any errors in previous parts could contribute to the error in

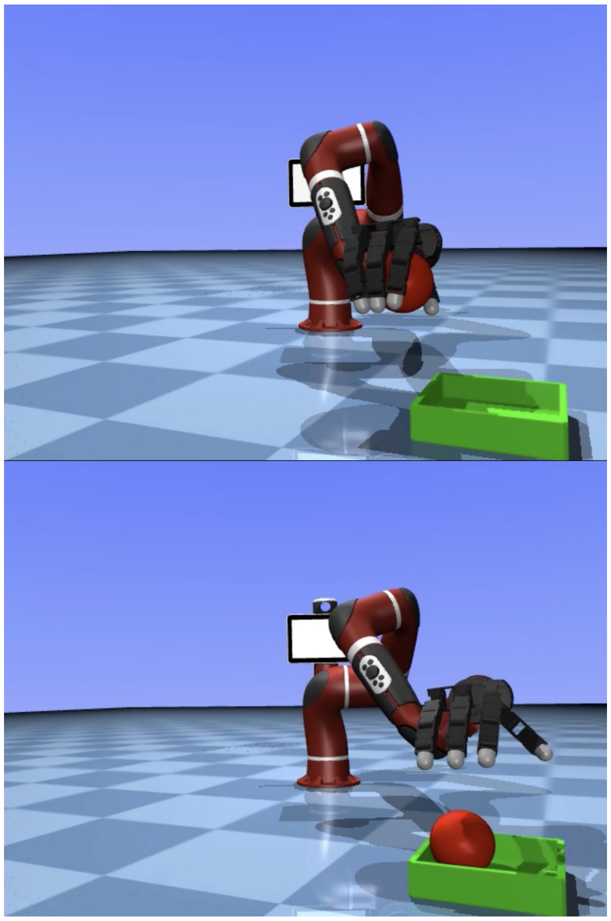

Fig. 6. Result of grasping and transferring ball to container.

sufficient condition function. Finally, we encountered some difficulties with MuJoCo simulation and DM Control library, such as not able to plot the contact force due to some strange errors.

## References

[1] R. M. Murray, Z. Li, and S. S. Sastry, A Mathematical Introduction to Robotic Manipulation, 1st ed. CRC Press, 2017. doi: 10.1201/9781315136370.

## IV. Appendix

https://github.com/antony-zhao/106b-lab

Demo Videos: https://drive.google.com/drive/folders/1vqGix–TEsPHofs3wh4V6n2eigXh˙Z2v?usp=sharing

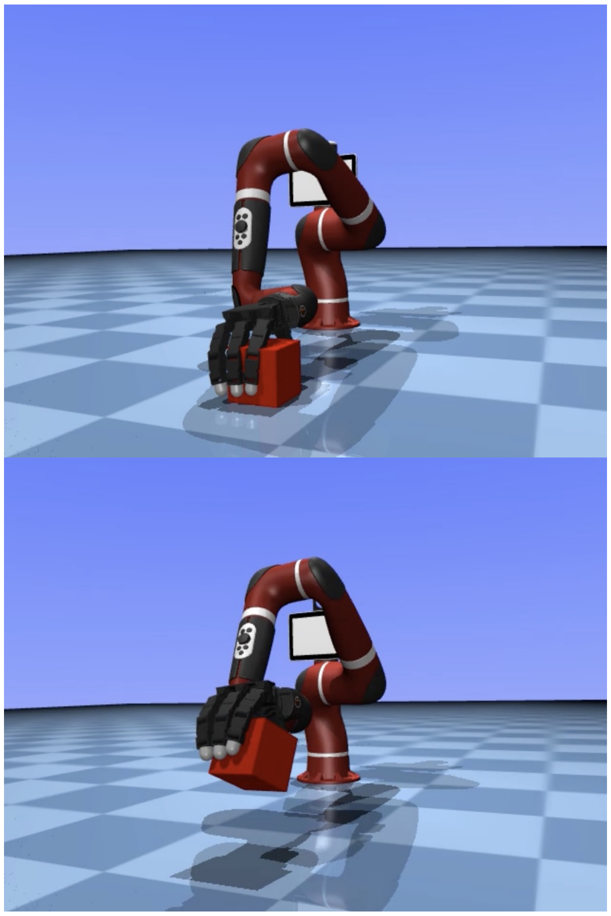

Fig. 7. Result of grasping a cube.

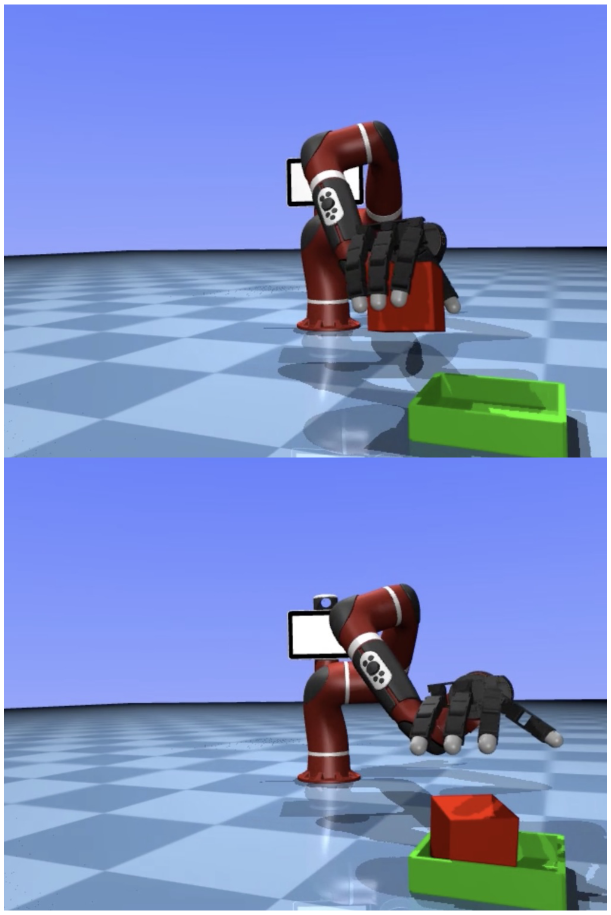

Fig. 8. Result of grasping and transferring cube to a container.