EE125 Lab 2: AdeptSix 300 Lab

Inverse Kinematics and Singularities

Working inverse kinematics and Jacobian programs due 11/22/05

1 Introduction

In this lab we will explore the inverse kinematics problem and the singularities for the AdeptSix 300 robot. In the inverse kinematics problem, the joint angles are determined for a given end effector pose. For some robots, the inverse kinematics can be solved in closed form. The AdeptSix is a robot of this type. In this lab, we use the subproblems presented in class along with geometric reasoning to solve the inverse kinematics and use the solutions to perform a pick and place task.

Often, multiple sets of joint angles give the same end effector pose. In the first part of the lab, we explore the multiple solutions for the AdeptSix robot.

Before coming to the lab, you are required to write a program to solve the inverse kinematics problem for the AdeptSix robot. In the second part of the lab, you test your inverse kinematics algorithm against the algorithm provided with the AdeptSix software.

In the third part of the lab, you bypass the inverse kinematic solutions provided by the AdeptSix software and use your programs to perform a pick and place task.

In the last part of the lab, you will explore what happens to the AdeptSix near singularities. You will need to calculate the manipulator Jacobian and evaluate it at singular location. You will identify an interior singularity, evaluate the Jacobian at that location, and calculate and observe what internal motion (i.e. a motion of the joints producing no motion of the endeffector) is possible and what end-effector velocities cannot be achieved at the singularity.

2 Pre-Lab Preparation

You need to prepare for the lab by writing programs to solve the inverse kinematics of the AdeptSix and for calculating the jacobian. In the inverse kinematics program, given an end effector pose, you need to calculate the multiple set of joint angles. Also, you need to identify an interior singularity of the AdeptSix before coming to lab.

3 Inverse Kinematics Programs

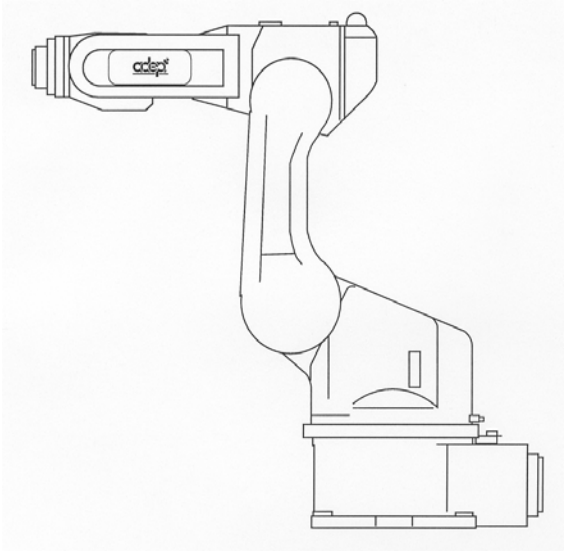
Your programs need to calculate the sets of joint angles given an end effector configuration, $g \in SE(3)$. Create subroutines for each of the subproblems and use the subroutines to solve the inverse kinematics. Write the program in *Matlab* so that you can calculate the inverse kinematics while running the robot.

The relationship between the tool frame and the spatial frame for this lab is shown in Figure 1 for the zero configuration. The dimensions of the AdeptSix are given as well as the positive directions for the revolute twists.

4 AdeptSix Robot

It is important to remember that the AdeptSix does not have any obstacle-avoidance capabilities, so it can easily hit itself or you.

Keep a hand on the **Emergency Stop Button** on the joystick when moving the AdeptSix. The units for this lab are radians and milimeters.



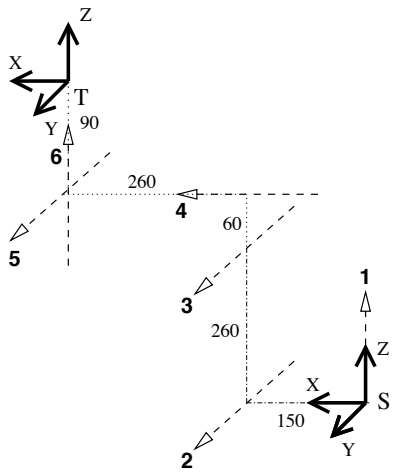


Figure 1: Zero Configuration for the AdeptSix Robot

5 Robot Control

5.1 Manual Control Pendant (MCP)

The MCP has quite a bit of functionality. From the the MCP you can completely control and even program the robot. To enable power, simply push the *comp/pow* button. To disable power, push the *dis/pow*. Once power is enabled, you can push the *MAN/HALT* button to control the robot from the MCP. If you wish to control the robot from the computer, you need to push the *COMP/POW*.

Movement can be controlled in one of two ways, either in joint space or cartesian space. You can how you want to move the robot by hitting the ***local button, and then chosing one of the axis buttons *(X/1, Y/2, Z/3, Rx/4, Ry/5, Rz/6)* The movement initiates once you touch either the positive (+) or negative (-) speed sensor which allows you to vary the speed of the motion. In joint space, you control each of the six joints individually. In world space, you control the motion in the X,Y and Z directions of the tool frame with the first three buttons *(X/1, Y/2, Z/3)*. In world space, you can also control the orientation of the tool frame. For example, selecting button *Rx/4*, allows you to rotate the tool frame around the x axis, without translating the tool frame.

5.2 Computer Control (V+)

To control the robot from the computer, we interface with an Operating styem called V+. This OS allows us to use a command line interface for controlling the robot.

Because of time limitations, we will not be able to implement a complete pick and place program using the V+ language. However, I do want you to outline the necessary steps for a pick and place program.

Useful commands:

All V+ Command line commands must be preceded by 'do'.

- 'calibrate' or 'cal' : must be done first, it determines the current robot orientation
- 'where' : returns the world and joint coordinates
- 'enable power' : enables power
- do ready : moves the robot to the ready position
- do move #ppoint (0,-pi/2,pi,0,-pi/2,0) : moves the motors to what we call the home position.
- do move trans(X,Y,Z,Yaw,Pitch,Roll) : moves the end effector to position X,Y,Z with orientation Yaw,Pitch,Roll.
- do moves ... :moves in a straight line in tool space.
- here varname : saves the current location to varname
- do above,do below,do lefty, do righty : The next configuration will be achieved with the elbow above

6 Exercises

6.1 Multiple Inverse Kinematic Solutions

In the lab you will be shown a tool frame position and orientation for which there are four inverse kinematic solutions. You will be asked to observe these and answer the following questions.

- How are configurations 1 and 2 related?
- How are configurations 3 and 4 related?

- What is the relationship between the pairs (1 and 2) and (3 and 4)?
- Answer in terms of the branching of solutions in your inverse kinematics solutions which arise from the output of your subproblem solutions.
- List some ways that you can use multiple solutions to your advantage.
- How many solutions are theoretically possible for this tool frame configuration and why do we only show you 4 solutions?

6.2 Checking your Inverse Kinematic Solutions

In the home configuration of the AdeptSix, the motor angles are [0,-pi/2,pi,0,-pi/2,0]. Make sure to take these offsets into consideration when comparing your kinematic solutions to those of the AdeptSix.

In order to check that your end effector frame is the same as the input to your inverse kinematics program, you need to correctly interpret the coordinates given by the AdeptSix *where* command. The *where* command returns the XYZ position of the tool frame and the orientation of the tool frame with respect to the spatial frame. The orientation is given in terms of three angles, y, p, and r. These three angles correspond to rotating the tool frame initially coincident with the spatial frame about the fixed Y axis by p radians followed by a rotation about the fixed Z axis by y radians. The last rotation is a rotation about the new Z axis. Since composition for fixed rotations are on the left and body rotations are on the right, the rotation matrix taking points in the tool frame to the spatial frame is $R_z(y)R_y(p)R_z(r)$. Notice that the end effector is pointing straight down if $y = 0$ rad, $p = pi$ rad, and $r = 0$ rad.

Use your program to calculate the joint angles for the following tool frame orientation.

$$\begin{bmatrix} -0.4317 & -0.5344 & 0.7266 & 325.3 \\ 0.5152 & -0.8073 & -0.2877 & 92.3 \\ 0.7404 & 0.2502 & 0.6239 & 508.6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \in SE(3)$$

The joint limits are $(\pm 2.967, -0.7854 + 2.618, -3.316 + 1.222, \pm\infty, -0.7854 + 3.9270, \pm\infty)$ rad.

- Provide a listing of your solutions.
- Do you have the maximum number of theoretical solutions? If not, what are the reasons(joint limits, no solution for one of the subproblems,...)?
- For one of the feasible solutions, command the motors to the new position. You can use the *move* command. Use the *where* command to record the X,Y,Z,y,p,r coordinates of the tool frame. Calculate the rotation matrix corresponding to these three angles and compare it to your desired orientation. How close is the tool frame to the desired location? Provide the experimental location of the tool frame in terms of the rotation matrix and origin position and the X,Y,Z,y,p,r coordinates.

6.3 Programming a pick and place task

Write a program to pick up an object at an arbitrary location by specifying the motor positions which you get from your inverse kinematics program. A bin will be placed within the workspace of the AdeptSix, and you will be given its coordinates relative to the base coordinate system shown in Figure 1. The part will be placed arbitrarily in the workspace of the AdeptSix and you need to pick up the part and place it in the bin by *only* specifying motor positions. You need to use your inverse kinematic programs to determine the sets of joint angles for several tool frame positions along the planned path.

- How did you use your multiple solutions to the inverse kinematics to perform the task?
- Were certain solutions "better" than others for the pick and place task?
- How many intermediate points did you use to perform the task?
- What was the path you followed?

7 Manipulator Jacobian and Singularities

For the last exercise of this lab, you need a matlab function called *jacobian.m* which takes as input a joint configuration and calculates the jacobian. Write this program before coming to lab. You should calculate the twists of the AdeptSix in the zero configuration before coming to lab.

Also, identify an interior singularity of the AdeptSix before coming to lab. You will need this for the exercise.

7.1 Fun with Singularities

Place the AdeptSix at a boundary singularity with $\theta 3 = -\pi/2$. Move the robot in the -Z direction with the joystick. What happens?

Place the robot at one of its interior singularities. Evaluate the jacobian at this configuration using your matlab function. Check to see if the jacobian is singular. Use the matlab function "null", which finds the nullspace of a matrix, to calculate what combination of joint movements produces no endeffector motion. Make the AdeptSix perform this "internal movement." (Note: you may need to set the "eps" variable in matlab to about .001 so that matlab finds a nullspace. The eps variable is used as a tolerance by the "null" function to decide if a matrix is singular.) Use the "null" function again to determine what set of endeffector velocities cannot be achieved. Try to move the AdeptSix with one of these unachievable endeffector velocities.

8 Write-up

The writeup for this lab is:

1. Answer the questions/requests in the Multiple Solutions section of the lab.
2. Provide a listing of your program to calculate the inverse kinematics. Provide the list of solutions to the given tool frame configuration and answer the questions/requests in the section.
3. Provide a listing of your pick and place program and answer the questions in the section.
4. Your explanation for why the AdeptSix cannot move away from the boundary singularity, the matlab program for calculating the jacobian, the interior singularity you found, the internal motion possible at the singularity, and the set of unachievable endeffector velocities at the singularity.