Advanced Robotic Manipulation

Handout

CS327A (Spring 2017)

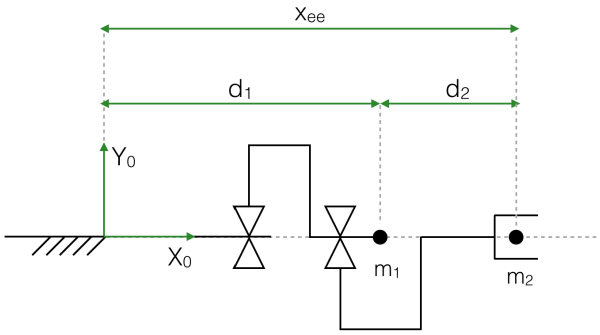
Solution Set #2

Problem 1 - Redundant robot control

The goal of this problem is to familiarize you with the control of a robot that is redundant with respect to the task it is required to perform.

(a) Weighted instantaneous inverse kinematic solutions

Consider the PP robot shown below.



The masses for link 1 and link 2 are $m_1 = 10\,\text{kg}$ and $m_2 = 5\,\text{kg}$ respectively. Consider the task of moving along the X_0 axis. Answer the following questions:

i. Is the robot redundant with respect to the task? How many degrees of freedom does the task require? How many degrees of freedom does the robot have?

Solution: The task requires 1 degrees of freedom (DOF) but the robot has 2. So it is indeed redundant with respect to the task.

ii. Find A, the generalized co-ordinates mass matrix of the robot.

Solution: To find the mass matrix of the robot, we look its total kinetic energy in terms of joint velocities.

$$\begin{aligned} KE = KE_1 + KE_2 &= \frac{1}{2} m_1 \dot{d}_1^2 + \frac{1}{2} m_2 (\dot{d}_1 + \dot{d}_2)^2 \\ &= \frac{1}{2} \left((m_1 + m_2) \dot{d}_1^2 + 2 m_2 \dot{d}_1 \dot{d}_2 + m_2 \dot{d}_2^2 \right) \\ &= \frac{1}{2} \begin{bmatrix} \dot{d}_1 & \dot{d}_2 \end{bmatrix} \begin{bmatrix} m_1 + m_2 & m_2 \\ m_2 & m_2 \end{bmatrix} \begin{bmatrix} \dot{d}_1 \\ \dot{d}_2 \end{bmatrix} \end{aligned}$$

Thus, we have

$$A = \begin{bmatrix} m_1 + m_2 & m_2 \\ m_2 & m_2 \end{bmatrix}$$

iii. Let x_{ee} be the displacement of the end-effector along the X_0 axis. Find the task Jacobian J such that

$$\dot{x}_{ee} = J\dot{q}, \;\; q = \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}$$

Solution:

$$x_{ee} = d_1 + d_2$$

So, we have

$$J = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

iv. Find J^+ and $[I - J^+J]$. Then express the family of "instantaneous inverse kinematic" solutions in terms of δx_{ee} and δq_0:

$$\delta q = J^+ \delta x_{ee} + [I - J^+J]\delta q_0$$

Solution:

$$J^+ = J^T \left(JJ^T\right)^{-1} = \frac{1}{2}\begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$I - J^+J = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

The family of instantaneous inverse kinematic solutions for this choice of right inverse of J is given by

$$\delta q = \frac{1}{2}\begin{bmatrix} 1 \\ 1 \end{bmatrix}\delta x_{ee} + \frac{1}{2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\delta q_0$$

v. Find $J^\#$ and $[I - J^\#J]$ where

$$J^\# = W^{-1}J^T(JW^{-1}J^T)^{-1}, \;\; W = \begin{bmatrix} w_1 & 0 \\ 0 & w_2 \end{bmatrix}.$$

Then express the family of weighted "instantaneous inverse kinematic" solutions in terms of δx_{ee} and δq_0:

$$\delta q = J^\# \delta x_{ee} + [I - J^\#J]\delta q_0$$

Solution:

$$J^\# = W^{-1}J^T \left(JW^{-1}J^T\right)^{-1} = \frac{1}{w_1 + w_2}\begin{bmatrix} w_2 \\ w_1 \end{bmatrix}$$

$$I - J^\#J = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \frac{1}{w_1 + w_2}\begin{bmatrix} w_2 & w_2 \\ w_1 & w_1 \end{bmatrix} = \frac{1}{w_1 + w_2}\begin{bmatrix} w_1 & -w_2 \\ -w_1 & w_2 \end{bmatrix}$$

The family of instantaneous inverse kinematic solutions for this choice of right inverse of J is given by

$$\delta q = \frac{1}{w_1 + w_2}\begin{bmatrix} w_2 \\ w_1 \end{bmatrix}\delta x_{ee} + \frac{1}{w_1 + w_2}\begin{bmatrix} w_1 & -w_2 \\ -w_1 & w_2 \end{bmatrix}\delta q_0$$

vi. Given a particular δx_{ee} and $\delta q_0 = 0$, qualitatively compare the solutions δq obtained with J^+ and $J^\#$ when the latter is calculated with $W = A$.

Solution: We first obtain the inertia weighted pseudo-inverse,

$$\bar{J} = A^{-1}J^T\left(JA^{-1}J^T\right)^{-1} = \begin{bmatrix}0\\1\end{bmatrix}$$

The instantaneous inverse kinematic solution for this choice of right inverse of J when $\delta q_0 = 0$ is given by

$$\delta q = \begin{bmatrix}0\\1\end{bmatrix}\delta x_{ee}$$

Comparing this to the solution using pseudo-inverse,

$$\delta q = \frac{1}{2}\begin{bmatrix}1\\1\end{bmatrix}\delta x_{ee}$$

we notice that the pseudo-inverse causes the motion to be equally distributed over the two joints where as the A-weighted generalized inverse causes only the second joint to be moved. This has to do with the fact that moving the end-effector requires motion δx_{ee} on the mass m_2 irrespective of whether or not mass m_1 is moved. More generally, using the A-weighted generalized inverse distributes the required end-effector motion on to the joints such that lightest links are moved the most.

(b) Puma end-effector position control

Now let us investigate the control of task-redundant robots through joint torques. The PUMA(560) arm is a 6-DOF manipulator you are probably familiar with through CS223A and the lecture slides. With respect to the 3-DOF positioning task at the end-effector, the robot is redundant.

i. After following the instructions to upgrade the dependencies above, compile and run the source file `hw2/p1-main.cpp`. Note the difference between this and the robot behavior in homework 1.

```
$ cd bin
$ pushd ../build && cmake -DCMAKE_BUILD_TYPE=Release .. && make && popd
$ ./hw2-p1 1
```

The robot should be in free fall under gravity when you run the starter code. This is because we are using the dynamics simulation framework for this homework, but no control torques are applied by default.

Solution: The solution code for all three parts of this problem are available under `cs327a/hw2_sol` in `p1-main-sol.cpp`. To compile and run it, from within the `cs327a/bin` folder, run

```
$ git stash
$ git pull --rebase
$ git stash pop
$ pushd ../build && cmake -DCMAKE_BUILD_TYPE=Release .. && make && popd
$ ./hw2-p1-sol 1
```

ii. Implement the following PD controller in the area marked as "FILL ME IN". This controller is designed to hold the end effector tip position stationary while providing some joint damping to stabilize the simulation. Note that we do not compensate for Coriolis or centrifugal forces as in practice, they tend to destabilize the controller due to estimation errors.

$$\Gamma = J_v^T(\Lambda_v(-k_{px}(x - x_d) - k_{vx}\dot{x}) + p) + A(-k_{vj}\dot{q})$$

where J_v is the linear velocity Jacobian at the end-effector tip, x is the end-effector tip position and x_d is the end-effector tip desired position, all calculated with respect to the base frame. Use $\mathbf{x_d} = [\mathbf{-0.15\ \ 0.81\ \ 0.58}]^\mathbf{T}$, $\mathbf{k_{px} = 50}$, $\mathbf{k_{vx} = 20}$, $\mathbf{k_{vj} = 20}$. These values are already set for you in the variables `ee_des_pos`, `kpx`, `kvx` and `kvj` respectively in the starter code.

Set your joint control torques `tau` as per the above control law in case **PART1** of the switch block. Run your code with a "1" argument to the executable as shown below

```
$ pushd ../build && cmake -DCMAKE_BUILD_TYPE=Release .. && make && popd
$ ./hw2-p1 1
```

What happens if you remove the joint damping? That is, what is the difference in observed behavior between the above controller and the one below?

$$\Gamma = J_v^T(\Lambda_v(-k_{px}(x - x_d) - k_{vx}\dot{x}) + p)$$

Solution: Without joint space damping, the robot's motions in the null-space are visibly faster. As a result there might be some motion at the end-effector while there is null space motion, primarily due to centrifugal/Coriolis forces. But the end-effector PD controller is able to compensate for it to some extent.

iii. Now, let us try performing some null-space (or self-) motions. First, lets use the following pseudo-inverse based null-space torque controller. We control only the second joint position while damping the motion on other joints and compensating for gravity.

$$\Gamma = J_v^T(\Lambda_v(-k_{px}(x - x_d) - k_{vx}\dot{x}) + p) + [I - J_v^T J_v^{+T}](A(-k_{pj}(q - q_d) - k_{vj}\dot{q}) + g)$$

where

$$q_d = \begin{bmatrix} q_1 \\ -\frac{\pi}{8} + \frac{\pi}{8}\sin\frac{2\pi t}{10} \\ q_3 \\ q_4 \\ q_5 \\ q_6 \end{bmatrix}$$

and t is time in seconds. x_d is the same as the previous part. Use $k_{pj} = 50$ (available as variable `kpj`).

Set your joint control torques `tau` as per the above control law in case **PART2** of the switch block. Run your code with a "2" argument to the executable as shown below

```
$ pushd ../build && cmake -DCMAKE_BUILD_TYPE=Release .. && make && popd
$ ./hw2-p1 2
```

Does the end effector position remain stationary?

Solution: No, the end effector does not remain stationary even if the end-effector gains are tuned up. As the pseudo-inverse based null-space torque projection does not decouple the self-motion forces from the end-effector acceleration, we see a lot of motion at the end-effector. The joint space damping k_{vj} can be increased to some extent to stabilize the system, but we will still get poor performance in self-motion tracking.

iv. Finally, let us try to perform the same null-space motion, but with the *dynamically consistent generalized Jacobian inverse*.

$$\Gamma = J_v^T(\Lambda_v(-k_{px}(x - x_d) - k_{vx}\dot{x}) + p) + [I - J_v^T\bar{J}_v^T](A(-k_{pj}(q - q_d) - k_{vj}\dot{q}) + g)$$

where x_d and q_d are the same as the previous parts.

Set your joint control torques `tau` as per the above control law in case **PART3** of the switch block. Run your code with a "3" argument to the executable as shown below

```
$ pushd ../build && cmake -DCMAKE_BUILD_TYPE=Release .. && make && popd
$ ./hw2-p1 3
```

Now, does the end effector position remain stationary? Briefly explain why the dynamically consistent generalized inverse works but the pseudo-inverse does not.

Solution: Indeed, with the *dynamically consistent generalized inverse*, the correct null-space torque projection matrix $(I - J_v^T\bar{J}_v^T)$ is obtained and the robot is able to perform the required self-motion without disturbing the end-effector position. As a result, the torques applied by the self-motion PD controller are projected first into a subspace of joint torques that do not produce any acceleration of the end-effector. This is by design of the dynamically generalized inverse, which is the only generalized inverse with this property.

Problem 2 - Operational space trajectory tracking

In this problem, you will implement a full trajectory tracking controller on the KUKA-IIWA simulation model. Unlike the position-based controller you had implemented in homework 1, you will need to control the robot through joint torques this time around.

The desired end-effector trajectory is identical to the one you implemented in homework 1:

$$\begin{aligned} x_d &= 0 \\ y_d &= 0.5 + 0.1\cos\frac{2\pi t}{5} \\ z_d &= 0.65 - 0.05\cos\frac{4\pi t}{5} \end{aligned}$$

where t is time in seconds. The desired orientation trajectory is represented in Euler parameters as

$\lambda_d = (\lambda_{0,d}, \lambda_{1,d}, \lambda_{2,d}, \lambda_{3,d})$ where

$$\begin{aligned}
\lambda_{0,d} &= \frac{1}{\sqrt{2}} \sin\left(\frac{\pi}{4}\cos\frac{2\pi t}{5}\right) \\
\lambda_{1,d} &= \frac{1}{\sqrt{2}} \cos\left(\frac{\pi}{4}\cos\frac{2\pi t}{5}\right) \\
\lambda_{2,d} &= \frac{1}{\sqrt{2}} \sin\left(\frac{\pi}{4}\cos\frac{2\pi t}{5}\right) \\
\lambda_{3,d} &= \frac{1}{\sqrt{2}} \cos\left(\frac{\pi}{4}\cos\frac{2\pi t}{5}\right)
\end{aligned}$$

(a) Desired operational space acceleration

Find $\ddot{\mathbf{x}}_d$, where $\mathbf{x}_d = [x_d\ y_d\ z_d\ \lambda_{0,d}\ \lambda_{1,d}\ \lambda_{2,d}\ \lambda_{3,d}]^T$ is the desired acceleration in operational space coordinates.

Solution:

$$\begin{aligned}
\ddot{x}_d &= 0 \\
\ddot{y}_d &= -\frac{2\pi^2}{125}\cos\frac{2\pi t}{5} \\
\ddot{z}_d &= \frac{4\pi^2}{125}\cos\frac{4\pi t}{5} \\
\ddot{\lambda}_{0,d} &= -\frac{\pi^4}{100\sqrt{2}}\sin\left(\frac{\pi}{4}\cos\left(\frac{2\pi t}{5}\right)\right)\left(\sin\left(\frac{2\pi t}{5}\right)\right)^2 - \frac{\pi^3}{25\sqrt{2}}\cos\left(\frac{\pi}{4}\cos\left(\frac{2\pi t}{5}\right)\right)\cos\left(\frac{2\pi t}{5}\right) \\
\ddot{\lambda}_{1,d} &= -\frac{\pi^4}{100\sqrt{2}}\cos\left(\frac{\pi}{4}\cos\left(\frac{2\pi t}{5}\right)\right)\left(\sin\left(\frac{2\pi t}{5}\right)\right)^2 + \frac{\pi^3}{25\sqrt{2}}\sin\left(\frac{\pi}{4}\cos\left(\frac{2\pi t}{5}\right)\right)\cos\left(\frac{2\pi t}{5}\right) \\
\ddot{\lambda}_{2,d} &= -\frac{\pi^4}{100\sqrt{2}}\sin\left(\frac{\pi}{4}\cos\left(\frac{2\pi t}{5}\right)\right)\left(\sin\left(\frac{2\pi t}{5}\right)\right)^2 - \frac{\pi^3}{25\sqrt{2}}\cos\left(\frac{\pi}{4}\cos\left(\frac{2\pi t}{5}\right)\right)\cos\left(\frac{2\pi t}{5}\right) \\
\ddot{\lambda}_{3,d} &= -\frac{\pi^4}{100\sqrt{2}}\cos\left(\frac{\pi}{4}\cos\left(\frac{2\pi t}{5}\right)\right)\left(\sin\left(\frac{2\pi t}{5}\right)\right)^2 + \frac{\pi^3}{25\sqrt{2}}\sin\left(\frac{\pi}{4}\cos\left(\frac{2\pi t}{5}\right)\right)\cos\left(\frac{2\pi t}{5}\right)
\end{aligned}$$

(b) Desired end-effector linear and angular accelerations

Give an expression for the linear and angular end-effector accelerations $\dot{v}$ and $\dot{\omega}$ in terms of the operational space position, velocity and acceleration.

Solution: The linear and angular end-effector accelerations are related to the operational space acceleration through the E matrix.

$$\ddot{\mathbf{x}} = E\begin{bmatrix}\dot{v}\\ \dot{\omega}\end{bmatrix} + \dot{E}\begin{bmatrix}v\\ \omega\end{bmatrix}$$

where

$$E = \begin{bmatrix} E_p & 0 \\ 0 & E_r \end{bmatrix}$$

$$E_p = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad E_r = \frac{1}{2} \begin{bmatrix} -\lambda_1 & -\lambda_2 & -\lambda_3 \\ \lambda_0 & \lambda_3 & -\lambda_2 \\ -\lambda_3 & \lambda_0 & \lambda_1 \\ \lambda_2 & -\lambda_1 & \lambda_0 \end{bmatrix}$$

Since E_p is constant, $\dot{E}_p = 0$. So we get

$$\begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} = E^+ \ddot{\mathbf{x}} - \begin{bmatrix} 0 \\ E_R^+ \dot{E}_R \omega \end{bmatrix} = E^+ \ddot{\mathbf{x}}.$$

since $E_R^+ \dot{E}_R \omega = 0$ for Euler parameters.

(c) Operational space controller design

Propose an operational space controller that meets the following requirements:

- Track the desired operational space trajectory with dynamically decoupled PD control and feed-forward acceleration
- Damp the null-space motion, and
- Compensate for gravitational forces

Note that you do not need to compensate for Coriolis/centrifugal forces.

Solution: The following controller achieves the above goals:

$$\begin{aligned} \Gamma &= J_0^T \left(\Lambda_0 \left(-k_{px} \begin{bmatrix} x_p - x_{pd} \\ \delta\phi \end{bmatrix} - k_{vx} \left(\begin{bmatrix} v \\ \omega \end{bmatrix} - E^+ \dot{\mathbf{x}}_d \right) + \begin{bmatrix} \dot{v}_d \\ \dot{\omega}_d \end{bmatrix} \right) + p \right) \\ &\quad + N_0^T \left(A(-k_{vj} \dot{q}) + g \right) \\ \delta\phi &= E^+ (x_r - x_{rd}) \end{aligned}$$

where

$$N_0 = I - \bar{J}_0 J_0$$

is the dynamically consistent generalized inverse of J_0.

Note that the E^+ matrix must be calculated here with the current values for the representation co-ordinates $\mathbf{x}$, not the desired values $\mathbf{x}_d$.

(d) Simulation

Compile the starter code located under `hw2/p2-main.cpp` as follows:

```
$ cd bin
$ pushd ../build && cmake -DCMAKE_BUILD_TYPE=Release .. && make && popd
$ ./hw2-p2
```

As with the previous problem, you should see the robot simply falling under the effect of gravity.

Now, implement your proposed controller in the area marked as "FILL ME IN" in `hw2/p2-main.cpp`. The resulting motion should be identical to the motion you observed in homework 1.

Solution: The solution code for this problem is available under `cs327a/hw2_sol` in `p2-main-sol.cpp`. To compile and run it, from within the `cs327a/bin` folder, run

```
$ git stash
$ git pull --rebase
$ git stash pop
$ pushd ../build && cmake -DCMAKE_BUILD_TYPE=Release .. && make && popd
$ ./hw2-p2-sol
```

Problem 3 - Unified motion and force control

In this problem, you will once again control the KUKA-IIWA arm to track a desired motion at its end effector. However, certain directions will be controlled to apply open loop forces rather than perform motions.

The desired end-effector trajectory is identical to the one you implemented in homework 1 and the previous problem, with the exception that we no longer desire a fixed motion along the x direction. Instead, we require a force of $10N$ to be applied in the x direction while maintaining a zero moment about the y and z axes.

$$y_d = 0.5 + 0.1 \cos \frac{2\pi t}{5}$$
$$z_d = 0.65 - 0.05 \cos \frac{4\pi t}{5}$$

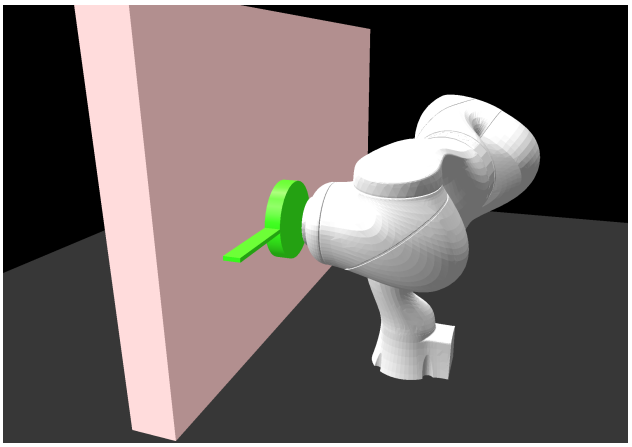
where t is time in seconds. The desired orientation trajectory is represented in Euler parameters as $\lambda_d = (\lambda_{0,d}, \lambda_{1,d}, \lambda_{2,d}, \lambda_{3,d})$ where

$$\lambda_{0,d} = \frac{1}{\sqrt{2}} \sin\left(\frac{\pi}{4} \cos \frac{2\pi t}{5}\right)$$
$$\lambda_{1,d} = \frac{1}{\sqrt{2}} \cos\left(\frac{\pi}{4} \cos \frac{2\pi t}{5}\right)$$
$$\lambda_{2,d} = \frac{1}{\sqrt{2}} \sin\left(\frac{\pi}{4} \cos \frac{2\pi t}{5}\right)$$
$$\lambda_{3,d} = \frac{1}{\sqrt{2}} \cos\left(\frac{\pi}{4} \cos \frac{2\pi t}{5}\right)$$

The desired end effector forces are constant at

$$F_{x,d} = 10$$
$$M_{y,d} = 0$$
$$M_{z,d} = 0$$

As shown below, the forces are to be balanced by contact against a plane surface located at $x = 0.05$.



(a) Force, motion selection matrices

Find the selection matrices Ω and $\bar{\Omega}$ that separate the controlled force directions from the controlled motion directions at the end-effector.

Solution: Since the desired forces are represented in the global frame for this problem, we can write the selection matrices as constant matrices:

$$\Omega = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \bar{\Omega} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

(b) Unified motion and force controller

Modify the controller from the previous problem to produce the desired response above. For controlling the end-effector forces, use feedforward control only.

Solution: The unified force and motion controller with feedforward force control is given by:

$$\Gamma = J_0^T \left(\Lambda_0 \Omega \left(-k_{px} \begin{bmatrix} x_p - x_{pd} \\ \delta\phi \end{bmatrix} - k_{vx} \left(\begin{bmatrix} v \\ \omega \end{bmatrix} - E^+ \dot{\mathbf{x}}_d \right) + \begin{bmatrix} \dot{v}_d \\ \dot{\omega}_d \end{bmatrix} \right) + p + \bar{\Omega} \mathbf{F}_{0d} \right) + N_0^T \left(A(-k_{vj}\dot{q}) + g \right)$$

where $\mathbf{F}_{0d} = [F_{x,d}\ 0\ 0\ 0\ M_{y,d}\ M_{z,d}]^T$.

(c) Simulation

Compile the starter code located under `hw2/p3-main.cpp` as follows:

```
$ cd bin
$ pushd ../build && cmake -DCMAKE_BUILD_TYPE=Release .. && make && popd
$ ./hw2-p3
```

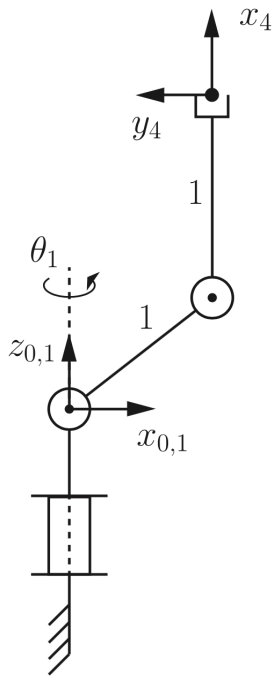
Implement your proposed controller in the area marked as "FILL ME IN" in `hw2/p3-main.cpp`. The resulting motion should be identical to the motion you observed in homework 1 and the previous problem, except the end effector should be flat against the given surface.

Solution: The solution code for this problem is available under `cs327a/hw2_sol` in `p3-main-sol.cpp`. To compile and run it, from within the `cs327a/bin` folder, run

```
$ git stash
$ git pull --rebase
$ git stash pop
$ pushd ../build && cmake -DCMAKE_BUILD_TYPE=Release .. && make && popd
$ ./hw2-p3-sol
```

Problem 4 - Singularity control

Consider the manipulator shown below.



In the configuration $\theta_1 = 0, \theta_2 = 90^\circ$, $\theta_3 = 0^\circ$, the manipulator is at a singularity with respect to the end-effector positioning task.

i. How many degrees of freedom does the end effector have in this configuration?

   Solution: The end effector has only one degree of motion in this configuration to which both joints 2 and 3 contribute.

ii. What is(are) the singular direction(s) in the $\{0\}$ frame?

   Solution: There are two singular directions in the given configuration, one in the Z_0 direction and the other in the Y_0 direction.

iii. What is(are) the direction(s) orthogonal to the singular direction(s)?

   Solution: X_0 is the direction orthogonal to both singular directions for linear motion. In addition, the end-effector can rotate about both Y_0 and Z_0 directions while in this configuration.

iv. What type(s) of singularity(ies) are present in this configuration?

   Solution: There are two types of singularity present in this configuration. The elbow-lock singularity with singular direction Z_0 is of Type I since internal motions in the associated null space can be directly used to move the end effector to the desired position along the singular direction. The overhead-lock singularity with singular direction Y_0 is of Type II since the end-effector must be rotated along a direction orthogonal to the operational force vector to first change the singular direction.

v. Briefly describe how you would control the robot close to this configuration in case you needed the end-effector to be moved along the singular direction(s).

   Solution: In the vicinity of the given configuration, we must control the robot in a different way for each desired task direction. If the position task requires accelerations in the singular directions, then the appropriate self-motion behavior must be employed. To accelerate along the Z_0 direction, a potential function must be constructed whose minimum corresponds to a retracted arm position. Note that this can be achieved simultaneously along with a specified X_0 acceleration by motions on joints 2 and 3. To accelerate along the Y_0 direction, a potential function must be constructed whose minimum corresponds to joint 1 being at 90°. The resulting motion close to the singularity will be to change the singular direction to X_0, thus allowing unconstrained motion along Y_0.