CARL SCHULTZ, STEFAN HALLERSTEDE, EMIL MADSEN

ROBOTIC ARM CASE

AARHUS UNIVERSITY

PUBLISHED BY AARHUS UNIVERSITY

TUFTE-LATEX.GOOGLECODE.COM

First printing, February 2023

Contents

Introduction

In manufacturing and other industries, software controlled *robotic arms* are used to interact with, manipulate, pick up, move around, and place down various objects, e.g. Figure 1. A robotic arm consists of a series of *links* (the straight "bars" or "rods") connected by *joints*, with an *end effector* (robotic "hand") at the end of the arm. Each joint has an electric actuator, i.e. a motor, that generates torque which changes the angle of the joint. This enables the end effector to be moved and positioned as desired, in order to accomplish various tasks.

Figure 1: Robotic arm positioning the end effector to interact with a laptop.

In this project you will create the software controller of a simplified one-joint robotic arm, as illustrated in Figure 2. The one-joint robotic arm specifically consists of:

- a *joint* positioned at angle q;
- a *motor* generating torque τ at the joint;
- a *link* of length r;
- an end effector of mass m;
- a software controller responsible for positioning the end effector at a desired location.

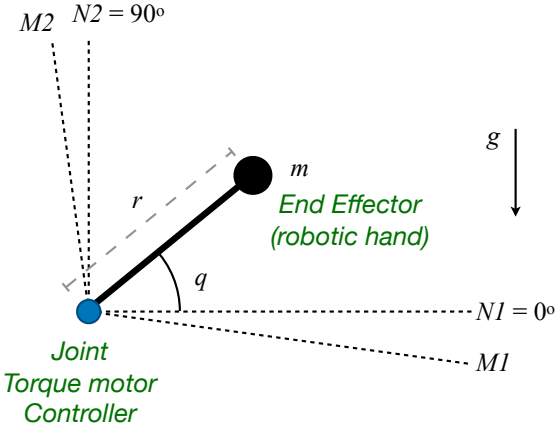

Figure 2: Diagram of the single-joint robotic arm.

Model Variables and Constants

Table 1 lists the variables in the robotic arm model.

Variable	Type	Units	Description
q	real	radians	Angle of joint
q'	real	radians/sec	Angular velocity of joint
q''	real	radians/sec^2	Angular acceleration of joint
q_t	real	radians	Target angle of joint
τ_{motor}	real	N m	Torque from the motor
$\tau_{gravity}$	real	N m	Torque from gravity
$\tau_{friction}$	real	N m	Torque from friction
I	real	kg m^2	Mass moment of inertia

Table 1: Variables in the robotic arm model.

The safe range of angles that the arm joint is allowed to be set to is from 0 rad to $\frac{\pi}{4}$. At any point in time the arm can be stationary (in which case $q' = 0$) or moving ($q' \neq 0$). Gravity acts on the mass of the end effector which generates a *gravity torque*. Friction torque is generated between contact points within the joint.

For simplicity, we can imagine that the gravity torque pushes the arm *clockwise*, friction torque works against the current direction of movement, and positive torque generated by the motor pushes the arm *anti-clockwise* from the perspective of the diagram in Figure 2.

In our case the robot hand interacts by making contact with objects, rather than picking up objects. Therefore the mass of the hand is constant. As our robotic arm only has one joint, with only one angle of motion, we can refer to the robotic arm as having a single degree-of-freedom.

End effector mass, gravity, and the length of the link do not change during operation under the assumptions of the environment, and are therefore modelled as constants. Table 2 lists the constants in the robotic arm model. All constants are *reals*.

Constant	Units	Description
m	kg	Mass of the end effector
g	m/sec^2	Acceleration due to gravity
r	m	Length of link
F_c	(none)	Coulomb friction factor
F_v	(none)	Viscous friction factor
$N1$	radians	Lower bound of the target joint angle range
$N2$	radians	Upper bound of the target joint angle range
$M1$	radians	Lower bound of the safe joint angle range
$M2$	radians	Upper bound of the safe joint angle range

Table 2: Constants in the robotic arm model. All constants are *reals*.

Modelling time and change

The evolution of the arm angle over time is governed by Newton's 2nd law for rotation (Euler's law). The law states that the sum of moments applied to a rigid body (i.e. the torques) equals the mass moment of inertia (I) multiplied by the angular acceleration (q'').

For this single degree-of-freedom case, we have three applied torques: one torque from gravity, one torque from friction, and one torque applied by the motor. This relationship is defined by the formula:

$$\tau_{motor} + \tau_{gravity} - \tau_{friction} = I \cdot q''.$$

As we are describing the *evolution* of these variables, we also need to represent time and change. By convention, we will denote the value of a variable v at time t_i as $v(t_i)$, for example $q(t_i)$ is the angle of the joint at time t_i, and $q(t_0)$ is the initial angle of joint. If the time is not explicitly denoted then it is assumed to refer to the value of the variable at the *current* time step in a simulation.

To determine the evolution of motion, we need to calculate the angular acceleration at time step t_i, denoted $q''(t_i)$, which we can obtain by rearranging the above formula:

$$q''(t_i) = \frac{\tau_{motor}(t_i) + \tau_{gravity}(t_i) - \tau_{friction}(t_i)}{I(t_i)}.$$

Having obtained angular acceleration, we can in turn calculate the updated angular velocity, and subsequently the updated angle of the arm. Let dt denote the amount of time that elapsed between time points t_{i-1} and t_i. Updated angular velocity is calculated with the formula:

$$q'(t_i) = q'(t_{i-1}) + q''(t_i) \cdot dt.$$

The updated angle of the arm is calculated with the formula:

$$q(t_i) = q(t_{i-1}) + q'(t_i) \cdot dt.$$

Calculating torque and inertia

Gravity torque is a function of the angle of the arm. For example, if the arm is oriented vertically, pointing directly upwards with an angle of $q = \frac{\pi}{4}$, then the effect of gravity on the rotation of the arm is zero. If the arm is oriented horizontally, pointing straight out with an angle of $q = 0$, then gravity has its largest effect by pushing down on the end effector. The gravity torque is defined by the equation:

$$\tau_{gravity} = m \cdot g \cdot r \cdot \texttt{cos}(q).$$

Friction acts *against* the current rotation direction, and is therefore a function of the angular velocity of the arm, q'. In our model it is defined by two components: Coulomb and viscous friction. Friction torque is defined by the following formula

$$\tau_{friction} = F_c \cdot \texttt{sign}(q') + F_v \cdot q'.$$

The `sign` operator returns 1 or -1 depending on whether the input is positive, zero, or negative, respectively:

$$\texttt{sign(x)} = \begin{cases} 1, & \text{if } 0 \leq x \\ -1, & \text{if } x < 0 \end{cases}$$

Unfortunately this `sign` operator can cause an abrupt change in friction with only a tiny change in velocity (i.e. the formula for calculating friction is *discontinuous*), which may cause numerical instability problems with our model and simulation. Consider what happens as a negative angular velocity approaches zero: as soon as $q' = 0$ the Coulomb term abruptly jumps from having magnitude $-F_c$ to magnitude F_c. An alternative is to smooth the Coulomb term when angular velocity is in the range $(-1, 1)$ using the hyperbolic tangent function:

$$\tau_{friction} = F_c \cdot \texttt{tanh}(q') + F_v \cdot q'.$$

Mass moment of inertia is calculated with the formula:

$$I = m \cdot r^2.$$

Table 3 presents suggested constant values and initial variable values. Simulation traces may start with different initial values to exercise various scenarios and use cases.

Variable	Suggested Initial Value	Constant	Suggested Value
q	0	m	1.2
q'	0	g	-9.81
q''	0	r	0.8
q_t	$(\frac{\pi}{8})$	F_c	0.1
		F_v	0.05
		$N1$	0
		$N2$	$(\frac{\pi}{2})$

Table 3: Suggested initial variable values at time t_0, and suggested constant values.

Controller Requirements

The controller must ensure that the robotic arm stabilises on a given target angle q_t. The target angle must only be set within the range $[N1, N2]$:

$$N1 \leq q_t \leq N2.$$

In addition, the controller must ensure that the arm joint angle does not exceed this valid target angle range beyond a threshold angle range for too long, defined as the interval $(M1, M2)$:

$$M1 \leq q \leq M2.$$

To achieve this, the controller will be implemented as a so-called *proportional - derivative* (PD) controller. Figure 3 illustrates a block diagram of the PD controller inputs and outputs.

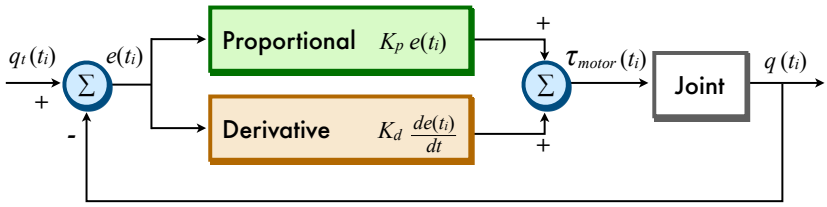

Figure 3: Block diagram of the PD controller used to control the joint angle in the robotic arm.

The idea is that the controller will continuously measure the *error* between the arm's current angle and the target angle, in order to determine the appropriate motor torque to apply at each time step, in a feedback-loop manner. This is the "proportional" component of the controller's logic.

In addition, the controller will keep track of the *rate of change* of this error to respond in a predictive way when setting the motor torque. This is the "derivative" component of the controller's logic.

Table 4 lists the variables in the PD controller, and Table 5 lists the constants in the PD controller.

The error between the target and current angle is calculated with

the formula:

$$e(t_i) = q_t(t_i) - q(t_{i-1}).$$

The *rate of change* of error is calculated with the formula:

$$e'(t_i) = \frac{e(t_i) - e(t_{i-1})}{dt}.$$

The error and its derivative are used to calculate the output motor torque:

$$\tau_{motor}(t_i) = K_p \cdot e(t_i) + K_d \cdot e'(t_i).$$

In addition, the controller can attempt to compensate for the torque due to gravity:

$$\tau_{motor}(t_i) = K_p \cdot e(t_i) + K_d \cdot e'(t_i) - \tau_{gravity}.$$

When substituting this gravity compensated τ_{motor} into the formula for calculating angular acceleration, $q''(t_i)$ we can observe that the gravity terms cancel, and thus the robot will act as if it is operating in a gravity-free environment. This assumes that the controller has perfect knowledge about the model.

Variable	Type	Units	Description
e	real	radians	Target and current angle error
e'	real	radians/sec	Rate of change of error

Table 4: PD controller variables.

Constant	Units	Description
K_p	(none)	Proportional factor
K_d	(none)	Derivative factor

Table 5: Constants in the PD controller. All constants are *reals*.

Figure 4 illustrates a simulation trace of the joint angle over a sequence of time steps, with the PD controller setting the motor torque at each time step according to the gravity compensated formula above. The joint angle is initially at $q = 0$ with a target angle of $q_t = (\frac{\pi}{8})$.

Observe that the controller eventually manages to stablise the joint at the target angle, which is approximately 0.39 radians. However, it over- and under-shoots the target angle a number of times; these are referred to as *oscillations*. Furthermore, the joint angle goes well above and below the target angle, reaching as high as approximately 0.65 radians, and as low as approximately 0.25 after the first oscillation. The largest difference between the target angle and the peak or trough of an oscillation is referred to as the *overshoot amplitude*.

You will use your model to determine values for constants K_p and K_d that minimise:

- the number of time steps before the joint settles on the target angle (within an error threshold that you should state),
- the number of oscillations, and
- the overshoot amplitude.

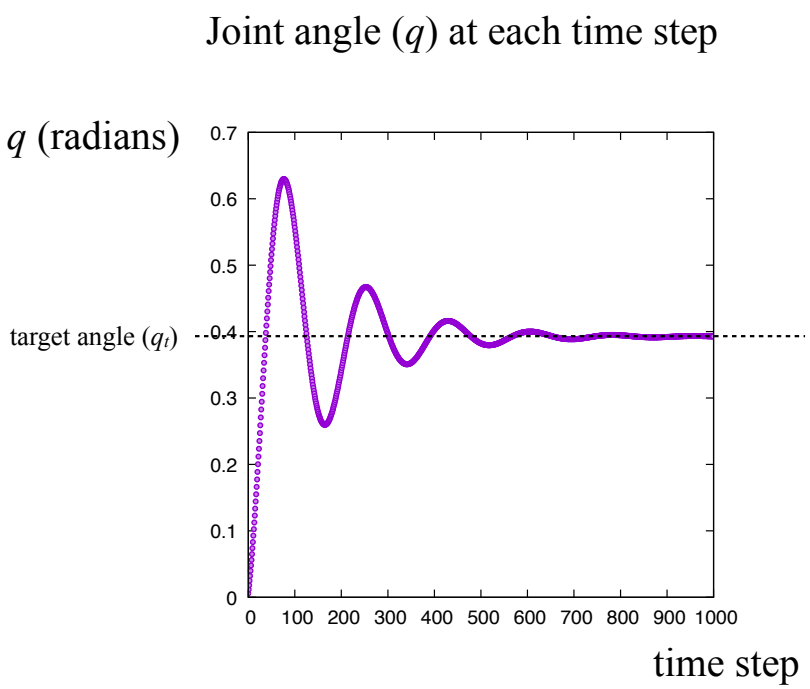

Figure 4: Angle of joint over time, with the PD controller setting the torque motor at each time step.

Bibliography

Index