BSc thesis

High precision control design for the PERA system

Authors:
Alex Sloot S3645010

Supervisors:
prof.dr.ir. J.M.A. Scherpen
Dr. L.P. Borja Rosales
C. Chan Zheng
A.Bosch

Groningen, June 16, 2021

BSc Industrial Engineering and Management
Faculty of Science and Engineering
University of Groningen

Abstract

Robotic arms may perform very high-precision tasks such as wafer production and eye surgery. However, only special classes of mechanical systems currently manage these precise tasks. Demand for a broader range of systems with high precision and accuracy exists in aerospace, medical, and manufacturing fields. In the non-linear field, a class of non-linear port-Hamiltonian mechanical systems with optimization opportunities exists. This thesis will explore the bounds of tuning non-linear passivity-based proportional-integral-derivative controller gains by testing a proposed tuning rules theory by [Chan Zheng et al., 2020]. First, the problem setting will briefly be explained, after which the technical background will be addressed. The remainder of this thesis is split up into three chapters each covering a part of the main question of this thesis: "How can one adequately select non-linear PID controller gains in order to reduce oscillations for port-Hamiltonian systems resulting in higher precision robots to satisfy customer requirements in fields such as aerospace, medical areas, and manufacturing". The Philips Experimental Robot Arm (PERA) will be used for testing the proposed tuning rules.

Keywords— port-Hamiltonian, PERA, PID control, non-linear tuning rules

Contents

Notation

This project involves many mathematical expressions, this chapter will briefly show some used notations.

Matrices of size $n \times m$ are denoted by $A_{n\times m}$, where A represents any matrix. The zero matrix and identity matrix, also of size $n \times m$, are denoted by $0_{n\times m}$ and $I_{n\times m}$, respectively.

Time derivatives can be displayed either using the notation $\frac{dA(x)}{dt}$ or $\dot{A}(x)$. This example uses $A(x)$ as any function at a state x.

The euclidian norm is represented as follows: $||A||_B = \sqrt{A^\intercal BA}$.

Desired points are represented using a * subscript, i.e. desired $q := q_*$.

Eigenvalues are depicted as λ. Maximum and minimum eigenvalues as λ_{max} and λ_{min}, respectively.

The Hessian matrix of function $A(x)$ over b is represented as $\nabla_b^2 A(x)$.

Chapter 1: Introduction

This chapter serves as an introduction by providing the reason(s) for this project, the particular objective this project tries to help realize, and a general overview of the thesis.

1.1 Project description

Many systems are linear, or can be fairly well estimated linearly, meaning ‘discarding’ the higher-order terms in the Taylor series produces a (negligible) error [Tailor and Bhathawala, 2011] [Stewart, 2016]. Currently, robot arms are often not precise enough when discarding the higher order terms. Think for instance of a robot performing eye surgery with a precision of 1 micron [Dormehl, 2017] or manufacturing electronics such as wafers ($diameter \leq 300mm$) [Mathia, 2010]. Although these processes may already be performed by certain robots, these are only small special classes of mechanical systems [Chan Zheng et al., 2020]. Demand exists for options to stabilize a broader range of non-linear systems. Within this novel field of research, discoveries have been made concerning proportional-integral-derivative (PID) controllers for port-Hamiltonian (pH) systems. Port-Hamiltonian systems are based upon the idea of formalizing basic interconnection laws combined with power-conservation. The Hamiltonian is defined as the total energy present in the system [Van Der Schaft, 2006]. Viewing the Hamiltonian as a Lyapunov function, which requires certain criteria (please refer to chapter 2), the system can be tested on stability [Spong et al., 2006]. Any physical system will converge towards a minimum of energy, an equilibrium, provided that no energy is added, the system does not create energy, and dissipation is present [Spong et al., 2006]. Utilizing energy shaping, a certain manipulation that creates an energy minimum at the desired state, the system can converge towards this new equilibrium (energy minimum). Energy shaping combined with a PID controller is proposed by [Zhang et al., 2018]. This research combines this proposed PID controller with a theory of PID gains tuning rules for non-linear pH systems by [Chan Zheng et al., 2020]. The Philips Experimental Robot Arm (PERA) will be used as the non-linear mechanical pH system for this empirical theory-testing research. The PERA tries to mimic human arm-like behavior and can, for instance, be used for pick-and-place type of tasks.

1.2 Problem statement

How can one adequately select non-linear PID controller gains to reduce oscillations for port-Hamiltonian systems - here tested on the PERA system - resulting in higher precision robots to satisfy customer requirements in field such as aerospace, manufacturing, and medical areas.

To solve the problem, a proposed non-linear PID controller using passivity based control by [Zhang et al., 2018] combined with non-linear tuning rules for port-Hamiltonian systems provided by [Chan Zheng et al., 2020] will be implemented and tested on the three degrees of freedom, mechanical PERA system. Ease of implement-ability and measurements of rise time performance and reduction in oscillations will determine the effectiveness of the tuning rules. The research needs to be performed within roughly 13 weeks.

1.3 Contents of this thesis

This thesis will contribute to the novel field of non-linear control theory by the validation of non-linear tuning rules for the proposed PID controller, tested on the PERA robot. The thesis is set up as follows: Chapter 2 presents the technical background of the project, chapter 3 concerns the connection between theory and the PERA, chapter 4 covers the experimental setup and usage of the PERA system, chapter 5 dives into the implementation and results of the tuning rules. A conclusion including potential future work recommendations is provided last.

Chapter 2: Technical background

This chapter concerns the technical background knowledge, which will be key for this project. The technical knowledge includes: port-Hamiltonian structures, passive systems, Lyapunov functions, saddle-point matrices, and tuning rules.

2.1 port-Hamiltonian

A general model for a port-Hamiltonian state-space representation can be seen in (2.1) with corresponding output given as (2.2). Note that this representation does not include the addition of an external force factor as, for instance, shown in [Muñoz Arias, 2015].

$$\begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} 0_{n\times n} & I_{n\times n} \\ -I_{n\times n} & -D(q,p) \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial H(q,p)}{\partial q} \\ \frac{\partial H(q,p)}{\partial p} \end{bmatrix} + \begin{bmatrix} 0_{n\times n} \\ G \end{bmatrix} u \tag{2.1}$$

$$y = G^{\top} \frac{\partial H(q,p)}{\partial p} \tag{2.2}$$

where $0_{n\times n}$ and $I_{n\times n}$ represent a zero matrix and the identity matrix, respectively. The value n equals the number of degrees of freedom (DoF) present. Furthermore, $D(q,p)$ represents the effect of damping due to friction, the dimensions need to match, therefore $D(q,p) \in \Re^{n\times n}$. The actuators incorporated as $G \in \Re^{n\times m}$ (n again as the DoF, m as the number of inputs). The input $u \in \Re^{m}$ will be torque in the case of the PERA, other systems may use force as an input. The Hamiltonian of the above system can be stated as the sum of kinetic energy and potential energy, which can be written as

$$H(q,p) = \frac{1}{2} p^{\top} M^{-1}(q) p + V(q) \tag{2.3}$$

with $M(q) = M^{\top}(q)$, positive definite ($M(q) > 0$) and $M(q) \in \Re^{n\times n}$. $M(q)$ represents the mass inertia matrix and $V(q)$ the potential energy. Chapter 4 will show both matrices for the three DoF PERA system.

2.2 Saddle point matrices

The tuning rules provided by [Chan Zheng et al., 2020] make use of the concept of saddle point matrices, which are a special class of matrices with certain useful properties. Using these properties, the behavior of the system's transient response can be verified which essentially started the theory of non-linear tuning rules. Furthermore, saddle point matrices enable a clear, visible difference between kinetic and potential energy as well as damping in the system.

Picture a matrix of size $n \times m$, a point in the matrix that is both a maximum of its row, n_* and minimum of its column m_*, or vice versa, is stated to be a saddle point of that particular matrix at (n_*, m_*). Matrices can have multiple saddle point numbers as well as multiple maxima or minima.

Linear systems can be described as shown in (2.4).

$$\begin{bmatrix} \dot{\tilde{x}} \\ \dot{\tilde{y}} \end{bmatrix} = -\Theta \begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix}, \Theta := \begin{bmatrix} A & B^{\top} \\ B^{-1} & C \end{bmatrix} \tag{2.4}$$

where $A \in \Re^{n \times n}$ and A is positive definite, $C \in \Re^{m \times m}$ and C is positive semi-definite, $B \in \Re^{m \times n}$ with full rank, $m \leq n$. The matrix Θ from (2.4) belongs to a class of saddle-point matrices. The usefulness of this particular matrix form is that the eigenvalues of the matrix Θ can be obtained by individually looking at the sub-blocks A, B, and C instead of the entirety of matrix Θ [Benzi, 2006].

2.3 Passive systems

A passive system is a system that does not generate energy by itself. This means that if no energy is inputted and dissipation is present due to friction, the system will eventually reach an equilibrium. If the following equations (2.5)(2.6)(2.7)(2.8) hold, a system $(u \mapsto y)$ is said to be passive.

$$s(x) \geq 0 \tag{2.5}$$

where $s(x)$ represents the storage of energy in state x.

$$D \geq 0 \tag{2.6}$$

with D representing frictional energy, which in the real world is always equal or more than zero. The assumption $D = 0$ may be used when dissipation is low. Later on, this assumption will simplify the port-Hamiltonian system for the PERA as Chapter 3 will show.

There exists a relation between output and power, which can be established by a simple unit check. Square brackets will be used to imply "units of". Note that $[input \times output] = [u].[G^{\mathsf{T}} \times \dot{q}] = [torque].[rotational\ velocity]$ as G^{T} has no units.

$$[u].[G^T\dot{q}] = Nm.\frac{rad}{s} = \frac{m.kg}{s^2}.m.\frac{m.m^{-1}}{s} = \frac{m^2kg}{s^3} = W = [power]$$

Input energy can thus be represented as input times output. The flow of energy, energy input minus energy output (dissipation), can now be formulated. Note that the units of energy flow are Watts, thus using the found relation from the above unit check we can define

$$\dot{s} = u^{\mathsf{T}}y - D \tag{2.7}$$

$\dot{s}$ is the derivative of energy storage, the flow of energy. If (2.6) and (2.7) are satisfied, then the following must also hold

$$\dot{s} \leq u^{\mathsf{T}}y \tag{2.8}$$

To tie in the Hamiltonian to passive mechanical systems, the following relation, equation (2.9), can be used. Which also implies (2.10)

$$H(q,p) = s(x) \tag{2.9}$$

$$\dot{H}(q,p) = \dot{s}(x) \tag{2.10}$$

Another method concluding the same result uses the time derivative of the Hamiltonian. This directly shows $\dot{H}(q,p) = u^{\mathsf{T}}y$ [van den Bos, 2019].

Using the equations (2.8) and (2.10) or the time derivative of the Hamiltonian, the following relation can be obtained by noting that $\dot{q}$ equals $\frac{\partial H(q,p)}{\partial p}$ when substituting the output from (2.2) using (2.1).

$$\dot{H}(q,p) \leq u^{\mathsf{T}}G^{\mathsf{T}}\dot{q} \tag{2.11}$$

Where $u^\intercal$ represents the system's (transposed) input and the remaining term, $G^\intercal \dot{q}$, the output. Again, note equation (2.11) uses 'less than or equal to' due to friction, the assumption $D = 0$ simplifies the equation.

It has now been established that a relation between the Hamiltonian and passive systems exist. However, one last connection still needs to be made, namely the connection to the mechanical PERA system. Luckily, this step is fairly simple as all that needs to be done is to verify whether the requirements in (2.5)(2.6)(2.7)(2.8) hold for the PERA. As the PERA exists as a machine in the real world (friction exists) running on electrical energy, (2.5) and (2.6) are immediately satisfied. The energy flow only depends on the power and dissipation as the machine does not generate movement (energy) in itself, (2.7) is thus true. As (2.8) is automatically satisfied when (2.7) is true, the PERA qualifies as a passive system. Any mechanical system can be evaluated in the same way, and will reveal to be a passive system.

2.4 PID-PBC

A proportional-integral-derivative passivity-based controller (PID-PBC) will be used in this project. The mechanical system as shown in (2.1) can be further investigated and altered. As will be shown in section 2.7, energy shaping can be used to change the Hamiltonian of the system. The desired Hamiltonian, denoted as H_d, can now be used to achieve a certain desired end state. The following type of PID-PBC will be considered [Chan Zheng et al., 2020].

$$u = -K_p y - K_I(\gamma(q) + \kappa) - K_d \dot{y} \tag{2.12}$$

where κ represents a constant vector to assign the equilibrium for the closed-loop system. The gains K_P, K_I and K_D satisfy $K_P, K_I > 0$ and $K_D \geq 0$. The variables y and $\gamma(q)$ represent the systems output, $G^\intercal \dot{q}$, and integral of the output, $G^\intercal q$, respectively. The closed-loop system can now be represented as in (2.13)

$$\begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \Upsilon^{-1}(q) F(q,p) \Upsilon^{-\intercal}(q) \nabla H_d(q,p) \tag{2.13}$$

where H_d is defined as (2.14), Υ as in (2.15) and $F(q,p)$ as (2.16). Note the representation of the euclidian norm, used in (2.14).

$$H_d(q,p) = H(q,p) + \frac{1}{2}||\gamma(q) + k||^2_{K_I} + \frac{1}{2}||y||^2_{K_D} \tag{2.14}$$

$$\Upsilon(q) = \begin{bmatrix} I_{n\times n} & 0_{n\times n} \\ GK_D(\nabla_q y)^\intercal & I_{n\times n} + GK_D G^\intercal M^{-1}(q) \end{bmatrix} \tag{2.15}$$

$$F(q,p) = \begin{bmatrix} 0_{n\times n} & I_{n\times n} \\ \text{-}I_{n\times n} & -D(q,p) - GK_P G^\intercal \end{bmatrix} \tag{2.16}$$

2.4.1 PID-PBC versus standard PID

An important note should be made regarding the nature of the PID-PBC compared to a standard PID controller. The widely used PID controllers are based on an error signal around the position of the system. The PID controller gains act upon the position of the system. A PID-PBC is based on the system's passive output, for instance, for mechanical systems it corresponds to the velocity. This implies the PID-PBC gains act on the velocity of the system. The proportional, integral and derivative gains thus do not perform exactly similar for a standard PID controller versus a passivity-based PID controller. The following equations summarize, where x represents the position and $\dot{x} = v$ with v as the velocity.

$$\begin{aligned} & K_P x + K_I \int x + K_D \dot{x} \\ & K_P v + K_I \int v + K_D \dot{v} = K_P \dot{x} + K_I x + K_D \ddot{x} \end{aligned} \tag{2.17}$$

An important observation, that will later on also be used, can be made. Equation (2.17) shows that a standard PD controller is similar to a passivity-based PI controller [1]. Later on in this project, the PERA will be tested using a passivity-based PI controller, which thus relates to a standard PD controller. As Chapter 5 will show, a steady-state error will be observed. A standard PD controller may solve this problem by adding an integrator term, however, for the passivity-based controller this imposes problems as an integrator term is already in use.

2.5 Linearization of port-Hamiltonian systems

Linearization is a commonly performed action to simplify a system and allow for further computations which can be very complex and/or exhausting for non-linear systems, but can be well approximated linearly [Stewart, 2016] [Charlet et al., 1989]. This research uses a linearization of the port-Hamiltonian system around the desired position in q, i.e. $\tilde{q} = q - q_*$, $\tilde{p} = p$, as shown below in (2.18).

$$\begin{bmatrix} \dot{\tilde{q}} \\ \dot{\tilde{p}} \end{bmatrix} = \Upsilon_*^{-1} F_* \Upsilon_*^{-\top} \nabla^2 H_{d*} \begin{bmatrix} \tilde{q} \\ \tilde{p} \end{bmatrix} \tag{2.18}$$

The three PID gains can be related to this linearized form in the following way. The terms, R, P, W, are related to damping injection, potential energy and kinetic energy, respectively [Chan Zheng et al., 2020].

$$\begin{aligned} R &:= G K_P G^\top + D_* \\ P &:= G K_I G^\top + \nabla^2 V_* \\ W &:= G K_D G^\top + M_* \end{aligned} \tag{2.19}$$

As mentioned, tte gains K_P, K_I and K_D satisfy $K_P, K_I > 0$ and $K_D \geq 0$, therefore, the above matrices, R, P, W, are all positive (semi-)definite matrices. Defining the matrices $\phi_W, \phi_P \in \Re^{n \times n}$ as full rank matrices and satisfying the following equation, a transformation can be made.

$$W^{-1} = \phi_W^\top \phi_W, P = \phi_P^\top \phi_P \tag{2.20}$$

The Cholesky decomposition is used to obtain the results. A transformation matrix, $T \in \Re^{2n \times 2n}$, is constructed with new coordinates: $z \in \Re^{2n}$. The following equation (2.21) relates z to this transformation matrix and the linearized system is given as (2.22). The saddle point matrix N as represented in (2.22) will be of importance for the tuning rules due to its saddle point properties.

[1]The proportional term for position relates to the integral term of velocity and the derivative term of position relates to the proportional velocity term.

$$T := \begin{bmatrix} 0_{n\times n} & \phi_W^{-\mathsf{T}} M_*^{-1} \\ \phi_P & 0_{n\times n} \end{bmatrix}, z := T \begin{bmatrix} \tilde{q} \\ \tilde{p} \end{bmatrix} \tag{2.21}$$

$$\dot{z} = -Nz, N := \begin{bmatrix} \phi_W R \phi_W^{\mathsf{T}} & \phi_W \phi_P^{\mathsf{T}} \\ -\phi_P \phi_W^{\mathsf{T}} & 0_{n\times n} \end{bmatrix} \tag{2.22}$$

2.6 Lyapunov stability

General non-linear systems can be represented as follows [Charlet et al., 1989]:

$$\dot{x} = f(x) + g(x)u \tag{2.23}$$

Where $\dot{x} \in \Re^n$, $f(x) \in \Re^n$, $g(x) \in \Re^{n\times m}$ and $u \in \Re$. Lyapunov's second method, also known as the direct method, uses a function $V(x)$ to determine stability. A system can be stated to be stable when the initial state of the system lies closely at the equilibrium and the current state of the system stays in a certain proximity of the equilibrium [Veen et al., 2020] [Spong et al., 2006]. Let us assume an equilibrium point x_* exists for a non-linear function of the form 2.23 with the function $V(x) \in \Re$ as a Lyapunov candidate. It can be stated that $V(x)$ is a Lyapunov function and x_* a stable equilibrium if the following criteria hold [Khalil, 2002] [van den Bos, 2019] [Veen et al., 2020]

- $V(x)$ is continuously differentiable
- $V(x)$ is positive definite relative to x_*
- $\dot{V}(x)$ is negative semi definite with respect to x_*, i.e. $\dot{V}(x) \leq 0$, $\forall x \in \Re$

$V(x)$ is now a Lyapunov function, however, stronger versions of stability exist. If the following criteria applies x_* is locally asymptotically stable.

- $\dot{V}(x)$ is negative definite, $\dot{V}(x) < 0$

Lastly, the strongest stability for x_*, globally asymptotically stable, is true when $V(x)$ is *radially unbounded.* Meaning the function 'blows up', thus no minimum exists as x approaches infinity:

- $V(x) \to \infty$ as $||x|| \to \infty$

For the PERA system, the Hamiltonian can be defined as a Lyapunov candidate. The above criteria can be used to assess Lyapunov stability. Note (2.3) and (2.11) can be substituted into the following.

$$\begin{aligned} H(q,p) &= V(x) \\ \dot{H}(q,p) &= \dot{V}(x) \end{aligned} \tag{2.24}$$

2.7 Energy shaping

Energy shaping concerns the manipulation of functions to "relocate" the desired state to a local minima for the system to converge towards the desired state. This is best illustrated in figure 2.1. The importance of Lyapunov stability can also be easily explained now. As stated, if proven a function indeed classifies as a Lyapunov function, converging behavior towards the equilibrium is guaranteed. Therefore, if the function classifies as a Lyapunov function, the current state will converge towards the equilibrium; minimum in figure 2.1. Proper usage of energy shaping manages to place the desired state at the minimum of the function, effectively guaranteeing the system will converge towards the desired state, provided the function is proven to be Lyapunov stable.

Energy shaping thus changes the Hamiltonian of the system, visible as the energy function on the left in figure 2.1. The desired Hamiltonian is represented as the energy function in the right side of figure 2.1.

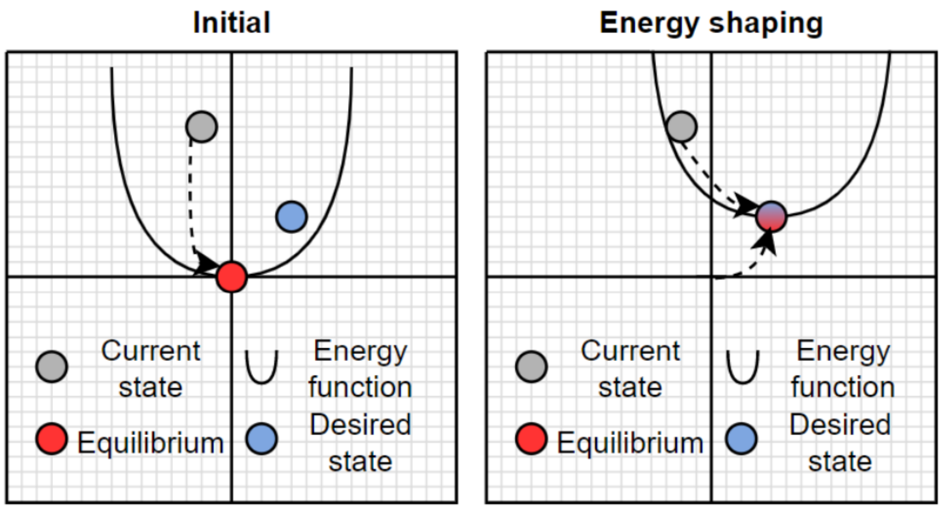


Figure 2.1: A visual representation of energy shaping. Note this is only a visual aid, energy shaping the Hamiltonian related to the PERA may not result as depicted.

2.8 Tuning rules

Tuning rules exist for linear systems. For example the Ziegler-Nichols method may be used[Meshram and Kanojiya, 2012]. However, for non-linear systems due to high complexity, less scientific work is available. The tuning rules provided by [Chan Zheng et al., 2020] have been mentioned a couple of times, but what they entail has not yet been explained. The tuning rules have specifically been made to work with passivity-based controllers and concern three propositions.

Proposition 1: The spectrum of system (2.18) is real and non-negative if expression (2.25) is satisfied. Non-negative, real eigenvalues imply oscillations will not be present in the system.

$$4\lambda_{max}(P)\lambda_{max}(W) \leq \lambda_{min}(R)^2 \tag{2.25}$$

Proposition 2: denoting with (λ_N, v) any eigenpair of (2.22), where $\lambda_N \in C$ and $v \in \Re^n$, then the damping ratio of λ_N is given by (2.26) and bounded by (2.27). The eigenvalues are constrained by (2.28). Note that this tuning rule can be used when oscillations are present in the system, opposed to *Proposition 1.*

$$\zeta_N := \frac{1}{2}\frac{v^* X v}{v^* v}\left(\sqrt{\frac{v^* Z^\mathsf{T} Z v}{v^* v}}\right)^{-1} \tag{2.26}$$

$$\zeta_{min} \leq \zeta_N^2 \leq \zeta_{max} \tag{2.27}$$

$$\begin{aligned} \zeta_{min} &:= max\left(0, \frac{1}{4}\frac{\lambda_{min}(R)^2}{\lambda_{max}(W)\lambda_{max}(P)}\right) \\ \zeta_{max} &:= min\left(1, \frac{1}{4}\frac{\lambda_{max}(R)^2}{\lambda_{min}(W)\lambda_{min}(P)}\right) \end{aligned} \tag{2.28}$$

Proposition 3: The lower bound of the real part of the spectrum of N can be denoted by $\Re(\lambda_u)$. Now the rise time from (2.18) is upper bounded by $t_{ru} \in R_+$. t_{ru} is defined as in (2.29).

$$t_{ru} := \frac{4}{\Re(\lambda_u)} \tag{2.29}$$

Where $\Re(\lambda_u)$ depends on the spectrum of N. Denoted by S1, S2 and S3 are three different forms of the spectrum of N, corresponding with these cases are three different values for $\Re(\lambda_u)$. Note that these cases essentially tell when to use the tuning rule from *Proposition 1* or *Proposition 2.*

S1. the spectrum of N is purely real

S2. all elements in the spectrum of N have an imaginary non-zero element (complex)

S3. some elements are purely real, other elements have a non-zero imaginary part

$$\Re(\lambda_u) = \begin{cases} min\{\lambda_{min}(W^{-1}R), \lambda_{min}(R^{-1}P)\} & \textbf{if S1} \\ \frac{1}{2}\lambda_{min}(W^{-1}R) & \textbf{if S2} \\ min\{\frac{1}{2}\lambda_{min}(W^{-1}R), \lambda_{min}(R^{-1}P)\} & \textbf{if S3} \end{cases} \tag{2.30}$$

Chapter 3: PID controller for the PERA

The key question to be answered this chapter is: "How does the proposed PID controller work theoretically for the three degrees of freedom port-Hamiltonian PERA system?". This question will bring insight into the actual modeling of the PERA system. Knowledge provided in Chapter 2 will be used.

First off, the general port-Hamiltonian model can be used for the PERA system. However, some additional information and assumptions can be used to change the model slightly. This thesis assumes the natural damping of the environment will be negligible. The damping matrix $D(q,p)$ can thus be simplified, in this case into a zero matrix of size $n \times n$. Another assumption regards the actuators of the system. It is assumed the system is fully actuated, which simplifies the port-Hamiltonian system as G now becomes an identity matrix of size $n \times n$. The port-Hamiltonian state-space representation thus becomes as shown in (3.1) with output as (3.2) and Hamiltonian in (3.3).

$$\begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} 0_{n\times n} & I_{n\times n} \\ -I_{n\times n} & -0_{n\times n} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial H(q,p)}{\partial q} \\ \frac{\partial H(q,p)}{\partial p} \end{bmatrix} + \begin{bmatrix} 0_{n\times n} \\ I_{n\times n} \end{bmatrix} u \tag{3.1}$$

$$y = G^{\mathsf{T}} \frac{\partial H(q,p)}{\partial p} = \dot{q} \tag{3.2}$$

$$H(q,p) = \frac{1}{2} p^{\mathsf{T}} M^{-1}(q) p + V(q) \tag{3.3}$$

Now, $M^{-1}(q)$ and $V(q)$ need to be defined. As stated before, these represent the mass inertia matrix and potential energy, respectively. Using the Denavit-Hartenberg conventions, both can be defined [Spong et al., 2006] [van den Bos, 2019]. Note that $V(q)$ resembles a general form of potential energy, $E = m \times g \times h$, where the height has been rewritten to be non-negative and changed from a sine into a cosine. This to more accurately resemble the state of the PERA, which is non-negative for q_2 [Chan-Zheng et al., 2020]. The zero-state for the PERA is stated as the robotic arm pointing downwards (see figure 4.2a).

$$M(q) = \begin{bmatrix} I_1 + I_2 + I_3 + m_3 d_{c2}^2 sin^2(q_2) & 0 & I_3 cos(q_2) \\ 0 & I_2 + I_3 + m_3 d_{c2}^2 & 0 \\ I_3 cos(q_2) & 0 & I_3 \end{bmatrix} \tag{3.4}$$

$$V(q) = g d_{c2} m_3 (1 + sin(q_2 - \frac{1}{2}\pi)) = g d_{c2} m_3 (1 - cos(q_2)) \tag{3.5}$$

The partial derivative of V(q) over q_1, q_2 and q_3, respectively, can be calculated and is shown in (3.6).

$$\frac{\partial V(q)}{\partial q} = \begin{bmatrix} 0 \\ g d_{c2} m_3 sin(q_2) \\ 0 \end{bmatrix} \tag{3.6}$$

Consequently, the Hessian of the potential energy over q, $\nabla_q^2 V(q)$, can be determined and is stated as in (3.7). Note that this matrix relates to equation (2.19) and will be used when calculating the PID controller gains using *Proposition 1.* The Hessian of the desired Hamiltonian is computed similarly, which is used in the linearized system from (2.18).

$$\nabla_q^2 V(q) = \begin{bmatrix} \frac{\partial^2 V}{\partial q_1^2}(q) & \frac{\partial V}{\partial q_1 \partial q_2}(q) & \frac{\partial V}{\partial q_1 \partial q_3}(q) \\ \frac{\partial V}{\partial q_2 \partial q_1}(q) & \frac{\partial^2 V}{\partial q_2^2}(q) & \frac{\partial V}{\partial q_2 \partial q_3}(q) \\ \frac{\partial V}{\partial q_3 \partial q_1}(q) & \frac{\partial V}{\partial q_3 \partial_2}(q) & \frac{\partial V}{\partial q_3^2}(q) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & gd_{c2}m_3cos(q_2) & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.7}$$

3.1 PERA closed-loop and linearized

The closed-loop representation as shown before in (2.13)-(2.16) represents the PERA's closed-loop. The matrices can now be filled in. Due to the number of computations and a high chance of mistakes when calculating by hand, the program MATLAB has been used. The code generating these results can be found in appendix B. This code also provides the linearized system for the PERA as can be calculated using (2.18). The resulting matrices have not been placed here due to size constraints.

Regarding the computation of the matrices. A key point is the calculation of the lower left block of $\Upsilon(q)$ from (2.15), $GK_D(\nabla_q y)^\mathsf{T}$. To obtain this $n \times n$ matrix block, the expression in (3.8) has been used. Where e_i, e_k are standard basis: [1 0 0], [0 1 0] and [0 0 1] depending on the iteration, here n = m = 3.

$$(\nabla_q y)^\mathsf{T} = \nabla_q (G^\mathsf{T} M^{-1}(q)p) = \sum_{i=1}^{m} \sum_{k=1}^{m} \sum_{j=1}^{n} e_i e_k^\mathsf{T} \left(\frac{\partial (G^\mathsf{T} M^{-1}(q))}{\partial q_k} p_j \right) \tag{3.8}$$

Although the above equations (3.1)-(3.8) can be used to model the closed-loop system for the PERA, this method will not be used. The sheer size of the individual entries of the matrices limits the use of them. The matrices can become to big to even be displayed in MATLAB, thus they are not provided here. The PERA itself measures certain variables such as its positions q1, q2 and q3. This simplifies computation into measurements, allowing for usage of the PERA without the knowledge of the entire closed-loop system. The tuning rules use a linearized system to allow for computational speed while staying accurate. To compute the PID controller gains, the closed-loop and linearized system do not need to be known. Therefore, to control the PERA, it is not necessary to obtain the closed-loop or linearized system dynamics. Certain components will still be needed of course. The computation relies on the equations (2.19)-(2.22). Appendix D shows the calculation of the PID controller gains. Note that *Proposition 1* from the tuning rules has been used.

The PID controller gains computed in MATLAB can be implemented into the python code and directly be tested on the PERA. The python code handles all matrix computations during the testing. MATLAB has been used as it is more suited for the usage of symbols in computations.

3.2 Tuning rule implementation

To compute the gains using the first tuning rule, a couple of variables need to be obtained. First off, the mass-inertia matrix, damping matrix (here $0_{3\times3}$) and (Hessian of the) potential energy. Second, the desired values for q and p, $[-\frac{2\pi}{3}\ \frac{\pi}{2}\ \frac{\pi}{2}]^\mathsf{T}$, $[0\ 0\ 0]^\mathsf{T}$, respectively. Third, the values for K_D and K_I need to be set arbitrarily. As this project proposes to use a passivity based PI controller, the K_D values can be set to zero, while the K_I only have to satisfy $K_I > 0$, low K_I values will not perform as well as higher K_I values. The K_P gains can now be calculated using *Proposition 1.* Afterwards, the system can be tested, using the matrices R, P and W from (2.19), combined with cholesky decompositions, the linearized system (2.22) and its spectrum can be obtained. Lastly, a formula to determine whether an overshoot should theoretically occur is used, this formula is shown in (3.9). This equation uses saddle-point matrix N from (2.22). An overshoot occurs if $d < 1$, otherwise no overshoot occurs. A system is critically damped for $d = 1$.

$$d = \frac{-real(eig(N))}{\sqrt{real(eig(N))^2 + im(eig(N))^2}} \tag{3.9}$$

Chapter 4: PERA's testing environment

What does the setup of the PERA system at the university of Groningen look like and how does one operate the robot using Python code. Lastly, how is the test comprised to appropriately test different situations and/or gain values.

4.1 Experimental setup of the PERA

The robot is set up at the University of Groningen. Figure 4.1 shows the available PERA for this project. An emergency stop button may be used to cut off power to the system in case of potential collisions. The emergency stop button is not shown in this figure.

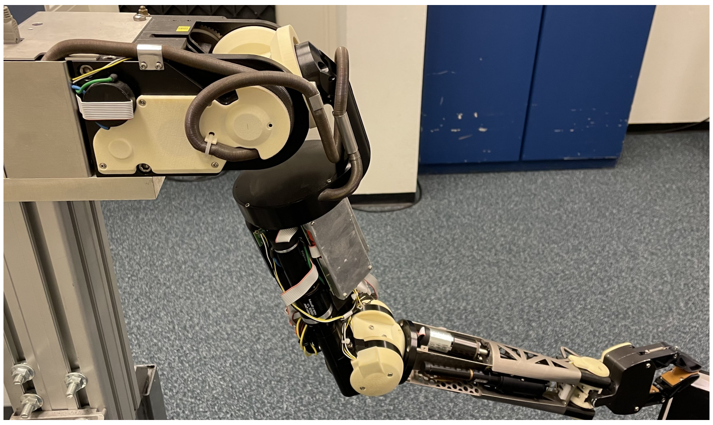
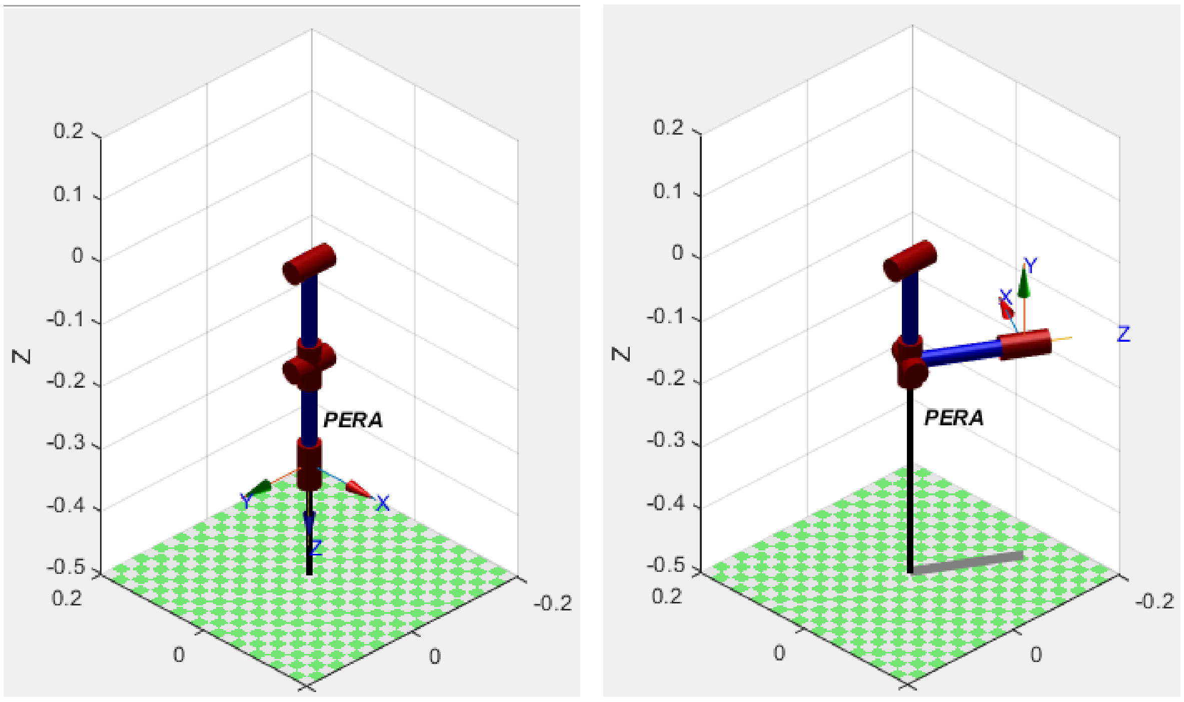


Figure 4.1: The PERA at the university of Groningen

Some approximations for the PERA's masses are given. The parts of interest for this research are the upper and lower arm as well as the gripper. The shoulder joints are not interacted with. The moving parts consist of the upper arm, lower arm and hand/griper, approximated as 2.9 kg, 0.8kg and 0.2kg, respectively [Rijs et al., 2010]. This project does not take into account a separate joint between the gripper and under arm. These can thus be combined to a total of 1kg. In terms of inertia for the three joints, each value is assumed to be 0.01. Lastly, the distance to the denter of the arm is measured as 0.16 meters, this distance is represented in the mass-inertia matrix as d_{c2}.

4.2 Methodology

The test to be performed in this project is comprised of a simple motion of the robotic arm from an initial downward position (see 4.2a) towards a bend in the arm (4.2b). The starting position has been chosen due to an easy and fairly accurate manual reset to this particular position, which is required to keep the system to start in the same position each test. The starting position is [-π 0 0]$^\top$, although the system does not fully reach $q_1 = -\pi$, the system is roughly 180 degrees rotated from its zero position. The shoulder's zero position would make the PERA point towards its own stand, limiting movement and causing potential collisions. The desired position in q is set as $[-\frac{2\pi}{3} \; \frac{\pi}{2} \; \frac{\pi}{2}]^\top$. Note p_* has been excluded as they equal zero.



(a) The PERA in the downward position *(b) The PERA in the desired position*

Figure 4.2: The test to be performed starting in the position shown left, ending in the desired position as shown right. Note the highest placed joint representing the shoulder is merely placed to make the illustration possible, the lower three joints represent q1, q2, and q3.

Appendix C shows the MATLAB code used. This code uses the MATLAB robotic toolbox [Corke, 2017]. The main focus of this project will be on *Proposition 1*, this proposition regards the tuning rule without oscillations, which is more compatible with the PERA. As described in Chapter 3, the gains using the tuning rules have been computed using MATLAB. To test the effectiveness of the computed gains, a comparison between arbitrarily set PID gains and the computed tuning rules' gains is made. An in depth analysis of this comparison is provided in Chapter 5.

Chapter 5: Testing the tuning rules

The chapter covers the question: "Does the method provided to tune the PID controller easily and safely tune the controller gain values to ensure the PERA system is not harmed and in what sense does this method of tuning the controller gains outperform not using the rules". First, a brief explanation regarding the method of testing will be provided. After which the results will be shown and explained. Lastly, some additional observations and remarks will be presented.

5.1 General method

Normally PID-controllers have three K-gain values/matrices. This research has focused on the use of only two of these: the proportional and integral terms, therefore the implemented controller is of the PI type. Both of these consist of a 3×3 diagonal matrix. The use of a diagonal matrix simplifies computations in, for instance, the closed-loop (2.13) or the linearized system (2.18). The K_D-gains have thus been set to zero, resulting in a $0_{3\times3}$ K_D matrix. The K_I values have been arbitrarily chosen. Two separate tests have been performed, one test uses relatively low values, referred to as *low* K_I, the other test uses higher K_I values, referred to as *high* K_I. All values are listed in Table 5.1. Reason for different tests with different K_I values is that the PERA reacts fairly differently due to these changes. Using K_I values lower than the low K_I values chosen here may result in the PERA not reaching the desired position at all. For both low and high Ki value tests, K_P values have been computed using the tuning rules (please refer to appendix D for the used method). Both situations have also been run with arbitrarily chosen K_P values (see Table 5.1) to compare the effect of the first tuning rule to arbitrarily chosen values. Each test has been conducted 5 times and produces a single data-set each time. In MATLAB, the data has been combined into averages to remove randomness. As the linearized system from equation (2.4) shows, the desired p values, P_*, are set as zero, while the desired q values can be set. The initial position is manually set to the downward position as shown in figure 4.2a, not that for the shoulder joint, q1, this position is at -180 degrees, or $-\pi$ radians. The other two joints both start at zero. The desired position has been set to $[-\frac{2\pi}{3}\ \frac{\pi}{2}\ \frac{\pi}{2}\ 0\ 0\ 0]^\mathsf{T}$, or $[-\frac{2\pi}{3}\ \frac{\pi}{2}\ \frac{\pi}{2}]^\mathsf{T}$ when leaving p_* out. The next sections will show most figures next to each other, however, full-sized figures are included in appendix A.

	Low Ki **diag([50 30 20])**	**High Ki** **diag([200 250 200])**
arbitrary Kp	diag([1 13 9])	diag([1 13 9])
tuned Kp	diag([3.3347 2.5830 2.1090])	diag([6.6693 7.4565 6.6693])
high Kp	-	diag([50 50 50])

Table 5.1: The integral and corresponding proportional gains used.

5.2 Low Ki values

As can be seen in figure 5.1, the K_P values using the first tuning rule has a positive effect on the result of all three joints. Especially the desired position of the second elbow joint, q3, which corresponds to the rotation of the lower arm, is reached significantly sooner when using the computed K_P values instead of the arbitrarily chosen ones. It can thus be stated that the first tuning rule, *Proposition 1*, has a positive effect on reaching the desired positions when using (relatively) low K_I values. It should be mentioned that although both q2, and q3 may seem to have a slight overshoot, this is not the case. The computed gains relate have been computed and tested theoretically and are meant to result in a situation where an overshoot is just barely avoided. If the value from (3.9) is less than 1, an overshoot occurs, if it is equal or higher than 1, no overshoot occurs. The computed K_P gains result in a d value of 1, therefore no overshoot should occur.

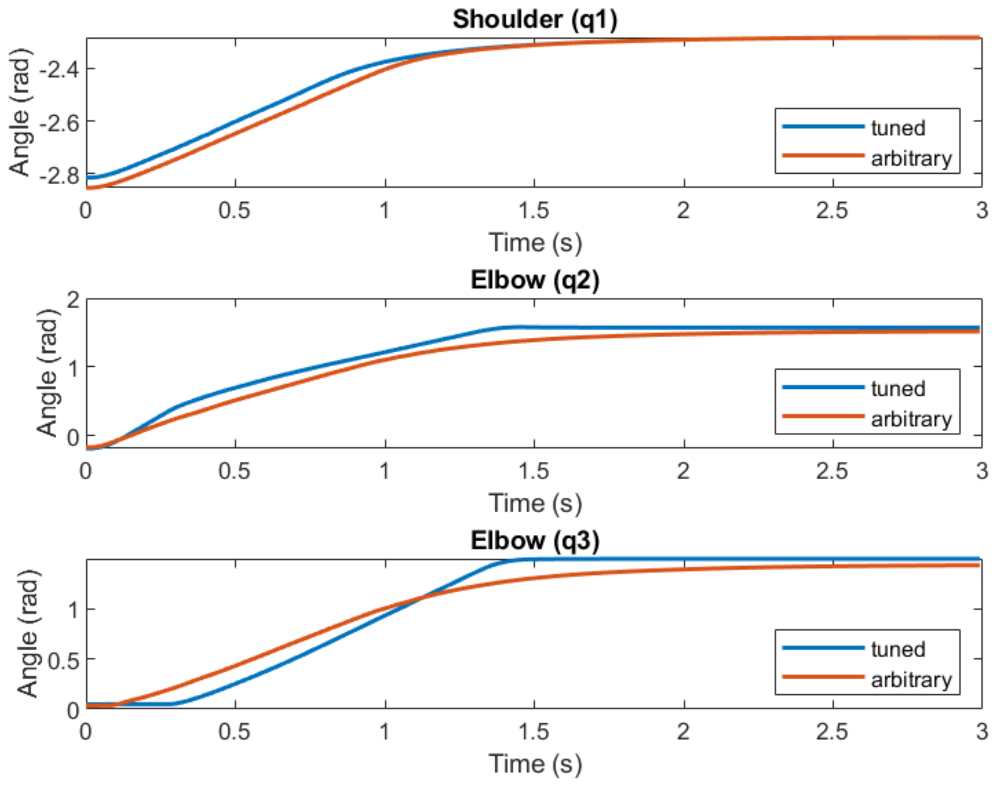
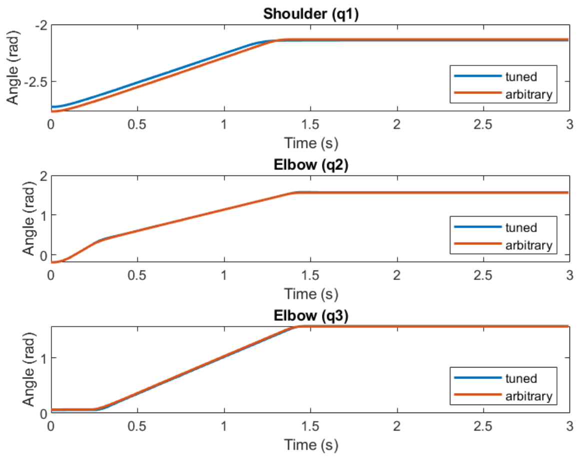


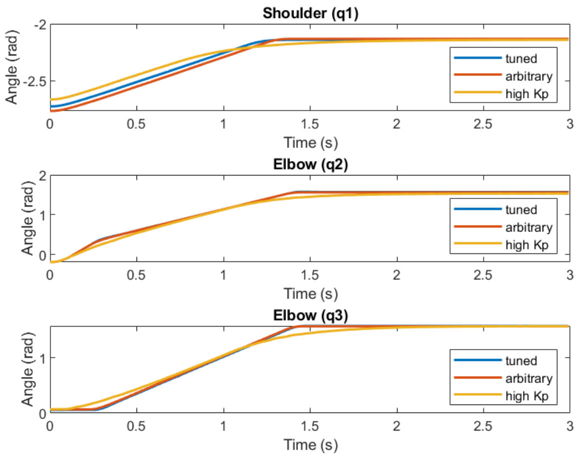
Figure 5.1: Position of q1, q2, and q3 using low Ki values combined with either arbitrary or tuned Kp values.

5.3 High Ki values

Using higher K_I values, the PERA generally moves quicker towards its desired position. However, a trade-off occurs where oscillations due to the faster movement may be present. The K_P values can stabilize this behavior. Again using *Proposition 1* from the tuning rules, the K_P gains are computed similarly but have changed due to different K_I gains.



(a) High Ki values

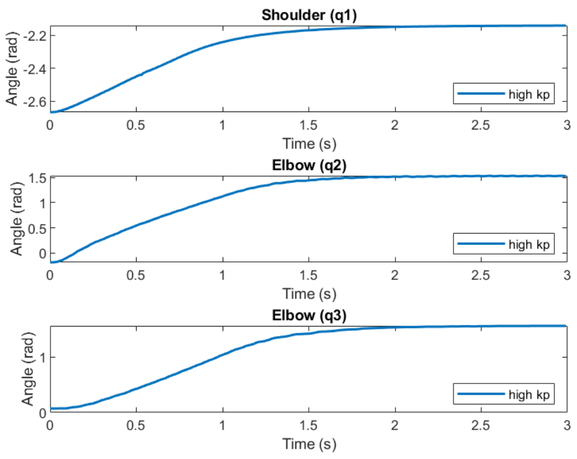


(b) High Kp included

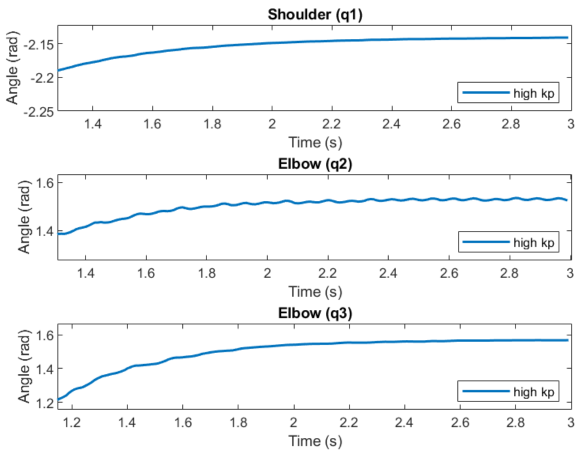
Figure 5.2: Position of q1, q2, and q3 using high Ki values combined with either arbitrary, tuned or high Kp values.

Figure 5.2a shows choosing to use the tuning rules computed gain values does not alter the outcome much. Arbitrarily chosen K_P gains seem to do a decent job. The K_I values are simply significantly higher, limiting the effect of the K_P values. The question then arises whether choosing much higher K_P values may have a positive effect. A test with both high K_I values and high K_P values has been conducted. The results from this test (after averaging) are compared with both arbitrary and tuned K_P values combined with high K_I values in figure 5.2b.

Although figure 5.2b may make it seem like choosing high K_P values is a valid option. A closer look at an individual data-set instead of the average reveals why this is not a good idea. This same result was also visually noticeable when testing the robot. The robot arm becomes unusable as the entire system will move towards the equilibrium so quickly, it overshoots and compensates too much. The system now oscillates around the desired point as can be seen for q_2 in figure 5.3b.

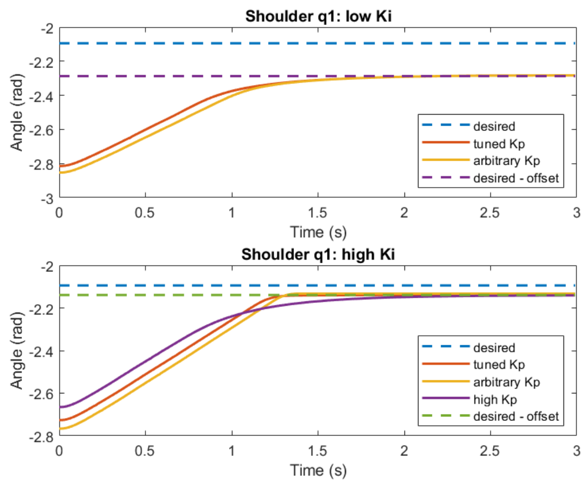


(a) Zoomed out.

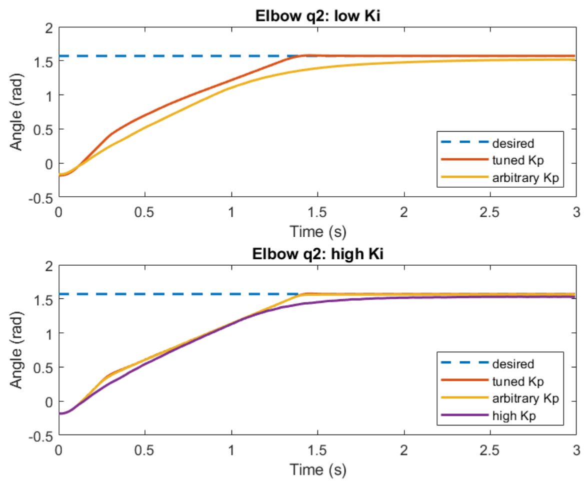


(b) Zoomed in.

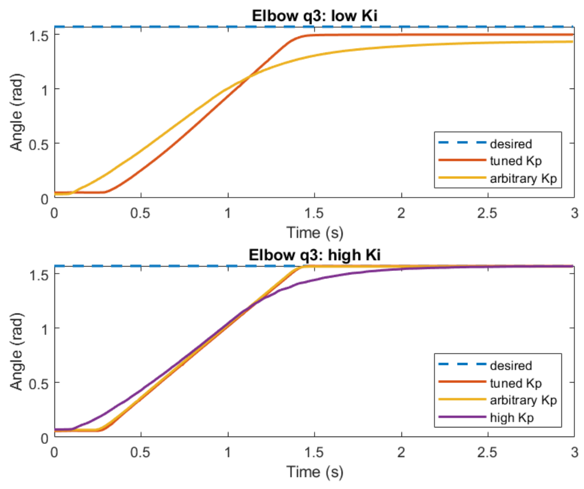
Figure 5.3: A single test of using high Ki and Kp values makes the robot arm oscillate.



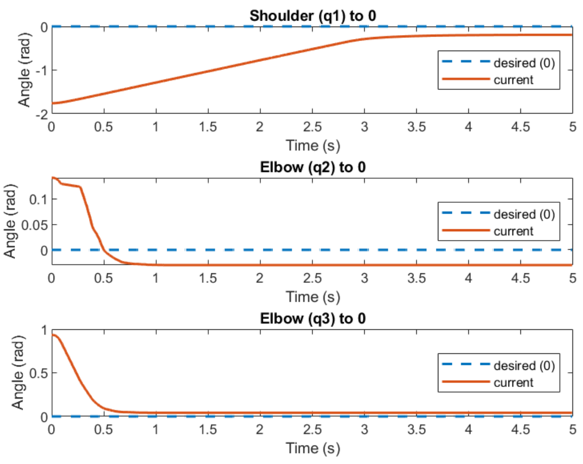
(a) Shoulder



(b) Elbow movement



(c) Elbow rotation



(d) Desired position as [0 0 0]$^{\mathsf{T}}$

Figure 5.4: Shoulder, elbow movement and rotation using low and high Ki values and moving the system towards its zero position.

It can clearly be seen from figure 5.4, using the first tuning rule, *Proposition 1*, has a more positive effect on the rise time and the transient response in general than arbitrary K_P values do.

5.4 Steady-state error

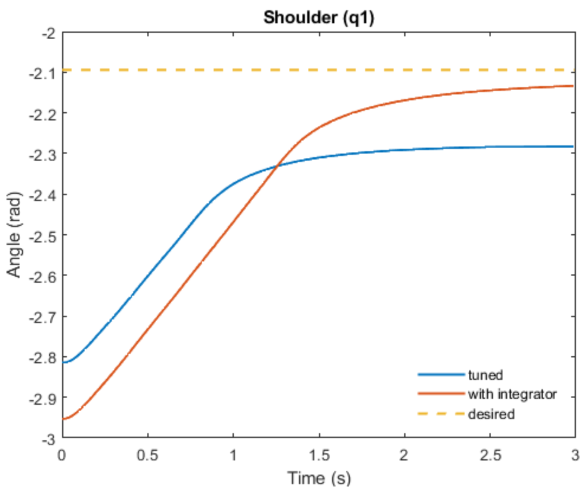
The PERA does not always seem to reach the initially defined desired position. However, as this occurred in every test, it was hypothesized that it may be possible the PERA would have a certain zero-offset. As figure 5.4d clearly shows, when the desired positions are stated as $[0\ 0\ 0]^{\mathsf{T}}$, the PERA does not reach this state, no matter the used gains. Taking this zero-offset, which is mainly present for the shoulder joint, as the other two are an order of magnitude smaller, it can be seen that indeed the desired position differs from the reached position by a similar offset. Figure 5.4a, therefore, incorporates a desired-offset line to show the convergence behavior towards the PERA's final position. An explanation for the PERA's offset may be that the damping matrix is assumed to be zero or the motors undergo a hysteresis error. It has been observed that moving the PERA manually to any position, gravity does not manage to bring the PERA perfectly into its downward position. Therefore, damping must have some effect. However, damping may still be assumed to be zero for the purpose of this project's objective, to test the proposed tuning rule. The general effect of the tuning rule will still be present, difference may be that instead of critical damping due to the tuning rules, the natural damping added may slightly overdamp the system. This coincides with the found results as no overshoots occur and some tests even show slight overdamping, therefore a lower convergence. This lies outside of this project's scope, however.

5.5 Passivity-based controller with extra integrator

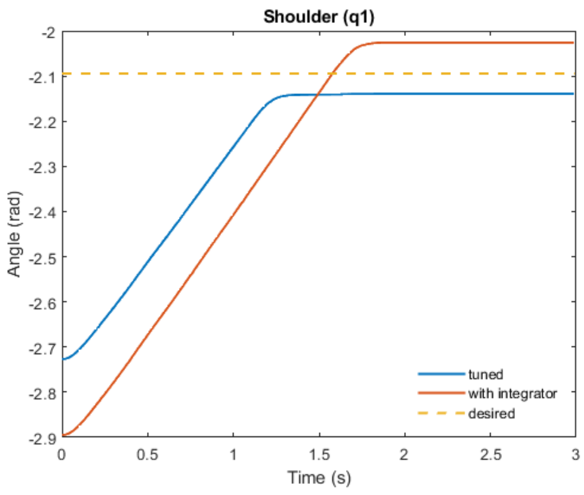
It has been mentioned several times that a passivity-based PI controller coincides with a standard PD controller. Removing the steady-state error with a standard PD controller can be managed using the integral term. This is more complicated for a passivity-based controller as the integral term is already in use. To show the effects of adding an extra integrator to reduce the steady-state error, a simple integrator has been implemented. Due to time constraints, the effect of this simple integrator is far from optimized, however, the same principles still apply. The extra integrator term has itspassivity based gains placed diagonally in a 3×3 matrix, represented as K_{II}. The closed-loop dynamics of the system are not altered, only the input to the system is changed by subtracting the value z. Where the dynamics of z are represented by (5.1), with δ as the elapsed time between consecutive loops to properly match units.

$$\begin{aligned} z &= z + \dot{z} \\ \dot{z} &= K_{II}(q - q_*)\delta \end{aligned} \tag{5.1}$$

The simple integrator has only been tested on the shoulder joint (q1) as this joint experienced the highest steady-state error and the integrator's effect is thus most noticeable for this joint. In 5.5a the effect of the integrator, $K_{II} = \text{diag}([9\ 0\ 0])$ combined with low K_I values can be seen. Figure 5.5b shows the effect of using too high values for K_{II}, diag([35 0 0]), combined with high K_I values, the system now overcompensates. It can be concluded that the extra integrator manages to reduce the steady-state error. However, to reduce the steady-state error optimally, several tests with different K_{II} gains need to be performed to find adequate values.



(a) Low Ki values with integrator



(b) Overtuning the integrator gains

Figure 5.5: The effects of adding an extra integrator term to a passivity-based PI controller.

5.6 Additional remarks

In figure 5.4 the tests with high K_P are also included. It can be seen they do not perform as well as using the tuning rules. Next to this, note these are the averages of the five separate high K_P tests, individually they look similar to the depiction in figure 5.3, implying heavy oscillations.

Some of the above-shown figures may show differences in starting positions. This effect is most noticeable in 5.5. Although each test starts of in the same downward position as figure 4.2a shows, the PERA is manually placed into this position, therefore slight differences in starting positions may be present. This does not affect the outcomes nor does it change the validity of the testing method.

Conclusion and recommendations

This final chapter will conclude the performed work, the outcomes of the project and provide some future work recommendations.

Tuning rules: *Proposition 1*

The performed tests for the PERA show that the implementation of the first tuning rule has a positive effect on the rise time performance and reduces the oscillations of the system. Although some simplifications have been made to the model of the PERA, the general impact of the tuning rule remains unchanged. Potentially, assuming damping of the system to be non-zero may ensure even better performance from the tuning rule as currently slight overdamping may be present. The effect from the tuning rule is most noticeable when using lower K_I values, due to the higher impact of the K_P values. However, the tuning rule still manages to outperform arbitrarily chosen, or high K_P values when using higher K_I values. Therefore, it can be stated that for a three degrees of freedom passive mechanical system, the first tuning rule, *Proposition 1*, computes adequate gains.

passivity-based control integrator

As has been previously mentioned, the implementation of the first tuning rule concerns the transient response and left a steady-state error. The system has reached a high precision, but lacks accuracy. Normally a standard PD controller incorporates an integrator term to solve this problem. However, as equation (2.17) shows, the passivity-based controller similar to a standard PD controller already uses an integral term. Therefore, a different approach is needed. A very simple extra integrator has been implemented and yielded promising results, provided proper gains are used. More complicated and quite possibly better-performing integrators can be implemented. Research surrounding this topic covers certain transformations to incorporate an integral term for a passivity-based controller while preserving the general framework. Some examples consist of [Dirksz and Scherpen, 2011], [Ferguson et al., 2017] and [Ortega and Romero, 2012]. Due to the assumption of zero damping, the integrator proposed by [Dirksz and Scherpen, 2011] matches best for the 3 degrees of freedom PERA. The control input can be changed to incorporate the integrator. However, a hurdle yet to be surpassed is finding a transformation of the linearized system with the integrator that satisfies the required saddle point matrix properties for the first tuning rule.

Future work recommendations

This project has focused on testing the first tuning rule on a three degrees of freedom system. This leaves potential future work to test the tuning rule on a system with more degrees of freedom, although in general, the tuning rule will likely perform adequately. Future tests would need to prove its effectiveness.
As this project has only focused on testing the first tuning rule, the second and third tuning rules still need to be experimentally verified. The second tuning rule incorporates oscillations from the system, therefore, this tuning rule will likely be more difficult to implement. The third tuning rule may give information surrounding the rise time performance of the system,

potentially a system where a very fast rise time is required and some resulting oscillations are allowed would be a suitable test option.
As stated above, the implementation of a passivity-based integrator is not yet complete. A transformation of the linearized system's dynamics satisfying saddle point matrix properties still needs to be found. Due to time constraints, this research has not explored options for such a transformation, which leaves potential future work.

References

[Benzi, 2006] Benzi, Michele & Simoncini, V. (2006). On the eigenvalues of a class of saddle point matrices. *Numerische Mathematik*, 103:173–196.

[Chan-Zheng et al., 2020] Chan-Zheng, C., Borja, P., Monshizadeh, N., and Scherpen, J. (2020). Exponential stability and tuning for a class of mechanical systems. *arXiv preprint arXiv:2011.14543.*

[Chan Zheng et al., 2020] Chan Zheng, C., Borja Rosales, P., and Scherpen, J. (2020). Tuning rules for a class of port-hamiltonian mechanical systems. In *Tuning Rules for a Class of Port-Hamiltonian Mechanical Systems*, page extended abstract. 21st IFAC World Congress ; Conference date: 12-07-2020 Through 17-07-2020.

[Charlet et al., 1989] Charlet, B., Lévine, J., and Marino, R. (1989). On dynamic feedback linearization. *Systems & Control Letters*, 13(2):143–151.

[Corke, 2017] Corke, P. I. (2017). *Robotics, Vision & Control: Fundamental Algorithms in MATLAB.* Springer, second edition. ISBN 978-3-319-54413-7.

[Dirksz and Scherpen, 2011] Dirksz, D. A. and Scherpen, J. M. (2011). Port-hamiltonian and power-based integral type control of a manipulator system. *IFAC Proceedings Volumes*, 44(1):13450–13455.

[Dormehl, 2017] Dormehl, L. (2017). Robot eye surgeon is 10x more precise than the most steady-handed human. Last accessed 18 February 2021.

[Ferguson et al., 2017] Ferguson, J., Donaire, A., and Middleton, R. H. (2017). Integral control of port-hamiltonian systems: Nonpassive outputs without coordinate transformation. *IEEE Transactions on Automatic Control*, 62(11):5947–5953.

[Khalil, 2002] Khalil, H. K. (2002). *Nonlinear systems; 3rd ed.* Prentice-Hall, Upper Saddle River, NJ. The book can be consulted by contacting: PH-AID: Wallet, Lionel.

[Mathia, 2010] Mathia, K. (2010). *Robotics for electronics manufacturing: principles and applications in cleanroom automation.* Cambridge university press.

[Meshram and Kanojiya, 2012] Meshram, P. and Kanojiya, R. G. (2012). Tuning of pid controller using ziegler-nichols method for speed control of dc motor. In *IEEE-international conference on advances in engineering, science and management (ICAESM-2012)*, pages 117–122. IEEE.

[Muñoz Arias, 2015] Muñoz Arias, M. (2015). *Energy-based control design for mechanical systems: Applications of the port-Hamiltonian approach.* PhD thesis, University of Groningen.

[Ortega and Romero, 2012] Ortega, R. and Romero, J. G. (2012). Robust integral control of port-hamiltonian systems: The case of non-passive outputs with unmatched disturbances. *Systems & Control Letters*, 61(1):11–17.

[Rijs et al., 2010] Rijs, R., Beekmans, R., Izmit, S., and Bemelmans, D. (2010). *Philips Experimental Robot Arm: User Instruction Manual V1.1.* Philips Applied Technologies, v1.1 edition. APT536-09-9962.

[Spong et al., 2006] Spong, M. W., Hutchinson, S., Vidyasagar, M., et al. (2006). *Robot modeling and control.* John Wiley & sons, INC.

[Stewart, 2016] Stewart, J. (2016). *Calculus.* CENGAGE Learning, eighth edition.

[Tailor and Bhathawala, 2011] Tailor, M. R. and Bhathawala, P. (2011). Linearization of nonlinear differential equation by taylor's series expansion and use of jacobian linearization process. *International Journal of Theoretical and Applied Science*, 4(1):36–38.

[van den Bos, 2019] van den Bos, F. (2019). *A pouring application for the Philips Experimental Robotic Arm, using saturated passivity-based control without velocity measurements.* PhD thesis, Rijksuniversiteit Groningen.

[Van Der Schaft, 2006] Van Der Schaft, A. (2006). Port-hamiltonian systems: an introductory survey. In *Proceedings of the international congress of mathematicians*, volume 3, pages 1339–1365. Citeseer.

[Veen et al., 2020] Veen, J. et al. (2020). *Passivity-based trajectory tracking of the Philips Experimental Robot Arm.* PhD thesis, Rijksuniversiteit Groningen.

[Zhang et al., 2018] Zhang, M., Borja, P., Ortega, R., Liu, Z., and Su, H. (2018). Pid passivity-based control of port-hamiltonian systems. *IEEE Transactions on Automatic Control*, 63(4):1032–1044.

Appendices

Appendix A: PERA plots

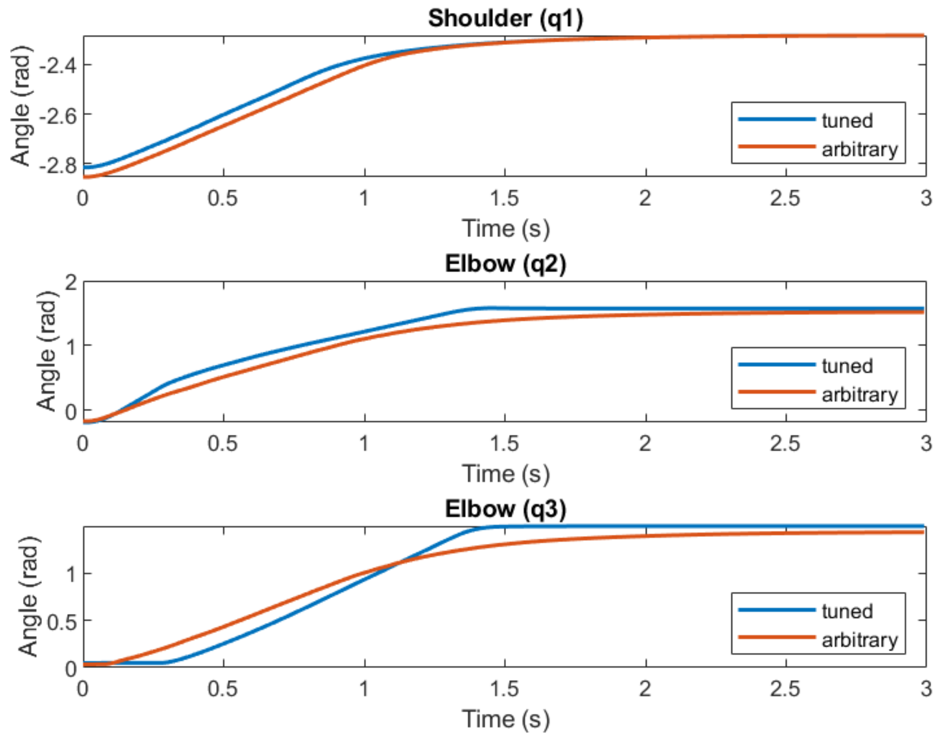


Figure A.1: Low Ki

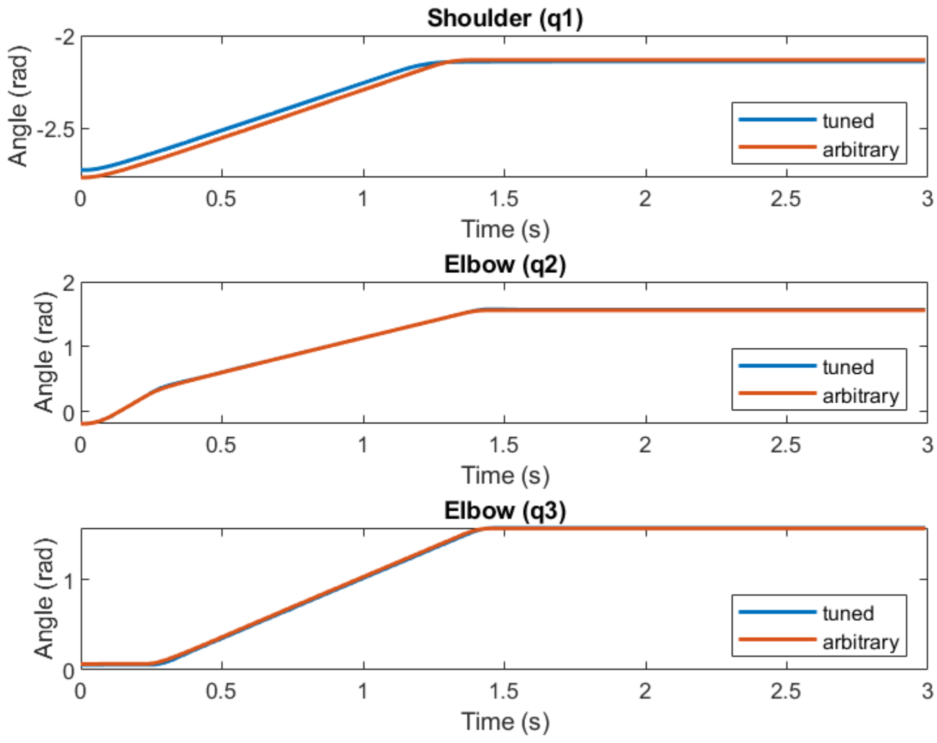


Figure A.2: High Ki

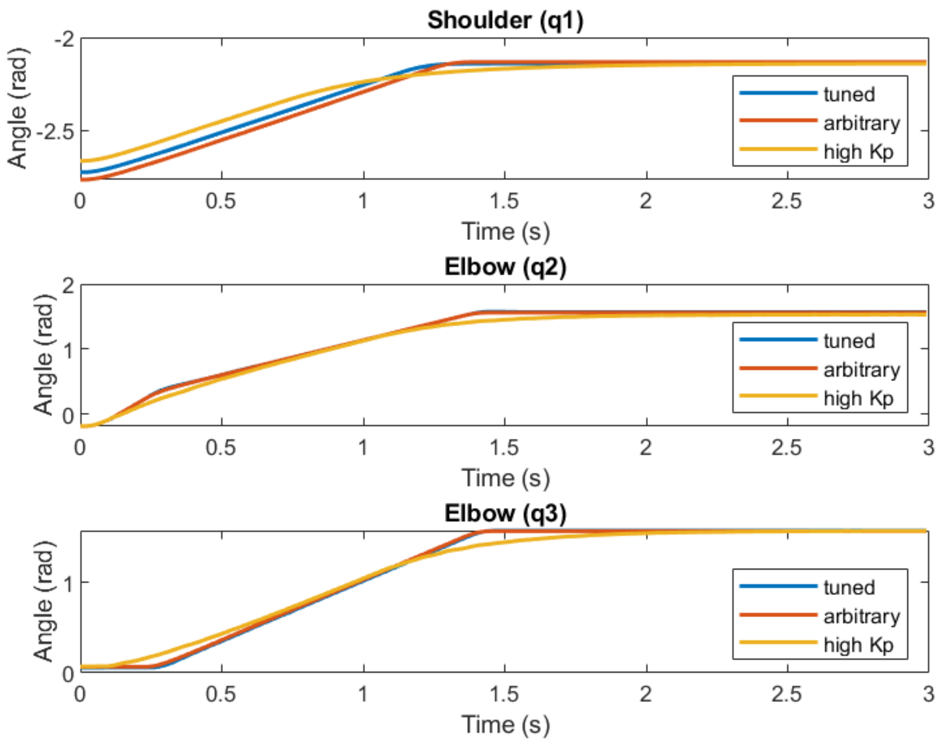


Figure A.3: High Ki, high Kp included

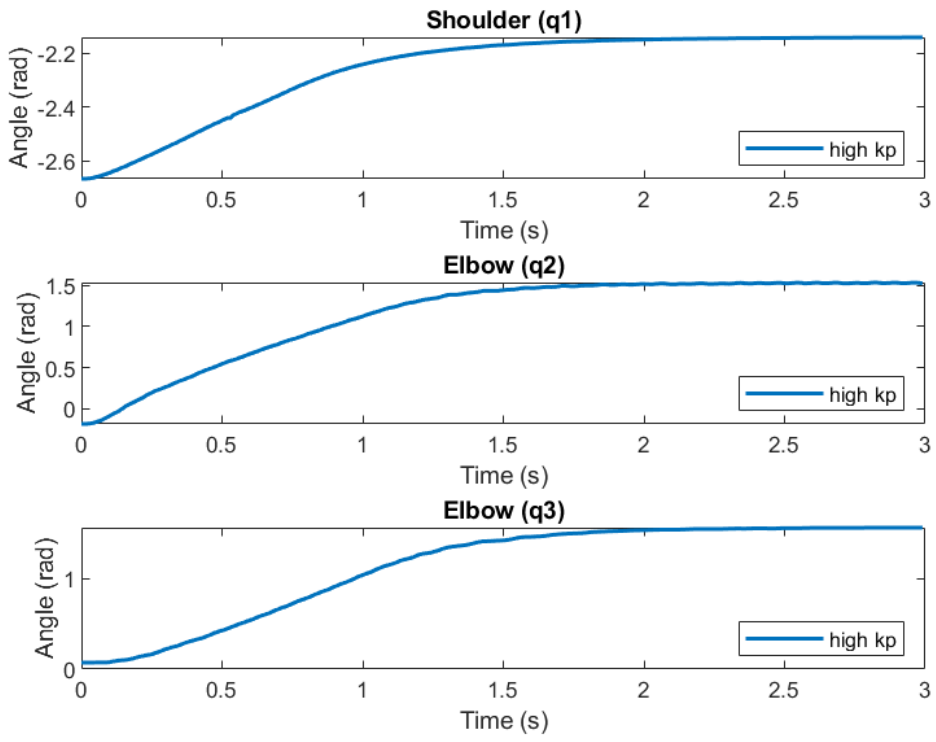


Figure A.4: High Kp zoomed out

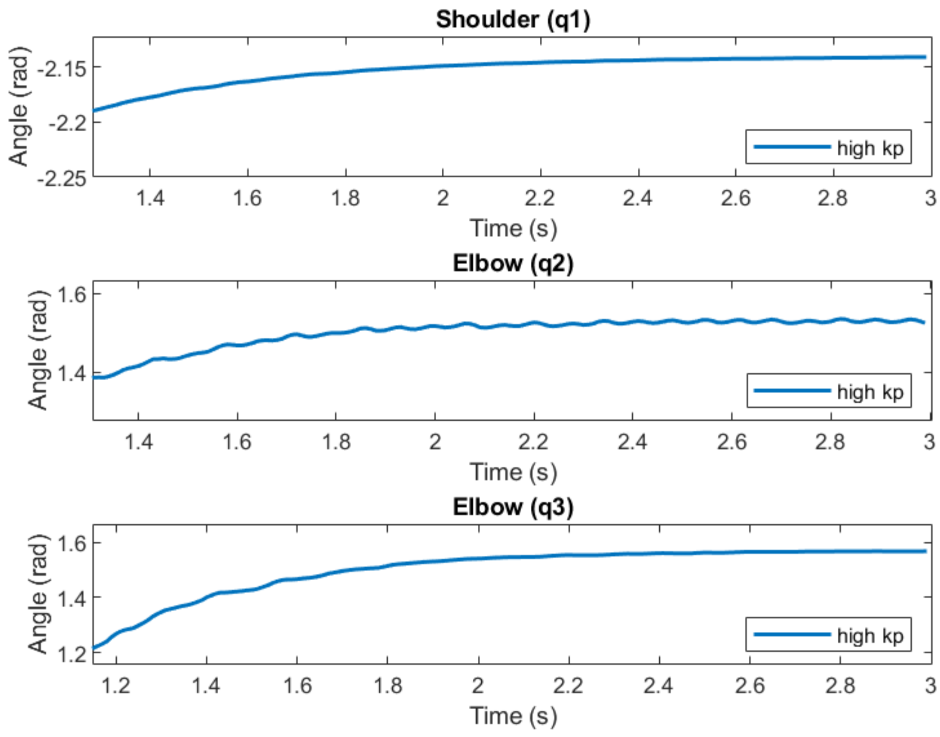


Figure A.5: High Kp zoomed in

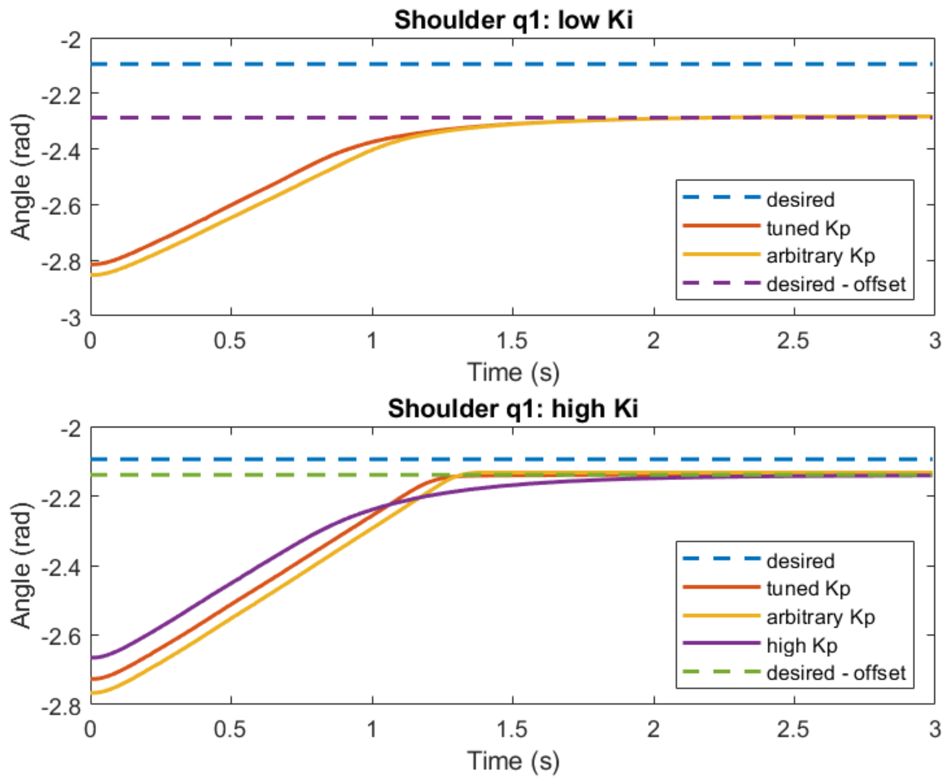


Figure A.6: Shoulder

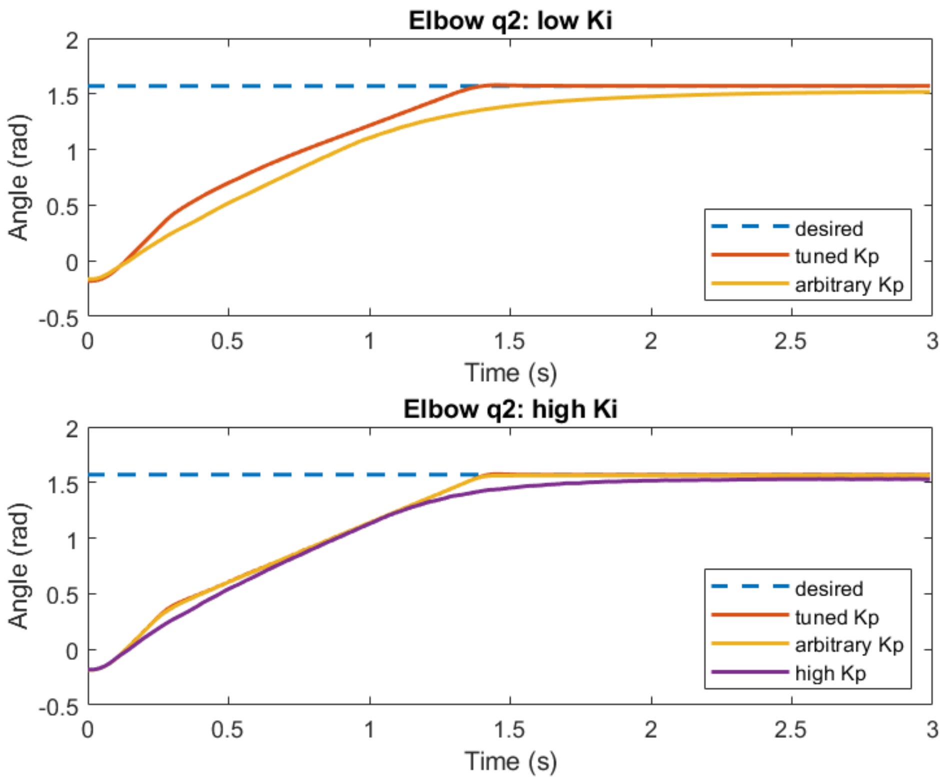
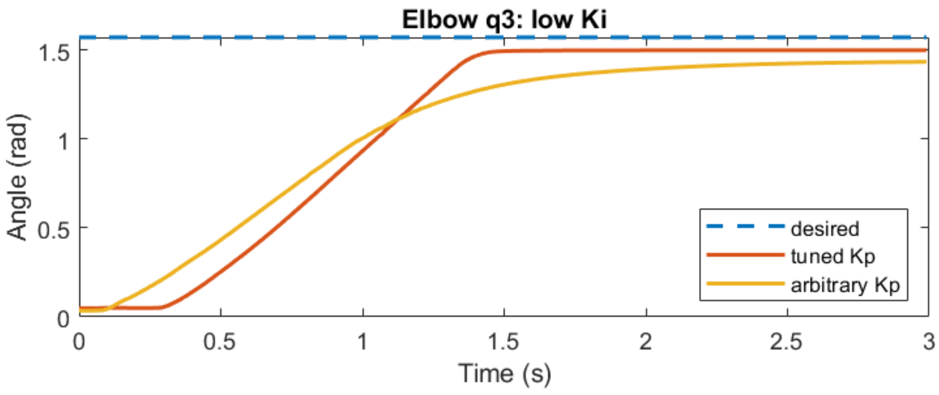


Figure A.7: Elbow movement



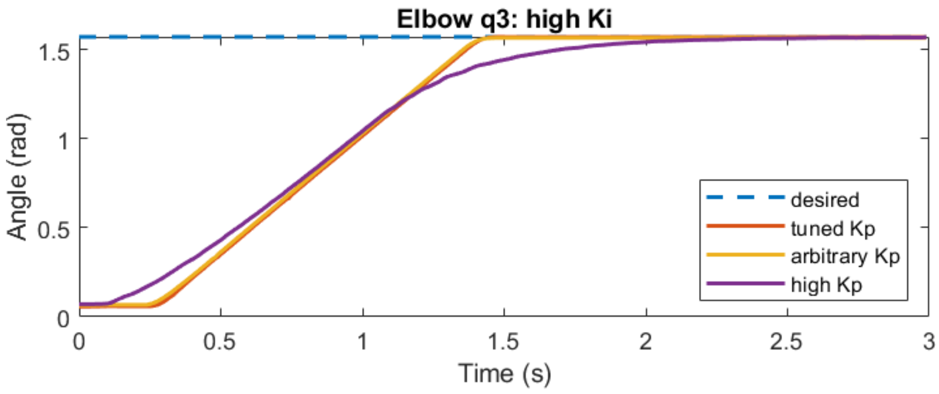


Figure A.8: Elbow rotation

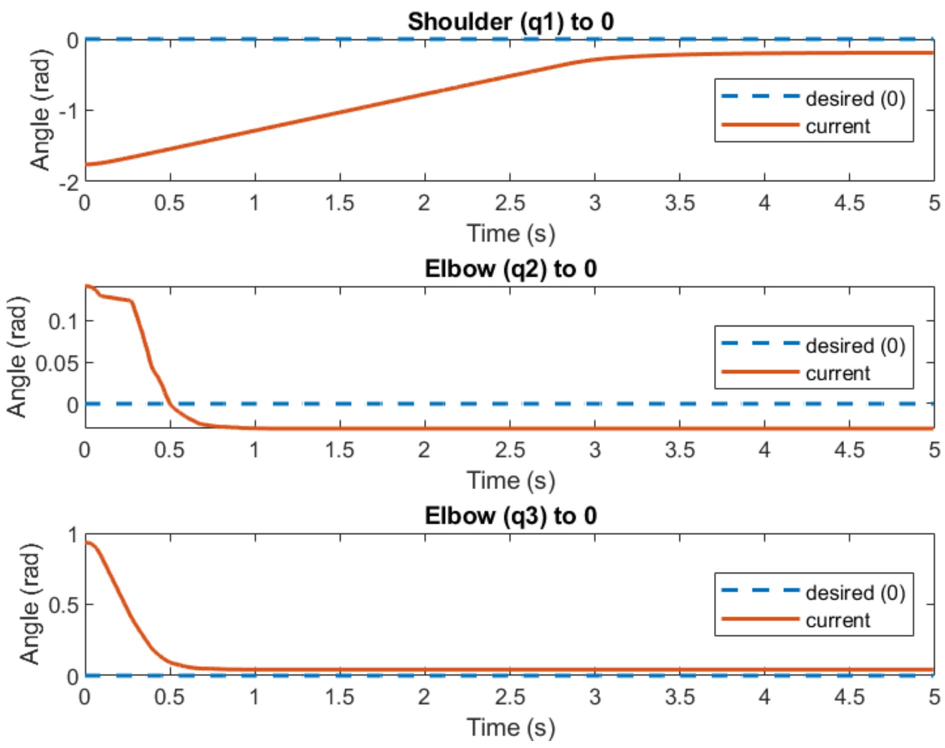


Figure A.9: Desired position as [0 0 0]$^{\mathsf{T}}$

Appendix B: Code for the linearized PERA matrices

```matlab

% Alex Sloot, S3645010
% this script calculates the linearized system of
% [tilde_q_dot; tilde_p_dot]
% the outcome will not be used for any calculations
% merely to show the sheer size of the computations
% and therefore the effect of the tuning rules theory

%% Variables with syms
clear all; clc;

syms q1 q2 q3
q = [q1; q2; q3];
syms p1 p2 p3
p = [p1; p2; p3];
syms q1dot q2dot q3dot
qdot = [q1dot; q2dot; q3dot];

syms y
G = sym(eye(3));
y = transpose(G) * qdot;

syms I1 I2 I3 m3 dc2 g
M = [I1 + I2 + I3 + m3*dc2^2 * sin(q2)^2, 0, I3*cos(q2);
    0, I2 + I3 + m3*dc2^2, 0;
    I3*cos(q2), 0, I3];
invM = simplify(inv(M));

% V(q) for potential energy, Vdiff for the partial derivative over q
V = -g * dc2 * m3 * cos(q1) * sin(q2);
Vdiff = [diff(V, q1); diff(V, q2); diff(V, q3)];

% Hamiltonian, note that G is identity so transpose(G) = G, k is a constant
syms k Kd Ki Kp
H = 1/2 * transpose(p) * invM * p + V;
% the following is used in Hd (euclidian norm): (||A||_K)^2 = ...
    (sqrt(transpose(A)*K*A))^2
Hd = H + 1/2 * (sqrt(transpose(G*q+k)*Ki*(G*q+k)))^2 + 1/2 * ...
    (sqrt(transpose(y)*Kd*y))^2;

% matrix for F
F(1:3, 1:3) = sym(zeros(3));           % 3x3 zeros
F(1:3, 4:6) = sym(eye(3));             % 3x3 identity
F(4:6, 1:3) = sym(-1*eye(3));          % 3x3 -1*identity
F(4:6, 4:6) = sym(-G * Kp * transpose(G)); % D(q, p) = 3x3 zeros, left out

% computations for nabla_y
L = G * invM;
nabla_y = zeros(3);
for i=1:3
```

```
    for k=1:3
        for j=1:3
            % standard basis vectors: e_i, e_k
            e_i = [0 0 0];
            e_i(i) = 1;
            e_k = [0 0 0];
            e_k(k) = 1;
            nabla_y = nabla_y + e_i * transpose(e_k) * diff(L, q(k)) * p(j);
        end
    end
end
nabla_y = simplify(nabla_y);

% matrix Y
Y(1:3, 1:3) = sym(eye(3));
Y(1:3, 4:6) = sym(zeros(3));
Y(4:6, 1:3) =  G * Kd * nabla_y;
Y(4:6, 4:6) = simplify(sym(eye(3)) + G * Kd * transpose(G) * invM);
Y = simplify(Y);

% the partial of the desired hamiltonian over q and p, (size = 6x1)
partial_Hd = [diff(Hd, q1); diff(Hd, q2); diff(Hd, q3);
              diff(Hd, p1); diff(Hd, p2); diff(Hd, p3)];
partial_Hd = simplify(partial_Hd);

% the hessian (2nd partial derivative over q and p) matrix of Hd
Hd_hessian = sym(zeros(6));
for row = 1:6
    for col = 1:6
       Hd_hessian(row, col) = partial_Hd(row)*partial_Hd(col);
    end
end
Hd_hessian = simplify(Hd_hessian);

%% Variables values
%Variables (should be calculated, assumption all=0.01)
I1 = 0.01;
I2 = 0.01;
I3 = 0.01;
g = 9.81;
% syms need to be substituted
dc2 = 0.16; % distance to center of the arm
m3 = 1; % m3 = lower arm (0.8kg) + gripper (0.2kg)
%(q*, p*) = (q*, 0)
q1 = -2/3*pi;
q2 = 1/2*pi;
q3 = 0;
p1 = 0;
p2 = 0;
p3 = 0;

%% linearized system matrices around (q*, p*) = (q*, 0)
Y_star = subs(Y)
F_star = subs(F)
Hd_hessian_star = subs(Hd_hessian)

%% linearized system
```

```
% in between steps are used as the computation is too long at once
a = simplify(Y_star \ F_star); % faster than inv(Y_star)*F_star
b = simplify(-1*transpose(Y_star));
c = simplify(a * b);
qp_tilde = c * Hd_hessian_star
qp_tilde = simplify(qp_tilde)

%% closed loop system (too long/complicated)
% [qdot; pdot], qdot = [qdot1; qdot2; qdot3], same for pdot, (size = 6x1)
% qdot_pdot = inv(Y) * F * transpose(Y)^-1 * partial_Hd;
% qdot_pdot = simplify(qdot_pdot);
```

Appendix C: Code for the PERA joints simulation.

```matlab
% File to give a representation of the to be performed test

% Alex Sloot, S3645010
clear all; clc;

% Initialize PERA
% the first link is unused, links 2,3,4 are q1, q2, q3, respectively.
L(1) = Link('revolute', 'd', 0, 'a', 0, 'alpha', -pi/2, 'modified', ...
    'offset', -pi);
L(2) = Link('revolute', 'd', -0.16, 'a', 0, 'alpha', -pi/2, 'modified', ...
    'offset', 0);
L(3) = Link('revolute', 'd', 0, 'a', 0, 'alpha', -pi/2, 'modified', ...
    'offset', 0);
L(4) = Link('revolute', 'd', 0.16, 'a', 0, 'alpha', -pi/2, 'modified', ...
    'offset', 0);

% Create homogeneous transformation matrix from final joint to end effector
A34 = eye(4);
% Create and plot PERA
PERA = SerialLink(L,'name','PERA');
PERA.tool = A34;
PERA.plotopt ={'workspace' , [-.2 .2 -.2 .2 -0.5 .2]};
figure(1)
PERA.teach
```

Appendix D: Code for the PERA gains using tuning rule 1

```

% Alex Sloot, S3645010
% this script calculates the gains for the PERA
% some assumptions have been used, such as the initial ki gains
% and the damping is set to a zero matrix
% due to an assumed to be fully actuated system, G is an identity matrix

%% Variables in syms
clear all; clc;
syms g dc2 m3 q1 q2 q3
q = [q1; q2; q3];

V = m3 * g * dc2 * (1 - cos(q2));
Vdiff = [diff(V, q1); diff(V, q2); diff(V, q3)];

% for the hessian: take the second partial derivative
% (row,col)=(1:3, 1:3) relate to partial over q1, q2, q3, respectively
for row = 1:3
    for col = 1:3
       V_hessian(row, col) = diff(Vdiff(row), q(col));
    end
end
V_hessian = simplify(V_hessian);

%% Variables with values
% the syms need to be substituted to allow for calculations
%Variables (should be calculated, assumption all=0.01)
I1 = 0.01;
I2 = 0.01;
I3 = 0.01;
g = 9.81;
dc2 = 0.16; % distance to center of the arm
m3 = 1;     % m3 = lower arm (0.8kg) + gripper (0.2kg)
%(q*, p*) = (q*, 0)
q1 = -2/3*pi;
q2 = 1/2*pi;
q3 = 0;
p1 = 0;
p2 = 0;
p3 = 0;

% mass-inertia matrix
M = [I1 + I2 + I3 + m3*dc2^2 * sin(q2)^2, 0, I3*cos(q2);
    0, I2 + I3 + m3*dc2^2, 0;
    I3*cos(q2), 0, I3];
invM = inv(M);

D = diag([0.1 0.1 0.1])*0; % 3x3 zero matrix

ki = [50 30 20]; % leave fixed
```

```
% ki = [200 250 200]; % leave fixed
%% Tuning rule 1:
% note that as G is an Identity matrix, it can be left out
kp = sqrt(4*max(ki+double(subs(V_hessian)))*max(eig(M)))-min(eig(D));

%% Setting the gains:
Kp = diag(kp);
Kd = 0;
Ki = diag(ki);

%% Testing the computed gains:
% matrices relating to the gains
R = Kp + D;
P = Ki + double(subs(V_hessian));
W = Kd + M;
% cholesky decomposition gives phi_W, phi_P
phi_W = chol(W);
phi_P = chol(P);
N = [phi_W * R * transpose(phi_W), phi_W * transpose(phi_P);
    -phi_P * phi_W, zeros(3)];
eigN = eig(N)
A = [zeros(3), -inv(M);
    Ki, R * inv(M)];
d = -real(eigN)./sqrt(real(eigN).^2 + imag(eigN).^2);
if d < 1
   fprintf("\n%f < 1, therefore an overshoot occurs", d)
else
   fprintf("\n%f ≥ 1, therefore no overshoot occurs", d)
end
fprintf("\n\n")
% some eig(N) do, some do not have an imaginary part
% this has effect on S1, S2, S3 of tuning rule 3
zplane([],eigN)
disp(Kp)
```

Appendix E: Code for the PERA plots using tuning rule 1

```matlab
% Alex Sloot, S35645010
% Plots made from experimental data tested on the PERA
% Note that data from all files goes up to a certain time period

%% Loading and processing data
clear all; clc;
% variables
all_tune_low = [];
all_ran_low = [];
all_tune_high = [];
all_ran_high = [];
all_ran_highKp = [];
all_tune_low_Kii = [];
all_tune_high_Kii = [];
maxRow = 300;
maxCol = 10;
tune_low_combined = zeros(maxRow, maxCol);
ran_low_combined = zeros(maxRow, maxCol);
tune_high_combined = zeros(maxRow, maxCol);
ran_high_combined = zeros(maxRow, maxCol);
ran_highKp_combined = zeros(maxRow, maxCol);
tune_low_Kii_combined = zeros(maxRow, maxCol);
tune_high_Kii_combined = zeros(maxRow, maxCol);
% reading in all data, only the first 400 rows are important
file_dir = 'D:\Documents\MATLAB\Thesis\csv_files\';
range = ['A1:J' num2str(maxRow+1)];
for fileId = 1:5 % fileId will also show the maximum number of files used
    read_tune_low  = xlsread([file_dir 'KI_503020_tuning' ...
       num2str(fileId,'%d') '.csv'], range);
    read_ran_low = xlsread([file_dir 'KI_503020_no_tuning' num2str(fileId, ...
       '%d') '.csv'], range);
    read_tune_high = xlsread([file_dir 'KI_200250200_tuning' ...
       num2str(fileId, '%d') '.csv'], range);
    read_ran_high = xlsread([file_dir 'KI_200250200_no_tuning' ...
       num2str(fileId, '%d') '.csv'], range);
    read_ran_highKp = xlsread([file_dir 'KI_200250200_no_tuning_highKp' ...
       num2str(fileId, '%d') '.csv'], range);
    read_tune_low_Kii = xlsread([file_dir 'KI_503020_Kii' num2str(fileId, ...
       '%d') '.csv'], range);
    read_tune_high_Kii = xlsread([file_dir 'KI_200250200_Kii' ...
       num2str(fileId, '%d') '.csv'], range);
    all_tune_low = [all_tune_low, read_tune_low];
    all_ran_low = [all_ran_low, read_ran_low];
    all_tune_high = [all_tune_high, read_tune_high];
    all_ran_high = [all_ran_high, read_ran_high];
    all_ran_highKp = [all_ran_highKp, read_ran_highKp];
    all_tune_low_Kii = [all_tune_low_Kii, read_tune_low_Kii];
    all_tune_high_Kii = [all_tune_high_Kii, read_tune_high_Kii];
end

```

```
%% combining the data into averages
% looping through each row and column, summing the values from the read-in
% files, and dividing by the number of read-in files to obtain the average
for col=1:maxCol
    for row=1:maxRow
        tune_low_sum = 0;
        ran_low_sum = 0;
        tune_high_sum = 0;
        ran_high_sum = 0;
        ran_highKp_sum = 0;
        tune_low_Kii_sum = 0;
        tune_high_Kii_sum = 0;
        for temp = 1:fileId
            tune_low_sum = tune_low_sum + all_tune_low(row, col + (temp-1) ...
                * maxCol);
            ran_low_sum = ran_low_sum + all_ran_low(row, col + (temp-1) * ...
                maxCol);
            tune_high_sum = tune_high_sum + all_tune_high(row, col + ...
                (temp-1) * maxCol);
            ran_high_sum = ran_high_sum + all_ran_high(row, col + (temp-1) ...
                * maxCol);
            ran_highKp_sum = ran_highKp_sum + all_ran_highKp(row, col + ...
                (temp-1) * maxCol);
            tune_low_Kii_sum = tune_low_Kii_sum + all_tune_low_Kii(row, ...
                col + (temp-1) * maxCol);
            tune_high_Kii_sum = tune_high_Kii_sum + all_tune_high_Kii(row, ...
                col + (temp-1) * maxCol);
        end
        tune_low_combined(row, col) = 1/fileId * tune_low_sum;
        ran_low_combined(row, col) = 1/fileId * ran_low_sum;
        tune_high_combined(row, col) = 1/fileId * tune_high_sum;
        ran_high_combined(row, col) = 1/fileId * ran_high_sum;
        ran_highKp_combined(row, col) = 1/fileId * ran_highKp_sum;
        tune_low_Kii_combined(row, col) = 1/fileId * tune_low_Kii_sum;
        tune_high_Kii_combined(row, col) = 1/fileId * tune_high_Kii_sum;
    end
end

fprintf("\n Columns are as follows:")
fprintf("\n Time, Shoulder, Elbow up, Elbow turn, Vel S3, Vel S2, Vel S1, ...
    Tau0, Tau1, Tau2\n ")

%% Creating a reference line
% the desired positions for q1, q2, q3 over 300 time instances
% q1 needs a slight change:
% -0.193 is a zero-offset for low Ki
% -0.045 is a zero-offset for high Ki
E0(1:maxRow) = pi/2;
E1(1:maxRow) = pi/2;
S0(1:maxRow) = -2*pi/3;
S1(1:maxRow) = -2*pi/3 - 0.193;
S2(1:maxRow) = -2*pi/3 - 0.045;
%% Making the plots
time = tune_low_combined(:, 1); % could be taken from any of the lists

% shoulder rotation q1 (with and without tuning)
figure(1)
```

```
subplot(2, 1, 1)
plot(time, S0, '--', time, tune_low_combined(:, 2), time, ...
    ran_low_combined(:, 2), time, S1, '--', 'LineWidth', 1.5)
title('Shoulder q1: low Ki')
xlabel('Time (s)')
ylabel('Angle (rad)')
legend('desired', 'tuned Kp', 'arbitrary Kp', 'desired - offset', ...
    'location', 'southeast')
% shoulder rotation high Ki values
subplot(2, 1, 2)
plot(time, S0, '--', time, tune_high_combined(:, 2), time, ...
    ran_high_combined(:, 2), time, ran_highKp_combined(:, 2), time, S2, ...
    '--', 'LineWidth', 1.5)
title('Shoulder q1: high Ki')
xlabel('Time (s)')
ylabel('Angle (rad)')
legend('desired', 'tuned Kp', 'arbitrary Kp', 'high Kp', 'desired - ...
    offset', 'location', 'southeast')

% elbow position q2 (with and without tuning)
figure(2)
subplot(2, 1, 1)
plot(time, E0, '--', time, tune_low_combined(:, 3), time, ...
    ran_low_combined(:, 3), 'LineWidth', 1.5)
title('Elbow q2: low Ki')
xlabel('Time (s)')
ylabel('Angle (rad)')
legend('desired', 'tuned Kp', 'arbitrary Kp', 'location', 'southeast')
subplot(2, 1, 2)
plot(time, E0, '--', time, tune_high_combined(:, 3), time, ...
    ran_high_combined(:, 3), time, ran_highKp_combined(:, 3), 'LineWidth', 1.5)
title('Elbow q2: high Ki')
xlabel('Time (s)')
ylabel('Angle (rad)')
legend('desired', 'tuned Kp', 'arbitrary Kp', 'high Kp', 'location', ...
    'southeast')

% elbow rotation q3 (with and without tuning)
figure(3)
subplot(2, 1, 1)
plot(time, E1, '--', time, tune_low_combined(:, 4), time, ...
    ran_low_combined(:, 4), 'LineWidth', 1.5)
title('Elbow q3: low Ki')
xlabel('Time (s)')
ylabel('Angle (rad)')
legend('desired', 'tuned Kp', 'arbitrary Kp', 'location', 'southeast')
subplot(2, 1, 2)
plot(time, E1, '--', time, tune_high_combined(:, 4), time, ...
    ran_high_combined(:, 4), time, ran_highKp_combined(:, 4), 'LineWidth', 1.5)
title('Elbow q3: high Ki')
xlabel('Time (s)')
ylabel('Angle (rad)')
legend('desired', 'tuned Kp', 'arbitrary Kp', 'high Kp', 'location', ...
    'southeast')

%% Subplots for the low Ki values:
figure(4)
```

```matlab
for j = 1:3
    subplot(3, 1, j)
    plot(time, tune_low_combined(:, j+1), time, ran_low_combined(:, j+1), ...
        'LineWidth', 1.5)
    xlabel('Time (s)')
    ylabel('Angle (rad)')
    legend('tuned', 'arbitrary', 'location', 'southeast')
    if j == 1
        title('Shoulder (q1)')
    elseif j == 2
        title('Elbow (q2)')
    else
        title('Elbow (q3)')
    end
end

%% Subplots for the high Ki values:
figure(5)
for j = 1:3
    subplot(3, 1, j)
    plot(time, tune_high_combined(:, j+1), time, ran_high_combined(:, ...
        j+1), time, ran_highKp_combined(:, j+1), 'LineWidth', 1.5)
%     plot(time, tune_high_combined(:, j+1), time, ran_high_combined(:, ...
    j+1), 'LineWidth', 1.5) % no highKp
    xlabel('Time (s)')
    ylabel('Angle (rad)')
    legend('tuned', 'arbitrary', 'high Kp', 'location', 'southeast')
%     legend('tuned', 'arbitrary', 'location', 'southeast') % no highKp
    if j == 1
        title('Shoulder (q1)')
    elseif j == 2
        title('Elbow (q2)')
    else
        title('Elbow (q3)')
    end
end

%% NOTES
% the system does not reach [0 0 0] even when trying to,
% so the lines S, E0, E1 can differ from the final reached value
read_000_pos = xlsread([file_dir 'to_initial_pos.csv']);
time0 = read_000_pos(1:500, 1);
for i = 1:length(time0)
    y0(i) = 0;
end

% highKp_combined probably filters out the heavy oscillations
for p= 1:3
    figure(6)
    subplot(3, 1, p)
    plot(time0, y0, '--', time0, read_000_pos(1:500, 1+p), 'LineWidth', 1.5)
    xlabel('Time (s)')
    ylabel('Angle (rad)')
    legend('desired (0)', 'current', 'location', 'east')
    if p == 1
        title('Shoulder (q1) to 0')
    elseif p == 2
```

```
        title('Elbow (q2) to 0')
    else
        title('Elbow (q3) to 0')
    end
    figure(7)
    subplot(3, 1, p)
    plot(read_ran_highKp(:, 1), read_ran_highKp(:, 1+p), 'LineWidth', 1.5)
    if p == 1
        title('Shoulder (q1)')
    elseif p == 2
        title('Elbow (q2)')
    else
        title('Elbow (q3)')
    end
    xlabel('Time (s)')
    ylabel('Angle (rad)')
    legend('high kp', 'location', 'southeast')
end

%% Plot for using the z integrator low Ki and Kii = [9 0 0]
figure(8)
plot(time, tune_low_combined(:, 2), time, tune_low_Kii_combined(:, 2), ...
    time, S0, '--', 'LineWidth', 1.5)
xlabel('Time (s)')
ylabel('Angle (rad)')
legend('tuned', 'with integrator', 'desired', 'location', 'southeast')
title('Shoulder (q1)')


%% Plot for using the z integrator high Ki and Kii = [35 0 0]
figure(9)
plot(time, tune_high_combined(:, 2), time, tune_high_Kii_combined(:, 2), ...
    time, S0, '--',  'LineWidth', 1.5)
xlabel('Time (s)')
ylabel('Angle (rad)')
legend('tuned', 'with integrator', 'desired', 'location', 'southeast')
title('Shoulder (q1)')
```