System dynamic design and control of the Plugless Robot Arm

Towards energy neutral robotics

Linda van der Spaa

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

System dynamic design and control of the Plugless Robot Arm

Towards energy neutral robotics

Master of Science Thesis

For the degree of Master of Science in BioMechanical Design and Systems and Control at Delft University of Technology

Linda van der Spaa

August 14, 2017

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

Current robots consume a lot of energy. The work required for a task is generally just a tiny fraction of the total energy consumed. Most energy wastefully dissipated. Design of the 2 degree of freedom (DoF) Plugless Robot Arm shows that clever design can reduce these energy losses to a tiny remaining fraction. The Plugless Arm is designed to perform a pick-and-place task displacing packages of 1 kg over a vertical distance of 1 m and some additional horizontal distance in a way that is capable of powering the full system.

Maximum energy recovery is achieved by the design of springs in parallel to the actuators such that minimal actuator currents lead to minimal electric heat losses. This requires simultaneous optimisation of the trajectory and the parallel elastic elements. In a Cartesian configuration a novel method is tested which allows for implicit optimisation of an elastic element without additional parameterisation. The optimal parallel spring characteristic can be expressed as a pure function of the trajectory if the system dynamics are defined as a function of position instead of time.

Afterwards the analytical optimal spring is replaced by a mechanism of low mechanical complexity showing similar energy characteristics. The mechanism is translated to a polar equivalent for the 2 DoF arm and optimised together with the trajectory of the arm. The optimal nominal trajectory is followed under undisturbed conditions when applying a pre-computed feedforward control signal to the actuators of the arm. Additional local optimal linear state feedback control is computed by means of Differential Dynamic Programming. All optimisation is done offline. Performance of the controller under disturbed conditions is tested in terms of accuracy as well as energy consumption.

The system achieves a nominal energy retrieval of 3.64 J per cycle of 2.5 s, which is sufficient to power a light controller, the necessary sensors and a special energy efficient gripper. Some energy is left which can be used by the controller to recover from disturbances which extract energy from the system. The controller is also able to recover additional energy from disturbances which add energy to the system.

The implementation of the controller is further optimised to achieve minimal computational energy consumption and memory requirements. By selective reduction of the control law, considerable data reduction is achieved with negligible impact on the controller performance.

Table of Contents

Acknowledgements

First I want to thank my supervisor ir. Wouter Wolfslag for his support during this extensive thesis project. Then of no less importance, I also want to express my thanks to the numerous friends who have inspired me to look far beyond the boundaries of my main educational background, and who brought me up to some speed in areas I felt could be of significance but I knew too little about. My thanks to those who have supported me, and to those who (tried to) put the brakes on me when I threatened to delve too deep into side matters of less importance.

Delft, University of Technology
August 14, 2017

Linda van der Spaa

Chapter 1

Introduction

The number of robots in industry is growing faster every year. And all of these robots consume energy. Currently they consume so much energy that even a simple mobile robot arm such as the KUKA YouBot can only function for 90 minutes before it needs recharging [1]. And this is a very optimistic estimate as even the specifications warn for the sensitivity of the battery. As long as robots keep consuming so much energy they will remain a slave of the power cable. Because of that, and for a more sustainable world, robots must consume less energy. The question is: How?

To challenge the limits of energy efficiency, I am part of a project team at the faculty of 3mE of the Delft University of Technology committed to design a robot arm which is fully powered by gravity only [2]. This "Plugless Arm" is to be a 3 degree of freedom (DoF) robot arm which will pick up packages of a kilogram and put them down at different positions one meter lower. The robot is to be fully powered by only the difference in potential energy: 9.81 J per task cycle. This excess energy will have to compensate for any dissipating effects and power all on board electronics. This really is very little. Considering a typical energy optimal task cycle of about 2 s, total power consumption may be only 4 - 4.5 W. For comparison how small this number actually is: The KUKA YouBot has a typical power consumption of 80 W [1], but the arm is smaller than the Plugless Arm will be. The industrial robot UR5 is of closer size, but a typical program already consumes 200 W [3]. The Plugless Arm will have a more limited functionality than the UR5, but the success of the project will be an important step towards functional robots with negligible energy demands.

In order to actually achieve this super energy efficient robot, all losses in the system need to be minimised. For minimising the losses in a structured way, extensive literature research was done in [4]. The survey systematically maps the energy losses encountered in the general class of electrically powered robots acting in the physical world and subsequently discusses an extensive number of solutions found in literature which would reduce energy loss significantly. The three main sources of energy loss in a robotic system were found to be: 1) mechanical friction between bodies in relative motion, 2) electric heating due to Ohmic resistance in electric actuators, and 3) active electric components losing energy to transistors switching when executing software instructions.

Of these three main loss factors, motor heating is dominant. This heat loss is proportional to the square of the electric current. This current is in turn directly proportional to the motor torque which supplies the system with mechanical power. Mechanical power in robotics is usually required at low speeds, demanding high torques. Large torques are best produced by a motor with a maximum torque/current ratio, in order to draw minimal current. However, motor currents can be reduced far more by designing the system dynamics such that actuators need to supply only minimum additional torque. Applying mechanical torque by means of springs in parallel to the electric actuators may vastly reduce the torque required from the actuators.

Next to that, electric actuators are also capable to function as generators. When used to regenerate electric energy from excess mechanical energy in the form of negative work, instead of wastefully dissipating it, the energy can be stored for later use and use by other electrically powered components such as the processor. Recovering enough energy to power the processor from the very little energy available from the task will be the major challenge of the Plugless Arm design. The design will focus on maximum electric energy regeneration. The electrical regenerative properties of the system also heavily depend on the backdrivability of the entire drive train. A drive train containing transmissions has its losses magnified by each transmission stage, which in backward direction results in rapidly diminishing regenerative properties. Therefore for maximum electric energy regeneration, the system should be actuated through minimum transmissions.

Finally, control software should be light, i.e. fast to compute with minimal instructions, while exploiting the optimised nonlinear system dynamics to add minimum actuation torque, only when absolutely necessary. Minimum actuator torques optimally using the system dynamics can be computed using optimal control. Solving the optimisation problem however takes a lot of computation power. Computationally light methods on the other hand tend to lead energy inefficient control signals. The most energy efficient control to date has been achieved in the Cornell Ranger [5]. Ranger has an optimal nominal trajectory wrapped in a low-bandwidth reflex based feedback controller. The feedback control is achieved by local linear optimal control in the form of a discrete Linear Quadratic Regulator (LQR). The LQR feedback controller computes the optimal parameters for a correcting control sequence only once a cycle. The moment in the cycle is picked strategically, such that sufficient robustness and stability is achieved while control is both energy and computationally efficient. A similar hybrid control strategy will have to be applied to the Plugless Arm.

This thesis is made up of two distinct parts: Mechanical design and control design. The first part focuses on the design of a spring mechanism which, installed in parallel to each actuator, will maximise the electric power return of the Plugless Arm. However, there is an important footnote. Every mechanical component in relative motion adds a source of energy loss to the system, whether it be in the form of friction, impact, hysteresis, etc. Therefore it is imperative to keep the system as simple of construction as possible.

In literature, the following three approaches for optimising springs can be distinguished:

- Parameter fitting of a simple linear spring mechanism [6, 7]
- Full optimisation of both the spring characteristics and the trajectory [8]
- Expressing the optimal spring parameters as a function of the trajectory [9]

The first approach contains only a very limited set of possible spring characteristics. The best spring characteristic from the set may still not be a very close fit to the task and better performance might have been achieved with a different spring mechanism with a different form of force/torque characteristic. No such form restriction is imposed by either of the other approaches. The second approach results in one very large and very nonlinear optimisation problem. The size of the optimisation problem becomes much smaller when applying the method of Schmit and Okada [9]. They solved for the optimal spring characteristic analytically, such that only the trajectory remained in need of numerical optimisation. The resulting reduced optimisation problem is much better solvable because the solution space has a drastically lower dimensionality.

Yet so far all methods rely on some form of parameterisation of the spring characteristic. A spring characteristic is a function of position while the trajectories are described as a function of time. Time is preeminently suitable as a base parameter for trajectory description as it always progresses monotonically. In the Plugless Arm case, monotonic progression is also observed for position when considering half-cycles: the loaded arm goes only down and after the event of unloading the arm only moves upward. As a result, the system dynamics can be described alternatively, as a function of position instead of time (time remains in the equations as a state variable). With all equations depending on position, the optimal spring characteristic no longer needs parameterisation and can be described as a direct analytical function of the trajectory. Then the trajectory is the only thing that remains to be optimised numerically.

Once the optimised trajectory and corresponding optimal spring are obtained, the resulting spring characteristic needs to be translated to a physical mechanism with low complexity. The mechanism with simple mechanical structure described by Plooij in [10] has yielded large actuator energy reduction in a horizontal pick and place task. For vertical pick and place, already some promising results were obtained when applying the mechanism on parts of the system [8]. The Plooij mechanism can be made suitable for the full system in a straightforward way by adding a spring for endpoint gravity compensation. The resulting combined spring mechanism has several properties which are similar to the previously obtained optimal spring when comparing both spring characteristics. The spring characteristic of the physical mechanism can be tuned by setting only a single parameter. Addition of this parameter to the optimisation increases the size of the optimisation only slightly.

The method just described will not be applied directly to the full system of the multiple DoF rotational arm. First the principles will be tested on a simple Cartesian description of the end effector, in which the system dynamics in vertical and horizontal direction are decoupled. This is done in Chapter 2. The results of the chapter are used subsequently in Chapter 3 to design a 2 DoF arm with favourable dynamical properties for minimal additional control.

Once the mechanical system is designed, the attention can shift to its control. Optimal nominal feedforward control was already considered during design of the mechanical system in Part I. Part II of this thesis reviews the nominal controller from a control design perspective and continues with the design of a feedback control law to stabilise the nominal trajectory when disturbances act on the system. With the very little energy available for this purpose, the maximum disturbance magnitude which can be rejected depends heavily on the energy efficiency of the controller. To assure minimum control energy, some form of optimal control will have to be implemented, but in a way that requires least online computation.

The most promising solutions found in literature [4] comprised combinations of local linear controllers based on the LQR principles. A finite number of local LQR controllers can be computed offline to guarantee stability of any predefined region of the state space [11]. The region of the state space of interest for the Plugless Arm is limited to the proximity of the nominal trajectory. For local optimal stabilisation of the nominal trajectory, iterative LQR (iLQR), extendable to Linear Quadratic Gaussian control (LQG) control [12, 13], is very suitable. Closely related to iLQG is Differential Dynamic Programming (DDP) [14], which achieves quadratic convergence of the optimisation through the use of additional second-order derivatives of the system dynamics.

None of the combined linear optimal control laws just mentioned are light to compute. However, computation of the linear state feedback matrices can be done offline. Then online only the state error needs to be multiplied by one of the predefined matrices. The matrices, if not too many, can be stored in memory on forehand, resulting in optimal control with very low online computation effort.

In Chapter 4 DDP is used to recompute the optimal trajectory for the mechanical arm designed in Chapter 3 and the corresponding nominal feedforward control signal. At the same time feedback gain matrices are obtained, at each time step providing optimal linear state feedback control. The state feedback will ensure the pick-and-place trajectory to be stable. Both the stability and the additional energy cost will be evaluated of recovering from end state deviations as well as state noise. A reduced feedback control law is proposed which reduces the memory requirements significantly while the control performance reduction is negligible.

Finally, the findings of this thesis will be discussed in Chapter 5 before the final conclusions are summarised in Chapter 6.

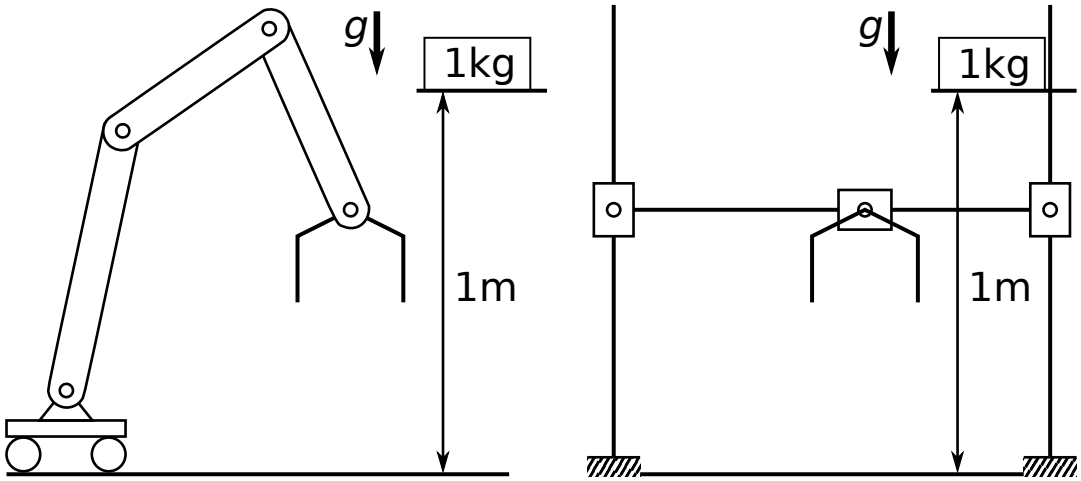
Part I

Mechanical Design

Chapter 2

Cartesian robot

The first goal of this thesis is to design a parallel spring mechanism such that the additional positive work required from the actuators is minimised. Figure 2-1a shows the general Plugless Arm configuration. The mobile platform on which the arm is mounted will not be considered in the rest of this thesis. In this polar configuration the forces acting in different directions on the joints are coupled. This coupled system is too hard to solve directly. Therefore this chapter will consider a decoupled system which is a simplification of the polar arm. Not the full arm, but only the end effector is described by a Cartesian model. This simplified arm model, depicted in Figure 2-1b, has decoupled dynamics in vertical and horizontal direction. This Cartesian model is used to test both mathematical principles as well as mechanism principles in a way that the results are relatively easy to interpret. Afterwards, the results



(a) Polar arm **(b)** Cartesian arm

Figure 2-1: Basic principle of the Plugless Arm in polar and Cartesian configuration.

Table 2-1: Masses and gravity constant used throughout this chapter

Slider mass m_{slider}	0.5	kg
Gripper mass m_{gripper}	0.2	kg
Package mass m_{package}	1.0	kg
Gravitational acceleration g	9.81	m/s^2

can be applied insightfully to a rotational arm in a form similar to Figure 2-1a, which will be done in Chapter 3.

The Cartesian "arm" consists of a gripper connected to a 'slider' which represents the mass of the rest of the arm. The masses are listed in Table 2-1, as well as the gravitational constant used throughout this thesis. A reasonable estimate of the mass of the lightweight gripper is made based on [15].

For the Cartesian arm, vertical and horizontal dynamics can be viewed separately. The only differences between the two directions is the external gravity force acting in vertical direction and perhaps some friction forces of which the magnitude depends on the direction with respect to the gravity. Since it is the gravitational potential energy which is to power the arm, and of which the retrieval has to be maximised, this chapter will focus on the vertical part of the system. Once the vertical part has been fully treated, a short section will discuss the application in the horizontal direction.

First the one-dimensional model is constructed in Section 2-1, including the system dynamical equations of motion as well as a drive train model including an actuator and the necessary transmission. In order to obtain the optimal spring characteristic analytically and without any prior parameterisation, the system model is described as function of position. The analytical optimal spring characteristic is derived in Section 2-2. The corresponding trajectory is optimised in Section 2-3.

Once the optimised trajectory and corresponding optimal spring are obtained, a physical spring mechanism is matched to the analytical optimal spring in Section 2-4. The section builds up the mechanism model consisting of two parts: a gravity balancing spring and a "resonant" mechanism which is the rectilinear equivalent of the Plooij mechanism [10]. The spring mechanism replaces the analytical optimal spring in the trajectory optimisation and the trajectory extended with the single parameter of the spring mechanism is optimised anew. The optimisation results with the analytical optimal spring and the physical mechanism are compared in Section 2-5 and performance of the spring mechanism is evaluated.

Section 2-6 reviews horizontal application of the principle, after which a brief discussion on how to combine the results of the two directions into a single 2D motion will be held in Section 2-7. Finally, the findings of the previous sections are discussed in Section 2-8 before Section 2-9 concludes the chapter.

2-1 System model

The model assumes that the centres of mass of the package, the gripper and the slider coincide. The slider is the spindle nut, which is centred around the spindle shaft through which the

motion is actuated. The actuator is an electric motor of which the rotational torque is transformed to a linear force on the load by the spindle, which has a transmission ratio n_s in rad/m.

The system acceleration is given by Newton's second law in (2-1).

$$\left(m + (J_\mathrm{m} + J_\mathrm{s})n_\mathrm{s}^2\right)\ddot{y} = -mg + F_\mathrm{s} + F_\mathrm{f} + Cn_\mathrm{s}T_\mathrm{m} \tag{2-1}$$

Mass m is the total mass at the actuated joint, i.e. the sum of the masses of the slider, gripper and package, the last only when going down. The two inertia terms: J_m and J_s, are of the motor and spindle respectively. The spring force acts through the term F_s. Friction force F_f can contain both Coulomb friction and viscous friction terms. However in the vertical case, friction of the spindle is negligible and F_f is zero. The constant C is an efficiency term expressing torque dependent loss of the motor and transmission. Torque T_m is the actuator torque at the shaft.

The resulting equations of motion of the 1 degree of freedom (DoF) system as function of time are:

$$\begin{bmatrix} \dot{y} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{-mg+F_\mathrm{j}}{\tilde{m}} \end{bmatrix} \tag{2-2}$$

with

$$\tilde{m} = m + (J_\mathrm{m} + J_\mathrm{s})n_\mathrm{s}^2 \tag{2-2a}$$

$$F_\mathrm{j} = F_\mathrm{s} + Cn_\mathrm{s}T_\mathrm{m} \tag{2-2b}$$

$$C = \begin{cases} \eta_\mathrm{p} & \text{when accelerating} \\ 1/\eta_\mathrm{p}' & \text{when decelerating} \end{cases} \tag{2-2c}$$

The combined joint force has been given its own symbol F_j.

In an ideal motor with zero no-load torque, the electric current consumed by the actuator is given by $I = \frac{T_\mathrm{m}}{k_\mathrm{t}}$ with torque constant k_t. However in reality, at low torques no load current may increase losses considerably. In the less ideal but more realistic case, current through the motor I relates to the motor torque T_m according to (2-3).

$$I = I_\mathrm{noLoad}\mathrm{sign}(T_\mathrm{m}) + \frac{T_\mathrm{m}}{\bar{k}_\mathrm{t}} \tag{2-3}$$

Here $\bar{k}_\mathrm{t}$ is the torque constant corrected for the constant current offset.

Motor and spindle selection First the motor is chosen, then an appropriate spindle. From (2-1) it is seen that the transmission ratio n_s should be as small as possible to lose least power to the reflected motor and spindle inertia, which scales with n_s^2. With also the relation $I \sim \frac{T_\mathrm{m}}{k_\mathrm{t}}$ in mind, a motor with as large a k_t as possible is desired. This also holds for rotational systems. The selected motor will therefore also be used in other parts of this thesis. Details of the selected motor can be found in Appendix A-1.

An appropriate spindle is selected based on preliminary results, which can be found in Appendix B-1. The preliminary results are obtained using the selected motor and the same optimisation as discussed in Section 2-3-1, using a simpler version of the drive train model. Table 2-2 summarises the motor and spindle specific constants which are used in the full model (2-2), (2-3).

Table 2-2: Motor and spindle specifications and derived constants

No load current I_{noLoad}	0.538	A
Terminal resistance R	0.343	Ω
Corrected torque constant $\bar{k}_{\text{t}}$	71.1	mNm/A
Rotor inertia J_{m}	306	kgmm^2
Nut mass m_{nut}	0.65	kg
Spindle inertia J_{s}	519	kgmm^2
Spindle reduction ratio n_{s}	157	rad/m
Forward efficiency η_{p}	0.89	
Backward efficiency η'_{p}	0.87	

Model as function of position In order to obtain the joint force as function of position, the equations of motions must be written as function of position as well. The alternative state $\begin{bmatrix} \dot{y} & t \end{bmatrix}^T$ is introduced. The resulting alternative state equations are:

$$\frac{d}{dy}\begin{bmatrix} \dot{y} \\ t \end{bmatrix} = \dot{y}^{-1}\begin{bmatrix} \ddot{y} \\ 1 \end{bmatrix} = \dot{y}^{-1}\begin{bmatrix} \frac{-mg+F_{\text{j}}}{\tilde{m}} \\ 1 \end{bmatrix} \tag{2-4}$$

These equations are only integrable for $\dot{y} \neq 0$. This means that the trajectories must be defined such that $\dot{y} = 0$ can only occur in the final state. This problem can be solved in the following way: Even without the restriction on the velocity it is a natural choice to split the task cycle in a downward and an upward trajectory, since the mass is different for the return motion. In either direction it will definitely not be optimal to have the velocity changing signs or becoming zero anywhere else than at the end positions. The initial velocity of each half cycle is set to a small number $\pm\epsilon$. The distance Δy that would have been traversed between $\begin{bmatrix} y_0 & 0 \end{bmatrix}^T$ and $\begin{bmatrix} y_0 \pm \Delta y & \pm\epsilon \end{bmatrix}^T$ and is now neglected is practically zero. Therefore no error is induced by starting each trajectory at some small speed $\|\dot{y}_0\| = \epsilon > 0$ instead of $\dot{y}_0 = 0$.

2-2 Analytically optimal spring

The net work of the task of the Plugless Arm originates from the difference in potential energy of the end positions, which is constant. Kinetic energy at the beginning and end of the task is zero. So the optimal spring characteristic function is the minimum of the electric energy equation.

The electric energy of a task cycle is described as function of the position in (2-5). The same separation of the downward and upward motion is observed as described at the end of the previous section, since the direction of the variable over which is integrated may not change under the integral.

$$E = \int_{\text{down}} \frac{P_{\text{e}}}{\dot{y}} dy + \int_{\text{up}} \frac{P_{\text{e}}}{\dot{y}} dy \tag{2-5}$$

Both the electric power P_{e} and velocity $\dot{y}$ need to be a function of position y. By definition of (2-4) velocity indeed is. The electric power is the sum of three parts: actuator heating,

mechanical work and overhead power consumption (2-6). The motor and transmission are for now assumed to be ideal, i.e. without no-load current, inertia and internal friction, efficiency equals one in both transmission directions.

$$P_{\text{heat}} = I^2 R \qquad = \frac{R}{k_{\text{t}}^2 n_{\text{s}}^2} F_{\text{a}}^2 \tag{2-6a}$$

$$P_{\text{mech}} = T_{\text{m}} n_{\text{s}} \dot{y} \quad = F_{\text{a}} \dot{y} \tag{2-6b}$$

$$P_{\text{oh}} = constant \tag{2-6c}$$

The constants R, k_{t} and n_{s} are the wire resistance, motor torque constant and spindle reduction ratio respectively. The motor current I and torque T_{m} are a linear function of the total actuation force F_{a} felt by the load:

$$F_{\text{a}}(y) = F_{\text{j}}(y) - F_{\text{s}}(y) \tag{2-7}$$

The joint force $F_{\text{j}}(y)$ (2-2b) as function of the position y results through $F_{\text{a}}(y)$ in the motor torque T_{m} also being a function of position. With the state description of (2-4), the joint force required for a certain trajectory is indeed obtained as a function of the position.

As a result of (2-6) and (2-7), the electric power is quadratic with respect to the spring force. Furthermore the derivative of the energy is continuous when derived to the spring force. Similar to the normal variable case, the minimum of the quadratic function can be obtained analytically by setting the derivative equal to zero. The necessary theory on functional derivatives will be reviewed in Section 2-2-1.

Subsequently, Section 2-2-2 derives the expression of the optimal spring force as a function of the total joint force and the state of the joint. The state trajectory and corresponding joint forces then remain to be optimised, but with the analytical expression for the optimal spring, the spring characteristic will always implicitly be optimal. How well the optimality holds under less ideal circumstances and the full model as presented in Section 2-1 will be discussed in Section 2-2-3.

2-2-1 Method: Analytical function optimisation by means of the functional derivative

Suppose there is some functional $\mathcal{F}(\alpha, \mathbf{x}(\alpha), \mathbf{x}'(\alpha))$ which should be optimised over some interval $[\alpha_0, \alpha_1]$ such that the corresponding cost function (2-8) is minimised.

$$\mathcal{J} = \int_{\alpha_0}^{\alpha_1} \mathcal{F}(\alpha, \mathbf{x}(\alpha), \mathbf{x}'(\alpha)) d\alpha \tag{2-8}$$

The objective is to find the optimal function $\mathbf{x}(\alpha)$. Analogous to the scalar version,

$$\frac{\partial \mathcal{J}}{\partial \mathbf{x}(\alpha)} = 0$$

needs to be solved.

Taking a step back, consider a normal function $f(x)$ with an optimum $x^\star$. Any nonzero disturbance ξ of the optimum $|f(x^\star + \xi) - f(x^\star)| > 0$. Therefore the derivative of $f(x^\star + \xi) -$

$f(x^\star)$ to the disturbance ξ is only zero when $\xi = 0$. Since $f(x^\star)$ is a constant, it drops out of the equation. In other words:

$$\left.\frac{df(x^\star + \xi)}{d\xi}\right|_{\xi=0} = 0 \qquad \text{with } x = x^\star + \xi \tag{2-9a}$$

$$\left.\frac{\partial f(x)}{\partial x}\right|_{x=x^\star} = 0 \tag{2-9b}$$

This relation is the generally known condition for an optimum of a function.

Extending this principle to functionals, let $\phi_\epsilon(\alpha) = \mathbf{x}(\alpha) + \epsilon\delta(\alpha)$ for some differentiable disturbance function $\delta(\alpha)$ and a small magnitude ϵ. Then the cost $\mathcal{J}$ (2-8) of the disturbed system is given by

$$\mathcal{J}_\epsilon = \int_{\alpha_0}^{\alpha_1} \mathcal{F}(\alpha, \phi_\epsilon(\alpha), \phi'_\epsilon(\alpha))d\alpha = \int_{\alpha_0}^{\alpha_1} \mathcal{F}_\epsilon d\alpha$$

As in the scalar case (2-9a), $\left.\frac{\partial J_\epsilon}{\partial \epsilon}\right|_{\epsilon=0}$ must be zero. And

$$\begin{aligned}\frac{d\mathcal{J}_\epsilon}{d\epsilon} = \int_{\alpha_0}^{\alpha_1} \frac{d\mathcal{F}_\epsilon}{d\epsilon} d\alpha &= \int_{\alpha_0}^{\alpha_1} \left(\cancel{\frac{\partial \mathcal{F}_\epsilon}{\partial \alpha}\frac{d\alpha}{d\epsilon}} + \frac{\partial \mathcal{F}_\epsilon}{\partial \phi_\epsilon}\frac{d\phi_\epsilon}{d\epsilon} + \frac{\partial \mathcal{F}_\epsilon}{\partial \phi'_\epsilon}\frac{d\phi'_\epsilon}{d\epsilon} \right) d\alpha \\ &= \int_{\alpha_0}^{\alpha_1} \left(\frac{\partial \mathcal{F}_\epsilon}{\partial \phi_\epsilon}\delta(\alpha) + \frac{\partial \mathcal{F}_\epsilon}{\partial \phi'_\epsilon}\delta'(\alpha) \right) d\alpha\end{aligned}$$

Then by partial integration

$$\begin{aligned}\left.\frac{d\mathcal{J}_\epsilon}{d\epsilon}\right|_{\epsilon=0} &= \int_{\alpha_0}^{\alpha_1} \left(\frac{\partial \mathcal{F}}{\partial \mathbf{x}}\delta(\alpha) + \frac{\partial \mathcal{F}}{\partial \mathbf{x}'}\delta'(\alpha) \right) d\alpha \\ 0 &= \int_{\alpha_0}^{\alpha_1} \left(\frac{\partial \mathcal{F}}{\partial \mathbf{x}} - \frac{d}{d\alpha}\frac{\partial \mathcal{F}}{\partial \mathbf{x}'} \right) \delta(\alpha) d\alpha + \left[\frac{\partial \mathcal{F}}{\partial \mathbf{x}'}\delta(\alpha) \right]_{\alpha_0}^{\alpha_1}\end{aligned}$$

Imposing the boundary conditions $\delta(\alpha_1) = \delta(\alpha_0) = 0$ (i.e. the system is not disturbed at the boundary of the interval), the disturbance function drops out of the equation entirely. The function $\mathbf{x}(\alpha)$ that meets the condition (2-10) extremises (2-8).

$$\frac{\partial \mathcal{F}}{\partial \mathbf{x}} - \frac{d}{d\alpha}\frac{\partial \mathcal{F}}{\partial \mathbf{x}'} = 0 \tag{2-10}$$

An example of this optimisation is found in every-day mechanics. Taking time t as α and the action, the difference between kinetic and potential energy in a system with general coordinates $\mathbf{q}(t)$, as the functional $\mathcal{F} = \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) = T(\mathbf{q}, \dot{\mathbf{q}}) - V(\mathbf{q})$, satisfaction of the optimality criterion (2-10) results in the Lagrange equations:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_j}\right) - \frac{\partial T}{\partial q_j} + \frac{\partial V}{\partial q_j} = 0$$

It is the relation the state must satisfy to keep the energy in the system constant. As conservation of energy is a law of nature, this optimisation result describes the (mechanical) behaviour of the system when no external forces are applied.

2-2-2 Optimal spring force as function of joint force

For the Cartesian robot the electric energy consumption is the cost function to be optimised. Combining Equations (2-5) to (2-7), the expression of the electric energy consumption per cycle (2-11) has the same form as (2-8).

$$E = \sum_{i \in \{\text{down,up}\}} \int_{y_{i,0}}^{y_{i,1}} \frac{R}{k_\text{t}^2 n_\text{s}^2} \frac{\left(F_{\text{j},i} - F_\text{s}\right)^2}{\dot{y}_i} + \left(F_{\text{j},i} - F_\text{s}\right) + \frac{P_\text{oh}}{\dot{y}_i} ds \tag{2-11}$$

Negative $(F_{\text{j},i} - F_\text{s})$ results in energy gain, while any $\|F_{\text{j},i} - F_\text{s}\| > 0$ brings energy loss. The total cycle energy is obtained by taking the sum over the trajectories $i \in \{\text{down,up}\}$. Since $y_{\text{down},0} = y_{\text{up},1}$ and $y_{\text{up},0} = y_{\text{down},1}$, the sum can be taken into the integral and the result of (2-10) can be applied directly.

$$\mathcal{F}(y, F_\text{s}(y)) = \frac{R}{k_\text{t}^2 n_\text{s}^2} \frac{\left(F_{\text{j,d}} - F_\text{s}\right)^2}{\dot{y}_\text{d}} + \left(F_{\text{j,d}} - F_\text{s}\right) + \frac{P_\text{oh}}{\dot{y}_\text{d}} - \left(\frac{R}{k_\text{t}^2 n_\text{s}^2} \frac{\left(F_{\text{j,u}} - F_\text{s}\right)^2}{\dot{y}_\text{u}} + \left(F_{\text{j,u}} - F_\text{s}\right) + \frac{P_\text{oh}}{\dot{y}_\text{u}} \right)$$

In this case $\mathcal{F}$ does not depend on $F_\text{s}'(y)$, which is the derivative of $F_\text{s}(y)$ with respect to y. Therefore the optimal spring characteristic function is obtained by solving $\frac{d\mathcal{F}}{dF_\text{s}} = 0$.

$$\frac{d\mathcal{F}}{dF_\text{s}} = -2\frac{R}{k_\text{t}^2 n_\text{s}^2} \frac{F_{\text{j,d}} - F_\text{s}}{\dot{y}_\text{d}} - 1 - \left(-2\frac{R}{k_\text{t}^2 n_\text{s}^2} \frac{F_{\text{j,u}} - F_\text{s}}{\dot{y}_\text{u}} - 1 \right) = 2\frac{R}{k_\text{t}^2 n_\text{s}^2} \left(\frac{F_{\text{j,d}} - F_\text{s}}{-\dot{y}_\text{d}} + \frac{F_{\text{j,u}} - F_\text{s}}{\dot{y}_\text{u}} \right)$$

When set to zero,

$$\begin{aligned} \frac{F_{\text{j,d}} - F_\text{s}}{-\dot{y}_\text{d}} + \frac{F_{\text{j,u}} - F_\text{s}}{\dot{y}_\text{u}} &= 0 \\ \frac{F_{\text{j,d}}\dot{y}_\text{u} - F_{\text{j,u}}\dot{y}_\text{d}}{-\dot{y}_\text{d}\dot{y}_\text{u}} - \frac{-\dot{y}_\text{d} + \dot{y}_\text{u}}{-\dot{y}_\text{d}\dot{y}_\text{u}} F_\text{s} &= 0 \end{aligned} \tag{2-12}$$

means that, unless $\dot{y}_\text{d}\dot{y}_\text{u} = 0$,

$$F_\text{s}\left(F_{\text{j,d}}(y), F_{\text{j,u}}(y), \dot{y}_\text{d}(y), \dot{y}_\text{u}(y)\right) = \frac{F_{\text{j,d}}(y)\dot{y}_\text{u}(y) - F_{\text{j,u}}(y)\dot{y}_\text{d}(y)}{-\dot{y}_\text{d}(y) + \dot{y}_\text{u}(y)} \tag{2-13}$$

Since $\dot{y}_\text{d}$ in itself is per definition negative (going down), Equation (2-13) is the different force profiles averaged over the different velocity profiles. The same result holds in the rotational case, when using joint torques and angular velocities instead of their linear counterparts. The result is also extendable to more trajectories by taking for i multiple downward and upward motions.

As was the case with the state equations (2-4), there is no proper solution when $\dot{y} = 0$. This problem is solved the same way, by introducing small initial velocities $\dot{y}_0 = \pm\epsilon$.

2-2-3 Implications of non-ideal circumstances

Any loss terms that can be brought outside the drive train can be included in the expression of the joint force, such that the results of the previous subsection remains valid. Of the remaining factors, any additional terms which do not hold a nonlinear relationship with F_s will cancel

out. This applies to terms such as the no-load current (2-3). Therefore, when considering the full model of Section 2-1, only the efficiency factor C (2-2c) remains. Including C in the energy function (2-11), the optimality criterion (2-12) would look like:

$$\sum_i \left(-2\frac{R}{k_\mathrm{t}^2 n_\mathrm{s}^2} \frac{F_{\mathrm{j},i} - F_\mathrm{s}}{C_i^2 \dot{y}_i} - \frac{1}{C_i} \right) = 0 \tag{2-14}$$

The term C_i is constant as long as $F_{\mathrm{j},i} - F_\mathrm{s}$ and $\dot{y}_i$ do not change sign. The velocity will not change sign during a half-cycle trajectory, however $F_{\mathrm{j},i} - F_\mathrm{s}$ might, also depending on F_s.

This nonlinear dependency on F_s makes it impossible to extract the spring force from (2-14) the way (2-13) was obtained from (2-12). However, when considering the optimal spring force by definition of (2-13) knowing that $\dot{y}_\mathrm{d}$ and $\dot{y}_\mathrm{u}$ will always have opposite signs, the spring force, being the weighed average of the two joint forces, will always be in between the two joint forces. That means that the sign of $F_{\mathrm{j,d}}(y_k) - F_\mathrm{s}(y_k)$ and $F_{\mathrm{j,u}}(y_k) - F_\mathrm{s}(y_k)$ will be opposite for any position y_k. Since C_i only depends on the sign difference between $F_{\mathrm{j},i}(y_k) - F_\mathrm{s}(y_k)$ and $\dot{y}_{i,k}$, C_d will always equal C_u such that it cancels out in (2-14) and (2-13) also holds for the full model described in Section 2-1.

2-3 Joint and motor force optimisation

The result of the previous section was an analytical expression of the optimal spring force as function of joint force and velocity, both of which are a function of position. What remains to be done is to find the optimal joint forces and velocities such that the total energy consumption of the system is minimised.

The energy costs depend for the major part on the motor torque. However, the motor torque depends on the spring force which in return depends on the state trajectory. Is is much more straightforward to calculate the forces and torques from the state trajectory than the reverse. As the forces appear in the top row of (2-4), the trajectory parameter for the optimisation is chosen to be the acceleration divided by the velocity $\dot{y}^{-1}\ddot{y}$. In the sequel it will be simply called by the name “acceleration profile”.

2-3-1 Optimisation

Indirect optimisation is done in the sense that the state space is not parameterised. Instead the system states are obtained by integrating the state equations using the acceleration profile along the trajectory parameter, which in this case is distance (usually time is used). All other state variables relate to the chosen optimisation variable by the dynamic equations of motion, such that only few variables need to optimised and the system dynamics need not be imposed as constraints. The result is a low dimensional, minimally constrained, but very nonlinear optimisation problem. The state equations (2-4) are integrated to obtain speed, time and joint force. The optimal spring force and remaining motor torque can then be deduced from the computed joint force.

The cost function of the optimisation is stated explicitly in Section 2-3-1-1. The trajectory is not optimised at every distance instant. The trajectory vector to be optimised consists

of only a small number of parameters. In between the trajectory is interpolated. This will is discussed in Section 2-3-1-2. Because of the severe nonlinearity of the problem, the way multi-start is used is described in Section 2-3-1-3. Section 2-3-1-4 discusses the constraints the optimisation is subject to and in Section 2-3-1-5 the used numerical integration method is discussed.

2-3-1-1 Cost function

The cost function is simply the required electric energy, as was already presented in Section 2-2, but this time using the full model of Section 2-1.

$$E = \int T_{\mathrm{m}} n_{\mathrm{s}} + \frac{I^2 R}{\dot{y}} ds + P_{\mathrm{oh}} t_{end} \tag{2-15}$$

The goal is to save as much energy as possible for the overheads. Rearranging (2-15), the retrieved power is given by:

$$P = -\frac{\int T_{\mathrm{m}} n_{\mathrm{s}} + \frac{I^2 R}{\dot{y}} ds}{t_{end}} \tag{2-16}$$

Ideally it is desired to maximise P. However, as the final time t_{end} is the result of state integration, (2-16) is too nonlinear to be used as cost function. Instead an estimation of P_{oh} is made and the resulting energy shortage (2-15) is minimised.

The overhead power is consumed mainly in the processor, sensors and end effector (i.e. gripper). For the Cornell Ranger the low overhead power was estimated at 8 W [16]. A microprocessor and three rotary encoders for the final 3 DoF arm would need about 0.65 W [4]. The remaining electronics will not be entirely free of loss, 1 W for arm control will probably still be quite optimistic. A gripper requiring minimal actuation energy is currently being designed for the project. This gripper expected to use less than 1 W of power as well. With a safety factor 2, and in between this back-of-the-envelope estimation and Rangers estimated overheads, P_{oh} is set to 4 W.

2-3-1-2 Trajectory parameterisation

By necessity, as described in Section 2-1, the regarded pick-and-place cycle is divided in the downward and the upward motion. The distance parameters over which is integrated are denoted $\mathbf{y}_{\mathrm{down}}$ and $\mathbf{y}_{\mathrm{up}}$ respectively, resulting in the total cycle trajectory $\mathbf{y} = \begin{bmatrix} \mathbf{y}_{\mathrm{down}} & \mathbf{y}_{\mathrm{up}} \end{bmatrix}$.

To obtain an integration interval $\Delta y = 10^{-3}$ m, $N = 1001$ is chosen. Optimising the acceleration profile at each small position step heavily taxes computation. Instead the trajectory is parameterised at a small number of positions. This number is chosen at 9 for down and up separately. $x_{\mathrm{down},0}$ denotes the acceleration over velocity at $y_{\mathrm{down},0}$, $x_{\mathrm{down},9}$ at $y_{\mathrm{down},N-1}$. The other values of $\mathbf{x}_{\mathrm{down}}$ are equally spaced in between. The same holds for $_{\mathrm{up}}$. During trajectory evaluation, the intermediate acceleration over velocity values are obtained by interpolation. For computational ease, this interpolation is done linearly. The total cycle acceleration profile is encoded by $\mathbf{x} = \begin{bmatrix} \mathbf{x}_{\mathrm{down}} & \mathbf{x}_{\mathrm{up}} \end{bmatrix}$, which has a total length of 18.

At the final positions, $y_{\mathrm{down},N} = 0$ and $y_{\mathrm{up},N} = 1$ respectively, both acceleration and velocity should be zero. The final values $x_{y_N} := 0$ are required for the state integration (Section 2-3-1-5), but because they are fixed, they can safely be left out of the parameterisation.

2-3-1-3 Multistart optimisation

The optimisation problem is very nonlinear. A multistart implementation is required to cover as much of the solution space as possible. If enough local minima are explored, the best can be selected. It is then to be hoped that the lowest minimum will be a reasonable approximation of the global optimum.

Though MATLAB has a `Multistart` option for optimisations, the following manual implementation yielded better results:

The initial elements of $\mathbf{x}$ are chosen randomly between 0 and 3. This is quite arbitrary, but both a larger range and a range that included negative numbers yielded worse results. A large range is desired for optimal exploration of the solution space. The optimisation is performed 500 times.

2-3-1-4 Constraints

The optimisation problem is formulated such that most constraints are kept inherently. The only constraints added as such to the solver, are the zero end-state velocities: $\dot{y}(y_{\text{end}}) = 0$ at arrival position $y_{\text{end}} = 0$ when going down respectively $y_{\text{end}} = 1$ when going up. Zero end state acceleration is enforced by defining the optimal spring such that the resultant force is zero: $F_{\text{s}}(y_{\text{end}}) = F_{\text{j}}(y_{\text{end}})$.

Aside from these endpoint constraints ensuring state equilibrium at the cycle extrema, there is the important constraint on the velocity $|\dot{y}| > \epsilon$ to ensure numerical stability. In both state equations and computation of the energetic cost, the velocity is found in the denominator. Division by zero may never occur. In the optimisations, ϵ is set at 10^{-3} m/s. A trajectory where $\dot{y} = \pm\epsilon$ will definitely not be optimal as the cycle time will be large as a result, therefore a crude state approximation suffices around $\dot{y} = \pm\epsilon$. During integration, whenever the acceleration at one step causes the velocity at the next step to have crossed the $\pm\epsilon$ boundary, the velocity of the next step is set at $\pm\epsilon$ (dependent on the direction of motion) and time and acceleration are corrected accordingly. How exactly is described in the next subsection.

Additionally, bounds can optionally be imposed on the velocity, acceleration and/or force. The choice of $\dot{y}^{-1}\ddot{y}$ as trajectory parameter results in these bounds all needing to be formulated as nonlinear constraints. The way the entire optimisation problem is implemented, these constraints are easily added. But in they complicate the optimisation process enormously. Only adding bounds to the velocity resulted in excessively large optimisation times. Adding multiple such bounds may very well make the problem practically unsolvable. Therefore no such bounds were set.

2-3-1-5 Numerical integration

For numerical integration of the system states Runge-Kutta 4^{th} method (RK4) is chosen. Its approximation error is much smaller than for Euler's method, yet it is still easy to implement. To suite the current application, two modifications are made to the algorithm.

The standard RK4 works as well for integration over distance as if it were time. Actually the base parameter (distance/time) does not matter as long as the state equations are defined accordingly. Yet aside from that the integration method is still applied in a non-standard way.

Regular numerical state integration would look like

$$\begin{bmatrix} \dot{y}_{k+1} \\ t_{k+1} \end{bmatrix} = \begin{bmatrix} \dot{y}_k \\ t_k \end{bmatrix} + \mathcal{G}\left(f\left(y, \begin{bmatrix} \dot{y} \\ t \end{bmatrix}, T_\mathrm{j}\right), \Delta y\right) \tag{2-17a}$$

$$y_{k+1} = y_k + \Delta y \tag{2-17b}$$

where $\mathcal{G}(\cdot,\cdot,\cdot)$ is some integration method to approximate the state increment and $f(\cdot,\cdot)$ is the function describing the state derivative (2-4). However, instead of knowing the joint force $T_\mathrm{j}(y)$ in f required to compute $\frac{d\dot{y}}{dy}$, $\frac{d\dot{y}}{dy}(y)$ is known since this is the same as acceleration over velocity, which is the parameter vector of the optimisation. So instead of simply having

$$\frac{d}{dy}\begin{bmatrix} \dot{y} \\ t \end{bmatrix} = f\left(y, \begin{bmatrix} \dot{y} \\ t \end{bmatrix}, T_\mathrm{j}\right)$$

this relation can only be used for the bottom row, while $\frac{d\dot{y}}{dy}(y)$ is obtained directly from interpolation of the trajectory parameter vector.

The other important concern is the velocity constraint $|\dot{y}| > \epsilon$ (Section 2-3-1-4). Each integration step (2-17) that results in $\dot{y}$ crossing the $\mathrm{sign}(\dot{y})\epsilon$-line, the step is recomputed using Euler's method. By redefining $\dot{y}_{k+1} := \pm\epsilon$, the maximum allowable $\frac{d\dot{y}}{dy}(y_k) = \frac{\dot{y}_k \pm \epsilon}{\Delta y}$. Then the with Euler's method computed t_{k+1} still holds. This inverse integration step cannot be computed using the Runge-Kutta method, since it uses multiple interdependent intermediate values of $\frac{d\dot{y}}{dy}$. This backward correction has no risk of occurring in the final optimal trajectory, as any solution with $\dot{y}_k := \pm\epsilon, k \neq 0$ will be suboptimal.

2-3-2 Results

Figure 2-2 shows the 15 optimal spring characteristics resulting in the least energy consumed out of 500 optimisations. Figure 2-3 shows the same spring characteristics but for the solutions showing large bumps. The characteristics clearly show they are composed of eight equally spaced piecewise smooth parts. At the points where the characteristic is discontinuous in its derivative, the jump in its derivative can be large, which results in undesired bumps in the trajectory. Therefore solutions with large first order discontinuities in the spring characteristic have been discarded.

Figure 2-4 shows the more detailed trajectory results to the optimal spring characteristic indicated in black in Figure 2-3. The left and right plots in Figure 2-4 show the same variables, only on the left they are plotted against time whereas on the right the distance is on the x axis. Note that the bottom two plots have double y axes. The left axis corresponds to the motor torque in Nm, the blue line, which is the difference between the joint forces in yellow and the spring force in red, scaled by the spindle reduction and transmission efficiency. The joint and spring forces are given in N on the right axis.

The top row of Table 2-3 shows the energy results to the result of Figure 2-4. The total energy cost is E from (2-15) including the overhead term. The recovered energy, without the

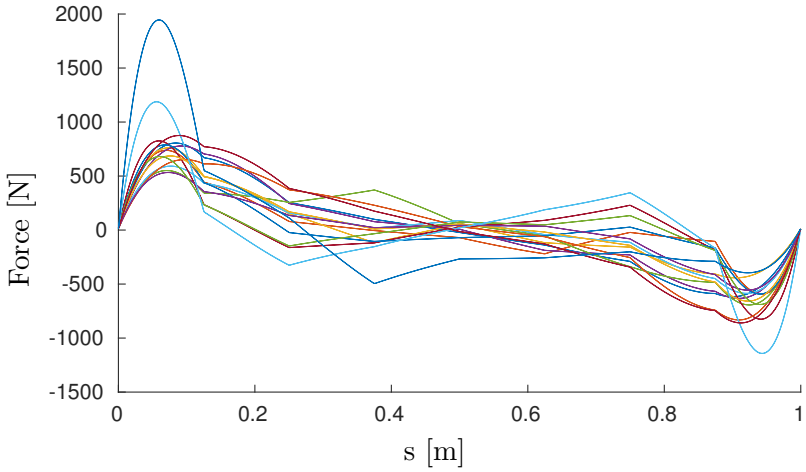


Figure 2-2: Fifteen optimised springs causing least system energy consumption.

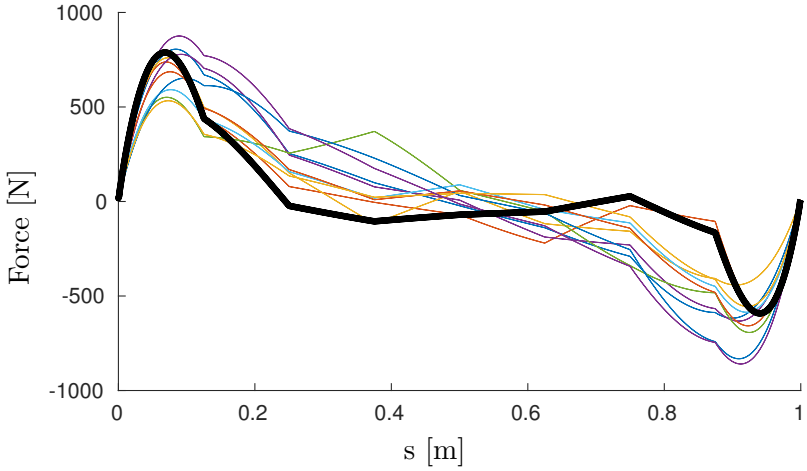


Figure 2-3: Springs with least system energy consumption, bumpy results removed. The optimal spring of the set is indicated by the bold black line.

Table 2-3: Energy results to the optimal trajectories with analytical optimal springs

	cycle time [s]	mechanical loss [J]	I^2R loss [J]	total energy cost [J]	recovered energy [J]	recovered power [W]
with reflected inertia						
smooth trajectory	2.02	1.31	0.43	0.01	8.07	4.00
bumpy trajectory	1.87	1.39	0.58	-0.36	7.84	4.19
without spring	4.85	4.06	10.05	23.71	-4.30	-0.89
without reflected inertia						
smooth trajectory	2.08	1.33	0.57	0.40	7.91	3.81
bumpy trajectory	1.75	1.28	0.26	-1.26	8.27	4.72
without spring	2.00	3.76	2.53	4.40	3.61	1.80

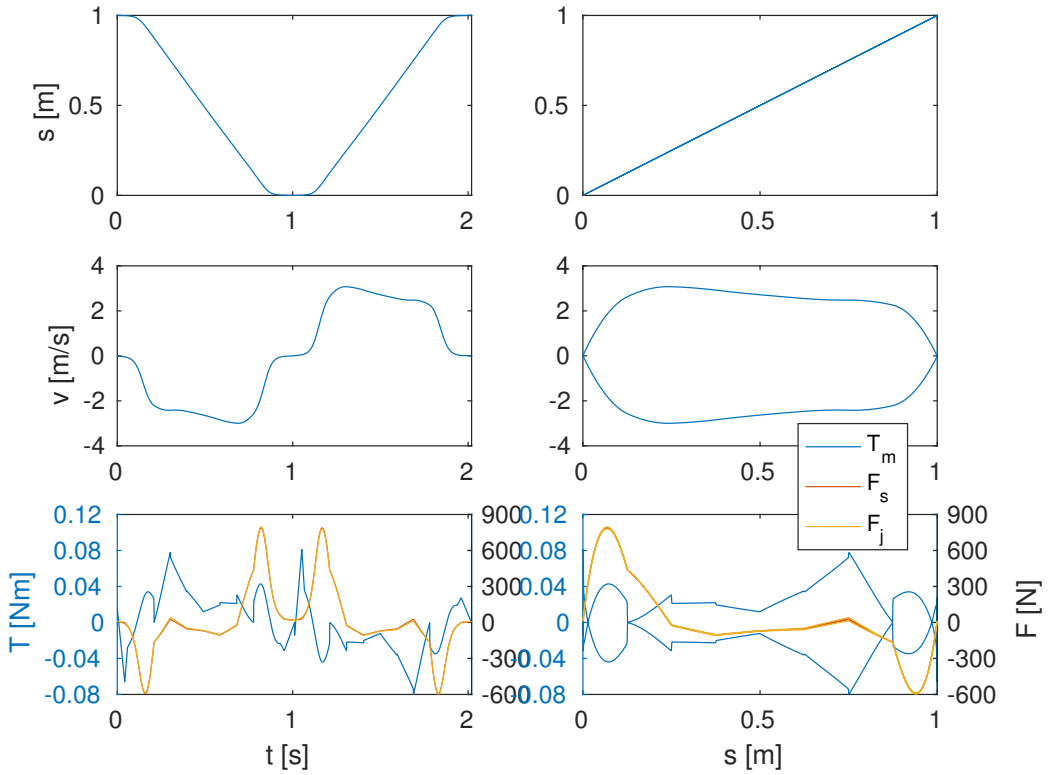


Figure 2-4: Optimised trajectory, spring and residual motor torque, on the left plotted against time, on the right against distance. The bottom plots show in yellow and red the joint forces and optimised spring characteristic in N on the right axis, and in blue the residual motor torque in Nm on the left axis.

overhead term, is also given. The recovered power is P from (2-16). For comparison, the table also shows the energy results to the optimal bumpy case as well as the case without spring. The bumpy trajectory recovers less energy, but because of the shorter cycle time the reduced overhead cost results in a smaller total energy cost and more power is recovered. Without a parallel spring it is seen it is not possible to recover any energy, even without overheads.

With spring, larger overall joint torques are achieved with only a fraction of the previous mechanical and actuator losses. With the smooth optimal parallel spring a mechanical energy loss reduction of over a factor 3 is observed with respect to the case without spring. The change in electric energy consumption is even more dramatic. Due to the reduced actuator torques, the spring has reduced the I^2R loss by over a factor 20. Without spring the actuator heating loss dominated the mechanical loss, with spring the reverse is the case.

The remaining mechanical losses originate partially from the transmission efficiency. Coulomb and viscous friction would also appear in this term, but those are currently zero. A less direct mechanical loss factor is the drive train reflected inertia, which increases the effective mass that to be accelerated (2-2a). Currently the reflected inertia term is large. For the spindle drive the reflected inertia is much larger than it will be in the polar arm with rotational joint, due to the heavy spindle shaft and the relatively large spindle reduction ratio. Therefore the bottom half of Table 2-3 shows the results when drive train inertia is not taken into account, also important for later comparison of results.

Without reflected inertia the optimisation returns a larger number of bumpy trajectories, which show much larger first order discontinuities than observed in Figure 2-2. This is because now large accelerations have a much lower energetic penalty. The smooth trajectory without reflected inertia performs slightly worse compared to the trajectory with reflected inertia mainly due to less optimised trajectories showing smooth behaviour. The bumpy trajectory outperforms all others in every respect except its unacceptable bumpyness.

Without reflected inertia the difference in energetic performance is larger between the smooth and the bumpy result, but on average the results with spring with and without reflected inertia are very similar. The analytical optimal spring is very capable to compensate for reflected drive train inertia. The overhead power drain is the major remaining loss.

Without spring the differences are large. Without reflected inertia the I^2R losses are much lower and the cycle time is much shorter. As a result, some energy is actually recovered.

The trajectories of Table 2-3 not shown in this section can be found in Appendix C-1.

2-4 Physical parallel spring

In the previous section the large benefit of a proper parallel spring mechanism was shown. However for practical application, the spring force must be delivered by a physical mechanism. The mechanism preferably has low complexity as complexity generally adds to energy loss. Therefore this section will present the design of a relatively simple spring mechanism with a force characteristic similar to optimal spring that was found in Section 2-3-2 and test its performance using the same trajectory optimisation.

The mechanism parameters used in this section are chosen for reasons of mathematical clarity and easy comparison of results in order to obtain important insights in the mechanism principles which can be used for design of the real robot arm discussed in the next chapter. This hypothetical spring mechanism for linear motion is never meant to be built. For the real arm, design of its rotational counterpart will be discussed in Chapter 3.

In Figure 2-3 all close to optimal springs are seen to have similar form of force characteristic. The arm is balanced in the end positions: in the top position without mass and in the bottom position with mass. Near the endpoints a rapidly growing force pulls towards the other endpoint while around the middle of the distance forces are small. The resonant mechanism described by Plooij in [17] shows similar peaks at the quarter distances, while at the endpoints and in the middle the torque (since it is a rotational mechanism) is zero. For our vertical case additional gravity compensation is required. Therefore the spring mechanism for the vertical motion consists of the following two parts:

1. A rectilinear “resonant” mechanism as described by Babitsky and Shipilov[18, Chapter 1]. This mechanism has equal potential energy at the endpoint positions. Furthermore, both positions are in equilibrium. In between the potential energy of the mechanism is lower. Once disturbed from one equilibrium endpoint, the mechanism is pulled towards the other. In lossless ideal world theory without external forces, the mechanism would need no energy to keep moving back and forth between the two positions indefinitely.
2. A linear spring balancing the gripper without package in the top position and with package in the bottom position. This additional spring is required to compensate for

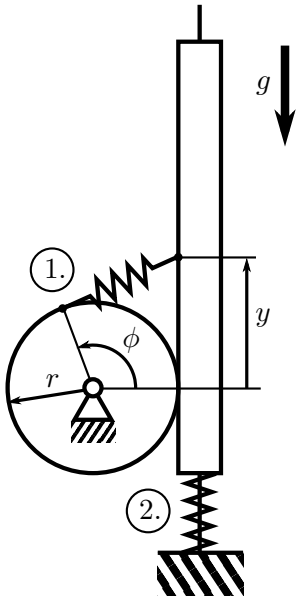


Figure 2-5: The 'resonant' mechanism (1.) consists of a wheel connected to a slider in two ways: by a rolling contact, on the right side of the wheel, and by a spring which is at its rest length when both ϕ and y are zero. The endpoint balancing mechanism (2.) consists of a linear spring connected to the slider which keeps the slider in position at the pick and place positions.

the external force present: gravity. By only balancing the mass in the end points, this spring also helps the motion to the other end.

The combined mechanism is shown in Figure 2-5. Both springs are linear. Spring (1.) is by definition a tension spring. Spring (2.) can be either be a tension spring or a compression spring, dependent on whether it is placed at the top of the actuated slider or at the bottom as depicted in Figure 2-5.

The resonant mechanism (1.) has stable positions in all cases $\phi = k \cdot 2\pi$, with k and arbitrary integer. The system has one degree of freedom which can be chosen either as y or as ϕ, which are related by:

$$y = r\phi \tag{2-18}$$

Here r is the effective wheel radius as depicted in Figure 2-5. Since all other forces in the system are a function of y, it is chosen as the general coordinate. Notice that there is a shift in the definition where $y = 0$ compared to the previous section. Now the package will be picked up at $y = \frac{1}{2}$ m and put down at $y = -\frac{1}{2}$ m. This shift of reference frame facilitates computation of the mechanism forces but has no influence on the other calculations. To ensure $y = \pm\frac{1}{2}$ corresponds with $\phi = \pm 2\pi$, the effective wheel radius $r = \frac{1}{4\pi}$. For comparative reasons the inertia of the wheel is neglected.

2-4-1 Mechanical model

The two parts of the mechanism are independent of the other. Therefore they can and will be described separately in this section before they are combined into a single spring energy and force expression for the full mechanism. Both springs in the mechanism are linear.

Resonant mechanism The energy stored in the spring ((1.) Figure 2-5) is given by (2-19).

$$E_{s1} = \frac{1}{2} k_{s1} u_{s1}^2 \tag{2-19}$$

Because of the mechanism geometry, the elongation u_{s1} is

$$u_{s1} = \sqrt{\Delta x^2 + \Delta y^2} \tag{2-19a}$$
$$\Delta x = r(1 - \cos\phi) \tag{2-19b}$$
$$\Delta y = y - r\sin\phi \tag{2-19c}$$

Then the spring force $F_{s1} = -\frac{\partial E_{s1}}{\partial y}$ with $\phi = \frac{y}{r}$ results in

$$F_{s1} = k_{s1} y \left(\cos\left(\frac{y}{r}\right) - 1 \right) \tag{2-20}$$

The spring stiffness k_{s1} can be chosen freely. Larger k_{s1} results in a steeper energy function and larger forces, resulting in a faster response. However larger speeds also tend to increase energy losses. Therefore it will have to show from the optimisation which spring stiffness is actually optimal.

Endpoint balancing spring The spring force of the endpoint balancing spring is straightforward in calculation since it is linear in y.

$$F_{s2}(y) = -k_{s2}(y - y_0) \tag{2-21}$$

The boundary conditions for endpoint balancing

$$F_{s2}\left(-\frac{1}{2}\right) = g\left(m_{\text{slider}} + m_{\text{gripper}} + m_{\text{package}}\right)$$
$$F_{s2}\left(\frac{1}{2}\right) = g\left(m_{\text{slider}} + m_{\text{gripper}}\right)$$

dictate

$$k_{s2} = g m_{\text{package}} \tag{2-21a}$$
$$y_0 = \frac{m_{\text{slider}} + m_{\text{gripper}} + \frac{1}{2} m_{\text{package}}}{m_{\text{package}}} \tag{2-21b}$$

Correspondingly, the spring energy is given by

$$E_{s2} = \frac{1}{2} k_{s2} \left(y - y_0\right)^2 \tag{2-22}$$

Combined spring mechanism Figure 2-6 shows on the left the energy and force characteristics of the two separate mechanism parts. Gravity for both the loaded and the unloaded case is indicated by the thin yellow lines. The right half of the figure shows the result when summing the two mechanism parts and the gravity. The bottom position $y = -\frac{1}{2}$ is in equilibrium in the loaded case, when going down, the top position $y = \frac{1}{2}$ is in equilibrium in the unloaded case, when going up.

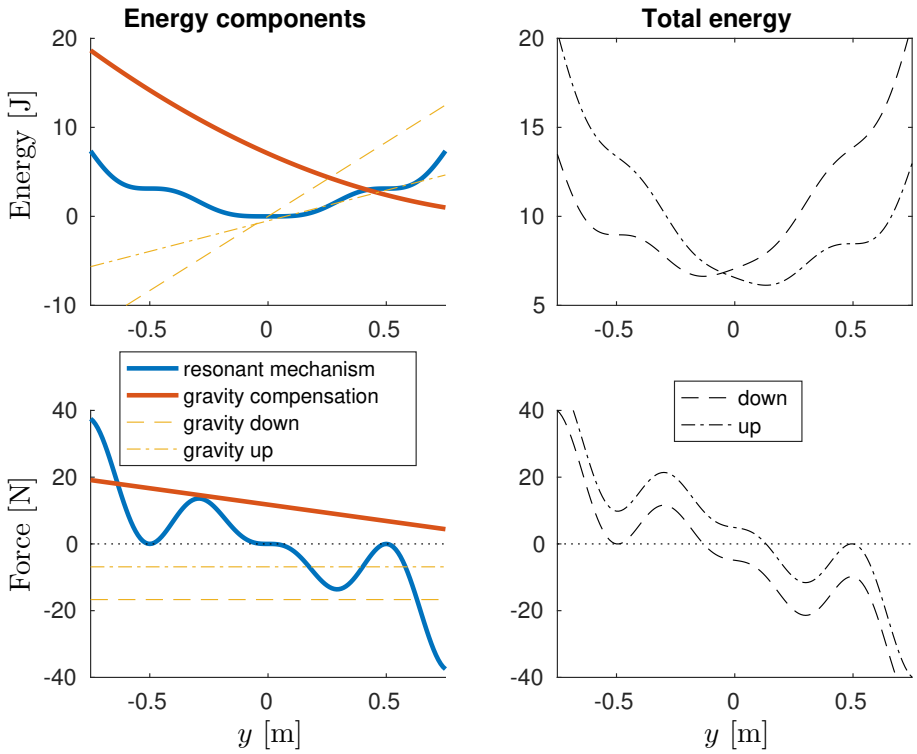


Figure 2-6: Energy and force characteristic of both 'resonant' mechanism and endpoint balancing spring. The thin yellow lines indicate the gravitational potential energy and force respectively for the load carrying case (dashed) and the unloaded case (dash-dotted). The right shows the sum of the parts on the left. When going down the bottom position $y = -0.5$ is stable, the same holds for the top position $y = 0.5$ when going up. For large enough k_{s1}, an additional global energy minimum lies around $y = 0$.

2-4-2 Resonant mechanism optimisation

The parameter vector has to be expanded by one variable: the spring stiffness k_{s1} (2-20), from now on simply called k_s. The function computing the optimal spring force from the total joint forces and trajectory (2-13) is replaced by the mechanism spring characteristic $F_{s,mech}(y, k_s) = F_{s1}(y, k_s) + F_{s2}(y)$ from (2-20),(2-21). However, the attempt to optimise both the trajectory and the mechanism spring stiffness simultaneously did not yield valid results. Therefore the following two alternative methods were explored.

1. The optimal trajectory found in Section 2-3-2 is used as a starting point. Subsequent optimisation of the mechanism spring stiffness and trajectory is done iteratively.

It is observed that the attempt to optimise a trajectory from an already decent initial trajectory does not lead to a new improved result. Presumably the valleys between the ridges in the solution space deepen towards the region of the optimum. Once in one of the deep valleys, the optimisation cannot come out. This is avoided by starting each trajectory optimisation with the random number initialisation as described in Section 2-3-1-3. So starting from an optimised trajectory, the spring is optimised to the trajectory. Then an entirely new trajectory is optimised to the spring. This set of optimisations is repeated 50 times.

2. The trajectory is optimised for a range of fixed spring stiffnesses. When taking the drive train reflected inertia into account, a range of $k_s = 0, 10, \ldots, 1000$ is chosen. When not taking reflected inertia into account, $k_s = 0, 1, \ldots, 25$ suffices. In case of $k_s = 0$ there is only the endpoint balancing spring.

This method covers the solution space in a more uniform way. The range of spring stiffnesses is chosen based on the range of quite well performing spring stiffnesses found in the results of the first method. The really well performing trajectories are only the outliers of a batch of optimisation results. Therefore, in order to compare the performance of the different spring stiffnesses it is important to optimise the trajectory a large number of times for each stiffness value. For each different value of the spring stiffness for the case without drive train reflected inertia, 1000 trajectory optimisations are performed.

2-4-3 Results

Figure 2-7 shows the result based on the result of Section 2-3-2. The optimal spring stiffness was found to be $k_s = 82.5$. The bottom plots of Figure 2-7 show that during most of the trajectory the joint force is delivered for the main part by the spring, except for at the start and end of the trajectory, where the motor torque characteristic show peaks for extra acceleration at the start and extra deceleration near the end. Despite of this extra boost, the accelerations and therefore velocity stays low, resulting in a long cycle time of over 4 s. The top row of Table 2-4 shows the energy values corresponding to the trajectory.

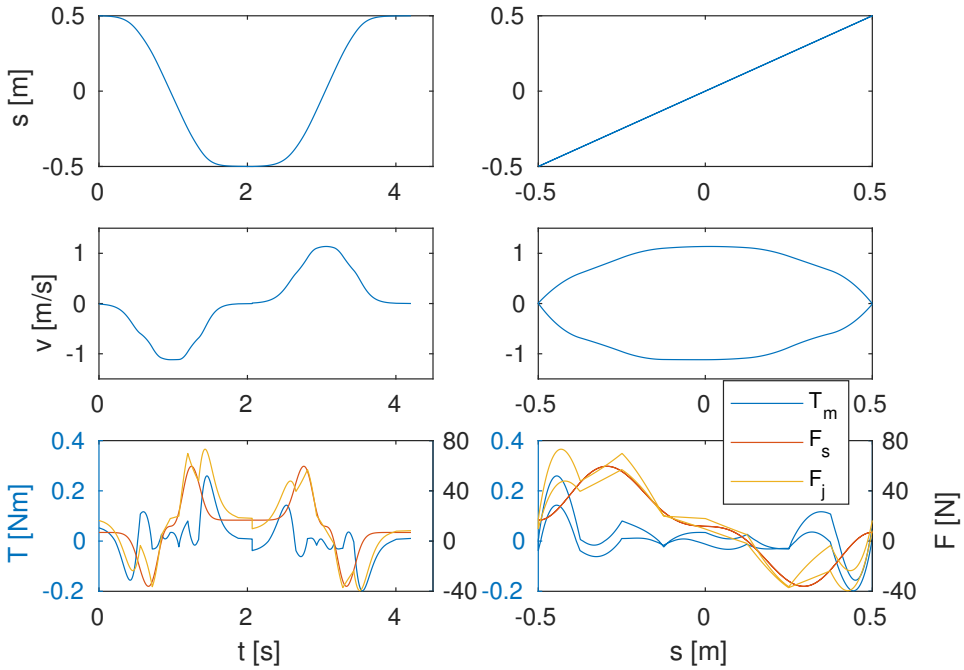


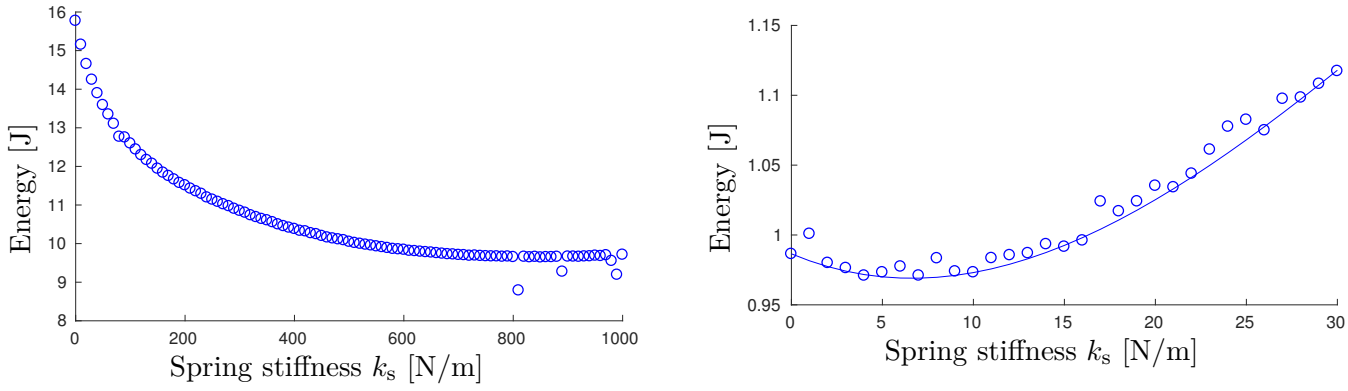
Figure 2-7: Optimised trajectory obtained when using optimal trajectory of Figure 2-4 as the starting point, $k_s = 82.5$. On the left plotted against time, on the right against distance. The bottom plots show in yellow and red the joint forces and spring characteristic in N on the right axis, and in blue the residual motor torque in Nm on the left axis.

When not taking reflected inertia into account, less force/torque is required to achieve a faster system response. This is seen when optimising the spring mechanism and trajectory without reflected inertia. Then the optimum is found for a lower spring stiffness, while the cycle takes only 2.26 s and all losses are less, resulting in over twice as much energy recovered (Table 2-4, for the trajectory see Appendix C-2).

Optimising from scratch results in improved energy return both with and without reflected inertia. Though it is not so straightforward to select the best result with the most suitable spring stiffness. Figure 2-8 shows the minimal energy cost for the ranges of different spring stiffnesses with and without reflected inertia. In the case without inertia (Figure 2-8b), a polynomial fit is drawn to predict the general trend and the minimum appears to be around $k_s = 7$. But since the curve is very level in the region, a mechanism with $k_s = 5$ or 9 may perform equally well. In the current data set, the result costing the least energy was found for $k_s = 4$.

From Figure 2-8a it is even less clear which spring stiffness is optimal. On this scale the figure shows very smooth looking trend suggesting a large spring stiffness is best, but above $k_s \approx 800$ it is all about the same, perhaps slightly worse when continuing past the $k_s = 1000$. However, at some high values for k_s there are suddenly outliers below the general trend line. Since the data making up the figure are the minimum cost values over a thousand optimisations per k_s, the outliers suggest the nice smooth trend does not show the real minimum. Actually, it becomes doubtful whether the cost function is still returning the value that needs to be minimised. After all, the cost function minimises the energy cost assuming a 4 W overhead power drain. When the system with large reflected inertia favours low accelerations leading to long cycle times, the overhead cost starts dominating the total cost to an extent that shorter cycle times become more important than large energy recovery.

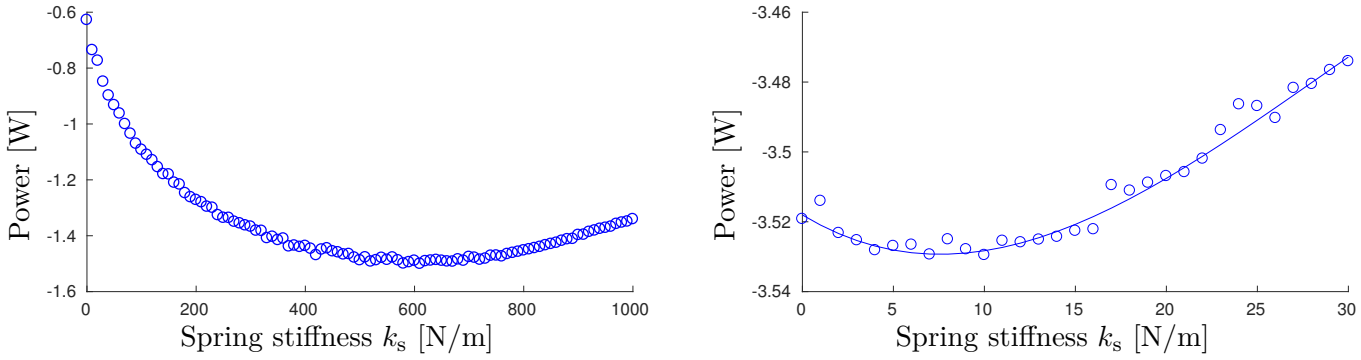
For the case without reflected inertia the minimum cost should still be a proper measure since the energy that is recovered over a cycle is close to the 4 W. However for the case with reflected inertia nowhere near enough energy is recovered to support this large a power drain. A different performance metric is tried. Instead of the minimum total cost, Figure 2-9 shows



(a) With reflected inertia

(b) Without reflected inertia

Figure 2-8: Minimal energy cost for the ranges of resonant mechanism spring stiffnesses. In **(b)** the bottom line trend is estimated by a third order polynomial.



(a) With reflected inertia

(b) Without reflected inertia

Figure 2-9: Minimal power consumption for the ranges of resonant mechanism spring stiffnesses, overhead power not taken into account. In **(b)** the bottom line trend is estimated by a third order polynomial.

for the ranges of spring stiffnesses the minimum power consumption (2-16) when not taking the overheads into account.

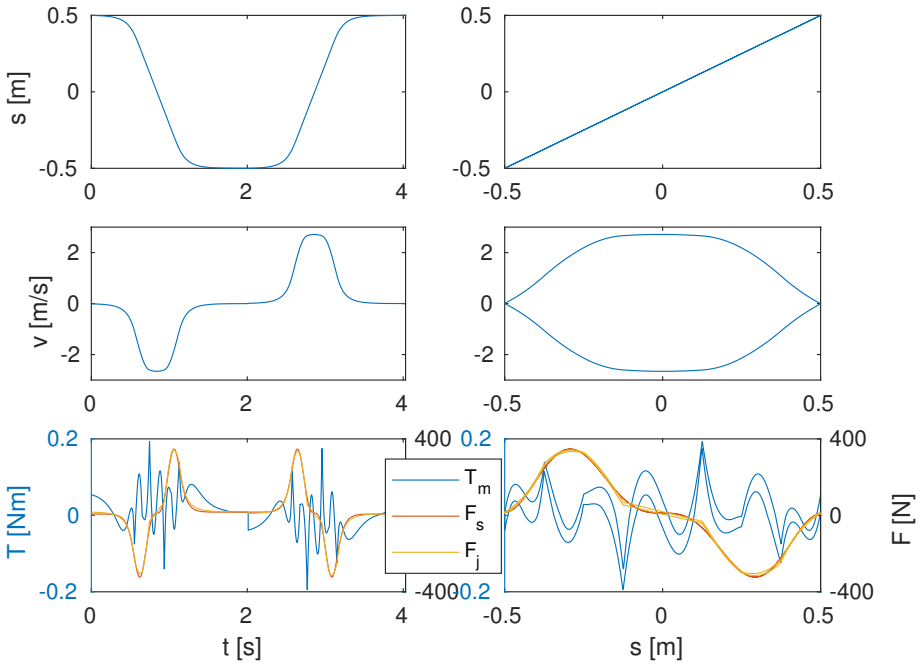
Figure 2-9a show a new trend with no strange outliers. Actually this trend makes more sense. A strong spring leads to a faster system response, but a faster response usually entails larger energy loss. Somewhere there must be the trade-off. Figure 2-8a shows only the strong spring, Figure 2-9a also shows the trade-off.

The second and third row of Table 2-4 show the energetic properties of the trajectories appointed as optimal by both Figure 2-8a and Figure 2-9a. The first one has a shorter cycle time and related to that a lower total energy cost. But both mechanical and electric losses are larger. Even when the recovered energy is scaled by the cycle time the trajectory to the minimum cost has less remaining recovered power.

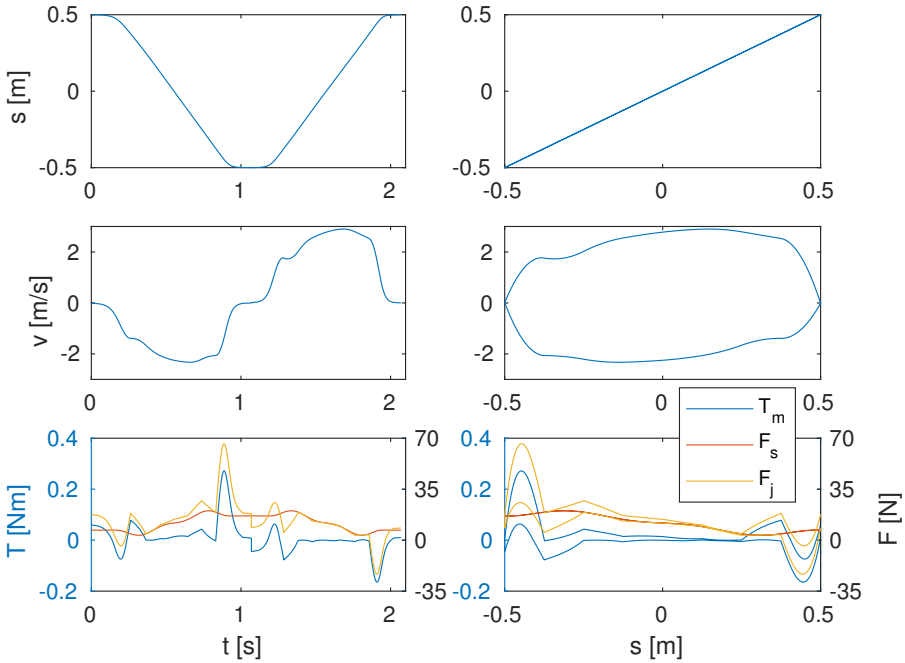
When comparing Figures 2-8b and 2-9b, there is indeed barely a difference. The minimum still lies between $k_s = 4$ and $k_s = 10$, this time $k_s = 10$ yielding a slightly better result. The bottom two rows of Table 2-4 show how little the difference is. The one has slightly lower

Table 2-4: Energy results to the optimal trajectories with optimised combined spring mechanism

	k_s [N/m]	cycle time [s]	mechanical loss [J]	I^2R loss [J]	total energy cost [J]	recovered energy [J]	recovered power [W]
with reflected inertia							
from opt. spring	82.5	4.20	2.22	4.03	13.3	3.56	0.85
from scratch min cost	810	3.41	2.98	2.00	8.79	4.84	1.42
from scratch max power	610	4.03	2.17	1.60	10.1	6.04	1.50
without reflected inertia							
from smooth	57.9	2.26	1.49	0.83	1.55	7.48	3.31
from scratch min cost	4	2.06	1.55	1.00	0.97	7.26	3.53
from scratch max power	10	2.07	1.53	0.97	0.97	7.30	3.53



(a) With reflected inertia, optimal spring stiffness $k_s = 610$



(b) Without reflected inertia, optimal spring stiffness $k_s = 10$

Figure 2-10: Optimal trajectories selected on basis of maximal power recovery, optimised from scratch with and without reflected inertia taken into account; on the left plotted against time, on the right against distance. The bottom plots show in yellow and red the joint forces and spring characteristic in N on the right axis, and in blue the residual motor torque in Nm on the left axis

energy losses, the other a tiny bit shorter cycle time, which in the balance results in equal total energy cost and equal recovered power when rounding to two decimals. This illustrates the high nonlinearity of the optimisation problem.

Figure 2-10 shows the trajectories to the results returning most power, with and without reflected inertia. When comparing the two figures, equal velocities are reached during the cycles. Only with reflected inertia it takes much longer to reach full velocity and deceleration is also started earlier. The trajectory shape is almost fully determined by the spring characteristic. The red and yellow lines (bottom plots) almost coincide.

This is certainly not the case in Figure 2-10b where initial acceleration and final deceleration are achieved by a large peak in the actuator torque. But it is not so much that the actuator torques are larger, they are within similar range for the case with reflected inertia, it is much more that the total joint forces are much lower, almost a factor 6.

It is the slow increase of the spring force near the end positions which causes the performance drop for large reflected inertia. With small reflected inertia, the actuator can nudge the system into motion at the start, then the spring mechanism takes over. When the reflected inertia gets large, the larger cycle time because of the slower response at some point starts costing too much because of the presence of overhead power drain.

The other trajectory plots to the results presented in Table 2-4 can be found in Appendix C-2.

2-5 Results compared

Table 2-5 summarises the energy results of the previous sections. The bumpy results observed for the analytical spring are regarded as only being made possible by an unfortunate choice of parameterisation of the trajectory (more about this in Section 2-8 Discussion). Nevertheless they show a number which is probably closest to the maximum power that can be retrieved by putting a parallel spring on the chosen spindle actuated mechanism and the chosen way of parameterising the trajectory. It suggests a more careful trajectory parameterisation has the potential to yield an analytical spring which can retrieve even more power from the system. It is however not useful for evaluating the performance of the resonant mechanism, as such bumpy profiles are not desired to be followed by a real mechanism. For comparing the two different mechanisms, the smooth result will be used.

In Table 2-5 it is seen that, after the overheads, the mechanical losses, in this case due to spindle efficiency only, are largest. For the analytically optimal springs, they are about 2 to 5 times larger than the I^2R motor heating losses. For the resonant mechanisms the difference is less than a factor 2, though the mechanical losses are still larger. All results in the table compared show in all cases increase in mechanical loss and motor heating go together, as they are both related to the motor torque, the mechanical losses by a linear relation, the I^2R losses quadratically.

In the cases with the mechanism and without any spring, the mechanical loss and especially the electric heat loss increase when the reflected inertia is added. This is not the case for the analytical optimal spring. The analytical spring can deliver any large force at the start and end of the trajectory (Figure 2-4). The resonant mechanism exerts its maximum force later,

Table 2-5: Summary of energy results from the previous sections

	k_s [N/m]	cycle time [s]	mechanical loss [J]	I^2R loss [J]	total energy cost [J]	recovered energy [J]	recovered power [W]
with reflected inertia							
analytical spring (smooth)		2.02	1.31	0.43	0.01	8.07	4.00
analytical spring (bumpy)		1.87	1.39	0.58	-0.36	7.84	4.19
spring mechanism	610	4.03	2.17	1.60	10.1	6.04	1.50
no spring		4.85	4.06	10.05	23.71	-4.30	-0.89
without reflected inertia							
analytical spring (smooth)		2.08	1.33	0.57	0.40	7.91	3.81
analytical spring (bumpy)		1.75	1.28	0.26	-1.26	8.27	4.72
spring mechanism	10	2.07	1.53	0.97	0.97	7.30	3.53
no spring		2.00	3.76	2.53	4.40	3.61	1.80

at a quarter of the distance to be travelled. Increased acceleration earlier along the trajectory has to come from the motor, or the trajectory takes longer. The optimal result (Figure 2-10a) shows a combination of both. In the top half of Table 2-5 it is seen the I^2R losses are a small factor 4 times larger when compared to the analytical (smooth) spring, and the cycle time has become twice as long. Without spring, all force for acceleration has to come from the motor, except for some help by gravity on the way down. As a result the I^2R losses are far larger, while the overhead costs have kept the cycle time from increasing much further.

The bumpy analytical springs are actually the only springs actually making the system retrieve more energy than is consumed by the overheads. This is for the largest part due to the short cycle time.

It is hard to compare the results to some absolute optimum. The theoretical maximum energy that can be retrieved from one cycle is 9.81 J, but that would be for zero cycle time, which is definitely unrealistic. Table 2-6 shows the recovered energy (overheads not taken into account) of the mechanisms of Table 2-5 compared to the maximum recoverable energy: $\frac{\text{recovered energy}}{9.81}$. The second column compares the power retrieved by the resonant mechanisms to the power retrieved with the smooth analytical optimal spring.

The optimal springs manage to recover 80-84% of the full recoverable energy, regardless of the reflected inertia. The mechanisms manage 74% without and 62% with reflected inertia. Without spring, only without reflected inertia any energy is recovered and then only just 37%.

The last column of Table 2-6, based on the last column of Table 2-5, places the results in a more relative perspective, considering the relative energy return over time. Without the large reflected inertia, the spring mechanism is capable of retrieving 93% of the power recovered by the smooth analytical spring, while the system without spring manages less than half. With reflected inertia, the spring mechanism achieves a power recovery of only 38%. But without spring no power is recovered at all, it only costs.

It is seen that the proposed physical mechanism performs well under circumstances without reflected inertia. With the large reflected inertia induced by the spindle drive performance is

Table 2-6: Recovered energy and power compared. The first column shows the recovered energy with respect to the maximum energy that could be gained in a cycle. The second column shows for the power recovered with respect to the power that was recovered with the analytically optimised smooth spring.

	recovered energy w.r.t. upper bound 9.81	recovered power w.r.t. smooth analytical spring
with reflected inertia		
analytical spring (smooth)	0.82	1
analytical spring (bumpy)	0.80	1.05
spring mechanism	0.62	0.38
no spring	-0.44	n.a.
without reflected inertia		
analytical spring (smooth)	0.81	1
analytical spring (bumpy)	0.84	1.24
spring mechanism	0.74	0.93
no spring	0.37	0.47

much less, but still capable of recovering energy in a reasonable cycle time. In the rotational arm, the transmission will have both lower inertia of itself and a lower transmission ratio. Since the reflected inertia scales with the square of the transmission ratio (2-2a), the reflected inertia in the polar arm will be only a fraction of the reflected inertia assumed in this chapter. Therefore it is justified to conclude good performance of the mechanism based on the results without reflected inertia.

2-6 Horizontal motion

The principle for horizontal application is the same as in the vertical direction. The only difference is that now the gravitational acceleration acts perpendicular to the direction of motion. This results in the following two changes in the mechanical model:

1. The computations for the are done with zero gravitational acceleration. For the resonant mechanism this means there is no longer an endpoint balancing spring.
2. The spindle model now does show Coulomb friction. F_f in (2-1), or μ_C, is set to μmg, with $\mu = 0.006$ from the spindle specifications [19]. Note that this friction depends on whether the 'robot' is loaded or unloaded.

Also in horizontal direction a distance of 1 m is used.

Horizontal motion always costs energy since there is no potential energy difference. Therefore it makes no sense to evaluate the performance in terms of power loss without overheads, as has been done in the vertical case. The optimum would erroneously lie at infinite cycle time. Yet for comparison between results it is interesting to see the energy loss over time, the values are still computed.

Since it was concluded at the end of the previous section that the case without reflected inertia will be a closer representative of the later system with rotational drive, this section will only consider the system without reflected inertia.

2-6-1 Optimisation

Again the trajectory and spring will first be optimised using the implicit analytical optimal spring. The best result out of 250 optimisations is selected.

Next, the result is used as initial trajectory for optimisation of the resonant mechanism. Finally, further optimisation of the mechanism is again achieved by optimising the trajectory for a range of mechanism spring stiffnesses, 1000 times per spring stiffness. The range $k_s = 0, 1, \ldots, 25$ proved sufficient. Since the optimum found from this last set of optimisations outperforms the mechanism optimised from the analytical optimal spring trajectory, only the result obtained from the range of spring stiffnesses will be presented.

2-6-2 Results

Figure 2-11 shows the optimal trajectory with an analytically optimal spring. The corresponding energy results can be found in Table 2-7. Again the more bumpy result, listed in the table as well, is discarded because of its larger first order discontinuities. But when comparing the two in Table 2-7 it is seen the bumpy result was not selected by the optimisation because of its lower energy losses but because of its shorter cycle time. In the total cost the overhead factor is dominant. Therefore it may be that in the acquired results some of the system energy has been sacrificed for the sake of a faster response.

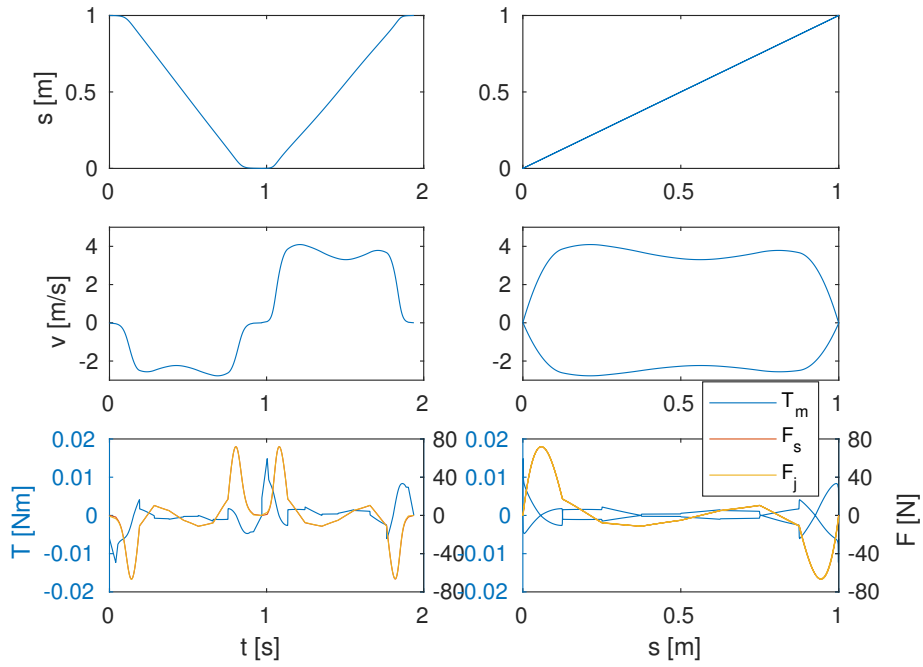


Figure 2-11: Optimised horizontal trajectory with analytically optimal spring, on the left plotted against time, on the right against distance. The bottom plots show in yellow and red the joint forces and optimised spring characteristic in N on the right axis, and in blue the residual motor torque in Nm on the left axis.

Table 2-7: Energy results of the horizontal optimisations

	k_s [N/m]	cycle time [s]	mechanical loss [J]	I^2R loss [J]	total energy cost [J]	energy loss [J]	power loss [W]
analytical spring (smooth)		1.94	0.21	0.14	8.11	0.36	0.19
analytical spring (bumpy)		1.81	0.25	0.14	7.61	0.39	0.22
spring mechanism	9	2.14	1.18	0.88	10.62	2.07	0.97
no spring		2.11	1.28	1.00	10.72	2.28	1.03

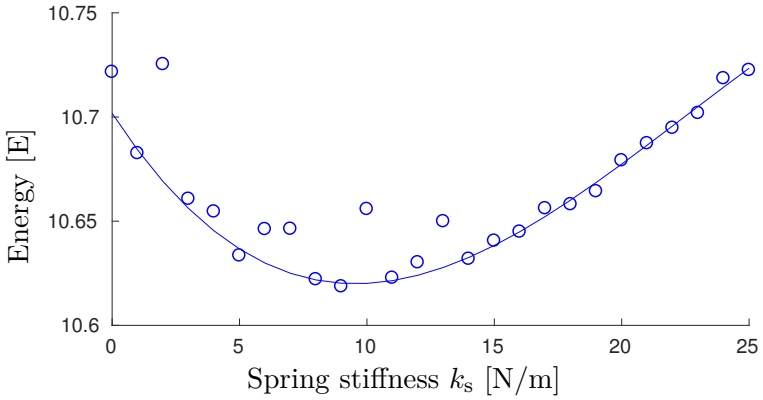


Figure 2-12: Minimum energy consumption for the horizontal resonant spring mechanism with different spring stiffness. The bottom line trend is estimated by a third order polynomial.

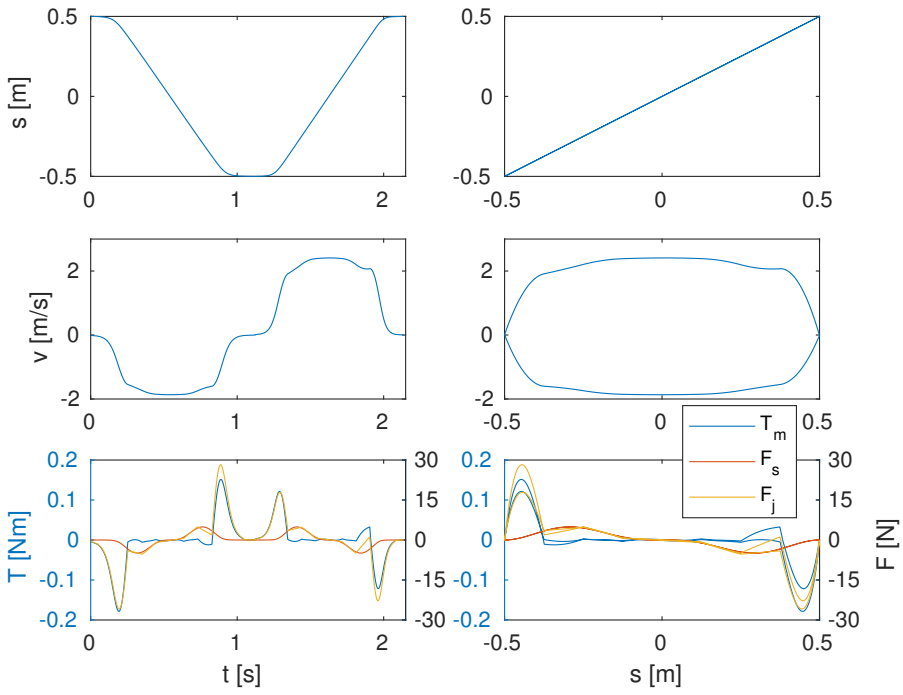


Figure 2-13: Optimal horizontal trajectory with mechanism with $k_s = 9$. On the left plotted against time, on the right against distance. The bottom plots show in yellow and red the joint forces and spring characteristic in N on the right axis, and in blue the residual motor torque in Nm on the left axis.

Figure 2-12 shows the minimum total energy cost for the range of springs with $k_s = 0, 1, \ldots, 25$. Again the general trend is estimated by a third order polynomial. Compared to the vertical case (Figure 2-8b) the results show a steeper curve with a minimum at a slightly larger spring stiffness. The lowest power consumption of the full set is found at $k_\mathrm{s} = 9$. Figure 2-13 shows the corresponding trajectory and the energy results can be found in Table 2-7. For comparison, the table also shows the energy results of the optimisation without spring.

The differences in energy loss between the optimal spring and the optimised resonant mechanisms are more pronounced than they were in the vertical case. The mechanism performs hardly any better than if there were no spring. This is because in contrast to the analytical spring (Figure 2-11), the resonant mechanism does not produce large forces right at the beginning and end of the trajectory (Figure 2-13). For a fast response it is important that the system is made to move as quickly as possible. The mechanism will perform better if there is less of a penalty on the cycle time.

Horizontal motion will never be implemented without the vertical motion. When combined, the overhead energy is consumed only once per cycle, for which the energy regenerated from the vertical motion compensates and is also used to power the horizontal mechanism, reducing the extreme time pressure currently present.

2-7 Combined motion

The result of the previous sections was a combination of an optimal spring characteristic/ mechanism in combination with an optimal trajectory, either in vertical or in horizontal direction. The motions are fully decoupled but for the shared factors of time and overheads. When applying the optimal vertical result (both spring and motor signal) to the vertical drive and the horizontal result to the horizontal drive, the motion that is slowest dictates the time of arrival at the endpoint. The overhead energy consumption will increase negligibly for the additional control of a second direction, but fast motion in one direction will no longer have the benefit of decreasing the overhead energy consumption when the motion in the other direction takes longer.

Table 2-8 shows the results of Sections 2-3-2, 2-4-3 and 2-6-2 combined. The total cycle time is the maximum of the cycle times of the vertical and horizontal trajectories. The combined energy cost is the sum of the mechanical loss and I^2R loss of the two trajectories, the overheads for the maximum cycle time and the -9.81 J provided by the task. The total recovered energy is the combined energy cost without the overheads and the total recovered power is the total recovered energy divided by the combined cycle time.

Table 2-8: Energy and power results of the combined motion (without reflected inertia)

	cycle time [s]	combined energy cost [J]	recovered energy [J]	recovered power [W]
analytical spring (smooth)	2.08	0.76	7.55	3.63
spring mechanism	2.14	3.31	5.25	2.45
no spring	2.11	7.20	1.24	0.59

The energy numbers are a worst case scenario, since the faster of the two motions has sacrificed more energy for speed than necessary. Though the differences will not be large, as the differences between the two cycle times was in all cases less than 10%. Despite the hard conditions for the mechanism in horizontal direction, combined with the vertical direction, the power recovery compared to the analytical springs is still a good 67%. This is still four times more than the amount recovered without spring.

2-8 Discussion

The choice of acceleration divided by velocity $\frac{\ddot{y}}{\dot{y}}$ as trajectory parameter results in very high numbers required at start and end when the velocity is small. As $\dot{y}$ rapidly in-/decreases, this results in the large bulbs at start and end of trajectory, while $\frac{\ddot{y}}{\dot{y}}$ is linear up to the next point of parameterisation. Linear interpolation from itself already leads to functions which are discontinuous in the first derivative, i.e. non-smoothness. Here, as $\dot{y}$ is very nonlinear, piecewise linear $\frac{\ddot{y}}{\dot{y}}$ results in a sequence of weird polynomials. It would probably have been better for the optimisation to have simply taken the acceleration $\ddot{y}$ as the trajectory parameter, not divided by the velocity.

The fact that this choice of trajectory parameterisation occasionally results in erratic definitely suboptimal forms of analytical optimal spring characteristics returning much energy in a cycle time considerably shorter than otherwise, i.e. capable of large power recovery, suggests that by a different choice of trajectory parameterisation, more power can be retrieved from the system.

On closer inspection of the spring characteristics it was seen that ideally the parallel spring provides a large force as close as possible to the end positions of the trajectory. The resonant mechanism reaches its peak only at a quarter of the distance. A mechanism which would achieve a rapid to instant increase in force when moving away from the endpoint followed by a low to zero force middle region, would achieve a faster response with less additional motor torque and would therefore be able to recover the energy during a shorter cycle time, thus increasing the power available for the overhead systems.

There is also the ongoing discussion on considering energy versus power. I feel it necessary to keep drawing the power into the discussion, even though the optimisation criterion is in terms of energy. This is because the overhead power drain is no real constant. The more power can be saved, the better. It is no use if all the energy can be saved and it takes an eternity to reach the other end, because time is as important. The only unbiased way to view the energy return versus time is to look at the power return. It was seen in Section 2-4-3 how unreliable an optimum computed with the total energy cost function can become if the overhead power estimate becomes too different from the amount of power that can be recovered by the system. Unfortunately the recovered power as a mathematical expression in terms of the trajectory parameters has some unmanageable nonlinear dependencies, originating from division by the final time which in the current system description has become a state variable which is obtained from integration of the (very nonlinear) system dynamics.

In the end, it is not as much the specific results in this chapter which are of interest, but what they can mean for design of more complex robots which can be of practical use. What do

these results mean for the design of a robot arm with multiple rotational degrees of freedom?

The spindle specific details are of no interest. Spindle drives do not apply in regular robotic joints. Their large inertia makes them unpractical for energy efficient actuation. Yet some form of drive train had to be selected in order to include a realistic drive train model to the system equations.

Rotary transmissions are both lighter and exist with smaller reduction ratios. For that reason, the drive train inertia was neglected in the final results presented in this chapter. But they have been seen to be a very important source of mechanical energy loss. Care must be taken to keep the reflected inertia as low as possible. And in design of a real robot they may not be ignored.

The linear resonant mechanism used in this chapter has a rotational counterpart which has proved its value in simple horizontal back and forward pick and place motions[17]. This chapter has shown that the type of mechanism is indeed capable of retrieving most of the energy externally added to the system (gravity on the package) when compared to a theoretical spring mechanism which is analytically optimal.

The large remaining challenge is to design a rotational arm such that the path of the trajectory has a similar potential energy profile. Then resonant mechanisms can be applied to create very energy efficient parallel elastic actuation.

2-9 Conclusion

In this chapter two types of mechanisms for parallel elastic actuation have been simulated in a simple Cartesian setup, compared and discussed. By defining the system dynamic equations of motion as a function of distance instead of time, it was possible to calculate the optimal spring analytically, given the trajectory. Optimising the trajectory automatically resulted in the optimal parallel spring. However this spring is theoretically optimal and not necessarily practically implementable.

A simple mechanical spring mechanism, in this chapter referred to as the ‘resonant mechanism’, had on forehand already shown a promising energy profile. The force characteristic of the analytically optimised spring shows similarity to the general form of the resonant mechanism force characteristic. By applying the resonant mechanism to the system in vertical motion instead of the analytically optimal spring and optimising the trajectory over a range of the mechanism parameter, showed in simulation that over 90% of the energy that was retrieved with the similarly formed optimal spring could also be retrieved by the resonant mechanism. Compared to only electric energy regeneration, without any parallel springs, the resonant mechanism recovered the double amount of energy.

When scaling the recovered energy with the cycle time of the corresponding trajectory, 3.53 W electric power was retrieved by the resonant mechanism, compared to 3.81 W with the analytical optimal spring. This power is needed by an autonomous system to balance overhead power consumption of components such as a processor and many types sensors.

These results apply when the reflected inertia of the drive train is not taken into account. The analytical spring is relatively insensitive to additional reflected inertia in terms of power recovery. This is less the case for the resonant mechanism. With the reflected inertia caused by

the spindle, only 76% of the energy is recovered compared to the analytical optimal spring. In combination with a much increased cycle time, this results in only 38% of the power recovered otherwise. However, without spring additional energy would be required to keep the system moving.

This performance drop is so dramatic because in the present Cartesian case the reflected inertia is unreasonably large due to the spindle drive. Therefore it is justified to ignore the reflected inertia for now. In the polar case it will be taken into account, but it will be much smaller.

How the results of this chapter can be applied to a multiple-link polar robot arm is to be read in the next chapter.

Chapter 3

Multiple degree of freedom robot arm

From Chapter 2 energy and mechanism results were obtained for two decoupled degrees of freedom: vertical and horizontal. However, in practice it is desirable to have a robot arm, consisting of multiple links connected by rotational joints. Robot arms are popular because they have a much smaller footprint for the same reachable workspace and can be much lighter of design. By mounting them on a mobile platform, they are easily made mobile. Furthermore, actuation of rotational joints, especially when backdrivability is a requirement, can be done much more efficiently when no transmission such as a spindle is needed for driving translational motion.

The problem now gets much more involved, as separate workspace dimensions are no longer decoupled in the robot's configuration space. This chapter will have its focus on only 2 DoF, which is the smallest number required for the task. Section 3-1 will introduce the kinematics of the robot arm and the terminology of the different coordinate spaces and their interdependence relationships used throughout this chapter.

Next, Section 3-2 will discuss the applicability of the optimal spring characteristic, obtained in the previous chapter for the Cartesian case, to the polar arm through direct mapping. In Section 3-3 the optimal physical Cartesian spring mechanisms are translated to the polar configuration, which is a more successful approach to the problem. Afterwards, the full system model of the rotational arm is drawn up in Section 3-4 and the trajectory of the arm is optimised in Section 3-5.

Extra DoF provide extra freedom of motion through redundancy. The larger solution space may contain better solutions, but also results in a larger space to explore when optimising unless a clever coupling between the different DoF can be found. This is one of the topics that will be discussed in Section 3-6 before Section 3-7 concludes the chapter.

3-1 The kinematic model and terminology

Figure 3-1 shows the 2 DoF robot arm. Compared to the Cartesian arm (Figure 2-1b), this arm has four additional parameters: the link lengths l_1, l_2 and the position (x_0, y_0) of the

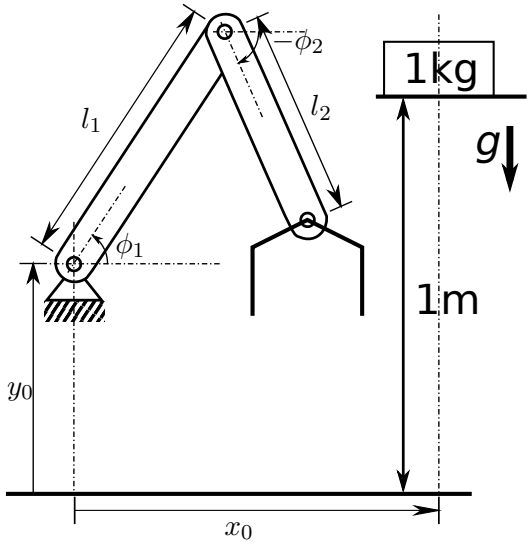


Figure 3-1: Schematic representation of the 2 DoF rotational arm. By choosing different values for the parameters x_0, y_0, l_1 and l_2, the same trajectory in workspace coordinates can be much different in the configuration space coordinates of the robot.

shoulder. The choice of these parameter values has a large influence on the system dynamics and will be made in Section 3-4-2.

The position the robot arm is in can be expressed in two different coordinate spaces: configuration space $\mathbf{q}$ and workspace $\mathbf{x}$. The configuration space is defined in terms of the independent coordinates ϕ_i (Figure 3-1), the absolute angular positions of the (in this case two) links of the robot. In contrast to the robot joint space, the workspace coordinates express the robot position in terms of x, y and θ in the global world. The workspace coordinates relate to the configuration space coordinates by the mapping:

$$\mathbf{x} = \mathbf{x}(\mathbf{q}) \tag{3-1}$$

The coordinate transformation (3-1) depends on l_1, l_2 and (x_0, y_0).

The derivative of the mapping with respect to the configuration space coordinates results in the Jacobian

$$\mathcal{T}(\mathbf{q}) = \nabla \mathbf{x}(\mathbf{q}) \tag{3-2}$$

such that

$$\dot{\mathbf{x}} = \mathcal{T}(\mathbf{q})\dot{\mathbf{q}} \tag{3-3}$$

and approximately

$$\Delta \mathbf{x} = \mathcal{T}(\mathbf{q})\Delta \mathbf{q} \tag{3-4}$$

The coordinates $\mathbf{q}$ cover the whole joint space, but $\mathbf{x}$ is restricted by the implicit kinematic constraints to only a part of the total workspace. In case of multiple DoF $\mathbf{x}$ is not unique for each $\mathbf{q}$. Therefore $\mathbf{x}(\mathbf{q})$ is generally not invertible.

The top position $\mathbf{q}_\mathrm{u}$ and a bottom position $\mathbf{q}_\mathrm{d}$ must be chosen such that the vertical difference between the two end positions $y_e(\mathbf{q}_\mathrm{u}) - y_e(\mathbf{q}_\mathrm{d}) = 1$ m. The other way around, this endpoint constraint and possible bounds on $\mathbf{q}$ also impose restrictions on the choice of l_1, l_2 and (x_0, y_0).

3-2 Direct mapping from Cartesian to 2D polar configuration

In the previous chapter, optimal trajectory results were obtained over the x and y position of the end effector. With $\mathbf{x} = \begin{bmatrix} x & y \end{bmatrix}^T$ and $\mathbf{q} = \begin{bmatrix} \phi_1 & \phi_2 \end{bmatrix}^T$ the angles of the two links of the robot arm $\mathcal{T}(\mathbf{q})$ (3-2) is square. As long as $\mathbf{q}$ does not denote a singular position of the arm, $\mathcal{T}(\mathbf{q})$ is invertible and (3-5) can be used to directly map the Cartesian trajectory to the polar case.

$$\Delta\mathbf{x} = \mathcal{T}(\mathbf{q})\Delta\mathbf{q} \tag{3-5}$$

Power is invariant under coordinate transformations $F_{\mathbf{x}}^T\dot{\mathbf{x}} = F_{\mathbf{q}}^T\dot{\mathbf{q}}$, such that

$$F_{\mathbf{q}} = \mathcal{T}^T(\mathbf{q})F_{\mathbf{x}} \tag{3-6}$$

Given the trajectory in terms of $\Delta\mathbf{x}$ and $F_{\mathbf{x}}$ as obtained in Section 2-3-2, the equivalents in configuration space coordinates can be computed using (3-5), (3-6).

However, optimality of the transformed results only holds under the following assumptions:

- The arm has negligible mass with respect to the concentrated mass at the end effector.
- The arm has negligible inertial moment.

With the very light gripper of Table 2-1 in mind, especially on the way up, when the arm is unloaded, this first assumption will very likely not hold. When replacing the slider by the construction of the arm and distributing its mass (Table 2-1), the assumption will definitely not hold. Even if it would, the second assumption of negligible inertia can never hold as long as masses are held by links of any length. The fact that a 1 kg package has to be transported by the arm over a distance of a full meter will per definition result in inertial effects in the dynamics of the arm.

Furthermore, the optimisations in Chapter 2 are done taking into account a drive train model (with a spindle transmission) specific for the Cartesian case. The differences in the drive train when actuating a polar joint will have considerable impact on the optimality of the results.

For these reasons this approach is abandoned.

3-3 Translation of drive mechanism principles

In Chapter 2 two different approaches were used to find an optimal spring characteristic for a Cartesian arm: directly optimising the characteristic and optimising the parameters of a heuristically selected physical mechanism. The previous section showed that the results of the direct optimisation cannot be readily translated to the polar arm. In order to find mechanisms for the polar arm, this section will therefore translate the results from the physical mechanism optimisation to the polar configuration.

3-3-1 Endpoint gravity balancing

Part of the physical mechanism keeps the arm balanced with gravity (Section 2-4). This balancing part enforces two important properties:

- In the bottom position the arm is in stable equilibrium when it is carrying the package.
- In the top position the arm is in stable equilibrium when it is not carrying the package.

The equilibrium condition can be expressed as

$$-\left.\frac{\partial E_{\text{pot}}}{\partial \phi}\right|_{\mathbf{q}^\star} = T_{\text{s}}(\mathbf{q}^\star) + T_{\text{g}}(\mathbf{q}^\star) = 0 \tag{3-7}$$

with spring torque T_{s} and gravity torque

$$T_{\text{g}} = \begin{cases} -(m + m_{\text{pack}})gl\cos\phi & \text{down} \\ -mgl\cos\phi & \text{up} \end{cases} \tag{3-8}$$

Stability requires satisfaction of

$$\frac{\partial^2 E_{\text{pot}}}{\partial \phi^2} > 0 \qquad \text{or} \qquad -\frac{\partial}{\partial \phi}\sum T > 0 \tag{3-9}$$

which will be referred to as positive system stiffness.

Linear torsion spring

In the Cartesian configuration gravity balancing was achieved by a simple linear spring $F_{\text{s}}(y) = -k_{\text{s}}(y - y_0)$ (2-21). Analogue to (2-21) in the Cartesian configuration, a linear torsion spring can be applied in the polar configuration:

$$T_{\text{s,lin}}(\phi) = -k_{\text{s}}(\phi - \phi_0) \tag{3-10}$$

Solving the equilibrium condition (3-7) for the linear spring (3-10) at $\phi_{\text{u}}, \phi_{\text{d}}$ results in the following spring stiffness k_{s} and rest length ϕ_0:

$$k_{\text{s}} = \frac{(m + m_{\text{pack}})gl\cos\phi_{\text{d}} - mgl\cos\phi_{\text{u}}}{\phi_{\text{u}} - \phi_{\text{d}}}$$
$$\phi_0 = \frac{(m + m_{\text{pack}})gl\cos\phi_{\text{d}}}{k_{\text{s}}} + \phi_{\text{d}}$$

The top plot of Figure 3-2 shows the linear spring force balancing the heavier mass in an arbitrary down-position while also balancing the lighter mass in an equally arbitrary up-position. The bottom plot of the figure shows the corresponding system stiffness. The system stiffness in this Cartesian case simply equals the spring stiffness because the position does not change the gravitational force on the arm.

However, gravitational torque on a joint rotating in a vertical plane is not constant with respect to position, as was already seen in (3-8). The half-sines of the gravity for both the

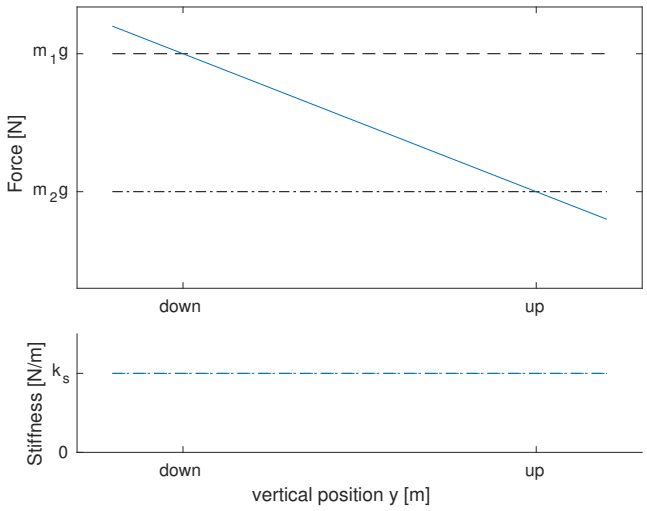


Figure 3-2: Force and stiffness for Cartesian gravity balancing spring and gravity forces in vertical direction, m_1 and m_2 being the masses of the loaded and unloaded arm respectively

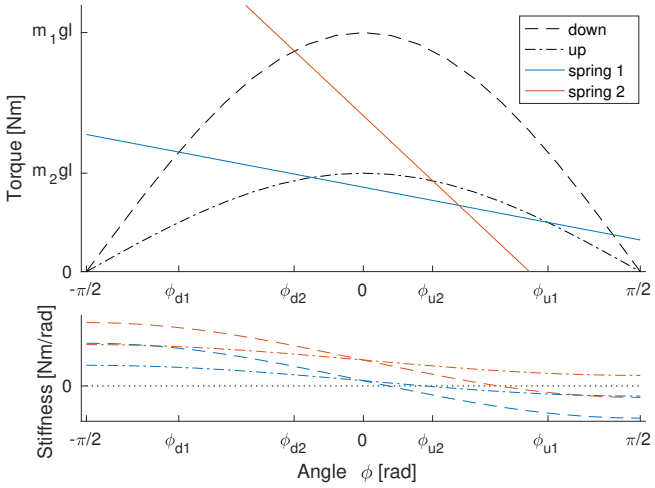


Figure 3-3: Torque of gravity and two linear torsion springs for one rotational link and corresponding stiffness, m_1 and m_2 being the masses of the loaded and unloaded arm respectively

loaded and the unloaded case are shown in the top plot of Figure 3-3. The figure also shows two linear springs: the blue one balancing the system in down and up positions $\phi_{\mathrm{d1}}, \phi_{\mathrm{u1}}$ lying far apart, the red one balancing the system in positions $\phi_{\mathrm{d2}}, \phi_{\mathrm{u2}}$ much closer together.

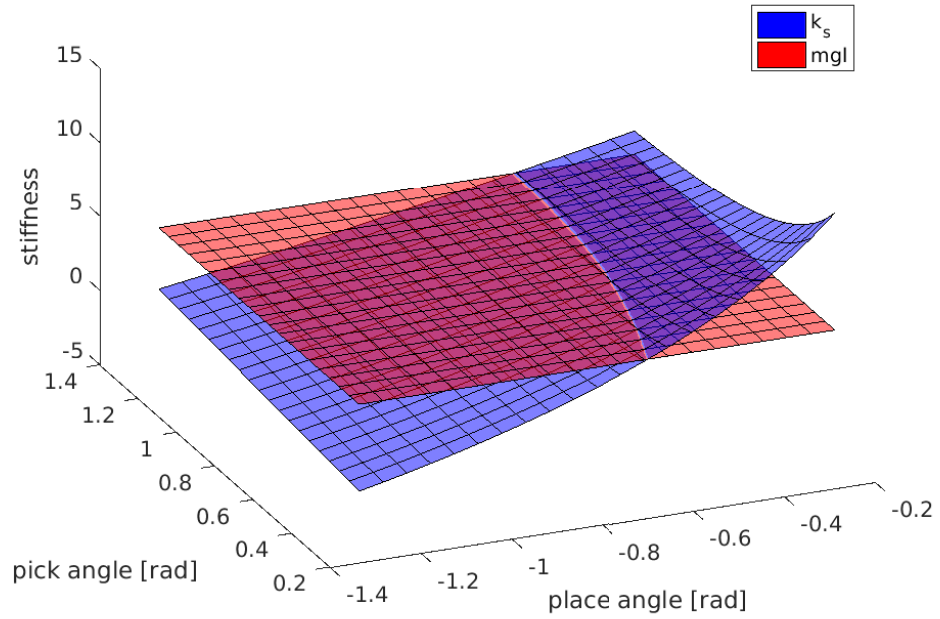
The red spring torque line crosses both mass torque lines from left to right from top to bottom in the equilibrium points. In and around both equilibrium points the system is stable. The system is stable as long as the spring torque has a steeper tangent than the gravity torque, which is the same as the full system having a positive stiffness (bottom plot).

With the blue spring, positive system stiffness is only obtained in the bottom position. In the top position the spring stiffness is smaller than the change in gravitational torque and the blue line crosses the dash-dotted gravity torque line from lower on the left to higher on the right. The top position is an unstable equilibrium. Stable equilibria at the top position are only obtained for relatively small ranges of $\phi_{\mathrm{u}} - \phi_{\mathrm{d}}$.

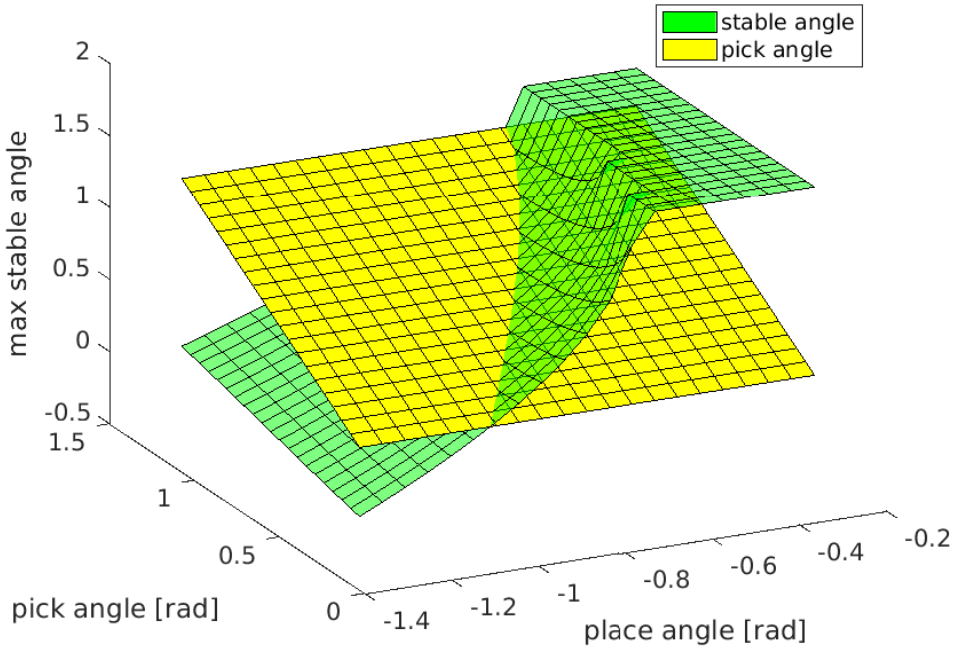
For which combinations of $\phi_{\mathrm{d}}, \phi_{\mathrm{u}}$ the top position is stable is shown in Figure 3-4a by the region where the blue plane lies above the red. The red plane denotes for the unloaded case the maximum change in gravity torque. As long as this maximum change in gravitational torque is smaller than the spring stiffness (blue), the unloaded arm will be pulled towards the top equilibrium for any ϕ.

The region increases as the arm becomes lighter, but not to a satisfactory extent.

However global stability is not required as long as the end position is included in the stable region and control ensures the system does not exit the stable region. Figure 3-4 shows in green the maximum angle at which, in the upward case, the total stiffness is positive, while the yellow plane shows the angle of the pick position. The flat part of the green plane is the same region that was found stable in Figure 3-4a. The sloping region where the maximum stable angle is larger than the pick angle will also result in an attractive equilibrium at the top



(a) Stable region where the blue plane lies above the red



(b) Stable region where the green plane lies above the yellow

Figure 3-4: Stability regions, where total system stiffness is positive and the equilibrium attractive. The arm has an equivalent mass $m_2 = 0.7$ kg (Table 2-1) at a length of $l = 0.7$ m.

position. A larger difference between the two angles is desirable for an increased attractive region.

Though the relaxation of the stability criterion provides some extension to the ranges of pick and place angles, the area of both large pick and large place angles still falls in the unstable region for all but the lightest arms. A linear spring force was perfect when the contribution of gravity was linear, Figure 3-2. Now that gravity has entered the equations as a sinusoid, a sinusoidal gravity balancing spring will be better suited for the situation.

Sinusoidal spring torque

The simplest mechanism resulting in a sinusoidal spring torque at the joint is drawn schematically in Figure 3-5. The corresponding spring torque is

$$T_{\mathrm{s,nonlin}} = k_{\mathrm{s}} a^2 \cos(\phi + \alpha) + f_0 \tag{3-11}$$

with zero free length/pretension, i.e. $f_0 = 0$. Considering only positive k_{s}, a and $0 < \alpha < \pi$, (3-7) with (3-11) has a unique solution for any $-\frac{\pi}{2} < \phi_{\mathrm{d}} < \phi_{\mathrm{u}} < \frac{\pi}{2}$. Furthermore, the end positions are stable. The parameters $k_{\mathrm{s}} a^2$ and α can be solved for numerically.

Figure 3-6 shows three sinusoidal springs together with the gravitational torque. The first spring (in blue) shows stability is achieved for large endpoint angles $|\phi_{\mathrm{d}}|$ and $|\phi_{\mathrm{u}}|$. In fact, the system stiffness remains positive (3-9) as long as $|\phi_{\mathrm{d}}|, |\phi_{\mathrm{u}}| < \frac{\pi}{2}$, though the stiffness approaches zero as the magnitude of the end position approaches $\frac{\pi}{2}$.

The system stiffness in the end positions is equal for $\phi_{\mathrm{d}} = -\phi_{\mathrm{u}}$. As the trajectory gets "decentred" (red in Figure 3-6), stability diminishes for the end position closer to 0, while the other benefits from lower stiffness. The stability can be balanced again by adding free length to the spring (in case $|\phi_{\mathrm{u}}| < |\phi_{\mathrm{d}}|$) or pretensioning the spring (in case $|\phi_{\mathrm{d}}| < |\phi_{\mathrm{u}}|$). Then f_0 in (3-11) needs to be set accordingly. This was done for the yellow spring in Figure 3-6.

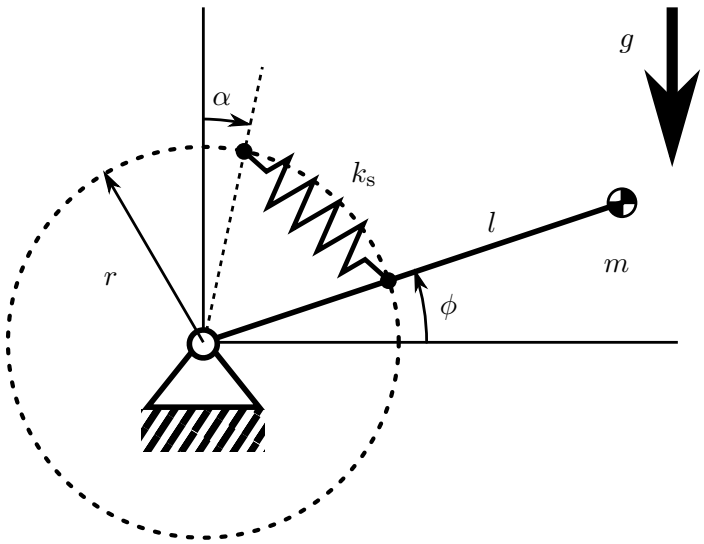


Figure 3-5: Nonlinear gravity balancing mechanism consisting of a linear zero free-length tension spring of stiffness k_{s} attached at radius r with an offset angle α

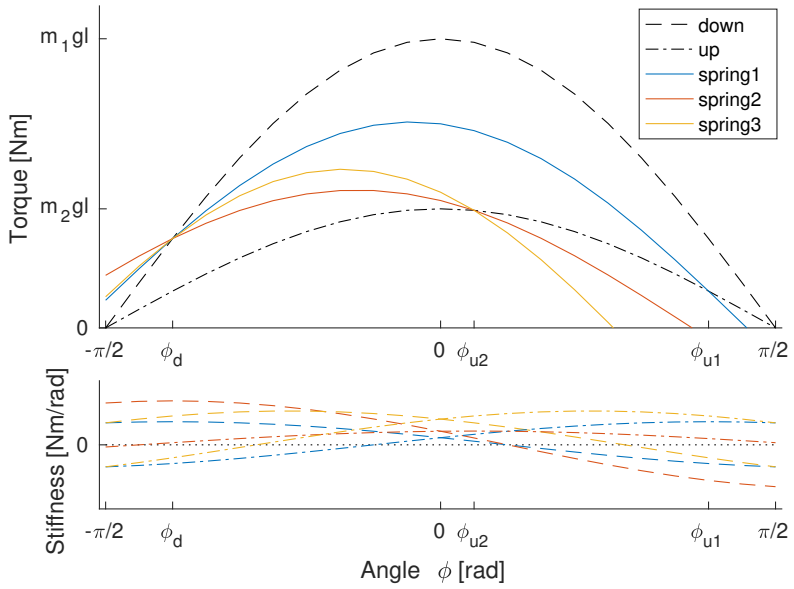


Figure 3-6: Torque and stiffness for two sinusoidal gravity balancing springs and gravity. Stable equilibria for large $|\phi_{\mathrm{d}}|$ and $|\phi_{\mathrm{u}}|$ (blue). Also when one of the endpoint angles has much smaller magnitude than the other, endpoint stability is still achieved, though the endpoint with the angle closer to zero has a stiffness closer to zero (red).

On the other hand for positioning at multiple bottom positions, it may be even favourable to have small stiffness at the down position. When over a range of bottom angles the sum of the spring torque and the gravity torque is smaller than the friction torques in the system, the arm can be positioned at any of the positions without requiring additional actuation to maintain its position.

In the rest of this chapter, the spring mechanism parameter k_{s}, α and f_0 are chosen numerically such that

$$\begin{aligned} \sum T_{\mathrm{down}}(q_{\mathrm{d}}) &= 0 \\ \sum T_{\mathrm{up}}(q_{\mathrm{u}}) &= 0 \\ \frac{\partial}{\partial q} \sum T_{\mathrm{down}}(q_{\mathrm{d}}) &= \frac{\partial}{\partial q} \sum T_{\mathrm{up}}(q_{\mathrm{u}}) \end{aligned}$$

In words: the equilibria in top and bottom position have equal system stiffness.

3-3-2 Rotational Resonant mechanism

The rotational resonant mechanism, also known as Plooij mechanism, is described extensively in [10]. Its principle is shown schematically in Figure 3-7, its energy and force characteristics are the same as in Figure 2-6 but with torque instead of force and angle instead of linear distance.

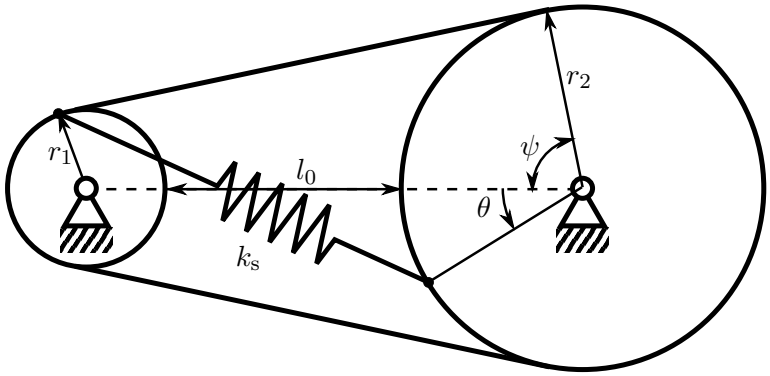


Figure 3-7: The rotational resonant mechanism by Plooij. Two wheels of different radius are connected by both a spring and a timing belt. The system is in equilibrium for $\theta = -\psi, 0, \psi$.

The potential energy in the spring as function of the mechanism angle θ is given by (3-12).

$$E_\mathrm{p} = \frac{1}{2} k_\mathrm{s} u^2 \tag{3-12}$$

$$u = \sqrt{x^2 + y^2} - l_0 \tag{3-12a}$$

$$x = r_2 \sin \frac{\theta r_1}{r_2} + r_1 \sin \theta \tag{3-12b}$$

$$y = r_1 + l_0 + r_2 - r_1 \cos \theta - r_2 \cos \frac{\theta r_1}{r_2} \tag{3-12c}$$

The following choice of parameters yield $\frac{\partial E_\mathrm{p}}{\partial \theta} = 0$ at $\theta = \psi, -\psi$:

$$\frac{r_1}{r_2} = \frac{\pi + \psi}{\psi}$$
$$l_0 = \frac{r_1 - r_2}{\cos \psi} - r_1 - r_2$$

The mechanism angle relates to the joint angle by $\theta = \phi - \beta$, where the offset angle β is chosen such that

$$\phi_\mathrm{u} = \psi + \beta$$
$$\phi_\mathrm{d} = -\psi + \beta$$

The magnitude of the energy dip between $\theta = \psi$ and $\theta = -\psi$ depends on k_s in combination with the choice of r_1 and r_2.

3-4 Full system model

The spring torque is now known as function of the pick and place angles and only very few parameters which can be used for scaling. For simulation and optimisation of the system, the remainder of the model consists of the system dynamical equations of motion and a model of the drive trains for actuation and electric energy regeneration.

3-4-1 System dynamics

The dynamical equations of motion will first be constructed as function of time using the "TMT-method"[20]. The map $\mathbf{x}(\mathbf{q})$ (3-1) of the system is given in (3-13).

$$\begin{bmatrix} x_{\mathrm{c},1} \\ y_{\mathrm{c},1} \\ \phi_1 \\ x_{\mathrm{j},2} \\ y_{\mathrm{j},2} \\ x_{\mathrm{c},2} \\ y_{\mathrm{c},2} \\ \phi_2 \\ x_{\mathrm{e}} \\ y_{\mathrm{e}} \end{bmatrix} = \mathbf{x}(\mathbf{q}) = \begin{bmatrix} \frac{l_1}{2}\cos\phi_1 \\ \frac{l_1}{2}\sin\phi_1 \\ \phi_1 \\ l_1\cos\phi_1 \\ l_1\sin\phi_1 \\ l_1\cos\phi_1 + \frac{l_2}{2}\cos\phi_2 \\ l_1\sin\phi_1 + \frac{l_2}{2}\sin\phi_2 \\ \phi_2 \\ l_1\cos\phi_1 + l_2\cos\phi_2 \\ l_1\sin\phi_1 + l_2\sin\phi_2 \end{bmatrix} \tag{3-13}$$

The corresponding mass matrix in workspace coordinates and gravitational force vector are given in (3-14) and (3-15) respectively.

$$M_{\mathbf{x}} = \mathrm{diag}(m_1, m_1, J_1, m_{\mathrm{j},2}, m_{\mathrm{j},2}, m_2, m_2, J_2, m_{\mathrm{e}}, m_{\mathrm{e}}) \tag{3-14}$$

$$F_{\mathrm{g}} = -g \begin{bmatrix} 0 & m_1 & 0 & 0 & m_{\mathrm{j},2} & 0 & m_2 & 0 & 0 & m_{\mathrm{e}} \end{bmatrix}^T \tag{3-15}$$

Once transformed to the configuration space coordinates the remaining torque and inertia terms can be added, resulting in the following dynamic equations of motion:

$$\begin{bmatrix} \dot{\mathrm{q}} \\ \ddot{\mathrm{q}} \end{bmatrix} = \begin{bmatrix} \emptyset & I \\ \emptyset & \emptyset \end{bmatrix} \begin{bmatrix} \mathrm{q} \\ \dot{\mathrm{q}} \end{bmatrix} + \begin{bmatrix} \emptyset \\ M_{\mathbf{q}}^{-1}\left(\mathbf{f}_{\mathbf{q}} + T_{\mathrm{f}}(\dot{\mathbf{q}}) + T_{\mathrm{s}}(\mathbf{q}) + Cn_{\mathrm{tr}}T_{\mathrm{m}}\right) \end{bmatrix} \tag{3-16}$$

with

$$M_{\mathbf{q}} = \mathcal{T}^T M_{\mathbf{x}} \mathcal{T} + (J_{\mathrm{m}} + J_{\mathrm{tr}})n_{\mathrm{tr}}^2 \tag{3-16a}$$

$$\mathbf{f}_{\mathbf{q}} = \mathcal{T}^T \left(F_{\mathrm{g}} - M_{\mathbf{x}}\nabla_{\mathbf{q}}(\mathcal{T}(\mathbf{q})\dot{\mathbf{q}})\dot{\mathbf{q}}\right) \tag{3-16b}$$

$$T_{\mathrm{f}} = -\mu_{\mathrm{v}}\dot{\mathbf{q}} - \mu_{\mathrm{C}}\mathrm{sign}(\dot{\mathbf{q}}) \tag{3-16c}$$

and spring torques T_{s} combining the sinusoidal gravity balancing mechanism and the Plooij mechanism from Section 3-3 and motor torques T_{m}. The motor and transmission constants J_{m}, J_{tr}, n_{tr} and C are diagonal matrices denoting respectively the rotor inertia, transmission inertia, transmission reduction ratio and transmission efficiency (2-2c) for each of the joints. The coefficients μ_{v} and μ_{C} describe the effects of viscous and Coulomb friction respectively.

The motor current, responsible for the I^2R heat losses in the motor, is given by (2-3) and repeated here for the overview.

$$I = I_{\mathrm{noLoad}}\mathrm{sign}(T_{\mathrm{m}}) + \frac{T_{\mathrm{m}}}{\bar{k}_{\mathrm{t}}} \tag{2-3}$$

No-load current I_{noLoad}, torque constant $\bar{k}_{\mathrm{t}}$ and wire resistance R can be found in the specifications of the motor. The same motor will be used as in Chapter 2. The motor constants are repeated in Table 3-1.

Table 3-1: Motor and gear transmission specifications and derived constants

No load current I_{noLoad}	0.538	A
Terminal resistance R	0.343	Ω
Corrected torque constant $\bar{k}_{\text{t}}$	71.1	mNm/A
Rotor inertia J_{m}	306	kgmm^2
Gear inertia J_{tr}	1.3	kgmm^2
Gear reduction ratio n_{tr}	35	
Forward efficiency η	0.76	
Backward efficiency η'	0.72	

In Appendix A-3 a minimum gearbox is selected to provide the necessary transmission between motor and joint. The choice was made based on the preliminary results obtained from the optimisation described in Section 3-5 without transmission ($J_{\text{tr}} = 0, n_{\text{tr}}, \eta = \eta' = 1$). The results of this preliminary optimisation are given in Appendix B-2. The transmission constants are listed in Table 3-1 as well. The backward efficiency η' was not given in the product specifications.

3-4-2 Mechanical system parameters

To finish the model, a choice has to be made for the lengths of the arm links l_1, l_2, as well as the position of the shoulder x_0, y_0. Comparing different robot arms, the following trends are observed in link lengths: In SCARA robots[21, 22], sometimes the lengths of the two main links are equal, but types both with a shorter first link and a shorter second link are common. Looking at other robots by KUKA[23], many different sizes and proportions are found, though often when considering the two main links of the robots, the second link is longer than the first link. On the other hand, Universal Robots are consequent in letting the links in their robot arms become shorter towards the end[3]. Even so in humans, the upper arm is longer than the lower arm. From preliminary optimisation by Vosse [8] (3 links), the last link was found to ideally be smaller. I have chosen to let the second link be 5 cm shorter than the first link.

The optimal total arm length was determined by Vosse [8] to be between 0.54 and 0.80 m. A longer arm means a larger reach, but also larger weight and inertia and a less compact robot. The total arm length is chosen such that singular configurations are avoided near the end positions, i.e. $\phi_1 - \phi_2 > 10\,\text{deg}$. For stability, $-0.4\pi < \phi < 0.4\pi$. Furthermore I want to be able to reach three different bottom positions which meet these requirements over a range of 0.25 m. As a result, y_0 has been chosen at 0.25 m, resulting in an x_0 determined by a very limited range of pick-angles. The chosen parameters are all listed in Table 3-2.

The masses of the links are estimated according to the following back-of-the-envelope equations: Common aluminium alloys of type 6061 [24] have a density of 2.70 $\times 10^3$ kg/m^3. The yield strength σ_{y} of the material, which should not be reached at any time, lies in the range of 55 - 276 MPa (depending on the precise alloy). The maximum tensile stress in a hollow tube with radius r and wall thickness t under a bending load causing a maximum moment M at a cross section of the tube is given by $\frac{Mr}{\pi r^3 t}$. With the rather pessimistic estimation of a a total arm

Table 3-2: Mechanical system parameters and end positions

links										
lengths [m]			masses [kg]			moments of inertia [gm^2]				
l_1	0.450		m_1	0.191		J_1	3.22			
l_2	0.400		m_2	0.170		J_2	2.26			
			$m_{\mathrm{j},2}$	0.10						
			m_{e}	0.30						
positions										
pick-position			place-position 1			**place-position 2**			place-position 3		
y_0	0.250	m								
x_0	0.385	m	x_{d0}	0.555	m	x_{d1}	0.680	m	x_{d2}	0.805	m
$\phi_{\mathrm{u},1}$	0.387π	rad	$\phi_{\mathrm{d0},1}$	0.093π	rad	$\phi_{\mathrm{d1},1}$	0.052π	rad	$\phi_{\mathrm{d2},1}$	-0.057π	rad
$\phi_{\mathrm{u},2}$	0.306π	rad	$\phi_{\mathrm{d0},2}$	-0.399π	rad	$\phi_{\mathrm{d1},2}$	-0.299π	rad	$\phi_{\mathrm{d2},2}$	-0.139π	rad
friction										
μ_{C}	0.48	Nm								
μ_{v}	0.00	Nms/rad								

weight with package of 1.7 kg (Table 2-1) concentrated at the end of a 1 m long arm extended perpendicularly to the direction of gravity, the maximum static moment is approximately 17 Nm. Adding a safety factor 5 for the dynamics, $r = 2.5$ cm and $t = 2$ mm leads to a maximum stress of 21.6 MPa, which is still well below the yield strength of the material.

Assuming half of the weight of the tube can be saved by some clever topology optimised open wall structure, the link weight per meter amounts to $\frac{1}{2}t\pi r\rho_{\mathrm{Al}} = \frac{1}{2} \cdot 0.002 \cdot \pi \cdot 0.050 \cdot 2.70 \times 10^3 = 0.424$ kg/m. This is comparable to the values used in [8], which are between 0.30 and 0.53 kg/m. Inertia of the separate links is taken to be $\frac{1}{12}ml^2$ which is the second moment of inertia of a rod.

The second part of Table 3-2 shows the positions of the shoulder and the arm configurations for the different end positions. In this chapter, only the second place-position will be used for optimisation.

The friction coefficients in the final section of the table are adopted from [10].

3-5 Trajectory optimisation

Now that the system model is fully defined, the trajectory can be optimised. Since the spring model now known, there is no more need to consider the dynamics as function of position. No special methods are required to optimise the trajectory in time.

3-5-1 Method

Now optimising in time, as conventional, no longer special care has to be taken to avoid zero velocity during integration. On the other hand, in space the start and the endpoint of the

trajectory were known, namely the end positions of the half-cycles. The trajectory end time simply followed from the system dynamics. Now the end time has become an extra variable, while the end positions have to be added as constraints to ensure they are reached at the end of each half-cycle. The following subsections will remark briefly on the different components defining the optimisation.

3-5-1-1 Cost function

The same as in the Cartesian case, the system energy consumption is minimised. However, since now the equations are kept a function of time, the I^2R losses can simply be integrated over time. Division by velocity is no longer necessary, as was the case in (2-15).

$$E = \int T_{\mathrm{m}} n_{\mathrm{s}} ds + \int I^2 R dt + P_{\mathrm{oh}} t_{end} \tag{3-17}$$

3-5-1-2 Trajectory parameterisation

This time plain acceleration is taken as the trajectory variable. The acceleration is optimised as eight values equally spaced in time for both up and down. The intermediate values are obtained by linear interpolation.

As a result of the final times (one for down and one for up) are in principle additional trajectory parameters to be optimised, this definition of the accelerations result in the acceleration as function of time becoming a nonlinear function of the eight 'corner-values'. No proper results were obtained from the optimisation when the two final time vales were added to the trajectory parameter vector.

In Chapter 2 it was found that the best results took close to one second per half-cycle. Therefore it was decided to predefine the time for the motion in the first place as 1 s for going down and 1 s for going up. The optimisation is run again for different final times in order to evaluate the effect on the performance.

3-5-1-3 Constraints

The final states are imposed as explicit constraints, ensuring the system ends op at the desired positions with zero velocity. Implicitly also zero acceleration is ensured in the end states. This is achieved by only optimising the accelerations between the end states and defining zero acceleration at the time the end state should have been reached.

3-5-1-4 Multistart optimisation

The same manual implementation is used as described in Section 2-3-1-3. Optimisations are done in batches of 500.

Table 3-3: Energy results to the optimisation for different cycle times, negative numbers for recovery show loss

k_s^T [Nm/rad]		cycle time [s]	mechanical loss [J]	I^2R loss [J]	total energy cost [J]	recovered energy [J]	recovered power [W]
0.102	0.243	2.0	5.17	2.42	5.78	2.22	1.11
0.127	0.205	2.2	4.90	2.06	5.95	2.85	1.29
0.082	0.148	2.4	4.78	1.83	6.40	3.20	1.33
0.069	0.152	2.5	4.74	1.73	6.66	3.34	1.34
0.071	0.135	2.6	4.71	1.67	6.97	3.43	1.32
no spring		2.0	8.83	7.94	15.0	-6.96	-3.48
		2.5	8.31	7.09	15.6	-5.59	-2.23

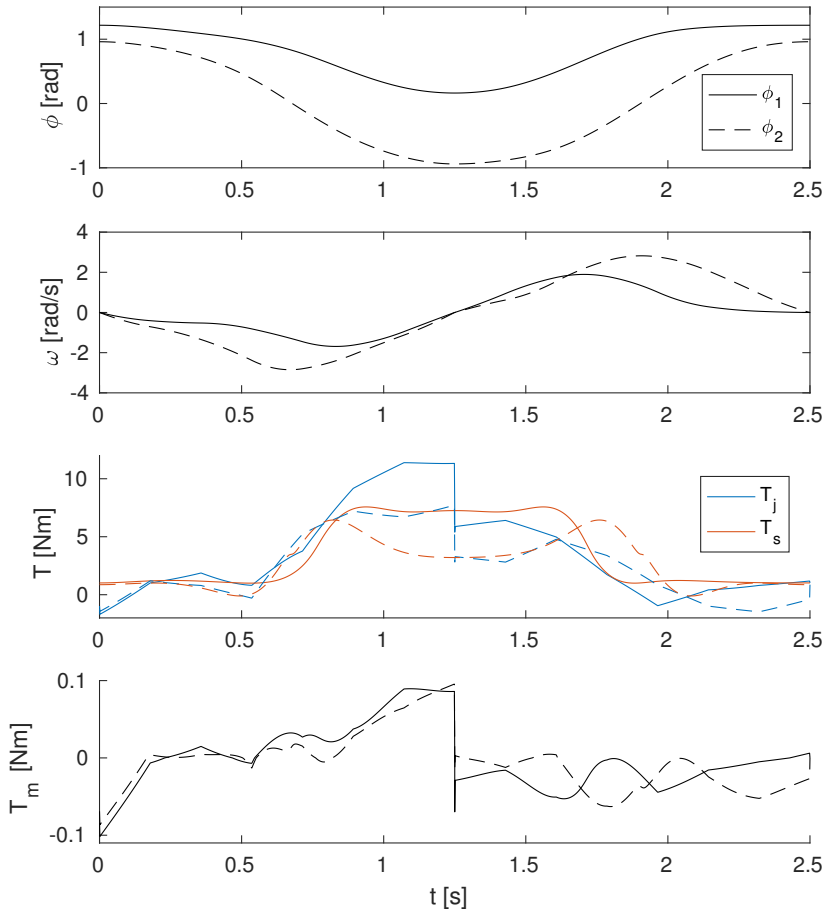


Figure 3-8: Optimal trajectory with when applying the combined spring mechanism

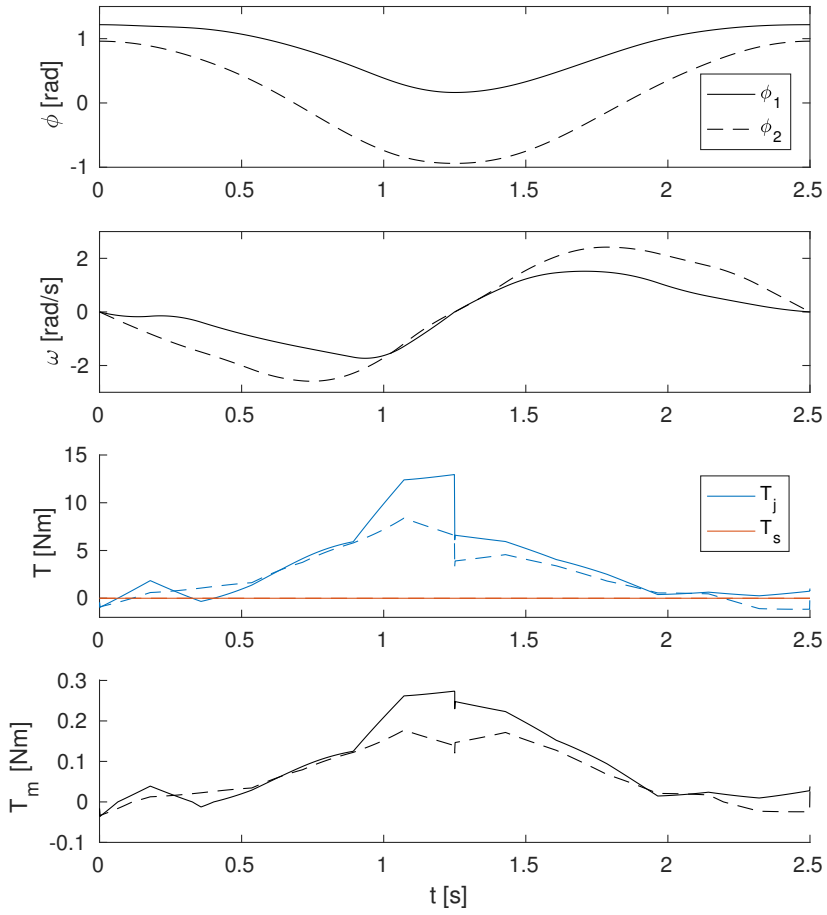


Figure 3-9: Optimal trajectory without springs

3-5-2 Results

Table 3-3 lists the optimisation results for different end times. Figures 3-8 and 3-9 show the optimised trajectories with and without springs respectively for a cycle time of 2.5. Optimal trajectories for different cycle times look very similar, only the motor torques get slightly smaller as the cycle time gets longer. The position and velocity follow a smooth trajectory, however the actuator torques that were found optimal show some bumps. Most of the time the motor torques are seen to have opposite sign with respect to the velocity and the controller runs in generator mode.

The comparative case without spring in Figure 3-9 shows actuator torques which are a factor 2 to 3 larger. The downward part of the cycle is again used for the major part for energy recovery. But now the upward motion has to be powered by the actuators without help from any spring. Now the upward part of the trajectory only costs a lot of energy, where in the case with springs energy could even be recovered. The numbers to the large difference in energy consumption in Table 3-3 show the springs achieve an energy cost reduction of about 9 J, which is about 60% of the combined mechanical and electric energy losses of the rigid

system. The springs have achieved a reduction in the mechanical energy cost of 43% and an electrical energy cost reduction of a factor 4. (The cost reduction is somewhat less for a shorter cycle time.)

Table 3-3 shows increased energy recovery for increased cycle time. However, with the cycle time the overhead energy consumption increases as well. The balance of the trade-off shows in the last column of the table, where the recovered energy is divided by the cycle time to show the recovery in terms of power. Maximum power is recovered for a cycle time of 2.5 s. For that cycle time 34% of the total provided mechanical is recovered.

The relatively large mechanical losses that remain in the system are due to three factors: Coulomb friction, transmission efficiency and reflected inertia of the drive train. With the current system parameters, Coulomb friction and low transmission efficiency are the dominant. Reduction of these factors would increase the energy recovery significantly.

3-6 Discussion

The percentage of the energy recovered with the spring mechanism in this chapter is considerably less than predicted in the previous chapter by the system without reflected inertia. This time reflected inertia is not the main factor reducing the energy efficiency. In the polar arm considered in this chapter, a relatively large number for Coulomb friction is assumed, especially when compared to the zero friction assumed for the vertical Cartesian case in the previous chapter. Also the transmission efficiencies have been considerably lower.

The motions of the arm considered in this chapter have been quite restricted by only allowing $-0.4\pi < \phi < 0.4\pi$. For an increased range of motions, and through that increased applicability of the arm, a way should be found to balance the arm on both sides of the vertical. However, with only one spring as described in Section 3-3-1 this is not possible.

So far the optimisations have been done for a single trajectory from one pick-position to one other place position. Ideally we would have a single spring per joint such that the system has several low-energy trajectories to multiple goal positions. However, one gravity balancing spring balances the arm in only two specific positions.

In some positions it is possible to achieve low system stiffness, but usually just in a single angle at a time. How low the stiffness has to be for stable positioning in a deviated end position depends on the static friction keeping the system in place. Ideally for low energy consumption, (dynamic) friction is low. It would in this case be useful to achieve a combination of negligible dynamic friction and high static friction. As a consequence a torque peak would be required to pull the system free when starting a motion, but this merely needs to be an impulse.

Even so, low stiffness with respect to the static friction will only achieve relatively small freedom of end position. Something else is required if different goal positions really require different configuration angles. Mechanisms for variable stiffness and increasing pretension add motors to the system and are therefore generally not desired.

A mechanism with a large potential is the Bi-directional Clutched Parallel Elastic Actuator (BiC-PEA)[25]. It does add another actuator to the system, however in combination with statically balanced brakes[26] the energy consumption can be very low.

The BiC-PEA also has the potential to improve energy recovery even when considering only a single pick and place position. Engaging the spring when going towards an end position and disengaging near the end position at near standstill when high tension is built up, a large mechanical torque to boost the return motion can be suddenly applied by reengaging the spring at a desired moment. The spring torque can even be applied in the same direction in which the spring was originally tensioned.

Even without springs, the system was found to consume little power. The 6.2 W, consumed by the system when the result without spring from Table 3-3 is combined with the assumed 4 W overhead power consumption, is still less than the 11.5 W used by the world's most energy efficient robot: Cornell Ranger[5]. Instrumental for this very low power consumption is the capability of the drive train to convert mechanical energy back into electric energy. Still, the spring mechanism designed in this chapter reduced the system power consumption even further by almost 3.6 W form 2.2 W to -1.3 W, passing the boundary below which power is actually recovered. Whether or not the recovered power will be enough to keep the full system, including overheads, running without an additional power source depends on whether the control can be designed so light that the required processor and sensors need no more than the available single Watt of power.

3-7 Conclusion

This chapter showed the mechanical design of a spring mechanism for energy efficient parallel elastic actuation of rotational joints in a polar robot arm. The mechanism adds a gravity balancing spring with a sinusoidal torque characteristic to the mechanism designed by Plooij. With the spring mechanism, an energy recovery of 3.34 J is achieved over a task cycle of 2.5 s. This is a large improvement with respect to the 5.59 J each cycle would have costed without springs. Note the verbal sign difference between the recovery and the cost.

Due to large Coulomb friction and low transmission efficiency, the amount of electric power retrieved is at most 1.34 W. This is less than the estimated 4 W required for the overheads. The next chapter will show how this will be enough.

Part II

Control Design

Chapter 4

Local linear optimal control for trajectory stabilisation

The result of the previous chapter is a spring mechanism for the 2 degree of freedom (DoF) arm which achieves maximum power return when the arm performs its task under nominal circumstances. However in the real world there are always disturbances. Therefore, to make the system more robust against these disturbances, an additional feedback controller is designed in this chapter.

The optimal nominal trajectory with a corresponding optimal feedback control law is designed using Differential Dynamic Programming (DDP)[12, 14]. The method uses the full nonlinear system to optimise the feedforward control signal in combination with the nominal trajectory. Along the trajectory, at each time step an LQR state feedback gain matrix is computed which is the local optimal linear policy for correcting state errors. Disturbances, both originating from the environment as well as from system model errors, are modelled as Gaussian state noise around the nominal trajectory. The resulting increase in actuator energy consumption will be evaluated.

Section 4-1 will explain the main principles of the optimisation method and where the stochastic noise enters. In Section 4-2 the settings of the DDP algorithm for the Plugless Arm will be determined such that the actuation energy of the full controller can be obtained from the result. The optimisation result is discussed in Section 4-3. The nominal trajectory is compared to the result of the previous chapter and the performance of the additional feedback controller is evaluated in terms of energy efficiency, precision and target accuracy under disturbed conditions.

Apart from the actuator losses which should be minimised it is also important to keep the processor energy consumption as low as possible. Section 4-4 will look into the computational energy cost of the controller and other aspects that require consideration. Based on the conclusions of that section, a reduced controller implementation is proposed in Section 4-5.

The findings of this chapter will be discussed in Section 4-6 before Section 4-7 concludes this last substantive chapter of this thesis report.

4-1 Method

The DDP algorithm takes the initial state and an initial control input sequence. Furthermore it requires the system dynamics to compute the next state from the previous state given a control input, and a cost function to evaluate a {state sequence, control sequence}-tuple. Both the cost function and the description of the system dynamics have to be continuously differentiable with respect to the state. In a forward pass the algorithm computes the state progression which is an recursive expression of the state and function of the control inputs. Subsequently the control inputs are optimised locally recursively from the final state back to the initial state using local linear approximations of the state dynamics and cost function around the trajectory obtained in the forward pass. These forward and backward passes are computed alternately until convergence. When the solution has converged, the algorithm returns: the nominal state trajectory, the corresponding feedforward control inputs and a list of state feedback gain matrices, one for each time step. The `iLQG` MATLAB function is used as supplied by Tassa [14].

4-1-1 Discrete system dynamics

The algorithm uses the discrete system dynamics describing the next state as function of the previous state and the input applied to that previous state:

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \tag{4-1}$$

Around a nominal trajectory $\left\{\bar{\mathbf{X}}, \bar{\mathbf{U}}\right\}$ with $\bar{\mathbf{X}} := \{\bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_N\}$ and $\bar{\mathbf{U}} := \{\bar{\mathbf{u}}_0, \mathbf{u}_1, \ldots, \bar{\mathbf{u}}_{N-1}\}$ where any $\bar{\mathbf{x}}_i$ with $i > 0$ is obtained using (4-1), the system dynamic deviation from the nominal trajectory can be linearly estimated.

$$\delta\mathbf{x}_{k+1} = A_k\delta\mathbf{x}_k + B_k\delta\mathbf{u}_k \tag{4-2}$$

with

$$A_k = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k} \tag{4-2a}$$

$$B_k = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k} \tag{4-2b}$$

4-1-2 Cost function

The cost function consists of a final term l_f which is only a function of the final state $\mathbf{x}_N$ and a number of running cost terms $l(\mathbf{x}_i, \mathbf{u}_i)$ which can be a function of input as well as state at each intermediate discrete time instance. The "cost-to-go" at a state k, with input sequence $\mathbf{U}_k := \{\mathbf{u}_k, \mathbf{u}_{k+1}, \ldots, \mathbf{u}_{N-1}\}$, is then given by:

$$\mathcal{J}_k(\mathbf{x}_k, \mathbf{U}_k) = \sum_{j=k}^{N-1} l(\mathbf{x}_j, \mathbf{u}_j) + l_f(\mathbf{x}_N) \tag{4-3a}$$

where $\mathbf{x}_j$ for $j > k$ are obtained by means of (4-1). In recursive notation:

$$\mathcal{J}_N(\mathbf{x}_N, [\,]) = l_f(\mathbf{x}_N) \tag{4-3b}$$
$$\mathcal{J}_k(\mathbf{x}_k, \mathbf{U}_k) = l(\mathbf{x}_k, \mathbf{u}_k) + \mathcal{J}_{k+1}(\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \mathbf{U}_{k+1}) \tag{4-3c}$$

Then the optimal cost-to-go per time step, starting from $k = N$, depends only on the input of the time step under consideration.

$$V_k(\mathbf{x}_k) = \min_{\mathbf{u}_k} [l(\mathbf{x}_k, \mathbf{u}_k) + V_{k+1}(\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k))] \tag{4-4}$$

For ease of optimisation, all parts of the cost function l, l_f are chosen such that they are quadratic in $\mathbf{x}$ and $\mathbf{u}$. Furthermore, given some nominal control signal $\bar{\mathbf{u}}$ corresponding to some yet to be optimised nominal trajectory $\bar{\mathbf{u}}$, the local linear control input deviation

$$\delta\mathbf{u}_k(\delta\mathbf{x}_k) = \mathbf{k}_k + K_k\delta\mathbf{x}_k \tag{4-5}$$

will be a local linear estimate optimising the trajectory at time k. At the same time, K_k provides the local optimal linear state feedback law. At the optimal nominal trajectory $\bar{\mathbf{u}}^\star$, $\delta\mathbf{u}_k(0) = 0 \Rightarrow \mathbf{k}_k = 0 \forall k$.

Reformulating the cost-to-go (4-4) as function of the state deviation $\delta\mathbf{x} = \mathbf{x} - \bar{\mathbf{x}}$ and applying the result of (4-5) which is linear in $\delta\mathbf{x}$, results in the following quadratic expression of the optimal cost-to-go:

$$v_k(\delta\mathbf{x}_k) = s_k + \delta\mathbf{x}_k^T\mathbf{s}_k + \frac{1}{2}\delta\mathbf{x}_k^T S_k \delta\mathbf{x}_k \tag{4-6}$$

where s_k, $\mathbf{s}_k$ and S_k are the combined scalar, linear and quadratic cost terms respectively, which are defined recursively as a function of the cost terms, system matrices and state feedback gains of the next time step. For more details the reader is referred to [12].

Important to know is that $\mathbf{k}_k, K_k$ optimising $\delta\mathbf{u}_k$ in (4-4) of which (4-6) is an alternative notation, are functions of $\mathbf{s}_{k+1}, S_{k+1}, A_k, B_k$ and the part of the cost l_k depending on $\mathbf{u}$, but not of s_{k+1}.

4-1-3 Including noise

Noise on the states is assumed to be normally distributed and, in case of the Plugless Arm, assumed independent of the input. The noise enters the system as an additional stochastic term in (4-2).

$$\delta\mathbf{x}_{k+1} = A_k\delta\mathbf{x}_k + B_k\delta\mathbf{u}_k + \mathcal{C}_k\boldsymbol{\xi}_k \tag{4-7}$$

with different nose modes $\xi_{k,i} \sim \mathrm{N}(0;1)$ for each of the state variables $i = 1, .., 4$. The matrix $\mathcal{C}_k$ contains weight factors scaling the standard deviation of the noise modes. The magnitude of the noise is taken constant for all k. Furthermore, the noise on the different states is assumed independent, therefore $\mathcal{C}$ is diagonal:

$$\mathcal{C} = \mathrm{diag}\left(\begin{bmatrix} c_{\xi,1} & c_{\xi,2} & c_{\xi,3} & c_{\xi,4} \end{bmatrix}\right) \tag{4-8}$$

The recursive term in the cost-to-go is no longer a deterministic value:

$$\mathcal{J}_k(\mathbf{x}_k, \mathbf{U}_k) = l(\mathbf{x}_k, \mathbf{u}_k) + \mathrm{E}\left[\mathcal{J}_{k+1}(\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \mathbf{U}_{k+1})\right]$$

Since the noise is assumed independent of state and input, translation to the form of (4-6) results only s_k receiving an extra term due to the noise:

$$\frac{1}{2}\sum_i \mathbf{c}_{\xi,i,k}^T S_{k+1}\mathbf{c}_{\xi,i,k}$$

However, this noise term will not influence the optimal trajectory, as s_k is the constant term in (4-6) and will drop out of any optimisation of the cost-to-go v_k with respect to $\delta\mathbf{x}_k$.

4-2 Optimisation settings

Again the downward and upward trajectories are optimised separately. The corresponding top and bottom position provide the initial states for the two half-cycles. The optimal control signal obtained in Chapter 3 is supplied as initial control sequence.

4-2-1 Cost function

The cost function in DDP has a backward recursive form: the cost of the final state, the final cost $l_f(\mathbf{x}_N)$ is a function of the final state only, the cost of every earlier state equals the cost of the next state plus a "running cost" term. This running cost $l_i(\mathbf{x}_i, \mathbf{u}_i)$ can be function of both the state and the control input at the specific time instant. In DDP the target state cannot be set by means of a constraint, as was done in the previous chapters.

The final cost is the weighted square distance to the target position:

$$l_f(\mathbf{x}_N) = (\mathbf{x}_N - \mathbf{x}_{\text{target}})^T C_f (\mathbf{x}_N - \mathbf{x}_{\text{target}}) \tag{4-9}$$

The weights in diagonal weight matrix C_f are chosen sufficiently large, such that the deviation in the final position and velocity of the end effector stays below 1 mm respectively 1 mm/s. The weights are all chosen equal to what from now on will be called c_f. Different values for c_f result in different nominal trajectories, different response to disturbances. Therefore the next subsection will evaluate a number of different values for c_f.

For the Plugless Arm, the running cost is the delivered work plus the I^2R actuator heat loss.

$$l(\mathbf{x}_k, \mathbf{u}_k) = \Delta t \sum_{i=1}^{2} \left(\left(I_{\text{noLoad}}\text{sign}(u_{k,i}) + \frac{u_{k,i}}{\bar{k}_\text{t}} \right)^2 R + u_{k,i} n_{\text{tr}} \dot{\phi}_{k,i} \right) \tag{4-10}$$

The sum of $l(\mathbf{u}_k)$ over all k is the electric energy consumption of the controller. Indeed (4-10) is the discrete equivalent of the energy cost equation of the previous chapter (3-17), but without the overhead power consumption. Since the final time has been fixed, the overhead term has become a constant and can safely be dropped for the optimisation.

4-2-2 System dynamics and state integration

The system dynamics are described in detail in Section 3-4. The spring stiffnesses of the Plooij mechanisms (Section 3-3-2) on the two joints are set to $k_{\text{s1}} = 0.07$ Nm/rad and $k_{\text{s2}} = 0.15$ Nm/rad respectively. These values are the rounded of result as presented in Table 3-3.

State integration in this chapter has been done using Euler's method.

4-2-3 Constraints

The provided implementation of the DDP algorithm supports bounds on the control input. Indeed the actuator torques should stay in the region $[-0.15, 0.15]$ Nm to ensure the gearbox, with reduction ratio 35:1 and maximum intermittent output torque of 6 Nm (Appendix A-3), will not break. However when the final state cost c_f is chosen wisely, the constraint is unnecessary, as the results of the unconstrained optimisation already stay within the bounds. Therefore the bounds on the optimisation are not activated. They would only slow down the optimisation.

4-3 Results

The controller is optimised multiple times, using final state costs:

$$c_f = 10^1, 10^2, 10^3, 10^4, 10^5, 10^6, 10^9$$

The resulting nominal trajectories are evaluated as well as the rejection of two types of disturbances.

4-3-1 Nominal trajectory

The nominal trajectories for the different final cost factors c_f are shown in Figure 4-1, the result to $c_f = 10^3$ plotted on top in black. The corresponding energy and power recovery values are listed in Table 4-1. The last two columns show the error in position and velocity in workspace coordinates (magnitude of the vector) of the final state of the end effector, either when having gone down or up depending on the which error was larger. For comparison, the table also repeats the result from the previous chapter.

For $c_f = 10^1$ the final state is outright inaccurate. For $c_f = 10^2$ the final state is already much closer to the goal state, but not yet sufficient. For the rest up to $c_f = 10^5$ both the optimal trajectories and the energy results are very similar with sufficient positioning

Table 4-1: Energy results of the nominal trajectory computed with DDP compared to the result of Section 3-5-2, all trajectories having a cycle time of 2.5s

	c_f	mechanical loss [J]	I^2R loss [J]	recovered energy [J]	recovered power [W]	max. end pos. error [mm]	max. end vel. error [mm/s]
from DDP	10^1	4.58	1.49	3.74	1.50	32.0	11.1
	10^2	4.62	1.54	3.65	1.46	2.84	1.50
	10^3	4.62	1.54	3.64	1.46	0.288	0.185
	10^4	4.63	1.55	3.64	1.46	0.025	0.016
	10^5	4.63	1.55	3.64	1.46	0.005	0.008
	10^6	5.37	3.24	1.19	0.48	0.003	0.004
	10^9	4.70	1.74	3.36	1.35	3.14	2.57
from Ch. 3		4.74	1.73	3.34	1.34	0	0

accuracy, increasing with c_f. When c_f gets really large $l_f(\mathbf{x}_N) >> \sum_k l(\mathbf{x}_k, \mathbf{u}_k)$ for most of the trajectories explored during the optimisation and the energy consumption of the trajectory only starts to matter once a trajectory has been found of which the final state lies really really close to the desired final state. As a result is is much more likely that the algorithm at the end ends up in some local optimum. This clearly happened for $c_f = 10^6, 10^9$. $c_f = 10^5$ appears to be the largest cost factor for which the optimisation still returns a proper solution. For larger c_f the optimisation results are no longer reliable because of the ill balanced cost function. The reason that the optimisation is done for very large values of c_f, even though the parameter appears to have no influence on the nominal trajectory, is for the sake of the feedback control, which will be evaluated in the next subsection.

Comparing the black lines, of position, velocity and motor torque for the two joints, in Figure 4-1 to the result of the previous chapter (Figure 3-8), the position and velocity profile are similar, only the velocity perhaps a bit less rounded of form. But improvement is apparent when comparing the bottom plots showing the motor torque signal. The DDP optimised actuator torques are much smoother. Comparing the results in Table 4-1, the DDP optimised nominal trajectory shows much lower I^2R loss and also decreased mechanical loss. In total less energy is dissipated and more power is recovered.

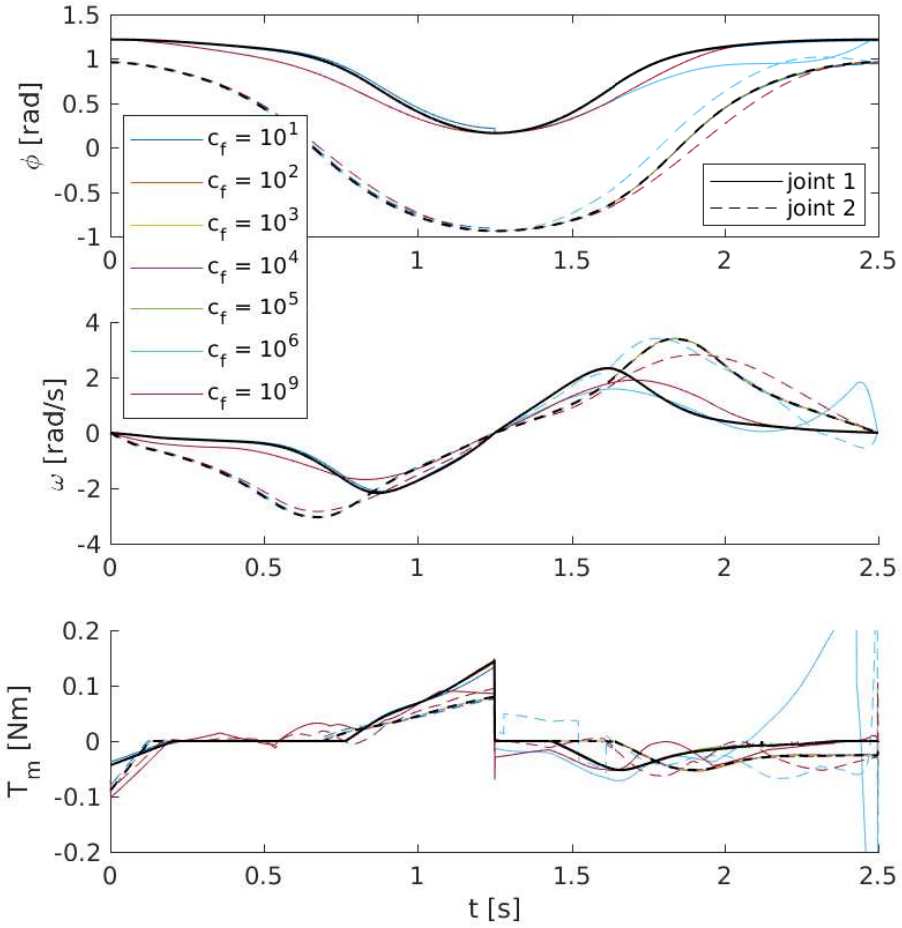


Figure 4-1: Optimal DDP trajectory for different final cost factors, for $c_f = 10^3$ extra plotted in black

4-3-2 State feedback

The feedback gains for $c_f = 10^2, 10^3, 10^4, 10^5, 10^6$ all show similar trends. The gains for $c_f = 10^3$ are shown in Figure 4-2. The spikes near the end of the cycle in blue are observed in some optimisation results. The gains increase as c_f increases, which is only logical since it means the relative importance of arriving at the desired final state is larger with respect to keeping the energy consumption low.

In all results the cross terms (right) are much smaller than the terms that feed back the link state to actuator controlling that same link (right). The velocity gains (bottom) show a spike at the start and end states of the first (downward) half-cycle. Apart from those specific spikes, the control gains are generally quite smooth over the cycle, increasing in magnitude towards the end of each half-cycle. At the beginning of each half-cycle the gains are small. For the velocity feedback this is more extremely the case.

Most differences are observed near the end of the second half cycle. In the case shown in Figure 4-2 the gains for the actuator on the first joint lose their smoothness over time. In the specific results to $c_f = 10^4, 10^6$, the position feedback terms (top) stay very low for the entire upward going half-cycle, not showing any increase towards the end at all. Near the end of the half-cycle a very narrow spike in the velocity feedback is still observed.

To avoid excessive control torques as a result of the large gains near the end of half-cycles, the actuator torque limits proposed in Section 4-2-3 will be applied to crop the control signal when necessary. If this poses too much of a limit on the feedback control, it can be considered to optimise the nominal trajectory with extra bounds on the torque.

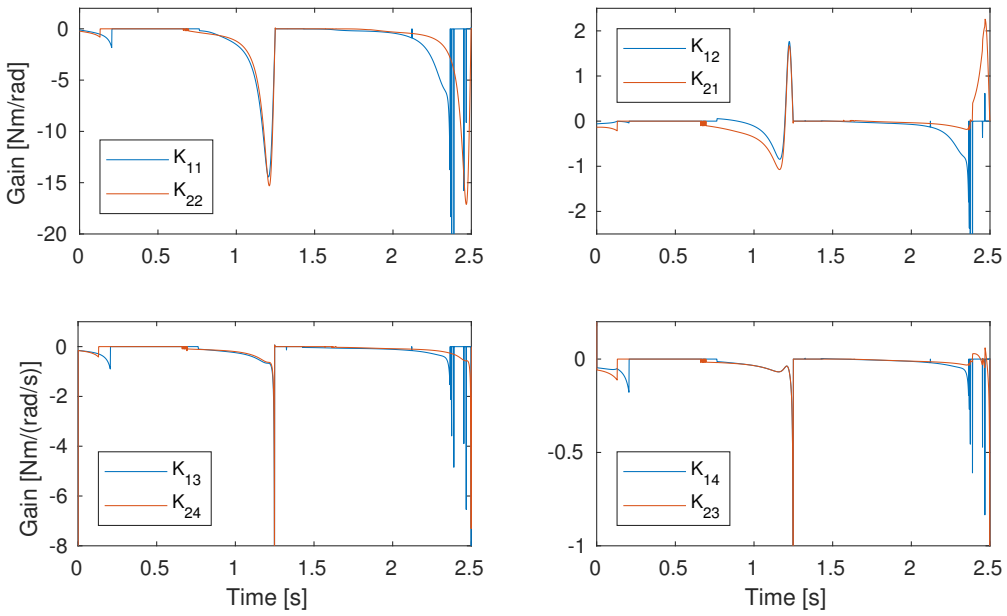


Figure 4-2: Optimal DDP feedback gains. Top: position feedback, bottom: velocity feedback, left: direct terms from link to corresponding actuator, right: cross terms from link to the other actuator. In bottom plots the values that fall outside the window at the first and the last state of the first half-cycle are up to several magnitudes larger than the rest of the feedback gains.

4-3-3 Disturbance rejection

Two kinds of disturbances are considered: a large initial position disturbance and continuous state noise.

4-3-3-1 Offset initial state

Figure 4-3 shows the trajectories obtained with two different controllers when starting a half cycle at a position very different from nominal. The initial positions (configuration space, joint angles) are listed in Table 4-2. In Figure 4-3 the resulting trajectories of the end effector are plotted in workspace coordinates, i.e. x and y position. The magnitude of the velocity is shown by the colours. The dark blue is zero velocity, the lighter the colour the larger the velocity. For reasons of clarity, the velocity of the nominal trajectory is not shown. The magnitudes of the endpoint deviations after one and two half-cycles respectively are also listed in Table 4-3, together with the energy consumption of the first two full cycles of the deviated trajectories. Note that the state errors listed in the table are with respect to the nominal trajectory from the optimisation. The end state error of the nominal trajectory, as listed in Table 4-1, is the maximum number that adds to this value when considering the end state with respect to the desired end state.

The feedback gains discussed in the previous subsection are valid close to nominal trajectory, around the point the system was linearised. The initial positions chosen for this test are definitely not close to the nominal initial positions (see Table 4-2). Yet the controller optimised for $c_f = 10^3$ (Figure 4-3a) shows convergence to the nominal trajectory in one half-cycle. The same is the case for the other controllers with significant position feedback near the end of each half-cycle: the controllers optimised for $c_f = 10^2, 10^5$.

The controllers to $c_f = 10^4, 10^6$ have next to no state feedback for the entire upward going half-cycle. They rely on large velocity feedback for deceleration when they have about arrived. It can be expected that as a consequence the disturbed trajectory takes longer to converge to the nominal one. The responses of the controller to $c_f = 10^4$ are shown in Figure 4-3b. Remarkably enough the upward motion does seem to achieve a large trajectory correction still. When disturbed in the bottom position (right), the end effector appears to reach the nominal top position at the end of the same half-cycle. Table 4-3 confirms that the position error at the end of the first half-cycle is indeed brought back to only few millimetres. However, the end velocity remains relatively large as the spike in the velocity feedback results in a control signal exceeding the posed bounds by far. This is the case for both down and up. As a result, the return motion still shows a large deviation from the nominal trajectory. More than one full cycle is required to get the end effector back on the original track within the accepted error margins of 1 mm and 1 mm/s respectively.

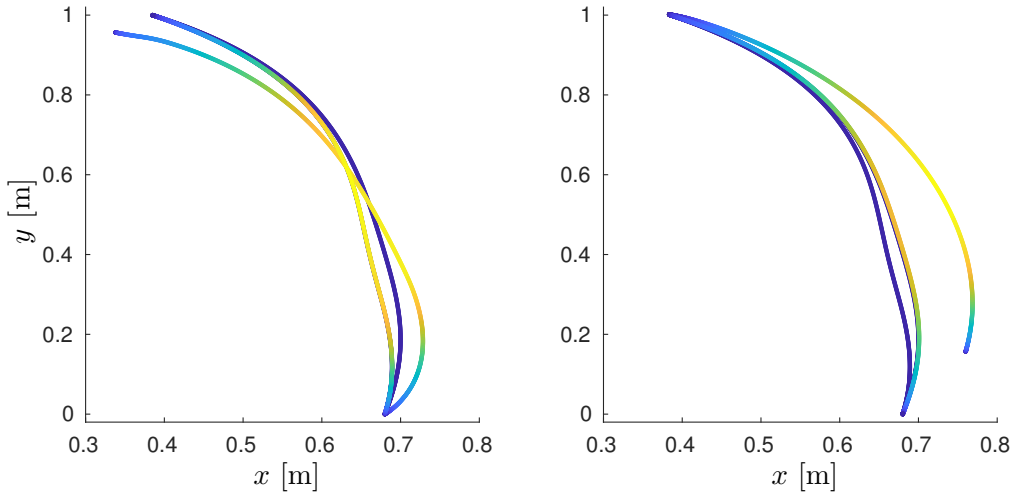
Table 4-2: Deviated starting positions of the trajectories shown in Figure 4-3

	deviated	nominal
ϕ_{top}^T	$\begin{bmatrix} 1.5 & 0.7 \end{bmatrix}$	$\begin{bmatrix} 1.217 & 0.961 \end{bmatrix}$
ϕ_{bottom}^T	$\begin{bmatrix} 0.3 & -0.6 \end{bmatrix}$	$\begin{bmatrix} 0.163 & -0.940 \end{bmatrix}$

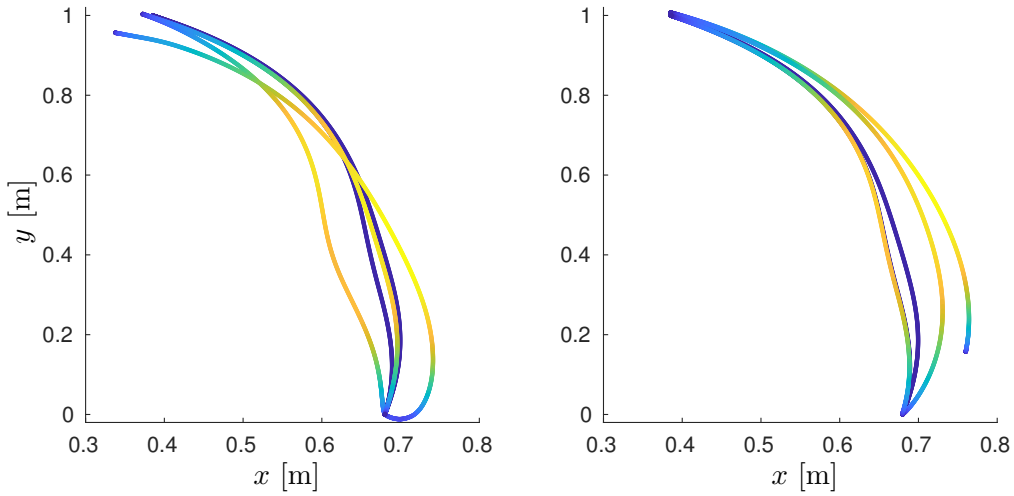
Table 4-3 shows the differences between the controllers with and without significant state feedback in the upward going half-cycle very clearly. The controllers with the feedback ($c_f = 10^2, 10^3, 10^5$) rapidly reduce the final state error each half cycle. The larger c_f, the smaller the final state error at the end of the first half-cycle.

Control signal cropping hardly occurs at all for $c_f = 10^2, 10^3, 10^5$, and then only for $c_f = 10^5$ in the last time step of the first half-cycle when starting from the deviated bottom position. This shows in the table as a larger final state deviation than expected when considering the result in the other direction being much more accurate compared to the result to $c_f = 10^3$.

Looking at the results to $c_f = 10^4, 10^6$, it stands out that the final state error after going up is not necessarily smaller than the error in the bottom position at the start of the half-cycle.



(a) $c_f = 10^3$



(b) $c_f = 10^4$

Figure 4-3: Trajectories in workspace started in alternative top position $(\phi_1, \phi_2) = (1.5, 0.7)$ (left) and bottom position $(\phi_1, \phi_2) = (0.3, -0.6)$ (right) respectively. The colour of the line shows the magnitude of the corresponding velocity vector at each state. The dark blue line is the nominal trajectory without the velocity shown.

Table 4-3: Energy recovery and end state errors to the deviated trajectories for the different controllers

		Retrieved energy [J]			Final state error (position [mm]; velocity [mm/s])	
c_f	deviated position	cycle 1	cycle 2	nominal	half-cycle 1	half-cycle 2
10^2	top	3.74	3.63	3.65	(2.2; 0.57)	(0.0003; 0.0004)
	bottom	2.60	3.62		(3.1; 0.57)	(0.00003; 0.00001)
10^3	top	3.73	3.63	3.64	(0.19; 0.054)	(0.00001; 0.00024)
	bottom	2.58	3.63		(1.0; 0.34)	(0.00000; 0.00000)
10^4	top	3.22	3.63	3.64	(2.0; 163)	(13; 10)
	bottom	2.62	3.64		(6.8; 42)	(0.68; 4.5)
10^5	top	3.74	3.64	3.64	(0.0018; 0.0005)	(0.0002; 0.0001)
	bottom	2.45	3.64		(0.65; 53)	(0.00014; 0.00004)
10^6	top	2.12	1.88	1.19	(0.0002; 0.0001)	(8.9; 282)
	bottom	0.99	1.88		(27; 209)	(0.0003; 0.0000)

This makes sense when considering the lack of position feedback observed in Section 4-3-2 in combination with excessive velocity feedback cropped low by the control input bounds. Disabling the bounds on the control signal show improved error correction, though still an order of magnitude worse than would have been expected when inter- and extrapolating the other error results.

Looking at the corresponding energy results listed in Table 4-3, apparently the first disturbance adds energy to the system while the second removes energy. Most controllers to recover extra energy during that first cycle. All controllers show convergence of energy per cycle after one cycle. This is also the case for the controllers which take longer to actually converge the trajectory. Interestingly, the last controller appears to converge to a more energy efficient trajectory than its nominal one. This is presumably due to the optimisation, being made unstable by the out of proportion cost function, having ended up in a local minimum.

4-3-3-2 Noisy state trajectory

Todorov and Tassa [27, 28] already warn for a poor response to additive noise. The reason for this is the fact that both the optimal trajectory and the feedback law are invariant to the noise term since the noise appears as a constant in the cost function (Section 4-1-3). Todorov and Tassa therefore propose an alternative controlle based on iterative Local Dynamic Programming (iLDP). This method no longer considers the optimal cost for a trajectory with only a single state per time step, but also for states in the neighbourhood. To support this in the optimisation the in DDP quadratic cost-to-go function is in iLDP extended to a cubic form, together with a quadratic (instead of linear) control policy. Effectiveness of the algorithm has so far only been proven in online optimisation. The increased computational demands of the controller, especially when it has to be implemented online, make the iLDP controller unsuitable for the Plugless Arm application.

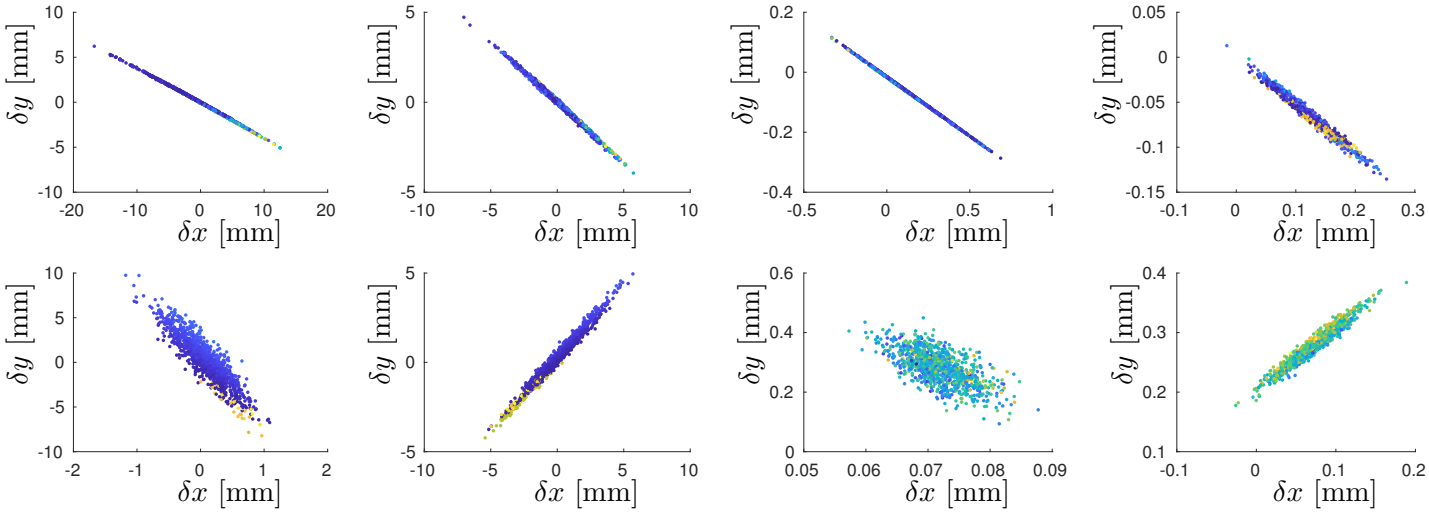
This section will evaluate the effect of noise on the current controller, optimised for $c_f = 10^3$, and how small the noise will have to stay for the system trajectory to stay within the accepted bounds of ± 1 mm, ± 1 mm/s respectively.

Because the noise is stochastic, the result is a probability distribution. The probability distribution is estimated using the Monte Carlo method, simulating 2000 trajectories for each noise distribution $\mathcal{C}$ listed in Table 4-4. Half of the trajectories go down, the other half go up. The point clouds of the end states show the form of the probability distribution of the end effector position at the end of a half-cycle. Figure 4-4 shows the end effector end states for noise applied to each of the states separately. Again the magnitude distribution of the velocity is shown by the colours of the dots.

The plots in Figure 4-4 clearly show the end positions distributed along the line that is the local direction of motion of the link that is disturbed. In the top position, link one has a larger angle than link two, which results in a lower slope of the range of disturbed end positions.

The results shown in Figure 4-4 correspond to the first four noise distributions of Table 4-4. The rest of the table shows different combinations of noise on the different states. Combinations of the separate noise modes result in point clouds with a form that is a combination of the point clouds seen in Figure 4-4. When combining the two position noise modes, the resulting top position distribution fans out between the two lines, but maintains an elongated shape in the general direction of the two lines. The bottom distribution on the other hand becomes more circular as the angle between the principal axes of the two separate distributions is large. The same holds for the velocity noise.

Noise on the first link has the largest the largest effect on the end effector disposition, which is not strange as it is the base link of the arm. It disturbs the top position more than the bottom position. For all the other noise modes, the bottom position is more sensitive.



(a) $c_{\xi,1} = 10^-3$ **(b)** $c_{\xi,2} = 10^-3$ **(c)** $c_{\xi,3} = 10^-3$ **(d)** $c_{\xi,4} = 10^-3$

Figure 4-4: End position distributions around the desired endpoint for the four single state noise modes. The top plots show the top positions, the bottom plots the bottom positions. Note that each subplot has its own axes range. Also the velocity colour coding is relative only within its own plot, i.e. the same colour in different plots stands most likely for a different magnitude of velocity.

Table 4-4: Standard deviations of the normal distributed noise modes and the resulting expected final state error and energy cost

Noise standard deviation [mm/Δt]		[mm/s/Δt]		Expected error position [mm]		velocity [mm/s]		Expected energy
$c_{\xi,1}$	$c_{\xi,2}$	$c_{\xi,3}$	$c_{\xi,4}$	top	bottom	top	bottom	cost [mJ]
1	0	0	0	3.86	2.20	1.99	0.21	58.5
0	1	0	0	1.81	1.95	1.95	0.28	49.8
0	0	1	0	0.21	0.29	0.87	0.19	2.52
0	0	0	1	0.13	0.29	0.35	0.19	0.153
1	1	0	0	4.24	3.12	3.20	0.30	105
0.5	0.5	0	0	2.06	1.61	1.42	0.21	29.7
0.2	0.2	0	0	0.82	0.69	0.67	0.19	6.01
0	0	1	1	0.22	0.29	0.91	0.19	4.24
0	0	0.5	0.5	0.16	0.29	0.53	0.19	0.303
0	0	0.2	0.2	0.14	0.29	0.38	0.19	0.343
0.2	0.2	1	1	0.83	0.67	1.10	0.19	6.52
0.2	0.2	0.5	0.5	0.80	0.67	0.74	0.19	5.78
0.1	0.1	0.2	0.2	0.42	0.41	0.49	0.19	1.79

Of all the separate noise modes, the velocity of the second link has the least disturbing effect. Also to all the separate noise modes, the velocity of the bottom position is the least disturbed. On the other hand, the inaccuracy of 0.19 mm/s in the bottom end point velocity appears to be a minimum that remains no matter how much further disturbances are lowered. Even so the lower bounds on the position accuracy of the bottom end point velocity appears to be 0.29 mm. For the rest, combinations of the different noise modes show no surprises.

In order for the final state disturbances to remain at an average below 1 mm respectively 1 mm/s, the state disturbances may not be larger than 0.2 mm and 0.5 mm/s per time step respectively. The combination of very low noise standard deviations of the bottom row of the table results in almost 99% of the bottom positions to stay within within the bounds of 1 mm and 1 mm/s, but still only 76% of the top positions satisfies these bounds.

The energy loss to these low amounts of noise are seen in the last column of Table 4-4 to be in the order of magnitude of milli-Joules. For noise which disturbs the system near bounds deemed acceptable, there is no significant extra energy lost.

4-4 Processor energy consumption

The energy consumed by the processor depends very much on the specific type of microprocessor and depending on the microcontroller there is a number of additional tasks, for example memory management, which are performed outside the processor itself, which also cost energy. For an exact answer how much energy the controller software really costs, measurements need to be done on the hardware. But this section will give an estimate about the order of

magnitude of the energy consumption of the software and and discuss the possibilities so that the robot arm with the proposed controller really can function pluglessly.

Components of microcontrollers already consume some energy by just being switched on, idly doing nothing. Additionally every software instruction is executed by switching transistors. Every transistor switch some small amount of energy is dissipated. This energy consumption is proportional to the number of software instructions that is executed.

To summarise how much energy is available for the Plugless Arm microcontroller: The mechanical system as designed in Chapter 3 with the DDP optimised controller, the second one from Table 4-3, yields a 3.64 J energy recovery over a 2.5 s cycle under nominal circumstances. This energy must power the additional control in case of disturbances, the processor for the control computations, but also the necessary sensors and the gripper being the end effector. A very energy efficient gripper has been designed for this robot arm, needing only 0.35 J per cycle [29]. Furthermore, assuming a sensor on each joint consuming 50 mW [4], over a cycle time of 2.5 s costs another 0.25 J. Disturbance rejection by the current controller cost little extra energy and part of the time even recovers extra energy from the disturbances Table 4-3. Keeping good half a Joule for coping with energy extracting disturbances, 2.5 J or 1 full Watt is left to power the microcontroller.

4-4-1 Number of processor instructions estimate

So far in MATLAB, all computations have been done using floating point numbers (floats). For this microcontrollers usually have a Floating Point Unit (FPU), which costs additional energy and is relatively slow. For high-speed simple control it is advised not to use floats[1]. The alternative are fixed point numbers which support decimal numbers, but are much closer to integers when it comes down to computing.

Not considering floats, calculations of the form $a \cdot b + c$ are generally considered one instruction. However, multiplication of fixed point numbers require some extra actions in the form of bit shifting. And of course there is always the required overflow checking and occasional handling of the resulting exceptions.

Furthermore, how hard it is to compute or do something on a (micro)controller/processor is usually measured in number of clock cycles, as so far the interest of most programmers is to minimise time and energy tends to be a less pressing concern. The number of clock cycles required for different types of instructions depends on the processor architecture, support for pipelining for parallel computing, etc. The simplest microprocessors just execute 1 basic instruction per clock cycle.

For the following back-of-the-envelope calculation I will assume:

- 2 instructions per addition
- 4 instructions per multiplication
- 2 instructions per min, max-operation
- 1 instruction for writing some value to some address (actuator for example)
- 1 instructions to load a value from memory or a sensor

[1] concluded after a several conversations with different people with experience in the area

Additional instructions may be required to locate something in memory, but if the data is stored efficiently, it should suffice to simply call the next data address.

To the following piece of pseudo-code, the number of software instructions according to the assumptions just made are added at the end of each line.

```
for each time step:                                                    xN
    load nominal control torques from memory                    m      2
    load state feedback gains from memory                     mxn      8
    load nominal state from memory                              n      4
    load current state from sensor                              n      4
    subtract the two states                                   2xn      8
    multiply the result with the state feedback gains       4xmxn     32
      and add the numbers                                 2xmx(n-1)   12
    for each of the two resulting numbers:                     xm    x2
        take the min with a constant                          2      2
        take the max with a constant                          2      2
    write the control torques to the actuators                  m      2
```

For the current 4-state ($n = 4$), 2-input ($m = 2$) system, this little piece of basic controller code costs 80 instructions. Multiplying that with the 2 times $N = 1000$ steps per half-cycle, results in a total number of 1.6×10^5 software instructions per cycle for the controller in its most basic implementation.

To understand how light this controller computationally is, consider computing the next state from the current state-input pair using the nonlinear system model (3-16). This requires computation of $M_{\mathbf{q}}(\mathbf{q})$, $\mathbf{f}_{\mathbf{q}}(\mathbf{q}, \dot{\mathbf{q}})$, $T_{\mathrm{f}}(\dot{\mathbf{q}})$, $T_{\mathrm{s}}(\mathbf{q})$, $Cn_{\mathrm{tr}}T_{\mathrm{m}}$ and combining these results to obtain $\ddot{\mathbf{q}}$ before integration can be applied to obtain the next state estimate. For these calculations, the previous list of numbers of instructions is extended by the following additional mathematical operations:

- 1 instruction for testing the sign of a number
- 1 instruction fir multiplying a number with the sign of another number
- 4 instructions for division, assumed similar to multiplication
- Square root – Several algorithms exist for taking square roots. A fast implementation is the Non-Restoring Algorithm [30]. Assuming one instruction per clock cycle in the paper, performing a square root operation takes 25 instructions.
- Trigonometric functions – The CORDIC optimised algorithm for computing sines and cosines requires one bit shift and an add/subtract operation per bit of the number [31]. Considering 16-bit numbers, a sine/cosine operation would cost 48 instructions.
- Matrix inverse – Instead of (left) multiplying a vector by the matrix inverse $\mathbf{x} = A^{-1}\mathbf{b}$, it is computational more efficient to obtain $\mathbf{x}$ by solving $A\mathbf{x} = \mathbf{b}$. Still, for a non-sparse $r \times r$ matrix such as $M_{\mathbf{q}}$ this takes $\frac{r(r-1)}{2}$ divisions, $\frac{r(r-1)(2r-1)}{6}$ multiplications and as many subtractions [32, Chapter 3]. With the previous assumptions, solving the matrix equation costs $2r(r-1) + r(r-1)(2r-1) = r(r-1)(2r+1)$ instructions, where in this case $r = \frac{n}{2} = 2$ leads to a total number of 10 instructions.

Applying these assumptions, the total number of instructions required just to compute the next system state is estimated at 1728. More details about how this number is obtained can be read in Appendix D.

Computation of the running cost term (4-10) takes one division, four multiplications (taking the square is the same as multiplying the value with itself) and two additions per input, plus the summation of the two inputs and the scaling with the time step, i.e. 54 instructions. The final cost (4-9) consists of eight subtractions, eight multiplications and a concluding three additions, also resulting in 54 instructions.

Now consider online optimisation using iterative Linear Quadratic Gaussian control (iLQG), which is a computationally lighter form of DDP, only taking into account the first order gradients of the system dynamics and cost with respect to the state and input. One iteration consists of a forward pass, estimation of the gradients of the dynamics and the cost function and a concluding backward pass. Especially when starting from a state close to the offline optimised trajectory and using that trajectory to initialise the optimisation, a single iteration should already provide a reasonably close to optimal result. The optimisation can be done from the current time step all the way to the end of the half-cycle with the final state deviation as final cost, or for a limited horizon, optimising only to get the system back to the pre-optimised trajectory in the next H time steps.

In the first case, at time step k of the half-cycle the optimisation has to be run over the next $N-k$ time steps, at the end of which the final state must be reached. The forward pass and numerical gradient estimation consist for the main part of $(1+n+m)(N-k)$ times computing the state and cost at the following time step. The backward pass consists of matrix-vector algebra, requiring separated a total of $(1+m+2(n+m)^2+3n^2(2m+n)+m^2n)$ multiplications, $(2n^3+6n^2m+3m^2n-n^2-2n+m^2+m)$ additions, solving of $2(n+1)$ 2×2 matrix equations and a Cholesky decomposition[2], all over $(N-k)$ time steps. The Cholesky algorithm for decomposing a 2×2 matrix [33, Chapter 7] takes two square roots, two divisions a multiplication and a subtraction: 64 operations, which brings the total number of instructions of the backward pass to $989(N-k)$. As a result, the total number of instructions for online optimisation of one half-cycle is $(7(1728+54)+989)\frac{N(N+1)}{2} = 6.74\times 10^9$, times 2 makes 13.5×10^9 instructions per cycle.

Optimising only over a small horizon $H = 10$ time steps, the number of required instructions would be $2N(7(1728+54)+989)H = 269\times 10^6$ instructions per cycle, 135×10^3 per time step. This is still over three orders of magnitude more than the proposed offline optimised controller.

So far only the computations required by the controller have been discussed, without any filtering on the incoming sensor signals. Depending on the sensors, some filter will probably have to be applied in order to obtain an adequate state estimate. This will add to the number of instructions. But even if a filter would double the number of instructions of the simple controller, the number remains very low compared to control methods based on model predictive methods and online optimisation.

4-4-2 Microcontroller energy consumption estimate

There are many different microcontrollers, ranging from very simple to about full computers in their own right.

[2]Obtained from the MATLAB code

The world's most energy efficient robot to date, the Cornell Ranger, uses a combination of several ARM type microcontrollers for its control [5]. The main control loop of Ranger runs on a ARM9 processor. Additional low level sensor monitoring and control is done by four separate ARM7 processors while two more are used for communications and display. According to the datasheets [34, 35] both ARM7 and ARM9 microprocessors have a maximum total power dissipation of 1.5 W with clock speeds upto 60 respectively 125 MHz and upto 40 KB RAM and 512 KB flash memory. The processors have 32-bit support and use pipelining for concurrent instruction execution and therefore faster operation.

For the Plugless Arm, the ARM microcontrollers would not be employable in their full capacity. Nor is their capacity required for the proposed controller. Unfortunately I was not able to gain insight in the distribution of internal energy consumption of the ARM microcontrollers. Instead I will give an overview two other devices designed for low-power applications.

The TMS320VC5510 is a fixed-point digital signal processor operating at a maximal clock speed of 200 MHz and having a maximum power consumtion of 730 mW [4, 36], which is about half of that of the ARM processors and well within the bounds set for the Plugless Arm. Of the different components, most power (upto 208 mW) is consumed by the External Memory Interface. This component is essential since the TMS320VC5510 itself is not a full microcontroller containing its own memory banks. The data required for the controller will therefore have to be accessed from some external location. Other necessary processes require together between about 100 and 365 mW. The CPU takes between 3 and 158 mW from idle to running at 100%, consuming 0.8 nJ per instruction. For the number of instructions required for the controller as computed in the previous subsection, this would amount to a computational energy consumption of 128 μJ. This is an insignificant amount of energy compared to the power consumption of the other processor components, and also compared to the other losses in the system.

Scoring even lower on energy consumption are Arduino microcontrollers. They are full microcontrollers with integrated memory, though a limited amount. The Arduino Nano [37] is the smallest microcontrollers of the Arduino family and lightest in energy demand. The base power consumption is only 95 mW, though code appears to cost 5 nJ/instruction based on measurements presented in [38]. As a result the controller implementation of the previous subsection would cost 0.8 mJ, which is still a negligible amount.

However if instead the online iLQG controller, computed in the previous subsection for comparison, would be implemented with its three to five orders of magnitude larger number of required instructions, the controller power consumption would rise to the order of magnitude of mW, even W. But then also the clock speed would become a problem. The online optimisation for the 10-step horizon controller, with the current control frequency of 800 Hz, would require a minimal clock speed of 110 MHz, and additional margin is advisable. The Arduino Nano would not support this and it will be very tight on the ARM9.

Table 4-5 summarises the energy specifications of the different microcontrollers/-processors discussed in this section, together with their clock speeds and flash memory. It is clear that with the proposed precomputed linear controller the online computational energy consumption and available clock speed will not be the problem. But with the number of precomputed values, memory may become a serious issue.

Table 4-5: Summary of microcontroller/-processor properties

Processor	ARM9 [34]	ARM7 [35]	TMS320VC5510 [4, 36]	Arduino Nano [37, 38]
Max. power consumption	1.5 W	1.5 W	730 mW	179 mW
Base power consumption			100 - 365 mW	95 mW
Controller power consumption			51.2 μW	320 μW
Clock speed	125 MHz	60 MHz	200 MHz	16 MHz
Flash memory	512 KB	512 KB	-	32 KB

4-4-3 Memory limitations

Small microcontrollers have limited memory. Especially when considering microcontrollers as small as the Arduino Nano this becomes a serious issue. Even when external memory can be added at will, like to the TMS320VC5510 microprocessor, large memory slows down data retrieval to the point where it can become the bottleneck of the program, especially when the program is based mostly on lookup tables, like the Plugless Arm controller.

The memory required for the current controller design can be computed using the following numbers:

- 16-bit numbers take the space of 4 bytes per number
- 2 nominal control values
- 4 numbers for the nominal state
- 8 numbers per feedback gain matrix
- 2000 time steps per control cycle

The result is $4(2+(8+4))2000 = 112$ KB required memory, 16 KB for the feedforward signal and 96 KB for the feedback controller.

The Arduino Nano however has only 32 KB, which includes 2 KB used by the bootloader and also the memory in which the program instructions have to be stored. The following section will explore if the controller can be reduced to fit within the available memory of the Arduino Nano.

4-5 Controller reduction

From the previous section it was concluded that not so much the number of computations needs to be reduced as the number of values that need to be stored in memory. However, the effect for the controller is the same: the number of computations must be reduced.

4-5-1 Structure

The current controller runs at 800 Hz, with its 2000 discrete time steps over a cycle of 2.5 s. This is the case for both the nominal feedforward signal and the feedback loop. In this section two different methods for controller reduction are proposed:

1. Reduction of the control frequency

2. Reduction of the feedback to only part of the cycle. After all, Section 4-3-2 showed that during a large part of the cycle the optimal feedback gains are next to zero.

In order to get the memory usage down from 112 KB to at most 25 KB, the controller frequency would have to be reduced to a mere 160 Hz. However when optimising the controller at this low a frequency, no stable solution is obtained. On the other hand, the feedback control would have to be reduced to only 9% if the controller data memory usage is to stay below 25 KB including the 16 KB required for the feedforward control. Too little of the feedback will remain. Therefore a combination of the two reduction options is required.

The first step is to reduce this nominal frequency. A reduction to 500 Hz is chosen, still yielding a proper nominal trajectory. Then the memory required for storing the nominal control values is reduced from 16 KB to 10 KB. The memory required for the feedback part of the controller would still be 60 KB.

It was already seen in Section 4-3-2, that after an initial push on the velocity, the feedback gains had insignificant magnitude until about the last quarter of the half-cycle. This suggests that it may be sufficient to add the feedback term only near the end of each half-cycle. Switching on feedback control only at the final quarter of each half-cycle would reduce the memory required for the feedback part of the controller further to 15 KB.

The resulting combined controller would need only 25 KB of memory. This would fit on the Arduino Nano. A remaining 5 KB of memory would be left for storing the program instructions, which is ample for this simple controller.

4-5-2 Optimisation

The main difference with respect to the earlier DDP optimisations is that there is no initial near optimal control signal available to start the optimisation from. Previously the nominal control signal obtained in the previous chapter could be used. This time however, the number of sampling instances is lower. Instead the optimisations (one for each half-cycle) are initialised with zero control signal.

Furthermore, the final cost coefficient c_f is set to 2×10^3.

4-5-3 Performance

The system performance in term of positioning accuracy will not improve with reduced control. The question is whether the reduction in accuracy will be small enough.

First only the frequency reduction is tested. The nominal result is shown in Figure 4-5 and its energy properties are listed in Table 4-6. For comparison, the result for 800 Hz is repeated in the table. Only little more energy is lost with the lower frequency controller, and the maximum final state error magnitude in workspace coordinates of the end effector is comparable. The trajectory in Figure 4-5 shows a slight overshoot in the position of the first joint when returning to the top position. Related to that, the control torque on the first joint near the end of the cycle is positive, actively (regeneratively) braking the link returning to the position it overshot.

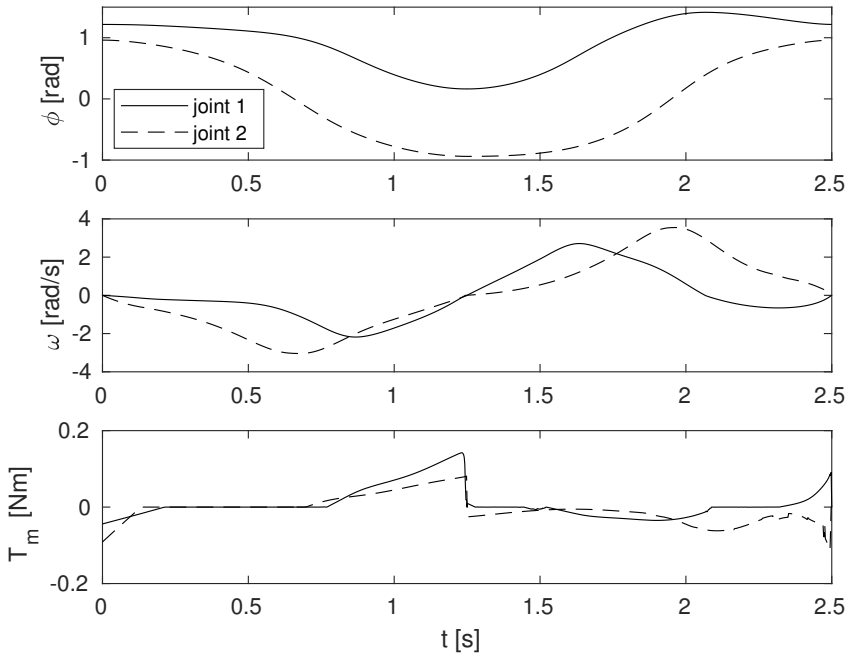


Figure 4-5: Optimal trajectory controlled at a frequency of 500 Hz

Table 4-6: Energy results of the nominal trajectory computed with DDP compared to the result of Section 3-5-2, all trajectories having a cycle time of 2.5s

control frequency [Hz]	mechanical loss [J]	I^2R loss [J]	recovered energy [J]	recovered power [W]	max. end pos. error [mm]	max. end vel. error [mm/s]
500	4.72	1.60	3.48	1.39	0.207	0.226
800	4.62	1.54	3.64	1.46	0.288	0.185

Figure 4-6 shows the optimal feedback gains to the trajectory. It is striking that there is hardy any feedback during the entire upward half-cycle. During the downward half-cycle there is significant feedback during about the first fifth and the second half of the motion.

Two levels of feedback reduction are proposed. The first one keeps the feedback in the downward half-cycle for the first 108 cycles. That is to the point where the blue line of the velocity feedback gains in the bottom plots of Figure 4-6 drop back to zero. Also the feedback is kept from halfway down to the end of the half cycle. The second reduction only keeps the feedback in the second half of the downward cycle. Both reduced controllers drop the feedback during the upward half-cycle since there is hardly any feedback there anyway.

Figure 4-7 shows the end effector position and velocity error at the end of each half-cycle for five full cycles for the full feedback controller and the two reduced feedback controllers. With this lower frequency control, but probably mainly because of the lacking feedback control in the upward half-cycle, the final state error does not get reduced to an invisibly small number, as was the case before. Instead it keeps oscillating, being closer to zero in the bottom position and less close to zero in the top position. The oscillations seem to remain bounded, a constant type of oscillation.

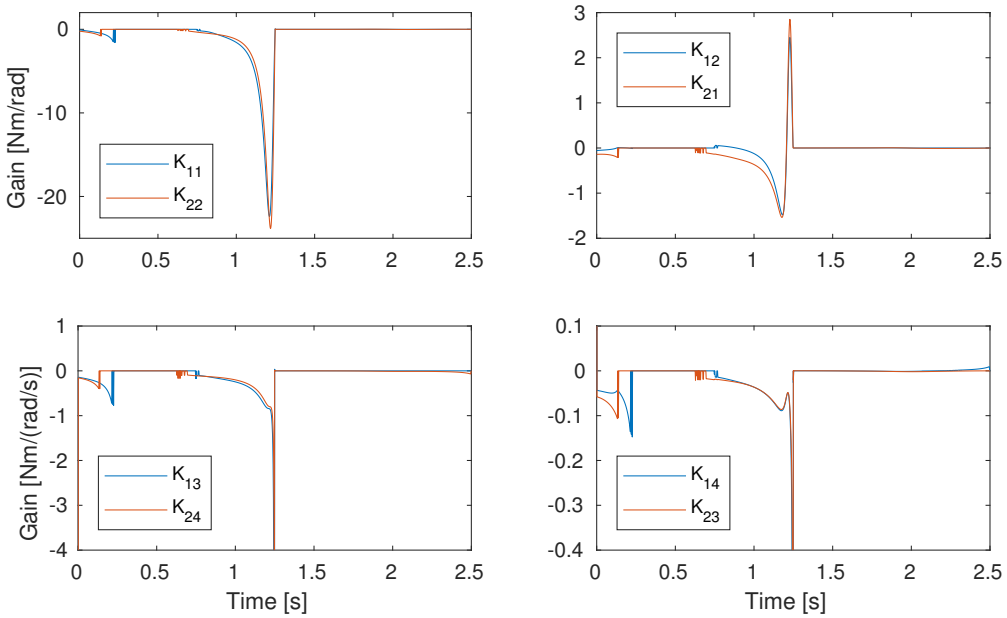


Figure 4-6: Optimal DDP feedback gains. Top: position feedback, bottom: velocity feedback, left: direct terms from link to corresponding actuator, right: cross terms from link to the other actuator. In bottom plots a few large values fall outside the window.

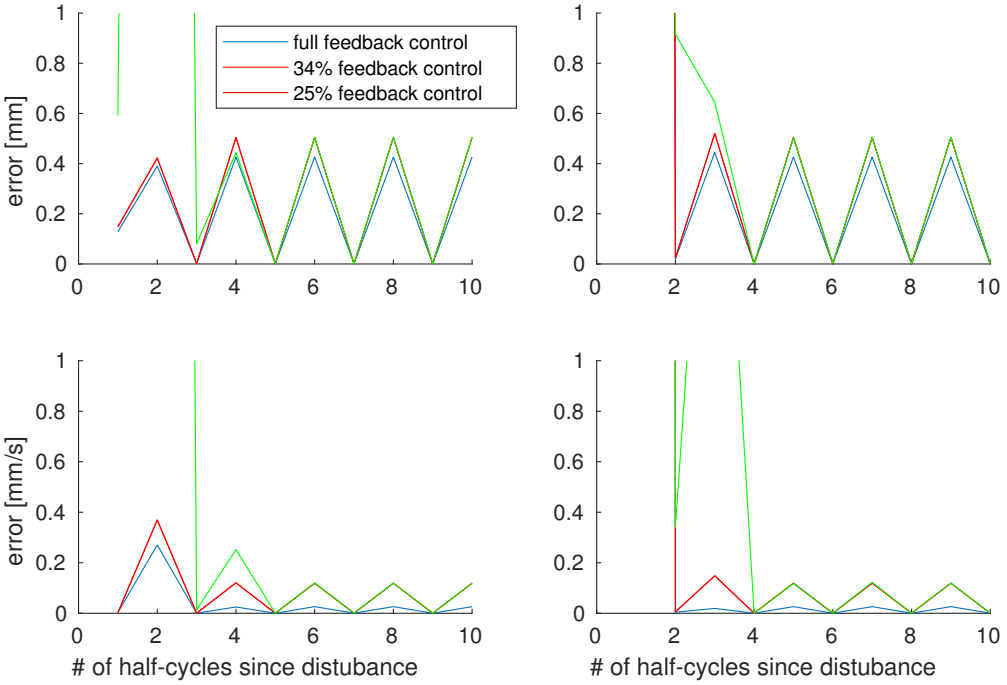


Figure 4-7: Position errors (top) and velocity errors (bottom) for the different 500 Hz controllers. In the left plots the system was started from a deviated top position, in the right plots from a deviated bottom position.

Table 4-7: Energy recovery to the deviated trajectories for full and partial 500 Hz feedback control, corresponding to the results shown in Figure 4-7

controller	deviated position	Retrieved energy [J] cycle 1	cycle 2
full feedback	top	3.59	3.49
	bottom	2.77	3.48
34% feedback	top	3.58	3.49
	bottom	2.59	3.48
25% feedback	top	2.77	3.47
	bottom	2.32	3.49

When keeping 34% of the feedback controller, the magnitude of the error becomes only slightly larger. Convergence to the steady oscillations still happens in the same amount of half-cycles. When disturbed above, the position is corrected within the first half-cycle, when disturbed below, it takes one half-cycle extra, since the upward half-cycle exerts little control. When keeping only the proposed quarter of the feedback controller, an extra full cycle is required before the system arrives at the same small oscillating error as with the slightly less reduced feedback law.

The steady state oscillations in all cases stay below 0.5 mm respectively 0.2 mm/s, which is perfectly acceptable. Not acceptable is the extra full cycle the 25% controller needs to get the system there.

Table 4-7 shows the energy recovery corresponding to the different trajectories of the different controllers. Again even though the trajectory of the second cycle may still look wildly different from nominal, the energy recovery values have returned to normal after only a single cycle.

The 34% controller needs a little more energy to get back on track after the bottom disturbance, but only 0.18 J, which is about 6% of the energy recovered during the cycle. The 25% controller loses another 10% of the energy that would otherwise have been recovered. More important is that the 25% controller even needs additional energy to correct the top deviation, while both other controllers achieve that correction with increased energy gain.

A look at the nominal torques in Figure 4-1 show that this is the part of the trajectory in which usually most positive work is delivered by the actuators, i.e. the part of the nominal trajectory that costs energy. The feedback control at the beginning of the cycle reduces this energy loss as much as the disturbance allows. This is the part of the feedback controller which harvests the excess mechanical energy added by the disturbance. It is important not to remove this part of the controller.

4-5-4 Final notes

It is not strictly necessary to reduce the controller as a slightly larger microcontroller than the Arduino Nano would still be supported by the Plugless Arm. There is the trade-off: On one hand a larger microcontroller with more memory will consume more energy, even/especially when not computing. On the other hand the tighter control would be able to recover more energy from the system.

With the 34% feedback controller, additional 5.2 KB of memory would be required. Considering only 5 KB was left for the program code, this is too much. Choosing a microcontroller with the additional capacity, the additional capacity is probably enough to support the full controller. However, the example of the extra controller reduction shows that large part of the feedback controller is simply unnecessary.

Also now the choice was made between either full state feedback or no feedback at all. More clever choices can be made by also allowing locally for single or partial state feedback. It is very likely that with this method enough extra memory can be saved to make the controller fit on the Arduino Nano.

An additional option for memory reduction is to store each number in only 8 bits instead of 16. The effects of the reduced precision will have to be tested together with the fixed-point implementation. After all, the test so far have all still been done in MATLAB, using floats. It is more accurate to test the bit lenght of the numbers in combination with the fixed-point implementation directly on the microcontroller.

With the use of smaller numbers, more numbers can be stored and a higher control frequency can be maintained wit all its benefits.

4-6 Discussion

A large point of improvement is the noise rejection. The suggestions on the topic by Todorov and Tassa [27, 28] are too computational intensive for the current application. Some other method will have to be found.

Proper noise rejection is important for functioning in the physical world where environmental factors like air displacement will act on the system in an unpredictable way. But not only external factors will disturb the system in a way that can be modelled as additive stochastic noise. Inherently the system model will contain some deviations with respect to the real system. Small deviations originating from estimation errors or negligence of higher order nonlinear factors such as friction may be modelled as noise.

If it turns out that some system parameters are not perfectly known or change over time, some form of adaptive control might need to be considered. Though this requires the control law to be a function of the model parameters, which is not the case if the controller is optimised offline before hand the way the current controller is.

With the current controller the computational costs are low. There is room to do more online computing, but not enough to support full online optimisations, which require several orders of magnitude more computations than the current simple controller.

The final state error has been evaluated in workspace coordinates, in mm and mm/s, while the term appearing in the cost function is in configuration space, in rad and rad/s. I chose the evaluation in workspace coordinates because the error of the end effector state will be visible in workspace coordinates. Conform to this, it might have been wiser to have formulated the penalty on the final state also though the state transformation in workspace coordinates.

Also several methods exist for adding constraints to the DDP control problem [39], for example using Lagrange multipliers [40]. None of these methods is currently supported by the provided

MATLAB algorithm. Furthermore, replacing the final state cost by an end state constraint will also have an effect on the computed optimal feedback gains. Then there will no longer be a factor in the cost function weighting the importance of arriving at the desired end state with respect to minimisation of the control input. I do not dare to speculate on the consequences for the feedback control.

Furthermore, in this chapter state integration was done using Euler's method, which provides a less accurate estimation of the state progression than the Runge-Kutta 4^{th} method used in the previous chapters. It may also be due to this increased integration error that a difference of energy recovery of the nominal trajectory as large as 0.3 J per cycle is observed with respect to the optimisation results presented in Chapter 3.

As the time between sampling instants becomes larger, Euler integration will cause larger integration errors. At some point these errors will become significant and seriously deteriorate the optimisation result.

However when the Euler method was replaced by the Runge-Kutta 4^{th} method (RK4), remarkably enough results did not improve. When optimising the controller in Section 4-3 using RK4, a proper nominal trajectory was found also for $c_f = 10^6$. The nominal trajectories were very similar to the ones shown in Figure 4-1 and the energy values were very close to those presented in Table 4-1. However except for the inaccurate controller to $c_f = 10^2$, none of the controllers showed any significant feedback gain during the upward half-cycle. Disturbance rejection was accordingly poor, as discussed in Section 4-3-3 for the controller to $c_f = 10^4$.

Also the optimisation appeared sensitive to the trajectory used for initialisation. Seemingly more so for RK4 than for Euler integration.

I suspect that the disappointing performance of RK4 has to do with the finite difference approximations used so frequently in DDP. Using Euler integration, the first order derivative obtained by the finite differencing algorithm exactly retrieves the state derivative. This is no longer the case when RK4 is applied. It may be that the reduced accuracy of the first and second order terms of the quadratic approximation of the system dynamics and cost function result in reduced optimisation performance. Though the fact that the optimisation "knows" that it is merely locally approximating the full dynamics and cost would suggest that these errors should be so small that they have no such influence.

4-7 Conclusion

The nominal controller is very energy efficient. The controller is able to retrieve 3.64 J from the system over a cycle of 2.5 s, resulting in a power generation of 1.46 W. This is even slightly more than was found when the system was optimised in Chapter 3.

The controller is capable of correcting large disturbances of the initial position within one half-cycle. Depending on the disturbance the correction either costs additional energy or extra energy can be regenerated from the correction.

To continuous state noise the controller responds less well, or actually just less. In a sense this is positive as is it means negligible energy is lost on countering noise. On the other hand,

state noise is not really countered. The controller can only disturbances up to a fraction of a millimeter per time step without losing accuracy of the pick and place positions.

The computation costs of the controller are very low. So low in fact that they are negligible with respect to the base line power consumption of some very energy efficient microprocessors. Because the controller depends on looking up a considerable number of precomputed values, it is important to choose a microcontroller with sufficient memory for data storage.

Taking all together, it is definitely possible to implement the controller on a microprocessor using at most the 1 W that is available from the Plugless Arm.

Chapter 5

Discussion

The mathematical method to determine the theoretical optimal spring is only applied to the Cartesian model. The mechanism fit is reasonably well, but more suitable mechanisms might exist. It depends on how low the internal friction can be made how much mechanical complexity can be added to the mechanism. On the other hand there is the question whether additional complexity is really required for a better fitting spring mechanism.

Translating the mechanism from Cartesian to polar configuration proved sufficiently effective in terms of sufficient energy recovery. However increased friction loss and reduced transmission efficiency turned out to be instrumental in the lower amount of anergy that could be retrieved in the polar arm.

The possibility certainly exists that the full prior analytical optimal spring analysis would have insights leading to a more efficient spring mechanism. The analytical method making time a function of position is less straigthforward for more complex systems including multiple degrees of freedom, but would still work as long as there is one DoF with a monotonic motion between defined endpoints, i.e. not changing direction during motion. Then the positions of all other DoF, as well as all velocities and time, can be made a function of the DoF. Then for that DoF, the optimal spring characteristic can be computed. For multiple springs, the DoF have to be decoupled. Constraining the end time to the same value for all DoFs, the optimal spring can be computed separately for each DoF, provided the position coordinates can be chosen in a way without interdependence. As a result, design for more degrees of freedom becomes increasingly complex very fast for the analytical optimisation method.

Furthermore, in the mechanical design part of this thesis an optimal spring mechanism has been designed to a 2 DoF arm. The arm itself has not been optimised. Furture optimisation of the arm will almost certainly increase the energy that can be recovered from the system.

The same holds for optimisation of the pick and place positions in angular coordinates. The spring design method of Section 3-3 can be extended to more DoF relatively easily, given the desired final states. For the current 2 DoF, the choice of the end state angles depended only the choice of position of the shoulder joint. The choice of shoulder placement was made on

considerations of reach. Another choice might have lead to a more efficient pick and place motion. The more DoF the arm has, the more free variables can be chosen, which will have an impact on the maximum achievable energy efficiency of the motion. This opens up an entire new space for optimisation. A space which increases rapidly in dimensionality for each additional DoF. The right choice for the endstates is its own complex optimisation problem, a challenging field for future exploration.

Then there is the controller. Differential Dynamic Programming works well for finding very energy efficient and effective nominal control trajectories. The feedback policies proved unpredictable, at least for this system which was mechanically designed to be self-stable in the target states. Minimal feedback control regularly amounted to no feedback at all. In the cases that significant feedback gain was found during both halves of the task cycle, recovery from a displaced starting position was adequate enough. But in most cases significant feedback was only obtained during the downward half-cycle. The state deviation would then grow during the upward half-cycle. In total the error measured at only one of the end positions would, if not decrease, at least stay bounded over time. But for real convergent behaviour some state feedback should be enforcable at least in the neighbourhood of each end position. The noise rejection capabilities leave even more to be desired.

The universal problem of effective noise and uncertainty handling in a computationally light way remains. The current controller is computationally very light, i.e. there is room for additional online computation to improve disturbance rejection. A method different from DDP or iLQG is required for design of a feedback controller which can guarantee disturbance rejection within definite bounds.

Not only have I been designing the controller for its control properties, I also wanted to stress the importance of implementability and the considerable energy that can be lost in the area if choises are not considered carefully. My background is in mechanical engineering and systems and control Engineering, not in electronics, embedded systems, algorithmics or some of the other fields I touched in this thesis. But by touching these fields I wanted to show the importance of them all in relation to eachother. This is not just the case for the Plugless Arm, but for any system combining elements of such a multitude of fields. The best designs are obtained when an interdisciplinary team optimises the system over all the interconnected fields simultaneously, or at least iteratively, each field considering the others as well.

Insights from control have been instrumental in the mechanical design of the Plugless Arm presented in this thesis. And controller design is of no use if the resulting controller cannot be implemented. On the other hand the mathematical possibility to view the system from a different perspective, position instead of time, has led to valuable new insights. The electromechanic properties of electric motors allow for electric energy regeneration, but only if other factors comply, such as drive trian backdrivability, but also the circuitry should not overheat from return currents. And a large number of questions, such as how to store the recovered electric energy in an efficient way and in a way that the components who require it are provided with a stable voltage, remain yet to be solved.

Chapter 6

Conclusion

In this thesis a spring mechanism for optimal parallel elastic actuation and a controller have been designed for the two degree of freedom (DoF) Plugless Arm. First the parallel spring has been optimised analytically. By rewriting the system dynamics such that all became a function of position, including time, the analytical optimal spring could be described as a function of the optimal trajectory without any need for additional parameterisation.

Based on the characteristic of the analytical optimal spring, a mechanism was designed based on simple mechanical principles, such that internal losses stay minimal. The proposed mechanism in combination with the proposed controller achieves an optimal energy retrieval of 3.64 J out of the available 9.81 J, over a cycle time of 2.5 s. The pick and place endpoints of the cycle are mechanically stable and maintaining these position costs in itself no energy.

The major factors responsible for the remaining energy loss are: the general friction (modelled as Coulomb friction), relatively low transmission efficiency and the considerable reflected inertia of the drive train. Yet even despite those losses, the recovered energy over the cycle time provides 1.46 W of power. This power is sufficient to power an energy efficient gripper, sensors, a light microcontroller and save some for additional disturbance rejection.

Occasionally recovery from a disturbance may reduce the energy consumption of a cycle, but there are also situations in which extra energy can be recovered from a disturbance. Precomputed state feedback gains achieve local optimal control while requiring negligible computational power from the processor, only sufficient memory. Implementing only a part of the feedback control law is found to be as effective as use of the full feedback controller, reducing the memory required for storing the precomputed values as well as the number of online computations.

The result is an extremely energy efficient robot arm controlled by a very light controller.

Appendix A

Specs and choices of drive train elements

A-1 Motor

The motor by Maxon is chosen for its high torque, low speed output, such that minimal additional transmission is required.

From the motorspecifications[41]:

Nominal voltage	24	V
Nominal current	6.06	A
Nominal torque	0.444	Nm
Nominal speed	2590	rpm
	271	rad/s
Stall torque	4.94	Nm
Stall current	70	A
No load speed	3190	rpm
	334	rad/s
No load current	0.538	A
Maximum efficiency	83	%
Terminal resistance	0.343	Ω
Torque constant	70.5	mNm/A
Corrected constant	71.1	mNm/A
Rotor inertia	3060	gcm^2

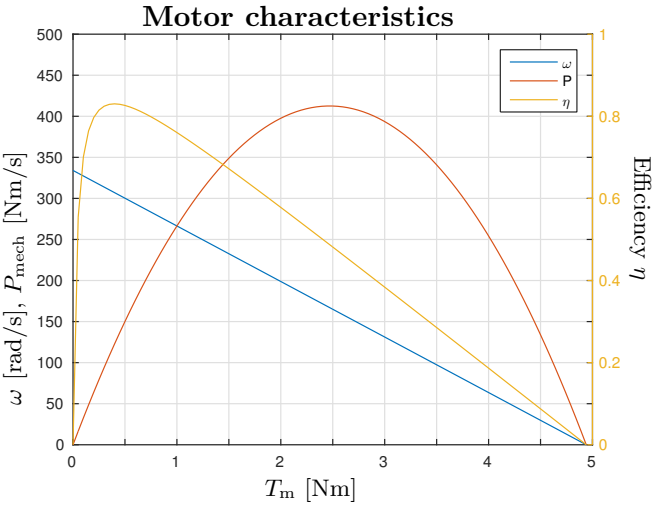


Figure A-1: Speed, torque, power and efficiency relations of the brushless DC motor

Figure A-1 shows the full motor torque, speed and power characteristics, as well as the efficiency. Optimal efficiency is achieved for torques in the range 0.2 - 0.8 Nm ($\eta \gtrapprox 0.8$).

A-2 Spindle

Based on preliminary calculations with guessed transmission ratio and friction parameters, an estimation of the velocities, actuation forces and required motor power is made:
$\|v\| < 8$ m/s
$\|F\| < 25$ N

Based on these bounds, a spindle is chosen for a reasonable approximation of parameters in the linear drive required in Chapter 2.

Spindles have speed limits, in the catalogues [19, 42] speed limits are given as the product of the shaft rotation speed ω_{rpm} in rpm and the nominal diameter d_0 in mm. Precision rolled ball screws with recirculation through a flange (further referred to as long lead ball screws) have a permissible speed limit of $\omega_{\text{rpm}} d_0 \leq 90000$. Planetary roller screws have a permissible speed limit of $\omega_{\text{rpm}} d_0 \leq 160000$ [42].

The motor described in Section A-1 may turn up to 3190 rpm, which is its no load speed. So in case of a long lead ball screw $d_0 \leq 28$ mm, in case of a planetary roller screw $d_0 \leq 50$ mm. However, the motor is so strong that for delivering 10 N to the load with 1 Nm of motor torque, a spindle reduction ratio of 10 is required. Consequently the spindle requires a minimal lead $P_{\text{h}} = \frac{2\pi}{10} \approx 0.6$ m. However, the largest available lead for a spindle which can handle the required speed is only 30 mm (planetary roller screw with $d_0 = 48$ mm). On the other hand, only a tenth of the power the motor can supply is required.

Friction

The dynamic coefficient μ of friction of long lead ball screws is 0.006. In planetary roller screws, the practical dynamic friction coefficient μ_{prac} depends strongly on the helix angle $\alpha = \text{atan}\left(\frac{P_{\text{h}}}{\pi d_0}\right)$ and varies from 0.009 for negligible angles up to 0.035 for larger angles. The static friction coefficient μ_{s} felt when motion is initiated, is taken to be twice the dynamic friction coefficient. Only Coulomb friction is assumed by SKF.

Efficiency

Efficiency in forward direction (motor driving load) is given by

$$\eta = \frac{1}{1 + \frac{\pi d_0}{P_{\text{h}}}\mu} \tag{A-1}$$

In case of long lead ball screws the practical efficiency is then given by $\eta_{\text{p}} = 0.9\eta$. For planetary roller screws the practical efficiency is simply obtained by calculating (A-1) using μ_{prac}.

Efficiency in the backward direction is given by

$$\eta' = 2 - \frac{1}{\eta}$$

Selection

Friction is smallest for long lead ball screws. However if $v_{\max} = 3.5$ m/s is to be achieved:

$$\frac{60 v_{\max}}{\frac{P_\mathrm{h}}{1000}} d_0 < 90000$$

$$\frac{P_\mathrm{h}}{d_0} > 2\frac{1}{3}$$

Unfortunately the largest available ratio $\frac{P_\mathrm{h}}{\pi d_0}$ for long lead ball screws is only 1.25, $P_\mathrm{h} = 40$ mm and $d_0 = 32$ mm. As a result the maximum obtainable linear speed is only 1.8 m/s. In this speed limited case, the practical efficiency would be $\eta_\mathrm{p} = 0.89$. The resulting backward efficiency $\eta'_\mathrm{p} = 0.87$.

In the case of planetary roller screws, which can handle higher speeds,

$$\frac{P_\mathrm{h}}{d_0} > 1.3$$

Unfortunately the largest ratio $\frac{P_\mathrm{h}}{\pi d_0}$ that is found in the planetary roller screws is only 0.67. With the planetary roller the same maximum speed of 1.8 m/s is achieved. Only with the resulting helix angle of $\alpha = 12°$, the practical friction $\mu_\mathrm{prac} > 0.04$ and the practical efficiency is only about 0.8.

The dynamic loads in the system are of such low value, that it is no issue.

For the final rotational 3 DoF robot arm there will be a gear instead of a spindle and speed limits will be less of a problem. For the purpose of Ch. 2 it is sufficient to select a hypothetical spindle with reasonable properties. It is assumed that the long lead rolled thread ball screw with $d_0 = 32$ mm and $P_\mathrm{h} = 40$ mm can actually handle the required speeds.

TL 32x40R

Nominal diameter d_0	32	mm
Lead P_h	40	mm
Preload T_pr (optional)	0.05 -0.50	Nm
Nut mass	0.65	kg
Nut length	54.8	mm
Screw inertia	490	kgmm^2/m

Preload torque is recommended for high positioning accuracy under load. The standard precision: over a distance of 1 m, the lead error is still below 0.1 mm. This is sufficient for the application.

A spindle length of 1.06 m is required to cover the 1 m distance, including the length of the nut. As result, the spindle inertia J_s amounts to 519 kgmm^2.

The spindle reduction ratio n_s in rad/m relates to lead according to $n_\mathrm{s} = \frac{2000\pi}{P_\mathrm{h}}$, resulting in a spindle reduction ratio of $n_\mathrm{s} = 157$ rad/m.

A-3 Gears

The acceleration profile of the arm in Chapter 3 is first optimised without any additional gear transmission. These preliminary results are presented in Apendix B-2. The gears need to be able to handle the output torque, upto about 8 Nm, though not continuously. The preliminary results also show the motor heat losses to be three orders of magnitude larger than the recovered mechanical energy. A reduction ratio of about 30:1 is required to get the I^2R losses within reasonable bounds. This reduction will also result in the torques required from the motor to lie in the area where the efficiency is largest. Such reduction cannot be achieved by a single gear stage. The choice was made for the two-stage planetary gear with maximum reduction ratio by Maxon [43].

GPX 32 HP ∅32 mm, 2-stage

Reduction ratio n_{tr}	4554/130	
Max. continuous torque	4	Nm
Max. intermittent torque	6	Nm
Maximum efficiency	0.76	
Mass inertia	13	gcm^2
Outer diameter	32	mm
Gearhead length	46.3	mm
Max. transmittable power (continuous)	110	W
Max. transmittable power (intermittent)	140	W

A larger reduction ratio through an additional gear stage would result in even lower efficiency. Also for this reduction ratio, the reflected inertia is still reasonably small with respect to the torque driving the joint. With an increased reduction ratio the motor heat losses start increasing again, this time because of the additional power required to compensate for the reflected inertia of the motor and transmission.

Appendix B

Preliminary optimisation results for component selection

B-1 Preliminary results to Ch. 2

Assumed is the motor as described in Appendix A-1 with a minimal transmission to connect the rotational actuator to the translational load. The dynamic system model (2-4) is used.

$$\frac{d}{dy}\begin{bmatrix} \dot{y} \\ t \end{bmatrix} = \dot{y}^{-1}\begin{bmatrix} -g + \frac{F_\mathrm{j}}{m} \\ 1 \end{bmatrix}$$

The spring torque F_s is given by (2-13). The remaining torque is delivered by the motor $F_\mathrm{m} = F_\mathrm{j} - F_\mathrm{s}$. The simple motor model (B-1) is used, which is linear in the transmission ratio n_s.

$$T_\mathrm{m} = \frac{C}{n_\mathrm{s}} F_\mathrm{m} \tag{B-1}$$

$$C = \begin{cases} \frac{1}{\eta_\mathrm{tr}} & \text{when accelerating} \\ \eta & \text{when decelerating} \end{cases}$$

The efficiency η_tr is set at 0.8. A well manufactured transmission will in all probability have a higher efficiency, but for the moment the slightly pessimistic estimation will suffice.

Motor current $I = \frac{T_\mathrm{m}}{k_\mathrm{t}}$, and through it the I^2R loss, decreases with n_s, whereas refected inertia and friction increase with n_s. Furthermore, η_tr tends to decrease as the transmission ratio grows larger. For larger n_s the optimisation results will overestimate the recoverable energy. The optimisation is performed as described in Section 2-3-1. Also the overhead power in the cost function $P_\mathrm{overheads} = 4$ W.

Table B-1 shows optimisation results for different n_s grouped in two sections. The recovered power, bottom row of the first section, is the amount of power that is available to support overhead costs. With the 80% efficiency and no other losses, expected recovered mechanical

Table B-1: Optimisation results for different n_s, recovered power excl. overheads

spindle reduction ratio n_s [rad/m]	2π	5π	10π	20π	50π	100π
cycle duration [s]	3.31	1.89	2.00	1.74	1.99	2.09
recoverend mechanical energy [J]	7.84	7.84	7.85	7.83	7.80	7.65
energy lost to I^2R heating [J]	7.30	3.11	1.57	0.64	0.41	0.54
recovered power [W]	0.16	2.50	3.13	4.14	3.72	3.40
$\max(\lvert F_m \rvert)$ [N]	7.10	9.48	9.86	9.45	18.3	38.6
$\max(\lvert v \rvert)$ [m/s]	13.4	6.96	5.00	3.92	2.54	2.80
$\max(\lvert a \rvert)$ [m/s]	660	377	109	130	32.0	27.6
$\max(\lvert P_{mech} \rvert)$ [W]	92.9	65.4	44.3	34.0	45.0	33.7

energy would in all cases be 7.85 J. In the table it is seen the results come close for low transmission ratios. As n_{tr} grows large, the reflected inertia becomes significant and a decrease in recovered mechanical energy is observed. Large decrease in I^2R is witnessed for increasing n_{tr} at first.

The second section of Table B-1 mechanical limits of the trajectories are listed. The maximum motor torques are seen to decrease with transmission ratio, but not proportional to the transmission ratio. Highest velocities are observed for low and high transmission ratios.

Increasing transmission ratios also tend to have an increasingly deteriorating effect on the backdrivability. For maximum regeneration the load to drive backward transmission should be as direct as possible, which means n_s as small as possible.

Considering the peak forces and velocities, a transmission ratio of 10 rad/m would already be sufficient.

B-2 Preliminary results to Ch. 3

Figure B-1 shows the motor torques found when optimising the 2 DoF robot arm (Section 3-5) without additional transmission between the motors and their respective joints. Table B-2 summarises the minimum, maximum and average values of the control torque signal.

From Table B-2 it is seen, without any gear reduction between the motor and the joint, the I^2R losses are excessively large.

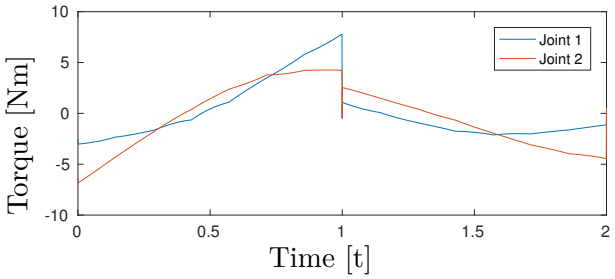


Figure B-1: Motor torques when driving the 2 DoF arm with gravity balancing springs without any additional transmission

Table B-2: Summary of torque values to Figure B-1

	maximum torque [Nm]	minimum torque [Nm]	average absolute torque [Nm]
First joint	7.80	-3.02	2.09
Second joint	4.27	-6.83	2.63

Table B-3: Energy results to the trajectory of Figure B-1

recovered mechanical energy	I^2R loss
6.97 J	2.30 kJ(!)

Appendix C

Additional optimisation results to Chapter 2

C-1 Analytical optimal spring

Extra trajectory results to Table 2-3.

With reflected inertia

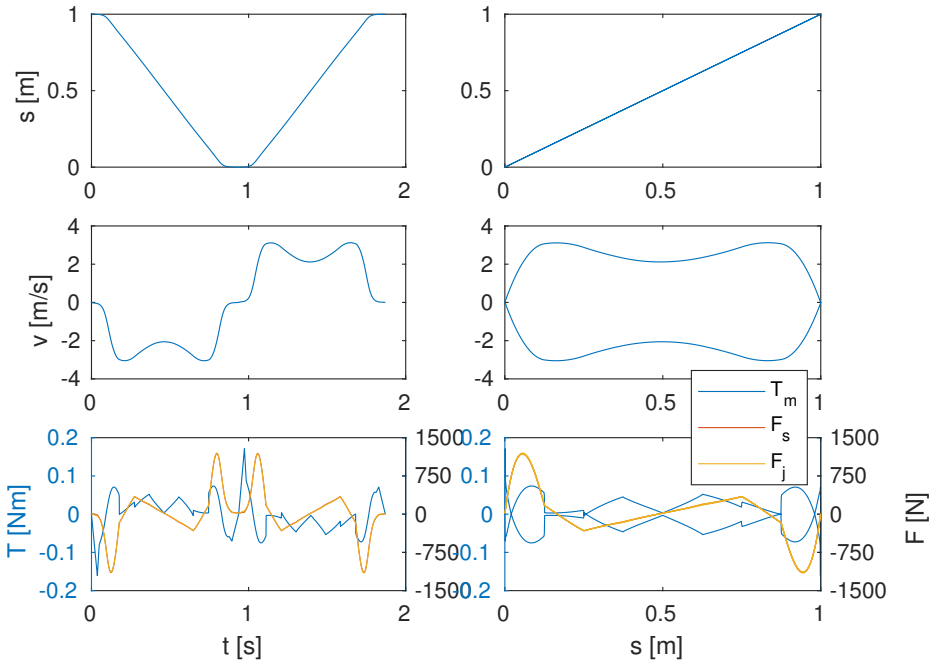


Figure C-1: Bumpy optimised trajectory

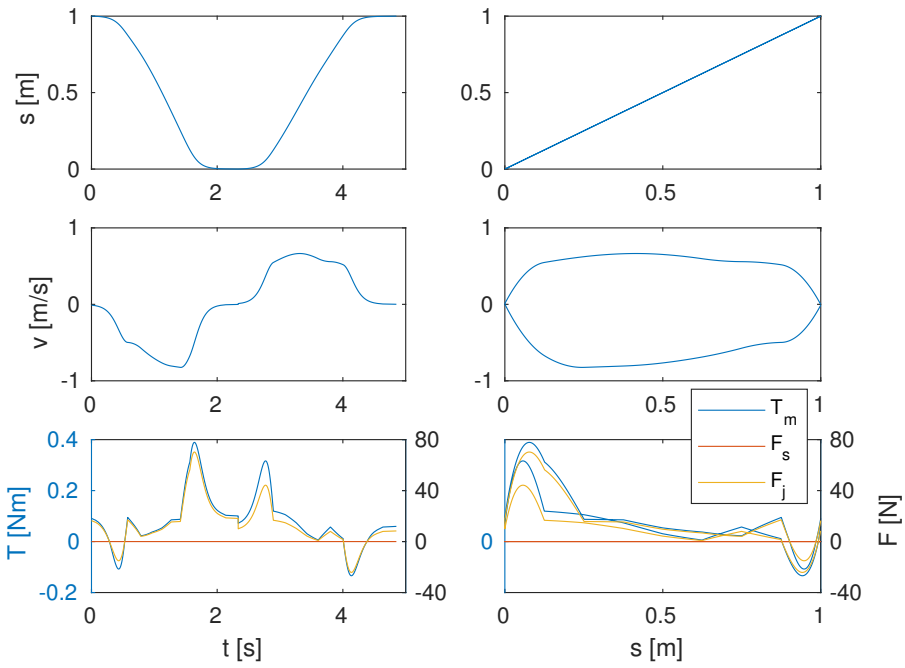


Figure C-2: Optimised trajectory without parallel spring

Without reflected inertia

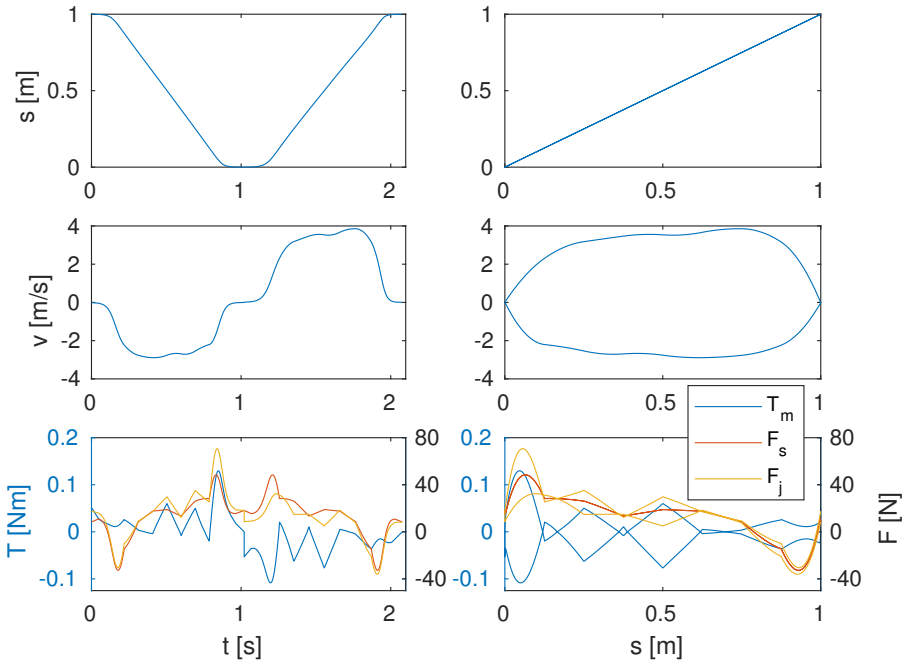


Figure C-3: Smooth optimised trajectory

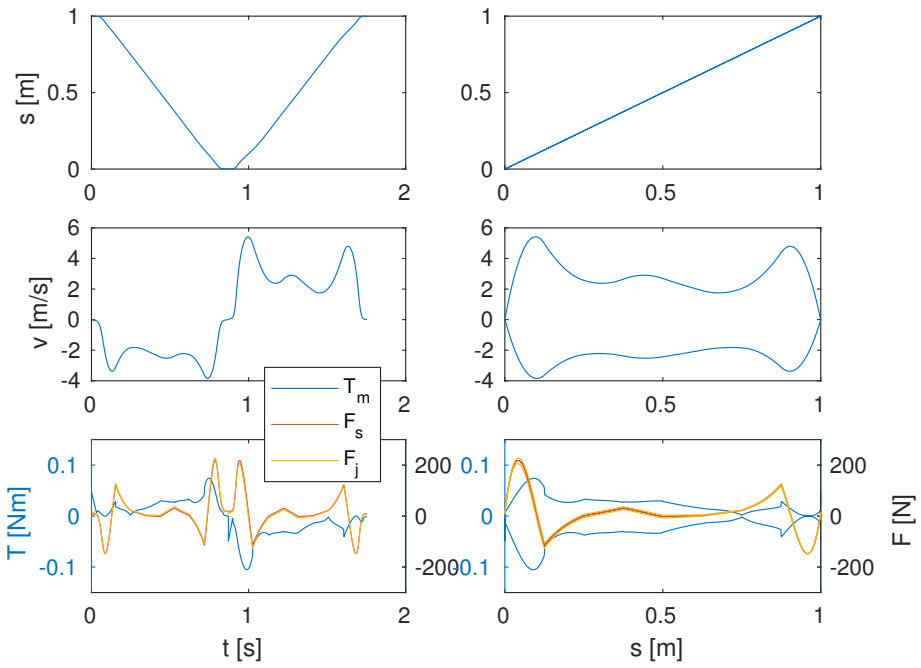


Figure C-4: Bumpy optimised trajectory

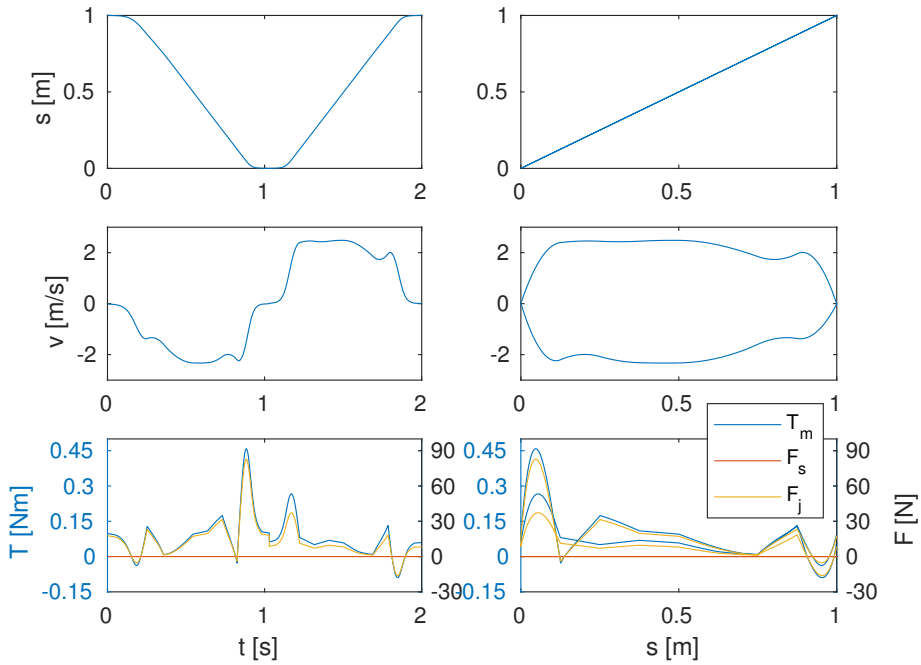


Figure C-5: Optimised trajectory without parallel spring

C-2 Physical mechanism

Extra trajectory results to Table 2-4.

With reflected inertia

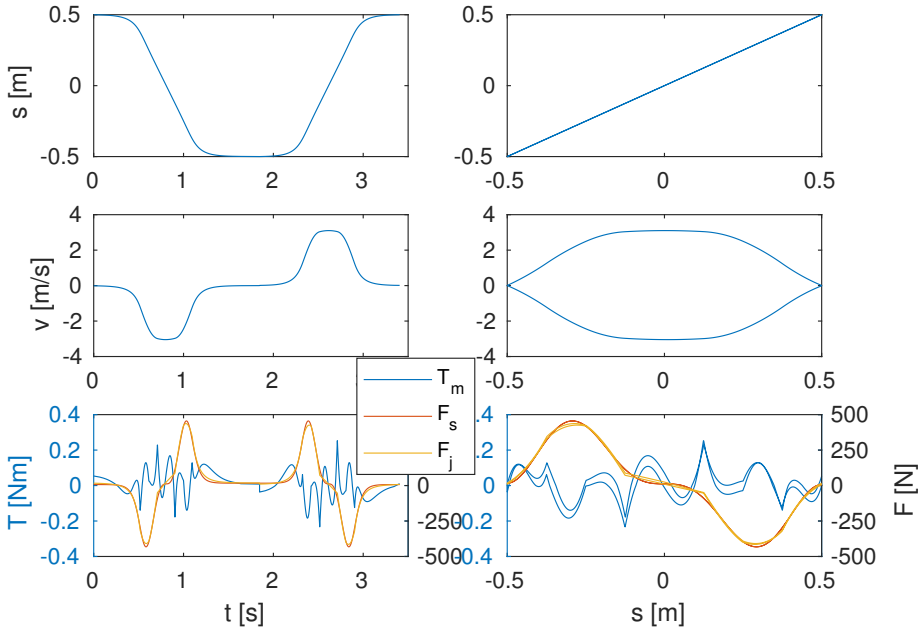


Figure C-6: Optimised for minimum total energy cost, optimal spring stiffness $k_s = 810$

Without reflected inertia

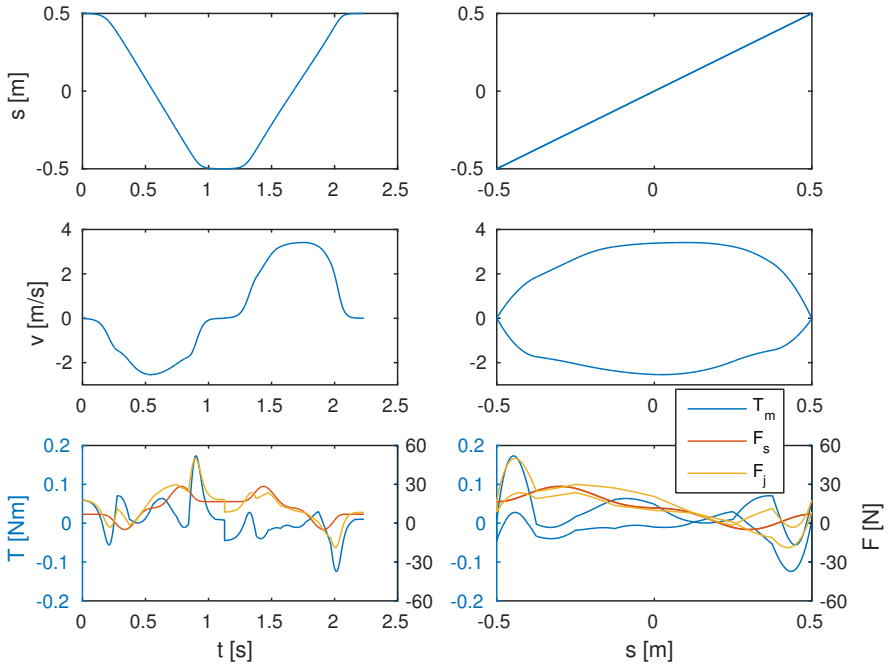


Figure C-7: Optimised trajectory obtained when using the trajectory of Figure C-3 as initial trajectory

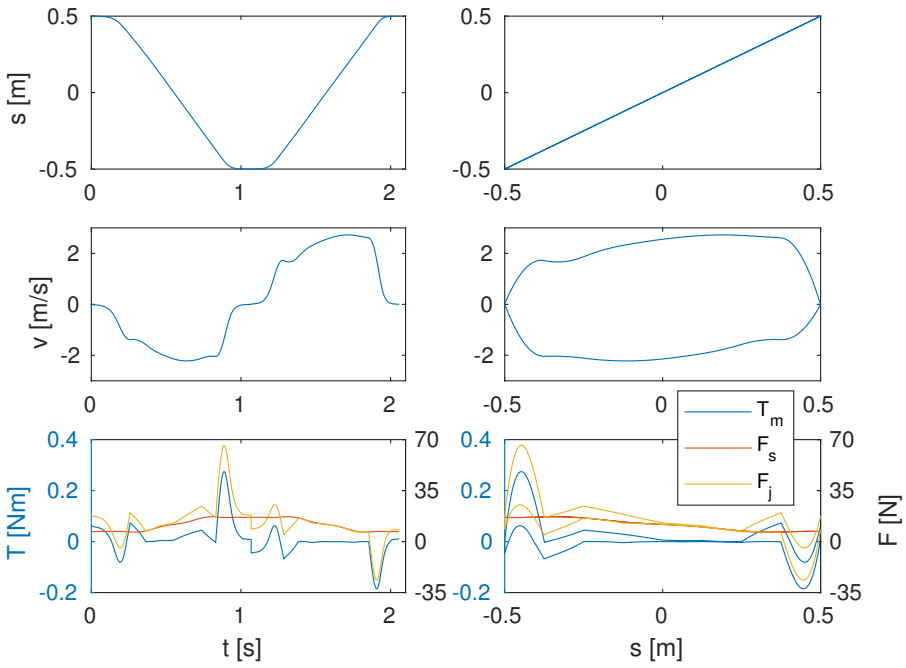


Figure C-8: Optimised for minimum total energy cost, optimal spring stiffness $k_s = 4$

C-3 Horizontal

Extra trajectory results to Table 2-7.

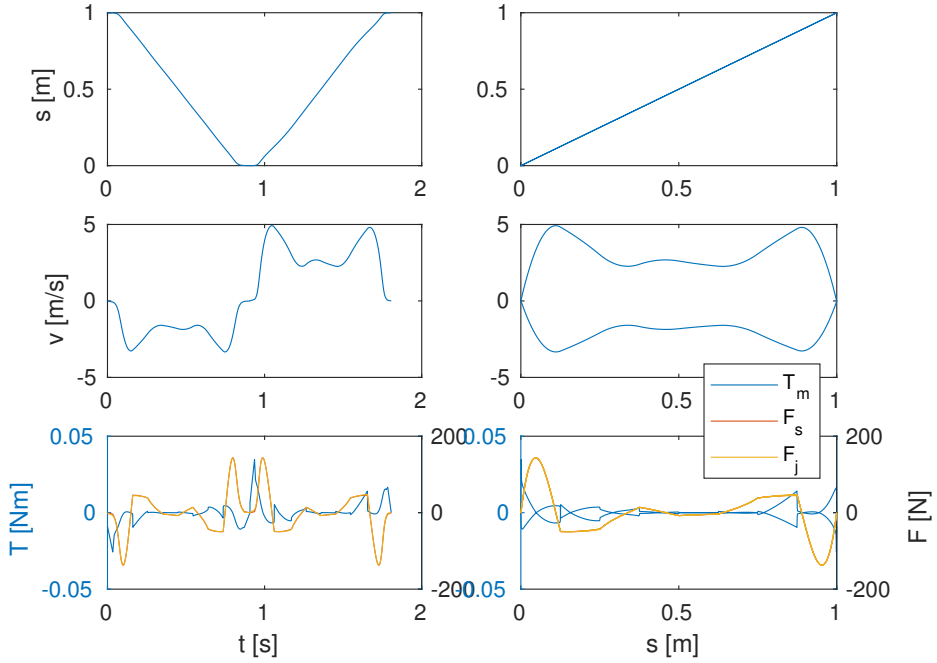


Figure C-9: Bumpy optimised trajectory with analytical spring

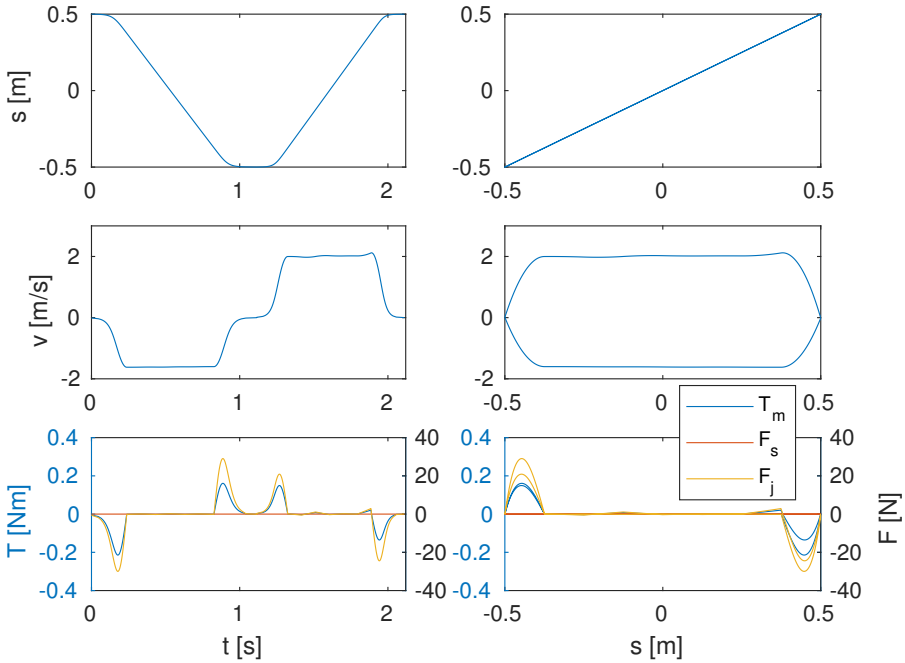


Figure C-10: Optimised trajectory without parallel spring

Appendix D

Calculation of number of instructions for computing next state

Table D-1 summarises the assumed numbers of instructions for the different mathematical operations.

Table D-1: Estimated number of instructions per operation

operation	number of instructions
load value	1
sine/cosine	48
multiply/divide	4
times sign	1
add/subtract	2
square root	25
test sign	1
solve for matrix	10

The following listing decomposes the calculation of the next state in a number of subcalculations, given state $\mathbf{x} = \begin{bmatrix} \mathbf{q}^T & \dot{\mathbf{q}}^T \end{bmatrix}^T$ and input T_m. Per subcalculation the number of operations is counted.

calculation operation	load	sin/cos	mul/div	×sign	+/−	$\sqrt{\ }$	test	matrix
$Cn_{\mathrm{tr}}T_{\mathrm{m}}$	2		4				2	
$T_{\mathrm{s,gravity}}(\mathbf{q})$		2	2		4			
$T_{\mathrm{s,Plooij}}(\mathbf{q})$		8	28		20	2		
$T_{\mathrm{f}}(\dot{\mathbf{q}})$			2	2	2			
$\mathbf{f_q}(\mathbf{q},\dot{\mathbf{q}})$		8	62		22			
$M_{\mathbf{q}}(\mathbf{q})$, $\mathbf{f_q}(\mathbf{q},\dot{\mathbf{q}})$		4	12		13			
$M_{\mathbf{q}}^{-1}\left(\mathbf{f_q}+T_{\mathrm{f}}(\dot{\mathbf{q}})+T_{\mathrm{s}}(\mathbf{q})+Cn_{\mathrm{tr}}T_{\mathrm{m}}\right)$					8			1
$\mathbf{x}_{\mathrm{next}}=\mathbf{x}+\dot{\mathbf{x}}\Delta t$	4		4		4			
subtotals	6	22	114	2	73	2	2	1

Multiplying the numbers of operations with their respective numbers of instructions from Table D-1 results in a total of 1728 instructions.

Bibliography

[1] K. Robotics, "Youbot detailed specifications," February 2015.

[2] W. Wolfslag, M. Plooij, W. Caarls, S. van Weperen, and G. Lopes, "Dissipatively actuated manipulation," *Control Engineering Practice*, vol. 34, pp. 68–76, 2015.

[3] I. Emoteq, "Ur5 technical specifications," tech. rep., Universal Robots, September 2016.

[4] L. van der Spaa, "Reducing energy consumption in robotics," literature survey, 2016.

[5] P. A. Bhounsule, J. Cortell, A. Grewal, B. Hendriksen, J. D. Karssen, C. Paul, and A. Ruina, "Low-bandwidth reflex-based control for lower power walking: 65 km on a single battery charge," *The International Journal of Robotics Research*, vol. 33, no. 10, pp. 1305–1321, 2014.

[6] A. Mazumdar, S. J. Spencer, C. Hobart, J. Salton, M. Quigley, T. Wu, S. Bertrand, J. Pratt, and S. P. Buerger, "Parallel elastic elements improve energy efficiency on the steppr bipedal walking robot," *IEEE/ASME Transactions on Mechatronics*, 2016.

[7] Z. Batts, J. Kim, and K. Yamane, "Design of a hopping mechanism using a voice coil actuator: Linear elastic actuator in parallel (leap)," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 655–660, IEEE, 2016.

[8] R. Vosse, "Plugless robot arm: Energy recuperation using spring mechanisms," Master's thesis, TU Delft, Delft University of Technology, 2016.

[9] N. Schmit and M. Okada, "Optimal design of nonlinear springs in robot mechanism: simultaneous design of trajectory and spring force profiles," *Advanced Robotics*, vol. 27, no. 1, pp. 33–46, 2013.

[10] M. Plooij and M. Wisse, "A novel spring mechanism to reduce energy consumption of robotic arms," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 2901–2908, IEEE, 2012.

[11] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "Lqr-trees: Feedback motion planning via sums-of-squares verification," *The International Journal of Robotics Research*, 2010.

[12] E. Todorov and W. Li, "A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *American Control Conference, 2005. Proceedings of the 2005*, pp. 300–306, IEEE, 2005.

[13] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 4906–4913, IEEE, 2012.

[14] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 1168–1175, IEEE, 2014.

[15] G. Smit, D. H. Plettenburg, and F. C. van der Helm, "The lightweight delft cylinder hand: First multi-articulating hand that meets the basic user requirements," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 23, no. 3, pp. 431–440, 2015.

[16] P. Bhounsule, A. Ruina, *et al.*, "Cornell ranger: energy-optimal control," *Dynamic walking*, 2009.

[17] M. Plooij, *Using a resonant mechanism to reduce energy consumption in robotic arms.* PhD thesis, TU Delft, Delft University of Technology, 2011.

[18] V. Babitsky and A. Shipilov, *Resonant Robotic Systems.* Springer-Verlag Berlin Heidelberg New York, 2003.

[19] SKF, *Presicion rolled ball screws*, August 2013.

[20] R. Q. van der Linde and A. L. Schwab, "Multibody dynamics b," 2011.

[21] L. Toshiba Machine CO., "Scara robot specifications," tech. rep., Toshiba Machine CO., LTD., September 2015.

[22] K. Robotics, "Kr 5 scara," tech. rep., KUKA AG, October 2009.

[23] K. Robotics, "Robots for low payloads," tech. rep., KUKA AG, January 2016.

[24] A. H. Committee, *Properties of Wrought Aluminum and Aluminum Alloys*, vol. ASM Handbook Volume 2: Properties and Selection: Nonferrous Alloys and Special-Purpose Materials. ASM International, 1990.

[25] M. Plooij, M. van Nunspeet, M. Wisse, and H. Vallery, "Design and evaluation of the bi-directional clutched parallel elastic actuator (bic-pea)," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 1002–1009, IEEE, 2015.

[26] M. Plooij, T. van der Hoeven, G. Dunning, and M. Wisse, "Statically balanced brakes," *Precision Engineering*, vol. 43, pp. 468–478, 2016.

[27] E. Todorov and Y. Tassa, "Iterative local dynamic programming," in *Adaptive Dynamic Programming and Reinforcement Learning, 2009. ADPRL'09. IEEE Symposium on*, pp. 90–95, IEEE, 2009.

[28] Y. Tassa and E. Todorov, "High-order local dynamic programming," in *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2011 IEEE Symposium on*, pp. 70–75, IEEE, 2011.

[29] J. Neven, "Design and evaluation of an energy-saving drive for a versatile robot gripper," Master's thesis, TU Delft, Delft University of Technology, 2017.

[30] K. Piromsopa, C. Aporntewan, and P. Chongsatitvatana, "An fpga implementation of a fixed-point square root operation," in *Proceedings of the International Symposium on Communications and Information Technology*, vol. 14, p. 16, 2001.

[31] K. Sharat, B. Uma, and D. Sagar, "Calculation of sine and cosine of an angle using the cordic algorithm," *IJITR*, vol. 2, no. 2, pp. 891–895, 2014.

[32] T. Heister and L. G. Rebholz, *Scientific Computing: For Scientists and Engineers*. Walter de Gruyter GmbH & Co KG, 2015.

[33] J. F. Epperson, *An introduction to numerical methods and analysis*. John Wiley & Sons, 2013.

[34] N. Semiconductors, "Rlpc2921/2923/2925: Arm9 microcontroller with can, lin, and usb," tech. rep., NXP B.V., April 2010.

[35] P. Semiconductors, "Lpc2141/42/44/46/48: Single-chip 16-bit/32-bit microcontrollers; up to 512 kb flash with isp/iap, usb 2.0 full-speed device, 10-bit adc and dac," tech. rep., Koninklijke Philips Electronics N.V., September 2005.

[36] T. Instruments, "Tms320vc5510 fixed-point digital signal processor."

[37] Arduino, "Arduino nano."

[38] T. Lextrait, "Arduino: Power consumption compared," May 2016.

[39] G. Lantoine and R. P. Russell, "A hybrid differential dynamic programming algorithm for constrained optimal control problems. part 1: Theory," *Journal of Optimization Theory and Applications*, vol. 154, no. 2, pp. 382–417, 2012.

[40] K. Ohno, "A new approach to differential dynamic programming for discrete time systems," *IEEE Transactions on Automatic Control*, vol. 23, no. 1, pp. 37–47, 1978.

[41] M. Motor, "Ec 90 flat ∅90 mm, brushless, 90 watt, with hall sensors."

[42] SKF, *Roller screws*, April 2014.

[43] M. Motor, "Planetary gearhead gpx 32 hp ∅32 mm, 2-stage."

Glossary

List of Acronyms

3mE	Mechanical, Maritime and Materials Engineering
TU Delft	Delft University of Technology
DDP	Differential Dynamic Programming
DoF	degree of freedom
iLDP	iterative Local Dynamic Programming
iLQG	iterative Linear Quadratic Gaussian control
LQG	Linear Quadratic Gaussian control
LQR	Linear Quadratic Regulator
RK4	Runge-Kutta 4^{th} method

List of Symbols

ϵ	A small positive number
η	Efficiency
μ_{C}	Coulomb friction
μ_{v}	Viscous friction
ξ	Normal distributed noise mode with distribution N$(0; 1)$ (Chapter 4)
ϕ	Angular position
ω	Angular velocity
A	State matrix of the linearised discrete system dynamics

B	Input matrix of the linearised discrete system dynamics
C	Transmission efficiency term with motor/generator switching mode
c_f	Final state cost coefficient
$\mathcal{C}$	Matrix scaling the standard deviations of the noise modes
d_0	Nominal (spindle) diameter (in mm)
E	Expectancy of the term which follows in square brackets
E	Energy
$\mathbf{f_q}$	Reduced force vector
$\mathbf{f}$	Discrete function of the nonlinear state dynamics
$\mathcal{F}$	Functional
F	Force
g	Gravitational acceleration (9.81 m/s^2)
I	Electric current
I	Identity matrix
i	Index
$\mathcal{J}$	Cost function
J	Inertia
k	Arbitrary integer
k_t	Motor torque constant
$\mathbf{k}$	Optimal feedforward input correction
k_s	Spring stiffness
K	Optimal state feedback gain
l	Cost (Chapter 4)
l	Link length (Chapter 3)
m	Mass
M	Mass matrix
m	Number of states (Chapter 4)
N	Normal distribution N($\mu; \sigma^2$), with mean μ and standard deviation σ
N	Total number of discrete steps of the trajectory
n	Number of states (Chapter 4)
n	Reduction ratio
P_h	Spindle lead (in mm)
P	Power
$\mathbf{q}$	State vector in configuration space coordinates
R	Electric resistance
r	Radius
$\mathbf{s}_k$	Linear cost term
S_k	Quadratic cost term
s_k	Scalar cost term
$\mathcal{T}$	Jacobian of the kinematic map

T	Torque
t	Time
$\mathbf{U}$	Full set of inputs to a trajectory
$\mathbf{u}$	Control input
V	Optimal cost to go as function of the state
v	Optimal cost to go as function of the state deviation
$\mathbf{x}$	State vector in workspace coordinates
$\mathbf{X}$	Full set of states of a trajectory
$\mathbf{x}$	Optimisation parameter vector (Chapter 2)
y	Vertical position

Subscripts

a	Actuation
d	Down
e	Electric
f	Friction
g	Gravity
j	Joint
m	Motor
oh	Overheads
rpm	Angular velocity in rotation per minute
s	Spring Spindle, in case of reduction ratio n_s Static, in case of friction coefficient μ_s
tr	Transmission
u	Up
$\mathbf{q}$	Generalized coordinates
$\mathbf{x}$	Workspace coordinates
e	End
f	Final