
# REAL-TIME PATH-CONSTRAINED TRAJECTORY PLANNING FOR ROBOT MANIPULATORS WITH ENERGY BUDGET OPTIMIZATION

Danilo Vannier Cunha

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Fernando Cesar Lizarralde

Rio de Janeiro
Março de 2020

REAL-TIME PATH-CONSTRAINED TRAJECTORY PLANNING FOR ROBOT MANIPULATORS WITH ENERGY BUDGET OPTIMIZATION

Danilo Vannier Cunha

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Orientador: Fernando Cesar Lizarralde

Aprovada por: Prof. Fernando Cesar Lizarralde
Prof. Fernando Alves Rochinha
Prof. Amit Bhaya

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2020

Cunha, Danilo Vannier

Real-Time Path-Constrained Trajectory Planning for Robot Manipulators with Energy Budget Optimization/Danilo Vannier Cunha. – Rio de Janeiro: UFRJ/COPPE, 2020.

XII, 90 p.: il.; 29, 7cm.

Orientador: Fernando Cesar Lizarralde

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2020.

Referências Bibliográficas: p. 79 – 82.

1. Trajectory Optimization. 2. Manipulator Dynamics. 3. Nonlinear Control. I. Lizarralde, Fernando Cesar. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

# Acknowledgements

I wish to thank my wife Nathalie and my daughter Melissa for the warm support during this epic journey. Without them it would have been much harder. I would also like to acknowledge my parents for the investments in my education.
Furthermore, I would like to express my gratitude to my advisor Fernando Lizarralde. His guidance and insights were fundamental in the development of this work and in my intelectual development.
Moreover, I would like to recognize the invaluable support from my friend Esteban from TFMC. Without his comprehension it would have been impossible to find the time required to complete this research.
Last but not least, I am indebted to the Brazilian people for funding the public university.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

# OTIMIZAÇÃO DE TRAJETÓRIAS DE MANIPULADORES ROBÓTICOS COM CAMINHO FIXO E LIMITE DE ENERGIA

Danilo Vannier Cunha

Março/2020

Orientador: Fernando Cesar Lizarralde

Programa: Engenharia Elétrica

Otimização de trajetórias é um tema de grande relevância na comunidade de robótica. Várias aplicações industriais têm limitação de energia, como equipamentos submarinos e veículos elétricos. Essa dissertação considera o rastreamento de trajetórias de caminho fixo de manipuladores robóticos onde as trajetórias são otimizadas para utilizar-se um limite pré-definido de energia.

A estratégia proposta é baseada na técnica de horizonte móvel do controle preditivo utilizando-se de uma parametrização do caminho que reduz a dimensão para uma variável. A dinâmica da trajetória parametrizada é governada por um sistema linear que pode ser considerado como um parâmetro de controle. Em seguida, uma função de energia baseada no conceito de trabalho e uma função de custo são definidas. Com isso, o controle é feito associando-se controle preditivo com o método de Newton minimizando-se então a função de custo em tempo real. Além disso, a técnica de torque computado é empregada para linearizar o sistema.

A solução proposta é também formulada no espaço de trabalho do manipulador permitindo-se o uso de controle cinemático, o que é fundamental considerando-se que vários manipuladores industriais permitem apenas controle de velocidade.

A solução proposta nessa dissertação foi verificada por meio de simulações numéricas. Resultados experimentais com um manipulador de quatro graus de liberdade ilustram a aplicabilidade da solução proposta.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

REAL-TIME PATH-CONSTRAINED TRAJECTORY PLANNING FOR ROBOT MANIPULATORS WITH ENERGY BUDGET OPTIMIZATION

Danilo Vannier Cunha

March/2020

Advisor: Fernando Cesar Lizarralde

Department: Electrical Engineering

Trajectory optimization is of great relevance for the robotics community. Many industry applications have energy limitations, such as subsea installed equipment or electric vehicles. This dissertation considers the optimization of path-constrained trajectory for robot manipulators with limited energy budget.

The proposed strategy is based on a Nonlinear Receding Horizon Predictive Control (NRHPC) using a path parameterization of dimension one. The dynamic of the parameterized trajectory is governed by a predefined linear system, then an energy and a cost functions are defined and a NRHPC based on a Newton method is used to minimize the cost function in real time. A computed torque linearization scheme is considered to simplify the Newton method.

The solution is also formulated in the manipulator task space permitting to be used with a kinematic control scheme, which is fundamental since many industrial manipulators only allow velocity control.

The proposed solution is verified with several numerical simulations. Experimental results using a four degrees of freedom manipulator illustrate the applicability of the proposed solution.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Over the last decades the field of robotics has been evolving substantially. More and more segments of the industry are using robots to automate tasks. In some segments, such as the subsea oil and gas industry, robots are being used to reach where human operators can not. Remotely Operated Underwater Vehicles (ROVs) have been used for inspection, military operations, rescue missions, support for oil and gas offshore activities and others.

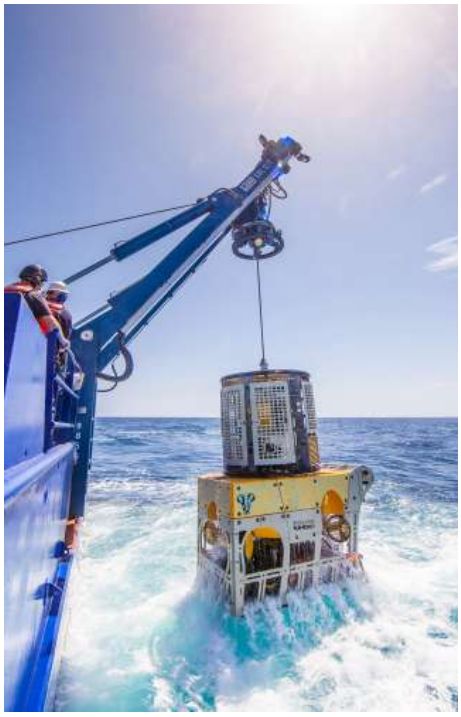

Figure 1.1: ROV being deployed - source: www.technipfmc.com

In the subsea oil and gas industry, robotics is transforming the way manifolds and christmas trees are operated. These equipments are installed on the seabed at depths up to 3000 meters. Christmas tree is the equipment installed on top of the oil (or gas) well to control and secure the production and manifolds are larger

equipment that combines the flow from multiple trees in one line to the topside platform. Traditionally, the valves on the manifolds and trees are either operated by ROVs (manual valves) or by hydraulic actuators. Hydraulic technology has some drawbacks such as size and weight of the actuators and the cost of the umbilical lines. The umbilical is a supply line that carries hydraulic pressure and electric cables. Besides that, there is a high cost of hydraulic fluid and maintenance of topside equipment to pressurize the hydraulic line (Moe et al. [28]).

Electric equipment connected to a long umbilical line has limited amount of power. The rationale for that is the ohmic losses in the electric cables and the fact that the transmission voltage is limited by the connectors and equipment involved. The fact that the power is limited implies that many of the subsea applications requires the use of local batteries to store the energy from the umbilical. Furthermore, due to limited installation capabilities, the size or the capacity of these batteries is usually constrained by the total weight allowed for the subsea equipment. Hence, there is limited amount of energy to operate subsea installed equipment.

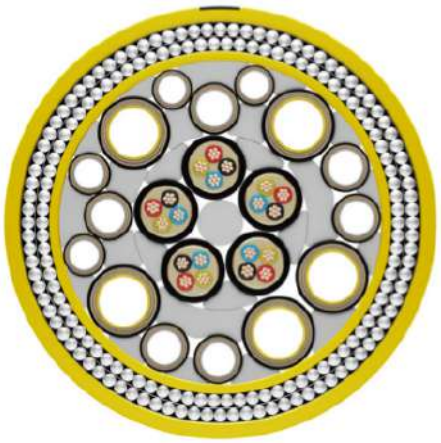

Figure 1.2: TechnipFMC Umbilical Cross-Section - source: www.technipfmc.com

Over the last two decades an alternative to hydraulic actuators had been gradually introduced into the subsea oil and gas industry. Electric actuators with local batteries began to replace the hydraulic actuators (Johansen et al. [19],Sten-Halvorsen & Koren [45],Aadland & Petersen [1],Hasan et al. [17],Moe et al. [28]). Consequently, reducing the size of the umbilical up to 30% (Moe et al. [28]). Besides, the electric actuator has much more position accuracy than the hydraulic actuator (Johansen et al. [19]). The advantage of the hydraulic system is that in case a shortage on hydraulic power occurs, there is a spring that pushes the valve to a safe position (fail safe functionality) (Moe et al. [28]). The electric solution requires a local energy storage solution (usually a Li-Ion battery) to accomplish the fail safe functionality (Sten-Halvorsen & Koren [45],Aadland & Petersen [1],Hasan et al. [17],Moe et al. [28]).

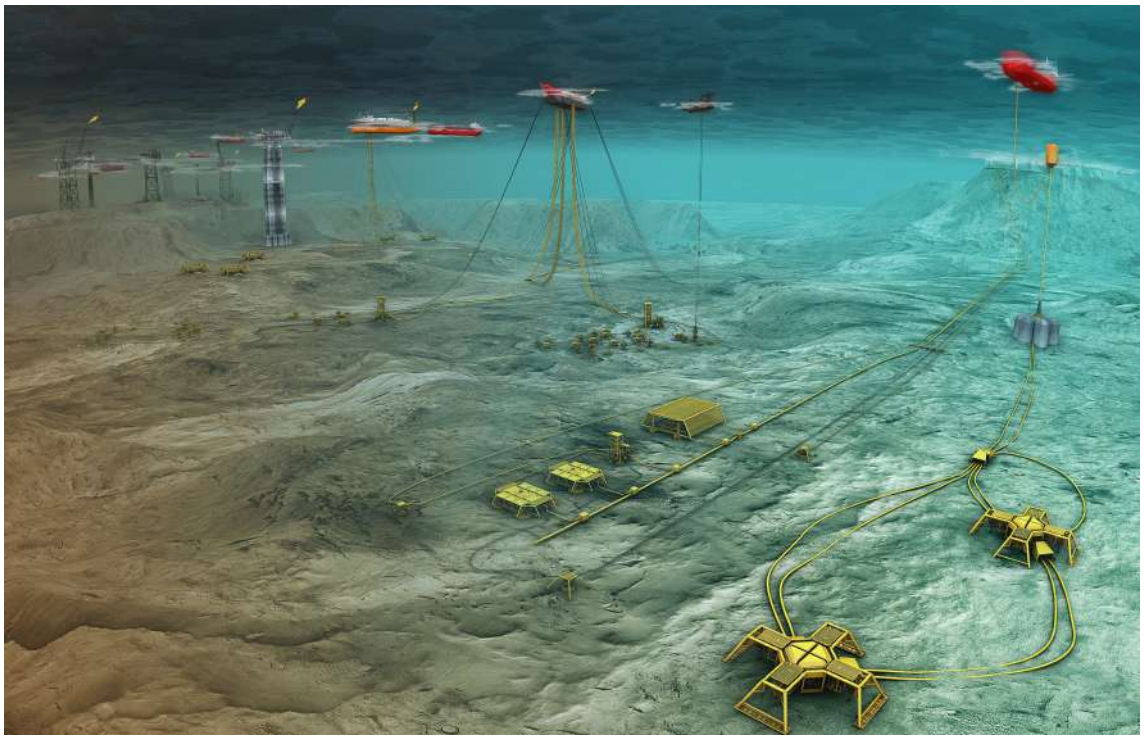

Figure 1.3: Subsea Production Field Layout - source: www.technipfmc.com

Figure 1.4: TechnipFMC electric actuator with battery - source: www.technipfmc.com.

Over the last years the subsea oil and gas industry evolved to shared actuation systems where one actuator is shared between all the valves on a manifold. In (Mair [27]) and (Moreira et al. [29]) the first system to present this type of solution is described. More recently, in patent (Azevedo et al. [4]) a robotic manipulator is used to operate subsea valves on production equipment. Later, in patent (Schilling & Cohan [34]) is proposed a vehicle with an actuator that moves on top of a manifold to operate valves. This vehicle would eventually dock to a charging station to recharge

the batteries with inductive coupling technology.

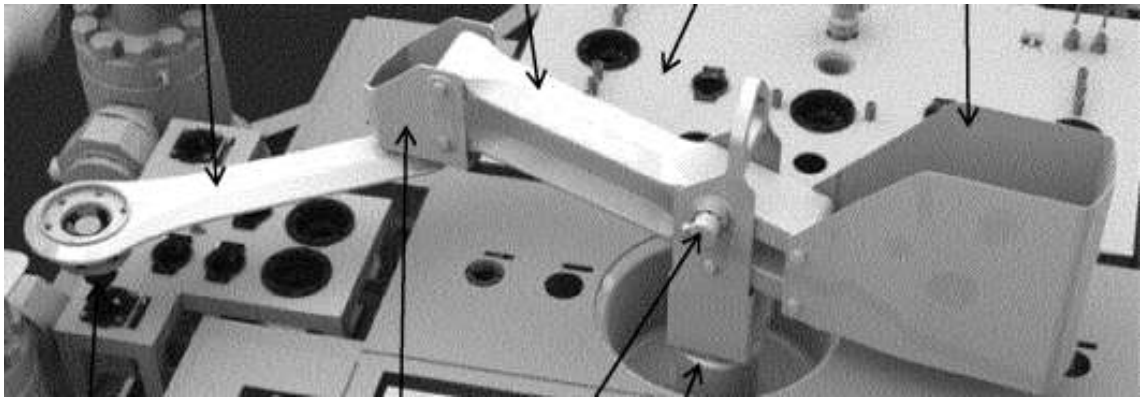

Figure 1.5: Shared actuation system - source: (Azevedo et al. [4]).

## 1.1 Motivation for Subsea Oil & Gas Application

In the context of electric actuators with local batteries used on the subsea oil and gas industry, energy optimization is a matter of great relevance. Specially on shared actuated systems, where the same battery could be shared.

Subsea installed equipments have size and weight restriction due to installation constraints. Consequently, the battery bank capacity is limited.

In case there is an emergency in the topside platform, the subsea manifold valves need to be actuated to their safe positions. This operation is called shutdown sequence. The shutdown sequence needs to be executed in a limited time. In the hydraulic actuated valves there is a spring mechanism in each valve that push the valve stem to the safe position. In the electric actuated valves the actuators move the valves to its safe position using energy from the batteries.

This scenario poses a challenge to plan the task, e.g. the shutdown sequence, based on the available energy on the batteries. It is desirable to execute the task as fast as possible but the available energy needs to be sufficient to finish the entire task.

In summary, a method to replan a task to use the entire energy storage of a system while minimizing the task completion time is crucial for the subsea oil and gas production industry.

## 1.2 Other Applications

There are other industry sectors that have similar energy problems and could share the same solution. For example, challenging applications where there is limited

amount of energy supply such as satellites or autonomous electric vehicles where the battery bank is size and weight limited.

Satellites and spacecraft are powered by solar panels, which leads to a limited energy budget (Patel [30]), where the amount of energy is constrained by the area of the solar panels, which is restricted to the size of the satellite.

Autonomous electric vehicles are too a similar application where on one hand the stored energy needs to last before the end of the task and on the other hand the task should be accomplished as fast as possible. There is always the constraint on the battery bank size driven by weight and size. So it is beneficial to make the most of the available energy. Examples of autonomous electric vehicles on the industry include underwater autonomous vehicles (AUVs) which are mainly used for surveillance, self driven autonomous electric cars, autonomous vacuum cleaners to name a few.

## 1.3 Review of existing literature

Several solutions are proposed to the time-optimal problem over the years (Bobrow et al. [5], Shiller [35], Shin & McKay [37]). The energy-optimal problem was also well studied over the years (Kim & Shin [21], Shiller [36], Field & Stepanenko [13] and Verscheure et al. [46]).

In robotics applications, motion planning is considered as a nonlinear multi-dimensional problem, where often, the task is splited in two: path planning and trajectory planning (Verscheure et al. [46]). The path planning stage deals with geometric matters such as obstacle avoidance and the trajectory planning stage deals with dynamic of the motion and actuators' limits for example.

The time-optimal solution presented by (Bobrow et al. [5]) and (Shin & McKay [37]) and later improved by (Slotine & Yang [41]) uses an arc-length parameterization, e.g. $s(0) = 0 \leq s(t) \leq 1 = s(T)$ where $t$ is the time with $t \in [0, T]$ and $T$ is the time duration. And then building a position vs velocity, i.e., $s(t) \times \dot{s}(t)$, parameterized phase plane with the boundaries of admissible region. Then, an algorithm is used to find the acceleration (or deceleration) switching points through the contour of the admissible region that leads to a trajectory with higher velocities.

In (Shin & McKay [37]), the torque limit velocity dependence is considered at the worst scenario quadratic, and this assumption is the base for the proof that the method converges in finite steps. In (Bobrow et al. [5]) a more general approach is used where torque limit is any arbitrary function of position and velocity. However, (Bobrow et al. [5]) only considers simply connected admissible region on the phase plane. In Shin & McKay [37] the general case is admitted.

Reference (Kim & Shin [21]) deals with the trajectory optimization considering

a time and fuel criteria. The following cost function is used:

$$J = \int_{t_0}^{t_f} \lambda + \sum_{i=1}^{n} |\tau_i| dt \tag{1.1}$$

where $t_0$ and $t_f$ are the initial and final times, $\tau$ is the generalized force/torque vector and $\lambda$ is the trade-off factor between time and fuel. The optimization is carried out with phase plane switching curves as in (Bobrow et al. [5]). Then, the authors propose a method called average dynamics where nonlinear parameters are continuously updated by their approximated averages. This simplification allows their method to be implemented in real-time. The proposed method in (Kim & Shin [21]) is verified only by numerical simulation.

Later in (Shin & McKay [38]) dynamic programming is used to find optimal phase plane trajectory. The goal is to extend the optimization to a more general case where the constraint on the actuator torque limits are dependent to each other.

In (Pfeiffer & Johanni [31]) a path coordinate is also used. The time-optimal solution is achieved via geometric properties of the manipulator dynamics parameterized in path parameter. A constrained phase-space is used to determine the time optimal trajectories on the boundaries of the constrains. And then a dynamic programming algorithm is used to optimize the trajectory according to cost function combining time, squared velocity (which is proportional to kinetic energy) and joint torques.

In (Field & Stepanenko [13]) the energy of a manipulator is minimized by means of an iterative dynamic programming algorithm in phase space (joint velocity vs joint angle). However, instead of using the boundaries, polynomials (cubic B-splines) are used to approximate angle and velocities vertices on grids. Several grids are created for each joint trajectory during the iterations. The grids start coarse and get finer along the iterations. The proposed algorithm in (Field & Stepanenko [13]) scales linearly with the number of joints.

In (Shiller [36]) also uses a trajectory parameterization to reduce the order of the optimization problem. The time-energy optimization problem with constrains is carried out using Hamiltonian and solving the co-state equation. In contrast to typical time optimal control (Bobrow et al. [5] and Shin & McKay [37]), the proposed time-energy optimal control is smooth, which presents implementation advantages avoiding jerks on the motors and requiring a simpler controller.

In (Verscheure et al. [46]) and (Verscheure et al. [47]) it is proposed a nonlinear change of variables to transform the time-energy optimization problem in a convex

problem. Furthermore, the cost function for the time optimal problem typically is

$$C = \int_0^T 1dt = \int_0^1 \frac{1}{\dot{s}} ds \tag{1.2}$$

A nonlinear change of variable is proposed

$$\begin{aligned} a(s) &= \ddot{s} \\ b(s) &= \dot{s}^2 \end{aligned} \tag{1.3}$$

with the following constraint

$$b'(s) = 2a(s) \tag{1.4}$$

which follows from $\dot{b}(s) = b'(s)\dot{s}$ and also $\dot{b}(s) = \frac{d(\dot{s}^2)}{dt} = 2\ddot{s}\dot{s} = 2a(s)\dot{s}$, where the dot indicates time derivative and the apostrophe partial derivative. The cost function is then reformulated

$$C = \int_0^1 \frac{1}{\sqrt{b(s)}} \, ds \tag{1.5}$$

with that change of variable the objective function is convex (Verscheure et al. [46], Verscheure et al. [47]), and as long as the constraints are linear the optimization problem is also convex (Verscheure et al. [46], Verscheure et al. [47]), which permits the use of second order cone programming solver (SOCP) to execute the optimization in an efficient manner. The drawback of this approach is that only a specific types of cost functions (and constraints) are allowed. Another example of cost function that is also convex is the thermal energy for joint $i$ (Verscheure et al. [46], Verscheure et al. [47])

$$C = \int_0^T \tau_i(t)^2 dt = \int_0^1 \frac{\tau_i(s)^2}{\dot{s}} ds = \int_0^1 \frac{\tau_i(s)^2}{\sqrt{b(s)}} ds \tag{1.6}$$

A similar method is proposed more recently in (Reynoso-Mora et al. [32]) to solve the time-optimal solution, where the full dynamic model of a manipulator including friction is considered. Second order cone programming is also used, and to formulate as a convex optimization problem, the velocity constraint is re-formulated. Also in (Reynoso-Mora et al. [32]), the authors propose a penalization on the total jerks on the actuators. Later, (Debrouwere et al. [11]) utilizes sequential convex programming (SCP) and considers non-convex constraints by decomposing them as a difference of two convex functions and linearizing the concave part of it. Examples of non-convex constraints are (Debrouwere et al. [11]) torque bounds which depends on speed, torque rate constraints and cutting forces at the end effector (milling applications).

In (Xu et al. [50]) a global optimization method considering minimum time and minimum energy criteria is proposed. The authors propose an algorithm entitled

Environment-Gene evolutionary Immune Clonal Algorithm (EGICA). The path is given and the trajectory is planned based on minimization of a cost function with weighted time and energy. The trajectories considered in (Xu et al. [50]) are restricted by cubic polynomials in the joint space. The method is then verified in simulation in the Stanford manipulator.

In (Lin et al. [24]) a method based on sequential quadratic programming (SQP) is presented to re-plan a path constrained trajectory in order to maximize the service life of a robot. This is achieved by reducing torque levels on the weakest joint. The service life of a given joint of a robotic manipulator in (Lin et al. [24]) is given by

$$L = \frac{\lambda K}{\sum_{k=1}^{K} |\dot{q}_k||\tau_k|^c} \tag{1.7}$$

where the system is discretized in $K$ intervals, $|\dot{q}_k|$ and $k$, $|\tau_k|$ are the absolute values of joint velocity and joint torque at instant $k$ respectively, $\lambda$ and $c$ are constants. The optimization problem is then defined to find the set of positions, velocities and accelerations that maximizes the service life of the weakest joint, i.e. the joint with smallest service life. The physical joint limits are considered as constraints. Since the optimization problem is not convex, the authors in (Lin et al. [24]) choose SQP, where the idea is to linearize the constrains around an operation point and locally solve an approximate convex program (Lin et al. [24]). Also, (Lin et al. [24]) proposes to invert the cost function in order to reformulate the problem as a minimum problem instead of maximum problem. This reformulation leads to a more efficient way to calculate the Hessian matrix.

In (Cunha & Lizarralde [10]) an iteration method associated with predictive control is used to minimize the difference between available energy budget and the expected trajectory energy. The proposed optimization method requires low computational cost which allows it to be used online and it is verified experimentally.

Table 1.1 summarizes the references indicating the optimization goal and the method used. Note that it is a novel optimization goal and the proposed strategy applied to robot trajectory optimization is a novel contribution.

Table 1.1: Literature Review Summary.

| | | [5, 35, 37, 41] | [21] | [38] | [31] | [36] | [13] | [46, 47] | [50] | [32] | [11] | [24] | Dissertation, [10] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Goal | Min. Time | X | | X | X | | | X | | X | X | | |
| | Minimum Energy | | | | | | X | | | | | | |
| | Minimum Time-Energy | | X | X | | X | | X | X | | | | |
| | Maximum Service Life | | | | | | | | | | | X | |
| | Energy Budget | | | | | | | | | | | | X |
| Strategy | Switch. Points on Phase Plane | X | X | X | X | | X | | | | | | |
| | Dynamic Programming | | | X | X | | X | | | | | | |
| | Hamiltonian and Co-State Eq. | | | | | X | | | | | | | |
| | SOCP | | | | | | | X | | X | | | |
| | EGICA | | | | | | | | X | | | | |
| | SCP | | | | | | | | | | X | | |
| | SQP | | | | | | | | | | | X | |
| | Pred. Ctr. & Newton Method | | | | | | | | | | | | X |

Although the problem is similar to the classical time-energy optimization, the goal here is to plan based on the available energy budget. The solution proposed in this dissertation is based on (Cunha & Lizarralde [10]), and also relies on a path parameterization to reduce the dimension of the system. The optimization algorithm is based on a receding horizon predictive control associated with a Newton method to find the zero of a given cost function (similar approach as done in (Cunha & Lizarralde [10]) and (Lizarralde et al. [26]) for non-holonomic systems). The proposed approach gives highly effective results with relatively low computational cost.

A similar solution is presented in (Fernandes et al. [12]) to find near-optimal non-holonomic motion planning. An optimal control approach is used to minimize a cost function that depends on the control effort. A solution is proposed considering the input represented in an infinite orthogonal base (e.g. Fourier serie). Furthermore, the input and the cost function is represented by the coefficients of the base. Hence, the coefficients become the optimization variables of a non-linear optimization problem. Ritz approximation theory is used to find the optimal solution by solutions of finite dimension problems. The solutions of the finite dimension problems are near-optimum solutions. But these near-optimum solutions converge to the optimum solution as the dimension of the problem grows. Finally, to solve the finite dimension problem the authors propose a Newton method iteration approach. Similar solution is used in (Lizarralde et al. [26]) and (Lizarralde [25]) for non-holonomic systems and (Sontag [43]) for systems without drift.

The solution used here fits the context of continuation methods, where the trajectory corresponding to an initial condition of the control variable is deformed (or iterative corrected) into the final trajectory that minimizes the error. Continuation methods are used to solve nonlinear problems by deforming the problem in a family of similar problems by changing a given parameter. For one value of this parameter the resulting problem has a known solution and for another value its the original problem. The parameter is then corrected by a iteration process starting on the known solution and ending on the original problem. As presented in (Richter & Decarlo [33]), for a given problem $F(x) = 0$, a homotopy (or deformation) is defined where $F_t(x) = 0$ with $t \in [0, 1]$ and $F_1(x) = 0$ is the original problem and $F_0(x) = 0$ a trivial solution. Then the continuation method comprises in iteratively calculating the solutions for each $F_t(x) = 0$ starting with $t = 0$ and ending in $t = 1$. In (Kontny & Stursberg [22]) and (Kontny & Stursberg [23]), continuation method is proposed to perform online obstacle avoidance. Trajectories homotopic to the optimal unconstrained solution are pre-calculated offline. Then in case of an obstacle in the current trajectory, the system is driven online to a homotopic trajectory free of collision.

In (Chaves & Lizarralde [6]) utilizes a combination of receding horizon with an

iterative Newton method to control autonomous mobile robots. An error function is defined being the difference between current and desired states. A non-singular discrete control vector is predicted for a short time window. The control signal is then refined by the Newton method. At each iteration the first element of the control vector is applied to the system and the vector is then shifted.

## 1.4 Objectives

The goal of this dissertation is to establish a method to optimize trajectories of robotic systems with a limited energy budget. This is a novel trajectory optimization objective for robot systems. The idea is to perform the trajectory as fast as possible with the available energy. Consequently, at the end of the trajectory there should be no energy left.

Typically, the systems that can benefit from the proposed method, such as subsea robots, autonomous vehicles and satellites, has low powered processing units. Thus, the algorithm must be fast enough to run in such embedded systems.

It is also an objective to validate numerically and experimentally the proposed method.

## 1.5 Methodology

In order to solve the energy budget optimization proposed in this dissertation, the path is parameterized and the dynamics of the parameterized path is controlled by a linear system in a similar approach as done in (Lin et al. [24]).

Then, a cost function comprising the difference of the available energy budget and the open loop prediction of the energy required for the trajectory is defined. Later, a linearized map is established between the linear system input and the error. This map is used to correct the linear system input towards minimizing the error on a Newton method iteration scheme.

Later, the linear map calculation is simplified by linearization of the system by using a model based control to cancel the manipulator non-linearities.

In real systems, the measurement of available energy is a rather complicated matter. In order to accommodate for imperfections in the state-of-charge estimation and for any unpredictable consumption on a system level, a receding horizon predictive control scheme associated to the Newton iteration similar to (Chaves & Lizarralde [6]) is used. The predictive control is formulated considering the discretized system. It could be argued that a discretized system could be considered all along but the robot manipulator is best described in continuous time.

The effectiveness of the proposed method is verified numerically using a Matlab simulation using a planar manipulator with two rotatory joints and experimentally on Tetis manipulator. Tetis is a light weight manipulator part of the rail guided vehicle Doris (Xaud [49]). Tetis has 4 rotatory joints and weights around $2.5Kg$. The experiment is done with Matlab running the proposed algorithm on a computer and sending velocity set points to Tetis using the ROS (Robot Operating System) framework. ROS is an open-source multi-computer collection of software tools and services designed to integrate robotics related hardware, e.g. sensors, actuators, cameras, etc. ROS provides hardware abstraction and a communication system between the nodes that can be synchronous using services or asynchronous using topics.

## 1.6 Organization

This dissertation is organized in the following manner:

- **Chapter 2** First, the robot manipulator system is introduced. Followed by the definitions of path and trajectory. After that, a parameterization of the path is proposed. Also, a cost function based on the concept of work is defined. Then, the control variable is lifted and a linear system is used to guide the dynamics of the trajectory parameter. Finally, the optimization problem is formulated.

- **Chapter 3** In this chapter the trajectory planning process is detailed. A Newton method is used as the basis of the optimization algorithm. And the non-linearities are canceled using a model based cancellation scheme. Simulation results are used in the end of the chapter to illustrate the method.

- **Chapter 4** In this chapter the problem of trajectory stabilization is addressed. In order to accommodate perturbations and any model uncertainty, a receding horizon predictive control is included in the process. Simulation results exemplifying the proposed method are presented at the end of the chapter.

- **Chapter 5** This chapter presents experimental results of the proposed method implemented in a 4 degrees of freedom manipulator named Tetis (Xaud [49],Silva [40]).

- **Chapter 6** In this chapter, the conclusions and a discussion about next steps and future works are presented.

# Chapter 2

# Problem Formulation

The problem of planning a trajectory using the entire system available energy can be formulated as an optimization problem where a cost function is defined as the difference between the predicted energy for the given trajectory and the available energy. Furthermore, the optimization problem can be stated in the following form: given a path and an energy budget, calculate the trajectory such that minimizes the cost function.

This chapter is organized as follows. First, there is an introduction on the concepts of robot manipulator, joint space and task space. Also the control of robot manipulators is introduced. After that, the concepts of path and trajectory are introduced. Then, a parameterization of the path is presented where the dimension of the problem is reduced. Following the path parameterization, an energy function associated to the work done by the manipulator is adopted. Finally, an error function is presented and the problem is formulated in a lifted space.

Here, it is considered that robot position and velocity are measured, and that the system available energy is being measured. Additionally, it is considered that the dynamic model of the manipulator is known.

## 2.1 Robot Manipulator System

A robot manipulator consists in a structure with articulated $N+1$ links connected by $N$ joints (Siciliano et al. [39]). It is assumed that the links are rigid. The joints can be prismatic or revolute, where the prismatic provides translational motion between the links and the revolute provides rotational motion between the links (Siciliano et al. [39]). The structure of links connected through joints is classified as open or closed kinematic chain. Open when there is only one sequence of links connecting the two extremes of the chain, and closed when the chain forms a loop (Siciliano et al. [39]). We only consider robotic manipulators forming open kinematic chains. Typically, in the end of the chain, i.e. the end-effector, is where the tools used by

the manipulator are attached.

Consider a reference coordinate system $O_B$ located in the manipulator base and a coordinate system $O_E$ located in the end-effector as represented in figure 2.1.


Figure 2.1: Manipulator with base and end-effector coordinate systems. This image is modified from https://new.abb.com

A minimal representation of the pose of the end-effector in a $D$-dimensional operational or task space is described with respect to the base frame as (Siciliano et al. [39])

$$p = \begin{bmatrix} \rho \\ \phi \end{bmatrix} \tag{2.1}$$

where $\rho \in \mathbb{R}^3$ represents the position in the base reference and $\phi \in \mathbb{R}^r$ is a representation of the orientation of $O_E$ with respect to $O_B$ (Siciliano et al. [39]), and $r$ is the dimension of the parameterization of the orientation representation. Note that the orientation of a rigid body is in $SO(3)$ space, but it is represented with Euler angles, roll-pitch-yaw or quaternion (Siciliano et al. [39]).

The end-effector pose can also be described in the joint space, i.e., in terms of

its joints positions:

$$q = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_N \end{bmatrix} \tag{2.2}$$

where $q \in \mathbb{R}^N$ and $q_i$ can be prismatic or revolute.

The kinematics maps the joint space representation to the task space representation (Siciliano et al. [39]), i.e.

$$p = k(q) \tag{2.3}$$

where $k(\cdot)$ is a nonlinear $\mathbb{R}^N \rightarrow \mathbb{R}^D$ operator (Siciliano et al. [39]).

Consider the Euler-Lagrange dynamic model of a robot manipulator described in the joint space given by

$$M(q)\ \ddot{q} + C(q,\dot{q})\ \dot{q}\ + G(q) = \tau \tag{2.4}$$

where $\tau \in \mathbb{R}^N$ is the joint torque, $M \in \mathbb{R}^{N \times N}$ is the inertia matrix, $C \in \mathbb{R}^{N \times N}$ is the Coriolis and centripetal effects matrix and $G \in \mathbb{R}^{N \times 1}$ is the gravity term (Siciliano et al. [39]).

Equivalently, the dynamics of a robot manipulator in the task space is given by

$$\bar{M}(p)\ \ddot{p} + \bar{C}(p,\dot{p})\ \dot{p}\ + \bar{G}(p) = F \tag{2.5}$$

where $F \in \mathbb{R}^D$ is the generalized force vector, which is equivalent to the forces and torque on the end-effector that would be generated if $\tau$ was applied on the joints (Siciliano et al. [39]).

Matrices $\bar{M}$, $\bar{C}$ and $\bar{G}$ are obtained from the joint space model as (Siciliano et al. [39]):

$$\begin{aligned} \bar{M} &= (\mathscr{J} M^{-1} \mathscr{J}^T)^{-1} \\ \bar{C}\ \dot{p} &= \bar{M}\, \mathscr{J} M^{-1} C\dot{q} - \bar{M}\, \dot{\mathscr{J}}\, \dot{q} \\ \bar{G} &= \bar{M}\, \mathscr{J} M^{-1} G \end{aligned} \tag{2.6}$$

where $\mathscr{J}(q)$ is the analytic Jacobian, i.e.

$$\dot{p} = \mathscr{J}(q)\dot{q} \tag{2.7}$$

### 2.1.1 Control of Robot Manipulators

The control of system (2.4) consists in finding $\tau(t)$ such that $q(t) \rightarrow q_d(t)$, where $q_d(t)$ is the desired configuration in the joint space over time (Hsu & Lizarralde [18]). In the task space, the control of system (2.5) consists in finding $F$ such that $p(t) \rightarrow p_d(t)$, where $p_d(t)$ is the desired configuration in the operational space over time (Hsu & Lizarralde [18]).

### 2.1.2 Computed Torque Control

The computed torque control technique consists in using the model of the plant to compensate the non-linearities (Spong et al. [44], Craig [9]). It is assumed that $M$, $C$ and $G$ in (2.4) are known and that $q$ and $\dot{q}$ are measured. The control law is given by

$$\tau = \hat{M}(q)\ u + \hat{C}(q, \dot{q})\ \dot{q} + \hat{G}(q) \tag{2.8}$$

where $u$ is an auxiliary control and hat indicates the model of the manipulator dynamics. Applying this controller to the dynamic model (2.4) results in

$$M(q)\ \ddot{q} + C(q, \dot{q})\ \dot{q} + G(q) = \hat{M}(q)\ u + \hat{C}(q, \dot{q})\ \dot{q} + \hat{G}(q) \tag{2.9}$$

And if the models are accurate and we consider $\hat{M} = M$, $\hat{C} = C$ and $\hat{G} = G$ we have

$$\ddot{q} = u \tag{2.10}$$

The equivalent system if the model compensates perfectly the non-linearities would be a double integrator.

The same process can be used in the task space formulation and the model can be used to cancel the non-linearities. So the control is given by

$$F = \hat{\bar{M}}\ \bar{u} + \hat{\bar{C}}\ \dot{p}\ + \hat{\bar{G}}(p) \tag{2.11}$$

where $\bar{u}$ is an auxiliary control signal and hat indicates the model. And again, if the model is considered ideal ($\hat{\bar{M}} = \bar{M}$, $\hat{\bar{C}} = \bar{C}$ and $\hat{\bar{G}} = \bar{G}$) we have

$$\ddot{p} = \bar{u} \tag{2.12}$$

Since the models are usually imperfect, it is common to include in (2.12) and in (2.10) terms of position and velocity feedback to improve path tracking performance, i.e.,

$$u = \ddot{q}_d + K_p\ (q_d - q) + K_d\ (\dot{q}_d - \dot{q}) \tag{2.13}$$

$$\bar{u} = \ddot{p}_d + K_p\ (p_d - p) + K_d\ (\dot{p_d} - \dot{p}) \tag{2.14}$$

where $K_p \in \mathbb{R}^{N\times N}$ and $K_d \in \mathbb{R}^{N\times N}$ for joint space and $K_p \in \mathbb{R}^{D\times D}$ and $K_d \in \mathbb{R}^{D\times D}$ for task space.

By inserting equation (2.10) in (2.13) for joint space and equation (2.12) in (2.14) for task space, the error dynamics is

$$\ddot{e} + K_d\dot{e} + K_p e = 0 \tag{2.15}$$

where $e = q_d - q$ for joint space and $e = p_d - p$ for task space. Choosing positive $K_p$ and $K_d$ gains ensures that the error decays exponentially and the control objective is satisfied.

## 2.2 Robot Manipulator Path Parameterization

The *path* consists of a collection of points in the joint or in the task space (Siciliano et al. [39]) regardless of the moment in time when the manipulator reaches these points. The *trajectory* on the other hand, defines velocities and accelerations for each point in the path (Siciliano et al. [39]).

The path is parameterized by a single variable $s = \{s \in [0, 1]\}$ that represents the normalized arc length along the path (Verscheure et al. [46], Verscheure et al. [47], Lin et al. [24]). Moreover, the path starts in $s(0) = 0$ and ends in $s(T) = 1$, where $T$ is the final time. Also, to guarantee the manipulator moves forward in time $\dot{s}(t) \geq 0\ \forall\ t \in [0, T]$. And it is considered that the robot manipulator has to follow a pre-defined path in the joint space $q_d(s)$ or in the task space $p_d(s)$.

The path parameter determines the spatial configuration of the path, where the trajectory time dependency follows from the relation $s(t)$ (Verscheure et al. [47]).

### 2.2.1 Trajectory Characterization in Joint Space

A trajectory in the joint space $\underline{q_d}$ connects the initial configuration $q_d(0)$ to the final configuration $q_d(T)$ and is defined as

$$\underline{q_d} = \{q_d(t) \in \mathbb{R}^N, t \in [0, T]\} \tag{2.16}$$

where $T$ is the final time. Or it can be defined in terms of the path parameter $s$

$$\underline{q_d} = \{q_d(s) \in \mathbb{R}^N, s \in [0, 1]\} \tag{2.17}$$

The map of the path parameterization in the joint space is defined as

$$\begin{aligned} q_d &= f_q(s) \\ \dot{q_d} &= \nabla f_q(s)\ \dot{s} \\ \ddot{q_d} &= \nabla^2 f_q(s)\ \dot{s}^2 + \nabla f_q(s)\ \ddot{s} \end{aligned} \tag{2.18}$$

It is considered that the system starts in the path, i.e.

$$\begin{aligned} q(0) &= q_d(0) \\ \dot{q}(0) &= \dot{q}_d(0) \\ \ddot{q}(0) &= \ddot{q}_d(0) \end{aligned} \tag{2.19}$$

### 2.2.2 Trajectory Characterization in Task Space

The desired task space trajectory is defined as $\underline{p_d}$ connecting the initial configuration $p_d(0)$ to the final configuration $p_d(T)$. And is defined as

$$\underline{p_d} = \{p_d(t) \in \mathbb{SE}^3, t \in [0, T]\} \tag{2.20}$$

where again $T$ is the final time. Or it can also be defined in terms of the path parameter $s$

$$\underline{p_d} = \{p_d(s) \in \mathbb{SE}^3, s \in [0, 1]\} \tag{2.21}$$

The map of the path parameterization is given by

$$\begin{aligned} p_d &= f_p(s) \\ \dot{p_d} &= \nabla f_p(s)\ \dot{s} \\ \ddot{p_d} &= \nabla^2 f_p(s)\ \dot{s}^2 + \nabla f_p(s)\ \ddot{s} \end{aligned} \tag{2.22}$$

Also, consider $\underline{s}$ as being a trajectory of the parameterized path, where $\underline{s}$ connects an initial configuration $s(0)$ to a final configuration $s(T)$. The trajectory $\underline{s}$, regardless if the problem is defined in the task or joint space, is denoted by

$$\underline{s} = \{s(t) \in \mathbb{R}, t \in [0, T]\} \tag{2.23}$$

Figure 2.2 illustrates a parameterized path in the task space.

## 2.3 Energy Function

The energy function represents the work done by the manipulator (see appendix A for more details about manipulator energy consumption and this particular choice

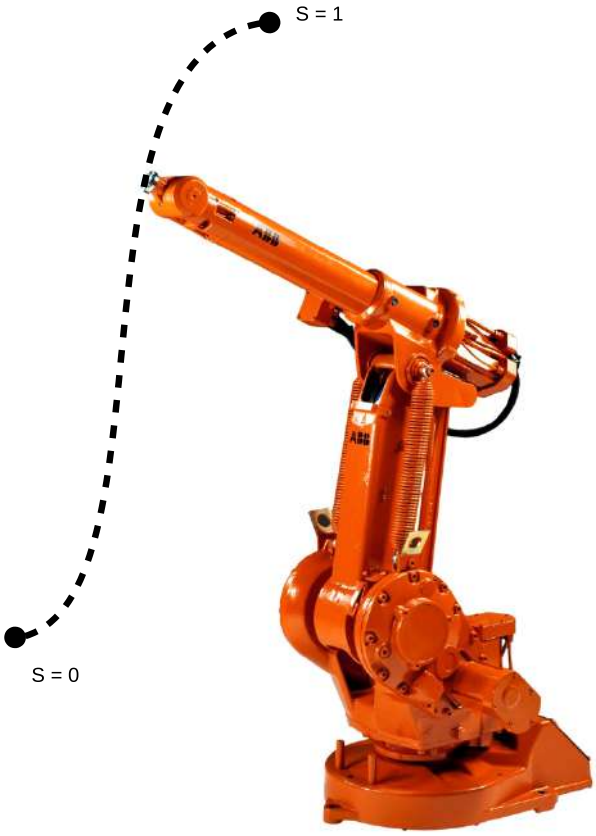


Figure 2.2: Parameterized Path of a Manipulator - This image is modified from https://new.abb.com

of energy function). Then

$$J_T = \int_0^T |\dot{q}|^T |\tau| dt$$
$$|\dot{q}| = \begin{bmatrix} |\dot{q}_1| \\ \vdots \\ |\dot{q}_N| \end{bmatrix} \qquad |\tau| = \begin{bmatrix} |\tau_1| \\ \vdots \\ |\tau_N| \end{bmatrix} \tag{2.24}$$

Note that since only absolute values of torque and joint speeds are considered, this choice of energy function does not contemplate the case where the actuators are generating energy. The actuators works as generators when the torque and the angular speed has different direction, e.g. during decelerations or when the actuator load decrease to a state with less potential energy. Not considering the regeneration on the energy function could be consider a minor restriction because most industrial manipulators dissipate the regenerated energy in shunt resistors.

The energy budget available on the system to perform the trajectory is denoted here as $J_B$.

Changing the integration variable, the energy function (2.24) can be expressed

in terms of trajectory parameterization:

$$J_T = \int_0^1 |\dot{q}|^T |\tau| \ \frac{1}{\dot{s}} \ ds \tag{2.25}$$

Equivalently, the energy function can be defined in the task space:

$$J_T = \int_0^1 |\dot{p}|^T |F| \ \frac{1}{\dot{s}} \ ds \tag{2.26}$$

## 2.4 Optimization Problem Formulation

The problem is formulated as an optimization problem where the trajectory is replaned to consume the entire energy budget. Hence, the problem is formulated as follows

*For a given path and an initial energy budget $J_B$, find the trajectory $\underline{s}$ such that $J_T(\underline{s}) = J_B$.*

### 2.4.1 Methods used for Trajectory Optimization

Note that the problem proposed above can be tackled as a minimum time with energy restriction. Although not a common restriction on a minimum time optimization for manipulators, in (Shin & McKay [38]) and in (Pfeiffer & Johanni [31]) dynamic programming was used to find optimal phase plane trajectory to solve the time-optimal problem while minimizing a cost function such as energy. Also in (Field & Stepanenko [13]), dynamic programming was used but to solve the minimum energy optimization problem with trajectory completion time as a constraint. In (Shiller [36]) the time-energy problem is solved using Hamiltonian and solving the co-state equation. By adjusting a weight factor to favor time or energy on the cost function the problem proposed here could be also addressed using the method in (Shiller [36]). In (Verscheure et al. [46]), instead of dynamic programming the authors use second-order cone programming. In (Xu et al. [50]) the authors use a evolutionary type of algorithm to solve time-energy optimization. Also the method used in (Lin et al. [24]) to optimize the service life of the actuators based on sequential quadratic programming could be adapted to solve an energy optimization.

Many of these solutions could be directly or indirectly in some cases used here to find the trajectory that consume a given energy budget. But none are fast enough to be used in a real time application.

The methodology proposed in this dissertation consists in minimizing the error iteratively in a lifted decision space. This approach allows the optimization to be solved in real time by a simple Newton method. A linear system is introduced to

control the dynamic of the parameterized trajectory. The input of this system is the lifted decision variable.

### 2.4.2 The Lifted Decision Space

Similar to (Lin et al. [24]), the optimization problem is solved in a lifted decision space. To that end, parameter $s$ is governed by a control input $v$ through a known (controllable and stable) linear system, i.e.

$$\begin{aligned} \dot{z} &= A_z\ z + B_z\ v \\ y &= s \\ z &= \begin{bmatrix} s \\ \dot{s} \\ \ddot{s} \end{bmatrix} \end{aligned} \tag{2.27}$$

with $z(0) = 0$ and constant input $v = \{v \in \mathbb{R} \mid v > K_{dc}\}$ guaranteeing that $s(T) = 1$ with finite time $T$; $A_z \in \mathbb{R}^{3\times3}$ and $B_z \in \mathbb{R}^3$. The system (2.27) can be considered a design variable. Furthermore, $K_{dc}$ is the DC gain of the transfer function corresponding to (2.27).

The eigenvalues of $A_z$ dictate the speed of the system and the settling time should have the same magnitude as the desired trajectory duration.

The trajectory is lifted to a decision space where the input $v$ controls its speed. The linear system maps the trajectory to the lifted space. So the error is not minimized directly.

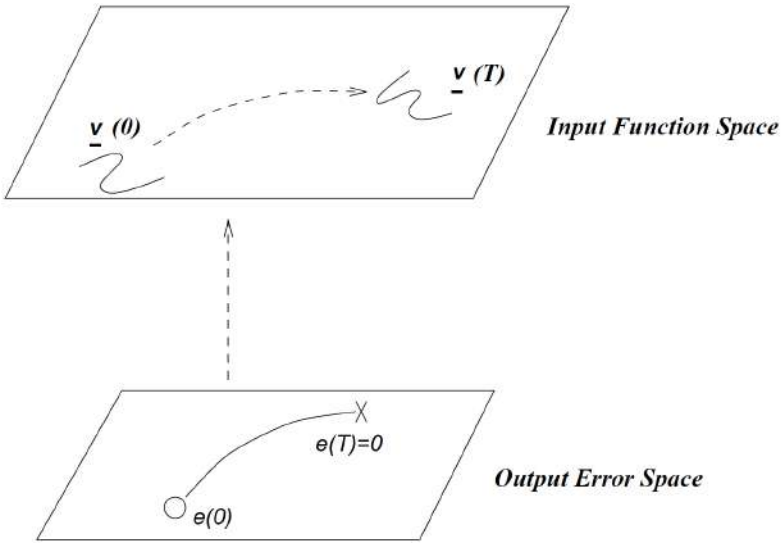


Figure 2.3: Lifted Decision Space - Modified from Wen [48]

With the lifted space the variable $v$ controls the parameterized trajectory $z$, which is used to calculate the energy $J_T$.

The block diagram in figure 2.4 shows the control scheme considering the path parameterization and the lifted space in the joint space formulation. The desired tracking values for joints position ($q_d$), joints velocity ($\dot{q}_d$) and joints acceleration ($\ddot{q}_d$) are obtained from the trajectory predictor block through the path parameterization. The parameterized trajectory is also used to calculate the energy based on the energy model. The energy budget $\tilde{J}_B$ is obtained from the system.

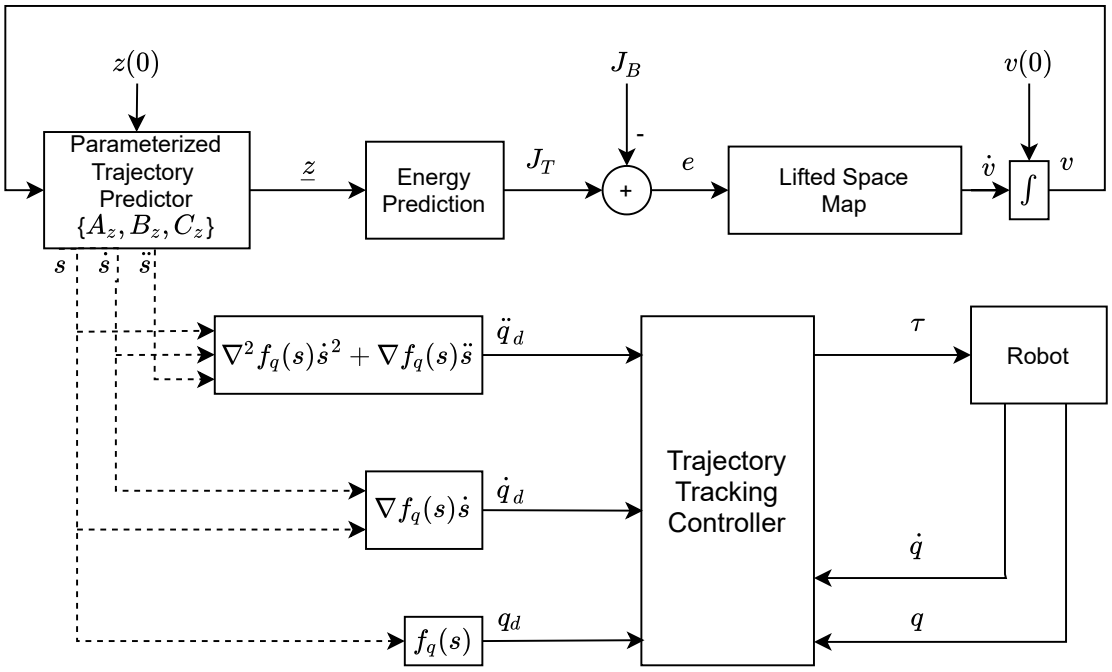


Figure 2.4: Joint Space Control block diagram

The block diagram in figure 2.5 shows the control scheme considering the path parameterization and the lifted space in the task space formulation. The desired tracking values for position ($p_d$), velocity ($\dot{p}_d$) and acceleration ($\ddot{p}_d$) are obtained from the trajectory predictor block through the path parameterization. The parameterized trajectory is also used to calculate the energy based on the energy model. The energy budget $\tilde{J}_B$ is obtained from the system.

**Remark 1** *The system 2.27 dictates the dynamics of the trajectory. Therefore $A_z$, $B_z$ and $v(0)$ should be carefully selected considering not only the task duration but its evolution along the path. Also, $s(t)$ must be smooth of class $C^2$ because otherwise the parameterization 2.18 will not hold.*

*In (Lin et al. [24]), a system with three poles at zero is used, i.e.*

$$A_z = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}; B_z = \begin{bmatrix} 0 \\ 0 \\ 0.01 \end{bmatrix} \tag{2.28}$$

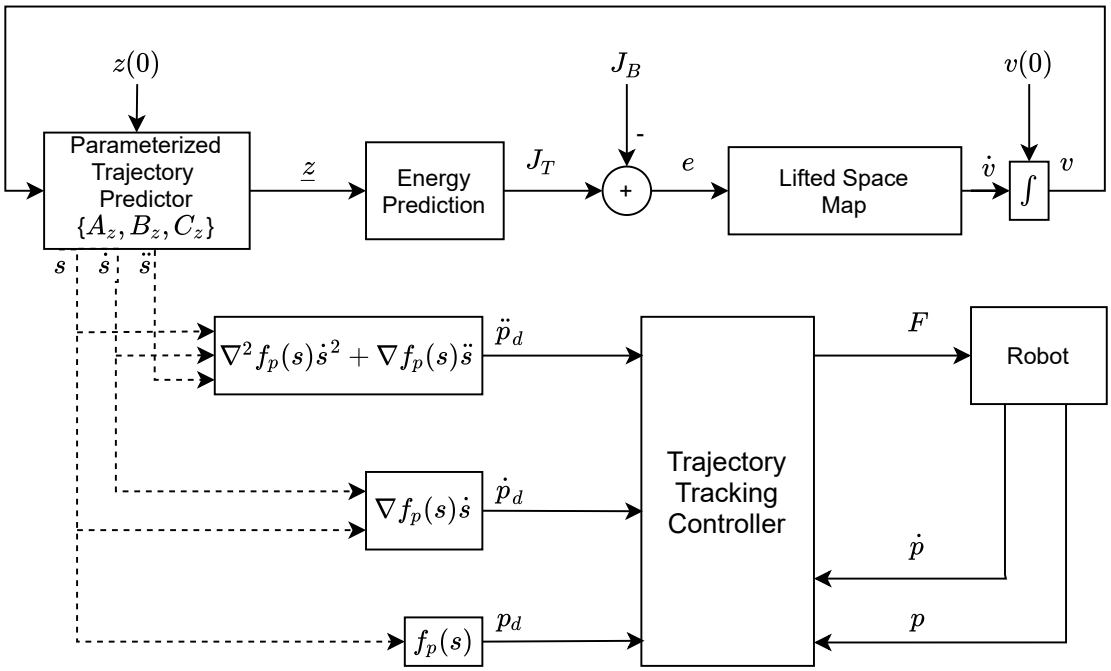


Figure 2.5: Task Space Control block diagram

*And the step response for a step with amplitude of* 10 *is shown in figure 2.6. For this particular example the path is executed in almost* $4s$.

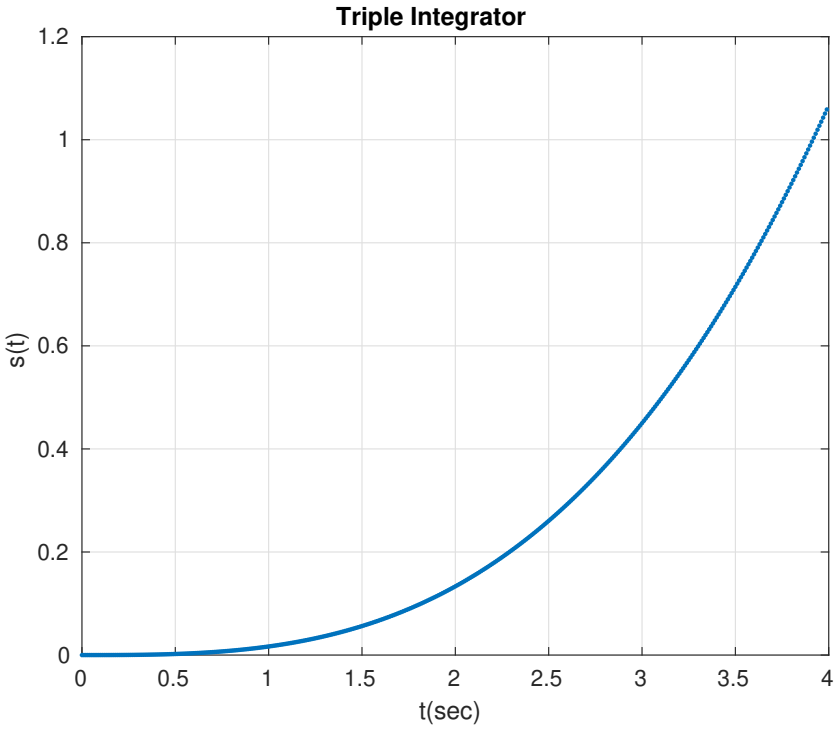


Figure 2.6: Lifted Decision Space - Triple Integrator - s(t) with v=10

*Here, in order to get more rapid response in the beginning of the trajectory, a*

*different approach is proposed. A system with three stable and real poles, i.e.*

$$A_z = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -5 \cdot 10^{-2} & -1 & -2 \end{bmatrix}; B_z = \begin{bmatrix} 0 \\ 0 \\ 5 \cdot 10^{-2} \end{bmatrix} \tag{2.29}$$

*This system has poles in* $-0.06$, $-0.74$ *and* $-1.20$. *Note that one pole is dominant and it can be used to scale the system desired path completion time for a given* $v$. *Meaning that the overall speed of the system can be adjusted with the dominant pole. The step response for a step with amplitude of* 10 *is on figure 2.7. For this particular example the path is executed in almost* $4s$.

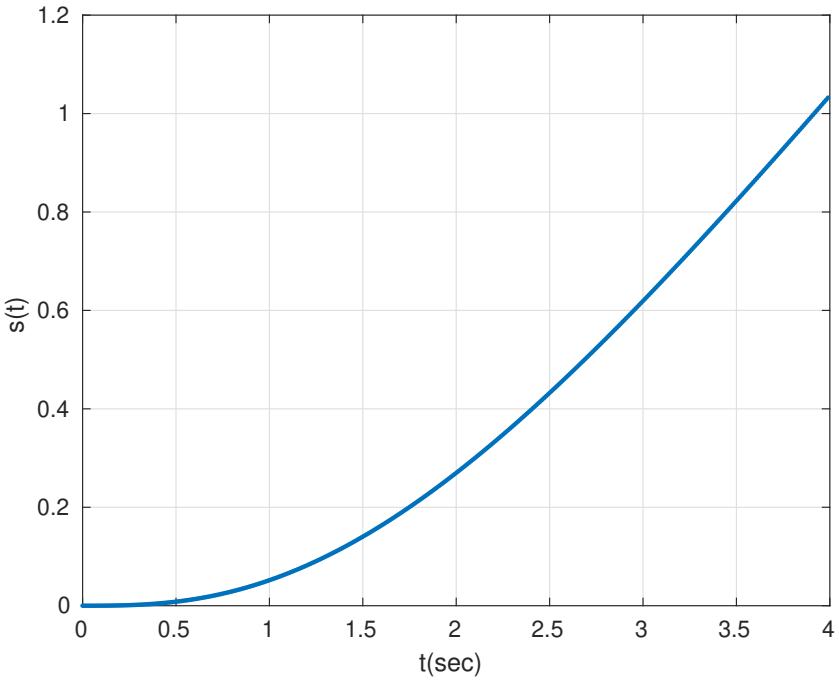


Figure 2.7: Lifted Decision Space - Stable System - $s(t)$ with $v = 10$

*Both systems have a limitation in the initial rise period. In the first* $1s$ *in the triple integrator and* $0.5s$ *in the proposed linear and stable system, the systems are almost static. This fact is not a major problem. The only consequence is that the trajectories, regardless of the available energy, starts slow. Figure 2.8 shows both solutions during the first second. The proposed system mitigates the slow start issue. This discussion is open for future developments.*

**Remark 2** *The choice of a constant* $v$ *limits the search space of the optimization problem to* 1 *degree of freedom. If more degree of freedom are needed (e.g. to satisfy an additional constraint)* $v$ *could be represented as a power series.*

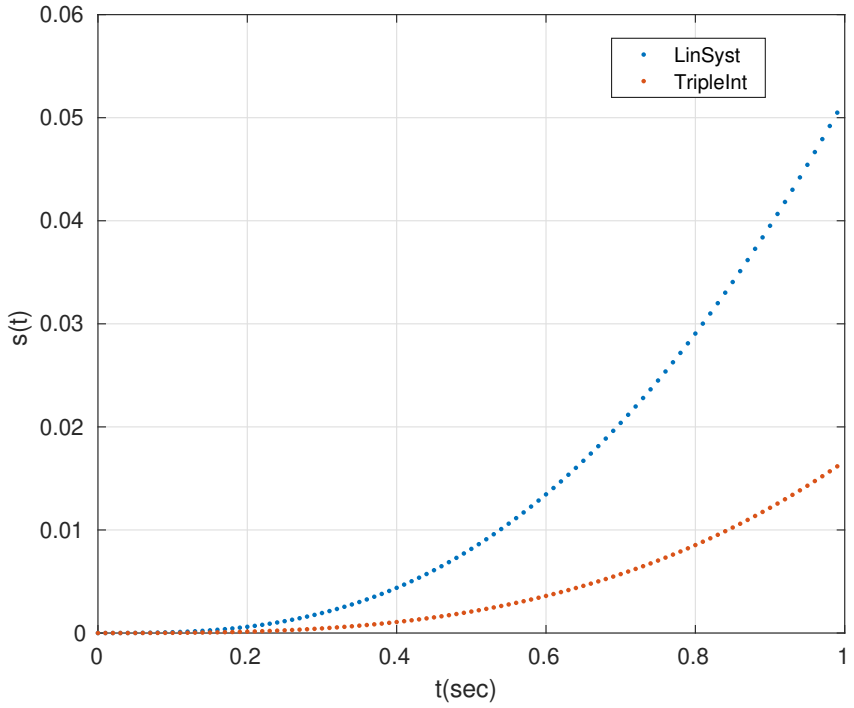


Figure 2.8: Lifted Decision Space - s(t) Rise Time Comparisson with v=10

## 2.5 Problem Formulation on the Lifted Space

The problem is formulated as an optimization problem where the difference between $J_T$ and $J_B$ must be minimized.

Therefore, the problem can be formulated as follows:

*Find the control signal $v$ that drives the error*

$$e = J_T(v) - J_B \tag{2.30}$$

*to zero, with $s$ subject to:*

$$\dot{z} = A_z\ z + B_z\ v \tag{2.31}$$

*with $z(0) = 0$.*

## 2.6 Conclusions

This chapter formulates the problem of trajectory optimization for a given energy budget. First, the manipulator dynamic equation in the task and in the joint space is presented followed by the concepts of path and trajectory. After that a trajectory parameterization is introduced to reduce the dimension of the problem. Later, an energy function representing the work performed by the actuators is presented. With all that in place the problem is formulated. Following the formulation of the problem,

a brief discussion showing how the optimization problem could be solved with the available techniques on the scientific robotics community. Furthermore, the method proposed in this dissertation consists in solving the optimization in a lifted decision space. And to that end, a linear system is used as a control variable to govern the trajectory dynamics. Finally, the optimization problem is reformulated considering the lifted decision space approach. In the next chapter a gradient descent newton method is used to drive the error 2.30 to zero by correcting the input $v$ iteratively.

# Chapter 3

# Trajectory Planning

Although the path or trajectory planning can be treated as an optimal control problem (Bobrow et al. [5], Shin & McKay [37], Shiller [35]), here the problem is optimized in a lifted decision space $v$. The control $v$ is corrected iterativelly in order to minimize the error using Newton method.

This chapter shows the tools used to plan the optimized trajectory: the iterative Newton method and model based linearization. Section 3.1 presents the iterative Newton method. It is the base to understand how control $v$ is corrected to minimize the error (2.30). Section 3.2 shows how the non-linearities are canceled. Finally, on section 3.3 the problem is addressed with the tools presented in the first two sections.

The proposed method for trajectory planning is demonstrated on numerical example 1 where a system with one revolute joint is considered. Later in example 2 a condition that could lead to oscillations on the error due to large Newton step is shown. Then, in example 3 this oscillation is dealt with the use of Armijo rule. Finally, a simplification is proposed to consider the map between the error and the control input $v$ as always positive.

## 3.1 Iterative Newton Method

The method proposed here consists in transforming the optimization problem in a root-finding problem of an algebraic function dependent on the input. And then solving the optimization in a lifted decision space - the input function space. After that, an iterative Newton method is used to find the zero by updating the input $v$ in the direction the error magnitude decreases (similar to (Lizarralde et al. [26]) for nonholonomic systems and (Sontag [43]) for systems without drift). Although in chapter 2 $v$ is considered constant, here in chapter 3 it is updated to minimize the error, however $v$ is constant in the sense that it does not change during a trajectory.

Hence, the objective is to find an input $v \in \mathbb{R}$ such that the error

$$e = J_T(v) - J_B = 0 \tag{3.1}$$

In a nutshell, the method consists in first choosing an initial guess $v_0$ that leads to error $e_1 = J_T(v_0) - J_B$. If the resulting error $e_1$ is not small enough, the input $v_0$ is corrected by a small change $\Delta v$, i.e., $v_1 = v_0 + \Delta v$. Then the new error is calculated $e_2 = J_T(v_1) - J_B$ and if the error still not small enough the process continues.

In order to calculate $\Delta v$, a continuous map from control $v$ to error $e$ has to be established. This mapping correlates infinitesimal variations between $v$ and $e$ over time.

Since budget $J_B$ is assumed to be constant, the differentiation of (3.1) with respect to the iteration variable $t$ leads to

$$\frac{de}{dt} = \nabla_v J_T(v) \ \frac{dv}{dt} \tag{3.2}$$

where $\nabla_v J_T(v)$ is the derivative (Sontag [42]) of $J_T$ with respect to $v$. The map from input $v$ to energy function $J_T(v)$ can be calculated through linearization of the system about $v$ (see (Sontag [42])[section 2.8] where is established that the transition map of linearization equals the linearization map of transition).

The condition is that the mapping from the input to the error shall be onto, which is equivalent to the existence of an input that can drive the error to zero in finite time. Hence, this condition implies global controllability (Sontag [43], Lizarralde et al. [26], Sontag [42], Lizarralde [25]). Global controllability implies that the system is globally controllable if and only if the map $\nabla_v J_T(.)$ is onto for every $v$ (Lizarralde [25]).

Thus, assuming that $\nabla_v J_T(v)$ is onto, the control variable $v$ can be updated in such a way that minimizes the error:

$$\frac{dv}{dt} = -\mu \left[\nabla_v J_T(v)\right]^{-1} \ e(t) \tag{3.3}$$

where $\mu > 0$ dictates the rate of convergence. Furthermore, the convergence of $e$ to zero can be verified by substituting (3.3) in (3.2):

$$\frac{de}{dt} = -\mu \, e(t) \tag{3.4}$$

which implies the exponential convergence of $e(t)$ to zero.

**Remark 3** *Continuation methods are used to solve non-linear problems by deforming the problem in a family of similar problems by changing a given parameter. For*

*one value of this parameter the resulting problem has a known solution and for another value its the original problem. The parameter is then corrected by a iteration process starting on the known solution and ending on the original problem.*

*Although the overall formulation is different, the solution used here fits the context of continuation processes, where the trajectory corresponding to an initial condition $v_0$ is deformed (or iterativelly corrected) until the final trajectory that minimizes the error.*

**Remark 4** *If necessary, the input $v$ could be modified to add degrees of freedom to the problem for considering torque and speed constraints. Furthermore, $v$ could be any function (e.g. $v \in L^2$), or belong to a finite dimension class (Lizarralde [25]) (e.g power series or Fourier series). Here, the first $M$ elements of a power series is considered as a base, i.e.,*

$$v(t) = \sum_{i=0}^{M} \lambda_i \; t^i \;, \forall t \in [\,0, T] \tag{3.5}$$

*A homotopy $Q$ can be established such as $v = Q^{-1}\lambda$ where $\lambda$ is the vector with the coefficients $\lambda = \begin{bmatrix} \lambda_1 & \lambda_2 & \dots & \lambda_M \end{bmatrix}$. Equation (3.2) is written in terms of $\lambda$*

$$\frac{de}{dt} = \nabla_\lambda J_T(\lambda) \; \frac{d\lambda}{dt} \tag{3.6}$$

*And to steer the error to zero $\lambda$ have the following update rule*

$$\frac{d\lambda}{dt} = -\mu \left[\nabla_\lambda J_T(\lambda)\right]^{-\dagger} \; e(t) \tag{3.7}$$

*where $[\,\cdot\,]^{\,\dagger}$ is the pseudo-inverse of Moore-Penrose.*

## 3.2 Model-Based Control

Here a model-based controller as described in section (2.1.1) is proposed to cancel the dynamics and to feedback linearize the robot manipulator system. So for joint space formulation, the system (2.4) is linearized with the following controller:

$$\tau = \hat{M}(q) \; u + \hat{C}(q, \dot{q}) \; \dot{q} + \hat{G}(q) \tag{3.8}$$

And for task space formulation the system (2.5) is linearized with the following controller:

$$F = \hat{\bar{M}}(p) \; u + \hat{\bar{C}}(p, \dot{p}) \; \dot{p} \; + \hat{\bar{G}}(p) \tag{3.9}$$

The feedforward term $u$ here is either $u = \ddot{q}_d$ if joint space formulation or $u = \ddot{p}_d$ if task space formulation. A linear system is defined where $u$ is the input and the state is $x = \left[\begin{smallmatrix} q \\ \dot{q} \end{smallmatrix}\right]$ or $x = \left[\begin{smallmatrix} p \\ \dot{p} \end{smallmatrix}\right]$ for joint space or task space respectively.

$$\dot{x} = A\ x + B\ u \quad ; \quad y = C\ x \tag{3.10}$$

where

$$A = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ I \end{bmatrix} \quad C = B^T$$

where $A \in \mathbb{R}^{2N \times 2N}$, $B \in \mathbb{R}^{2N \times N}$, $0 \in \mathbb{R}^{N \times N}$ is a null matrix and $I \in \mathbb{R}^{N \times N}$ is the identity matrix.

The energy function (2.25) is redefined for the linearized system as

$$\tilde{J}_T = \int_0^1 |y(s)|^T\ |u(s)|\ \left(\frac{1}{\dot{s}}\right) ds \tag{3.11}$$

For the joint space for example we have that

$$y(s) = \dot{q}(s) = \dot{q}_d(s) + r_v(s) \tag{3.12}$$

$$u(s) = \ddot{q}(s) = \ddot{q}_d(s) + r_a(s) \tag{3.13}$$

where $r_v$ and $r_a$ are the residual tracking error for velocity and acceleration. Under the assumption that the tracking errors can be neglected and from (2.18) we have

$$y(s) \approx \dot{q}_d(s) \tag{3.14}$$

$$y(s) \approx \nabla f_q(s)\ \dot{s} \tag{3.15}$$

$$u(s) \approx \ddot{q}_d(s) \tag{3.16}$$

$$u(s) \approx \nabla^2 f_q(s)\ \dot{s}^2 + \nabla f_q(s)\ \ddot{s} \tag{3.17}$$

Equivalent conclusions can be drawn for the task space equations. In the remaining of the chapter the residual tracking errors are considered zero. Furthermore, the energy function (3.11) is

$$J_T = \int_0^1 |\nabla f(s)\dot{s}|^T\ \left|\left(\nabla^2 f(s)\dot{s}^2 + \nabla f(s)\ddot{s}\right)\right|\ \left(\frac{1}{\dot{s}}\right) ds \tag{3.18}$$

$$= \int_0^1 |\nabla f(s)|^T\ \left|\left(\nabla^2 f(s)\dot{s}^2 + \nabla f(s)\ddot{s}\right)\right| ds \tag{3.19}$$

where $f(s) = f_q(s)$ if the system is defined in the joint space or $f(s) = f_p(s)$ if defined in the task space. And $z^T = [s\ \dot{s}\ \ddot{s}]$ is governed by (2.31).

By optimizing the energy of the linearized system one can also optimize the energy of the actual manipulator. This assumption holds for the energy function (2.25) where it is considered that the joint actuators cannot work as generators.

The linear system energy $\tilde{J}_T$ does not account for energy spent due to Coriolis effects or gravity. From the joint space equations (2.24) and (2.4):

$$J_T = \int_0^T |\dot{q}|^T |M(q)\ \ddot{q} + C(q,\dot{q})\ \dot{q}\ + G(q)|\ dt \tag{3.20}$$

While for the linear system the energy is

$$\tilde{J}_T = \int_0^T |\dot{q}|^T\ |\ddot{q}|\ dt \tag{3.21}$$

By considering the linearized system in the optimization the energy losses caused by Coriolis and gravity are considered disturbances on the energy budget. Moreover, comparing 3.20 and 3.21 we can see that $\tilde{J}_T$ is scaled from $J_T$ due to the fact that the mass of the double integrators are normalized. So by optimizing the trajectory for $\tilde{J}_T$, the trajectory is also optimized for the real system energy. But on the other hand, there is a scale factor to be found in order to obtain the linear system energy budget from the real system energy budget.

**Remark 5** *In a real system the energy budget would typically be measured from an energy reservoir, e.g. battery state of charge (SoC), fuel tank level, flywheel angular speed, etc. Then, the following is performed to find $\tilde{J}_B$:*

1. *At instant $t = 0$ and with the system's reservoir at its full capacity, i.e., $J_B(0)$, a trajectory that corresponds to a drop into the linear system energy of $\Delta\tilde{J}_B$ is executed.*

2. *Measure the system's reservoir level after the trajectory. The decrease from initial energy level $J_B(0)$ is $\Delta J_B$ and corresponds to $\Delta\tilde{J}_B$.*

*Then the linear system energy level is calculated from the energy reservoir level with*

$$\tilde{J}_B(t) = \frac{\Delta\tilde{J}_B}{\Delta J_B}\ J_B(t) \tag{3.22}$$

## 3.3 Trajectory Planning using Newton Method

Now, considering the energy error function for the linearized system is given by

$$\tilde{e} = \tilde{J}_T(v) - \tilde{J}_B \tag{3.23}$$

the optimization problem is reformulated considering the energy function of the linearized system:

*Find the control $v$ that drives the error function (3.23) to zero, with $s$ subject to:*

$$\dot{z} = A_z z + B_z v \tag{3.24}$$

*with* $z(0) = 0$.

Thus, the Newton method can be applied with update control law (3.3)

$$\frac{dv}{dt} = -\mu \left[\nabla_v \tilde{J}_T(v)\right]^{-1} \tilde{e}(t) \tag{3.25}$$

and the resulting $v$ solution can be used to generate $z = [s\ \dot{s}\ \ddot{s}]$ from (3.24). Then, considering the feedforward term $u$ in (3.8) as:

$$u = \ddot{q}_d$$

with $\ddot{q}_d$ obtained from (2.18) the trajectory planning problem is solved considering the energy budget $\tilde{J}_B$.

**Example 1** *To illustrate the proposed planning method a simple example is presented. The trajectory of a single rotatory joint is re-planned to use a desired energy budget.*

*The linearized system is given by*

$$\dot{x} = A\ x + B\ u \quad ; \quad y = C\ x \tag{3.26}$$

*where*

$$x = \begin{bmatrix} q_1 \\ \dot{q}_1 \end{bmatrix} \quad A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad C = B^T$$

*The z-dynamic system (2.31) is the stable and controllable system given by*

$$A_z = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -3 \cdot 10^{-4} & -1 & -2 \end{bmatrix}; B_z = \begin{bmatrix} 0 \\ 0 \\ 3 \cdot 10^{-4} \end{bmatrix}$$

*The path is connecting $q(0) = 0$ to $q(T) = q_f$ and is described by the function $f_q(s)$:*

$$f_q(s) = sq_f$$

*with derivatives given by:*

$$\nabla f_q(s) = q_f$$

$$\nabla^2 f_q(s) = 0$$

*From equation (3.19), the energy function for the linearized system is*

$$\tilde{J}_T(v) = \int_0^1 |q_f| \, |q_f \ddot{s}| \, ds$$

$$\tilde{J}_T(v) = q_f^2 \int_0^1 |\ddot{s}| \, ds$$

*But, from (Chen [7]) we have that considering $z(0) = 0$ the solution to the linear system is*

$$z = \int_0^t e^{A_z(t-\theta)} \, B_z \, v \, d\theta$$

*And since $v$ is constant can be taken outside the integral sign:*

$$\ddot{s} = v \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \int_0^t e^{A_z(t-\theta)} \, B_z \, d\theta$$

$$\ddot{s} = v \, \alpha$$

*where*

$$\alpha = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \int_0^t e^{A_z(t-\theta)} \, B_z \, d\theta$$

*And then we have*

$$\tilde{J}_T(v) = v \, q_f^2 \int_0^1 |\alpha| \, ds$$

*In order to calculate the mapping $\nabla_v \tilde{J}_T(v)$, a small perturbation $\Delta v$ is applied in $\tilde{J}_T(v)$*

$$\tilde{J}_T(v + \Delta v) = (v + \Delta v) \, q_f^2 \int_0^1 |\alpha| \, ds$$

$$\tilde{J}_T(v) + \tilde{J}_T(\Delta v) = v \, q_f^2 \int_0^1 |\alpha| \, ds + \Delta v \, q_f^2 \int_0^1 |\alpha| \, ds$$

$$\tilde{J}_T(\Delta v) = \Delta v \, q_f^2 \int_0^1 |\alpha| \, ds$$

*We can see that a small change $\Delta v$ in the input $v$ caused the small change in the energy function $\tilde{J}_T(\Delta v)$. And the mapping is then*

$$\nabla_v \tilde{J}_T(v) = \frac{\tilde{J}_T(\Delta v)}{\Delta v} = q_f^2 \int_0^1 |\alpha| \, ds$$

$$\nabla_v \tilde{J}_T(v) = \frac{q_f^2}{v} \int_0^1 |\ddot{s}| \, ds$$

*Note that this mapping is always positive.*

*The Newton method iterations were calculated in Matlab with* $\mu = 1$*. The initial condition is* $v = 1000$ *and the energy budget for the linearized system is* $\tilde{J}_B = 1.3\ 10^5$*.*

*In figure 3.1 the planned energy* $\tilde{J}_T$ *starts in* $2.6\ 10^5$ *and in* $3$ *iterations converges to the energy budget (blue line indicates the energy budget).*

*Figure 3.2 also shows that in 3 iterations the error converges to zero and* $v$ *to final value corresponding to the energy budget.*

*Since the energy level decreases, the control signal also decreases. This can be observed in figure 3.3.*

*In figure 3.4* $s(t)$ *is plotted for the initial and the final Trajectories. It can be noted that for a small energy level the trajectory takes longer to complete.*

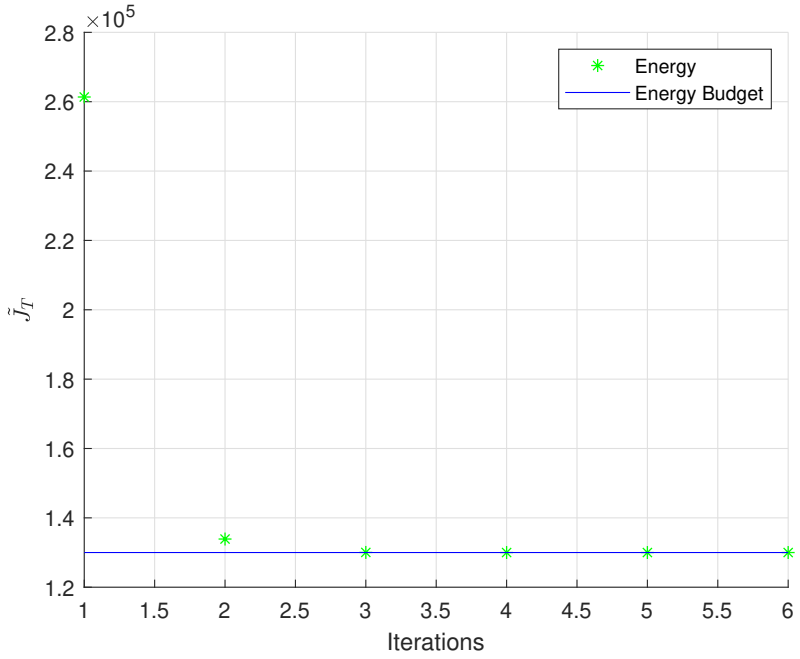


Figure 3.1: Numerical Results 1 Joint - Total energy $J_T$ along the iterations

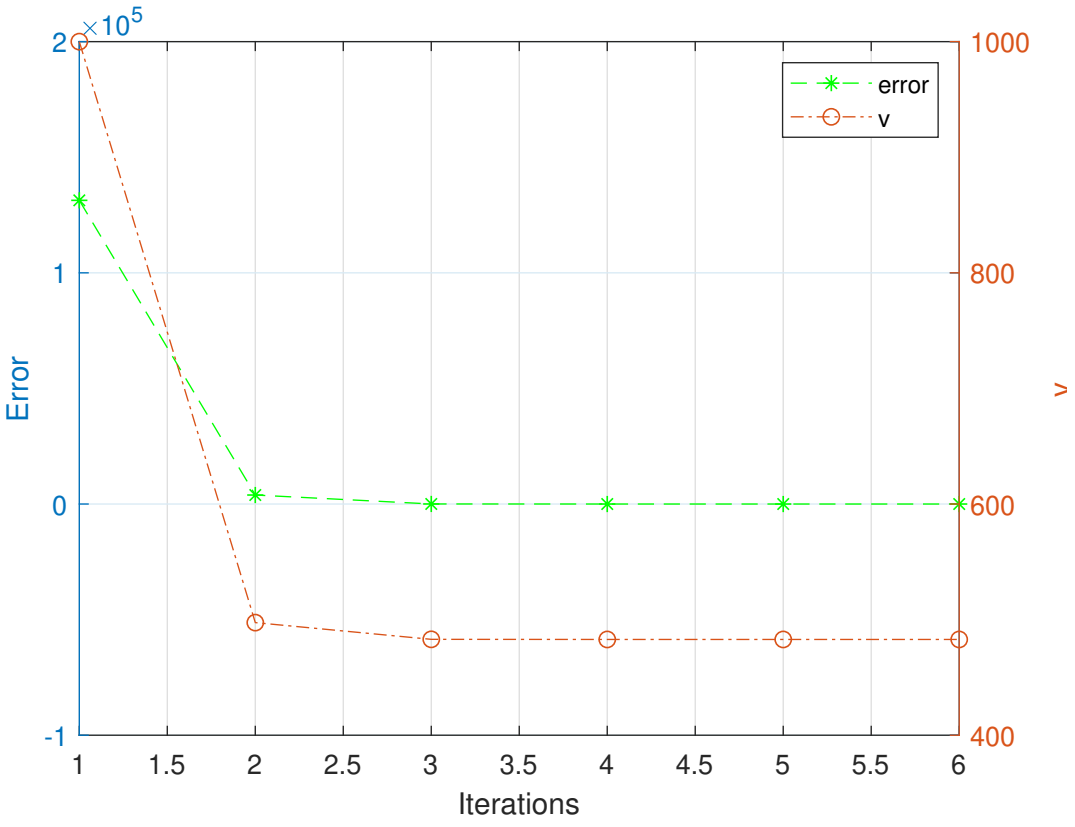


Figure 3.2: Numerical Results 1 Joint - Error $e$ and control variable $v$

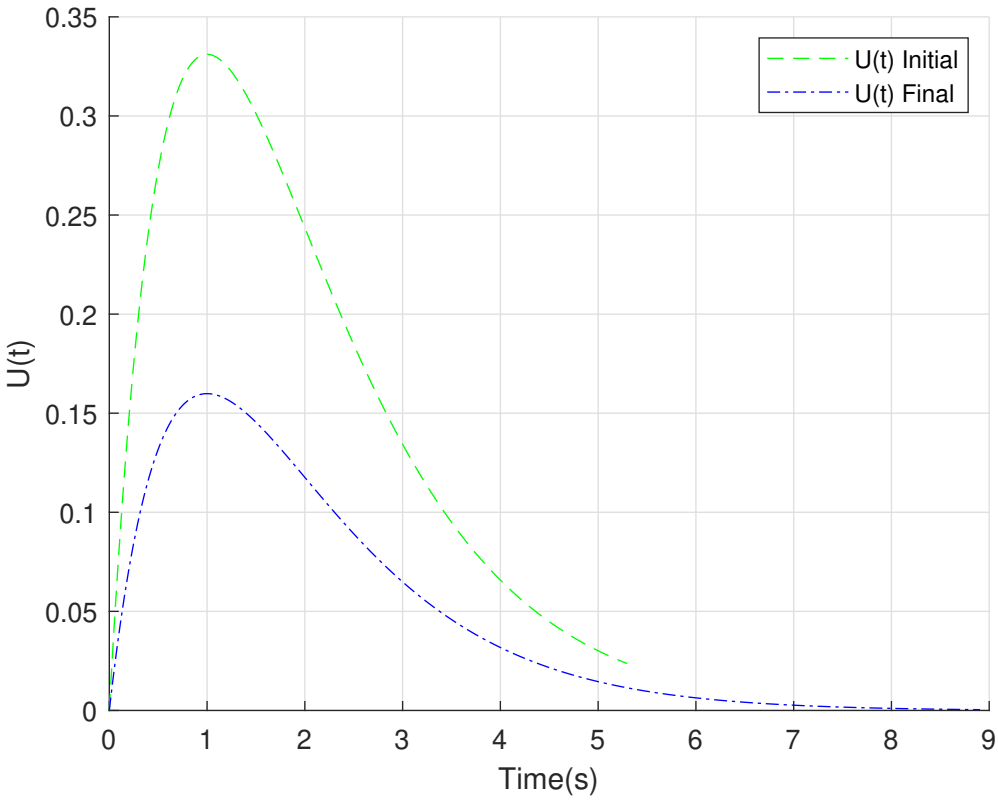


Figure 3.3: Numerical Results 1 Joint - Input $u(t)$ before and after the planning

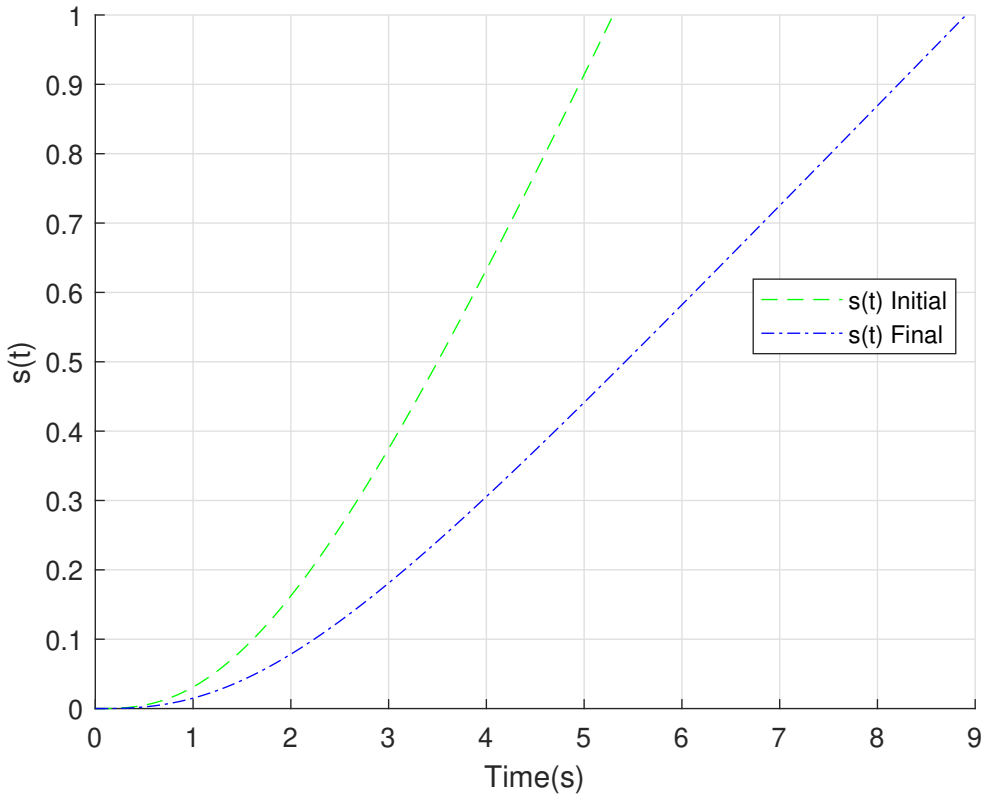


Figure 3.4: Numerical Results 1 Joint - Parameterized trajectory $s(t)$ before and after the planning

Note that the size of the Newton step depends on the parameter $\mu$. If the step is too small, the convergence requires many iterations and is slow. But if the step is too big the error can oscillate around zero. That can happen if the step is bigger then the accepted error level. And instead of reducing the error it changes its signal. Example 2 illustrates this oscillation.

**Example 2** *To illustrate the oscillation of the error due to big $\mu$, the example 1 was modified so $\mu = 2$, $\tilde{J}_B = 1.3 \times 10^6$ and $v(0) = 10^4$. Note that the energy level and $v(0)$ were increased to prevent the Newton step (which now is bigger) to change $v$ to a negative value. The other aspects of the numerical simulation remained the same.*

*Figure 3.5 shows the error along the iterations. The error approximate zero and start oscillating between* 1.608 *and* $-1.608$.

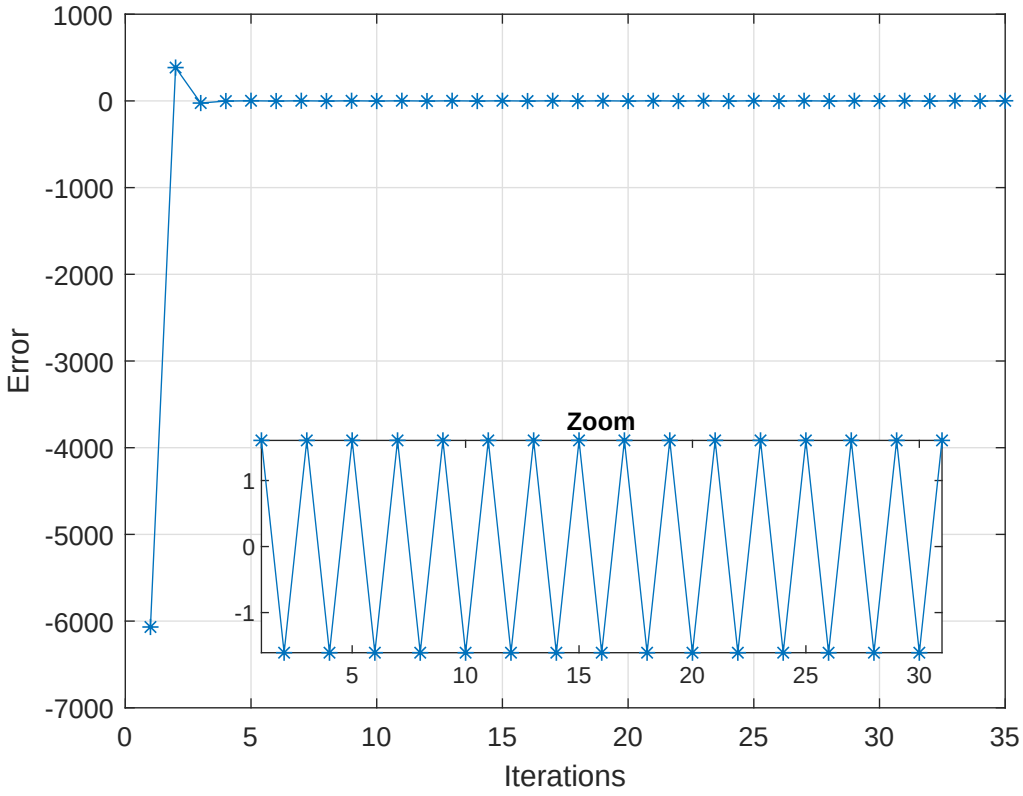


Figure 3.5: Numerical Results 1 Joint Oscillating - Example 2 - Error Oscillating

The Newton step size can be adjusted dynamically using the Armijo rule as described in (Armijo [2]) with the name Modified Steepest Descent. In (Lizarralde [25]) the Armijo rule is also used and its thoroughly described. In short, the Armijo rule reduces the Newton step size if the absolute value of the error did not decrease enough. Example 3 shows the use of Armijo rule.

**Example 3** *In order to illustrate the Armijo rule, example 2 is revisited implementing the steps described in (Lizarralde [25]). The test to decide if the descent was steep enough is*

$$|\tilde{e}(t+\Delta t)| < (1-\delta\mu)\ |\tilde{e}(t)| \tag{3.27}$$

*where* $\delta = 10^{-4}$*. And if the error did not decrease enough* $\mu$ *is corrected with a factor* $\sigma = 0.5$*:*

$$\mu = \sigma\ \mu \tag{3.28}$$

*Initially, as in example 2,* $\mu = 2$*. The error sequence along the iterations is* $-6068$*,* $384.5$*,* $-24.72$ *and* $-1.6077$*. At this point the error would begin to oscillate and the next value would be* $1.6077$*. But with the Armijo rule* $\mu$ *is decreased to* $\mu = 1$*. And the next error is* $-8.51 \times 10^{-7}$*.*

*In figure 3.6 the error along the iterations is plotted.*

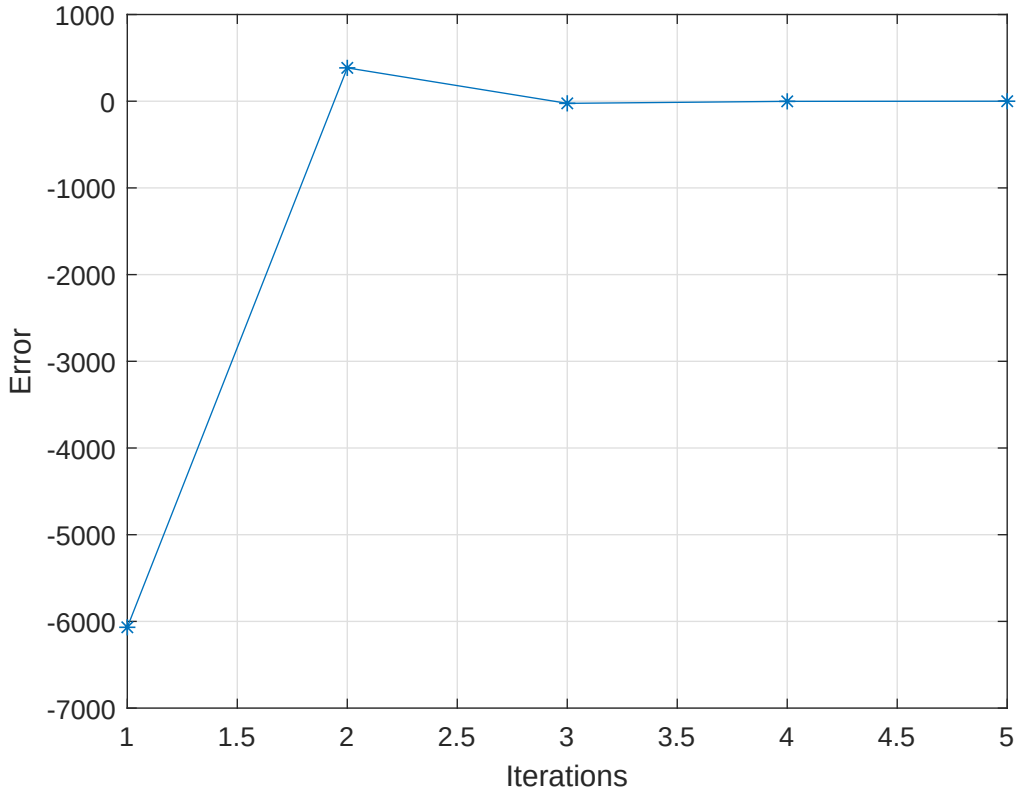


Figure 3.6: Numerical Results 1 Joint With Armijo Rule- Example 3 - Error vs iterations

**Remark 6** *The system (2.31) is chosen so* $s$ *always increase when* $v > 0$*, i.e.* $\dot{s} > 0$*. Consequently,* $v$ *is directly proportional to* $\dot{s}$*.*

*Furthermore, by the work-energy principle a change in the kinetic energy of a body equals the net work performed on it.*

*We can conclude that an increase in $v$ will cause an increase in $\dot{s}$. And this increase in $\dot{s}$ corresponds to higher kinetic energy on the movement and consequently the net work is higher as well. This logic could fail when a decrease in potential energy could make the net work negative regardless of the velocity. But, the energy function (3.19) corresponds to the absolute value of the net work. Hence, $\nabla_v J_T$ is always positive. Thus a simplified update rule for $v$ can be proposed:*

$$\frac{dv}{dt} = -\mu \; e(t) \tag{3.29}$$

## 3.4 Conclusions

In this chapter the method used to plan the optimal trajectory is presented. A mapping between the control input $v$ and the error $e$ is used to correct the control $v$ in the direction that the error decreases. A Newton iterative method is used to update the $v$.

Later, the non-linearities of the system are canceled using a model-based control. The error function and the optimization problem are reformulated for the linearized system.

The method is then verified with a numerical example considering a system with 1 rotatory joint. A second example is devised to show that a big Newton step can lead to oscillations. After that, a third example uses the Armijo rule to reduce the Newton step whenever the error descent is not satisfactory.

Finally, an assumption is made in order to simplify the update rule for the control $v$. Since the proposed energy function does not consider the case where the actuators are regenerating, the map from $v$ to $e$ is always positive.

In the next chapter, the trajectory tracking problem is approached by adapting the method to be done in a closed loop fashion.

# Chapter 4

# Trajectory Tracking

This chapter introduces the receding horizon predictive control that is used for trajectory tracking in a closed loop fashion. This is done to accommodate imperfections or disturbances in the process.

Section 4.1 is the introduction to this chapter, in section 4.2 a strategy for trajectory tracking associating Receding Horizon Predictive Control with the Newton Method is discussed. After that, the method is exemplified with numerical simulations. In section 4.3 is presented an example of the method with a planar RR manipulator and another example with a $4R$ light robot manipulator, named Tetis (Xaud [49],Silva [40]).

## 4.1 Introduction

The optimization process described so far consists in calculating using Newton method the optimized trajectory in open loop. The Newton method calculates the control $v$ that corresponds to the desired energy usage for the given task. Note that if there are imprecisions on the model used on the computed torque or on the system energy budget for example, this open loop strategy would fail. Furthermore, the proposed method is adapted to be executed in closed loop. This strategy is based on predictive control technique associated with the Newton method. The method used here is based on (Lizarralde [25]) for non-holonomic systems. Later similar solution was used in (Chaves & Lizarralde [6]) for autonomous navigation of mobile robots.

A popular way to deal with imprecisions or disturbances is to use a predictive control. In a nut shell, an initial prediction is made for the entire trajectory. The system follows the trajectory for a short period of time and based on the current state update the prediction. And then this is performed over and over until completion of the trajectory.

The Newton method is then merged with the predictive control. Meaning that at every time the prediction is recalculated, the control input is corrected following

the Newton method.

In section 4.2 the Receding Horizon Predictive Control method is detailed for both the joint and task spaces. Next, in section 4.3 the method is demonstrated with numerical examples.

## 4.2 Receding Horizon Predictive Control with Newton Method

The main idea is to calculate the control and the energy for the entire trajectory based on the current value of $v$. Then, two things are done: first, $v$ is refined with one Newton step. Second, the calculated control is executed until the next prediction. In the following prediction, the process repeats with the new value of $v$ and the current state of the system. This iteration is repeated until the completion of the trajectory. The predictions can occur at the same rate as the trajectory control or at a lower rate. Since the Newton step guarantees that the predicted error is strictly decreasing, it is possible to show the convergence to the desired value of energy $\tilde{J}_T$.

The process is first detailed in the joint state and then in the task space.

### 4.2.1 Trajectory Tracking in the Joint State

To describe this procedure analytically, it is convenient to consider the system discretized in time. To this end, the robot manipulator system (2.4) is discretized with ZOH (Zero-Order Hold) ([14]) assumption with sampling time $\Delta t$:

$$M(q_k)\ \ddot{q}_k + C(q_k, \dot{q}_k)\ \dot{q}_k + G(q_k) = \tau_k \tag{4.1}$$

where $q_k \coloneqq q(k\Delta t)$ for $k = 0, 1, 2, ..., K$ where $T = K\Delta t$.

Then, at current time $k$, and for a given control $v_k$, the total energy is calculated until $k + m$, where $m$ is the number of steps remaining to complete the trajectory, i.e. the horizon length. The energy function (3.19) considering $f(s) = f_q(s)$ is discretized as:

$$\tilde{J}_T(v_k) = \sum_{i=k}^{k+m} |\nabla f_q(s_i)|^T \ |\big(\nabla^2 f_q(s_i)\dot{s}_i^2 + \nabla f_q(s_i)\ddot{s}_i\big)| \tag{4.2}$$

where $z_i = [s_i\ \dot{s}_i\ \ddot{s}_i]$ is calculated from a discretized linear system (3.24).

The energy error at current time $k$ is given by:

$$\tilde{e}_k = \tilde{J}_T(v_k) - \tilde{J}_{Bk} \tag{4.3}$$

where $\tilde{J}_{Bk}$ is the available energy budget at current time $k$. Then it is applied to the system the computed torque $\tau_k$ with the feedforward term given by:

$$u_k = \ddot{q}_{dk} = \nabla^2 f_q(s_k)\ \dot{s}_k^2 + \nabla f_q(s_k)\ \ddot{s}_k \tag{4.4}$$

to drive the system to the next time interval.

The control variable $v$ is updated using:

$$v_{k+1} = v_k - \mu\ \tilde{e}_k \tag{4.5}$$

The proposed receding horizon procedure is described in Algorithm 1.

**Algorithm 1** RHPC Based on Newton Method

1: **while** $s \leq 1$ **do**
2: Calculate $z_i = [s_i\ \dot{s}_i\ \ddot{s}_i]$ for $i = k, ..., k+m$ from (3.24)
3: Calculate $\tilde{J}_T(v_k)$ from (4.2)
4: Calculate energy error $\tilde{e}_k = \tilde{J}_T(v_k) - \tilde{J}_{Bk}$
5: Calculate $u_k$ from (4.4)
6: Apply joint torques: $\tau_k = M(q_k)u_k + C(q_k, \dot{q}_k)\dot{q}_k + G(q_k)$
7: Update control variable: $v_{k+1} = v_k - \mu\ \tilde{e}_k$ from (3.3)
8: Update the energy budget $\tilde{J}_{Bk}$ from the system.
9: **End while**

**Remark 7** *Note that the predictive control loop can be implemented at a slower frequency then the control frequency. In that case, at every prediction the next $w$ torques are calculated: $\tau_{k,w} = \begin{bmatrix}\tau_k & \tau_{k+1} & \cdots & \tau_{k+w-1}\end{bmatrix}$, where $w$ is the the number of intervals between each prediction.*

### 4.2.2 Trajectory Tracking in the Task Space

The system (2.5) is discretized with ZOH assumption with sampling time $\Delta t$:

$$\bar{M}(p_k)\ \ddot{p}_k + \bar{C}(p_k, \dot{p}_k)\ \dot{p}_k\ + \bar{G}(p_k) = F_k \tag{4.6}$$

where $p_k \coloneqq p(k\Delta t)$ and $F_k \coloneqq F(k\Delta t)$ for $k = 0, 1, 2, ..., K$ where $T = K\Delta t$.

Then, following the same steps as in section (4.2.1), the energy function (3.19) considering $f(s) = f_p(s)$ is discretized as:

$$\tilde{\bar{J}}_T(v_k) = \sum_{i=k}^{k+m} |\nabla f_p(s_i)|^T\ |\big(\nabla^2 f_p(s_i)\dot{s}_i^2 + \nabla f_p(s_i)\ddot{s}_i\big)| \tag{4.7}$$

where $z_i = [s_i\ \dot{s}_i\ \ddot{s}_i]$ is calculated from a discretized linear system (3.24).

The energy error at current time $k$ is given by:

$$\tilde{\bar{e}}_k = \tilde{\bar{J}}_T(v_k) - \tilde{J}_{Bk} \tag{4.8}$$

where $\tilde{J}_{Bk}$ is the available energy budget at current time $k$. Then it is applied to the system the control $u_k$

$$u_k = \dot{p}_{dk} + K_p(p_{dk} - p_k) \tag{4.9}$$

to drive the system to the next time interval. Where $K_p$ is the positive proportional position gain.

The control variable $v$ is updated using:

$$v_{k+1} = v_k - \mu \, \tilde{\bar{e}}_k \tag{4.10}$$

The proposed receding horizon procedure is described in Algorithm 2.

**Algorithm 2** RHPC Based on Newton Method

1: **while** $s \leq 1$ **do**
2: Calculate $z_i = [s_i \ \dot{s}_i \ \ddot{s}_i]$ for $i = k, ..., k+m$ from (3.24)
3: Calculate $\tilde{\bar{J}}_T(v_k)$ from (4.7)
4: Calculate energy error $\tilde{\bar{e}}_k = \tilde{\bar{J}}_T(v_k) - \tilde{J}_{Bk}$
5: Apply the control $u_k$ from (4.9)
6: Update control variable: $v_{k+1} = v_k - \mu \, \tilde{\bar{e}}_k$
7: Update the energy budget $\tilde{J}_{Bk}$ from the system.
8: **End while**

The block diagram in figure 4.1 summarizes the control scheme considering the model-based linearization and the predictive control.

The robot is controlled with model based controller compensation (the hat indicates the model of the robot parameters). The desired tracking values for joints position ($q_d$), joints velocity ($\dot{q}_d$) and joints acceleration ($\ddot{q}_d$) are obtained from the predictive controller through the path parameterization. The parameterized trajectory is also used to calculate the energy prediction. The energy budget $\tilde{J}_B$ is obtained from the system and accounts not only for the robot energy consumption but for energy losses and other unmodelled consumption.

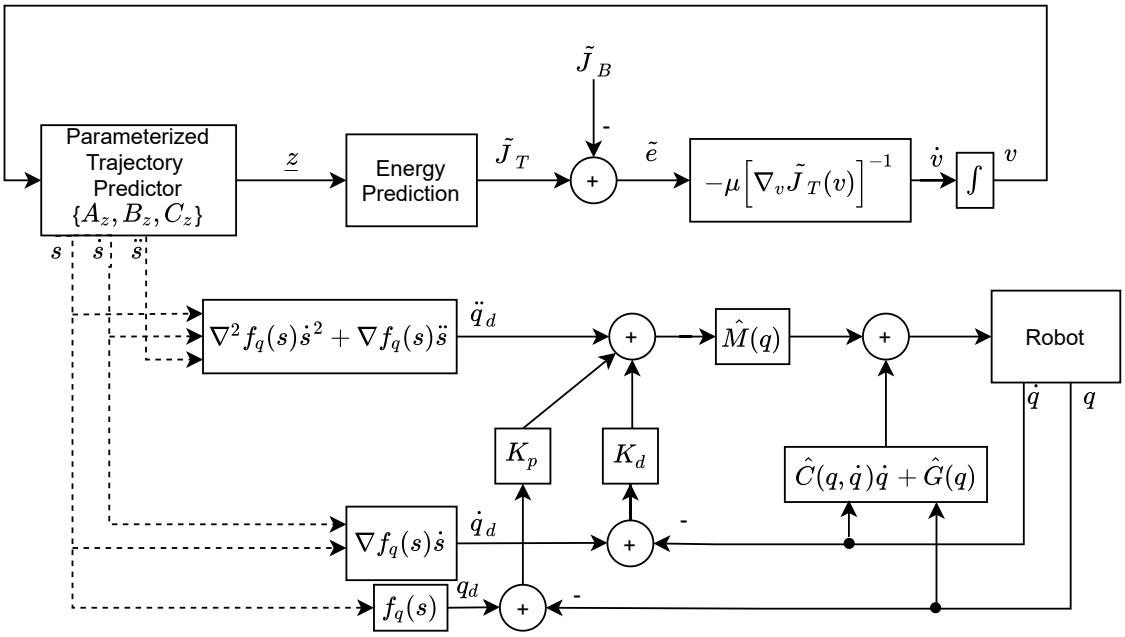


Figure 4.1: Control scheme block diagram

## 4.3 Numerical Examples

### 4.3.1 RR Planar Manipulator

The proposed method is verified with a two link planar manipulator where the manipulator parameters are the same as described in (Siciliano et al. [39]) example 7.2: $a_1 = a_2 = 1m, l_1 = l_2 = 0.5m, m_{l1} = m_{l2} = 50kg, I_{l1} = I_{l2} = 10kgm^2, k_{r1} = k_{r2} = 1, m_{m1} = m_{m2} = 5kg, I_{m1} = I_{m2} = 0.01kgm^2$, where $a_i$ is the link length, $l_i$ is the distance of center of mass to joint axis, $m_{li}$ is the link mass, $m_{mi}$ is the mass of the joint actuator, $k_{ri}$ is the actuator gearbox reduction ratio, $I_{mi}$ is the moment of inertia with respect to the the axis of the actuator and $I_{li}$ is the moment of inertia with respect to the center of mass of the link.

#### Description of the Path

The desired path is considered 10 turns around a circle with radius equal to 1 in $q_1 - q_2$ plane which is given by the following map:

$$q_d(s) = f_q(s) = \begin{bmatrix} \sin(20\pi s) \\ \cos(20\pi s) \end{bmatrix} \qquad s \in [0, 1]$$

#### Description of the Simulation with RR Planar Manipulator

The discretization time period is $\Delta t = 0.01s$, and $\mu = 10^{-5}$. The predictions are also calculated every $0.01s$.

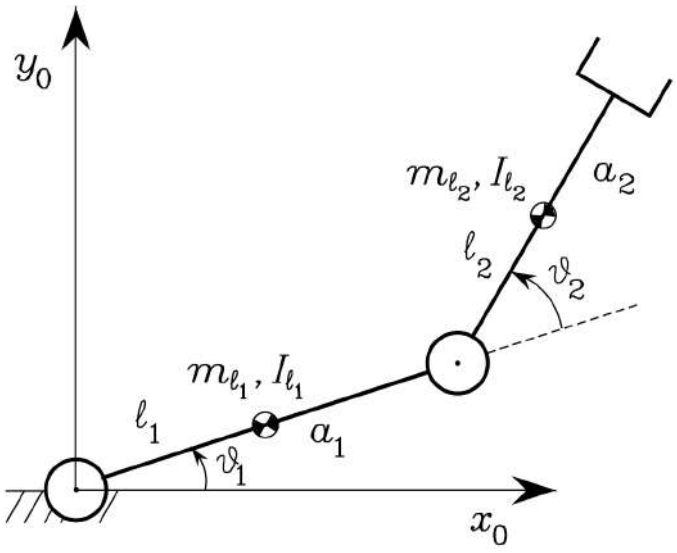


Figure 4.2: Manipulator RR planar manipulator - Ref: (Siciliano et al. [39])

In order to guarantee path tracking performance, position and velocity feedback terms were included in (3.8):

$$u = \ddot{q}_d + K_p\ (q_d - q) + K_d\ (\dot{q}_d - \dot{q}) \tag{4.11}$$

where $K_p = 500$, $K_d = 200$ and $q_d, \dot{q}_d, \ddot{q}_d$ from (2.18).

The $z$-dynamic system (3.24) is given by

$$A_z = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -3 \cdot 10^{-4} & -1 & -2 \end{bmatrix}; B_z = \begin{bmatrix} 0 \\ 0 \\ 3 \cdot 10^{-4} \end{bmatrix}$$

Note that this system has DC gain equal to 1.

The energy budget for the linearized system is $\tilde{J}_B = 2\ 10^5$. The system was simulated with two different initial condition: $v(0) = 200$ and $v(0) = 10$.

In order to verify the controller robustness, the simulation considers an extra $10kg$ inertia and $12.2kgm^2$ moment of inertia on link 2. It also considers a disturb on the energy budget. The disturb is a 20% drop on the budget on instant $s = 0.3$.

The algorithm (1) is implemented and robot dynamic (2.4) is integrated with a $\Delta t$ period using Matlab ode45 function.

### RR Planar Manipulator Simulation Results

The evolution of the energy budget along the iterations is shown in Figures 4.3 and 4.4 together with the control variable $v$. The initial condition $v(0) = 200$ is higher than the optimal value. And $v(0) = 10$ is lower than the optimal value.

For both initial conditions $v(0) = 200$ and $v(0) = 10$, the proposed algorithm

converges in less than $200ms$ and then $v$ stays at $v = 42.48$ until the disturb at $s = 0.3$. The disturb is compensated in also less than $200ms$ and then $v$ stays at $v = 33.8$ during the remaining of the trajectory.

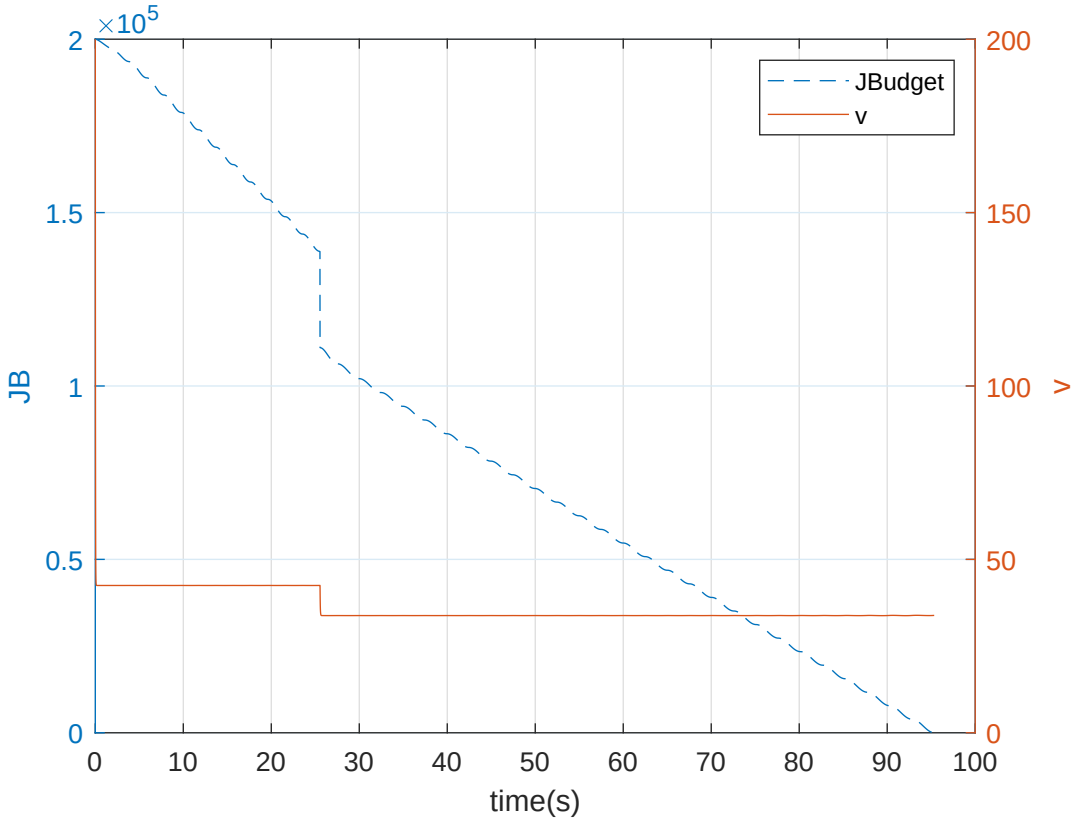


Figure 4.3: Simulation Results Predictive Control with Newton Method - RR Planar Manipulator: Budget $\tilde{J}_B(k)$ and control variable $v$ with $v(0) = 200$

The joint position, position error and the joint torques are shown in figures 4.5 and 4.6.

Initially, the error is $\pm 0.23deg$ for $q_1$ and $\pm 0.32deg$ for $q_2$. Then, after the energy drops at $s = 0.3$ the algorithm reduces the speed to adapt to the new budget and consequently the error drops to $\pm 0.18deg$ for $q_1$ and $\pm 0.24deg$ for $q_2$. After the disturb in the energy budget the torques also decrease. The simulations with both initial condition shows the same performance.

### 4.3.2 Tetis Manipulator

A numerical simulation is carried out with a light weight 4R manipulator named Tetis. A simplified dynamic model of the manipulator is considered with a payload of $200g$. This payload is not canceled with the computed torque.

In order to observe the optimization method proposed, two simulations were performed with two different levels of energy budget.

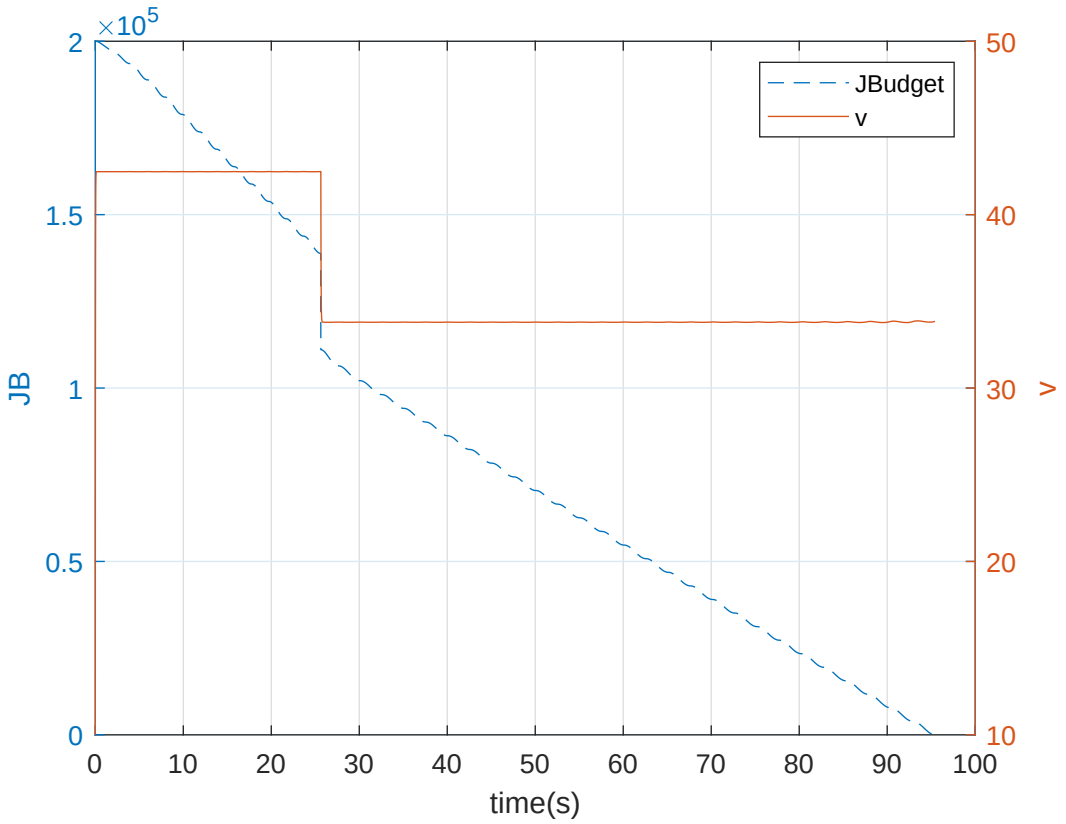


Figure 4.4: Simulation Results Predictive Control with Newton Method - RR Planar Manipulator: Budget $\tilde{J}_B(k)$ and control variable $v$ with $v(0) = 10$

### Description of the Manipulator

Tetis is part of the mobile robot Doris. Doris is a rail guided robot used for offshore inspections in oil and gas facilities (Galassi et al. [15]).

Since it is part of a mobile robot, Tetis is a light weight manipulator. Its links are made of carbon fiber tubes and the joints made of 3-D printed Ti64 alloy (Xaud [49]).

The actuators are from Harmonic Drive Ag Mini servo actuator line. And the drivers are model EPOS2 70/10 from Maxon Motor. There is an embedded computer running a ROS network. This embedded computer reads the sensors available on Doris. And through a Controller Area Network (CAN), controls the motor drives. See the hardware architecture diagram in figure 4.9.

Tetis weights 2.5Kg and has a payload of 0.3Kg. Link 1 is 52.5mm in length, link 2 is 320mm in length, link 3 is 225mm in length and link 4 is 167.25mm in length.

Figure 4.7 shows a picture of Tetis and figure 4.8 shows Tetis coordinate systems.

The dynamic model of Tetis used in these simulations is described in appendix B.

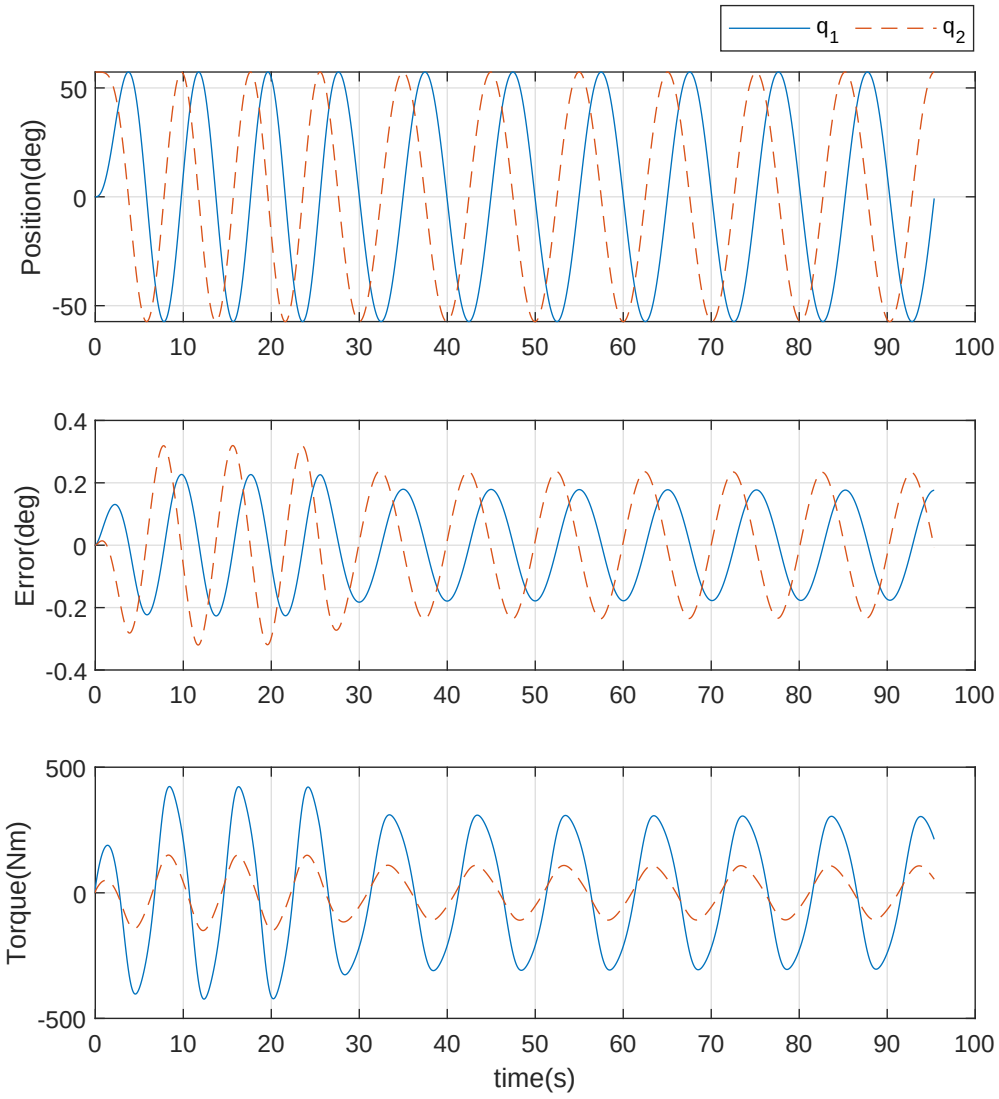


Figure 4.5: Simulation Result Predictive Control With Newton Method - RR Planar: Positions $q$, Position Error $q - q_d$ and Torque $\tau$ with $v(0) = 200$

### Description of the path

The path in the task space is described as two turns around a 50 mm radius circle in $X - Z$ plane. The circle is centered in position $(x, yz) = (500, 57, -67)$. To obtain the path in joint space, inverse kinematics is used. Since the path is fixed on $X - Z$ plane, $\theta_1$ is fixed. Then, the joint positions $\theta_2$, $\theta_3$ and $\theta_4$ are modeled with a Fourier series:

$$\theta_N(t) = a_0 + \sum_{i=1}^{n} a_i cos(i\omega t) + b_i sin(i\omega t)$$

where $N$ is the joint number, $n$ is the number of terms in the Fourier expansion and $\omega$ is the frequency. The Matlab Curve Fitting Tool Box is used to find the coefficients and calculate the metrics. An expansion with $n = 2$ already yields a

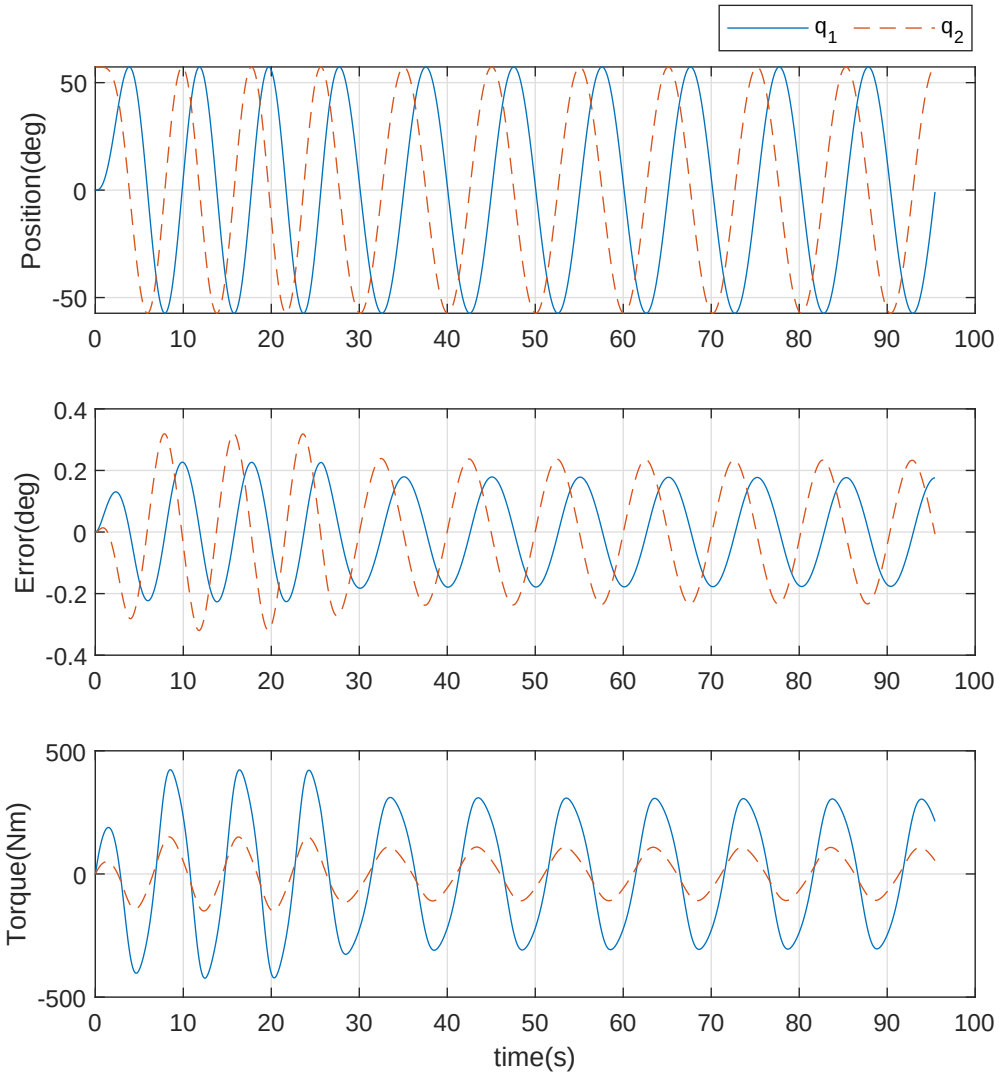


Figure 4.6: Simulation Result Predictive Control With Newton Method - RR Planar: Positions $q$, Position Error $q - q_d$ and Torque $\tau$ with $v(0) = 10$

good result.

Table 4.1 shows the Fourier coefficients and table 4.2 shows the statistics metrics of the model for each joint. RMSE is the root mean squared error and R-Square is the coefficient of determination. The R-Square ranges from 0 to 1 where 1 means a perfect fit.

Table 4.1: Curve Fitting of the Path used in Tetis Numerical Simulation - Model Coefficient (with 95% confidence bounds)

| $N$ | $a_0$ | $a_1$ | $b_1$ | $a_2$ | $b_2$ |
|---|---|---|---|---|---|
| 2 | $-0.8746$ | 0.1957 | 0.1441 | 0.003245 | $-0.02521$ |
| 3 | 1.804 | 0.07579 | $-0.3625$ | $-0.007746$ | $-0.003591$ |
| 4 | $-0.9298$ | $-0.2717$ | 0.2181 | 0.004388 | 0.02882 |

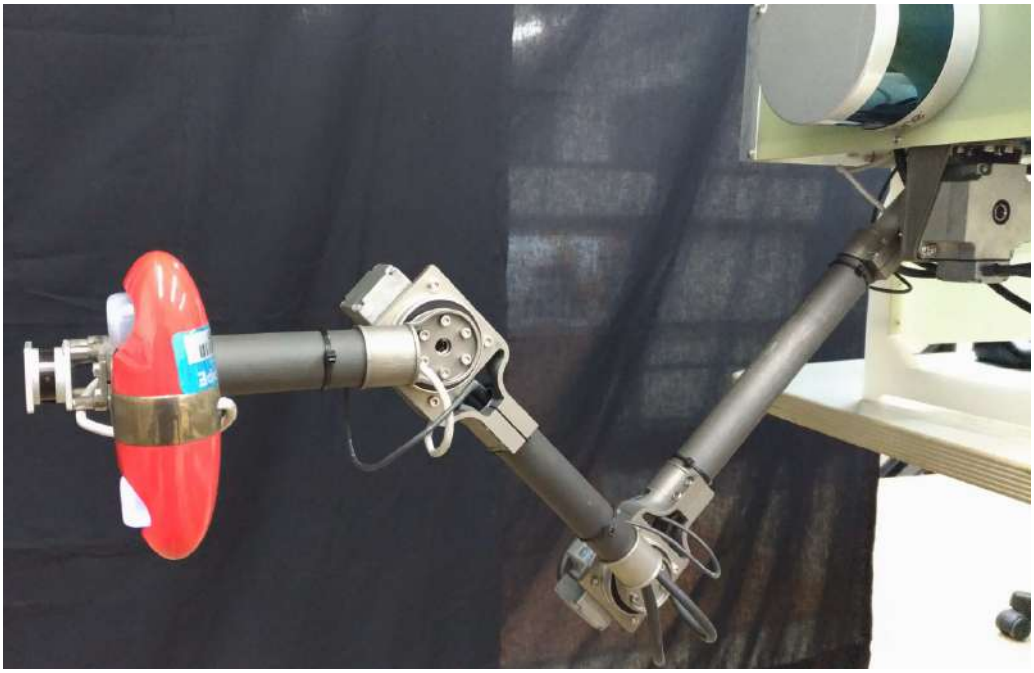

Figure 4.7: Tetis 4R Light Manipulator

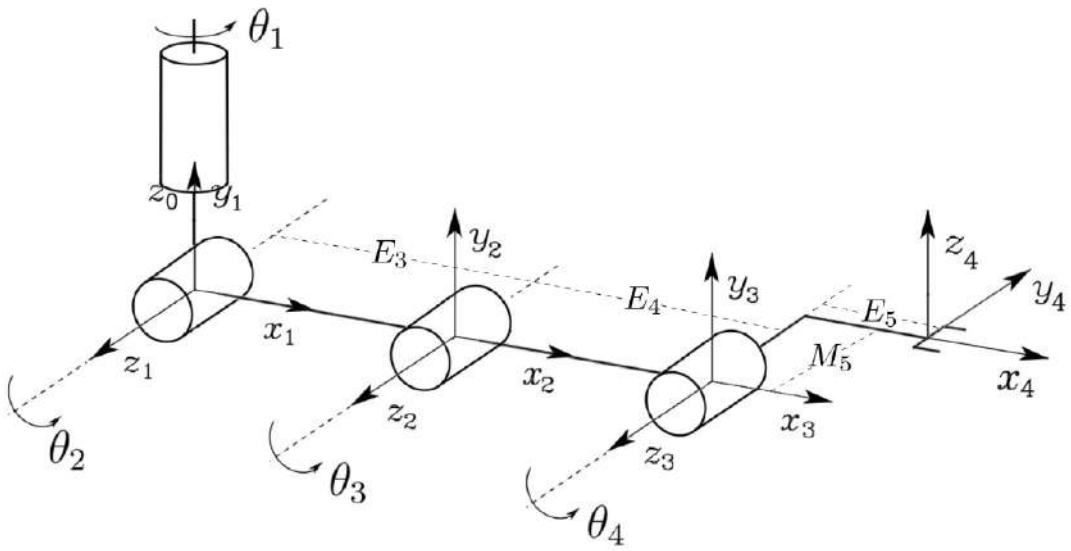


Figure 4.8: Tetis Reference System - Diagram extracted from (Silva [40])

### Description of the Simulation with Tetis

In order to verify robustness of the control used in the simulation, a payload of $200g$ is considered in Tetis end-effector.

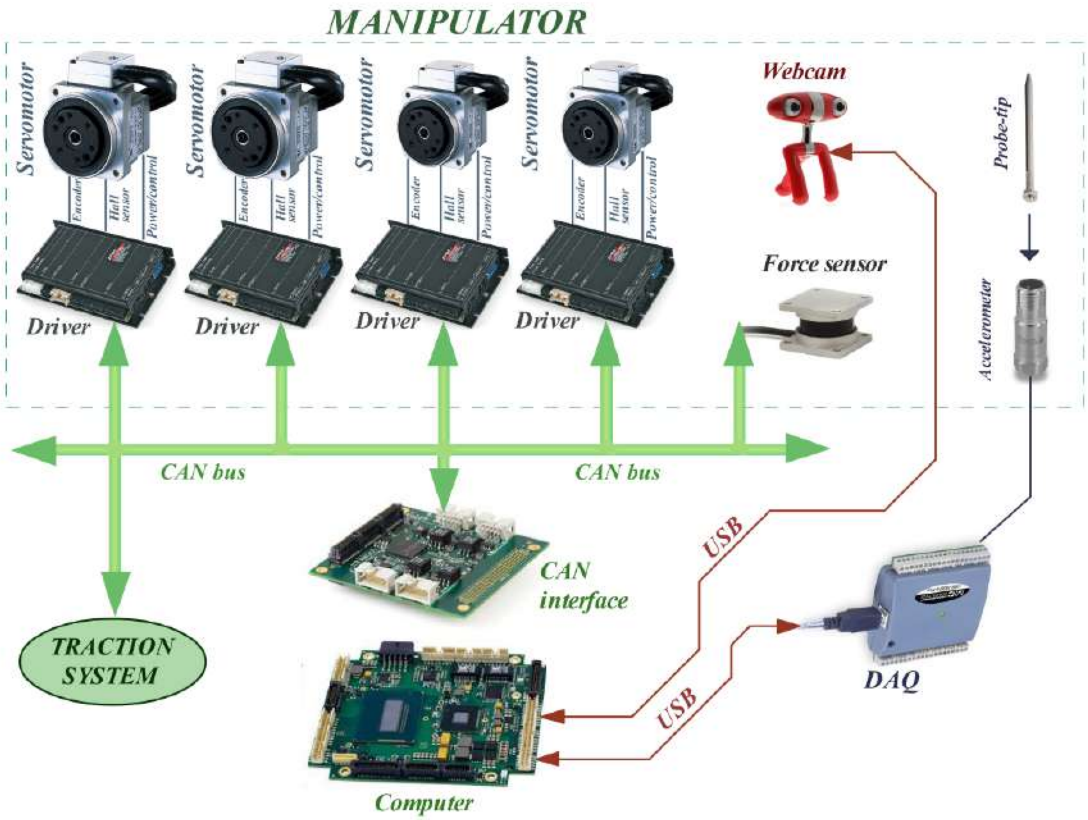


Figure 4.9: Tetis Hardware Architecture - Diagram extracted from (Xaud [49])

Table 4.2: Curve Fitting of the Path used in Tetis Numerical Simulation - Model Fit Metrics

| $N$ | $R-Square$ | $RMSE$ |
|---|---|---|
| 2 | 0.9998 | 0.002688 |
| 3 | 1 | 0.001796 |
| 4 | 0.9998 | 0.003622 |

The discretization time period used is also $\Delta t = 0.01s$, and $\mu = 10^{-5}$. The predictions are also calculated every $0.01s$.

As in the previous simulation, position and velocity feedback terms are included to guarantee path tracking performance in (3.8). And $K_p = 5000$, $K_d = 2000$ and $q_d, \dot{q}_d, \ddot{q}_d$ from (2.18).

The simulation is carried out with the following linear system:

$$A_z = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -3 \cdot 10^{-2} & -1 & -2 \end{bmatrix}; B_z = \begin{bmatrix} 0 \\ 0 \\ 3 \cdot 10^{-2} \end{bmatrix}$$

The algorithm (1) is implemented and robot dynamic (2.4) is integrated with a $\Delta t$ period using Matlab ode45 function.

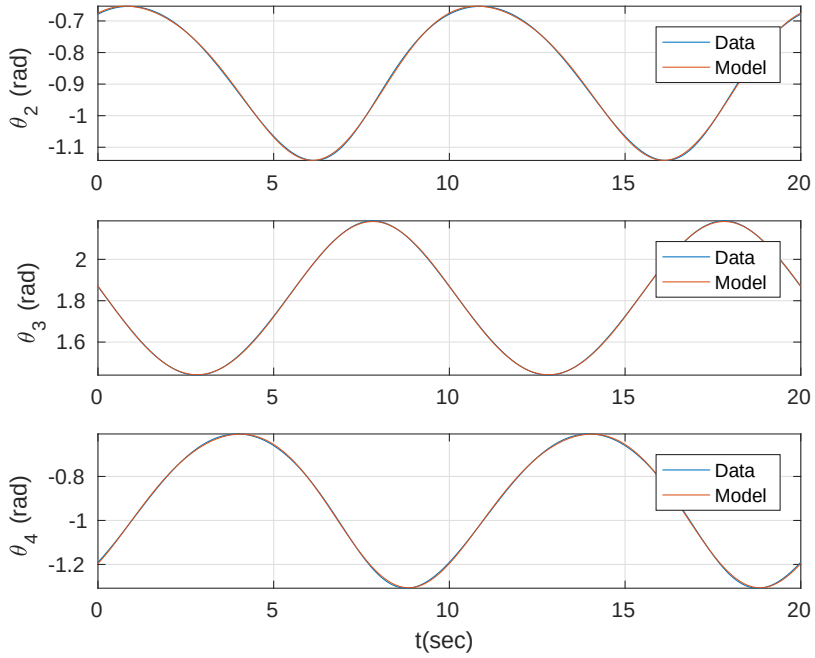


Figure 4.10: Curve Fitting of the Path used in Tetis Numerical Simulation - Joint Positions Data and Model

To observe the behavior of the optimization, two simulations are performed. One with $\tilde{J}_B = 2600$ and the other with $\tilde{J}_B = 3200$. The former is less energy then the initial $v = 5$ requires and the latter is more energy.

### Tetis Simulation Results

Figure 4.11 shows the case with less energy then required for the initial $v$. The $v$ is corrected in around $1s$. The speed of convergence is ruled by $\mu$. Higher $\mu$ yields faster convergences. As expected, the energy is entirely consumed during the trajectory.

Figure 4.12 shows the case with more energy then required for the initial $v$. The $v$ is corrected in less then $1s$. Again, the energy finishes only at the end of the trajectory.

Figures 4.13 and 4.14 show the joint positions, the joint positions error and the torques during the trajectory. Joint 1 was omitted because it is not used for the given path.

The position error is around $0.2deg$ in both cases. In the simulation with less energy the position errors are slightly smaller. That is related to the fact that the speeds are also slightly smaller. Note that the completion time is $12s$ for the system with less energy and $9s$ for the system with more energy.

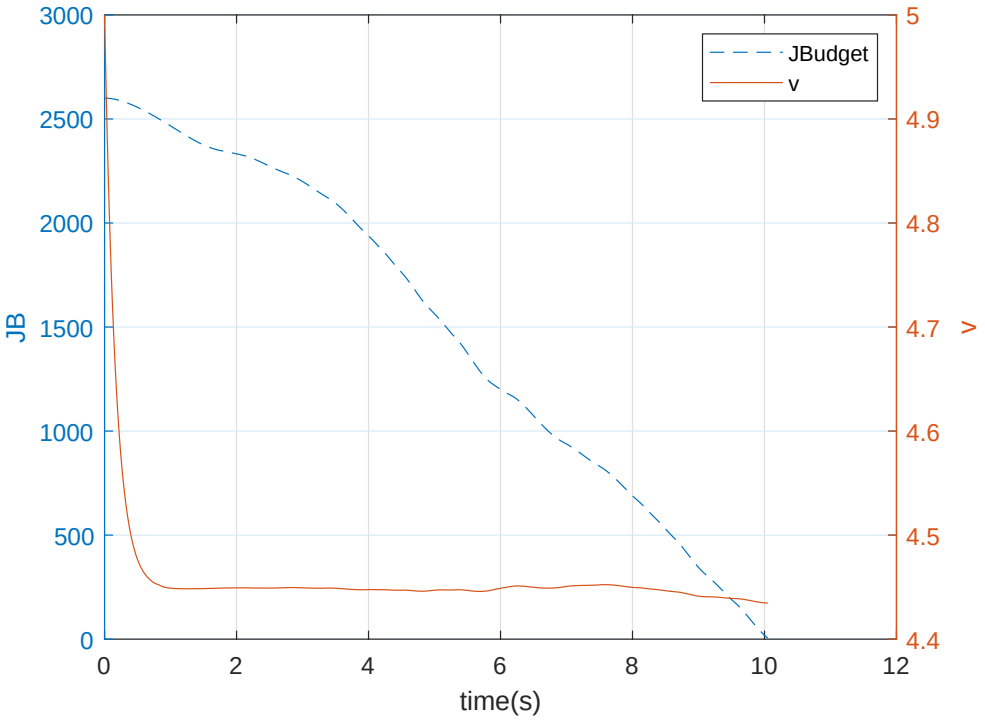


Figure 4.11: Simulation Result Predictive Control With Newton Method - Tetis: Budget $\tilde{J}_B(k)$ starting in 2600 and control variable $v$

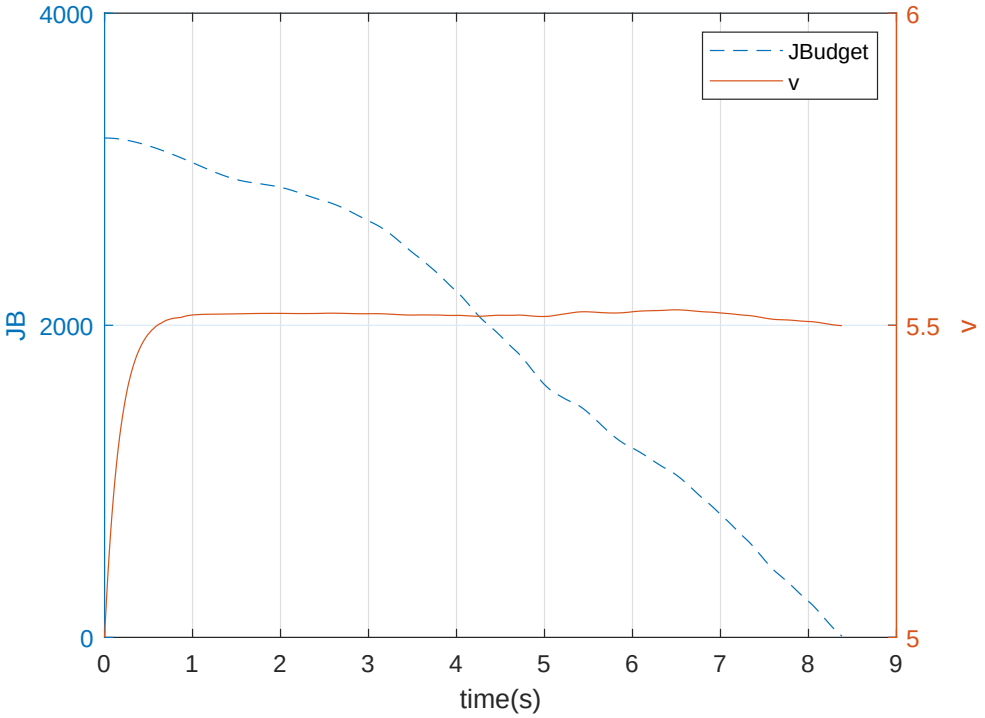


Figure 4.12: Simulation Result Predictive Control With Newton Method - Tetis: Budget $\tilde{J}_B(k)$ starting in 3200 and control variable $v$

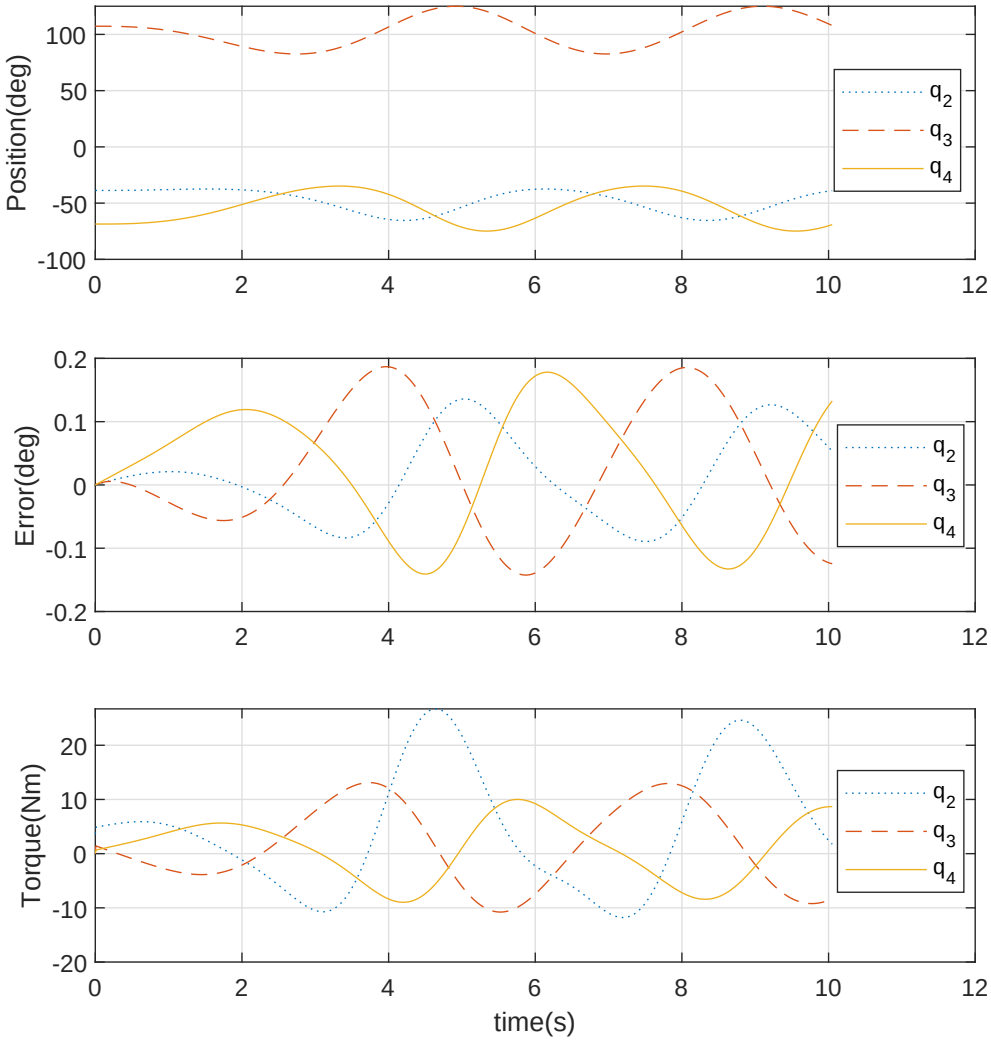


Figure 4.13: Simulation Result Predictive Control With Newton Method - Tetis: Positions $q$, Position Error $q - q_d$ and Torque $\tau$ for $\tilde{J}_B = 2600$

### 4.3.3 Manipulator Tetis with Actuator Friction

On chapter 5 the method is verified experimentally using Tetis. Due to bandwidth limitations, it is very difficult to control the manipulator with the method proposed here so far by using computed torque and sending torque setpoints to each joint. Besides the bandwidth limitations, in order to use a model based controller the joint frictions would have to be mapped and added to the model. Moreover, Harmonic drive gear boxes have high non-linear internal friction. If this friction is not mapped on the model the path tracking performance is degraded.

A numerical example is done with Tetis considering friction in the actuators. In each manipulator joint it is included a fixed friction loss equivalent to 30% of the rated torque. Since this is not mapped in the model, position errors increase

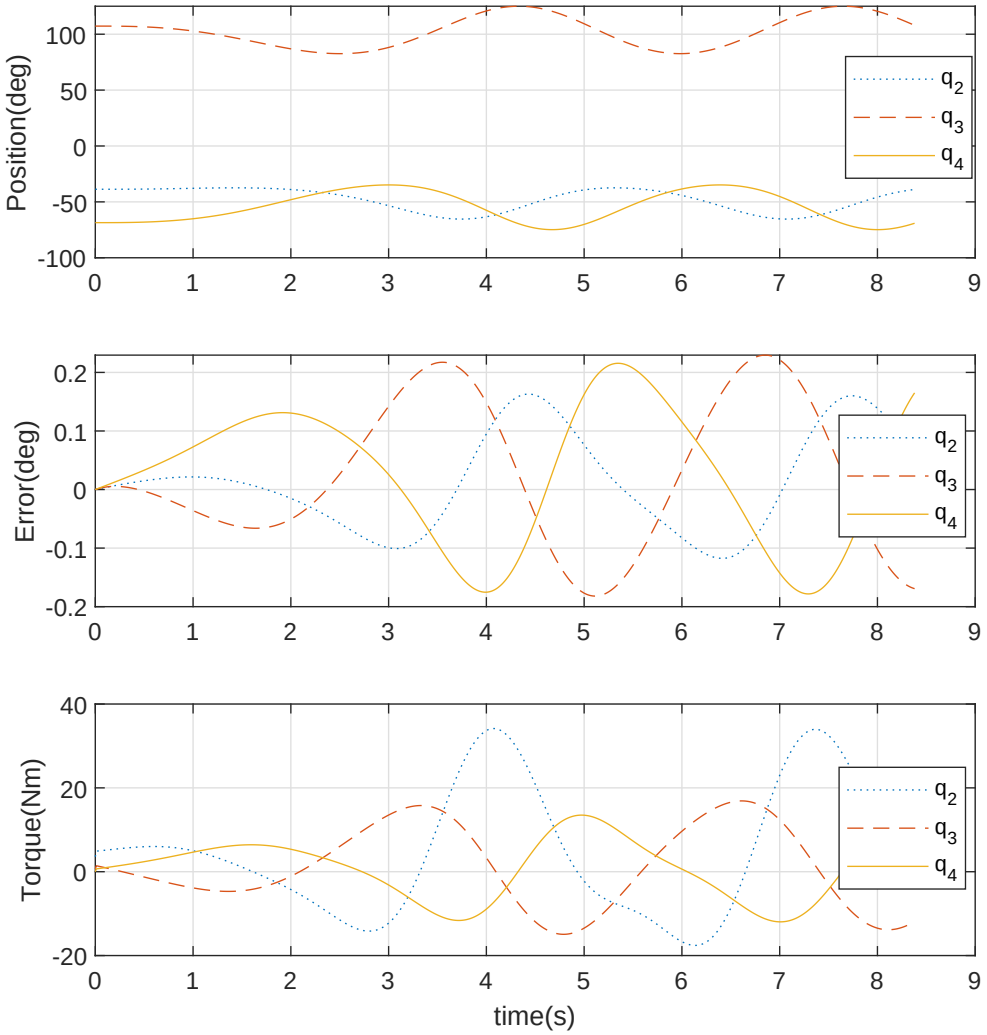


Figure 4.14: Simulation Result Predictive Control With Newton Method - Tetis: Positions $q$, Position Error $q - q_d$ and Torque $\tau$ for $\tilde{J}_B = 3200$

significantly. In figure 4.16 the position errors reach 3 degrees of amplitude. Also, the torque levels on the joints increases significantly. Figure 4.17 shows the degraded path on the $X - Z$ plane.

The energy losses due to the friction on the joints are not accounted on the predictions. But this is absorbed by the Newton steps and the predictive control. In figure 4.18 the energy budget and the $v$ is plotted. The optimization is not affected by the joint frictions.

### 4.3.4 Tetis in Task Space with Kinematic Control

The reality of many industrial manipulators is that the individual joint control is restricted to the internal controllers. And it is only possible to send position and

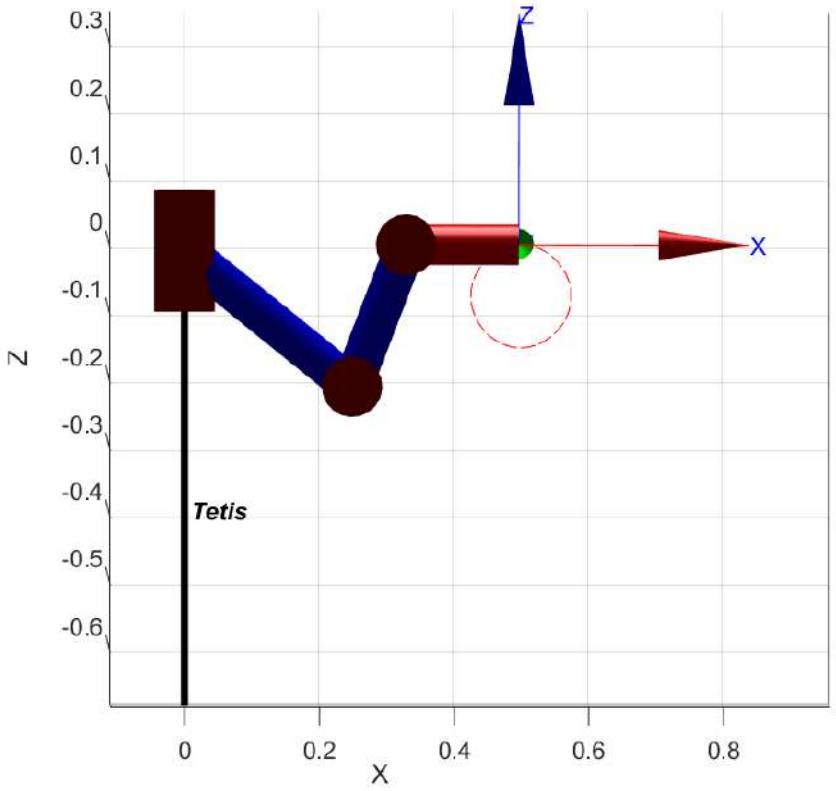


Figure 4.15: Simulation Result Predictive Control With Newton Method - Tetis: Position in the task space

velocity set points in the task space. The manipulator Tetis is an example where direct access to the current loop control of the joints is limited by the bandwidth. So this numerical example illustrates a kinematic control of Tetis with the optimization defined in the task space.

A simulink model is used to represent the kinematic control of Tetis manipulator and the task space formulation. It is worth mentioning that kinematic control assumes perfect velocity tracking and neglect some of the dynamic effects. Also, the dynamic effects are greatly minimized when the joints have high gear ratio, which is the case in Tetis.

Kinematic control means the control input is $u = \dot{q}$ in joint space and $u = \dot{p}$ in the task space. But here a position feedback term is included to guarantee path tracking performance. The following control law then is used

$$u = \dot{p}_d + K\left(p_d - p\right) \tag{4.12}$$

where $K$ is a positive gain.

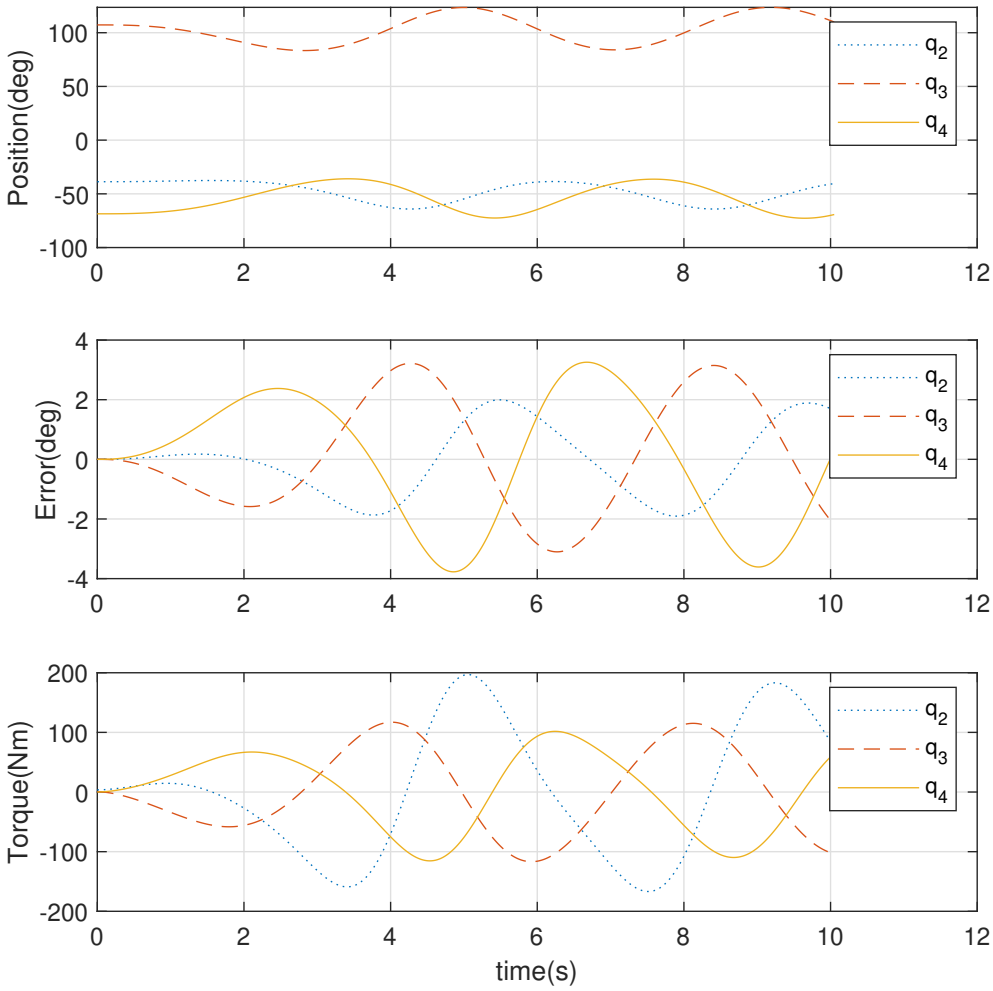


Figure 4.16: Simulation Result Predictive Control With Newton Method - Tetis with unmodelled joint frictions: Positions $q$, Position Error $q - q_d$ and Torque $\tau$

A Matlab Simulink model is used implementing the block diagram described in figure 4.19

### Description of the Path

The path is described by the function below:

$$f_p(s) = \begin{bmatrix} 75\ (\sin(2\pi\ s) + \sin(4\pi\ s)) + 500 \\ 57 \\ 75\ (\cos(2\pi\ s) + \cos(4\pi\ s)) - 67 \\ 0 \end{bmatrix} \qquad s \in [0, 1]$$

and in the workspace is represented in figure 4.20.

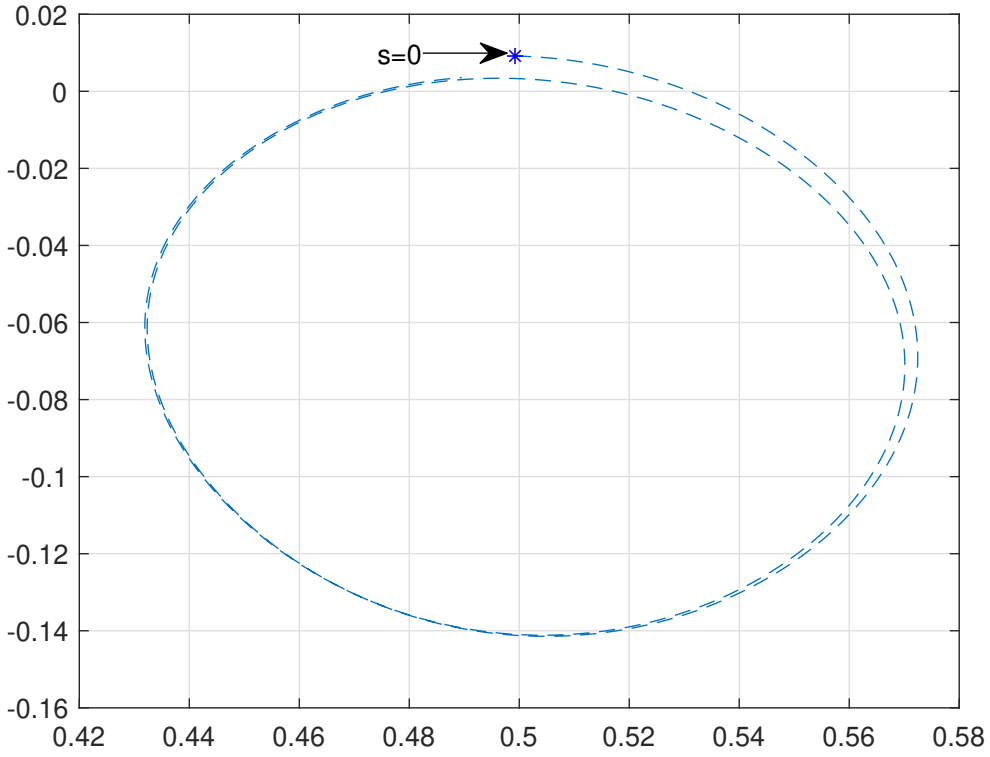


Figure 4.17: Simulation Result Predictive Control With Newton Method - Tetis with Unmodelled Joint Frictions: Task-Space $X$ and $Z$ position

### Description of the simulation with Tetis Kinematic Control

The linear system used is

$$A_z = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -3 \cdot 10^{-3} & -1 & -2 \end{bmatrix}; B_z = \begin{bmatrix} 0 \\ 0 \\ 3 \cdot 10^{-3} \end{bmatrix}$$

The joints initial condition is

$$q(0) = \begin{bmatrix} 0.1073 \\ -0.4427 \\ 1.806 \\ -1.361 \end{bmatrix}$$

The predictive control loop runs at $5Hz$ and kinematic control loop at $50Hz$.

The simulation is performed with the energy budget of $\tilde{J}_B = 2 \times 10^8$. And the initial condition for $v$ is $v(0) = 10$ and then the simulation is repeated for $v(0) = 50$. And $\mu = 5 \times 10^{-8}$.

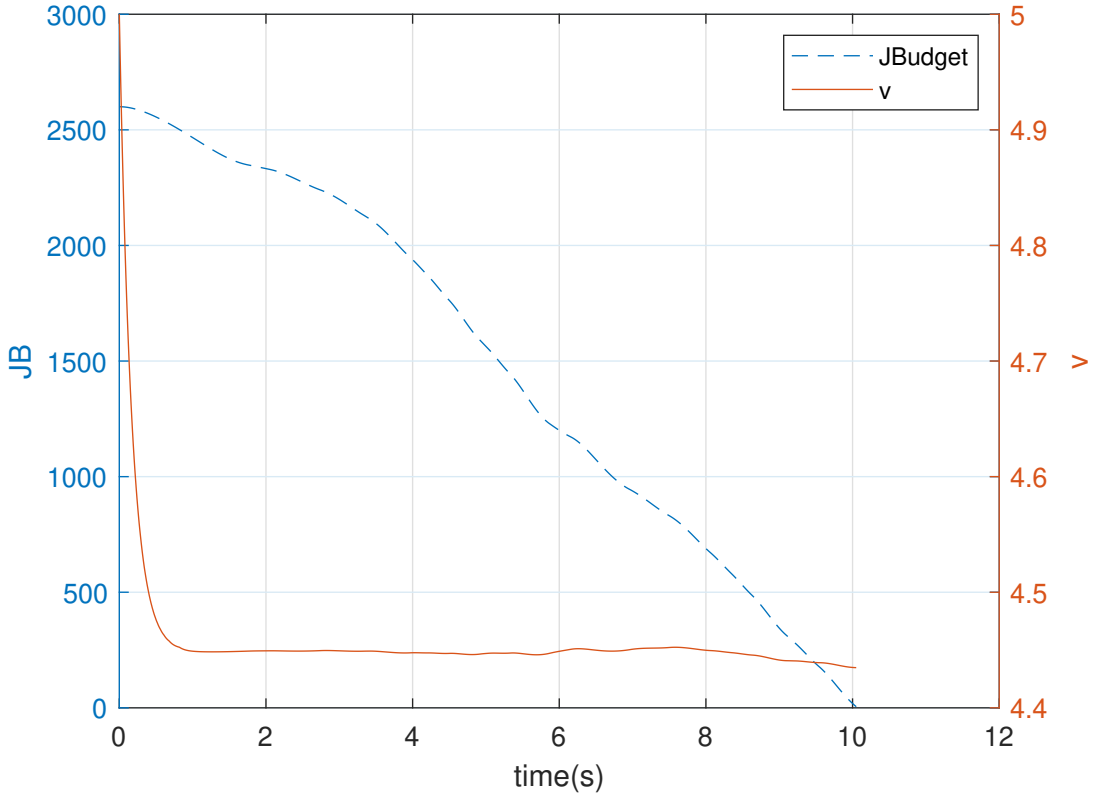


Figure 4.18: Simulation Result Predictive Control With Newton Method - Tetis with Unmodelled Joint Frictions: Budget $\tilde{J}_B(k)$ and control variable $v$

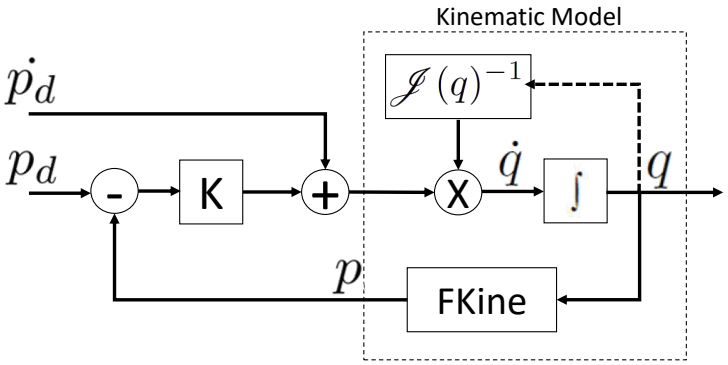


Figure 4.19: Tetis kinematic Model Numerical Simulation

### Tetis Kinematic Control Simulation Results

For both $v(0) = 10$ and $v(0) = 50$ the optimization method defined in the task space corrected the control $v$ for the energy level corresponding to the initial budget. In both cases it takes around 150 iterations or $3s$ to converge. Figures 4.21 and 4.22 shows the control $v$ and the budget level $\tilde{J}_B$ along the iterations. Note that $\tilde{J}_B$ is

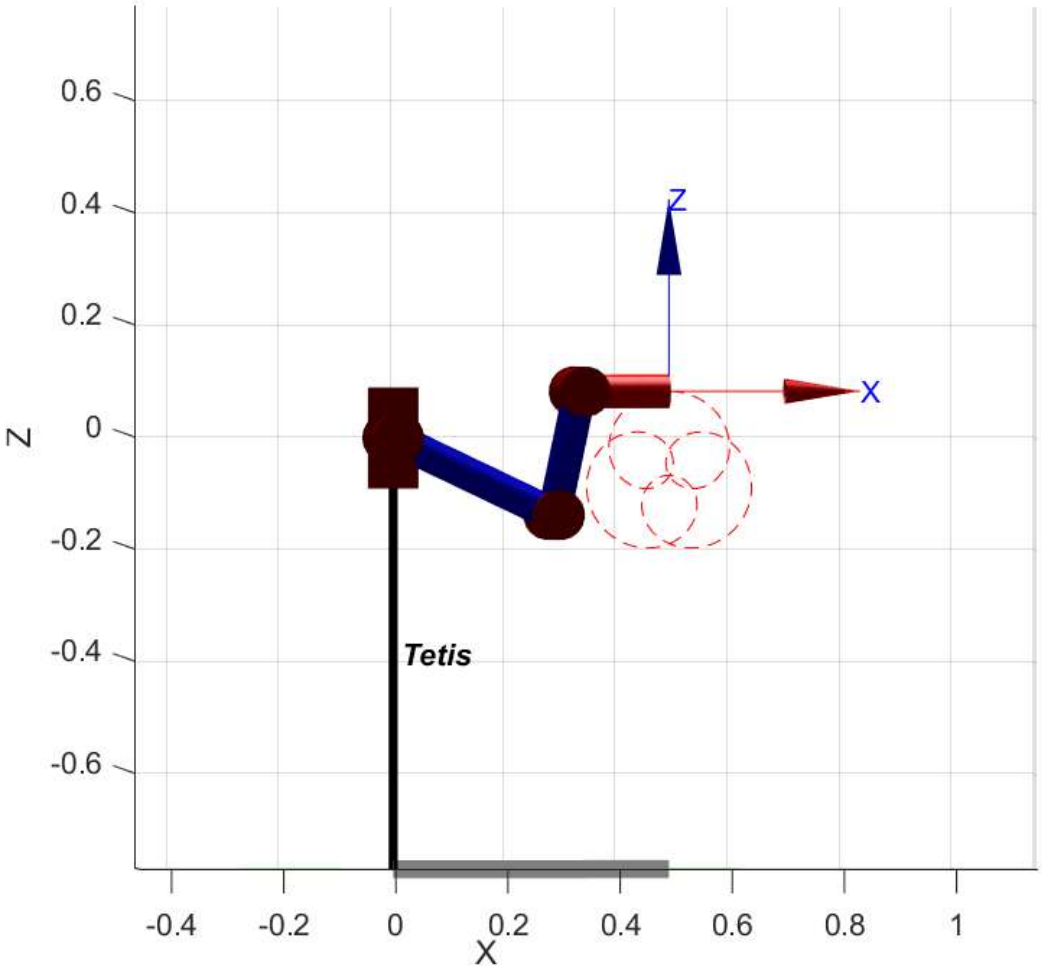


Figure 4.20: Simulation Results - Tetis in the Task Space: The Path

consumed at the kinematic control loop (at $50Hz$) and $v$ is corrected at the predictive control loop (at $5Hz$).

Figures 4.23 and 4.24 shows the joint positions and the position error in the task space respectively. The error is small with around $0.5mm$ amplitude.

Figure 4.25 shows the velocity for each joint for the simulation with $v(0) = 10$. The velocity for joints $q_3$ and $q_4$ have $1rad/s$ amplitude. Joint $q_2$ have around $0.5rad/s$ amplitude and joint $q_1$ around $0.05rad/s$. Joint $q_1$ has such a small velocity because the path is limited to the plane $X - Z$.

In figure 4.26 the $X - Z$ task space position is plotted. The reference path is also plotted (in red).

## 4.4 Conclusions

This chapter shows a modification in the proposed method to render the optimization in a closed loop manner. The technique of Receeding Horizon Predictive Control

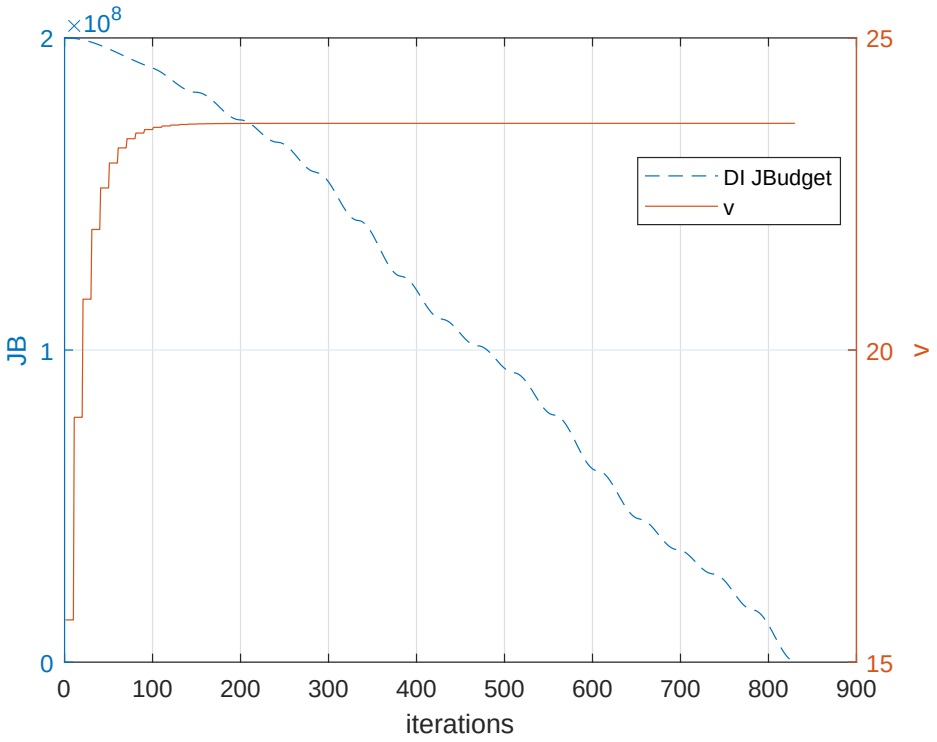


Figure 4.21: Simulation Results - Tetis in the Task Space: Budget $\tilde{J}_B(k)$ and control variable $v$ with $v(0) = 10$

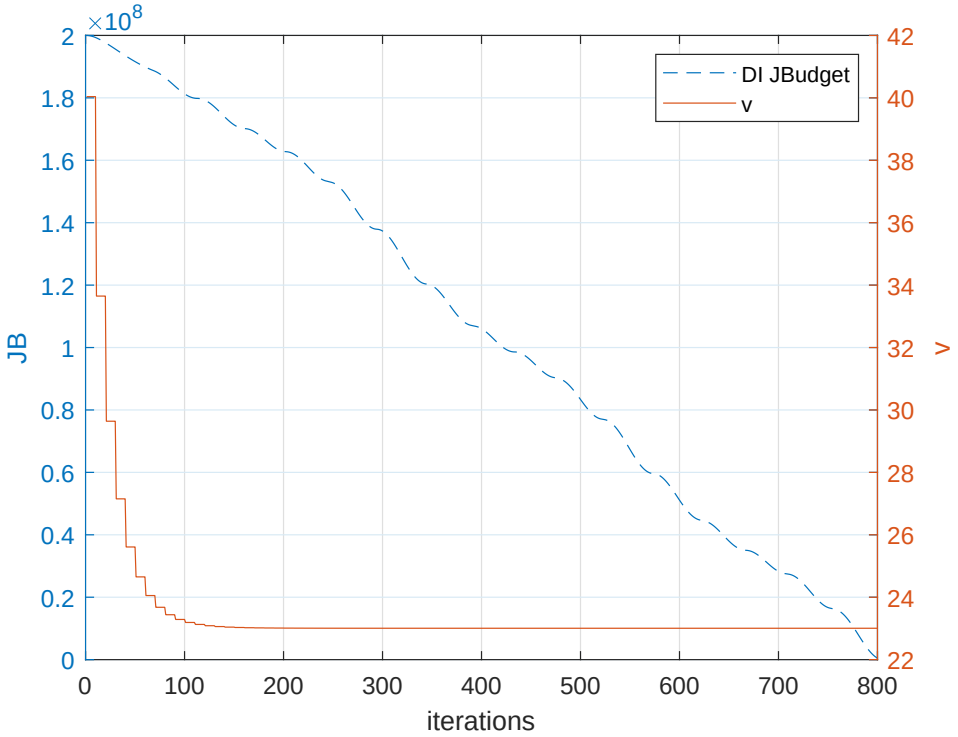


Figure 4.22: Simulation Results - Tetis in the Task Space: Budget $\tilde{J}_B(k)$ and control variable $v$ with $v(0) = 50$

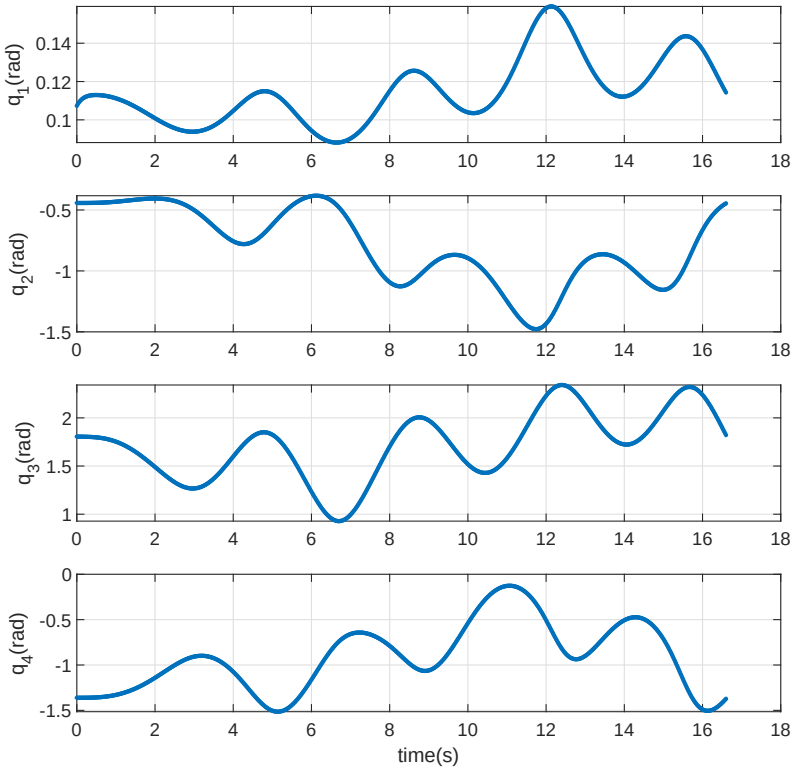


Figure 4.23: Simulation Results - Tetis in the Task Space: Joint Positions for $v(0) = 10$

is used to reach that objective. This results in robustness to disturbances in the energy budget and losses not covered by the energy model.

The optimization method now is suitable to be used in real time application. To illustrate the method, numerical examples are used. First, an example with RR planar manipulator where a drop in the energy budget during the trajectory shows the robustness of the method. Second, the manipulator Tetis is introduced and used in two examples. In the second example with Tetis, the simulation considers unmodelled friction in the joints, which results in large position error and shows that the torque control would be a challenge experimentally. The last numerical example considers the manipulator Tetis controlled with kinematic control in the task space. It is also with kinematic control in the task space that in the next chapter the method is verified experimentally in Tetis.

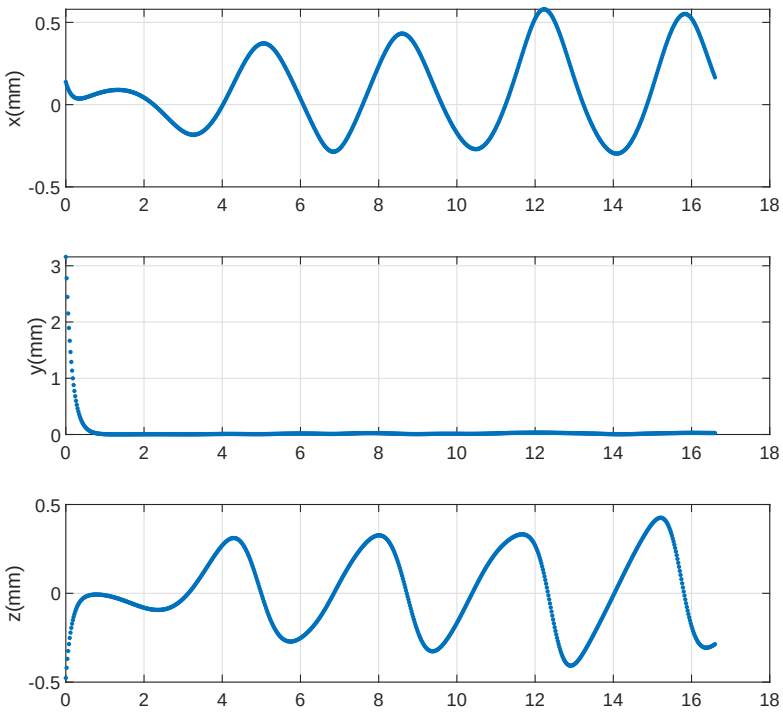


Figure 4.24: Simulation Results - Tetis in the Task Space: Task Space Position Error for $v(0) = 10$

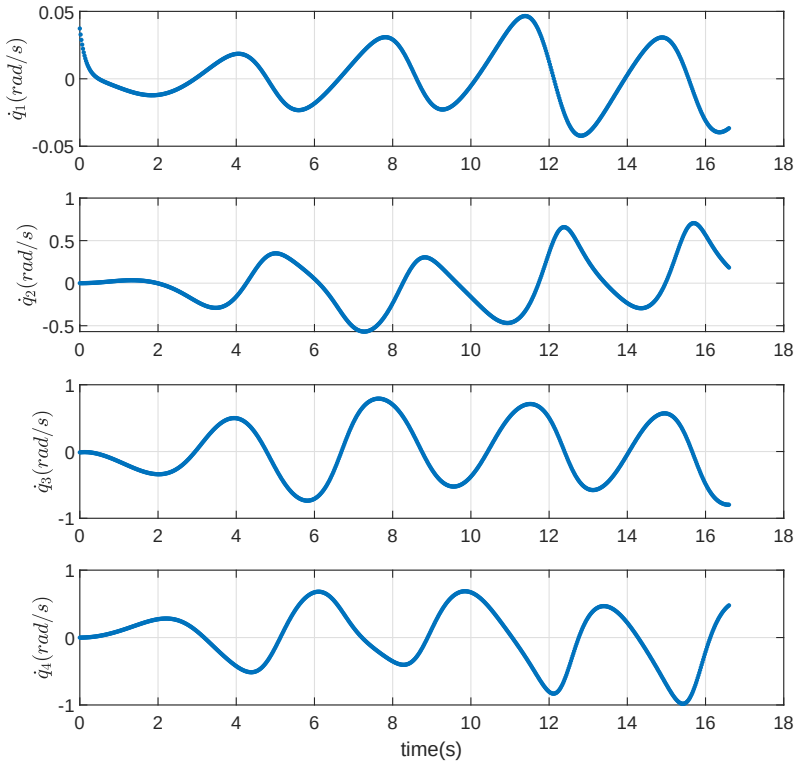


Figure 4.25: Simulation Results - Tetis in the Task Space: Joint Velocities for $v(0) = 10$

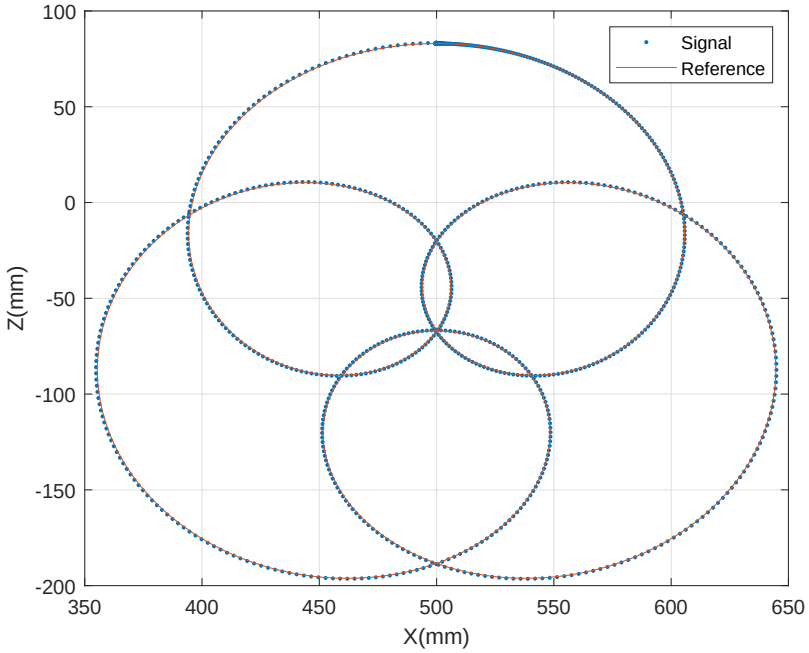


Figure 4.26: Simulation Results - Tetis in the Task Space: Task Space position in $X - Z$ plane for $v(0) = 10$

# Chapter 5

# Experimental Results

So far the problem of energy budget optimization is illustrated with the use of numerical simulations. In these simulations the use of Newton method associated with predictive control in the optimization yield excellent results. This chapter describes the use of the proposed method experimentally using the manipulator Tetis. See section 4.3.2 for details about Tetis.

During the experiment, task space kinematic control is used. This strategy is mainly justified because direct access to the drivers current control loop is impractical due to bandwidth limitation. Besides that, considering the high gear ratio of Tetis actuators, neglecting the dynamics effects poses no compromise to the performance. The numerical simulation described in 4.3.4 uses the same kinematic controller used in the experiment.

In section 5.1 there are considerations about the energy budget estimation. In short, the real energy consumption is estimated with the active current and velocities of each actuator.

In subsection 5.1.1 there are details about the kinematic controller used.

## 5.1 System Description

In order to emulate the energy reservoir consumption, information from the actuators are used to calculate the energy used in the trajectory. Considering that the motor torque is proportional to the active current, the energy consumption is estimated with the following relation

$$J_{Est} = \sum_{k=1}^{K} |i_k|^T |\dot{q}_k| \tag{5.1}$$

where $i_k \in \mathbb{R}^4$ is the current vector at instant $k$. This energy estimation is then compared with the energy calculated for the linearized system (4.7). A trajectories of two turns around a circle as described in section 5.2 is used as a reference to

measure the energy of the linearized system and the estimated energy (5.1). A ratio of approximately 80 was found between estimated energy (in $mA \times mm/s$) and linearized system energy. Figure 5.1 shows the two energies along the reference trajectory.

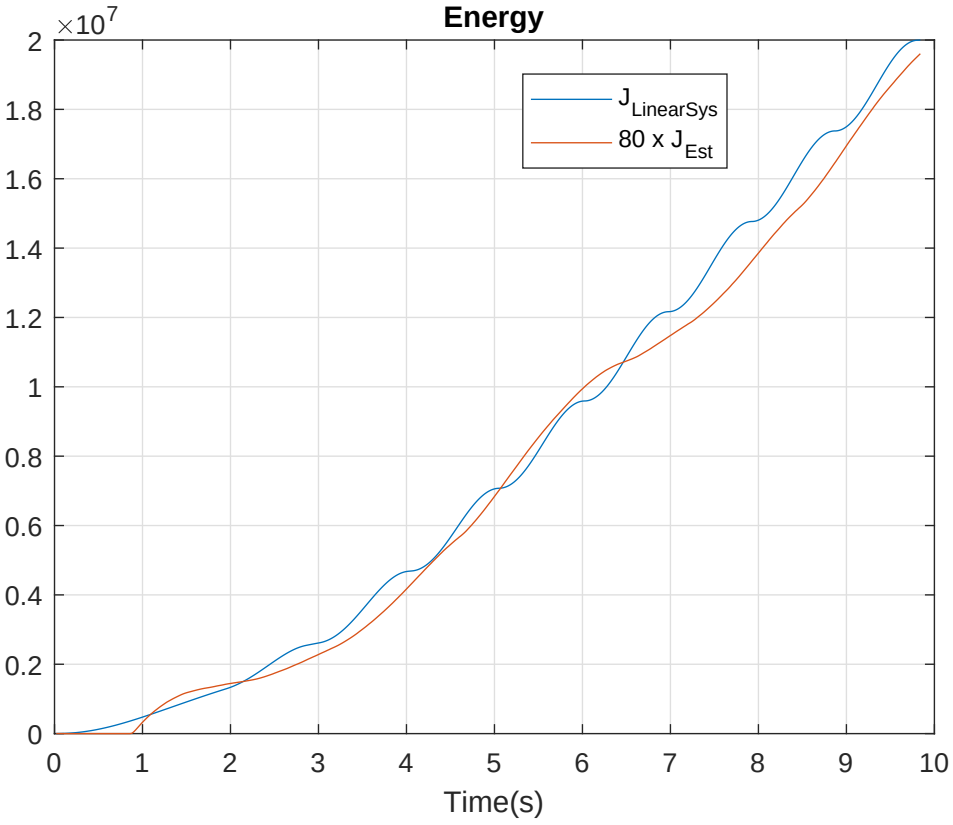


Figure 5.1: Estimated Energy Comparison With Linearized System Energy

For reference, figure 5.2 shows the currents when executing the experiment 5.3 case 1.

### 5.1.1 Description of the Controller

The Tetis manipulator software, as described in section 4.3.2, is based on ROS Framework. And a dedicated ROS node allows the proposed control to be implemented in a Laptop computer.

Here, as in many industrial manipulators, a kinematic control is implemented. Hence, the control signal is (5.2), which is the same approach used in the numerical simulation described in section 4.3.4

$$u = \mathscr{J}(q)^{-1} \left(\dot{p}_d + K_p \left(p_d - p\right)\right) \tag{5.2}$$

where $K$ is a positive constant gain, $p_d$ and $\dot{p}_d$ are calculated according to equation

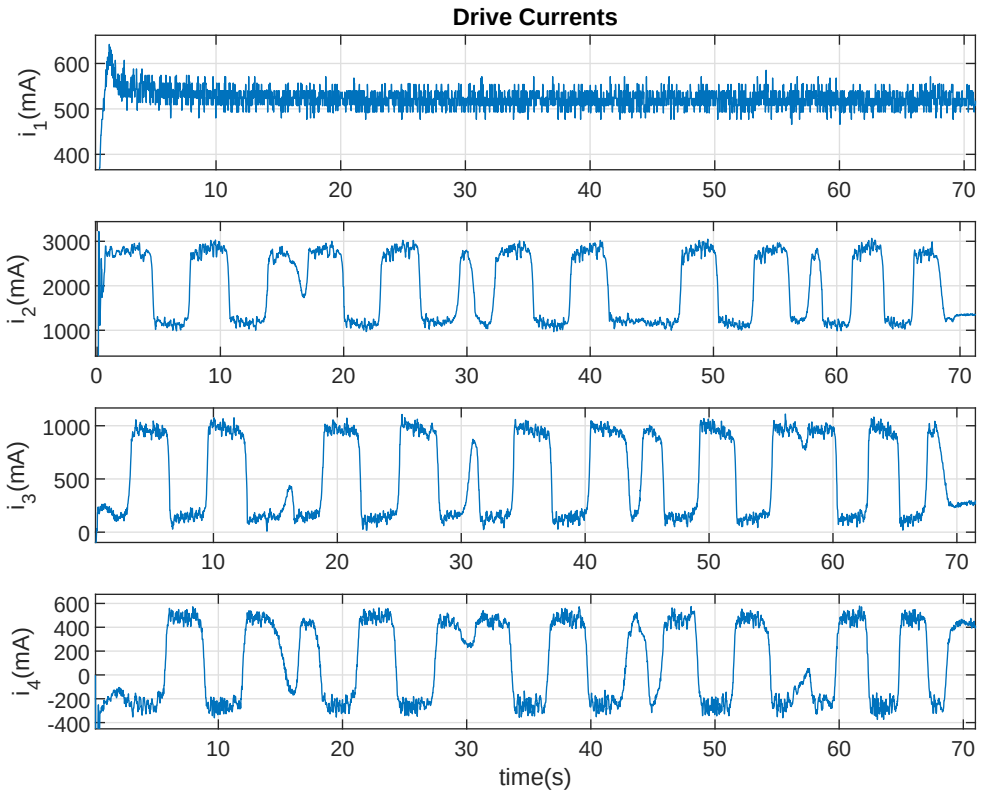


Figure 5.2: Drive Currents Measured During Experiment 2

(2.22). Note that $\mathcal{J}(q)^{-1}$ is calculated on Tetis controller. Figure 5.3 summarizes the control scheme.

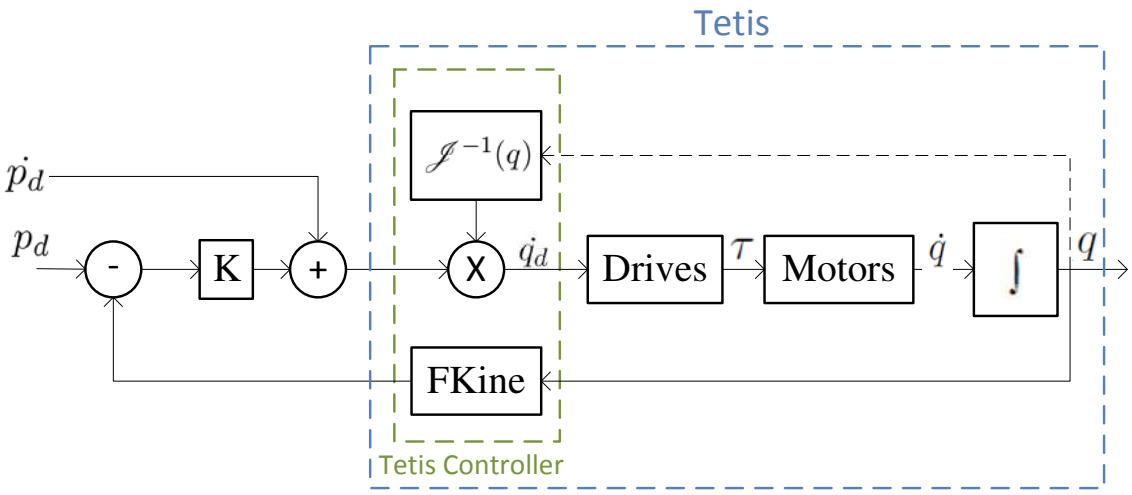


Figure 5.3: Kinematic Control Block Diagram

In the robot embedded system, the velocity commands are sent to the motor driver as joint velocity setpoints.

## 5.2 Experiment 1 - Circle Trajectory

The path is two turns on a 50 mm radius circle in $X-Z$ plane. The path is described by the function $f_p(s)$ below:

$$f_p(s) = \begin{bmatrix} 50\ \sin(4\pi\ s) + 500 \\ 57 \\ 50\ \cos(4\pi\ s) - 67 \\ 0 \end{bmatrix} \qquad s \in [0, 1]$$

The predictive control loop runs at $5Hz$ and the kinematic control loop at $50Hz$.

The dynamic system used to govern $z$ is as stated below:

$$A_z = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -3 \cdot 10^{-5} & -1 & -2 \end{bmatrix}; B_z = \begin{bmatrix} 0 \\ 0 \\ 3 \cdot 10^{-5} \end{bmatrix}$$

This system has DC gain equal to 1 and is discretized with ZOH assumption with $\Delta t = 0.02$. The correction of $v$ on the Newton method is done with $\mu = 10^{-4}$ and the initial value for $v$ is $v(0) = 2000$. The control gain is $K_p = 2$.

The experiment is performed with two different energy budgets: $\tilde{J}_B = 2 \times 10^7$ and $\tilde{J}_B = 3 \times 10^7$.

The position errors are plotted in figures 5.4 and 5.5. For $\tilde{J}_B = 2 \times 10^7$, the error for $x$ and for $z$ have a peak of $5mm$. For $\tilde{J}_B = 3 \times 10^7$, the error for $x$ have a peak of $13mm$ and for $z$ a peak in $10mm$. It is notable that the error increases with the energy budget. That is expected because the control (and consequently the velocities) have higher magnitudes with higher energy budget. Also, there is a trade-off between the position error and the feedback gain $K$.

The control signals are plotted in figures 5.6 and 5.7. The control of $x$ and $z$ are between $\pm 100mm/s$ for $\tilde{J}_B = 2 \times 10^7$ and between $\pm 150mm/s$ for $\tilde{J}_B = 3 \times 10^7$.

The X-Z plane trajectories are on figures 5.8 and 5.9. The manipulator starts outside the path (starting point indicated by arrow). But the control law (5.2) steers it back and keep it on the path. And the extra energy used to that end is handled by the predictive control.

Figures 5.10 and 5.11 shows the evolution of $v$ and $\tilde{J}_B$ for initial condition of $\tilde{J}_B = 2 \times 10^7$ and $\tilde{J}_B = 3 \times 10^7$ respectively.

In both experiments the energy controller regulates $v$ to use all the available energy during the trajectory.

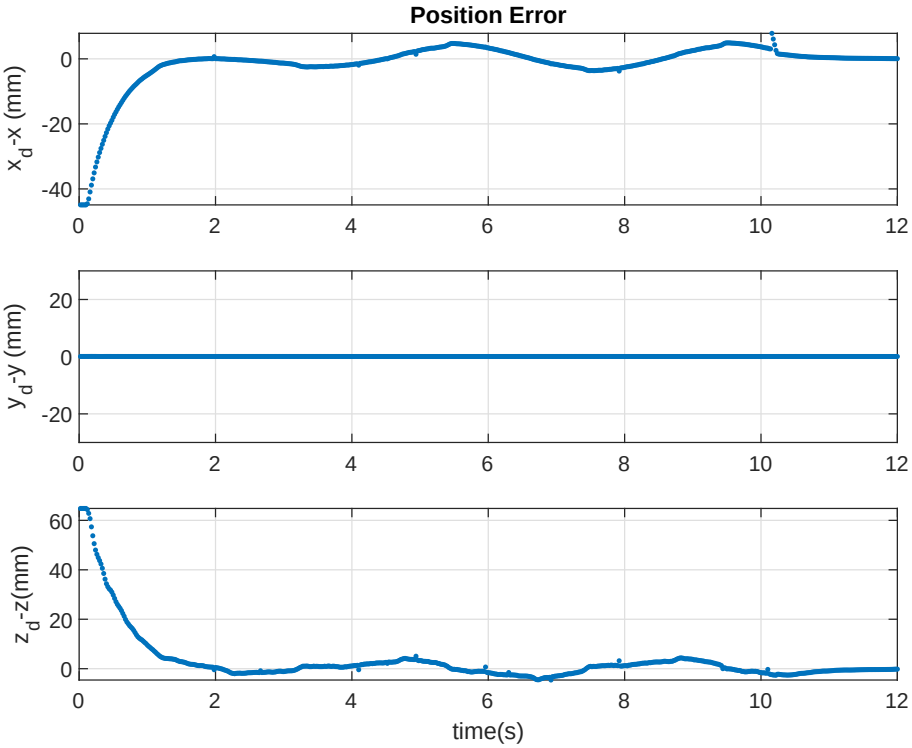


Figure 5.4: Experimental Results - Traj. Circle: Position error $x_d - x$ for $\tilde{J}_B = 2 \times 10^7$

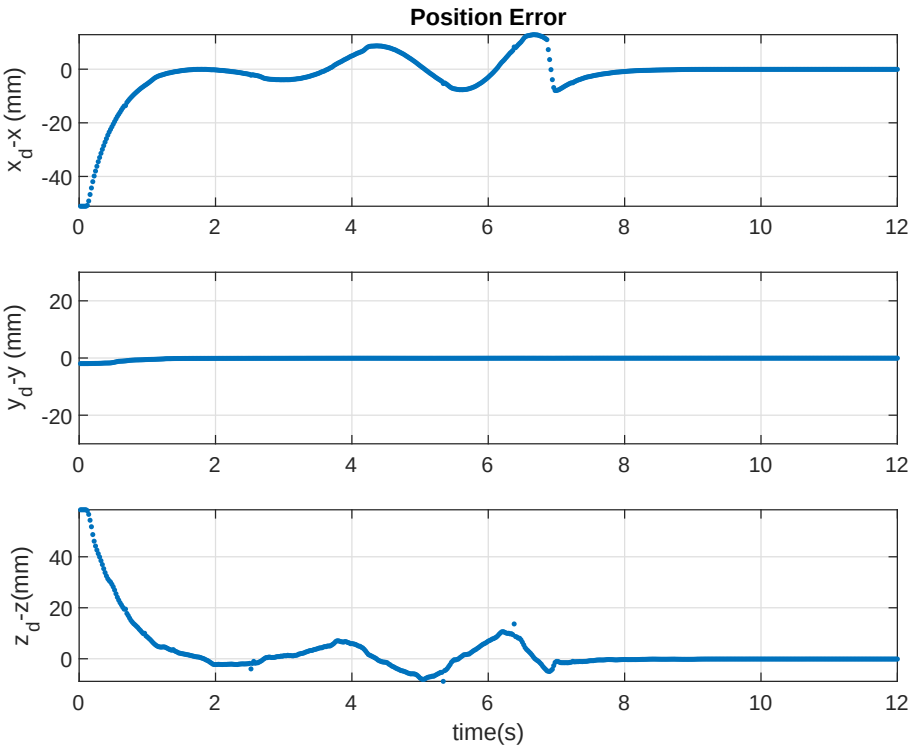


Figure 5.5: Experimental Results - Traj. Circle: Position error $x_d - x$ for $\tilde{J}_B = 3 \times 10^7$

## 5.3 Experiment 2 - Star Trajectory

The path is a combination of sinusoidal functions on $x$ and $z$ plane forming a star. The path is described by the function $f_p(s)$ below:

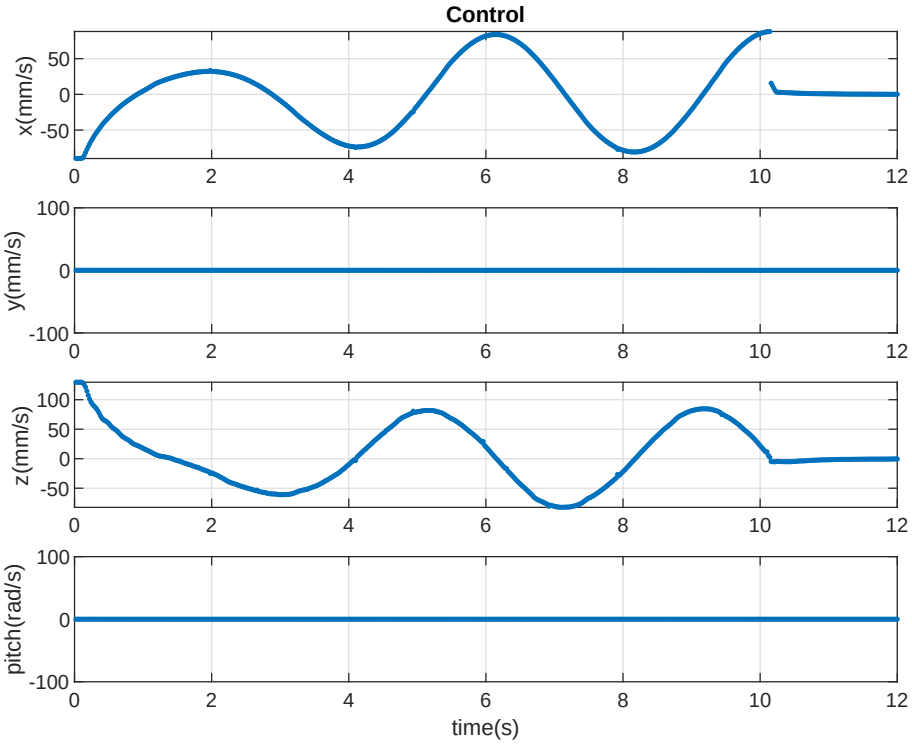


Figure 5.6: Experimental Results - Traj. Circle: Control $u$ for $\tilde{J}_B = 2 \times 10^7$

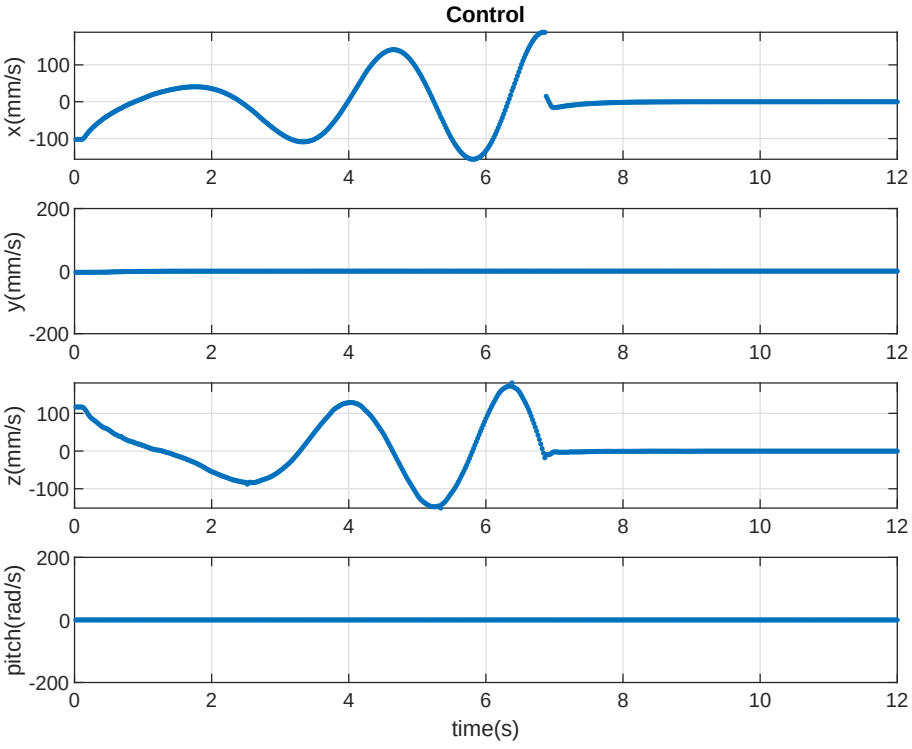


Figure 5.7: Experimental Results - Traj. Circle: Control $u$ for $\tilde{J}_B = 3 \times 10^7$

$$f_p(s) = \begin{bmatrix} 7.2 \cdot 1.5556\ cos(18\pi\ s) - 7.2\ cos(1.5556 \cdot 18\pi\ s) + 500 \\ 57 \\ 7.2 \cdot 1.5556\ sin(18\pi\ s) - 7.2\ sin(1.5556 \cdot 18\pi\ s) - 67 \\ 0 \\ 70 \end{bmatrix} \qquad s \in [0, 1]$$

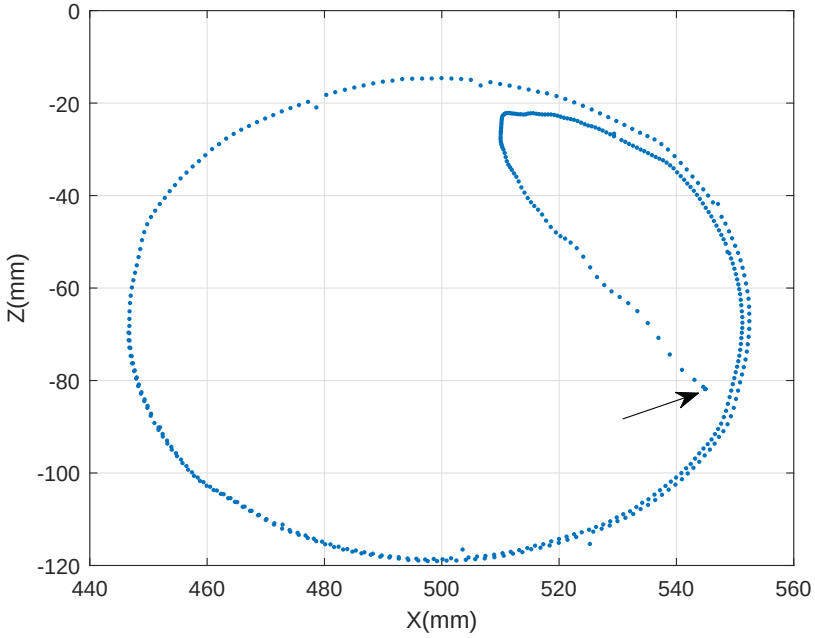


Figure 5.8: Experimental Results - Traj. Circle: Trajectory in X-Z plane for $\tilde{J}_B = 2 \times 10^7$

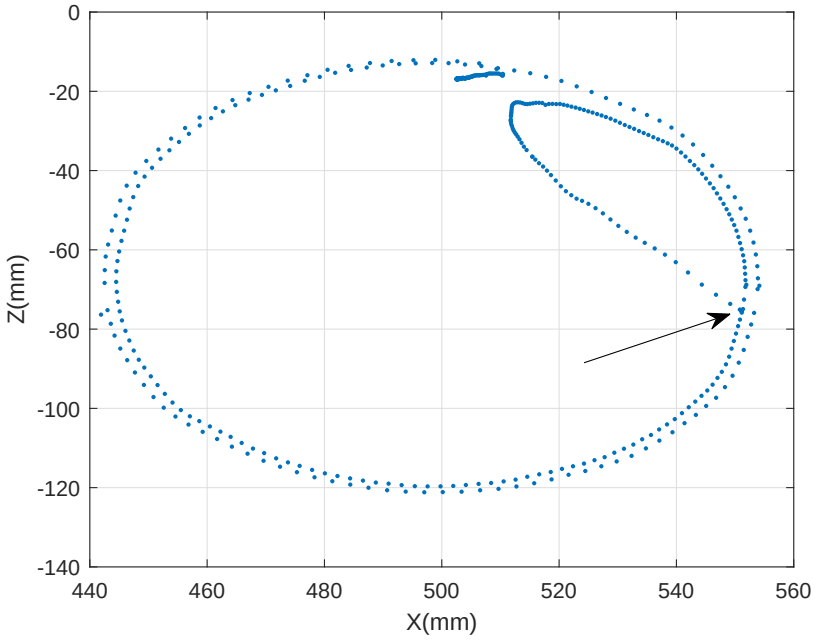


Figure 5.9: Experimental Results - Traj. Circle: Trajectory in X-Z plane for $\tilde{J}_B = 3 \times 10^7$

The predictive control loop runs at $5Hz$ and the kinematic control loop at $50Hz$. The dynamic system used to govern $z$ is as stated below:

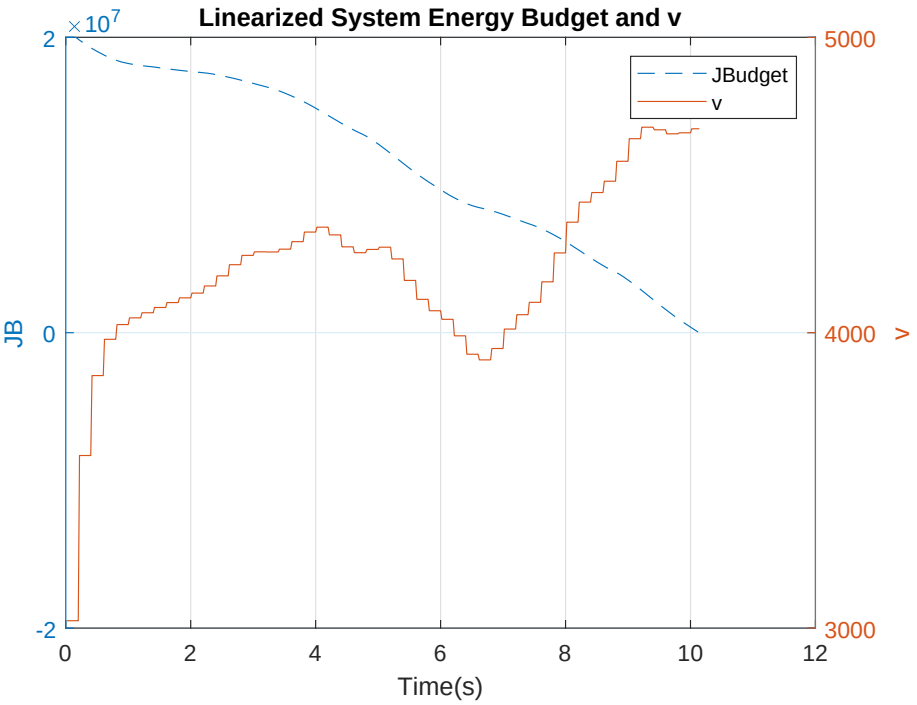


Figure 5.10: Experimental Results - Traj. Circle: $\tilde{J}_B(k)$ and $v$ for $\tilde{J}_B = 2 \times 10^7$

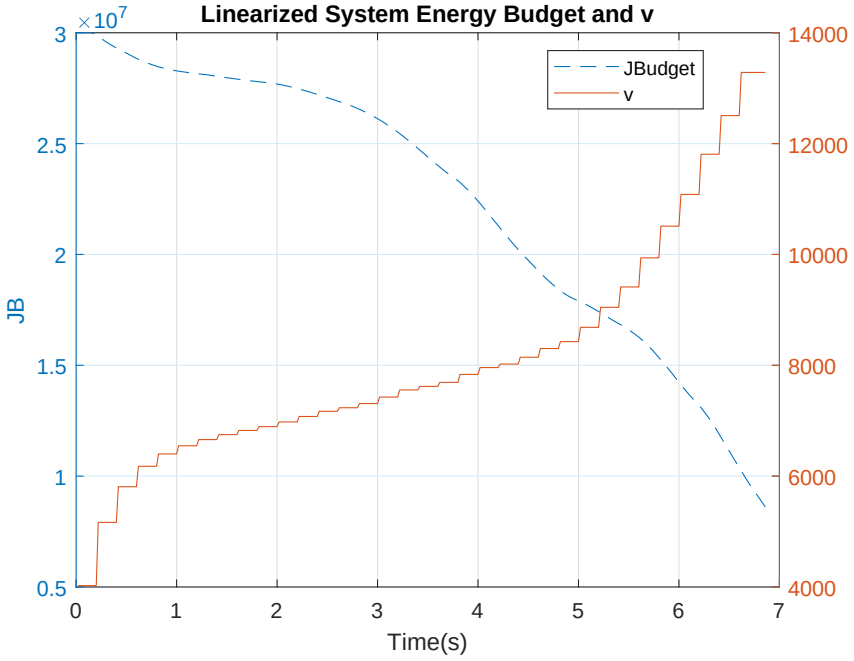


Figure 5.11: Experimental Results - Traj. Circle: $\tilde{J}_B(k)$ and $v$ for $\tilde{J}_B = 3 \times 10^7$

$$A_z = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -3 \cdot 10^{-3} & -1 & -2 \end{bmatrix}; B_z = \begin{bmatrix} 0 \\ 0 \\ 3 \cdot 10^{-3} \end{bmatrix}$$

This system has DC gain equal to 1 and is discretized with ZOH assumption

with $\Delta t = 0.02$. The correction of $v$ on the Newton method is done with $\mu = 10^{-7}$ and the initial value for $J_B$ is $J_B(0) = 2.6 \cdot 10^7$. The control gain is $K_p = 4$.

The experiment is performed with two different initial values for $v$: $v(0) = 2$ and $v(0) = 8$.

### 5.3.1 Case 1: $v(0) = 8$

The position error is plotted in figure 5.12. For $x$ the error is within $\pm 1mm$ and for $z$ the error is within $\pm 2mm$. The control $u$, shown in figure 5.13, is for both $x$ and $z$ within $\pm 20\ mm/s$.

Figure 5.14 shows the $J_B$ being entirely consumed and the control variable $v$ corrections accordingly. The closed loop method using predictive control within the Newton step presents satisfactory results.

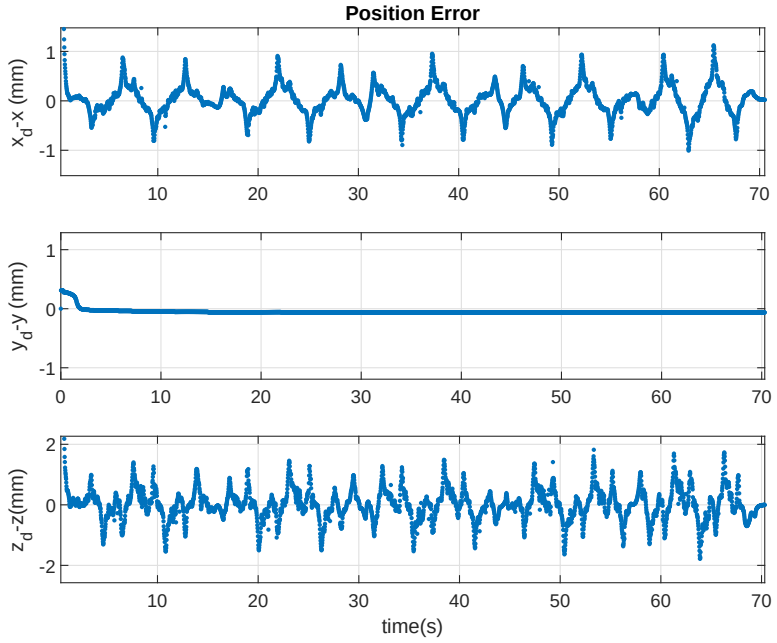


Figure 5.12: Experimental Results - Traj. Star: Position error $x_d - x$ for $v(0) = 8$

### 5.3.2 Case 2: $v(0) = 2$

The position error is plotted in figure 5.15. For $x$ the error is within $\pm 1mm$ and for $z$ the error is within $\pm 2mm$. The control $u$, shown in figure 5.16, is for both $x$ and $z$ within $\pm 20\ mm/s$.

Figure 5.17 shows the $J_B$ being entirely consumed and the control variable $v$ corrections accordingly. Again, the proposed methods presents satisfatory results.

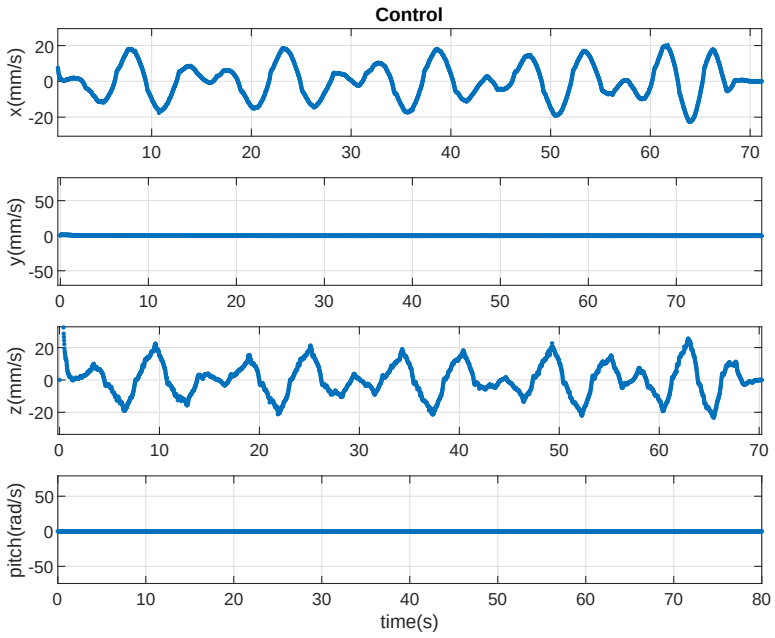


Figure 5.13: Experimental Results - Traj. Star: Input $u$ for $v(0) = 8$

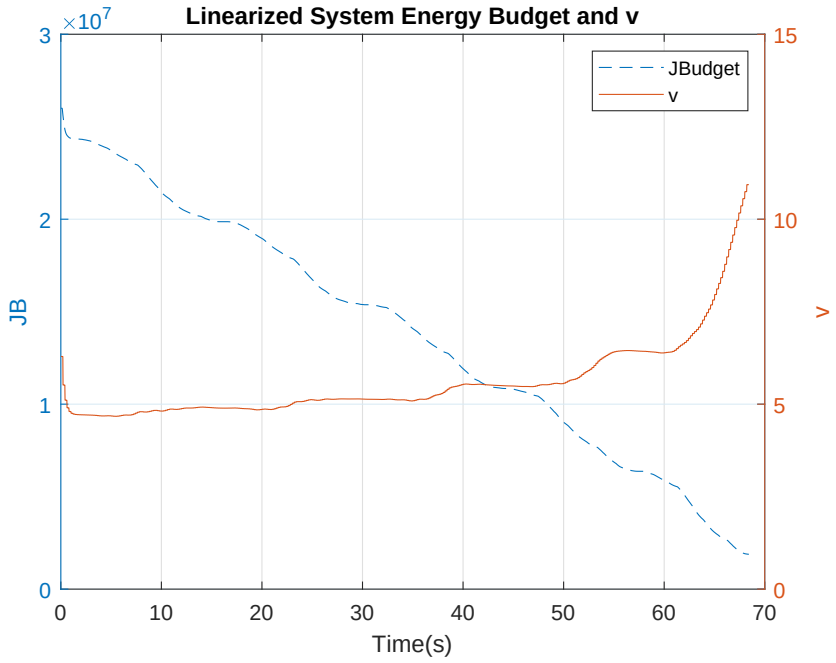


Figure 5.14: Experimental Results - Traj. Star: $\tilde{J}_B(k)$ and $v$ for $v(0) = 8$

## 5.4 Conclusions

This chapter presents the experimental results of the energy budget optimization with Newton method and predictive control applied to a robot manipulator. Two different trajectories are used to verify the method. For each trajectory, two different initial conditions for $v$ or for $J_B$ are considered. The consumption of the energy

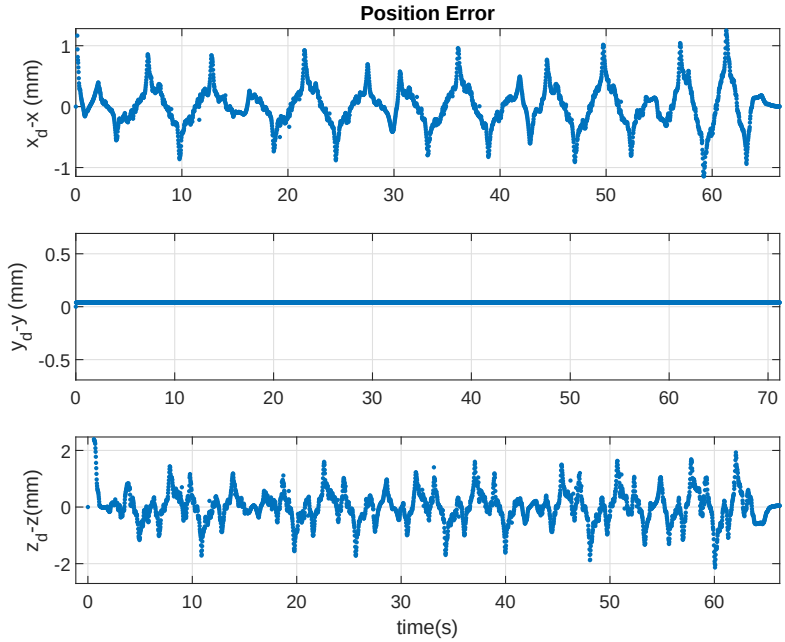


Figure 5.15: Experimental Results - Traj. Star: Position error $x_d - x$ for $v(0) = 2$

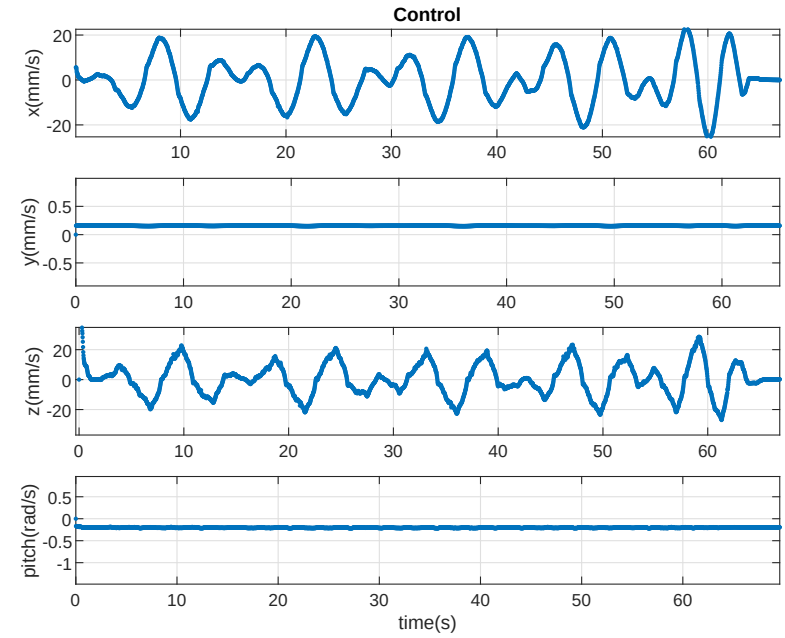


Figure 5.16: Experimental Results - Traj. Star: Input $u$ for $v(0) = 2$

budget is emulated with information from the drives. The active current and the velocity are used to estimate the energy consumption and update the budget.

The positive results presented in this chapter confirms the numerical results obtained in the previous chapters showing that the proposed solution is suitable for online application on a real industrial system.

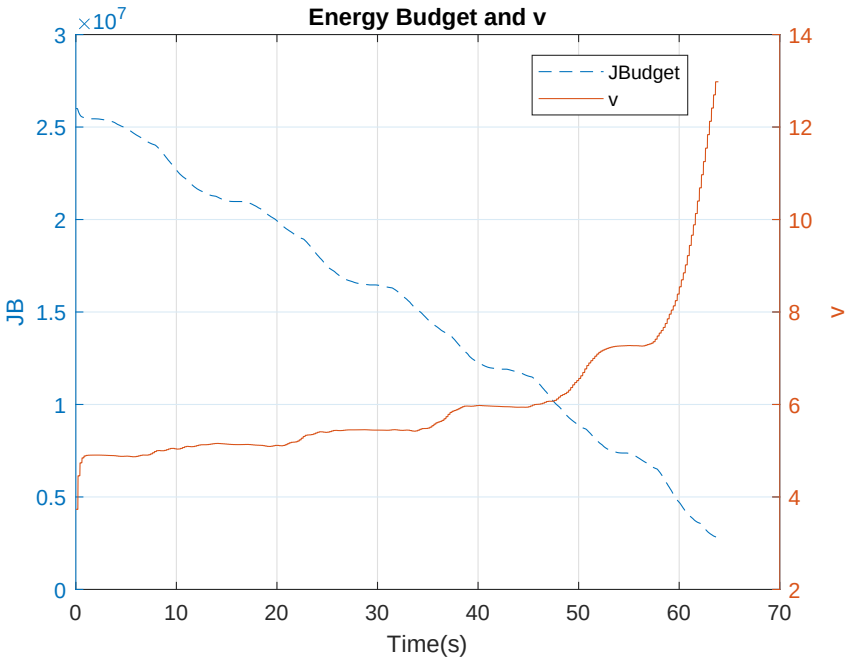


Figure 5.17: Experimental Results - Traj. Star: $\tilde{J}_B(k)$ and $v$ for $v(0) = 2$

# Chapter 6

# Conclusions

This work deals with the problem of optimization of robot manipulator trajectories. Trajectory optimization has been widely studied over the years, but here a novel optimization goal is addressed. The trajectories are optimized to consume a given energy budget. This problem is present in many sectors of the industry where there is energy restriction or a compromise between available energy and speed. The proposed solution is based in a Newton iteration method where a cost function defined as the difference between the available energy and the expected energy required for the trajectory is minimized. The iteration variable is lifted and the iteration occurs in the lifted space. This variable is corrected in a closed loop by merging the Newton method in a predictive control scheme where the control is executed every iteration step.

The main advantages of the proposed method is the fact that is robust to variations in the energy budget or unmodelled energy losses and the low computational cost, which permits the method to be used online in a real time system.

The proposed method could be adapted to other systems such as electric vehicles in applications where the path is known, such as autonomous vacuum cleaning robots, drones with delivery tasks and Formula E race cars. Also, the method could be adapted to be used in satellites.

Other optimization methods could be used with modifications to solve the proposed optimization goal (Bobrow et al. [5], Verscheure et al. [47]), but the simplicity and the possibility to be used online are notable advantages of the proposed method in this dissertation.

The method is verified through numerical simulations of robotic manipulator both in joint space and task space formulations. Furthermore, the proposed method is verified experimentally using the 4 joint light manipulator Tetis. The energy consumption is measured indirectly by using the actuator currents and actual velocities.

The success of the experiments and simulations shows that the method can be used online in a robotic manipulator system.

An additional contribution is the development of dynamic model for the Tetis manipulator. As far as the author knows this is a novel contribution.

## 6.1 Future Work

Despite the good results obtained so far, there are points still open for further development. Suggestions to continue the work started here are proposed:

- Include the use of restrictions in the formulation such as joint torque and speed limits. This could be approached by adding a penalty factor in the cost function. The control variable would have to be defined in a less restrictive way then a constant, i.e., a power series.
- The linear system used to control the dynamics of the parameterized trajectory could be changed into a different type of system. It is still not clear what could be achieved by using different types of system but this is definitely an open point for future discussions.
- The use of an energy function that takes into consideration the braking energy generated by the motors.
- The extension of the method to be used in electric vehicles and mobile robots. This is a natural step to be followed since energy budget optimization is quite relevant in applications such as Formula E and the vacuum cleaning autonomous robots.

# Bibliography


[1] Aadland, A.-K. & Petersen, K. (2010), Subsea all electric, *in* 'Proc. of Offshore Technology Conference', Offshore Technology Conference, Houston, Tx, USA.

[2] Armijo, L. (1966), 'Minimization of functions having lipschitz continuous first partial derivatives', *Pacific Journal of mathematics* **16**(1), 1–3.

[3] Arnol'd, V. I. (2013), *Mathematical methods of classical mechanics*, Vol. 60, Springer Science & Business Media.

[4] Azevedo, A., Bernardo, L. & Labes, A. (n.d.), 'Shared actuation system', 7 jan. 2014, 10 may 2017. Patent Number: EP3165709A1.

[5] Bobrow, J., Dubowsky, S. & Gibson, J. (1983), On the optimal control of robotic manipulators with actuator constraints, *in* 'Proc. of American Control Conference', San Francisco, CA, USA, pp. 782–787.

[6] Chaves, L. & Lizarralde, F. (2006), Uma nova solução ao problema de navegação de robôs móveis autônomos, *in* 'Proc. of Congresso Brasileiro de Automática', Salvador, BA, Brazil, pp. 2045–2050.

[7] Chen, C.-T. (1998), *Linear system theory and design*, Oxford University Press, Inc.

[8] Corke, P. (2017), *Robotics, vision and control: fundamental algorithms in MATLAB® second, completely revised*, Vol. 118, Springer.

[9] Craig, J. J. (2005), *Introduction to Robotics Mechanics and Control*, Pearson.

[10] Cunha, D. & Lizarralde, F. (2019), Real-time path-constrained trajectory tracking for robot manipulators with energy budget optimization, *in* 'Int. Conf. on Automation Science and Engineering', Vancouver, BC, Canada, pp. 1327–1332.

[11] Debrouwere, F., Van Loock, W., Pipeleers, G., Tran, D. Q., Diehl, M., De Schutter, J. & Swevers, J. (2013), 'Time-optimal path following for robots with convex-concave constraints using sequential convex programming', *IEEE Transactions on Robotics* **29**(6), 1485–1495.

[12] Fernandes, C., Gurvits, L. & Li, Z. (1994), 'Near-optimal nonholonomic motion planning for a system of coupled rigid bodies', *IEEE Transactions on Automatic Control* **39**(3), 450–463.

[13] Field, G. & Stepanenko, Y. (1996), Iterative dynamic programming: an approach to minimum energy trajectory planning for robotic manipulators, *in* 'Proc. of IEEE International Conference on Robotics and Automation', Vol. 3, Minneapolis, MN, USA, pp. 2755–2760.

[14] Franklin, G. F., Powell, J. D. & Workman, M. L. (1990), *Digital Control of Dynamic Systems*, Addison and Wesley.

[15] Galassi, M., Røyrøy, A., Carvalho, G., Freitas, G., From, P., Costa, R., Lizarralde, F., Hsu, L., de Carvalho, G., de Oliveira, J., Lima, A., Prego, T., Netto, S. & Silva, E. (2014), Doris—a mobile robot for inspection and monitoring of offshore facilities, *in* 'Proc. of Congresso Brasileiro de Automática', Belo Horizonte, MG, Brazil, pp. 3174–3181.

[16] Goldstein, H., Poole, C. & Safko, J. (2001), *Classical mechanics*, Addilson Wesley.

[17] Hasan, Z., Kapetanic, N., Vaughan, J. & Robinson, G. (2015), Subsea field development optimization using all electric controls as an alternative to conventional electro-hydraulic, *in* 'Proc. of SPE/IATMI Asia Pacific Oil & Gas Conference and Exhibition', Nusa Dua, Bali, Indonesia.

[18] Hsu, L. & Lizarralde, F. (2007), 'Robôs manipuladores', in Aguirre, Luis Antonio, Enciclopédia de automática (vol. 3): controle e automação, Editora Blucher.

[19] Johansen, J., Magnus, H. & Almedal, R. (n.d.), 'Electric actuator', 16 may. 2000, 22 jul 2003. Patent Number: US6595487B2.

[20] Kibble, T. W. & Berkshire, F. H. (2004), *Classical mechanics*, World Scientific Publishing Company.

[21] Kim, B. K. & Shin, K. G. (1985), 'Suboptimal control of industrial manipulators with a weighted minimum time-fuel criterion', *IEEE Transactions on Automatic Control* **30**(1), 1–10.

[22] Kontny, D. & Stursberg, O. (2016), Fast control using homotopy properties for obstacle-avoidance of systems with input constraints, *in* 'Proc. of IEEE Conference on Computer Aided Control System Design', IEEE, Buenos Aires, Argentina, pp. 654–660.

[23] Kontny, D. & Stursberg, O. (2017), Online adaption of motion paths to time-varying constraints using homotopies, *in* 'Proc. of IFAC World Congress', Vol. 50, Toulouse, France, pp. 3331–3337.

[24] Lin, C.-Y., Zhao, Y., Tomizuka, M. & Chen, W. (2016), Path-constrained trajectory planning for robot service life optimization, *in* 'Proc. of American Control Conference', Boston, MA, USA, pp. 2116–2122.

[25] Lizarralde, F. (1998), 'Stabilization of affine nonlinear control system based on a newton-type method', D.Sc. Thesis, Federal University of Rio de Janeiro, COPPE, Rio de Janeiro, RJ, Brazil.

[26] Lizarralde, F., Wen, J. T. & Hsu, L. (1997), Feedback stabilization of nonlinear systems: a path space iteration approach, *in* 'Proc. of the 36th IEEE Conference on Decision and Control', Vol. 4, San Diego, CA, USA, pp. 4022–4023.

[27] Mair, J. (1997), Mac manifold-a revolutionary concept in deepwater production, *in* 'Proc. of Underwater Technology International: Remote Intervention', Aberdeen, UK.

[28] Moe, S., Monsson, O., Rokne, ø., Kumar, A. & Johansen, C. (2018), Electric controls technology: The role in future subsea systems, *in* 'Proc. of Offshore Technology Conference Asia', Offshore Technology Conference, Kuala Lumpur, Malaysia.

[29] Moreira, C., Ribeiro, L., Cerqueira, M., Garcia, A. & Cosentino, L. (1998), Shared actuator manifold-an innovative conception to minimize costs, *in* 'Proc. of Offshore Technology Conference', Houston, Tx, USA.

[30] Patel, M. (2004), *Spacecraft Power Systems*, CRC press.

[31] Pfeiffer, F. & Johanni, R. (1987), 'A concept for manipulator trajectory planning', *IEEE Journal of Robotics and Automation* **3**(2), 115–123.

[32] Reynoso-Mora, P., Chen, W. & Tomizuka, M. (2013), On the time-optimal trajectory planning and control of robotic manipulators along predefined paths, *in* 'Proc. of American Control Conference', IEEE, Washington, DC, USA, pp. 371–377.

[33] Richter, S. L. & Decarlo, R. A. (1983), 'Continuation methods: Theory and applications', *IEEE Transactions on Systems, Man, and Cybernetics* **30**(4), 459–464.

[34] Schilling, A. & Cohan, S. (n.d.), 'Subsea system comprising a crawler', 30 jun. 2016. Patent Number: US20160186534A1.

[35] Shiller, Z. (1994), 'On singular time-optimal control along specified paths', *IEEE Transactions on Robotics and Automation* **10**(4), 561–566.

[36] Shiller, Z. (1996), 'Time-energy optimal control of articulated systems with geometric path constraints', *Journal of Dynamic systems, measurement, and control* **118**(1), 139–143.

[37] Shin, K. & McKay, N. (1985), 'Minimum-time control of robotic manipulators with geometric path constraints', *IEEE Transactions on Automatic Control* **30**(6), 531–541.

[38] Shin, K. & McKay, N. (1986), 'A dynamic programming approach to trajectory planning of robotic manipulators', *IEEE Transactions on Automatic Control* **31**(6), 491–500.

[39] Siciliano, B., Sciavicco, L., Villani, L. & Oriolo, G. (2009), *Robotics, Modelling, Planning and Control*, Springer-Verlag.

[40] Silva, L. G. (2017), 'Control of a 4-dof lightweight robotic manipulator with ros and qt based software', BE Final Project, Federal University of Rio de Janeiro, Rio de Janeiro, RJ, Brazil.

[41] Slotine, J.-J. & Yang, H. S. (1989), 'Improving the efficiency of time-optimal path-following algorithms', *IEEE Transactions on Robotics and Automation* **5**(1), 118–124.

[42] Sontag, E. (1998), *Mathematical Control Theory: Deterministic Finite Dimensional Systems*, Springer Verlag.

[43] Sontag, E. D. (1995), 'Control of systems without drift via generic loops', *IEEE Trans. on Automatic Control* **40**(7), 1210–1219.

[44] Spong, M. W., Hutchinson, S. A. & Vidyasagar, M. (2005), *Robot Modeling and Control*, Wiley.

[45] Sten-Halvorsen, V. & Koren, E. (2008), All electric subsea tree system development, *in* 'Proc. of Offshore Technology Conference', Vol. 19547, Houston, Tx, USA.

[46] Verscheure, D., Demeulenaere, B., Swevers, J., De Schutter, J. & Diehl, M. (2008), Time-energy optimal path tracking for robots: a numerically efficient optimization approach, *in* 'Proc. of the 10th international workshop on advanced motion control', Trento, Italy, pp. 727–732.

[47] Verscheure, D., Demeulenaere, B., Swevers, J., De Schutter, J. & Diehl, M. (2009), 'Time-optimal path tracking for robots: A convex optimization approach', *IEEE Transactions on Automatic Control* **54**(10), 2318–2327.

[48] Wen, J. T. (1995), 'Control of nonholonomic systems', in W. Levine, The Control Handbook, CRC Press.

[49] Xaud, M. F. S. (2016), 'Modeling, control and electromechanical design of a modular lightweight manipulator for interaction and inspection tasks', M.Sc. Dissertation, Federal University of Rio de Janeiro, COPPE, Rio de Janeiro, RJ, Brazil.

[50] Xu, H., Zhuang, J., Wang, S. & Zhu, Z. (2009), Global time-energy optimal planning of robot trajectories, *in* 'Int. Conf. of Mechatronics and Automation', IEEE, Changchun, China, pp. 4034–4039.

# Appendix A

# Energy of the Manipulator

The energy consumed by a manipulator can be calculated by the work performed by its actuators. Work on a particle can be informally defined as the product of a force with the displacement of the particle resulted by this force. Similarly, the work performed by the motors is the product between torque and speed of its shaft. The two basic conditions to have work is that the force needs at least a component parallel to the displacement direction and that the displacement is not zero. If the work is positive that means the force was in the same direction as the displacement and if the work is negative then the force was in opposite direction as the displacement.

## A.1 Mechanical Energy

The mechanical energy of a system is the sum of kinetic and potential energy of its parts (Arnol'd [3],Kibble & Berkshire [20]).

The kinetic energy of a robotic manipulator is the net sum of the links and joints kinetic energies (Siciliano et al. [39] section 7.1.1). Also, the potential energy of a robotic manipulator is the net sum of the links and joints potential energy (Siciliano et al. [39] section 7.1.2).

Furthermore, the change in the summed kinetic plus potential energy of a system equals the rate of the work performed by external forces (Kibble & Berkshire [20]).

## A.2 Work

The work of a force applied on a particle along a curve is the line integral of the product between force and displacement along the curve (Goldstein et al. [16],Arnol'd [3]).

$$W_p = \int_C F \cdot dr \tag{A.1}$$

where $F$ is the force, $r$ is the displacement and $C$ is the curve.

With a change of integration variable, equation A.1 is written in terms of power

$$\begin{aligned} W_p &= \int_C F \cdot \frac{dt}{dt} dr \\ W_p &= \int_{t_0}^{t_f} F \cdot \frac{dr}{dt} dt \\ W_p &= \int_{t_0}^{t_f} F \cdot v \, dt \end{aligned} \tag{A.2}$$

where $v$ is the velocity, $t_0$ is the initial time and $t_f$ the final time.

Equivalently, work can be defined for angular velocity and torque:

$$W_p = \int_{t_0}^{t_f} \tau \cdot \omega \, dt \tag{A.3}$$

where $\tau$ is the torque and $\omega$ is the angular velocity.

## A.3 The Energy Function

For a robotic manipulator, the energy used in a given task can be calculated by the summed work done by each actuator. But, for simplicity, it is only considered the case where the regenerated energy due to braking or external forces (this would be negative work) is dissipated. Ergo, only the absolute values of torque and joint speed are used. This simplification doesn't restrict much the method presented here because most industrial manipulators doesn't store energy that comes back from the electric motors. Typically, the regenerated energy is dissipated on a shunt resistor by a chopper circuit.

It is worth mentioning that the work equals the variation of mechanical energy.

With all that said, we consider here that the energy used by the robot manipulator is given by [1]

$$\begin{aligned} J_T &= \int_0^T |\dot{q}|^T |\tau| dt \\ |\dot{q}| = \begin{bmatrix} |\dot{q}_1| \\ \vdots \\ |\dot{q}_N| \end{bmatrix} & \qquad |\tau| = \begin{bmatrix} |\tau_1| \\ \vdots \\ |\tau_N| \end{bmatrix} \end{aligned} \tag{A.4}$$

[1] J will be used for the calculation of manipulator energy instead of W to avoid confusion with the concept of work presented before.

# Appendix B

# Dynamic Model of Tetis Manipulator

Here the dynamic model (2.4) of Tetis is devised to be used in numerical simulations. It is assumed that the links are rigid and the mass and inertia of the link (carbon fiber tubes) and actuators are considered. The inertia and center of gravity of the links are obtained from the 3D model using Solid Works (from Harmonic Drive Ag website https://cad.harmonicdrive.de/).

The mass and inertia parameters are loaded in link objects in Robotics Toolbox for Matlab (Corke [8]). The toolbox is used then to calculate the robot dynamic equation based on this parameters.

The manipulator kinematic is defined in the robotics tool box using the Denavit–Hartenberg parameters as described in (Siciliano et al. [39]). The parameters for Tetis contained in Silva [40] are used here.

## B.1 Kinematic Model: Denavit–Hartenberg Parameters

Reference frames $O_0$ to $O_4$ are positioned as shown in Figure B.1. The actuators 1 to 4 are located in the joints counting from the base to the end-effector.

The parameters devised in Silva [40] are presented in table B.1.

Table B.1: Tetis Denavit–Hartenberg Parameters

| Link | $a_i$ | $\alpha_i$ | $d_i$ | $\theta_i$ |
|---|---|---|---|---|
| 1 | 0.000 | $\pi/2$ | 0 | $\theta_1$ |
| 2 | 0.320 | 0 | 0 | $\theta_2$ |
| 3 | 0.225 | 0 | 0 | $\theta_3$ |
| 4 | 0.167 | $-\pi/2$ | $-0.057$ | $\theta_4$ |

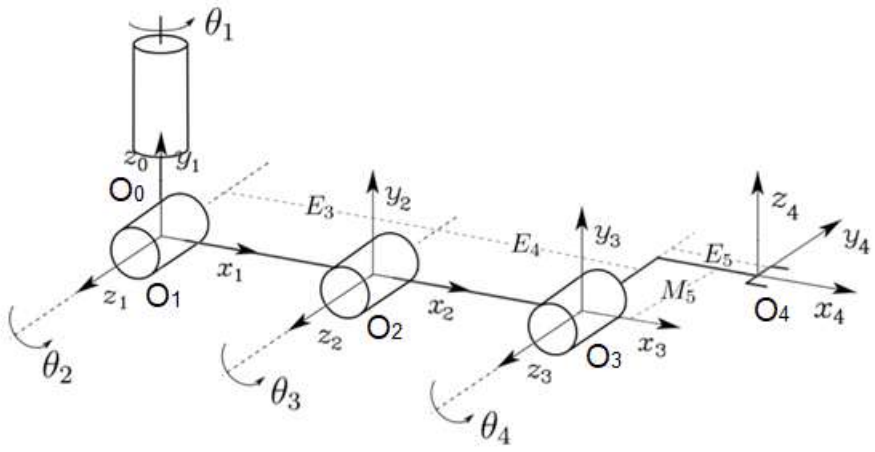


Figure B.1: Tetis Coordinate Frames

## B.2 Mass and Inertia Parameters

The inertia parameters are calculated using the SolidWorks model of each link. In figure B.2 the lateral and top view of the manipulator is illustrated. The center of mass of $i$-links is represented in $O_i$, and the inertia matrix $I_i^c$ is calculated at center the center of mass of the $i$-link aligned with frame $O_i$.

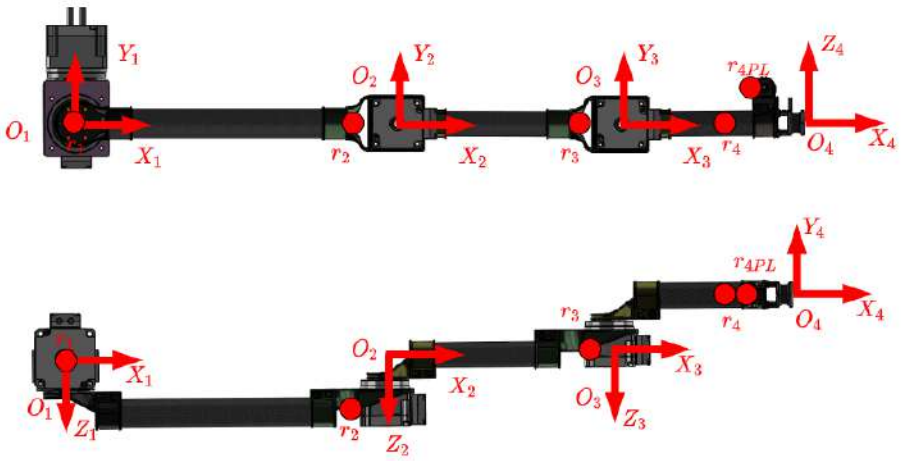


Figure B.2: Tetis SolidWorks Model: lateral view and top view, with coordinate frame $O_i$ and center of mass $r_i$ (red circle) of each link.

Link 1, which include motor 2, is presented in figure B.3 with center of mass, $r_1$, expressed in coordinate frame $O_1$, the mass of link 1 $m_1$ and the inertia matrix $I_1^c$

at the center of mass aligned with $O_1$:

$$m_1 = 0.687 \quad r_1 = \begin{bmatrix} 0 \\ 0 \\ 0.00534 \end{bmatrix} \quad I_1^c = 10^{-9} \begin{bmatrix} 444442.46 & -493.15 & 38.88 \\ -493.15 & 351453.26 & 16515.24 \\ 38.88 & 16515.24 & 480951.05 \end{bmatrix} \tag{B.1}$$

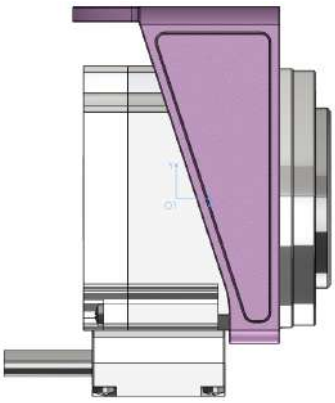

Figure B.3: Tetis Link 1 including actuator 2

Link 2, which include motor 3, has center of mass, $r_2$, expressed in coordinate frame $O_2$, mass $m_2$ and the inertia matrix $I_2^c$ at the center of mass aligned with $O_2$ given by:

$$m_2 = 0.5867 \quad r_2 = \begin{bmatrix} -0.04776 \\ 0.0 \\ -0.05023 \end{bmatrix} \quad I_2^c = 10^{-9} \begin{bmatrix} 193184.65 & -201.05 & -9038.46 \\ -201.05 & 5776891.08 & 16.52 \\ -9038.46 & 16.52 & 5779194.58 \end{bmatrix} \tag{B.2}$$

Link 3, which include motor 4, has center of mass, $r_3$, expressed in coordinate frame $O_3$, mass $m_3$ and the inertia matrix $I_3^c$ at the center of mass aligned with $O_3$ given by:

$$m_3 = 0.5565 \quad r_3 = \begin{bmatrix} -0.02833 \\ 0.0 \\ -0.005 \end{bmatrix} \quad I_3^c = 10^{-9} \begin{bmatrix} 198271.88 & -260.27 & -187130.34 \\ -260.27 & 2491276.83 & 10.21 \\ -187130.34 & 10.21 & 2477820.78 \end{bmatrix} \tag{B.3}$$

Link 4, which include a webcam, has center of mass, $r_4$, expressed in coordinate frame $O_4$, mass $m_4$ and the inertia matrix $I_4^c$ at the center of mass aligned with $O_4$ given by:

$$m_4 = 0.257 \quad r_4 = \begin{bmatrix} -0.06489 \\ -0.002 \\ 0.03115 \end{bmatrix} \quad I_4^c = 10^{-9} \begin{bmatrix} 401293.05 & 51869.93 & 203652.40 \\ 51869.93 & 1010541.25 & 16459.43 \\ 203652.40 & 16459.43 & 745681.16 \end{bmatrix} \tag{B.4}$$

## B.3 Numerical Simulation using the Robotics Toolbox

In this section the Matlab code used to instantiate the manipulator Tetis in the Robotics Toolbox (Corke [8]) is presented. The manipulator model is presented in the script *mdl_tetis.m*:

```
% mdl_tetis  Create model of Tetis planar manipulator
 % Everything is in SI units (m, kg, m^3)
E3=0.320;    E4=0.225;   E5=0.16725;   M5=0.057;
friction = 0;%0.0002;

clear L
%             th    d       a         alpha
L(1) = Link([ 0     0       0         pi/2    0], 'standard');
L(2) = Link([ 0     0       E3        0       0], 'standard');
L(3) = Link([ 0     0       E4        0       0], 'standard');
L(4) = Link([ 0    -M5      E5       -pi/2    0], 'standard');

%  Originally in  gr mm^2
L(1).I = [444442.46 -493.15 38.88;  -493.15 351453.26 16515.24;
38.88 16515.24 480951.05]/1000/1000/1000;
L(1).r = [0; 0; 0.00534];
L(1).m = 0.687;
L(1).G = 100;
L(1).Jm = 0.067 * 0.0001;
L(1).B = friction;


L(2).I = [193184.65 -201.05 -9038.46;  -201.05 5776891.08 16.52;
-9038.46 16.52 5779194.58]/1000/1000/1000;
L(2).r = [-0.04776; 0.0; -0.05023];
L(2).m = 0.58671;
L(2).G = 100;
L(2).Jm = 0.067 * 0.0001;
L(2).B = friction;


L(3).I = [198271.88 -260.27 -187130.34;  -260.27 2491276.83 10.21;
-187130.34 10.21 2477820.78]/1000/1000/1000;
L(3).r = [-0.02833; 0.0; -0.005];
L(3).m = 0.55646;
L(3).G = 100;
L(3).Jm = 0.029 * 0.0001;
L(3).B = friction;
```

```
L(4).I = [401293.05 51869.93 203652.40;  51869.93 1010541.25 16459.43;
203652.40 16459.43 745681.16]/1000/1000/1000;
L(4).r = [-0.06489; -0.002; 0.03115];
L(4).m = 0.25696;
L(4).G = 100;
L(4).Jm = 0.029 * 0.0001;
L(4).B = friction;

Tetis= SerialLink(L, 'name', 'Tetis', 'manufacturer', 'GSCAR', 'comment', '');
```

To complete the example, the script to perform a numerical simulation is presented.

For example, a numerical simulation can be performed using $ode45()$ matlab function, thus one can use $[t, yT] = ode45(@tetisdyn, [0\ 30], [0\ -\pi/4\ \pi/2\ -\pi/4\ 0\ 0\ 0\ 0])$; and plot the response $plot(t, yT(1:4))$ using the following differential equation with the robot dynamic model:

```
function [dx] = tetisdyn(t,x)
global Tetis;

a = 10*pi/180;     w = pi/4;
kp= 2;             kd= 1;

qd = a*sin(w*t)*ones(4,1) + [0; -pi/4; pi/2; -pi/3];
dqd = a*w*cos(w*t)*ones(4,1);
ddqd = -a*w*w*sin(w*t)*ones(4,1);

q  = x(1:4);
dq = x(5:8);

% Euler-Lagrange Model
mHat = Tetis.inertia(q');
cHat = Tetis.coriolis(q', dq');
gHat = Tetis.gravload(q')';

% Control law: Computed Torque + PD
uBar = ddqd + kp*( qd-q ) + kd*( dqd-dq );
tau = mHat*uBar + cHat*dq + gHat;

% For a more efficient computation use Recursive Newton-Euler method
% tau = Tetis.rne(q', dq', uBar')';

%ddq = -inv(mHat)*(cHat*dq + gHat - tau);
ddq = Tetis.accel(q', dq', tau');

dx = [dq; ddq];
```

Another option to perform a numerical simulation is to use $SerialLink/fdyn$ to Integrate forward dynamics. $[T, Q, QD] = Tetis.fdyn(t, torqfun, q0, qd0)$ integrates the dynamics of the robot over the time interval 0 to $t$ and returns vectors of time $T$, joint position $Q$ and joint velocity $QD$. The initial joint position and velocity are given by $q0$ and $qd0$.

The torque applied to the joints is computed by the user-supplied control function $torqfun$: $tau = torqfun(t, q, qd)$, where $q$ and $qd$ are the manipulator joint coordinate and velocity state respectively, and $t$ is the current time.

For example, if the robot was controlled by a PD controller we can define a function to compute the control

```
function tau = mytorqfun(t, q, qd, qstar, P, D)
tau = P*(qstar-q) + D*qd;
```