Linköping Studies in Science and Technology. Dissertations, No. 1170

Design Optimization in Industrial Robotics

Methods and Algorithms for Drive Train Design

Marcus Pettersson

Linköping University
INSTITUTE OF TECHNOLOGY

Department of Management and Engineering
Division of Machine Design
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköping 2008

ISBN 978-91-7393-945-4
ISSN 0345-7524

Department of Management Engineering
Linköpings universitet
SE-581 83 Linköping, Sweden

Printed in Sweden by LiU-Tryck Linköping, 2008.

Abstract

ROBOT MANUFACTURERS, like many other manufacturers, are experiencing increasing competition in a global market where one way to confront the challenge is by making the development process more efficient. One way to speed up the time to market for new products is to take advantage of design optimization based on simulation models. By optimizing performance with the help of dynamic simulation, an immense amount of both time and money may be saved.

In this thesis, design optimization strategies for industrial robot design are studied. Often, the trade-offs between performance, cost and quality are essential for design decisions. These trade-offs can be investigated with the help of simulation models. Generating the trade-offs can be both cumbersome and time-consuming, but the process may be partly automated with the help of optimization algorithms. How the optimization problem needs to be formulated to generate the trade-off is discussed in this thesis.

Robot design problems usually consist of a mixture of deciding continuous parameters as well as selecting components from catalogs and databases. Hence, there is a need for optimization algorithms which can handle variables of both a discrete and a continuous nature. A new method has been developed to address this problem. The method has also been improved by adding adaptive characteristics for further efficient design optimization.

The ideas in this thesis have been applied to both simulation models of conceptual degrees of elaboration as well as simulation models of complete robot systems. An optimization procedure which shows how optimization can be used in the early phases of a development process is developed. The objective of the optimization is to determine optimal gearboxes and arm lengths from an acceleration capability perspective. An optimization based design method for robot drive trains is also presented. For further efficient use of already installed robots the concept of application adapted performance optimization is introduced. This means that the robot control is optimized with respect to thermal and fatigue load for the specific program that the robot performs. The motion program itself, i.e. the path planning, can be optimized at the same time in order to get the most out of the robot.

Acknowledgments

THERE ARE SEVERAL people to whom I am very grateful for their support. First I would like to thank Associate Professor Johan Ölvander for his more than splendid supervision and support. I also wish to thank Professor Petter Krus, Head of Division, for always having the time for discussion and sharing his expertise.

I would like to thank my industrial supervisor Dr Torgny Brogårdh at ABB Robotics, for guiding my work in the best of ways from an industrial point of view. Hans Andersson, ABB Corporate Research, has helped me out any number of times by sharing his expertise in robot theory and simulations. In addition, I would like to thank Dr Xiaolong Feng, also at ABB Corporate Research, for initiating the project and always being positive and supportive.

I would also like to thank VINNOVA (the Swedish Governmental Agency for Innovation Systems) and the ProViking research program (supported by the Swedish Foundation for Strategic Research) for their financial support.

Linköping, March 2008

Marcus Pettersson

Papers

THE FOLLOWING SEVEN papers are appended and will be referred to by their Roman numerals. The papers are printed in their originally published state except for changes in formatting and correction of minor errata.

[I] Pettersson M., Andersson J., Krus P., Wäppling D., Feng X., Industrial Robot Design Optimization in the Conceptual Design Phase, In *Proceedings of IEEE International Conference on Mechatronics and Robotics,* Aachen , Germany, 2004.

[II] Pettersson M., Andersson J., Krus P., Methods for Discrete Design Optimization, In *Proceedings of ASME 31st Design Automation Conference*, Long Beach, USA, September 24-28, 2005.

[III] Pettersson M., Andersson J., Krus P., On Optimal Drive Train Design in Industrial Robots, In *Proceedings of IEEE International Conference on Industrial Technology,* Hong Kong, December,2005.

[IV] Lundén B., Pettersson M., Ölvander J., A Component Based Optimization Approach for Robot Modular Design, In *Proceedings of DETC'2007 ASME Design Automation Conference*, Las Vegas, Nevada, USA, September 2007.

[V] Pettersson M., Ölvander J., Andersson H., Application Adapted Performance Optimization for Industrial Robots, In *Proceedings of IEEE International Symposium on Industrial Electronics*, Vigo, Spain, June,2007.

[VI] Pettersson M., Ölvander J., Adaptive Complex Method for Efficient Design Optimization, in *Proceedings of ASME 33rd Design Automation Conference*, Las Vegas, USA, September 4-7, 2007.

[VII] Pettersson M., Ölvander J., Drive Train Optimization for Industrial Robots, Submitted to *IEEE Transactions on Robotics.*

Pettersson is the main contributing author of all papers except paper [IV] which is jointly written by the first two authors. In paper [IV] Pettersson has been responsible for robot modelling and simulation as well as for the optimization, and Johansson has developed component models. Professors Ölvander (formerly Andersson) and Krus mainly assisted in discussions in their role as thesis advisors. Feng, Wäppling and Andersson H., assisted in discussions as industrial advisors.

Contents

Appended papers

1 Introduction

ROBOT MANUFACTURERS, like many other manufacturers, are today experiencing ever increasing competition in a global market. For many of the robot manufacturers' customers, a key component in their strategy for greater efficiency has been robot automation. This is not case for the robot manufacturers themselves, who traditionally have little in-house production and an assembly process where merely marginal savings can be made through robot automation. Another way to improve the odds of being one of the fittest in this struggle for survival is instead to speed up the time to market for new products by shortening lead times in the development process [71]. This should of course be achieved without lowering any requirements with regard to quality and performance. However, stricter requirements often increase the cost and a key factor for success is finding the most profitable balance between quality, performance, and cost.

Different computer based tools have been used as one means of achieving this in industry for quite some time. The most familiar tool is Computer Aided Design (CAD). Bringing CAD into play has led to substantial time and cost savings as well as new possibilities [62]. In the area of solid mechanics finite element analysis (FEM) is used by many product developers, for example [52] describes how FEM is used to support product development in crashworthiness design. In addition, in order to analyze the behavior and performance of complex mechanical systems, multi-body dynamic simulation (MBS) are also used. How MBS can be used in the development process is discussed in [35] and [38]. Additionally, Cederfeldt [10] discusses how computer tools in general can be used in a structured way for efficient design automation.

According to Ryan [57], when they developed the Camry model, Toyota began the development process with multi-body simulations in order to optimize performance and function before generating any detailed geometry. These changes led to a factor of three in development cost reduction and a factor of four in time savings. This indicates that significant gains may be made if the trade-offs between performance, cost, and quality can be investigated early on and throughout the whole design process with the help of simulation models. Generating the trade-offs can be both cumbersome and time consuming, but the process may be partly automated by using optimization algorithms.

As is widely recognized, engineering design is an iterative process where new design proposals are generated and evaluated, see [14], [49], [55], and [64]. According to Roosenburg and Eekels [55] the iterative part of the design process consists of synthesis, simulation, evaluation, and decision. As early as 1967, Simon [60] stated that this description can be seen as an optimization process. If the design problems are formalized the optimization can be, if not totally, to some extent closed by mathematical optimization algorithms. Parts of the design process can thereby be automated, and savings in development time are possible. It is also the author's belief that the process of formulating the design problem as an optimization problem helps the designer gain insight about the problem at hand. Since the design process is seldom a one way street and the designer often has to return to earlier activities and reconsider, optimization may speed up these feedback loops (and avoid situations where dead ends appear too late in the design process). A central part of this thesis is thus to formalize the engineering design problem as an optimization problem incorporating the simulation programs used within the development process.

1.1 Contributions

This thesis focuses on questions that arise when using optimization together with simulation models as a means of effective robot design. As stated by Pahl and Beitz [49], the iterations of improvement follow the whole process from clarifying the task and finding the objectives to the detailed design phase. Design optimization can thus be applied in all phases of the development process. It is, however, important to keep in mind that the questions asked (objective function formulations) should suit the simulation model. In the conceptual phase, it is unnecessary to create too elaborate models since the whole situation is characterized by uncertainties and assumptions. Nevertheless, if suitable models are established and the concepts are optimized towards sustainable performance indices which are viable throughout the process into the detailed phase, great savings can be achieved. In paper [I] it is shown how a closed form mathematical model of the dynamics of a robot can be utilized together with optimization in the conceptual design phase in order to help the designer to make decisions on requirements such as payload and reach.

The second issue focuses on how the simulation-based optimization loop can be used for the purpose of generating trade-offs between cost, performance, and lifetime when designing robot drive trains. In paper [III] optimization is used to determine which gearboxes to use and in paper [VII] both the motors and gearboxes are considered. In both cases an in-house simulation code with a trajectory generator identical to the one in real ABB robot systems is used. In paper [IV] a component based design approach for modular drive train design is presented. The simulation model of the complete robot system including mechanics, electronics and software systems used in paper [IV] is developed in Dymola [19].

For further efficient use of already installed robots the concept of application adapted performance optimization is introduced in paper [V]. This means that the robot control is optimized with respect to thermal and fatigue load for the specific program that the robot performs. The motion program itself, i.e. the path planning, can be optimized at the same time in order to get the most out of the robot.

Robot design problems usually consist of a mixture of deciding continuous parameters as well as selecting components from catalogs and databases. There is therefore a need for optimization algorithms which can handle variables of both discrete and continuous character. A new method has been developed to address this problem see paper [II] and section 4.2.5. The method has also been improved by adding adaptive characteristics for further efficient design optimization, see paper [VI] and section 4.2.7.

The contributions mentioned above can summarized as:

- Methods for industrial robot design optimization in the conceptual design phase.
- Methods for industrial robot drive train design in the detail design phase, including trade-off analysis of cost, performance, and expected lifetime.
- Methods for adaptive performance of already installed robots at end users, as a tool for customer tailored optimal performance.
- A modification of the Complex algorithm for mixed variable design problems.
- Adaptive Complex algorithm for efficient and robust design optimization.

2
Industrial robotics

INDUSTRIAL ROBOTS HAVE been around since the sixties and are today widely used in industry. An industrial robot is a machine with significant characteristics of adaptability, agility and flexibility. According to the widely accepted definition of the *Robotics Institute of America*, “a robot is a re-programmable multi-functional manipulator designed to move material, parts, tools, or specialized devices, through variable programmable motions for the performance of a variety of tasks”. The word manipulator refers to a mechanism which consists of a series of segments jointed to one another, for the purpose of grasping and moving objects, pieces or tools in several degrees of freedom. The segments that make up the mechanism are called links and the joints which connect them can be of several different types. The two most common ones for industrial robots are revolute joints which permit two paired elements to rotate in relation to each other, and prismatic joints which allow paired elements to slide in relation to each other.

The mechanism is made up of a *kinematic chain* which is called *closed loop* if every link is connected to every other link by at least two distinct paths, and *open loop* if every link is connected to every other link by one and only one path. In the figure below, a robot with a closed kinematic loop and one with an open kinematic loop are shown.

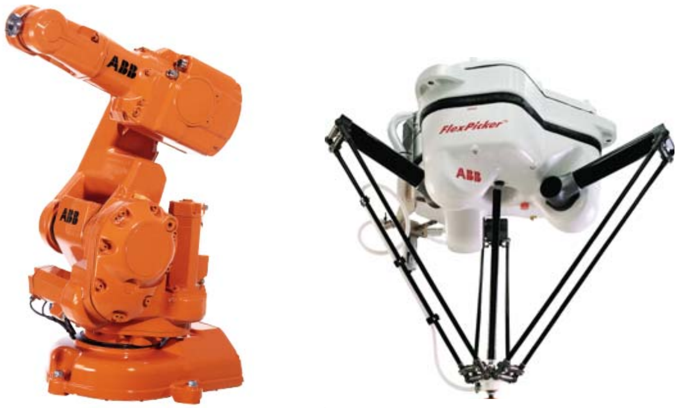


Figure 2.1 On the left, an ABB IRB 140, which is an articulated robot with an open kinematic chain. On the right, an ABB Flexpicker robot, which is a parallel robot consisting of closed kinematic loops.

Mechanisms are often classified by their *degrees of freedom* (DOF), which are the number of independent parameters or inputs needed to specify the configuration of the kinematic chains completely. In the case of open loop robots like the IRB 140 in Figure 2.1, the number of manipulated degrees of freedom equals the number of actuated joints, in this case six. The *end effector* is designed according to the task the robot is to execute. For material handling tasks, the end effector is a gripper. For other tasks, the end effector may for instance be represented by a welding torch, a drill or a spray gun. The volume made up of all the positions reachable by the end-effector is called the *work space* or *task space* and the motion the robot performs during operation is called its *working cycle*.

2.1 The robot system

A typical joint actuating system for an industrial robot is shown in Figure 2.2. The joint actuating system in robots generally consists of a power supply, a servo amplifier, a servomotor, and a transmission.

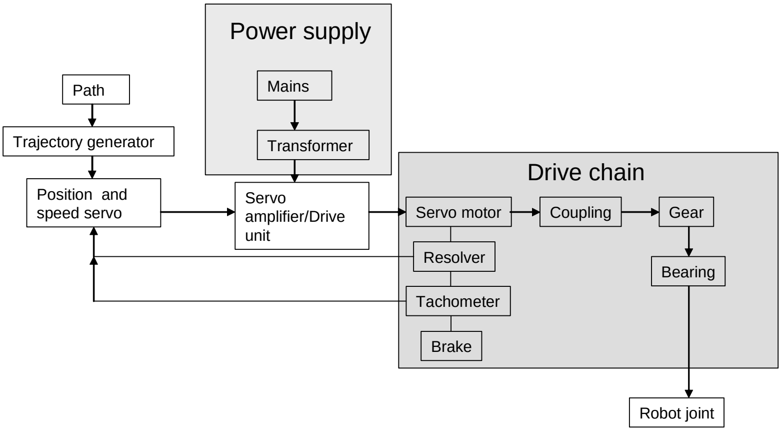


Figure 2.2 The drive system of industrial robots.

The task of the power supply is to supply the servo amplifier with the voltage and current needed to operate the actuating system. The power supply consists of a transformer and a typically uncontrolled bridge rectifier (also exists without a transformer). These allow the alternating voltage available from the distribution to be converted into direct voltage of suitable magnitude which is required to feed the servo amplifier. The servo amplifiers (drive units) have the mission of modulating, under the action of a control signal, the power flow to the actuators for the execution of the desired motion. In other words, the amplifier takes a fraction of the power available at the source which is proportional to the control signal and transmits it to the motor.

The electric motors need to be supplied with a voltage and current of suitable form. Voltage is direct for permanent-magnet DC servomotors and alternating for brushless DC motors (sometimes also named AC servomotors). Brushless DC motors are today more or less standard in industrial robots. The main reason for using a brushless DC motor is to eliminate the problems caused by mechanical commutation of the brushes in a DC motor. The inversion between the functions of stator and rotor compared to permanent-magnet DC motors also has other advantages. The presence of a winding on the stator instead of the rotor facilitates heat dispersal. The absence of a rotor winding, together with the possibility to use rare-earth permanent magnets allows more compact rotors to be designed which are in turn characterized by a low moment of inertia [59].

The choice of transmission depends on the power requirements, the kind of motion, the allocation of the motor with respect to the joint and, of course, cost. Typical quality requirements for robot gears are: very small backlash, high efficiency, large reduction in few steps, low inertia, low friction, high torsional stiffness, high power density and low weight. All these requirements cannot be fulfilled simultaneously. For example, the minimization of backlash requires pretensioned gear teeth, which on the other hand leads to higher friction and, as a consequence, to reduced efficiency. The following transmissions are typically used in revolute joints for industrial robots:

- Spur gears
- Planetary gears
- Harmonic drives
- Planocentric reduction mechanisms (Cycloid)

Spur gears are commonly used for robot wrist axes and typical properties include low price and high efficiency. Planetary gears are characterized by high efficiency and used for the robot base axes. For more information about spur gears and planetary gears in mechatronic applications see [53]. Harmonic drives [25] are commonly used for wrist axes and have very high reduction in one gear stage. Cycloid gearing, see for example [44], has high efficiency and is able to handle high torques and is therefore used for robot base axes.

To transform the output from the motor into a linear motion of the joint, lead screws are commonly used. Mechanical linkages or timing belts may be used to locate the motor remotely from the actuated joint. By mounting some of the motors to the base of the robot the static and dynamic properties of the manipulator may be improved by decreasing the total weight and thereby increasing the power to weight ratio. In the Figure 2.3 , a design is shown where the motor for axis three is mounted on the stand of the robot and then connected to axis three through a linkage called a parallel rod (p-rod). This reduces the inertia of the main joints since no actuator is needed to be mounted on joint three and since the inertia of joint two with respect to the upper arm, wrist and load will be independent of the joint three angle.

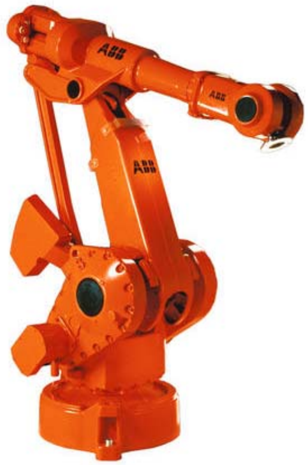


Figure 2.3 ABB IRB 4400.

2.1.1 Modeling

The relation between position, velocity, and acceleration of the end effector and the joints is analyzed using *kinematics*. Kinematics deals with aspects of motion without regard to the forces and/or torques that cause it. There are two types of kinematic problem: direct and inverse kinematics. In the programming of a robot manipulator, typically a set of desired positions and orientations and its time derivates of the end effector are specified in work space. The problem is to find all possible sets of actuated joint variables and their corresponding time derivatives which will bring the end-effector to the set of desired positions and orientations with the desired motion characteristics. This is known as *inverse kinematics*. The *direct kinematics* problem is to find the end effector position and orientation and their time derivatives when the position and velocity of the joints are known. The velocity (differential) kinematics gives the relationship between the joint velocities and the corresponding end effector linear and angular velocities. The velocity kinematics can be written as

$$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{v} \\ \boldsymbol{\omega} \end{bmatrix} = \mathbf{J}(\mathbf{q}) \times \dot{\mathbf{q}}, \tag{2.1}$$

where ***J(q)*** is the manipulator *Jacobian,* and $\dot{\mathbf{x}}$ represents the linear and angular velocities of the end effector with respect to the robot base frame and $\mathbf{q}, \dot{\mathbf{q}}$ represent vectors of joint positions and velocities.

Furthermore, *dynamics* deals with the forces and torques that cause the motion of a system of bodies. Analogously to direct and inverse kinematics analysis, there is direct and inverse dynamic analysis. The equations of motion for a multibody system can be derived, for example with recursive *Newton-Euler*, *Kane* or *Lagrangian* formulations. The *Newton-Euler* formulation incorporates all the forces acting on the individual links in a robot arm. These forces of constraint are useful for sizing the links and bearings during the design stage. The method consists of a forward computation of accelerations and velocities and then a backward computation of the torques and forces in each joint. This method is commonly implemented in robot simulation programs.

Formulating the equation of motion using a set of independent generalized coordinates (i.e. the joint variables) leads to Lagrangian equations of the second type. Unlike the Lagrangian equation of the first type and the Newton-Euler formulation, the second type does not need any forces of constraint between adjacent links, i.e. non redundant variables. The Lagrangian function is defined as the difference between the kinetic (*K*) and potential energy (*U*) of a mechanical system:

$$L = K - U. \tag{2.2}$$

The Lagrange's equations of motion are then formulated in terms of the Lagrangian function

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = Q_i, \tag{2.3}$$

where **q** is a vector consisting of the generalized coordinates. **Q** does not necessarily mean force, but the product **Q** times **q** always has the dimension of work. By placing the coordinate systems for each link according to the *Denavit-Hartenberg* convention

[17], it is easy and straightforward to write the transformation and rotation matrices between the different link frames. Together with the inertia matrices for every link and the Jacobian for the manipulator, it is now possible to rewrite the equation of motion as:

$$\mathbf{Q} = \mathbf{M}(\mathbf{q}) \cdot \ddot{\mathbf{q}} + \mathbf{V}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{B}(\dot{\mathbf{q}}), \tag{2.4}$$

where **M** is the inertia matrix, **V** is the vector of Coriolis and centrifugal forces, **G** is a vector of gravity forces and **B** is a vector of viscous friction forces. **Q** is the vector of forces or torques at the joints. This equation is called the general form of dynamical equations. The manipulator inertia matrix **M** is symmetric and positive definite and therefore always invertible. The off-diagonal elements of **M** represent acceleration coupling effects between joints. For more information on how mathematical models of manipulators are derived, see for example [1] and [63]. The dynamic model is essential for industrial robots in many ways and is very useful for mechanical design of the structure, choice of motors and gears, determination of control strategies, computer simulations of manipulator motions and real time control.

Traditionally, the dynamic model for robot control has been constituted by a rigid model of the structure and elastic model of the robot joints. However, a trend in robotics today is to design lightweight robots, which have a higher load-to-mass ratio. The motive for lightweight robots are cost reduction, reduced power consumption and safety issues. A lighter robot will result in a weaker mechanical structure and hence enhanced elastic effects of the material. Therefore, dynamic models including the elastics of the structure have become more important. For more details on dynamic models including structural flexibilities, see [42] and [70].

2.1.2 Control

A robot is either programmed online in the robot controller system directly or offline by using a simulation model of the robot system. The action of programming the motion of the robot is often a manual process performed by the operator. In material handling tasks it is sufficient to assign only the pick-up and release locations, whereas in e.g. machining tasks the end-effector has to follow a specified path. The goal of the trajectory planning is to generate the time laws for the relevant variables (joint or end effector) starting from a description of the desired motion given by the operator (motion planning). The trajectories must be feasible, i.e. the manipulator must always be able to execute the desired motion. The robot dynamics and kinematics must be taken into account, as do various constraints related to the maximum capacity of the actuators and the robot structure. The trajectory planning is crucial for the performance of the robot and is further explained in section 3.2.

The motion control system ensures that the reference signal given by the trajectory generator is executed. There are many control techniques that can be applied to the control of manipulators. A standard procedure is to measure only the motor angular position and a common architecture for the robot controller is to use both feedback and feedforward control. If the generated trajectories are feasible then will the feedforward controller ideally give zero tracking error on the motor side. Nevertheless, it still requires the use of error contributions between the desired and the actual trajectory. This is due to the considered dynamic model, even though a quite complex one is anyhow an idealization of reality. A common choice for the feedforward controller is to use *computed torque* [61], which in principle means the inverse of the robot dynamics. This leads to very complicated expressions that must be evaluated at high sampling rate. Therefore these expressions are often simplified by updating slowly varying parts less frequently e.g. configuration dependent parts of Coriolis and centrifugal terms. These are instead handled by the feedback controller. For further reading about robot control, see [13], [59] and [61].

2.2 The design process

As in any design process, the main goal in robot design is to develop optimal solutions based on the robot's functional specifications. In Figure 2.4, a stepwise development process for robot design according to Wahrnecke et al. [69] is illustrated. A more general design process model can be found in [14],[49] or [64]. The *product planning and the system analysis* phases include all the steps in the process from an initial product idea to a specification that includes performance data, anticipated cost, quantities, and project development time and cost.

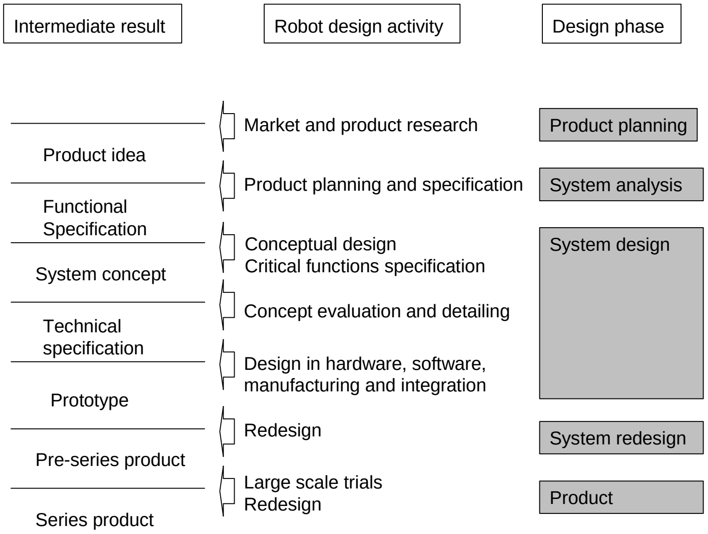


Figure 2.4 Industrial robot design process according to Wahrnecke et al. [69].

The system design phase (conceptual and embodiment design phases according to Pahl and Beitz [49], see Figure 2.5) includes a final functional specification, concept generation, and detailed design of both hardware and software. The conceptual phase includes selection of kinematic structure and estimation of link and joint parameters. An example of another activity in the conceptual design phase is investigation of transmission principle and its components.

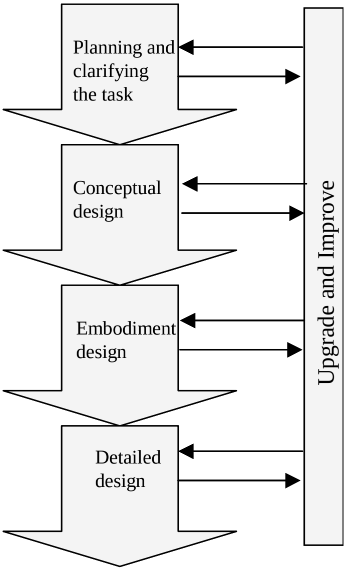


Figure 2.5 Design process according to Pahl and Beitz [49].

After the conceptual phase, one starts to optimize the link and joint parameters and the kinetic performance i.e. motion times. Components for the drive system such as motors, gears, bearings, and couplings are also selected. Cabling, material selection, and dimensioning of axes, housing, base etc are other activities in the detailing phase of *system design*. Design of an industrial robot is a very complex process involving extensive modeling and simulation efforts. Figure 2.6 shows all the activities involved in robot design according to Ölvander et al. [73].

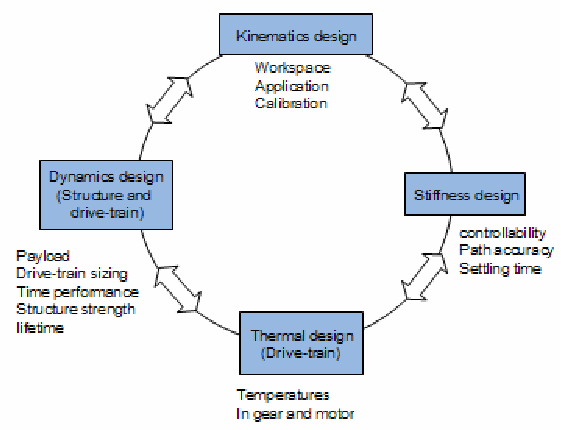


Figure 2.6 Workflow for the robot design process according to Ölvander et al. [73].

This loop is iterated throughout the design process, from the conceptual phase to detailing. Even though all activities in Figure 2.6 are carried out throughout the process, most of the kinematic design activities belong in the early phases of robot design.

The outcome from the system design phase in Figure 2.4 is a complete documentation for the manufacture, assembly, operation, and maintenance of the robot. The system design phase ends with an experimental verification of the robot prototype. Typical activities towards the end of the system design phase are defining the construction layout and preparing preliminary production and assembly documents. The last phase in the development process in Figure 2.4 is *redesign*, which covers all activities to improve the robot system on the basis of detected deficits, quality, performance, cost potentials, requested modifications, and planned product variants.

Many design methods which are applicable to robot design exist in academic literature. It is outside the scope of this thesis to give a detailed outline of the whole body of literature but a few examples will be given. Since the design of robot manipulators begins with dimensioning its various links to meet performance specifications, most of them stress the kinematic layout and its optimization.

The concept of manipulability was introduced by Yoshikawa [67] as a means to measure the ability of robotic mechanisms in positioning and orienting end effectors. Asada [4] introduced the generalized inertia ellipsoid as a tool to measure the capability of changing the velocity of the end effector. Furthermore, Graettinger and Krogh [23] developed the acceleration radius. The acceleration radius is a global generalization of the point wise local measures of dynamic responsiveness proposed by other researchers such as Yoshikwa [67]. The acceleration radius is a uniform lower bound on the magnitude of the acceleration that can be achieved at the end effector from any state (joint position and velocity) in the operating region. Furthermore, Bowling presents a thorough analysis of robotic manipulator dynamic performance in [6].

Ma and Angeles [37] showed how the architecture of a manipulator is optimized under dynamic isotropy conditions. Here, the design strategy is to render the generalized inertia matrix of a manipulator as close to dynamic isotropy and hence achieve optimum dynamic performance. In recent years, Angeles has put forward methods which focus on the kineostatic optimization of manipulators [29]. These methods rely on the minimization of a condition number of the Jacobian matrix over the architectural parameters and the posture variables of the manipulator. The minimization of the condition number of the Jacobian has been put forward by others [58], but Angeles introduced the *characteristic length* in order to cope with matrices which have entries that bear different units since these matrices have singular values of disparate dimensions, which prevents the evaluation of any version of the condition number. However, this concept has been slow in finding acceptance within the robotics community, probably because it lacks a direct geometric interpretation according to Angeles. In [29], the concept is revisited and put forward from a different point of view. The concept of *homogeneous space* is introduced in order to relieve the designer from the concept of characteristic length. Within this space the link lengths are obtained as ratios, their optimum values as well as those of all angles involved being obtained by minimizing a condition number of the dimensionally homogeneous Jacobian.

A more straightforward kinematic measure, normally used by industrial robot manufacturers, is based on the maximum reach (see Figure 2.7) of a robot manipulator. Other kinematic performance measures used by industrial manufacturers that affect the shape of the workspace is the so called *stroke*. The stroke is defined as the offset between maximum reach and minimum reach of the end effector of a robot.

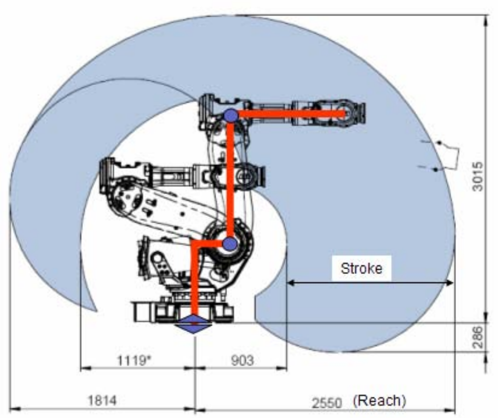


Figure 2.7 Shape, reach, and stroke of the workspace of an ABB IRB6640-185/2.8 robot.

However, kinematic conditioning is not the only criterion for robot design. Additional criteria based on payload, structural behaviour, actuation methods, manufacturing, etc., must be brought into play. These criteria are not addressed in any of the methods mentioned above.

The robot configuration, structure components, and drive train components are preliminarily designed based on either cycle time (time required to execute working cycles) or speed and acceleration requirements. The mass data is obtained based on an initial design. Drive train components, consisting of motors and gears, are dimensioned based on this mass data. The mass data of the initial design is critical for the drive-train dimensioning. Design variables here are gear ratio, rated torque and speed of gears, rated torque and speed of motors, as well as some electrical properties of motors.

End effector linear acceleration or axis rotational speed and acceleration at a large number of predefined points in robot workspace, are normally used as design criteria. This is an iterative process, meaning that the mass data of drive train components are updated while the drive-train dimensioning is in progress. Once the drive train components are correctly dimensioned, structure stress analysis for ultimate strength and fatigue lifetime is conducted. Modifications of structure components proposed by the stress analysis will result in changes in the mass property of the structure components, which in turn requires a new iterative design process of drive train components.

Upon completion of the conceptual design, the data describing the robot manipulator is sufficient for a concurrent mechanical, drive train and controller design in a mechatronic simulation environment. Normally, a large number of robot motion programs, or robot cycles, are used in this phase of the design to ensure the representative robot usage in some dedicated robot applications. This approach is normally referred to as task-, application- or working cycle based robot design.

When it comes to the design of the mechanical components of the drive train for mechatronic products, the way the problem is approached in literature, according to Roos [54], is in general to reduce the problem to selection of the optimal gear ratio for a given electric machine. Pash and Seering [51] derive the gear ratio that optimizes the actuator's output torque for the single axis case with a pure inertial load. The method finds the gear ratio which minimizes the sum of the torques required to drive the inertia of the load and the torques required to drive the inertia of the motor's rotor and the transmission. The method is only applicable if the inertia of the load is constant. In an industrial robot, the inertia varies with the configuration (posture) of the robot and therefore this method is not directly applicable to robot design.

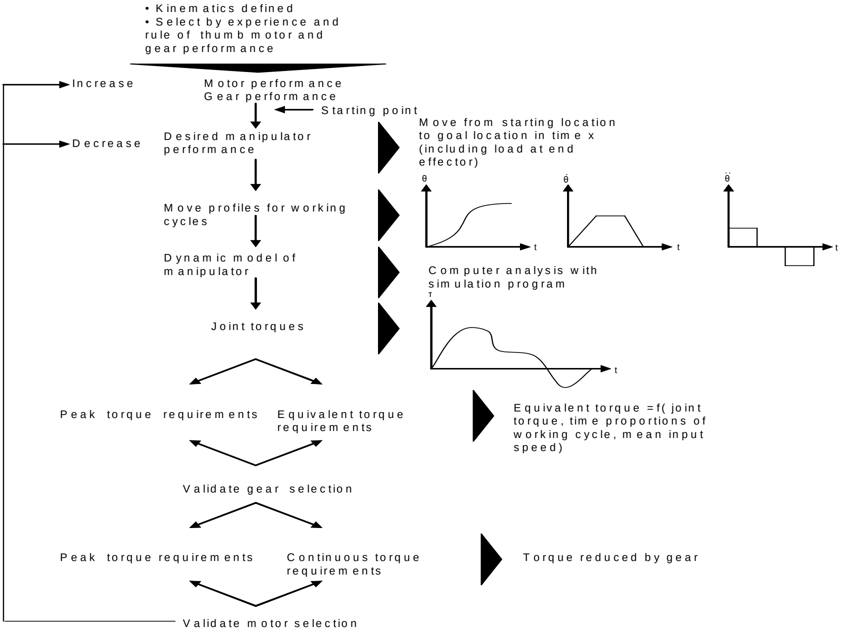
Van de Straete et al. [65] and [66] propose a general method for servo drive selection and optimization when selecting the components from a discrete set of already existing components. Chedmail [11] presents a method for optimum choice of robot actuators for a given trajectory. The optimum actuator set is that which corresponds to the minimization of the global mass of all actuators with respect to maximum pulse torque and the maximum temperature constraints for each one. The method focuses on the motors and considers neither the mass of the gears nor the transmission ratios.

Roos [54] presents a method for optimal selection of electric machine and gearbox, but instead of choosing from a set of alternatives the methodology aims at solving the problem of finding the optimal weight, size and other physical properties of the constituent components, without being limited to already existing components.

The methods mentioned above have been modified and further developed in order to suit the design of robot drive trains as put forward in this thesis. The method is outlined in the next chapter.

3
Drive train design

IN THIS CHAPTER an approach to drive train design for robot manipulators is described. In order to give the reader an overview of robot drive train design, a scheme for computation of the drive train according to Wahrnecke et al. [69] is shown in Figure 3.1. Once a set of components are selected and the mass data of the manipulator exists the performance of the robot is evaluated. If the design does not meet all performance requirements there are two courses of action: choose a better (and often more expensive) design or decrease the performance requirements.



• Kinematics defined
• Select by experience and rule of thumb motor and gear performance
Increase
Decrease
Motor performance
Gear performance
Starting point
Desired manipulator performance
Move from starting location to goal location in time x (including load at end effector)
θ
θ̇
θ̈
t
t
t
Move profiles for working cycles
Dynamic model of manipulator
Computer analysis with simulation program
τ
t
Joint torques
Peak torque requirements
Equivalent torque requirements
Equivalent torque =f(joint torque, time proportions of working cycle, mean input speed)
Validate gear selection
Peak torque requirements
Continuous torque requirements
Torque reduced by gear
Validate motor selection

Figure 3.1 Scheme for computation of drive train according to Wahrnecke et al. [69].

The selection of gearboxes and motors is an iterative process where the starting point is chosen based on experience and rules of thumb. If the performance of the pre-selected motor and gearbox does not meet the desired manipulator performance, there are, as mentioned above, two methods of adaptation:

1. Increase motor and gearbox performance
2. Decrease manipulator performance

Motor performance is increased by modifying an existing one, for instance by adding a fan, or by selecting a better/larger – and thus more expensive –motor. Gearbox performance is increased simply by selecting a more expensive one. Manipulator performance may be decreased in many different ways. For instance, reduced payload or a smaller reach lightens the demands on the actuators. Likewise running the robot at a slower pace means an easier situation for the actuators to cope with. The reference signal to the robot's actuators controls the desired performance in terms of speed and acceleration and is created by the trajectory generator. Decreasing manipulator performance in terms of cycle time (time required to execute a specific working cycle) means decreasing the speed and acceleration of the manipulator along the path. This may be done directly by decreasing the maximum speed and acceleration of the joints or indirectly by decreasing for example the maximum torque generated from the actuators. We will use the latter approach here. For further details about the trajectory generator, see chapter 3.2.

After the joint torques have been calculated in the scheme in Figure 3.1, the peak torque and equivalent torque requirements for the gearboxes are checked. The peak torque requirement refers to the maximum allowed acceleration/deceleration torque of the gearbox stated by the manufacturer. The equivalent torque is a function of speed, joint torques, and time proportion of working cycle. The equivalent torque, together with the rated torque, is then used for lifetime prediction of the gearbox, see equation (3.4). Lifetime is the most important design criterion for gearboxes, but in some cases with high intermittence also temperature must be considered.

The next step is to validate the motor selection. The peak torque requirements of the motor are the sum of the acceleration/deceleration torques of the gearbox plus the torques required to accelerate/decelerate the rotating parts of the motor and the gearbox as well as torque needed to overcome friction. The peak torque of the motor depends on the maximum current of the power supply system and the torque constant of the motor K_t. The maximum speed of the motor depends on the back emf. constant K_e and the maximum voltage of the power supply system. The peak torque requirement prevents the permanent magnets from being demagnetized. The continuous torque requirement for the motor prevents the armature from burnout during operation. The continuous torque is often measured as the root mean square (rms) torque for the working cycle. This torque is compared with the rated torque of the selected motor in order to see if overheating is at risk. Unlike the situation for gearboxes, heat generation is the most critical problem for electric motors and depends on thermal characteristic of the mounting of the motors.

Before outlining the explicit design method used in this thesis, the characteristics of servomotors and cycloid gears are further explained together with a more detailed outline about trajectory generation.

3.1 Characteristics of servomotors and gearboxes in industrial robots

This section provides some general information about common motors and gearboxes in industrial robots.

3.1.1 Electric servo motors

Brushless DC motors are today more or less standard for industrial robots. The main reason for using brushless DC motor is to eliminate the problems due to mechanical commutation of the brushes in a DC motor. The relation between torque and current for a brush less direct-current permanent-magnet motor is

$$T_m = i \cdot K_{t,} \tag{3.1}$$

where K_t is called the torque constant and i is the current. There is also a corresponding relation between voltage and rotational speed

$$e = K_e \cdot \omega, \tag{3.2}$$

where K_e is called back emf constant or voltage constant. The relation between applied voltage, back emf and current is hence given by:

$$u = R \cdot i + K_e \omega. \tag{3.3}$$

From (3.3) one can see that the current, i, will decrease with increasing speed due to the back emf for a constant voltage, u. Hence the available torque T_m will also decrease with increasing speed. The torque constant is manipulated by changing the windings of the motor, but changing the windings will also influence the back emf constant by an equivalent magnitude.

There are several losses in a brushless DC motors. Some of the losses depend on the speed of the motor:

- *Friction* losses in bearings, sealing etc are proportional to the rotational speed. There are also losses due to air drag of the rotor which increase with the square of the speed.
- *Eddy currents*. It is not possible to achieve perfect magnetic flux. Occasionally magnetic fields in the wrong direction will occur and generate currents through rotor and stator laminate. Increases with both rpm and load.

There is also a purely electrical loss which is not dependent on the speed but is nevertheless the largest loss:

- *Ohmic loss* is the resistance times the square of the current and hence increases with the square of the current.

In the Figure 3.2 the characteristics of any electric machine are shown in a graph.

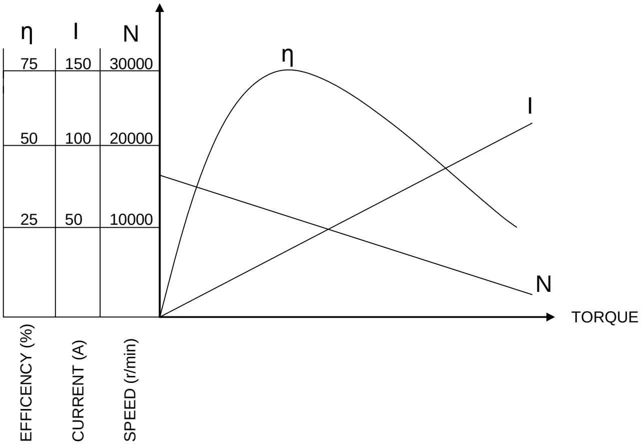


Figure 3.2 Efficiency, current, and speed as functions of torque in an electric motor.

In the graph in the figure above one can see that the current is linearly proportional to the torque and the speed is reciprocally proportional to the torque. The efficiency is low at low loads due to large mechanical friction losses, but as the load increases efficiency improves. After reaching an optimum, efficiency then decreases due to large ohmic losses.

In the Figure 3.3 the intermittent maximum speed-torque characteristics of a brushless DC motor is illustrated. The maximum torque is either limited by maximum amplifier current I_{max_ampl}, or maximum current of the motor without irreversible magnetization I_{max}. The maximum motor speed is limited by the maximum voltage supplied from the amplifier. The inclination of the maximum torque is due to increased back emf with increased motor speed.

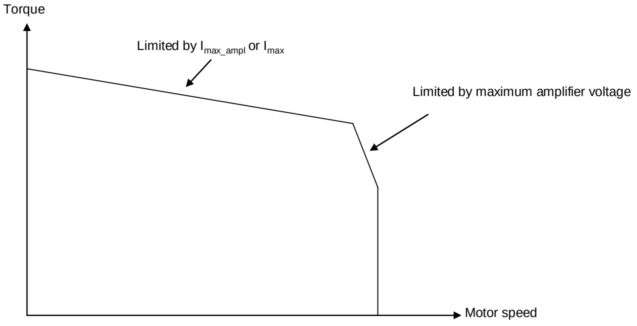


Figure 3.3 Illustration of the intermittent maximum speed-torque characteristics of a brushless DC motor

There is also a corresponding thermal speed-torque curve for brushless DC motors. This curve has lower values of the maximum torque (smaller area under the curve) and limits the torque for use of the motor for an infinite period of time.

3.1.2 Gearboxes

Some of the requirements with regard to robot gears are large reduction in few steps, low inertia, low friction, zero backlash, high stiffness, low lost motion and low weight. Compact speed reducers based on cycloid gearing are therefore an attractive alternative for robot manufacturers [44]. Speed reducers based on cycloid gearing are specified by a rated torque which corresponds to the stamina of the speed reducer, i.e. the larger the rated torque the longer the lifetime under constant load conditions. Together with the rated torque, a rated speed is also included in the equation for predicting lifetime.

$$L_{10h} = K \cdot \frac{\omega}{\omega_0} \cdot \left(\frac{T_0}{T_a} \right)^{10/3} \tag{3.4}$$

where,

L_{10h} is predicted life time in hours,

K is a converting factor,

ω_0 is rated output speed,

T_0 is rated output torque,

ω is averge output speed,

T_a is cubic average output torque.

The equation for prediction of lifetime is derived by the manufacturer [44]. The manufacturer also states a maximum allowed acceleration/deceleration torque (~2-2.5 times the rated torque) as well as maximum allowed speed on output shaft. A momentary maximum torque for emergency situations is also stated.

3.2 Trajectory generator

The planning of optimal dynamic motion is a generic optimal control problem. It can be transformed into a parametric optimization problem by expressing the joint variables as a function of a set of parameters [72]. We assume that the desired path is given in a parameterized form in the task space.

$$\begin{aligned} &\mathbf{x} = \mathbf{f}(s) \\ &\dot{\mathbf{x}} = \mathbf{f}^{'}\dot{s} \\ &\ddot{\mathbf{x}} = \mathbf{f}^{'}\ddot{s} + \mathbf{f}^{''}\dot{s}^{2} \end{aligned} \tag{3.5}$$

Where $\mathbf{f}(\bullet)$ is a n-dimensional (n is the number of DOFs of the robot) vector function and s is a scalar path parameter. The notation $\mathbf{f}^{'}$ is used for differentiation with respect to s, $\mathbf{f}^{'} = \frac{d\mathbf{f}}{ds}$ and $\dot{\mathbf{f}}$ for differentiation with respect to time, $\dot{\mathbf{f}} = \frac{d\mathbf{f}}{dt}$. It is assumed that $\mathbf{f}(s)$ is continuous for all s and that s is strictly increasing. A good candidate for s is the path length. In this case s is the natural path parameter and hence $\mathbf{f}^{'}$ is a unit vector tangent to the path and $\mathbf{f}^{''}$ is a vector of magnitude $\frac{1}{\rho}$ normal to the path, ρ being the radius of path curvature.

The task space coordinates **x** can be described with respect to the joint coordinates **q** according to the following equations:

$$\begin{aligned} &\mathbf{x} = \mathbf{p(q)} \\ &\dot{\mathbf{x}} = \mathbf{J(q)}\dot{\mathbf{q}} \\ &\ddot{\mathbf{x}} = \mathbf{J(q)}\ddot{\mathbf{q}} + \dot{\mathbf{J}}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}, \end{aligned} \tag{3.6}$$

where **x** is an n-dimensional (n is 6 for a 6 DOF robot) vector of task positions, **p** is n-dimensional vector function representing the direct kinematics, $\mathbf{J}$ is the Jacobian matrix and $\dot{\mathbf{J}}$ its time derivative. Combining (3.5) and (3.6) yields the path in joint coordinates,

$$\begin{aligned} &\mathbf{q} = \mathbf{p}^{-1}(\mathbf{f}(s)) \\ &\dot{\mathbf{q}} = \mathbf{J}^{-1}\mathbf{f}^{'}\dot{s} \\ &\ddot{\mathbf{q}} = \mathbf{J}^{-1}((\mathbf{f}^{'}\ddot{s} + \mathbf{f}^{''}\dot{s}^{2}) - \dot{\mathbf{J}}\dot{\mathbf{q}}). \end{aligned} \tag{3.7}$$

The motion of the manipulator is often constrained. The constraints are of two types: the system constraints imposed by the manipulator itself due for example to limits in actuator torque and task constraints given by the task (geometric constraints, path velocity and acceleration, obstacle avoidance etc.). The objective of trajectory planning is to find feasible (and optimal with respect to some objective) trajectories for a given path with simultaneous utilization of the maximal capabilities of the manipulator and without violating any task or system constraints.

A path denotes the locus of points in the joint space or operational space, which the manipulator has to follow. Hence, one can say that a path is merely a geometric representation of motion. A trajectory, on the other hand, is a path on which a time law has been specified, for instance in terms of velocities and/or acceleration in every point.

The motion of a rigid manipulator can be described by

$$\mathbf{Q} = \mathbf{M}(\mathbf{q}) \cdot \ddot{\mathbf{q}} + \mathbf{V}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{B}(\dot{\mathbf{q}}), \tag{3.8}$$

where **M** is the inertia matrix, **V** is the vector of Coriolis and centrifugal forces, **G** is a vector of gravity forces and **B** is a vector of viscous friction forces. **Q** is the vector of forces or torques at the joints. Using (3.7) in (3.8) the equation of motion along the path becomes

$$\mathbf{Q} = \mathbf{a}_1(s)\ddot{s} + \mathbf{a}_2(s)\dot{s}^2 + \mathbf{a}_3(s)\dot{s} + \mathbf{a}_4(s), \tag{3.9}$$

where

$$\begin{aligned} \mathbf{a}_1 &= \mathbf{M}\mathbf{J}^{-1}\mathbf{f}^{'} \\ \mathbf{a}_2 &= \mathbf{M}\mathbf{J}^{-1}(\mathbf{f}^{''} - \dot{\mathbf{J}}(\mathbf{q}, \mathbf{J}^{-1}\mathbf{f}^{'})\mathbf{J}^{-1}\mathbf{f}^{'}) + \mathbf{V}(\mathbf{q}, \mathbf{J}^{-1}\mathbf{f}^{'}) \\ \mathbf{a}_3 &= \mathbf{B}\mathbf{J}^{-1}\mathbf{f}^{'} \\ \mathbf{a}_4 &= \mathbf{G}. \end{aligned} \tag{3.10}$$

Note that **M**, **J**, **J**$^{-1}$ and **G** are functions of **q** and therefore also of s. As a result of the parameterization, there are only two state variables $[s, \dot{s}]$ and (n+1) equations. Hence the problem is reduced to a two-state optimization problem.

When solving the optimization problem a cost function is required. The cost function for time-optimal motion is the time to execute the motion, t$_f$ and the optimization formulation for time-optimal motion can be written as:

$$\begin{array}{ll} \min\ t_f = \displaystyle\int_{s_0}^{s_f} \frac{1}{\dot{s}(s)} ds & \\ s.t. & \\ \text{Motor torques:} & \left|\tau_{motor}(t)\right| \le \tau(\dot{q})_i^{motor,\max}, i = 1,..,n \\ \text{Gear torques:} & \left|\tau_{gear}(t)\right| \le \tau(\dot{q})_i^{gear,\max}, i = 1,..,n \\ \text{Joint accelerations:} & \left|\ddot{q}(t)\right| \le \ddot{q}_i^{\max}, i = 1,..,n \\ \text{Joint velocities:} & \left|\dot{q}(t)\right| \le \dot{q}_i^{\max}, i = 1,..,n \\ \text{Joint positions:} & \left|q(t)\right| \le q_i^{\max}, i = 1,..,n \\ \text{Cut forces:} & \left|F(t)\right| \le F_i^{\max}, i = 1,..,k \\ \text{Cut torques:} & \left|T(t)\right| \le T_i^{\max}, i = 1,..,k \end{array} \tag{3.11}$$

where s$_0$ and s$_f$ are the initial and the final points on the path, respectively. Observing the cost function in (3.11), it is obvious that the path velocity should be as large as possible at every time instant without violating any system constraints. The constraints are here stated symmetrically but may be different in different directions, i.e. acceleration and deceleration. The first two system constraints are limits in motor torque and gear torque. The gear torque refers to the output shaft torque and limits the maximal load on gearboxes. The constraints on motor torque and gear torque are speed dependent, where a higher torque is usually allowed at low speed. The constraints on cut torques and cut forces limit the load on the manipulator structure at sensitive parts and components such as bearings.

The equation of motion described in (3.9) is used to estimate the required gear torques and motor torques. If the required torques are larger than the system constraints admit, the trajectories are suitably time scaled, so that the speed and acceleration dependent torque becomes smaller [59]. Typically the so called shooting methods construct the time optimal velocity profile by forward and backward integration in the $(s,\dot{s})$ plane using maximal possible acceleration $\ddot{s}_{\max}$ to increase the velocity and maximal deceleration $\ddot{s}_{\min}$ to decrease it [72]. Actually the problem is solved by connecting a number of curves inside the admissible region of $(s,\dot{s})$ which is bounded by the system and task constraints. The curves in the $(s,\dot{s})$ plane represent alternately the segments with maximal acceleration / deceleration or with maximal allowable velocity. To see how this is made in detail, see for example [46].

3.3 Design method

Designing drive trains for industrial robots is a complex task where a holistic mechatronic approach is necessary in order to avoid sub-optimization. The goal is to derive parametric models where all interesting properties are functions of some design variables. For instance a motor model is derived where all its properties of interest are functions of the length and the radius of the motor's rotor.

In order to dimension the components of the drive train it is necessary to analyze the torques acting on the gear and on the motor. Assuming a stiff connection between gearbox and motor, the torque T_m required by the motor to follow a given load profile is

$$T_m = (J_m + J_g + J_0)\ddot{q}_l \cdot i + \frac{T_l}{i} + T_{fric} \tag{3.12}$$

where J_m represents the inertia of the rotor of the motor, J_g is the inertia of the gearbox at the motor side of the gear and J_0 represents all the other inertias on the motor side (brake, resolver, shafts etc.) $\ddot{q}_l$ is the acceleration of the output gear shaft and i is the gear ratio. T_l is the load torque on the output gear shaft and T_{fric} is the torque required to overcome the friction in the gearbox and the motor. The torque required to overcome the friction is modeled with coulomb friction and a speed dependent viscous friction as shown in the equation (3.13). If the viscous friction compensation is included here it has to be removed from the equation of motion as it is described in equation (3.8). The static and Stribeck effects are ignored. For more information about friction models see [3], [15] and [68].

$$T_{fric} = sign(\dot{q}_m) \cdot c_{coloumb} + c_{visc} \cdot \dot{q}_m \tag{3.13}$$

The parameters in the friction model are identified by comparison with efficiency charts and no-load running torque charts of the gear boxes. The output shaft of the gearbox is directly connected to the load. Assuming a stiff coupling between the load and the gearbox, gives that the gear output torque, T_g equals the load torque T_l.

3.3.1 Motor model

This chapter outlines how the motor is designed. The goal is a parametric model where all necessary properties of the motor can be derived from the length and the radius of the rotor.

The power losses in the motor are assumed to be dominated by copper losses. The iron losses are therefore ignored and the friction losses are included in T_{fric} as shown above. Selecting electric motors implies that three important criteria must be investigated. First, the continuous torque requirement which prevents the armature from burnout during operation. The continuous torque rating for the motor has to be higher than the rms torque for the working cycle. Furthermore, the maximum torque of the load profile has to be lower or equal to the motor's peak torque rating. Finally, the speed limit of the machine has to be higher or equal to the top speed required by the load profile. The root mean square of the required motor torque is given by,

$$T_{m,rms} = \sqrt{\frac{1}{\tau}\int_0^{\tau} T_m^2 dt} \qquad (3.14)$$

where τ represents the time in order to complete the load cycle. By combining (3.12) with (3.14) the following constraint for the rated torque of the motor is obtained:

$$T_{m,rated} \geq \sqrt{\frac{1}{\tau}\int_0^{\tau} ((J_m + J_g + J_0)\ddot{q}_l \cdot i + \frac{T_l}{i} + T_{fric})^2 dt}\,. \qquad (3.15)$$

The second constraint implies that the maximum torque during the load cycle is lower than the peak torque rating of the motor $T_{m,peak}$.

$$T_{m,peak} \geq \max\left|(J_m + J_g + J_0)\ddot{q}_l \cdot i + \frac{T_l}{i} + T_{fric}\right| \qquad (3.16)$$

In addition the maximum allowed speed of the motor, $\omega_{\mathrm{m,peak}}$ has to be higher than or equal to the maximum speed during the load cycle.

$$\omega_{m,peak} \geq \max\left|\dot{q}_l\right| i \qquad (3.17)$$

We also need to express the parameters of an electric motor as function of its size. Roos [54] presents models based on a scaling approach where data from existing motors is scaled to retrieve data for fictive motors of the same type but of different sizes. We will use the same models here with the constants adjusted against existing motors used in robotics. The rated torque of the motor is modeled as a function of the rotor radius r_m and rotor length l_m,

$$T_{m,rated} = C_m l_m r_m^{2.5} \qquad (3.18)$$

where C_m is constant for a specific motor type and for the same cooling conditions. Most of the time it is the risk of overheating the motor during continuous motion that determines the size of the motor, but for load cycles with short periods of large loads and long periods of low loads the peak torque capabilities of the motor may be dimensioning. The model for peak torque offered by Roos is based on the electromagnetic properties of the motor and is merely a linear combination of the rated continuous torque,

$$T_{m,peak} = C_{pt} T_{m,rated} \tag{3.19}$$

where C_{pt} usually lies between 3 and 6 for permanent magnet machines, assuming passive cooling. Using cooling results in a lower value. The limit of the motor's peak speed is set to a constant value:

$$\omega_{m,peak} = C_{\omega}. \tag{3.20}$$

The inertia of the rotor depends on the length and radius of the rotor according to

$$J_m = C_{mj} l_m r_m^4 + J_0, \tag{3.21}$$

where C_{mj} is again constant for a specific motor type and is adjusted against data of existing motors in robotics. The weight of the motor, m_m is approximated to

$$m_m = \pi r_m^2 l_m \rho_m, \tag{3.22}$$

where ρ_m is the average mass density, a realistic value may be 6000 -7000 kg/m^3.

3.3.2 Gear box model

The gearboxes addressed in this thesis are RV-reduction gears [44], which use a planocentric reduction gear mechanism (cycloid). These are chosen discretely from a list. Each gearbox is modeled as a point mass with transmission inertia and friction. The most important criterion for the gearboxes is the number of possible cycles before they break down. The equation for predicting the number of cycles is provided by the manufacturer [44], and is given in (3.4). Every gearbox also comes with a price tag. Even though the gearbox size is chosen discreetly, the gear ratio can be modified continuously for each gearbox. The relation between gear ratio and transmission inertia for the RV-reduction gears is

$$J_g = C_{inertia} \cdot i^{-p}, \tag{3.23}$$

where the constants $C_{inertia}$ and p are unique for every gear box size. The gearboxes are chosen discreetly, but it is possible to develop similar parametric models, as described above for motors, for the RV-reduction gears and thereby use continues variables in the design optimization.

Finally we will introduce some parameters which implicitly affect the shape of the load cycle. As described in (3.11) there is a system constraint which controls the load on the gear box when generating trajectories. Since the gear torque is the same as the load torque $T_{l,}$, changing this constraint will directly affect the shape of the load cycle. There are three parameters which generate the gear torque constraints implemented as a speed-torque curve. These are $T_{low\ speed}$, permitted torque at low speed, $T_{high\ speed}$, permitted torque at high speed and ω_{max}, maximum permitted speed on the output side of the gearbox. There is also two parameters which decide the boundary between high speed and low speed w_{low_speed} and w_{high_speed}, these parameters are however not included in the optimizations. An illustration of the speed-torque curve is shown in Figure 3.4. The way the constraint on gearbox load is parameterized is inherited from the gearbox manufacturers.

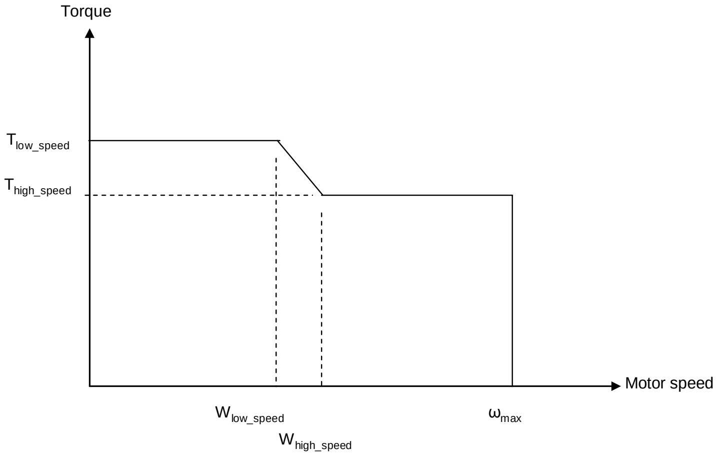


Figure 3.4 Illustration of the speed-torque curve for output side of the gearboxes, i.e. the constraint on maximum torque on the gearboxes in the trajectory planner.

How the method described above can be formalized in an optimization formulation and solved with optimization algorithms is shown in section 5.3.

3.4 Simulation programs

In order to simulate the behavior of the robot the equations described above need to be implemented in a software environment. The optimization strategies presented in this thesis have been applied to simulation models from several different software programs. In paper [I] the simulation model is developed in the symbolic math package Mathematica [40]. The simulation code used in papers [III], [V] and [VII] is developed by ABB Robotics and has a trajectory planner identical to the planner in the real robot system. The trajectory planner creates references (position, speed, acceleration) based on both kinematic and dynamic constraints as discussed earlier. These references are used as inputs to the simulation program which simulates the behavior and performance of the robot. In paper [IV] the complete robot system is modeled in Dymola [19]. Dymola is a commercial simulation package based on the Modelica language. For more information about Modelica see [21] and [43].

4 Optimization

OPTIMIZATION MEANS FINDING the best solution for a problem under given circumstances. Mathematical optimization means that the problem at hand is formalized in a stringent mathematical way and the best solution under the given circumstances is found by using mathematical algorithms. When it comes to *design optimization,* Papalambros et al. give the following definition in [50]:

"Informally, but rigorously, we can say that design optimization involves:

1. The selection of a set of variables to describe the design alternatives.
2. The selection of an objective (criterion), expressed in terms of the design variables, which we seek to minimize or maximize.
3. The determination of a set of constraints, expressed in terms of the design variables, which must be satisfied by an acceptable design.
4. The determination of a set of values for the design variables, which minimize (or maximize) the objective, while satisfying all the constraints."

Design variables are in this thesis also called optimization variables when they are used in optimization formulations. With this viewpoint, one may regard design optimization as a way of thinking and a way of approaching the tasks during the whole design process in any situation where analysis and synthesis is needed.

4.1 Characteristics of objective functions for design optimization based on robot simulations

The characteristics of the objective function and constraints decide which optimization algorithm that is most suitable. For continuous and differentiable functions classical nonlinear methods based on gradients are the most efficient. Optimization as it is employed here is based on computer simulations. A serious problem that arises in optimization using simulations is that the associated functions are not expressed in algebraic or analytical form and the computed results may be non-smooth and noisy.

The objective function shown in the plot in Figure 4.1 is from the optimization problem described in chapter 5.3. One source for the discrete and noisy behavior originates from the trajectory generator used in the simulation program. The two plots (a) and (b) are from running the simulation program with different trajectory generators. The trajectory used in plot (a) has best performance and is used in robots today. The noise is less in plot (b) but the plateaus are wider (due to less frequent sampling times). Also notice how the steps in the "stairway" are inclined away from the optimum (minimum). It is widely appreciated in the simulation-based optimization community that the results of complex calculations like those shown in Figure 4.1 may fail to have the level of precision necessary for a reliable finite-difference approximation to the gradient, ruling out an off-the-shelf finite-difference quasi-Newton code [32]. In this example, derivative estimates with a small finite-difference interval are wildly inaccurate.

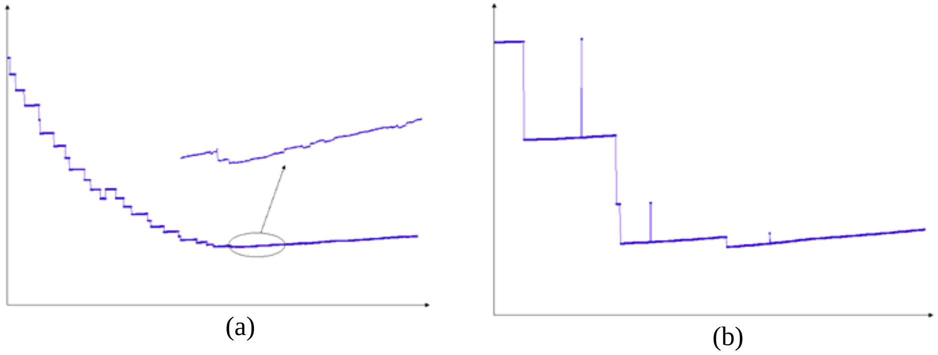


Figure 4.1 Objective function from the problem stated in paper [VII] as a function of one of its variables for two different trajectory generators.

One way to improve the quality of the derivative approximations is to have optimized finite difference procedures for different variables. It is also possible to filter some of the characteristics from the simulation which give rise to the noise. However, a major downside with "tailor made" finite differences and post processing of simulation results is of course that the effort needed to obtain the derivatives for the simulation based problems would be even more substantial.

This is one reason for applying non-gradient methods such as the Complex method. Another reason is that these methods are more robust in locating the global optima in multi-modal search spaces. These methods could be applied to a wide range of problems without any modifications to the algorithms. The disadvantages with direct search methods are sometimes a low convergence rate and a limited size of the problem, i.e. the number of optimization variables (design variables). However, the term convergence rate may be investigated from different perspectives. Gradient based methods such as steepest descent have a faster asymptotic convergence rate than direct search methods. Nonetheless, asymptotic convergence rates do not tell the whole story, especially in the context of the problems to which direct search methods are most applicable.

Another aspect, which is also highlighted by Koda et al. [32], concerning "slowness", is whether this means only the elapsed time for the computer to solve the optimization problem, or the total time from the first formulation of the problem to the point where the designer has obtained a satisfactory result. In many engineering problems, the latter is the most significant.

4.2 Optimization algorithms

There are many different optimization algorithms used in engineering design. The algorithms can be divided into gradient based and non-gradient based methods. The gradient based methods have been thoroughly researched and a considerable body of literature is available on the subject, see for example [45], [48] and [56]. The gradient methods are widely used and are suitable for problems where the gradient of the objective function can be calculated or accurately approximated with a finite difference method at each point. The non-gradient methods on the other hand do not rely explicitly on gradient information in each point. They are therefore of more general use since gradient information is not generally available for engineering design problems. A large number of different non-gradient methods exist.

Direct search methods are one example of algorithms which do not calculate derivatives. Examples of direct search methods include the Nelder-Mead simplex method [47], Box's Complex method [7], the Hooke and Jeeves pattern search [27], and the Dennis and Torczon parallel direct search algorithm (PDS) [18]. A thorough review of direct search methods can be found in [32]. Other non-gradient methods are stochastic methods such as Genetic algorithms (GA) [28] and are comprehensively studied in [24]. Simulated annealing was developed by Kirkpatrick [31] in the early 1980s. More recent methods include Tabu search, developed by Glover [22], response surface approximations [39], Taguchi methods [12], and Particle Swarm (PS) [30].

Since gradient based quasi-Newton methods and Genetic algorithms have been used for benchmark material in this dissertation (paper [II] and paper [VI]) a short description of the nature of the algorithms will be given before presenting a detailed outline of the Complex algorithm.

4.2.1 Gradient based algorithms

All gradient based search methods start with an initial estimate for the optimum point and iteratively improve it until a solution is found. The gradient based methods are suitable for problems with continuous variables and differentiable functions since they operate with gradients of the problem functions. If this is the case or if the gradients can be approximated accurately by finite differences, gradient based methods can be very efficient to use. One of the most effective methods for nonlinearly constrained optimization generates steps by solving *quadratic subproblems* (QP). This *sequential quadratic programming* (SQP) approach can be used in *line search* and *trust region* frameworks. In the line search strategy, the algorithm chooses a direction p_k and searches along this direction from the current iterate x_k for a new iterate with lower function value. The distance to move along $\mathrm{p_k}$ can be found by approximately solving the following one-dimensional minimization problem to find a step length α.

$$\min_{\alpha>0} \; f(x_k + \alpha p_k) \tag{4.1}$$

The search direction often has the form

$$p_k = -B_k^{-1}\nabla f_k \tag{4.2}$$

where B_k is a symmetric and nonsingular matrix. In the steepest descent method (move in the direction in which f changes the most) B_k is simply the identity matrix I, while in Newton's method B_k is the exact Hessian $\nabla^2 f_k(x_k)$. In quasi-Newton methods, B_k is an approximation to the Hessian that is updated at every iteration by means of a low-rank formula. Line search methods and trust region methods both generate steps with the help of a quadratic model of the objective function, but they use this model in different ways. In the trust region, the information collected about f is used to construct a *model function* m_k,

$$m_k(x_k + p) = f_k + p^T\nabla f_k + \frac{1}{2}p^T B_k p, \tag{4.3}$$

for which the behavior near the current point x_k is similar to that of the actual objective function f. Because the model m_k may not be a good approximation of f when x is far from x_k, one restricts the search for a minimizer of m_k to some region (trust region) around x_k. In effect, one chooses the direction and the length of the step simultaneously. If a step is not acceptable, the size of the region is reduced and a new minimizer is found. In general, the step direction changes whenever the size of the trust region is altered.

The general constrained optimization problem can be stated as

$$\begin{aligned} &\min_{x\in R^n} && f(x) \\ &s.t. && c_i(x) = 0, \quad i \in \mathcal{E}, \\ & && c_i(x) \geq 0, \quad i \in \mathrm{I}, \end{aligned} \tag{4.4}$$

where the objective function f and the constraint functions c_i are all smooth, real-valued functions on a subset of $\mathbb{R}^n$ and I and $\mathcal{E}$ are finite index sets of inequality and equality constraints, respectively. The optimization problem can be solved with different approaches. One approach is to combine the objective function and the constraints into a *penalty function* and solve the problem by a sequence of unconstrained problems. For example if only equality constraints are present in (4.4), one can define the penalty function as

$$f(x) + \frac{1}{2\mu}\sum_{i\in\varepsilon} c_i^2(x) \tag{4.5}$$

where μ > 0 is referred to as a penalty parameter. This unconstrained function is then minimized for a series of increasing value of μ, until a solution of the constrained problem is identified to sufficient accuracy. In the *augmented Lagrangian methods* a function which combines the properties of the Lagrangian function,

$$\mathcal{L}(x,\lambda) = f(x) - \sum_{i\in\varepsilon\cup I} \lambda_i c_i(x), \tag{4.6}$$

and the quadratic penalty function (4.5). This so-called augmented Lagrangian function has the following form, when only equality constraints are present in the problem stated in (4.4):

$$\mathcal{L}_A(x,\lambda) = f(x) - \sum_{i\in\varepsilon} \lambda_i c_i(x) + \frac{1}{2\mu}\sum_{i\in\varepsilon} c_i^2(x). \tag{4.7}$$

In sequential quadratic programming methods a QP subproblem is solved to find search directions and a line search or trust region approach is then used to calculate the step size in that direction. More specially in the case of equality constraints in (4.4), the search direction p_k at the iterate (x_k,λ_k) is given by the solution of

$$\begin{aligned} &\min_p \quad \frac{1}{2} p^T W_k p + \nabla f_k p \\ &s.t. \quad A_k p + c_k = 0. \end{aligned} \tag{4.8}$$

The objective in this subproblem is an approximation of the Lagrangian function and the constraints are linearizations of the constraints in (4.4).

4.2.2 Genetic algorithms

Genetic algorithm (GA) is a search algorithm based on the hypothesis of natural selection and genetics. In the methods, each optimization variable is encoded by a *gene* using an appropriate representation. The corresponding genes for all parameters form a *chromosome* (or point) capable of describing an individual design solution. A finite length string, such as a binary string of zeros and ones is usually used to represent each chromosome. Today real value chromosomes are common, whereas original GAs used binary representations of the optimization variables. A set of alternative points (called *population*) at an iteration (called *generation*) is used to generate a new set of points. In this process, combinations of the most desirable characteristics of the current members (*individuals*) of the population are used to generate new populations better than the current ones.

When comparing different points the term *fitness* is used. Fitness is defined using the objective function or the penalty function for constrained problems. The fitness value is calculated for each member of the population, such that the fittest individuals are the ones with the highest likelihood of survival.

The Genetic Algorithm starts with a set of randomly generated individuals (points). Three operators are then needed to implement the algorithm: (i) selection; (ii) crossover; and (iii) mutation. *Selection* is an operator where an old string (point) is copied into the new population according to its fitness. Individuals with higher fitness are more likely to produce offspring. The *crossover* operator corresponds to allowing the selected individuals to exchange characteristics among themselves. Crossover entails selection of starting and ending positions on a pair of mating strings at random and simply exchanging the strings of zeros and ones between these positions. *Mutation* corresponds to selection of a few individuals of the population, determining a location on the string at random and switching the 0 to 1 or vice versa. The foregoing three steps are repeated for successive generations of the population until no further improvement in the fitness is possible, or the number of generation reaches a specified limit. The individual in this last generation with the best fitness value is taken as the optimum.

4.2.3 The Complex algorithm

The modified version of the original Complex method developed by Box [7] in 1965 is an easy-to-use direct search method. This method has been used to a large extent in this work and an outline will be given here followed by a description of a modified version for discrete variables.

The Complex algorithm progresses between each iteration by manipulating a set of points. The number of points *m* in the complex must be such that *m* >= *n* + 1, where *n* is the number of design variables. The number of points *m* used, however, is often more than *n+1*. The geometrical figure in R^n with *m* >= *n* + 1 points is referred to as a complex, see Figure 4.2.

The starting values at each point, **q**, are generated using random numbers.

$$q_{ij} = x_i^l + R_{ij}\,(x_i^u - x_i^l); \qquad i = 1,...,n;\ j = 1,...,m \tag{4.9}$$

Here *j* is an index that indicates a point in the complex and *i* an index that indicates a variable. R_{ij} is a random number in the interval [0,1]. If the implicit constraints are not fulfilled, a new point is generated until the implicit constraints are fulfilled.

The objective function is evaluated at each point. The method progresses by replacing the worst point by a new point obtained by reflecting the worst point through the centroid $\mathbf{q}_c$ of the remaining points by a factor α

$$\mathbf{q}_{new} = \mathbf{q}_c + \alpha(\mathbf{q}_c - \mathbf{q}_{worst}). \tag{4.10}$$

The centroid $\mathbf{q}_c$, is calculated as:

$$\mathbf{q}_c = \frac{1}{m-1}\left(\left(\sum_{j=1}^{m} q_{ij}\right) - q_{ij=worst}\right); \qquad i = 1,...,n. \tag{4.11}$$

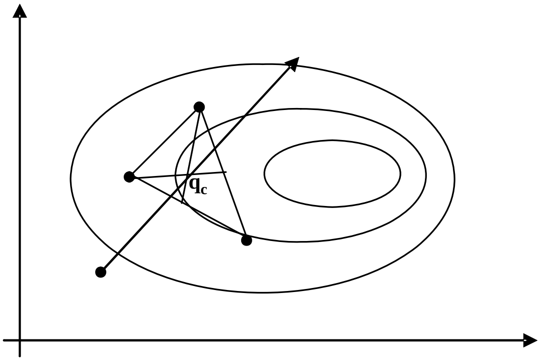


Figure 4.2 Reflection of the worst point through the centroid of the remaining points.

Box recommends α= 1.3. If a point repeats as the lowest value in consecutive trials, it is moved one half the distance towards the centroid of the remaining points

$$\mathbf{q}_{new'} = \mathbf{q}_c + (\frac{\mathbf{q}_{new} - \mathbf{q}_c}{2}). \tag{4.12}$$

4.2.4 The Complex-RF

The Complex-RF optimization method is a modified version of the Complex method developed by Krus [34]. It is modified by introducing some randomization into the search in order to avoid premature collapse of the method

$$\mathbf{q}_{new'} = \mathbf{q}_c + (\frac{\mathbf{q}_{new} - \mathbf{q}_c}{2}) + \mathbf{r}, \tag{4.13}$$

where **r** is calculated according to:

$$\mathbf{r} = r_{fac} \cdot \Delta \cdot \left(x_i^u - x_i^l \right) \left(R - 0.5 \right), i = 1, \dots, n. \tag{4.14}$$

r_{fac} is a randomization factor, x^l and x^u the variable limits, Δ represents the maximum relative spread in the current complex, and is calculated as:

$$\Delta = \max_{i \in I} \left(\frac{\max_{j \in J}(q_{ij}) - \min_{j \in J}(q_{ij})}{x_i^u - x_i^l} \right). \tag{4.15}$$

R is a random variable in the interval [0,1]. This formulation implies that the noise added is a function of the convergence Δ, and the shape of the original design space, i.e. the variable limits.

Furthermore, the formulation makes it possible for the complex to maintain diversity and also regain lost dimensionality. Since the noise is a function of the maximum spread, perturbations could be added to dimensions in which the complex has already converged. This facilitates avoidance of local optima. The randomization factor thus makes the method more robust in finding the global optimum at a cost of somewhat slower convergence. Experiments have shown that a randomization factor of 0.3 is a good compromise between convergence speed and performance [33].

If a local minimum is located at the centroid, the method will continue to move new points towards the centroid, where the whole complex will collapse at one point. In order to avoid this, the new point could gradually be moved towards the best point [33]. This could be expressed as:

$$\mathbf{q}_{new'} = \left(\left((1-a)\mathbf{q}_c + a\mathbf{q}_{best}\right) + \mathbf{q}_{new}\right)\frac{1}{2}. \tag{4.16}$$

Where $\mathbf{q}_{\text{best}}$ is the best point and

$$a = 1 - e^{-\frac{k_r}{b}} \tag{4.17}$$

where k_r is the number of times the point has repeated as the worst and b is a constant that equals to 4. Thus, the more times the point repeats as the worst, the larger a becomes and the new points are moved towards the best point.

It is also possible to include a forgetting factor, which ensures that the complex is made up predominantly of recent parameter sets. This is necessary if the objective function varies over time. In that case, old objective function values become increasingly unreliable and should be replaced by new ones. This is particularly true if the optimization is to be used to optimize parameters in a real process. In this case there may be drift in the parameters of the physical system. Introducing a forgetting factor has also been found to improve the success rate in other situations as well. One of these situations is when the function space consists of plateaus where the complex runs a risk of getting stuck. By introducing a forgetting factor, these plateaus are inclined which helps the complex to “slip off” the plateaus. The forgetting factor is explained in detail in Krus et al. [33]. The suffix “-RF “ is short for randomization factor and forgetting factor.

4.2.5 Complex-RD – A modified version for discrete variables

Many real design problems include a mixture of determining continuous parameters as well as selecting components from catalogs and databases. Discrete problems such as selecting components from a database can be handled by introducing explicit relations between the properties of the different components. The optimization thus works with continuous variables and after the optimization is completed the closest valid alternatives are chosen. However, changing every variable to its closest integer alternative after the optimization has converged may move the solution away from optimum due to dependencies between variables. For instance the choice of gearbox on axis three in a serial manipulator influences the torque requirements of joint two. In Figure 4.3, the feasible area and the feasible points of a discrete optimization problem are marked.

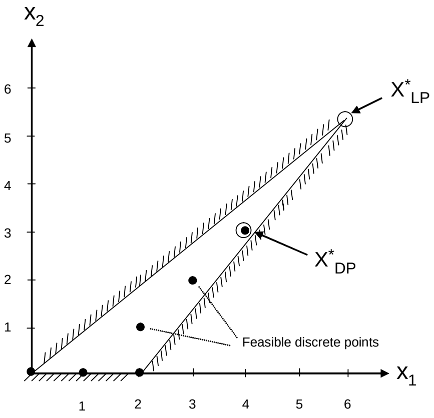


Figure 4.3 Example of feasible area in a discrete optimization problem [36].

The discrete problem has an optimum at $X^*_{DP} = (4, 3)$. If we remove the constraint on discrete variables we get a linear programming problem, an LP-problem, of continuous variables. The optimum to this problem is at $X^*_{LP} = (5.9, 5.3)$.

Deriving the explicit relations between the properties of the components may also be a cumbersome task. For example, if two types of gearboxes, planetary gears and harmonic drives, are available, a common explicit relation between e.g. cost and mass can be hard to derive.

Another way of handling component selection in design optimization is to introduce some function within the objective function which merely takes the integer part of continuous numbers, thus ensuring that the optimization always works with valid alternatives. Unfortunately, this gives rise to plateaus in the objective function space which is troublesome for the optimization algorithm. A function with plateaus due to this is shown in Figure 4.4

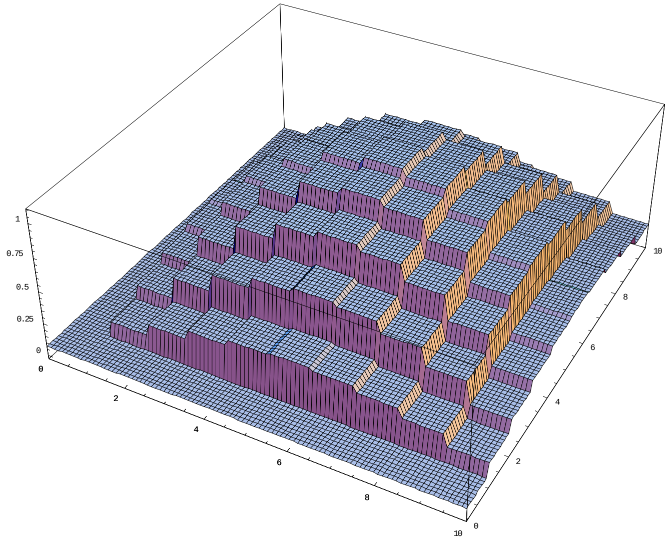


Figure 4.4 Graphic representation of (4.18).

$$
\begin{aligned}
&\min\ f(x,y) = \frac{n\cdot\sin\left(\lfloor x\rfloor\cdot\dfrac{\pi}{10}\right)\cdot\sin\left(\lfloor y\rfloor\cdot\dfrac{\pi}{10}\right)+0,3}{n}\\
&s.t\\
&0\le x\le 10\\
&0\le y\le 10
\end{aligned}
\tag{4.18}
$$

where $n=4$ and $\lfloor a\rfloor$ denotes the integer part of a.

The main idea when solving problems of a discrete character with the Complex-RF is to take advantage of the forgetting factor. Since the forgetting factor gradually reduces the value of old solutions, the points in the complex will not have the same values even if they are all on the same plateau. The complex will thus keep moving and is therefore capable of jumping to the next plateau. The forgetting factor can thus be seen as an introduction of a virtual gradient of the objective function. How the forgetting factor and the randomization factor can be optimized for objective functions with plateaus is discussed in [33].

A way to get around the plateaus in the objective function is to have the Complex always contain valid discrete alternatives. This can be done by moving the “rounding off procedure” within the optimization algorithm. The allowable points for the Complex-RF are made discrete by simply rounding off the values of the discrete variables after every move in the solution space. Consider a problem with n design variables where the first p variables are discrete. The reflection move in (4.10) could then be expressed according to equation (4.19), where Int(x) is function that returns the integer closes to x.

$$\mathbf{q}_{new} = \begin{cases} q_{new,i} = \mathrm{Int}\left(q_{c,i} + \alpha\left(q_{c,i} - q_{worst,i}\right)\right), & i = 1,\ldots,p \\ q_{new,i} = q_{c,i} + \alpha\left(q_{c,i} - q_{worst,i}\right), & i = p+1,\ldots,n \end{cases} \tag{4.19}$$

A schematic representation of the rounding off can be seen as putting a grid on the solution space in those dimensions where the variables are discrete and the feasible region is made up of the points of intersection in the grid, see Figure 4.5. In [26], Hague shows how a similar modification of the complex algorithm can handle the discrete variables when designing civil engineering structures.

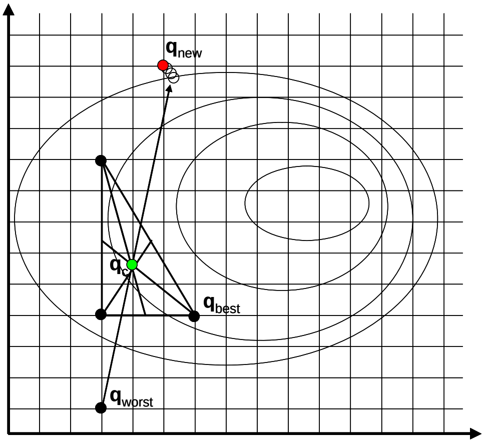


Figure 4.5 Visualization of a reflection move in the discrete complex method.

The modified algorithm, according to (4.19), for discrete problems is called Complex-RD where R stands for randomization factor and D for discrete. The forgetting factor is unnecessary because the plateaus become vertices when the rounding off procedure is within the optimization algorithm. In paper [II], an optimization was performed of both the Complex-RF and the Complex-RD for problems of discrete character. The algorithms were also evaluated against each other, together with a genetic algorithm, on a problem of discrete character based on robot simulations. The Complex-RD proved to be 20 times as effective as the Complex-RF and 60% more effective than a genetic algorithm. However, the GA was used without optimized settings (crossover, mutation) and might have been more effective after some fine tuning.

4.2.6 Complex-RFD – An optimization algorithm for mixed variables

It is easy and straightforward to combine the Complex-RF and the Complex-RD into an algorithm for mixed variable problems. This is done by dividing the variables into two groups, one with continuous variables and one of variables of discrete character. A Complex-RF algorithm with optimum settings for continuous problems works with the continuous part and a Complex-RD with optimum settings for discrete problems works with the discrete variables. How the Complex-RFD can be applied to robot design is presented in paper [III], [IV] and [VII].

4.2.7 Adaptive Complex method

In the Complex method the number of points must be at least one plus the number of variables. However, in order to avoid premature termination and increase the likelihood of finding the global optimum more points are often used at the expense of the required number of evaluations. Nevertheless, the efficiency can be increased if points are removed gradually during the optimization. By gradually removing points as the Complex method converges, the required number of function evaluations is reduced while the likelihood of finding the global optimum is maintained. The pace at which points are removed is determined by how fast the spread (see equation (4.15)) of the complex shrinks. The number of points is eventually reduced to a minimum, i.e. n+1. Paper [VI] shows how this is done in detail. The proposed method shows encouraging results when compared to the Complex method with a fixed number of points and a quasi-Newton method.

4.2.8 Conclusions

Both the Complex-RD and the Complex-RF can handle discrete problems, but the Complex-RD is better suited for the task. An optimization of the optimization method was performed in order to find the most suitable settings of the modified algorithm. The methods were then evaluated against each other on two different discrete problem formulations using a global performance index, see paper [II]. The comparison showed that Complex-RD is the most efficient method. When solving a component based design problem for an industrial robot based on simulation, Complex-RD proved to be about 20 times as effective as Complex-RF. Complex-RD has also been compared to a genetic algorithm and it has proven to be at least 60% more effective than the GA in both the mathematical test problem and the robot design problem.

By gradually removing points as the Complex method converges the required number of function evaluations is reduced while the likelihood of finding the global optimum is maintained. Hence, the adaptive complex method outperforms the original complex method in all problems studied in paper [VI]. In a simulation based optimization problem the adaptive complex algorithm also outperforms the gradient based quasi-Newton method both in terms of finding a better solution and in the number of simulation calls needed.

One difficulty when using the off-the-shelf finite difference quasi-Newton code together with simulation is to estimate the derivatives accurately. The gradient methods often need extensive parameter tuning in order to work at all. When using a direct search methods such as the Complex method this is avoided and hence the time from formulating the problem until a satisfactory solution is found is reduced.

4.3 Optimization loop

A schematic illustration of the simulation based optimization loop used in this work is shown in Figure 4.6. If a system model in the form of a simulation model is defined, it is possible to use optimization based on simulation.

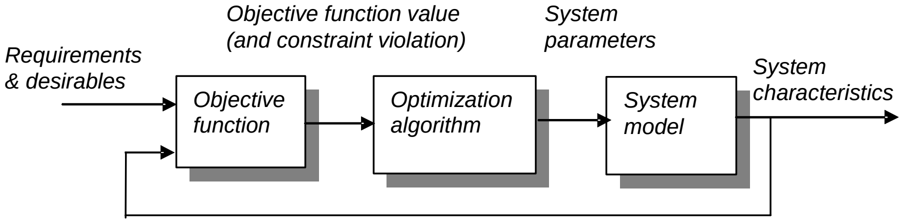


Figure 4.6 Optimization based on simulation.

Using this method, the system is simulated using different sets of system parameters $\boldsymbol{x}_{\mathrm{sp}}$, e.g. mass and rated torque of gearboxes. From each system evaluation, a set of system characteristics $\boldsymbol{y}_{\mathrm{sc}}$, e.g. cycle time and life time, is obtained and using these, an objective function is formulated. In general the simulation v is used to obtain the performance characteristics of the system.

$$\mathbf{y}_{sc} = v(\mathbf{x}_{sp}). \tag{4.20}$$

In the general case, many explicit relations exist between parameters in the system. In fact, in manual design, great efforts are made to obtain explicit design relations and there are many cases where system parameters are coupled and cannot be chosen independently of each other. It is therefore appropriate to define a layer of explicit design relations where relatively few independent design variables are expanded to the full set of system parameters, see Figure 4.7. In Design exampled B and C in the next chapter the gearboxes are chosen discretely from various existing alternatives. There are several properties associated with each gearbox alternative. Hence a change of a gearbox corresponds to several changes in the system parameters such as mass, inertia, cost and rated torque

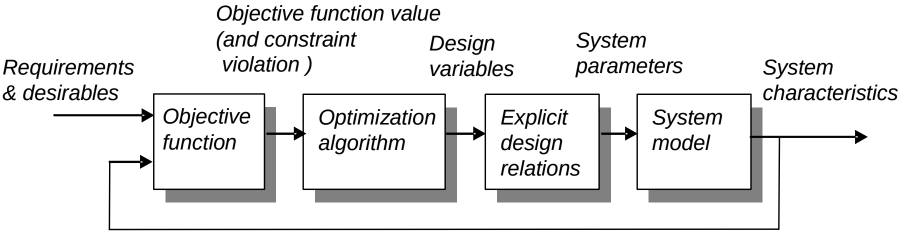


Figure 4.7 Optimization based on simulation with a layer of explicit design relations.

The explicit design relations can be written as:

$$\mathbf{x}_{sp} = u(\mathbf{x}), \tag{4.21}$$

where **x** is the vector of design variables.

The optimization algorithm starts from an initial point (given by the user or based on a random guess). Before running the simulation the model is updated. For instance, in design example C, the motors are considered and one of the design variables is the length of the motor's rotor. The weight of the motor is calculated based on this parameter, i.e. a explicit design relation. The mass of the motor is then updated in a model of the dynamics of the mechanism. When all parameters have been updated a trajectory is generated and a simulation is executed in order to investigate the performance of the design. The objective function, *f*, is then calculated as a function of the system characteristics e.g. cycle time and life time.

$$f = f(\mathbf{y}_{sc}). \tag{4.22}$$

Based on the value of the objective function, a new solution, i.e. a new set of design variables, is generated and the loop is closed and new iterates generated until certain convergence criteria are fulfilled. The whole optimization problem can then be written as:

$$\min \; f = f(v(u(\mathbf{x}))). \tag{4.23}$$

In this thesis, the general optimization problem will be stated in following way: The design variables are denoted by *x*, and all *n* design variables are represented by the vector **x**.

$$\mathbf{x} = [x_1, ..., x_n] \tag{4.24}$$

The objective function is denoted by *f*. Both objective and the $\mathcal{E}$ equality constraints and the *I* inequality constraints are functions, explicitly or implicitly, of the design variables. The general optimization problem could be described in mathematical terms as:

$$\begin{aligned}
&\min_{\mathbf{x}} f\left(\mathbf{x}\right) \\
&s.t. \quad c_i(x) = 0, \qquad i \in \mathcal{E}, \\
&\qquad\;\; c_i(x) \geq 0, \qquad i \in \mathrm{I}, \\
&\qquad\;\; x_j^l \leq x_j \leq x_j^u,\; j = 1, \ldots, n
\end{aligned} \tag{4.25}$$

The problem described in (4.25) should be read as follows: Find the **x** that minimizes $f(\mathbf{x})$ subject to the inequality constraints and the equality constraints and where all n design variables are within their lower (x^l) and upper (x^u) bounds.

5
Design examples

THIS SECTION WILL outline how the presented methods and algorithms have been implemented at different stages in the development of industrial robots, from concept to end user. The first design example belongs to the conceptual design phase, see Figure 2.4, of industrial robots where characteristics such as link dimensions, payload and reach are considered. The next two examples focus on drive train design in industrial robots. Drive train components are considered throughout the system design phase, as it is described by Wahrnecke (see section 2.2), from concept to detail design. However, these examples incorporate properties such as lifetime, and cycle times, which are aspects that are normally considered after the conceptual phases. Finally, it is shown how the ideas in this thesis can be applied to application adapted performance optimization for industrial robots. The means that the robot control is optimized with respect to the thermal and fatigue load on the robot for the programs that the robot performs during repetitive production, i.e. after the robot has been sold and is operated by the end user.

5.1 Design example A

In this example the focus is on the conceptual design stage and on how rather simple mathematical models could be applied together with optimization techniques in order to support the designer. The objective is to determine the size of the gearboxes and arm lengths from an acceleration capability perspective. The object of the study is a three degree of freedom robot modeled in the Mathematica program [40] and optimized using the Complex optimization algorithm. The arm lengths are treated as continuous variables whereas the gearboxes are selected from a list of available units. For more information about how the optimization problem is solved in detail, see paper [I]. With the help of the presented techniques the designer can investigate different conceptual designs and evaluate how changing requirements affect the optimal design.

The optimization problem is formulated so as to minimize the weight of the gearboxes, by choosing different discrete gearboxes, and changing the lengths of the arms continuously, subjected to a few requirements on reach, acceleration, and payload capacity. The acceleration should be achieved in the x, y, and z direction of the base frame. The acceleration requirement is set to 7.35 m/s^2 (0.75 g) at the predefined points in the workspace see Figure 5.1.

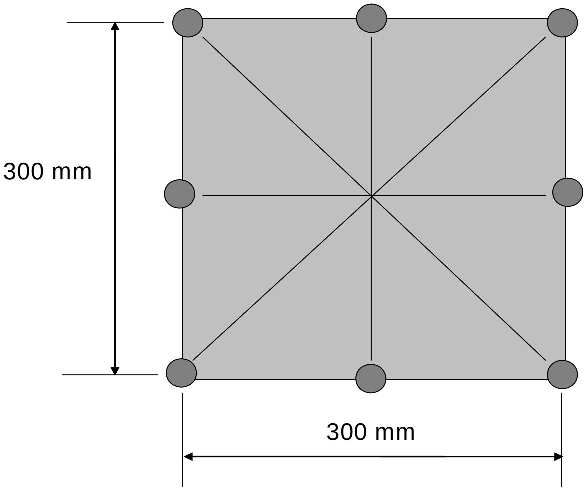


Figure 5.1 Pre-defined points in workspace for performance evaluation.

Furthermore, the robot should have a payload capacity of 100 kg and a minimum reach of 2.5 meters. The available gearboxes for the application are chosen from a list of alternatives. All the gearboxes are from the same manufacturer and are of the same type. In order to handle an increase in output torque, a larger gearbox is thus needed and hence mass increases.

Example of a possible trade-off study with this approach is shown in Figure 5.2. The payload requirement is gradually decreased in order to investigate at which point the change in payload has an effect on size of the gearbox and hence the objective function. As the gearboxes represent discrete selections, the sensitivities remain zero until it is possible to change to a smaller gearbox, and then the sensitivity is infinitely large instead. As can be seen in Figure 5.2, gearbox 1 can be changed from 47 kg to 28 kg if the payload requirement were to be decreased to 60 kg. Gearbox 2 can be changed at a payload of 55 kg.

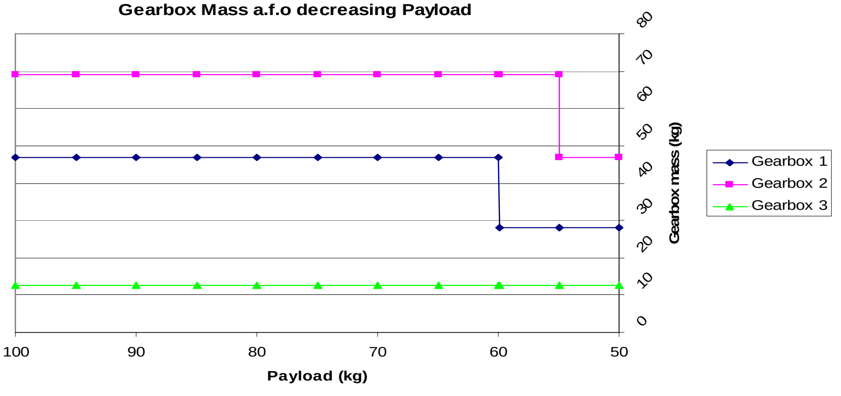


Figure 5.2 The correlation between decreasing payload and size of gearbox.

5.2 Design example B

The design mission here is to select gearboxes for a 4 DOF palletizing ABB-robot, see Figure 5.3.

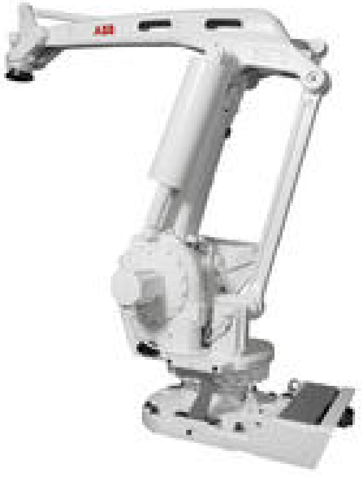
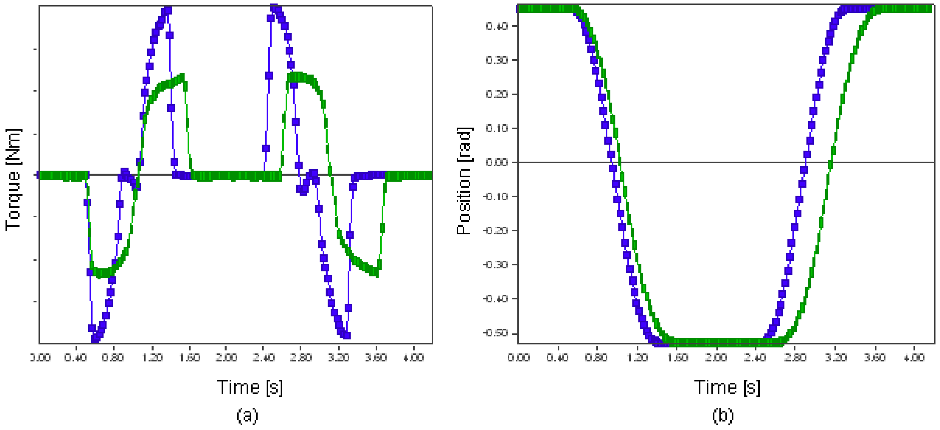


Figure 5.3 Palletizing robot IRB 660.

The gearboxes are major contributors to the overall cost of typical articulated industrial robots. Hence, there is often a desire to decrease the cost, i.e. size, of the gearboxes in order to reduce the overall cost without lowering the requirements with regard to life-time and performance. Even though the design method focuses on choosing the gearboxes, the motors are also included in the simulations. They are, however, not treated as variables. There are twelve design variables in the optimization. Four of them correspond to gearbox alternatives, one for each axis. The other variables generate the speed-torque curve, two variables for every curve and one curve for each axis gives eight variables. There are several properties associated with each gearbox alternative. Hence, a change of gearbox corresponds to several changes in the system parameters such as mass, inertia, cost, and rated torque. Mass and inertia properties affect the dynamics of the mechanism. For example, changing to a heavier gearbox on axis four will increase the mass of the mechanism upstream of the other three axes and hence influence their drive chains. Bigger gearbox inertia means that a larger portion of the torque from the motor will be used to accelerate the rotating parts of the gearbox itself instead of accelerating the load. A change of gearbox means that the dynamic models used in the trajectory generator need to be updated before running the simulation.

We assume there are 20 alternative gearboxes for axes one to three. For the fourth axis we assume there are nine alternatives. This leads to an overall possibility of 72000 combinations. The two variables generating the speed-torque curve on the other hand are handled as continuous variables. Situations may of course occur when components other than the gearboxes limit performance, such as saturation of motor capacity or the power supply. In Figure 5.4a, one can see the torque exerted on the gearbox on axis one during a small part of the cycle for two different torque-speed curves. In Figure 5.4b, the position in radians as a function of time for the two scenarios is shown. Notice how a more limiting speed-torque curve leads to lower torque usage, i.e. a trajectory with less aggressive references (the lower of the two curves in Figure 5.4a) and hence a longer cycle-time, i.e. the curve in Figure 5.4b is shifted rightwards.



(a)

(b)

Figure 5.4 (a): Difference in exerted torque on axis one due to different speed-torque curves. (b): Difference in position of joint one due to different speed-torque curves.

We will here use the optimization approach based on running robot paths. Unlike the approach of evaluating the speed and acceleration capability of a manipulator we will here run actual working cycles. The approach is called cycle based optimization [20].

The objective function is formulated as the sum of the cost of the gearboxes. There are also some constraints: The life time of the gearboxes should exceed 35,000 hours and the time to execute the working cycle (cycle time) should be less than 42 seconds. Optimization variables x_1 to x_4 refer to gearbox choices and are handled as discrete variables. The last eight variables x_5 *to* x_{12} refer to speed-torque curves for the gearboxes. The optimization problem is solved with the Complex-RFD algorithm.

$$
\begin{aligned}
&\min \sum_{i=1}^{4} c_i \\
&s.t \\
&g_i(\mathbf{x}) \geq 35000\,h \qquad , i = 1,\ldots,4 \quad \text{lifetime constraints axis i} \\
&x_j = \{1,2,3,\ldots,20\} \;, j = 1,\ldots,3 \quad \text{variable limits} \\
&x_4 = \{1,2,3,\ldots,9\} \qquad\qquad\qquad \text{"} \\
&x_{low} \leq x_k \leq x_{up} \qquad , k = 5..12 \qquad \text{"}
\end{aligned}
\tag{5.1}
$$

In Figure 5.5 one can see how the gearbox choices change during the optimization for axes 1 through 3. Here the gearbox alternatives converged to alternatives 17, 12, and 5 for axes 1 through 3 respectively. Note that only discrete alternatives for the gearboxes are considered. In Figure 5.6, the values of the speed-torque parameters for axis 1 are shown. These are examples of real value parameters.

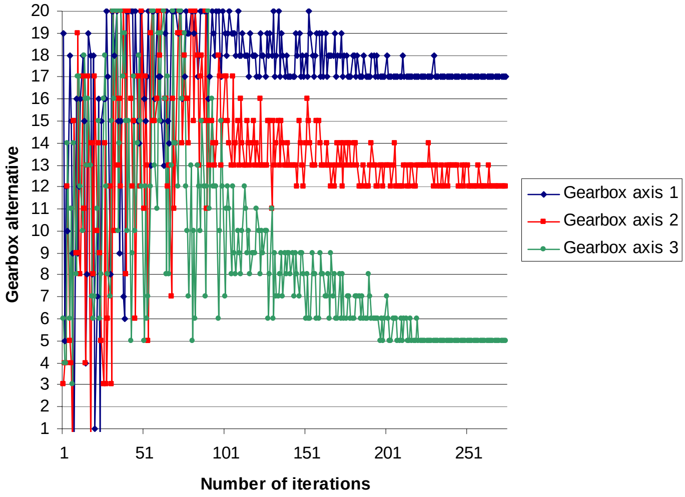


Figure 5.5 Convergence of gearbox alternatives for axis 1, 2 and 3 as functions of the number of iterations.

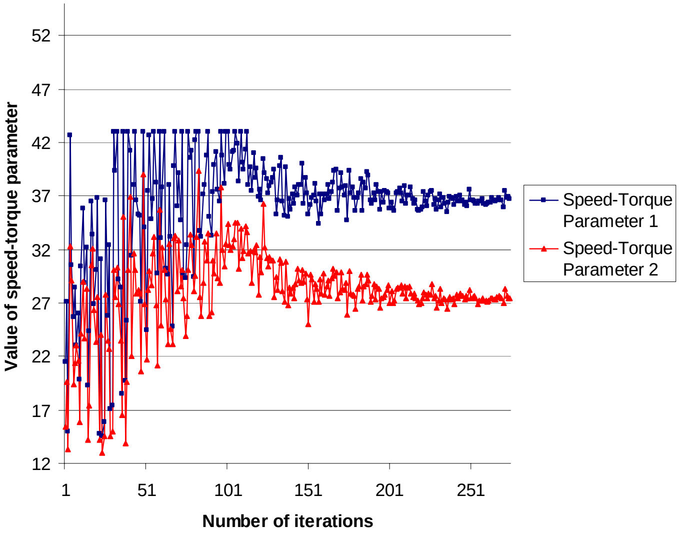


Figure 5.6 Convergence of the parameters controlling the speed-torque curve as functions of the number of iterations.

In order to provide the designer with reliable decision-making information, the constraints on lifetime and cycle-time in (5.1) can be varied. The requirement with regard to cycle-time, for example, can be slowly relaxed and for each relaxation a new optimization problem is solved. This would give information about how the cost of the gearboxes could be decreased with increasing cycle-time. By doing this, the designer can get a feeling for the correlation between gearbox cost, manipulator performance and lifetime over the entire solution space. With this approach, a set of optimizations has to be performed in order to obtain a trade-off diagram. An alternative method would be to formulate the problem as a multi-objective optimization problem and solve it using for example a multiobjective genetic algorithm as described in Deb [16]. The trade-off diagram below was created by optimizing the problem stated in (5.1) for different constraints on cycle-time and lifetime. The performance in cycle-time has been relaxed in ten discrete steps for constant values of the lifetime constraint. The lifetime requirement has been kept constant on three levels. These levels are 30,000h, 35,000h and 40,000 h. Thus, the curves shown in Figure 5.7 were created based on 30 different optimizations.

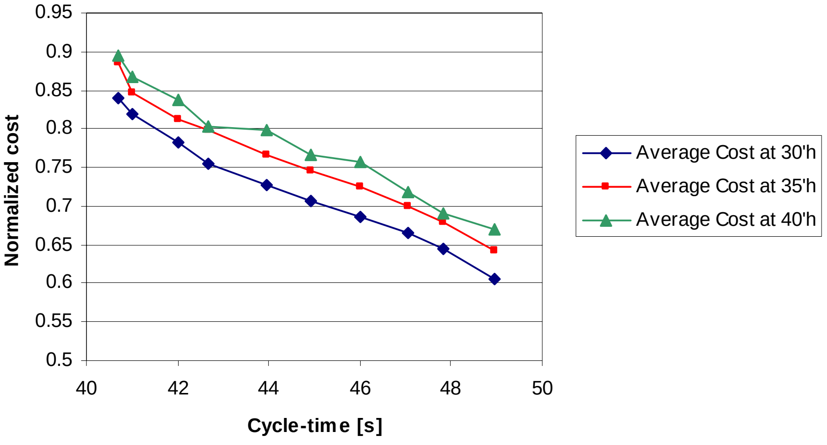


Figure 5.7 Cost of the gearboxes as a function of cycle time. The three different curves correspond to different expected lifetimes of the gearboxes.

By moving horizontally in the diagram in Figure 5.7, one can see how much performance is lowered if one wishes to increase lifetime at constant cost. In the vertical direction, one can investigate how much the cost is increased if one wishes to increase lifetime at constant performance. For more details see paper [III].

5.3 Design example C

In this example the gearboxes and the motors are considered. The design example is based on the design method described in chapter 3.3. In order to illustrate the method we will design the drive trains for axis one and axis two of a six-axis serial manipulator, with a size corresponding to the ABB IRB 6600 shown in Figure 5.8 .

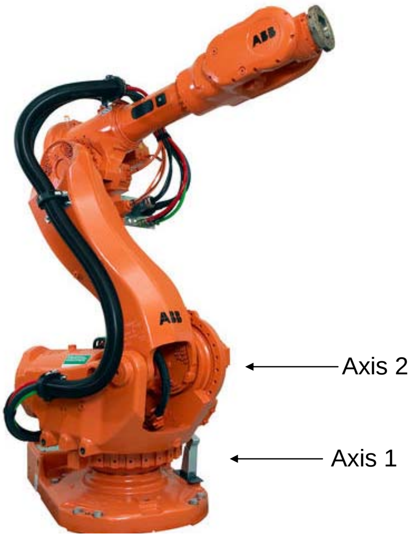


Figure 5.8 ABB robot IRB 6600

We will use the optimization approach based on running robot paths here as well. The cycle time requirements will affect the torque, acceleration, and speed capabilities of the actuators. For the cycle-based approach it is important that the motion programs include all design aspects of interest. How the motion programs could be chosen is not further investigated here but Feng et al. penetrate the question in [20].

We will use a design for cost for the optimization formulation. This refers to a conscious use of engineering resources to ensure that the design process leads to the most economical solution and is modeled by incorporating the cost function, c(x), as an objective in the basic formulation of the design optimization. In this case the cost function consists of the sum of the normalized masses (proportional to the price) of the motors and the normalized price of the gear boxes. The formulation is complemented with a constraint on performance, here using cycle time as a figure of merit, see constraint (I) in (5.2). For each drive train there are another four constraints. The first three originate from the motors' continuous torque, peak torque, and speed requirements, see (II) to (IV) in (5.2). The maximum speed of the motors is set to 420 rad/s (constraint (IV)). The maximum speed of the motors is for simplicity reasons not variable. Constraint (V) exists in order to guarantee a certain lifetime, measured in millions of cycles for the gearboxes. In this example the lower limit is 20 million cycles. Furthermore there are upper and lower bounds on the design variables (VI). There are six variables for each axis and two axes involved in the optimization which results in 12 design variables. The first three design variables manipulate the admissible torque and speed of the gearboxes i.e. the speed-torque curve. The values on the upper and lower bounds for the speed-torque curve are normalized values. The radius of the rotor, r_m, is fixed in the optimization, only the length of the rotor, l_m, controls the size of the motors, and *i* is the gear ratio. By increasing the admissible torque and speed for the gearboxes, the load cycle will be tougher, leading to larger motors in order to avoid conflict with constraints (II) and (III) in (5.2) and gearboxes with higher rated torque due to the constraint on life time (V). The gearboxes are chosen discretely from a list of alternatives.

$$
\begin{aligned}
& \min \sum_{i=1}^{2} \lambda_i \left(\frac{m_{motor,i}}{m} \right) + \beta_i \left(\frac{c_{gearbox,i}}{c} \right) \\
& s.t. \\
& CT \le 3.96\, s && \text{(I)} \\
& \text{for each axis:} \\
& T_{m,rated} \ge \sqrt{\frac{1}{\tau} \int_0^{\tau} ((J_m + J_g + J_0)\ddot{q}_l \cdot i + \frac{T_l}{i} + T_{fric})^2 \, dt} && \text{(II)} \\
& T_{m,peak} \ge \max \left| (J_m + J_g + J_0)\ddot{q}_l \cdot i + \frac{T_l}{i} + T_{fric} \right| && \text{(III)} \\
& 420\,\text{rad/s} \ge \max |\dot{q}_l|\, i && \text{(IV)} \\
& L_{10} = \frac{3600}{CT} \cdot K \cdot \frac{\omega}{\omega_0} \cdot \left(\frac{T_0}{T_a} \right)^{10/3} \ge 20\,\text{million cycles} && \text{(V)} \\
& \left. \begin{array}{l}
0.8 \le T_{low\,speed} \le 1.6 \\
0.8 \le T_{high\,speed} \le 1.6 \\
0.8 \le \omega_{\max} \le 1.6 \\
0.11 \le l_m \le 0.33 \\
50 \le i \le 250 \\
N_{gearbox} \in [1,2,3]
\end{array} \right\} && \text{(VI)} \\
& \lambda, \beta \in \mathrm{R} \,\big|\, \lambda_i, \beta_i > 0, \sum \lambda_i + \beta_i = 1
\end{aligned}
\tag{5.2}
$$

Before every function call in the optimization the simulation model is updated. The current lengths of the motor’s rotor give the current masses according to (3.22). Both the mass of the motors and the mass of the gearboxes are then updated in a model of the dynamics of the mechanism. When all parameters have been updated a trajectory is generated and a simulation is executed in order to investigate the performance of the design. The constraints and the objective function are then evaluated, for instance the rated values of the motors are assessed against the rms values from running the current working cycle (constraint (II)).The constraints are implemented as penalty functions and will affect the objective function value if violated.

The objective of the design optimization is to find suitable motors and gears for different levels of performance. The designer states the level of performance in terms of cycle time and lifetime. The optimization finds the cheapest possible motors and gear boxes which fulfill the stated requirements. In the figures below the progress of different design variables during the optimization is shown. The motion program is in this case designed for evaluating axes one and two, but the optimization method can be used for any kind of motion programs, i.e. paths. The payload in the motion program is 150 kg, but can of course be varied.

In Figure 5.9 one can see how the motor's rated torque and rms torque change during the optimization. Due to constraint (II) in (5.2) the optimization will find a motor with a rated torque that matches the rms torque.

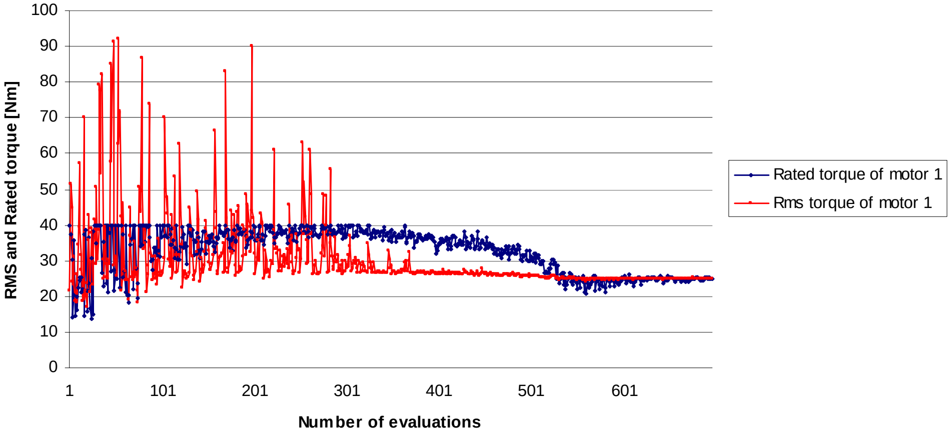


Figure 5.9 Values of the rms torque and rated torque of the motor on axis one during the optimization.

Figure 5.10 shows how the predicted lifetime L_{10} changes for each function evaluation during the optimization.

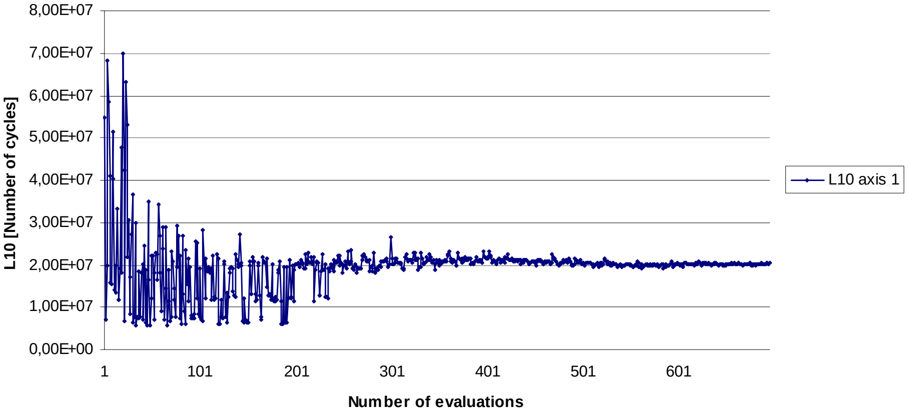


Figure 5.10 Predicted lifetime L10 for each function evaluation during the optimization.

In Figure 5.11 the length of the rotor of motor 1 is shown for each evaluation in the optimization. The length of the rotor determines in turn the size of the motor. The upper and lower bounds on the length of the rotor are 0.33 and 0.11 m respectively.

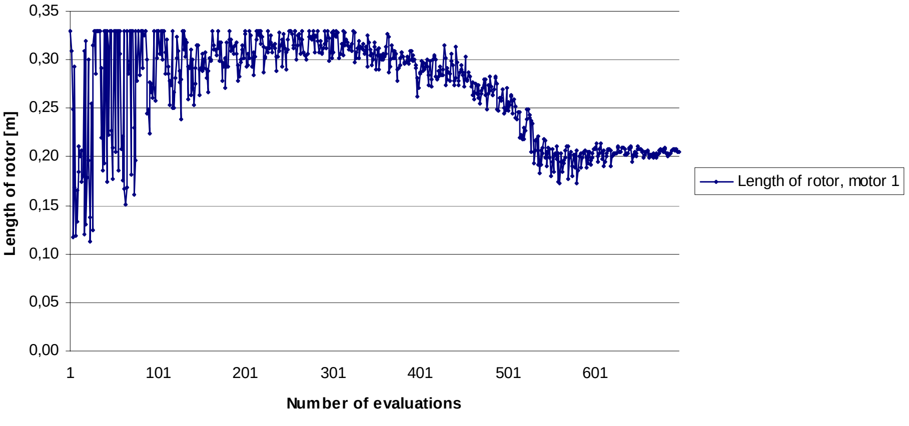


Figure 5.11 Length of rotor for each function evaluation during the optimization.

How the gear ratio converges to an optimal value of 205 is shown in Figure 5.13. However, a larger gear ratio is desirable from a motor size perspective since the rms torque would be lower with an even higher ratio, see Figure 5.12. But since the maximum speed of the motors is 420 rad/s (kept constant) and the required speed of the arm, optimization variable $\omega_{\max}$, is 2.04 rad /s (in order to meet requirement on cycle time) a higher ratio than 205 is not possible in this case.

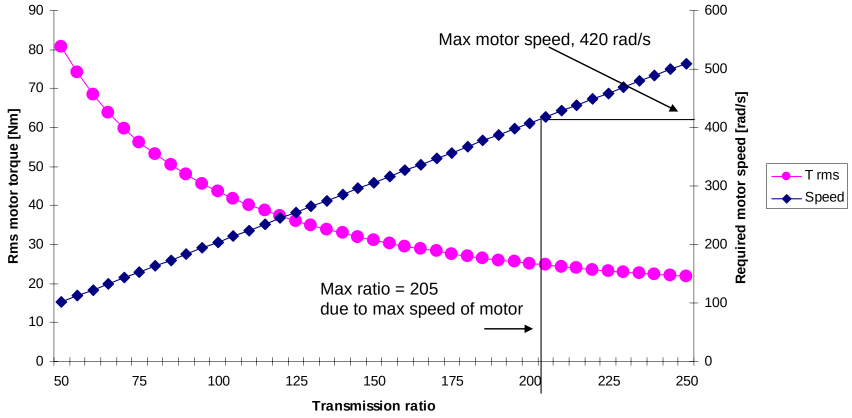


Figure 5.12 Rms motor torque and required max speed of motor as a function of transmission ratio.

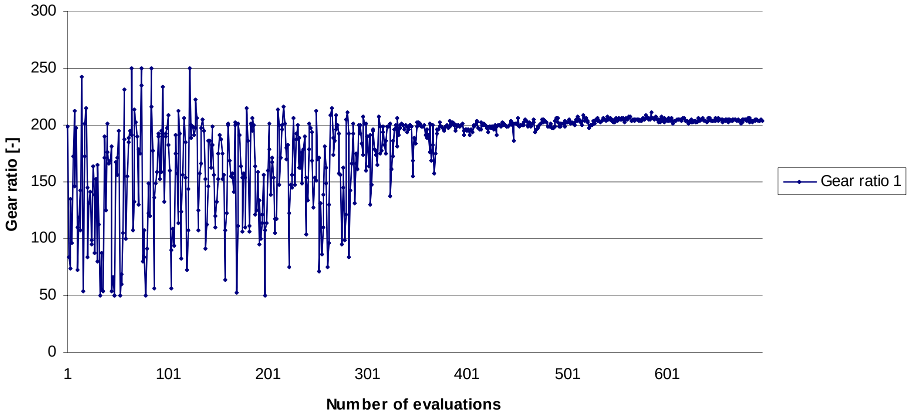


Figure 5.13 Gear ratio for each function evaluation during the optimization.

In Figure 5.14 the total cost of the drive trains for axes 1 and 2 are shown for different levels of performance improvement, i.e. decrease in cycle time. The cost of the motors is calculated from a price per weight ratio and the gearboxes have three different prices. Every point on the graph is an optimal solution from solving the problem stated in (5.2). In order to generate the trade-offs this has been done several times for different requirements on cycle time. The cycle time requirement has been relaxed in seven discrete steps while the lifetime constraint is kept at 20 million cycles. Furthermore, the optimizations have been carried out for several different payloads.

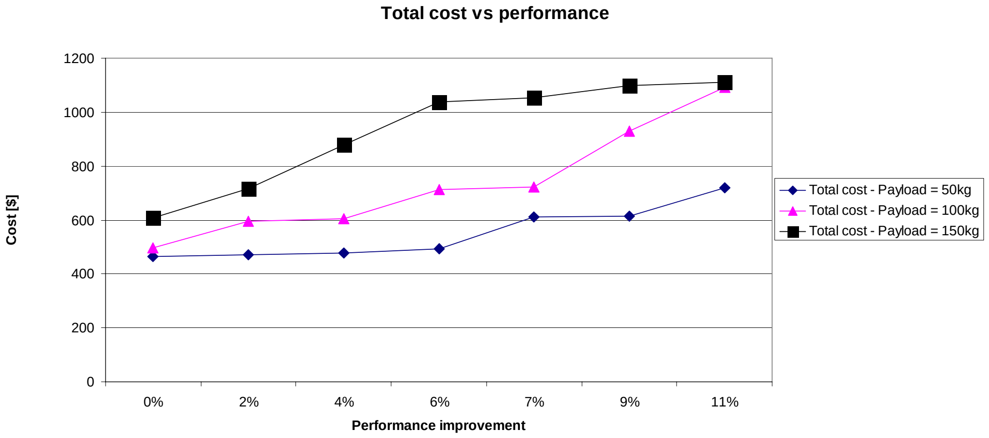


Figure 5.14 Total drive train cost vs. performance for axes one and two.

This optimization problem contains twelve optimization variables and the Complex algorithm requires approximately 700 evaluations in order to solve it. Each evaluation takes about 0.5 seconds on a standard lap top (1.6 GHz). One optimization therefore takes approximately 5 minutes. Fore more details see paper [VII].

5.4 Application adapted performance optimization.

For further efficient use of already installed robots, application adapted performance optimization may be used. This means that the robot control is optimized with respect to the thermal and fatigue load on the robot for the programs that the robot performs during repetitive production. The application adapted performance optimization is based upon the work of Brogårdh et al. in [9].

Another area of potential improvement of robot performance is the motion program itself. Robots have a complex and highly non-linear behavior which makes finding the time optimal program difficult and non-intuitive. Bobrow discussed time optimal path planning for a two link planar motion more than 15 years ago [5]. Here we will use similar ideas on a six axis robot and combine this technique with the application adapted performance optimization concept

By finding the time optimal robot program for a specific application and adapting the drive system parameters, i.e. the system constraints of the trajectory generator to the specific task with the help of simulation based optimization, a tailor-made control configuration may be achieved which optimally uses its assets. In order to investigate the potential of application adapted performance optimization we have tested the idea on a typical press tending application with an ABB IRB 6650 robot. The optimization of the motion program will be referred to as path optimization, and optimization of the constraints of the trajectory planner, will be referred to as drive system optimization. The first optimization finds the time optimal path between two presses and the second optimization adapts the robot with regard to thermal and fatigue load to the program that the robot performs.

In Figure 5.15 a scheme of the optimization process is shown. Before starting the optimization the robot is simulated with the original motion program and the default settings of the drive system configuration parameters. The optimization is then made in two steps. As the first step a path parameters optimization is performed to minimize the cycle time, i.e. a search is made for the fastest path for the robot movements. The second step is to optimize the drive system parameters for the robot movements optimized in the first step. In the drive system optimization the fatigue and thermal loads are not allowed to be higher than in the original case.

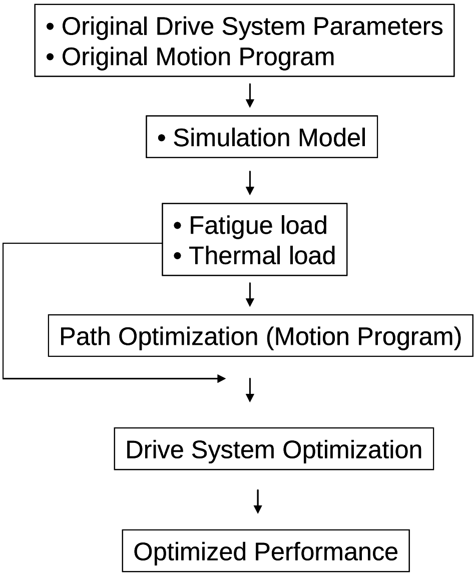

• Original Drive System Parameters
• Original Motion Program
• Simulation Model
• Fatigue load
• Thermal load
Path Optimization (Motion Program)
Drive System Optimization
Optimized Performance

Figure 5.15 Optimization scheme for application adapted performance optimization

In this press tending application, the robot moves metal sheets from one press to the other at the same time as it rotates the sheets 180 degrees. In the original motion program only axes one and six are used for moving the sheets between the presses. For the path optimization the program is made with three via points as shown in Figure 5.16.

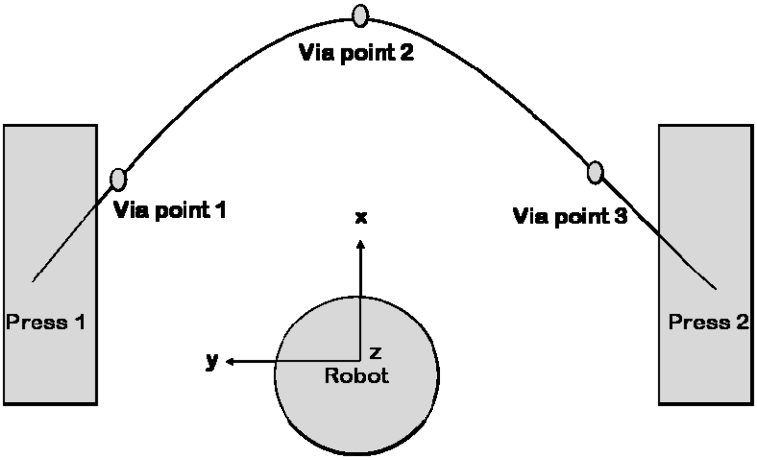


Figure 5.16 Schematic graph of the movement between the presses

There are 12 optimization variables in the path optimization. At the first point all six axes are variables (indexes 1 to 6 in (5.3)), together with one variable for the size of the path zone. (index 7 in (5.3)) of the via point. Path zones exist around the intersection points of every two adjacent path segments in order to make a path smooth in Cartesian space. Within such a zone, the path is allowed to deviate from the segments as long as the path does not leave the zone. For more information about zones in path generation see [46]. Due to symmetry approximations axes one and six are forced to be half way to the end position at the midpoint. Hence only axes 2,3,4 and 5 (index 8 to 11 in (5.3)) are variables at the midpoint together with a variable corresponding to the size of the zone (index 12 in (5.3)). The third point is a reflection of the first point. There are some geometrical constraints: at the beginning of the motion one has to avoid the press and at the midpoint one has to avoid a collision between the metal sheet or the gripper and the robot itself. The objective is to minimize the cycle time.

$$
\begin{aligned}
&\min\ f\left(\mathbf{x}\right) = \text{Cycle Time}\,(CT)\\
&s.t.\\
&x_9 + x_{11} \le 60^o \quad \left(\text{Avoidance of robot}\right)\\
&x_1 \in \left\{0^o, 39^o\right\} \qquad \text{, pos. axis 1}\\
&x_2 \in \left\{-17^o, 85^o\right\} \quad \text{, pos. axis 2}\\
&x_3 \in \left\{-40^o, 70^o\right\} \quad \text{, pos. axis 3}\\
&x_4 \in \left\{-230^o, 230^o\right\}\text{, pos. axis 4}\\
&x_5 \in \left\{-119^o, 90^o\right\} \quad \text{, pos. axis 5}\\
&x_6 \in \left\{0^o, 137^o\right\} \qquad \text{, pos. axis 6}\\
&x_7 \in \left\{0.05, 0.9\right\}\\
&x_8 \in \left\{-17^o, 85^o\right\} \quad \text{, pos. axis 2}\\
&x_9 \in \left\{-40^o, 70^o\right\} \quad \text{, pos. axis 3}\\
&x_{10} \in \left\{-230^o, 230^o\right\}\text{, pos. axis 4}\\
&x_{11} \in \left\{-119^o, 90^o\right\} \quad \text{, pos. axis 5}\\
&x_{12} \in \left\{0.05, 0.9\right\}
\end{aligned}
\tag{5.3}
$$

When the program is executed the trajectory planner generates trajectories based on the programmed path. The trajectory planner calculates trajectories without breaking any constraints, such those as stated in (3.11). These constraints are usually established in the design process and are not adapted exclusively for every application but rather for a large number of different scenarios. There is thus a possibility to gain cycle time by adapting the constraints on motor torque, motor speed etc. to the specific press tending application. In the original case axes one and six are heavily loaded and both the temperature of the motors and the fatigue life of the gearboxes of these axes limit the cycle time. Hence there are constraints in the original optimization of the drive system parameters that prevent the critical axes from being loaded more.

The formulation of the drive system parameter optimization is found in (5.4).

$$\begin{array}{lll} \min\ f\left(\mathbf{x}\right) = \text{Cycle Time}\,(CT) & & \\ s.t. & & \\ g_i\left(\mathbf{x}\right) \geq 5.1\cdot 10^6\ \text{, i=1,2} & \text{Predicted life of gearbox i} & \\ g_i\left(\mathbf{x}\right) \geq 12\cdot 10^6\ \text{,i=3,4,5,6} & \text{Predicted life of gearbox i} & \\ g_{7-12}\left(\mathbf{x}\right) \leq 60^{\circ} C & \text{Gearbox oil temperature axis 1-6} & \\ g_{13-18}\left(\mathbf{x}\right) \leq 90^{\circ} C & \text{Motor temperature axis 1-6} & \\ 0.95 \leq x_{1-18} \leq 1.17 & & \end{array} \tag{5.4}$$

The life time constraints for axes one and two are set to be the same as the original lifetime constraint for axis one. The life time constraints for the other axes are based on the lifetime of axis six in the original cycle. The thermal constraints are given by the maximum allowed motor temperature and the maximum allowed gearbox oil temperature. The motor temperature constraint prevents burn out of the windings. The temperature constraint on the oil in the gearbox prevents situations with too poor lubrication. The ambient temperature is 20°C in the simulations. The constraints in the optimization problem are handled as penalty terms in the objective function, where each penalty term is proportional to the square of the constraint violation.

There are 18 variables in the optimization problem. These variables correspond to the constraints on permissible motor torque, motor speed and gearbox output shaft torque. The limits of the optimization variables are normalized based on the original drive system parameters.

Figure 5.17 shows how the two optimization steps influence the speed of axis six. Notice how the optimizations have increased the acceleration (steeper speed profile). Also notice that the acceleration is reduced at higher speed, which is due to the speed dependence of the motor torque and hence the trajectory planner is forced to reduce the acceleration torque to be able to stay on the path. It is also evident that the maximum speed has been reduced, which allows more motor torque to be used for acceleration, gaining more area under the speed curve during the acceleration phase.

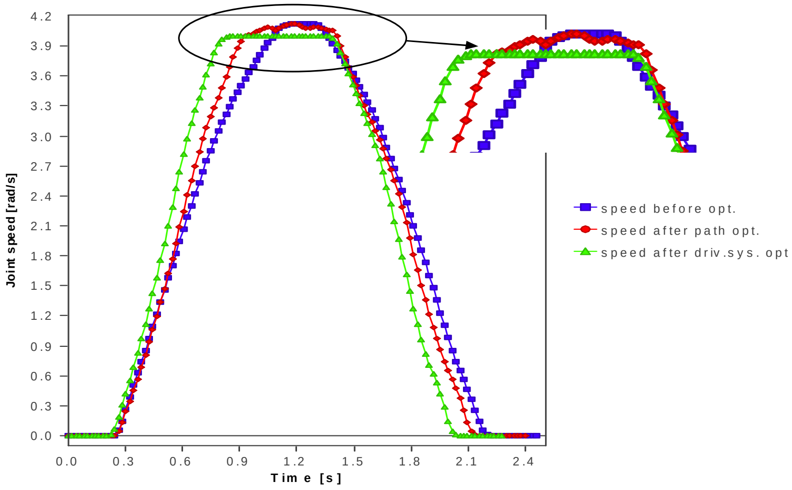


Figure 5.17 Speed of axis six before (blue) and after the path (red) and drive system (green) optimization.

The path optimization contributed to a cycle time reduction of 3.7%. After the drive system parameters were optimized with regard to thermal and fatigue load the cycle time was further reduced by 5.8% without increasing the load on critical components. For more details and results see paper [V].

6
Concluding remarks

IN THE ACADEMIC world there exist a vast number of different optimization algorithms. Which is most applicable to design optimization depends to a large extent upon the problem in question. This thesis discusses optimization techniques suitable for robot design based on simulations. A new method has been developed for handling problem with both continuous and discrete variables. The method has also been improved by adding adaptive characteristics for further efficient design optimization. The proposed optimization methods have been applied to several real engineering problems in order to evaluate the practical use. Robot design itself is investigated and a method for drive train design by running robot paths has been put forward.

During the development process of robots the problem formulations change. In the conceptual phase, questions regarding the outer structure (kinematics) are in focus. As the concepts are developed, one gradually moves from the robot’s outer structure to its inner structure (drive train). Common throughout the process, however, is a mixture of problems of a discrete nature such as selecting gearboxes and bearings and others of a continuous nature such as determining link geometry and tuning control system parameters. Since many problems in robot design are defined by mixed variables a new modified version of the non-gradient Complex-RF method, called Complex-RD, was developed. The Complex-RD is adapted to handling variables of discrete character and can work together with a Complex-RF on problems with mixed variables. The Complex-RD has also been compared to a genetic algorithm and proved to be at least 60% more effective than the GA when solving a robot design problem of a discrete nature. The combination of the Complex-RF and Complex-RD methods has been used in a real engineering design problem for industrial robots.

For the problems addressed in this thesis, the Complex method needs approximately 50-200 function evaluations per optimization variable in order to converge, depending on the problem and convergence criteria. Several different simulation models have been used in this thesis and different simulation environments have different execution times.

The simulation model used in paper [VII] takes about 0.5 seconds to execute and a problem of 12 optimization variables takes about 5 minutes to solve on a standard laptop (1.6 GHz). It is the authors' belief that in order to make design optimization efficient it is important to keep the response as quick as possible since most of the time the designer needs several trials before the optimization works satisfactory. If the response time is too long the work of tuning objective functions, convergence criteria etc in order to get a satisfactory result becomes too cumbersome.

Even though direct search methods have been used successfully in problems with a few hundred variables (e.g. [41]), limitations on the size of problems is a setback for direct search methods such as the Complex. For the problem addressed in paper [III], paper [V] and paper [VII], it is the author's experience that 20-25 optimization variables is the upper limit for the Complex algorithm and still be able to maintain satisfactory convergence performance.

Another issue with the direct search algorithm is slow asymptotic convergence rate compared to gradient methods. However, in the context of the problems addressed here, direct search methods may be more efficient. For functions such as illustrated in Figure 4.1 it is quite possible that a quasi-Newton method with finite-difference gradients may converge quickly, but only to one of the many local minimizers, thus returning a meaningless solution. A further point to be considered regarding slowness is, as stated by Kolda et al. in [32], whether this means only the time needed for the computer to run or the whole time that elapses between formulating the problem, writing the code, and obtaining a satisfactory result. In most engineering design problems the latter is the most important. A common situation in practice is also that the designer wants improvements rather than full-blown optimality. This is often due to the designer only needing one or two correct digits, either because that is sufficient for the application or because the values that can be obtained from the objective function are so inaccurate that seeking higher accuracy would be pointless.

When using numerical simulation and optimization one can always question whether the true optimum is really found. However, from an engineering perspective, the main contribution is not always to identify the global minimizer, but to obtain a foundation for decision making. It is the author's belief that optimization, as it is described here, presents a structured and efficient way of addressing an engineering problem where much insight is gained during the iterative process of solving the optimization problem.

Design often involves finding the most suitable trade-offs between different requirements. This thesis shows how optimization techniques can be used to generate trade-off curves for robot design. By solving the optimization problem with different settings for the constraints, trade-off curves for cost, performance, and lifetime are achieved. By studying these trade-off curves vital insight could be gained about the nature of the problem. These types of curves thus constitute vital support for engineering decision making. For example, it could be seen how higher performance drives the need for larger gearboxes and motors, or how a requirement for longer lifetime also requires larger and more expensive gearboxes.

One of the most critical areas when using design optimization is how to choose a proper evaluation criterion. This is very much the case for industrial robots since it is in their nature to cope with a larger spectrum of tasks. Robot performance may be evaluated in many different ways where different criteria suit different stages during the development process. In this thesis robots are evaluated both by investigating acceleration capabilities as well as running robot paths. In the case of running robot paths, the path itself will have a substantial impact on the result. It is therefore important that the paths include the type of tasks the robot will be used for. In paper [III] the gearboxes are chosen for a robot dedicated for palletizing and a comprehensive palletizing cycle has been used. One should, however, keep in mind that if the robot is evaluated by running paths it is only optimal for the path used during the optimization. The choice of evaluation path is therefore vital to the success of the optimization. How robot paths should be chosen has not been covered in this work but it is a topic for future research to investigate which performance criteria are optimal for industrial robot design at different stages throughout the development process.

The methods developed for drive train design in this thesis are based on stiff system models. Roos concludes in [54] that flexibilities, backlash and controller gains influence both the control performance and the system's torque requirements. How flexibilities and backlash influence the required motor torque is a topic for future research. Furthermore, in order to reach the true system optimum it is necessary to iterate between physical system design and control system design. It is also a goal for future research to include control system design in the optimization loop. If one wishes to minimize characteristics such as energy consumption or the total cost of the robot system it is also important that all components (drive units, transformers, control etc) and their losses are included in the optimization.

Finally, robot control is a key competence for robot manufacturers and is very important in order to get as much performance as possible out of a robot. However, even with the most accurate control the robot's performance cannot surpass the fatigue limits of the robot structure and its mechanical components and the torque, current, and temperature limits of the drive system. Thus, to go further in the optimal use of industrial robots these limits must be tuned to the tasks of the individual robots [8]. One scenario for this is to introduce application adapted robot performance. This means that the controller automatically tunes the drive system parameters to optimize the robot performance for the robot programs that are running. For this, thermal and mechanical fatigue models must be executed together with the dynamic robot models in real time to estimate temperature and mechanical stress in critical components and structures. An offline technique based on this is shown in paper [V]. This adaptation will result in better use of the installed robot and will also make the robot's design more efficient since it will not be critically dependent on worst case movements. It is therefore the author's belief that components should be chosen rapidly by using automated design techniques as described here and the potential performance of the robot for a specific application then pushed towards its limits by using adaptive performance techniques.

References

[1] Angeles J., *Fundamentals of Robotic Mechanical Systems. Theory, Methods and Algorithms*, 2nd. ed., Springer-Verlag, New York, 2002.

[2] Angeles J., A methodology for the optimum dimensioning of robotic manipulators, *Memoria del 5o. Congreso Mexicano de Robótica*, UASLP, San Luis Potosí, SLP, pp. 190-203. México 2003.

[3] Armstrong-Hélouvry B., Dupont P. and Canudas de Wit. A survey of models, analysis tools and compensation methods for the control of machines with friction. *Automatica*, Vol. 30, No.7, pp. 1083-1138, 1994.

[4] Asada, H., A geometrical representation of manipulator dynamics and its application to arm design, *Transactions of ASME, Journal of Dynamic Systems.*, Meas. and Control, Vol. 105, pp. 131-135. 1983.

[5] Bobrow J. E., Optimal robot path planning using the minimum time criterion, *IEEE Journal of Robotics and Automation*, Vol. 4, No. 2, pp. 174-185, 1988.

[6] Bowling A., Analysis of robotic manipulator dynamic performance: Acceleration and force capabilities, PhD-thesis, Stanford University. 1998.

[7] Box M. J., A new method of constrained optimization and a comparison with other methods, *Computer Journal*, 8:42-52, 1965.

[8] Brogårdh T., Present and future robot control development – an industrial perspective, *Annual Reviews in Control*, Vol. 31, No. 1, pp. 69-79, 2007.

[9] Brogårdh T., Ahlbäck M., Bergsjö J., Elfving S., Lager A., Moberg S., Myhr M. and Rylander D., Method for thermal optimization. US Patent 7084595. Available from http://www.patentstorm.us/patents/7084595.html.

[10] Cederfeldt M., Planning Design Automation – A structured Method and Supporting Tools, Doctoral thesis, Chalmers University of Technology, Gothenburg, Sweden 2007.

[11] Chedmail P. and Gautier M., Optimum choice of robot actuators, *ASME Journal of Engineering for Industry*, Vol. 112, pp. 361–367, 1990.

[12] Connor A. M., “Parameter Sizing for Fluid Power Circuits Using Taguchi Methods”, *Journal of Engineering Design*, Vol. 10, No. 4, 1999.

[13] Craig, J.J. *Introduction to Robotics Mechanics and Control,* Addison Wesley. 1989.

[14] Cross N., *Engineering design methods, strategies for product design*, John Wiley &Sons Ltd, Baffins Lane, West Sussex, England,1994

[15] Dhaoudi R., Ghorbel F.H. and Ghandi, P.S., A new dynamic model of hysteresis in harmonic drives. *IEEE Transactions on Industrial Electronics*, Vol. 50, No. 6, pp. 1165-1171, 2003.

[16] Deb K., *Multi-objective Optimization using Evolutionary Algorithms*, John Wiley & Sons, 2001.

[17] Denavit, J. and Hartenberg R. S., 1964, *Kinematic Synthesis of Linkages*,

McGraw-Hill, New York.

[18] Dennis J. E. and Torczon V., Direct search methods on parallel machines, *SIAM Journal of Optimization*, Vol. 1, pp. 448-474, 1991.

[19] Dynasim AB, Dymola, version 6.0d, Lund, Sweden, Nov. 2006 [Online]. Available: http://www.dynasim.se

[20] Feng X., Sander-Tavallaey S. and Ölvander J., Cycle based robot drive train optimization utilizing svd analysis, in *Proceedings of ASME 33rd Design Automation Conference*, Las Vegas, USA, September 4-7, 2007.

[21] Fritzson P., *Principles of Object-Oriented Modeling and Simulation with Modelica*, Wiley-IEEE Press, 2003.

[22] Glover F., Tabu Search - Part I, *ORSA Journal on Computing*, Vol. 1, pp. 190-206, 1989.

[23] Graettinger T. and Krogh B.H., The acceleration radius: A global performance measure for robotics manipulators, *Journal of Robotics and Automation*, Vol 4, No 1, Feb 1988.

[24] Goldberg D. E., *Genetic Algorithms in Search and Machine Learning*, Addison Wesley, Reading, 1989.

[25] Harmonic Drive, www.harmonicdrive.net

[26] Hauge M. I., Optimal frame design with discrete members using the complex method, *Computers and Structures*, Vol. 59, No. 5, pp. 847-858. 1996.

[27] Hooke R. and Jeeves T. A., Direct search solution of numerical and statistical problem, *Journal of the Association for Computing Machinery*, Vol. 8, pp. 212-229, 1961.

[28] Holland J. H., *Adaption in Natural and Artificial System*, University of Michigan Press, Ann Arbor, 1975.

[29] Kahn W.A. and Angeles J., The kineostatic optimization of robotic manipulators: the inverse and the direct problems, *Transactions of the ASME Journal of Mechanical Design*, Vol. 128, No. 1, pp. 168-178, 2006.

[30] Kennedy J. and Eberhart R., Particle swarm optimization, in *Proceedings of the IEEE International Conference on Neural Networks*, Piscataway, NJ, pp. 1942–1948, 1995.

[31] Kirkpatrick S., Gelatt C. D., and Vecchi M. P., “Optimization by simulated annealing,” *Science*, Vol. 220, pp. 671-680, 1983.

[32] Kolda G.T., Lewis R.M. and Torczon V., Optimization by Direct Search :New Perspectives on Some, Classical and Modern Methods, *SIAM REVIEW*, Vol. 45, No. 3, pp. 385–482, Society for Industrial and Applied Mathematics, 2003.

[33] Krus P. and Andersson J., Optimizing Optimization for Design Optimization, *Proceedings of DETC’03 2003 ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Chicago, Illinois USA, 2003.

[34] Krus P. and Gunnarsson S., Numerical Optimization for Self Tuning Electrohydraulic Control Systems. *Proceedings of JHPS International Symposium on Fluid Power*, Tokyo, Japan. 1993.

[35] Larsson T., Multibody Dynamic Simulation in Product Development, Doctoral thesis, Department of Mechanical Engineering, Luleå University of Technology, Sweden, February 2001.

[36] Lundgren J., Rönnqvist M. and Värbrand P., *Optimeringslära*, Studentlitteratur, Lund, 2003.

[37] Ma Ou and Angeles J., Optimum Design of Manipulators Under Dynamic

Isotropy Conditions, *Proceedings of IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 470 – 475. 1993.

[38] Makkonen P., On Multi Body Systems Simulation in Product Design, Doctoral thesis, Department of Machine Design, Royal Institute of Technology. Sweden. 1999.

[39] Marvis D. and Qiu S., An Improved Process for the Generation of Drag Polars for use in Conceptual/Preliminary Design, in *Proceedings of SAE World Aviation Conference*, San Francisco, USA, October 19-21,1999.

[40] Mathematica, http://www.wolfram.com/products/mathematica/index.html

[41] Meza J. C., Judson R. S., Faulkner T. R and Treasurywala A. M., A comparison of a direct search method and a genetic algorithm for conformational searching, *Journal of Computational Chemistry*., Vol. 17, pp. 1142-1151, 1996.

[42] Moberg S., On modeling and control of flexible manipualtors, Thesis, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, 2007.

[43] Modelica Association, Modelica A Unified Object-Oriented Language for Physical Systems Modeling, Language Specification, http://www.modelica.org, 1999.

[44] Nabtesco Motion Control Inc, http://www.nabtesco-precision.com/

[45] Nash S. and Sofer A., *Linear and nonlinear programming*, McGraw-Hill, 1996.

[46] Nyström M. and Norrlöf M., Path generation for industrial robots, Technical report LiTH-ISY-R-2529, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, 2003.

[47] Nelder J.A. and Mead R., A simplex method for function minimization, *Computer Journal*, Vol. 7, pp. 303-313, 1965.

[48] Nocedal J. and Wright S. J., *Numerical optimization*, Springer-Verlag, 2006.

[49] Pahl G. and Beitz W., *Engineering Design*, Springer-Verlag, London, 1999.

[50] Papalambros Y. P. and Wilde J. D., *Principles of Optimal Design – Modeling and Computation*, Cambridge University Press, 1988.

[51] Pash K. A. and Seering W. P., On the drive Systems for High Performance Machines, *ASME Journal of Mechanisms, Transmissions and Automation in Design*, Vol. 106, No. 1,pp. 102-108, 1984.

[52] Rehde M., On Vehicle Crashworthiness Design Using Structural Optimization, Doctoral thesis, Linköping University of Technology, Sweden 2004.

[53] Roos F., Spiegelberg C., Relations between size and gear ratio in spur and planetary gear trains, Technical Report, TRITA-MMK-2005:01, Mechatronics Lab, Department of Machine Design, Royal Institute of Technology. Sweden. September 2007

[54] Roos F., *Towards a Methodology for Integrated Design of Mechatronic Servo Systems*. Doctoral Thesis, Department of Machine Design, Royal Institute of Technology, Sweden, September 2007.

[55] Roozenburg N.F.M. and Eekels J., *Product Design Fundamentals and Methods*, John Willey & Sons Inc., England, 1995.

[56] Ruszczynski A., *Nonlinear optimization*, Princeton University Press, 2006.

[57] Ryan R., Keynote speaker at ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Long Beach, California, 2005

[58] Salisbury, J. K., and Craig, J. J., Articulated Hands: Force Control and Kinematic Issues, *International Journal of Robotics Research*, Vol. 1, No. 1, pp. 4–17, 1982.

[59] Sciavicco L. and Sicilano B., *Modeling and Control of Robot Manipulators*, Springer Verlag, 2001.

[60] Simon H., The *Science of the Artificial*, MIT Press, 1969.

[61] Spong W. M. and Vidyasagar M., *Robot Dynamics and Control,* John Willey & Sons Inc., 1989.

[62] Taylor D. L., *Computer* **-** *Aided Design,* Addison **-** Wesley Publishing Company, 1992.

[63] Tsai L. W., Robot Analysis, The Mechanics of Serial and Parallel Manipulators, John Willey & Sons Inc., 1999.

[64] Ullrich K.T. and Eppinger S.D., *Product design and development*, Mcgraw-Hill/Irwin, New York, 2004.

[65] Van de Straete H. J., De Schutter J. and Belmans R., An Efficient Procedure for Checking Performance Limits in Servo Drive Selection and Optimization, *IEEE/ASME Transactions on Mechatronics*, Vol. 4, No. 4, pp 378-386, 1999.

[66] Van de Straete H. J., Degezelle P., De Schutter J. and Belmans R., Servo Motor Selection Criterion for Mechatronic Applications, *IEEE/ASME Transactions on Mechatronics*, Vol. 3, No. 1, pp 43-49,1998.

[67] Yoshikawa, T., Manipulability of Robotic Mechanisms, *International Journal of Robotic Research*, Vol. 4 No. 2, pp. 3–9. 1985.

[68] Waiboer R., Aarts R. and Jonker B., Velocity dependence of joint friction in robotic manipulators with gear transmissions, Multibody Dynamics 2005, ECCOMAS Thematic Conference, Madrid, Spain, 2005

[69] Warnecke H. J., Schraft R.D., Hagele M., Barth O. and Schmierer G., *Handbook of industrial robotics*, Wiley, 1999.

[70] Wernholt E., *Multivariable Frequency-Domain Identification of Industrial Robots*, Doctoral thesis, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, 2007.

[71] Wheelwright S. and Clark K., *Revolutionizing Product Development – Quantum Leaps in Speed, Efficiency, and Quality,* The Free Press, 1992.

[72] Zlajpah L. and Nemec B., Implementation of time-optimal path-tracking control on palletizing robots, In *Proceedings of the IEEE International Symposium on Industrial Electronics*, pp. 861-866. Bled, Slovenia, 1999.

[73] Ölvander J., Feng X. and Holmgren B., Optimal kinematics design of an industrial robot family, sent to *ASME 34th Design Automation Conference*, New York, New York, USA, August 4-6, 2008.