The optimal design of a planar Stewart platform for prescribed machining tasks

Willem Jacobus Smit

Submitted in partial fulfillment of the requirements for the degree of

Master of Engineering (Mechanical Engineering)

in the

Faculty of Engineering

University of Pretoria

January 2000

Acknowledgements

The author would like to thank professor Snyman for his guidance and input into this study. He is a mentor in the true sense of the word. "If there is something difficult you cannot do, there is some easier that you also cannot do".

Thanks are also extended to the National Research Foundation as well as the University of Pretoria for their sponsorship of study.

All honor belongs to our God, who breaths fire into equations and created a universe that they can describe.

Summary

The optimal design of a planar Stewart platform for prescribed machining tasks

by Willem Jacobus Smit

Supervisor: Prof. J.A. Snyman

Department of Mechanical and Aeronautical Engineering, University of Pretoria

Degree: Master of Engineering (Mechanical Engineering)

Keywords: Mathematical optimization, optimal design, Stewart platform, machine design, parallel platform, robot design, dynamic simulation.

Recently parallel platforms, also known as Stewart platforms, have been the subject of much active research because of their distinct advantages over serially linked manipulators. Parallel platforms may have a great impact, especially in the field of machine tools. Parallel platforms are however, not yet commonly used as machine tools. The main reason for this is the lack of a general and rationally based design system that is also easily implementable. The availability of such a system will allow for the set-up of a platform so that, not only will the task be executable, but it will also be performed in an optimum manner according to a criterion specified by the user. This study proposes an easy to use methodology that may by applied to the optimum design of planar parallel platforms for machining applications.

The design methodology presented here is based on mathematical optimization. This approach is simple and intuitive, and all the most important design criteria can be implemented, in some

mathematical form, in the application of the proposed optimization methodology. Six possible design variables are defined, all related to the physical dimensions and placement of the platform for a prescribed task. Different platform designs are studied by using different combinations of design variables, where some design variables are fixed at specific values while the remaining design variables are allowed to vary. Two types of design constraints are considered, namely geometrical constraints that specify physical bounds on the platform size and placement, and secondly, limits on the maximum and minimum allowable actuator leg lengths. The platform design is optimized according to a prescribed criterion. Two design criteria, also called cost functions, are considered in this study. The first is the minimization of the actuator forces as the manipulator executes a prescribed task. The actuator forces are calculated by means of a dynamical analysis software package, *DADS*. The other design criterion is the maximization of the so-called quality index over the prescribed tool path.

The success of mathematical design optimization depends largely on the optimization algorithm that is used to solve the minimization problem. It is shown in this study that the minimization of the actuator forces is a difficult problem to solve by mathematical optimization. Important reasons for this are that some of the cost functions considered here have discontinuities in their gradients with respect to the design variables, and that they are also highly non-convex and non-quadratic. Another difficulty is the presence of numerical noise superimposed on the cost functions. For this reasons the robust and reliable *LFOPC* optimization algorithm, of Snyman was used. This method, although relatively slow to converge, was highly successful in solving the various design optimization problems.

Because of the slow convergence of the method and the computational cost of evaluating the actuator force cost function and gradients when many design variables are involved, it was

decided to attempt to speed up convergence by the use of an approximation method. The approximation algorithm. *Dynamic-Q* also proposed by Snyman, was selected and its application illustrated by solving a design problem with many design variables. This algorithm quickly converged to an acceptable optimum design.

The main conclusion of this study is that the design methodology proposed here can successfully be applied to the design of planar Stewart platforms and may easily be extended to also apply to spatial Stewart platforms. This methodology solves difficult design problems that are inherent to the design of parallel manipulators. Its future implementation in a more comprehensive design and operating system for Stewart platforms to be used for machining tasks is therefore imperative.

Opsomming

Die optimale ontwerp van 'n vlak Stewart platform vir voorgeskrewe masjineringstake

deur Willem Jacobus Smit

Studie-leier: Prof. J.A. Snyman

Departement Meganiese en Lugvaartkundige Ingenieurswese, Universiteit van Pretoria

Graad: Magister in Ingenieurswese (Meganiese Ingenieurswese)

Sleutelterme: Wiskundige optimering, optimale ontwerp, Stewart platform, masjienontwerp, parallel-platform, robotontwerp, dinamiese simulasie.

Parallel-platforms, ook bekend as Stewart platforms, is onlangs die onderwerp van baie navorsing weens die besondere voordele wat hierdie manipuleerders het bo serie-gekoppelde manipuleerders. Parallel-platforms mag in besonder moontlik 'n groot bydrae lewer in die veld van masjinering. Parallel-platforms word egter nog nie algemeen gebruik as masjineringsgereedskap nie. Die hoofrede hiervoor is die tekort aan 'n algemeen- en rasioneel-gebaseerde ontwerpstelsel wat maklik implementeerbaar is. Die beskikbaarheid van so 'n stelsel sal die opstelling van 'n platform moontlik maak sodat, nie alleen die voorgeskrewe taak uitvoerbaar is nie, maar ook sodanig dat die taak optimaal uitgevoer word ten opsigte van 'n voorgeskrewe kriterium. Hierdie studie stel 'n eenvoudige ontwerpsmetodologie voor wat toegepas kan word op die optimale ontwerp van vlak parallel-platforms vir masjineringstoepassings.

Die ontwerpsmetodologie wat hier gegee word is gebaseer op wiskundige optimering. Hierdie benadering is eenvoudig en intuïtief. Al die belangrikste ontwerpskriteria kan in wiskundige vorm in die aanwending van die voorgestelde optimeringsmetodologie geimplementeer word. Ses moontlike ontwerpsveranderlikes word gedefinieer, waar al ses verband hou met die fisiese afmetings en plasing van die platform vir 'n voorgeskrewe taak. Verskillende platform-ontwerpe word bestudeer deur die ontwerpsveranderlikes in verskillende kombinasies te gebruik. Sommige ontwerpsveranderlikes word beperk tot vaste waardes, terwyl die ander onderwerpsveranderlikes toegelaat word om te varieer. Twee tipes ontwerpsbegrensings word beskou, naamlik geometriese begrensings wat fisiese grense plaas op die platform grootte en plasing, en tweedens, word daar limiete op die minimum en maksimum toelaatbare aktueerder-lengtes geplaas. Die platform word geoptimeer volgens 'n voorgeskrewe kriterium. Twee ontwerp kretiria, ook genoem koste funksies, word in hierdie studie beskou. Die eerste is die minimering van die aktueerderkragte soos die platform die voorgeskrewe baan volg. Die aktueerder kragte word bepaal met behulp van 'n dinamiese analise sagteware pakket, *DADS*. Die ander ontwerp kriterium is die maksimering van die sogenaamde kwaliteitsindeks oor die voorgeskrewe baan wat die beitel volg.

Die sukses van wiskundige optimering hang grootliks af van die optimeringsalgoritme wat gebruik word om die minimeringsprobleem op te los. Daar word aangetoon dat die minimering van aktueerder kragte 'n moeilike probleem is om op te los deur middel van wiskundige optimering. Belangrike redes hiervoor is dat sommige van die kostefunksies wat hier beskou word, diskontinuïteite het in hulle gradiënte met betrekking tot die ontwerpsveranderlikes, en verder dat die kostefunksies ook hoogs nie-konveks en nie-kwadraties is. Nog 'n probleem is die teenwoordigheid van numeriese geraas in die kostefunksies. Vir hierdie redes is die robuuste

en betroubare *LFOPC* optimeringsalgoritme van Snyman gebruik. Alhoewel hierdie metode relatief stadig konvergeer, het dit die verskillende optimeringsprobleme suksesvol opgelos.

Omdat die metode stadig konvergeer, en weens die betreklike lang berekeningstyd wat vereis word om die aktueerderkrag kostefunksie en gradiënte te bereken as daar baie ontwerpsveranderlikes betrokke is, is daar besluit om die konvergensie te probeer versnel deur gebruik te maak van 'n benaderingsmetode. Die benaderingsmetode *Dynamic-Q*, ook deur Snyman voorgestel, is gekies en die toepassing daarvan is geïllustreer deur 'n ontwerpsprobleem met baie ontwerpsveranderlikes op te los. Die algoritme het vinnig na 'n aanvaarbare optimumontwerp gekonvergeer.

Die hoofgevolgtrekking uit hierdie studie is dat die voorgestelde ontwerpsmetodologie suksesvol toegepas kan word op die ontwerp van vlak Stewart platforms. Die metodologie kan ook maklik uitgebrei word om ruimtelike Stewart platforms te ontwerp. Hierdie metodologie los moeilike ontwerpsprobleme op wat inherent is aan die ontwerp van parallel-manipuleerders. Dit is noodsaaklik dat hierdie ontwerpsmetodologie in die toekoms geimplimenteer word in 'n meer omvattende ontwerp-en-beheerstelsel van Stewart platforms vir masjineringstake.

Table of contents

Table of figures

Chapter 1: Introduction

Recently parallel platforms, also known as Stewart platforms, have been the subject of much active research and development because of their distinct advantages in many practical applications over serially linked manipulators [20]. In particular machine tool manufacturers are already designing and fabricating platform devices to replace conventional milling machines [1].

A major and essential requirement for the practical use of a planar platform in machining is that, for a prescribed tool path in the workpiece, the design (or set-up) of the platform should be such that the completion of the tool path within the workspace is guaranteed. This problem is addressed in this study. An optimization methodology is proposed and demonstrated which not only yields a design which accommodates the prescribed tool path, but also places the tool path in a manner which minimizes the actuator forces required for the execution of the task.

Introductory to this study this chapter describes general parallel platforms and then discusses the problems that may be encountered in designing such platforms. Work published on the design of parallel platforms are also briefly reviewed. The chapter concludes with a brief overview of the special purpose of the present study. An outline of the presentation in the chapters that follows will also be given in the final section of this chapter.

1.1 Description of a Stewart platform

A Stewart platform, also known as a Gough platform, is a parallel linked mechanism as shown in Figure 1.1. The Stewart platform consists of a rigid mobile end-effector (manipulator) which

is attached to a fixed base via linear actuators. The actuators are attached to the mobile platform and the fixed base with suitable joints, such as universal- or ball-and-socket joints.

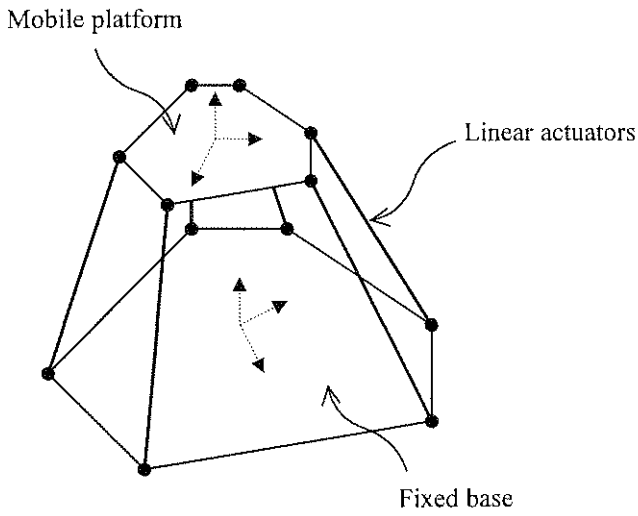

Figure 1.1: Stewart platform

As the actuators extend and contract, the mobile platform moves inside the workspace. The size and shape of the workspace is determined by the minimum and maximum lengths of the actuators, as well as the placement of the actuator end-points on the fixed base and on the mobile platform. The workspace is not freely reachable, i.e. some points in the workspace can only be reached for certain orientations of the mobile platform. This gives rise to the term, dextrous workspace. The dextrous workspace refers to the region in the reachable workspace, for which the mobile platform can have all the orientations from a set of prescribed orientations.

A Stewart platform has several advantages over serially linked manipulators [10, 20]:

1. the load is distributed approximately equally amongst the actuators,
2. loads in the actuators are mostly traction-compression, which make the use of linear actuators ideal,
3. the platform has a high load-to-weight ratio,

4. the position of the manipulator is less sensitive to errors in the actuator sensors, and
5. the platform is more stiff.

These advantages make Stewart platforms ideal for certain applications, such as flight simulators, accurate positioning devices, force measuring devices, milling machines, six degree-of-freedom joysticks, and active vibration dampers [20].

Despite the obvious advantages these platforms have over classical manipulators, it is not commonly used. Their use is limited due to the lack of a good design methods. The architecture of parallel platforms are much different from classical manipulators, therefore the design methods developed for these manipulators cannot in general be applied to parallel platforms.

1.2 Stewart platform design

The design of a Stewart platform for specific practical applications is difficult for several reasons. One reason is that the workspace of a Stewart platform, for a particular geometry and given dimensions, is difficult to determine. The fact that the workspace is much smaller compared to that of a similar sized serially linked manipulator, stresses the importance of having a detailed knowledge of the size and shape of the workspace.

The platform workspace should contain the actual desired workspace in which the practical task is to be performed. But as the platform workspace is fixed relative to the fixed base, this translates to proper placement of the fixed base relative to the desired workspace. This optimal placement of the fixed base relative to the desired workspace is, in general, not straightforward.

Another property of Stewart platforms that complicate the design process, is that there may be singularities inside the workspace. Singularities are positions and orientations of the platform in which it cannot balance external forces, and the platform collapses. In the vicinity of singularities, the actuator forces tend to become infinite. It is therefore important to have a well-conditioned workspace, i.e. a workspace in which the mobile platform will not become unstable or difficult to control[1].

From the above it is clear that a design method is needed that will find the optimal platform design according to some suitable criteria, which takes into account singularities, the placement of the fixed base and workspace requirements. Unfortunately it appears from the literature that very little work has been done on this subject.

1.3 Review of work done on Stewart platforms

The parallel manipulator was first studied by Cauchy [3] around 1813, as an "articulated octahedron". In 1949 Gough [13] used a parallel manipulator to test wear on tires (a parallel linked mechanism is sometimes referred to as a "Gough platform"), and in 1965 Stewart [38] made use of a parallel structure for a flight simulator. In spite of this earlier work, it was not until 1987 that interest into parallel manipulators grew widespread [21] with the study of their potential application to a wide variety of practical areas.

[1] The quality index can be used to determine the condition of the platform in a certain position and orientation. Refer to Appendix C for a discussion on the quality index.

1.3.1 Workspace

Many researchers have published work on the determination of the workspace, but none have yet published work on determining the workspace for a completely general Stewart platform. Du Plessis gives a good review of the work done in this area [8].

Recent work on a new method for workspace determination was done by Snyman et. al. [31] and du Plessis [8]. They present a general numerical optimization technique for determining the workspace of a planar Stewart platform, as well as a spatial 6-3 Stewart platform. The latter platform is depicted in Figure 1.2. This platform has actuator joints on the mobile platform that are paired so that two actuators are effectively joined at one point. The proposed technique makes use of mathematical optimization to find the boundary of the workspace. The work done by du Plessis includes the determination of various types of accessible workspaces.

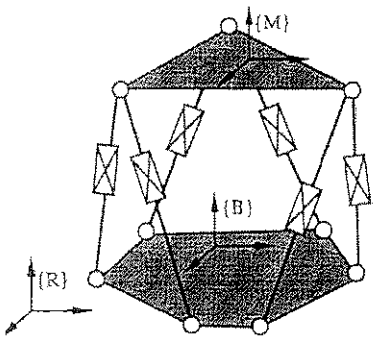

Figure 1.2: 6-3 Stewart platform.

The work of du Plessis was extended for planar Stewart platforms by Hay and Snyman [16] to allow for non-convexity of the workspace and voids in the workspace. Their method also takes into account limits imposed on the orientation of the actuators by the mechanical joints.

The numerical methods proposed by Snyman, du Plessis and Hay are general, and can be extended to determine the workspace of any manipulator.

1.3.2 Singularities

Singularities in the workspace of parallel platforms have been studied [23]. It is known that for singular manipulator configurations, the determinant of the associated Jacobian matrix becomes zero. This property is commonly used to plot singular loci inside the workspace. According to Merlet [20], two accepted approaches used in determining the singular loci, are the numerical continuation method of Haug et. al. [15] and the geometrical approach of Gosselin [11].

The design problem concerning singularities in the workspace is not to determine their location, but to find a suitable platform design for which the mobile platform will not come near these singularities. There are almost always singularities in the workspace, but the platform can be designed so that in performing a prescribed task, it never comes close to these singular configurations.

1.3.3. Design

As already stated, research into the design of a spatial Stewart platform has been limited, but some important work on this subject have been have been done [17, 22]. Most of the published work is on the design of three degree-of-freedom Stewart platforms. Since so little work has been published on this subject, it is useful to also look at the work done on the optimal design of other types of manipulators. Some representative examples of the work done in the area of design are presented in the following subsections.

1.3.3.1 Spatial Stewart platform

The work done by Ji [17] on the design of the 6-3 Stewart platform shown in Figure 1.2, can be used to determine the optimal placement of the fixed platform in a reference system, for a given manipulator task. The design criterion is that the actuator lengths during the manipulator task, should be close to the average actuator lengths (the average of the minimum and maximum

allowable actuator lengths). A placement is penalized if the actuator lengths are outside the specified minimum and maximum lengths at some point during the manipulator task. A placement is evaluated using geometrical relationships, and the optimal placement is found using mathematical optimization. This method cannot be used for the complete design of a platform, as it determines only the optimal placement of the fixed base, given a platform geometry.

In the study of spatial Stewart platform design, Merlet [22] uses a geometrical approach based on the geometrical relationships of the platform to find the optimal platform design. A reduced set of design variables are used in investigating a special case of the six degree-of-freedom platform. For this special case, the only two unknowns are the radii at which all the actuator joints are placed on the fixed base, and radii at which all the actuator joints on the mobile platform are placed. For this case the allowable region of the reduced set of design variables are mapped. This allowable region is a region that includes all platform designs whose workspaces will embody a prescribed workspace geometry, such as for example a tool path trajectory. Out of this allowable region, an exhaustive search reveals the optimal design for some specified criterion. The design criteria that can be used in this approach is maximum accuracy, minimum articular forces for a given load, maximal stiffness in some direction and maximum velocities or accelerations of some point of interest for given actuator velocities and accelerations. This method does not allow for a dynamical analysis of the platform.

1.3.3.2 Three degree-of-freedom parallel platform

Gosselin and Angeles [12] used an analytical approach to find the optimum kinematic design of a planar three degree-of-freedom parallel manipulator. This platform uses rotational actuators, in contrast to linear actuators, as shown in Figure 1.3. They derived the kinematic equations of the platform, as well as the analytical equations describing the design criterion. These equations

are then solved analytically where possible, otherwise numerically. The work does not include a dynamical analysis, although it should be possible to also include dynamical aspects in the analytical design criteria.

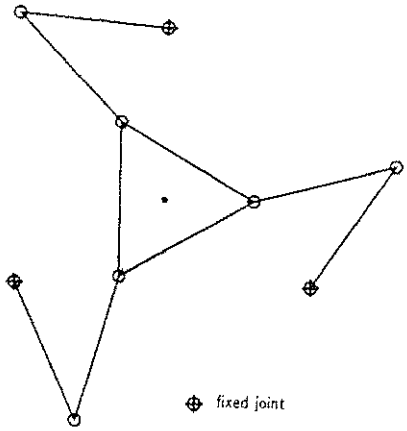

Figure 1.3: Planar Stewart platform with rotational actuators.

Stamper et al. [37] studied a three degree-of-freedom translational platform shown in Figure 1.4. They derived analytical expressions that describe the platform kinematics and the design criterion. The optimal design was found using mathematical optimization to solve the system of analytical equations. The two design criteria used was the maximization of the workspace, and the conditioning of the workspace. This work does not allow for dynamical effects, although it could also be included.

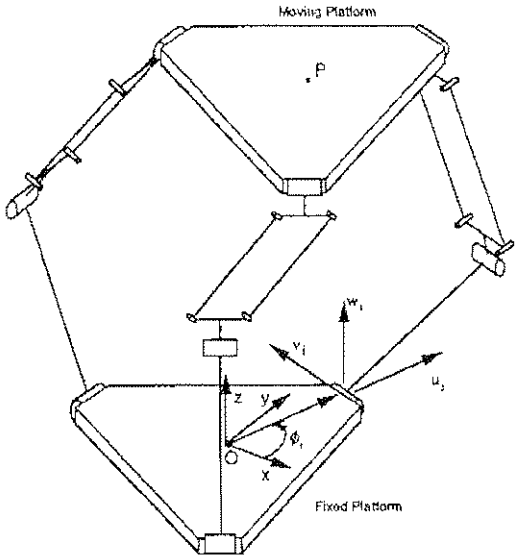

Figure 1.4: Three degree-of-freedom translational Stewart platform.

1.3.3.3 Optimal design of other types of manipulators

This sections briefly investigates methods used to find the optimum designs for some other types of manipulators. These methods may very well be applied to the design of Stewart platforms.

Datoussaïd et al. [6] used a dynamical simulation to predict the behavior of a multibody system. Analytical equations that describe the dynamics of the system, are programmed, and solved numerically by computer. The output of this program is used in a mathematical optimization algorithm which optimize the design parameters. Once the analytical equations are programmed, the process of finding the optimal design is done automatic by the computer. This method was successfully applied to the optimization of an urban railway vehicle.

Takano et al. [39] developed a system called "TOCARD" (Total Computer Aided Robot Design) to design a serial linked robot arm. This system uses the kinematic and dynamic analysis of the robot arm to evaluate objective functions. Some of the objective functions

considered are the workspace, deflection, position accuracy and controllability. The computer program automatically generates and solves the kinematic and dynamic equations for any given robot arm. The program does not find the optimum design, but a user interface allows for easy changes in the design.

Snyman and Berner [2, 30], and Snyman and van Tonder [36] also proposed optimal design methodologies to find the optimum designs of serially linked robotic manipulators. The optimization objectives were the minimization of torque and energy requirements subject to prescribed design constraints.

1.4 Purpose and outline of present study

This thesis proposes a method for the optimal design a planar Stewart platform for prescribed machining tasks. It is a combination of the computer aided kinematic and dynamical analysis approach used by Takano et. al. [39], and the mathematical optimization approach used by Datoussaïd et. al. [6] and Snyman et. al. [2, 30, 40].

The advantages of using a parallel manipulator instead of a serially linked manipulator have already been discussed in section 1.1. These advantages give a parallel manipulator great potential to be applied to the machine tool industry. As already indicated, the platform will be faster, more stiff and more accurate than conventional machine tool equipment. One reason that parallel platforms are not yet generally applied in the machine tool industry, is the lack of a complete design and operating system.

This study illustrates how numerical simulation combined with mathematical optimization, can be used to design a planar Stewart platform for use in a milling application. All the factors

influencing the design are taken into account, such as the workspace, singularities in the workspace, placement of the fixed platform relative to the desired workspace and dynamical effects. The method is general and intuitive, and although it is not applied to a spatial Stewart platform, it can easily be extended to be utilized for the design of such a platform.

This work also has immediate impact on the South African industry. The plastic injection and blow moulding manufacturing industry requires affordable five-axis CNC-machines to produce moulds. As these machines are very expensive, there is a need for an affordable alternative.

There are many three-axis, non-NC machine tools in South Africa that are currently under utilized. These machine tools can be fitted with a planar Stewart platform, to produce a machine tool that will meet the needs of the plastic injection and blow moulding industry. Figure 1.5 shows a three-axis machine tool fitted with a planar Stewart platform.

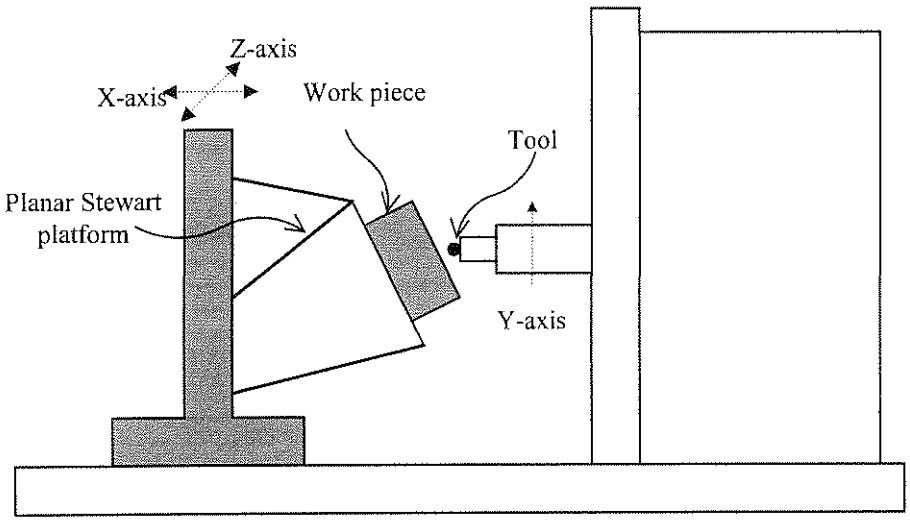

Figure 1.5: Side view of three-axis milling machine fitted with a planar Stewart platform.

The University of Pretoria in co-operation with local industry, is currently doing research on the possibility of fitting a planar Stewart platform to a three-axis milling machine, and has indeed already built a planar Stewart platform for research purposes.

In the presentation of this study extensive use are made of appendices (A-G). This is done to ensure fluency and continuity in the presentation of the main theme of the study, namely the development of an optimal design methodology for a planer Stewart platform required to perform a prescribed task. The rationale behind this kind of presentation is to avoid a situation where the reader "cannot see the wood for the trees". Although placed separately the appendices are as important as the main chapters and should be read in parallel with the central theme of the study.

Briefly the outline of the presentation is as follows. Chapter 2 gives the formulation of the design problem of a planar Stewart platform in a form that may be tackled by mathematical optimization. Chapter 3 discusses the implementation of the design procedure and also gives the optimum designs that were obtained using the *LFOPC* optimization algorithm. In chapter 4 the optimum designs given by the approximation method, *Dynamic-Q*, are discussed. The thesis concludes in chapter 5 with a discussion of the implications of the present study, and with suggestions for further research in this area.

Chapter 2: Formulation of the design problem

This chapter presents the formulation of the design problem for a planar Stewart platform in a form that allows for the optimal design to be found using mathematical optimization.

2.1 Basic optimization methodology

An optimization problem can formally be stated as:

$$\begin{aligned} &\underset{\boldsymbol{x}}{\text{minimize}}\ F(\boldsymbol{x}) \text{ with } \boldsymbol{x} = [x_1, x_2, ..., x_n]^T; x_i \in \Re \\ &\text{such that } c_j(\boldsymbol{x}) \le 0; j = 1,2,..., m \text{ and } h_k(\boldsymbol{x}) = 0; k = 1,2,..., p < n \end{aligned} \quad \text{(2-1)}$$

where F is the cost function that is being minimized, $\boldsymbol{x}$ is the vector of design variables, $\boldsymbol{c}$ is the vector of inequality constraint functions and $\boldsymbol{h}$ is the vector of equality constraint functions.

The design variables may comprise some geometrical parameter, e.g. the width of the moving platform, or the distance between the tool and the fixed base. The design constraints on a planar Stewart platform may include maximum and minimum actuator lengths that should not be exceeded as the platform executes its task, or maximum allowable actuator velocities. The following sections expand on the cost function, design variables and design constraints that are specified in designing a planar Stewart platform.

2.2 Manipulator model

The manipulator consists of a moving platform, to which the workpiece is fixed. Figure 2.1 shows the platform with the tool. The global coordinate system OXY is fixed to the tool tip. Forces induced by the milling process are modeled as external forces to the platform.

For the application considered here the dimensions (length×height×width) of the moving platform are chosen as 0.75m×0.1m×0.5m. The workpiece have dimensions of 0.5m×0.2m×0.3m. The platform weighs 135kg and is made out of steel. This platform weight takes fitting-holes in the platform into account. The workpiece is made out of aluminium and weighs 80kg.

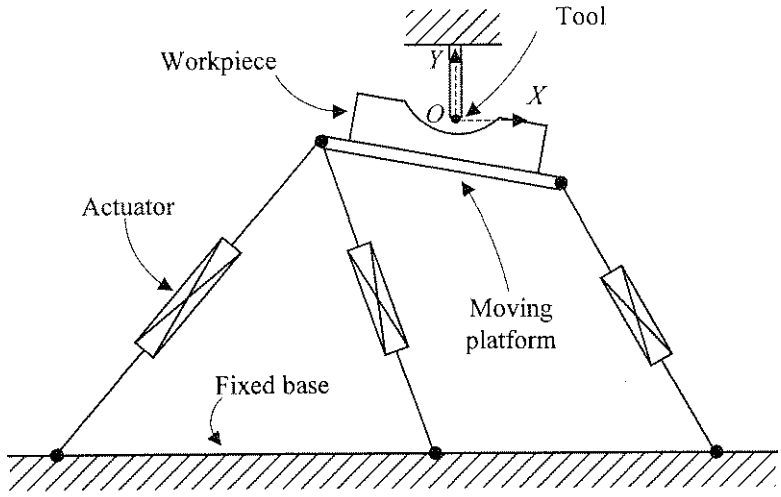

Figure 2.1: Manipulator model.

The actuators are modeled as driving constraints, with negligible mass and moment of inertia compared to the platform and workpiece. The driving constraints are distance drivers, i.e. the lengths of the actuators are specified as functions of time.

2.2.1 Tool force

The tool force is modeled as a force of constant magnitude and direction relative to the global reference system. The effect of the tool force on the platform, i.e. the resultant of the tool force that works through the platform origin, and the moment of the tool force about the platform origin, is calculated and applied to the platform origin as external forces. In Appendix A the calculation of the effect of the tool force on the platform is given. In the applications considered here the tool force has a component perpendicular to the tool path (and axially along OY) of 440N, and a tangential component of 1600N perpendicular to OY and opposite to the direction

of relative motion between the tool and the workpiece surface. Appendix A shows how the magnitude of these components were determined for the particular prescribed type and speed of the machining operation.

2.3 Design variables of manipulator

The manipulator together with six possible design variables x_i, i=1, 2, ..., 6, are shown in Figure 2.2. The origin O of the fixed global coordinate system OXY coincides with the fixed tool tip as shown in Figure 2.2. The global coordinates of the base of the left leg D is denoted by (x_4, x_3). The local coordinate system $O_P\xi\eta$ is attached to the platform as shown with the origin O_P at the midpoint of AC.

Many other design variables can be chosen, but only the six indicated in Figure 2.2 are used here to illustrate the method of finding an optimal design.

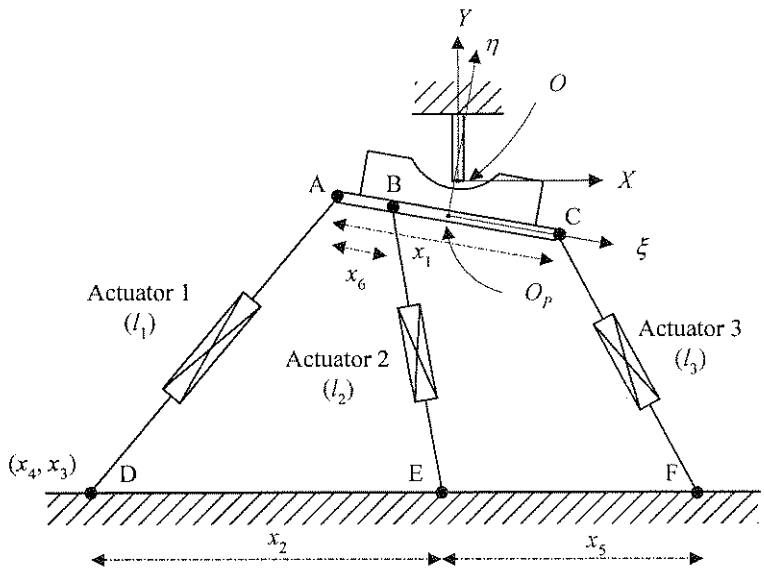

Figure 2.2: Possible design variables x_i, i=1, 2, ..., 6.

2.4 Constraints on manipulator

Constraints on the manipulator are specified by the designer, and may differ from one design problem to another. Here only inequality constraint are considered.

Limits are imposed on the maximum and minimum actuator lengths. Notice that for a prescribed path the maximum and minimum values attained by the actuators are dependent on the design vector $\boldsymbol{x}$. The respective maximum and minimum values for the different actuators are given by $l_i^{\max} = \max[l_i(t,\boldsymbol{x})]$ and $l_i^{\min} = \min[l_i(t,\boldsymbol{x})]$ for $i = 1,2,3$ over the time interval $t \in [0,T]$. The allowable maximum and minimum actuator lengths are given by $\bar{l}_i$ and $\underline{l}_i$ respectively. The first six constraints, written in standard form (see (2-1)), are therefore:

$$\begin{aligned} c_i(\boldsymbol{x}) &= \max[l_i(t,\boldsymbol{x})] - \bar{l}_i \le 0 \quad i = 1,2,3 \text{ with } t \in [0,T] \\ c_{i+3}(\boldsymbol{x}) &= \underline{l}_i - \min[l_i(t,\boldsymbol{x})] \le 0 \quad i = 1,2,3 \text{ with } t \in [0,T] \end{aligned} \tag{2-2}$$

Physical bounds on the design variables define the following additional constraints:

$$\begin{aligned} c_{6+i}(\boldsymbol{x}) &= x_i - \bar{x}_i \le 0 \quad i = 1,2,\ldots,n \\ c_{6+n+i}(\boldsymbol{x}) &= \underline{x}_i - x_i \le 0 \quad i = 1,2,\ldots,n \end{aligned} \tag{2-3}$$

with $\bar{x}_i$ and $\underline{x}_i$ respectively denoting the upper and lower limits on variable x_i.

2.5 Cost functions

The choice of cost function depends on the design criterion and the particular application. For example, a high accuracy positioning device could be designed by using stiffness as a cost function, or for a good design for a computer plotter, the size of the workspace area may be the appropriate criterion.

In the optimal design of a milling machine considered in this study, two cost functions are used, namely the maximum actuator force and the quality index. Using maximum actuator force as a cost function will give a design which minimize the forces in the actuators, and will thereby reduce the required actuator capacity. It is difficult to give an exact physical meaning to the quality index, other than to say that it gives a measurement of "closeness" to a singular position[2]. Using the quality index as a cost function to be maximized for the prescribed path, will give a design where the platform is "far away" from singular positions. Such a design should therefore also correspond to one that also reduces the magnitude of the required actuator forces.

2.5.1 Maximum actuator force as design criterion

The manipulator is to be designed such that the forces in the respective actuators $f_k(t)$, k=1, 2, 3, are minimized in some sense over the time interval $[0,T]$. The particular criterion used here is that of the minimization of the maximum absolute value of the actuator forces, i.e. the cost function $F(\boldsymbol{x})$ is defined as:

$$F(\boldsymbol{x}) = \max_{k=1,2,3}\left\{\max_{t}\left|f_k(t)\right| \quad t\in[0,T]\right\} \tag{2-4}$$

2.5.2 Quality index as design criterion

The optimum design using the quality index $\lambda(t)$ as criterion (see Appendix C) can be found by maximizing the minimum value of the quality index, over the time interval $[0,T]$. In this case the cost function for the minimization problem (2-1) can mathematically be stated as:

$$F(\boldsymbol{x}) = -\min_{t}\left|\lambda(t)\right| \quad t\in[0,T] \tag{2-5}$$

---

[2] A singular position can mathematically be described as a configuration for which the Jacobian matrix of the system is singular. Physically this is a configuration in which the driving forces become infinitely large, for example where the line of action for the three actuators meet in a point.

2.6 Manipulator tasks and tool path

A parallel platform is to be designed for the particular task it should perform. In this study, only machining tasks are considered. The machining task is completely specified by:

1. the tool path that the tool should cut out of the workpiece,
2. the relative angle between the tool axis OY (Figure 2.1) and the tool path, and
3. the tool velocity, i.e. the velocity with which uncut material is fed to the tool.

The tool path should be specified in the local coordinate system, $O_P\xi\eta$, because coordinates in this system are independent of the design, $\boldsymbol{x}$. The two tool paths, A and B, that are considered in this study are shown in Figure 2.3(a) and (b) respectively. Note from Figure 2.3 that here the tool paths are chosen to start and end on the top level surface of the workpiece, which is located at $\eta = 0.2\text{m}$. Also note that for these cases the tool paths are centered about ξ=0 (the η-axis). For both tasks, it is required of the tool axis OY to remain perpendicular to the tool path throughout the execution of the manipulator task.

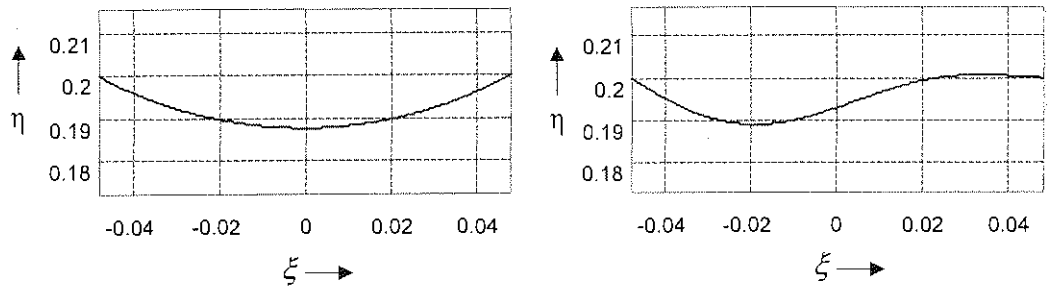

Figure 2.3 (a), (b): Respective tool paths, A and B, that are considered in this study.

In machining applications the tool velocity is a function of time, and here has a profile that is shown in Figure 2.4. This profile is calculated from the maximum allowable tool velocity, v_{max}, as well as the maximum magnitude of any acceleration of the workpiece tangential the tool path, a_{max}. Appendix B gives details of the calculations involved in determining the velocity profile.

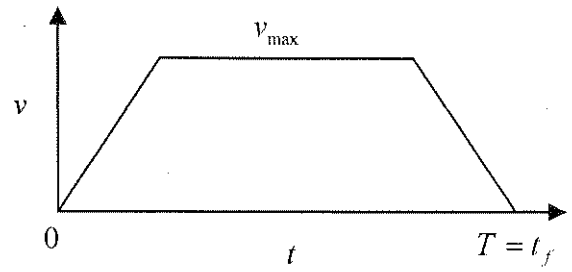

Figure 2.4: Velocity profile of tool relative to workpiece.

2.7 Actuator drivers

The actuator lengths change continuously with time as the manipulator executes its task. In order for the manipulator to execute a specific task, the actuator drivers for this task need to be calculated. An actuator driver is a curve specifying the actuator length as a function of time, i.e. the distance driver between the joints of the actuator.

The actuator drivers, which are functions of the manipulator design, $\boldsymbol{x}$, are found by performing the inverse kinematics of the prescribed manipulator path. Appendix B describes how the actuator drivers are found given the manipulator task.

2.8 Discretisation of continuous functions

The manipulator executes a continuous task, which implies that all functions of time, the tool path, actuator lengths, actuator forces etc., are continuous. These continuous functions cannot be used directly by a computer, as a computer can only handle discrete information. By using however the values of the function at a large number, n_p, of discrete points, a continuous function can be approximated as accurately as required.

In practice a continuous function of time, $f(t)$, is discretised by evaluating the function at discrete time instants, i.e. $f(t)$ is represented by the sequence $f(t_i)$, i=1, 2, ..., n_p, where $t_i \in [0,T]$

such that $t_1 = 0;\ t_i < t_{i+1};\ t_n = T$. The discretisation of the tool path, platform path and actuator drivers are described in Appendix B

Chapter 3: Optimization using the *LFOPC* algorithm

The minimization problem stated in (2-1) has to be solved by a suitable optimization algorithm. This chapter describes the implementation of the *LFOPC* (Appendix E) constrained optimization algorithm of Snyman [26, 27]. This is a very robust and reliable gradient based optimization algorithm. The optimum design problem is solved using *LFOPC* for various conceptual designs, cost functions and manipulator tasks. All the results presented are for the tool path shown in Figure 2.3(a), unless otherwise stated.

3.1 Implementation of optimization procedure

The procedure of finding the optimum platform design, given an initial and usually infeasible design, $\boldsymbol{x}^0$, is shown in Figure 3.1 and Figure 3.2 for the respective cost functions. The initial design is evaluated and the design variables are then changed by the algorithm. The effect of the changes on the design criterion is tracked and new changes to the design are computed by the optimization algorithm. The process of changing the design variables in a systematic manner and checking the design criterion is continued until no further significant improvement in the design criterion is obtained. This design is then taken as the optimum design, $\boldsymbol{x}^*$.

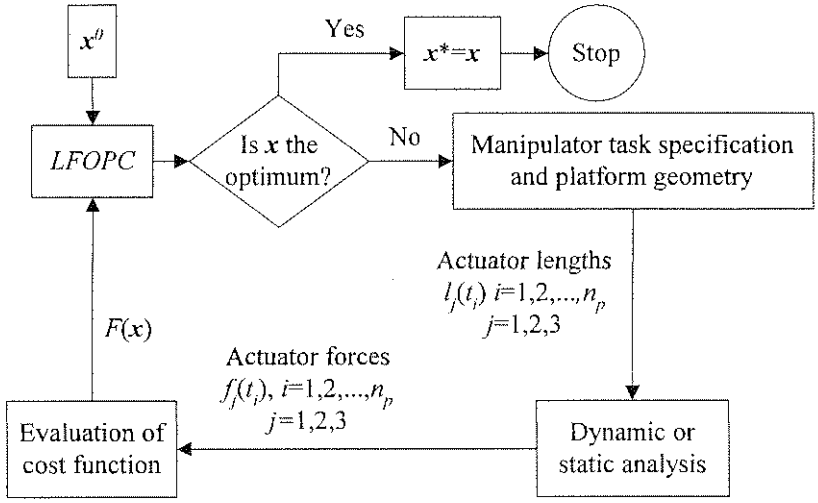

Figure 3.1: Diagram of optimization procedure for the maximum *actuator force* cost function.

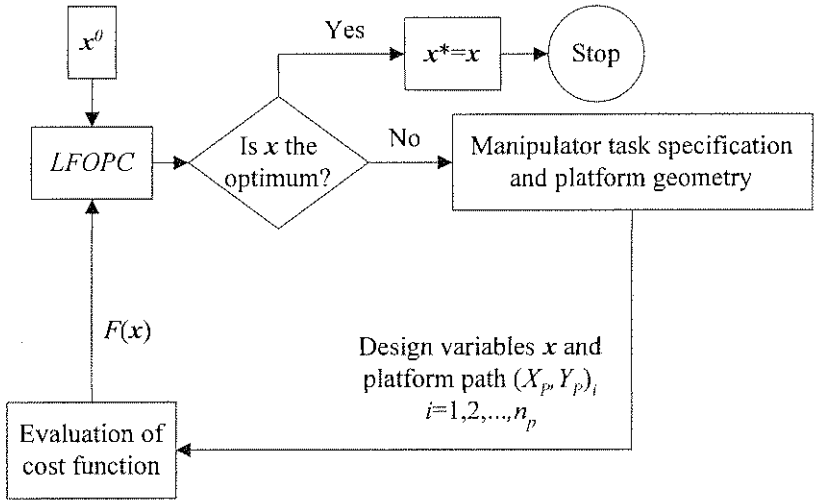

Figure 3.2: Diagram of optimization procedure for the *quality index* cost function.

The following sections describe the different practical aspects related to the implementation of the optimization algorithm such as, for example, the calculation of the actuator forces and evaluating the gradient vector of the cost function.

3.1.1 Cost functions

Evaluating the actuator force cost function (2-4), requires a dynamic analysis. This analysis is done using the numerical *Dynamic Analysis Design System* (*DADS* v.9.0) [14].

The prescribed manipulator tasks do not require large accelerations of the platform. This means that a static analysis can be used as an approximation to the dynamic analysis. In this static analysis, the actuator forces are calculated at each time instant, t_i, i=1, 2, ..., n_p. Since the configuration of the platform may be determined at each time instant, t_i, the actuator forces, $f_j(t_i)$, may be solved for as if the platform is static in that configuration. The static analysis of the platform is performed using the Jacobian matrix of the structure (see Appendix D).

The quality index cost function requires that the quality index of the platform be calculated at each time instant, t_i, i=1, 2, ..., n_p. At each time instant, t_i, the configuration of the platform is determined, and the quality index, $\lambda(t_i)$, is calculated (see Appendix C).

The different cost functions are labeled as follows:

1. $F_{A.F.D}(\boldsymbol{x})$ refers to the actuator force cost function based on the dynamic analysis,
2. $F_{A.F.S}(\boldsymbol{x})$ refers to the actuator force cost function based on the static analysis, and
3. $F_{Q.I}(\boldsymbol{x})$ refers to the quality index cost function based on the kinematic analysis.

3.1.1.1 Scaling

For efficient performance the *LFOPC* algorithm requires the magnitude of the cost function gradient vector at the initial design, $\boldsymbol{x}^0$, should be of the order of unity. The actuator force cost function is therefore multiplied by a factor of 0.001, which implies that the forces are measured in kN instead of N.

3.1.1.2 Singularities

The $F_{A.F.D}$ cost function requires the use of dynamic analysis software. This software calculates the actuator forces as the platform executes its prescribe task. If the platform at any time during

the analysis assumes a near singular configuration, the actuator forces become infinitely large and the numerical solver breaks down. In these cases the actuator forces cannot be calculated over the whole time interval, and the cost function is therefore undefined and cannot be computed.

This problem is overcome by defining the actuator forces over the remainder of the path and time interval, to be the same as it was at the last instant where it could be calculated [29]. Since it is expected that the values at the last non-singular point would be large, the resultant artificial value of the cost function will also be artificially large. The optimization algorithm, using the artificial cost function where necessary, will therefore drive the design away from singular configurations.

The $F_{A.F.S}$ cost function also exhibits this problem. If the Jacobian that is used to calculate the actuator forces, is ill-conditioned the actuator forces are extremely large, or if the Jacobian is singular, the actuator forces cannot be calculated at all.

In these cases the same strategy is applied to the cost function as is described above. The condition of the Jacobian is determined before the actuator forces are computed. If the Jacobian is ill-condition or singular, the actuator forces over the remainder of the time interval are defined to have the same values it had at the last time instant when the Jacobian was well-conditioned.

For near-singular configurations, the quality index approaches zero, hence the $F_{Q.I}$ cost function is defined for all possible design variables, $\boldsymbol{x}$.

3.1.2 Gradient vector evaluation

The *LFOPC* algorithm is a gradient based algorithm which requires that the gradient components of the cost function and of the constraint functions be known at any design $\boldsymbol{x}$. Some of these gradient components are not analytically known. In these cases approximations to the gradients are computed using forward finite differences. For example, the approximation to the *i-th* component of the gradient vector of $F(\boldsymbol{x})$ at $\boldsymbol{x}_k$ is taken as:

$$\frac{\partial F}{\partial \boldsymbol{x}_i} \approx \frac{f(\boldsymbol{x}_k + \boldsymbol{\delta}_i) - f(\boldsymbol{x}_k)}{\delta_i} \qquad \textbf{(3-1)}$$

where $\boldsymbol{\delta}_i = [0, 0, \ldots, \delta_i, 0, \ldots, 0]^T$, $\delta_i > 0$, i.e. only the *i-th* component of $\boldsymbol{\delta}_i$ is non-zero and equal to δ_i, a suitably chosen small positive number.

The gradient components of the constraints are calculated in the same manner where these constraint gradients are not analytically available.

The accuracy of the gradient components depends largely on the values of δ_i. Figure 3.3 graphically shows the dependence of the approximation to the first component of the gradient vector of the cost function, $F_{A.F.D}$, on different δ_1-values at two different design points. The approximations to the other components of the cost function gradient vector with corresponding steps, δ_i, i=2, 3, ..., 6, show similar trends. From the study of the approximations to the cost function gradient components, δ_i-values for each of the design variables are chosen. In particular δ_1 is chosen as $2 \cdot 10^{-4}$ (Figure 3.3).

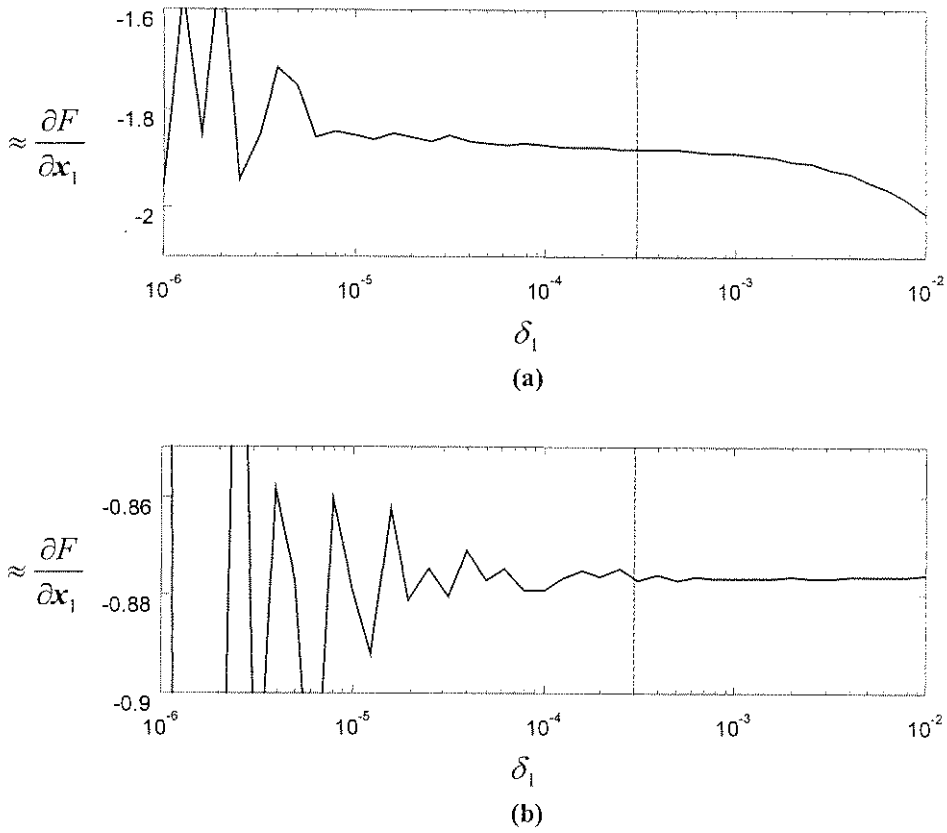

Figure 3.3: Approximate values of the first gradient component, $\approx \frac{\partial F}{\partial x_1}$, plotted against δ_1, for (a) x=(0.4, 0.6, -0.6, -0.8, 0.7, 0.0)^T, (b) x=(0.7, 0.8, -1.2, -0.8, 0.7, 0.0)^T.

The gradient components for the $F_{A.F.S}$ and $F_{Q.I}$ cost functions, as well as for the constraint functions are less sensitive to the choice of the values of δ_i, than is the case with the $F_{A.F.D}$ cost function. The reason is that in the latter case a comprehensive numerical analysis is necessary for the dynamical simulation, which adds noise to the cost function and as a result complicates the computation of the gradient approximations. This is not the case for the other two cost functions where the noise componenents are negligibly small.

Typically Figure 3.4 shows negligible noise in the approximation of $\frac{\partial c_1(\boldsymbol{x})}{\partial \boldsymbol{x}_1}$ at $\boldsymbol{x}$=(0.4, 0.6, -0.6, -0.8, 0.7, 0.0)^T. This trend is similar for the approximations to the gradient

components of the $F_{A.F.S}$ and $F_{Q.I}$ cost functions and for the approximations to the gradient components of all the constraints. For these cases δ_i is chosen as 10^{-6} throughout.

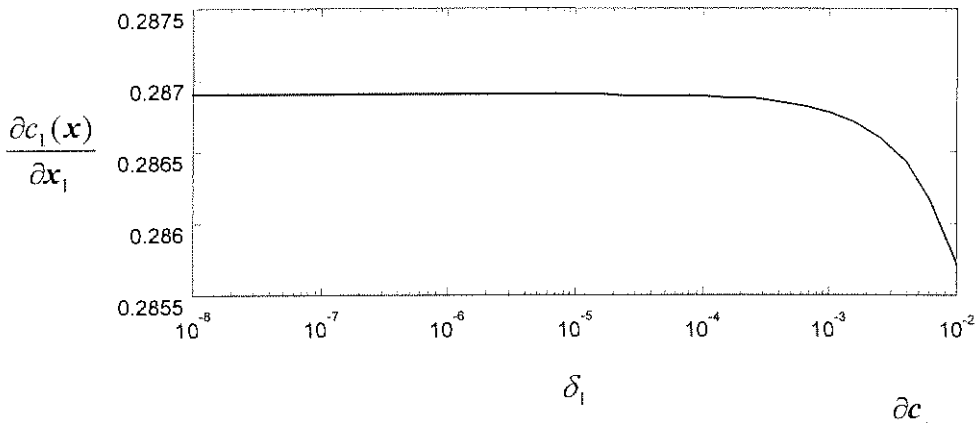

Figure 3.4: Approximate value of the gradient of c_1 w.r.t. x_1, $\approx \frac{\partial c_1}{\partial x_1}$.

3.1.3 Number of discretisation points, n_p

The number of discretisation points also influence the accuracy to which a cost function is calculated. The inaccuracies are illustrated by Figure 3.5 and Figure 3.6 for task path A (see Figure 2.3) over the interval $[0,T]=[0,12]$. Figure 3.5 shows the actuator force, $f_3(t)$ plotted at time instants t_i, $i=1, 2, ..., n_p$. Figure 3.6 is an enlargement of Figure 3.5 showing plots for different numbers of discretisation points, n_p.

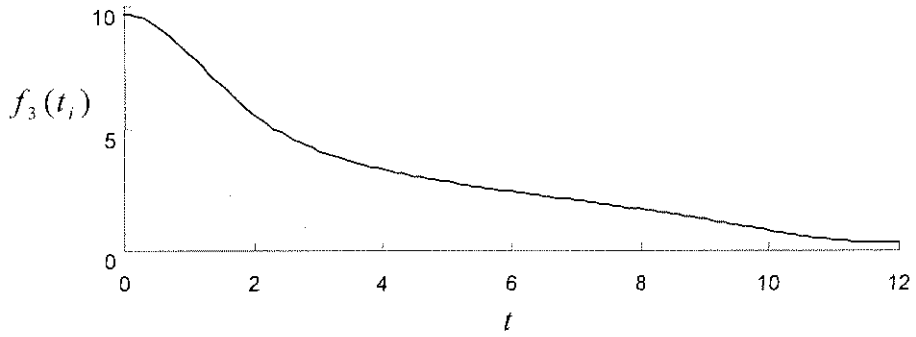

Figure 3.5: The variation of actuator force $f_3(t)$, for the design x=(0.4, 0.6, -0.6, -0.8, 0.7, 0.0)^T and tool path A.

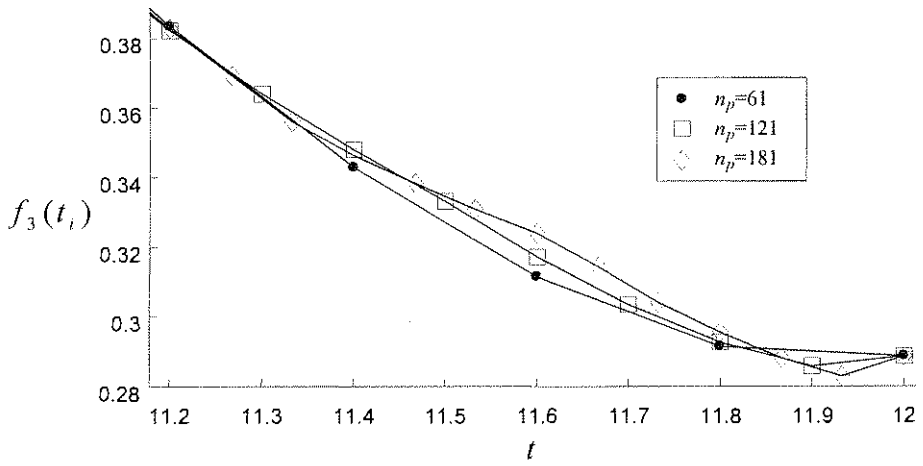

Figure 3.6: An enlargement of part of Figure 3.5, showing the effect of the number of discretisation points, n_p, on f_3.

Figure 3.5 show that near the end of the time interval $[0,T]$, the actuator force f_3 appears to be inaccurate for n_p large, with the presence of oscillations relative to the smooth behavior for smaller n_p.

Too many points therefore cause numerical inaccuracy of the actuator forces near the end of the manipulator task. These inaccuracies originate from the manner in which the tool path is discretised (Appendix B). A time instant, t_i, is associated with each point on the tool path (which is transformed to the actuator drivers). Near the end of the tool path, these time instants are calculated inaccurately due to the cubic-spline approximation which is used to determine them. Figure 3.7 shows how the time instants, t_i, are calculated from the distance, s_i, the tool has

traveled along the tool path. At the end of the tool path, the cubic-spline has a very large slope, which cause the time instants to be calculated inaccurately, as is shown by the large error in time, ε_t, that results from a small error, ε_s, in the distance.

On the other hand, if too few points are sampled along the prescribed path, the peak in the cost function may not be accurately determined. On the basis of the available evidence the compromised choice of n_p=61 was used throughout.

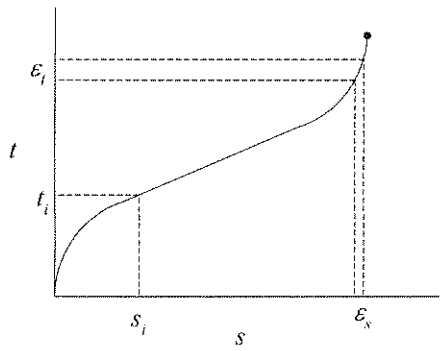

Figure 3.7: The cubic-spline used to find the time instant, *t*, at which the tool has travelled a distance *s*.

3.1.4 Accuracy and noise

The dynamic analysis software system, *DADS*, gives all results to single precision accuracy. The cost function is therefore also only accurate to single precision. The actual practical accuracy of the cost function is shown in Figure 3.8, where the cost function value $F(\boldsymbol{x})$ is plotted against the design variable x_1 at $\boldsymbol{x}=(0.4, 0.6, -0.6, -0.8, 0.7, 0.0)^T$, over a very small interval . From this figure it is clear that the function value is accurate to within 10^{-6}.

The cost function is expected to be smooth, but Figure 3.8 reveals the existence of noise in the cost function. It is common for numerical simulations, which is also used in *CFD* or *FEM*

software, to have numerical noise. The noise is a result of limited computer precision and cumulative computational errors.

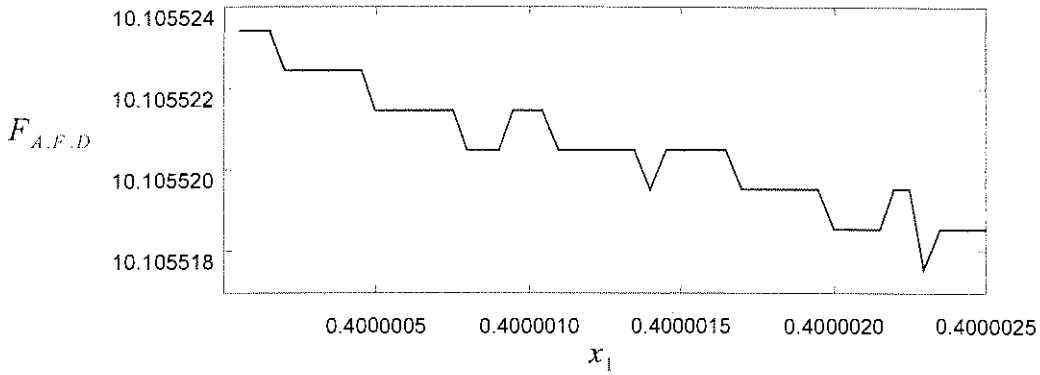

Figure 3.8: The accuracy of the value of the cost function *F* is shown to be within 10^{-6}.

3.1.5 Algorithm parameters

LFOPC requires only three parameter settings, ε_x, ε_g, and *DELT* (see Appendix E). Different algorithm parameters are used for the different cost functions, because of the nature of the cost functions. The $F_{A.F.D}$ cost function has numerical noise in its gradient, because it assumes discrete values (Figure 3.8). The other cost functions $F_{A.F.S}$ and $F_{Q.I}$ are calculated to double precision accuracy, which gives a smoother cost function and therefore a smoother gradient with negligible noise.

For $F_{A.F.D}$ the convergence stop tolerance for the design variables, ε_x, was set to 10^{-5}. The convergence tolerance for the cost function gradient, ε_g, was set to 10^{-5} (see termination criteria for *LFOPC* in Appendix E). The maximum optimization trajectory step size, *DELT*, is set to $0.2\sqrt{n}$, where 0.2 is the order of the variable range expected and n is the number of design variables. The maximum number of steps per phase is set to 1000 (see paragraph E.2 of Appendix E).

For $F_{A.F.S}$ and $F_{Q.I}$ the same values are used as for $F_{A.F.D}$, except that the convergence stop tolerance, ε_x, is changed to 10^{-7}, because these cost functions are calculated more accurately than $F_{A.F.D}$. The maximum number of steps per phase is changed to 2000.

3.2 Optimum designs

Here follow the optimum designs obtained for different combinations of cost functions, tool paths, and choice of number of design variables. The results for three design variables are presented in detail in Section 3.2.1, whilst the results for the other cases are dealt with more concisely in Section 3.2.2 and summarized in Appendix G.

In all cases the actuator lengths are constrained (see (2-2)) to a minimum value of 0.6m and a maximum value of 1.2m, i.e. $\underline{l}_i$=0.6 and $\bar{l}_i$=1.2, i=1,2,3.

3.2.1 Results for three design variables

The three design variables that are used here are, x_i, i=1, 2, 3. The other design variables are fixed at the values x_4=-0.8, x_5=0.7 and x_6=0.0. The constraints on the variables (see (2-3)) are set to: $0.1 \le x_1 \le 0.75$, $0.1 \le x_2 \le 0.8$, $x_3 \le 0$.

Tool path A is prescribed here (Figure 2.3(a)). The initial design is chosen as $\boldsymbol{x}^0$=(0.4, 0.6, -0.6)^T. For the given initial design the value of the cost function is $F_{A.F.D}(\boldsymbol{x}^0) = 10.107$ kN. The optimum design is found to be $\boldsymbol{x}^*$=(0.123, 0.8, -0.995)^T, with a considerably reduced cost function value of $F_{A.F.D}(\boldsymbol{x}^*) = 4.887$ kN. The actuator forces for the initial and optimum designs are shown in Figure 3.9 and Figure 3.10 respectively.

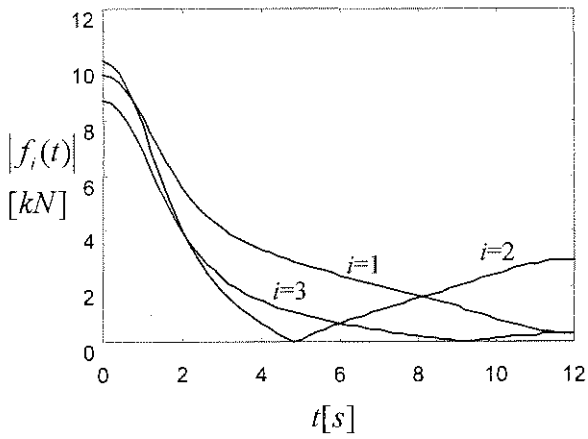

Figure 3.9: The actuator forces vs. time plot for the iuitial configuration.

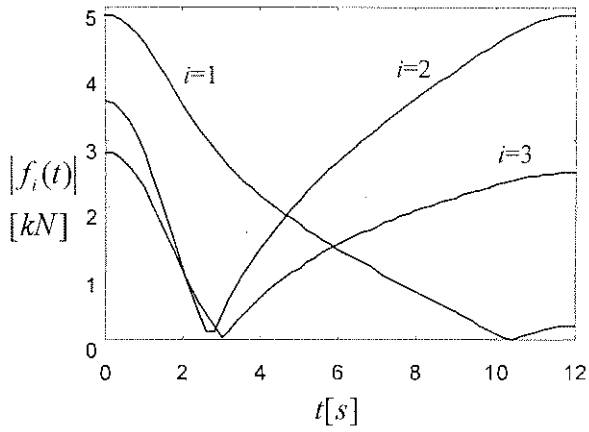

Figure 3.10: The actuator forces vs. time plot for the optimal configuration.

The actuator length variation for the initial configuration violates the constraints on the actuator lengths, as shown in Figure 3.11. After optimization the optimum configuration satisfies the actuator length constraints as shown in Figure 3.12. The active constraints at the optimum, x^*, are $\max[l_1] \leq \bar{l}_1$ and $x_2 \leq 0.8$.

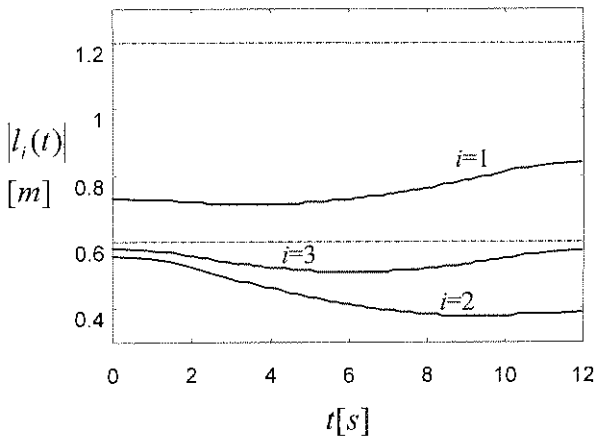

Figure 3.11: The actuator lengths $l_i(t)$ for the initial configuration.

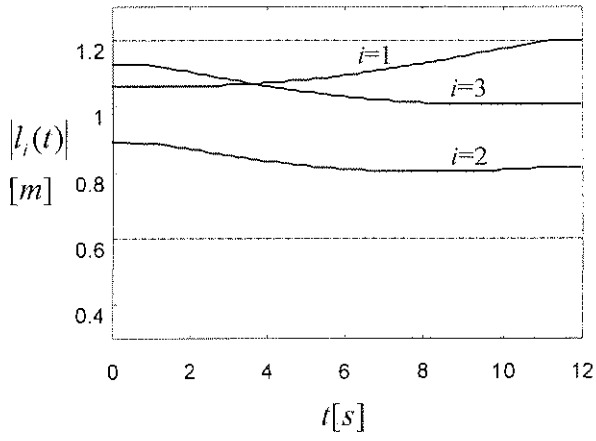

Figure 3.12: The actuator lengths $l_i(t)$ for the optimum configuration.

The convergence histories of the cost function $F_{A.F.D}(x)$ and the design variables x_i, i=1, 2, 3 are depicted in Figure 3.13 and Figure 3.14 respectively. Effective convergence is clearly obtained

in little more than 60 *LFOPC* trajectory steps. The total computation time on a Pentium II-266 is less than 20 minutes.

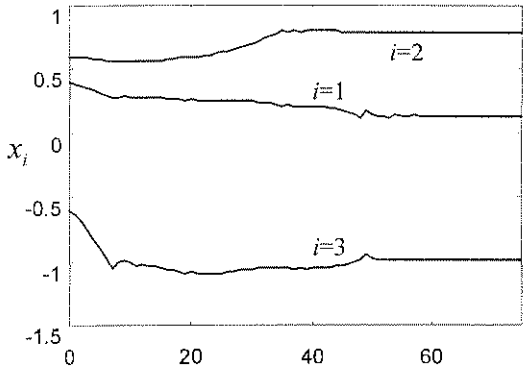

Figure 3.13: The convergence history of the design variables x_i , i=1,2,3 are shown.

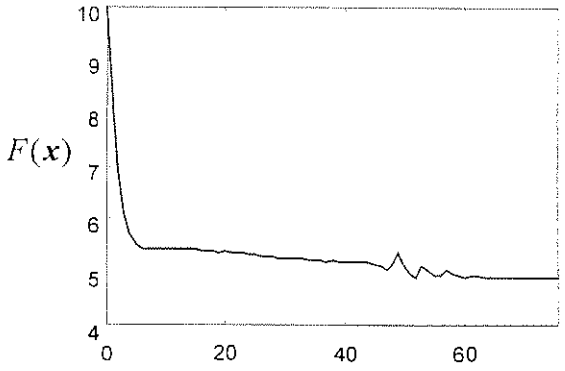

Figure 3.14: The convergence history of the cost fnnction $F_{A.F.D}(x)$ is shown.

Figure 3.15 to Figure 3.18 show the configurations for the initial design, $x^0=(0.4, 0.6, -0.6)^T$ and the optimum design $x^*=(0.123, 0.8, -0.995)^T$.

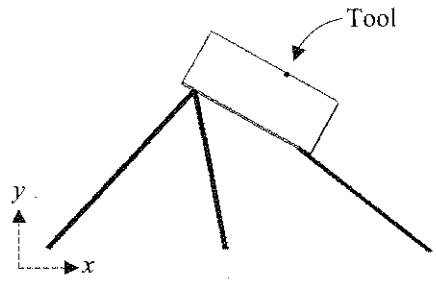

Figure 3.15: Start position of platform for initial design, x^0.

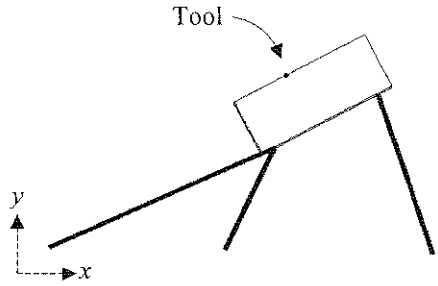

Figure 3.16: End position of platform for initial design, x^0.

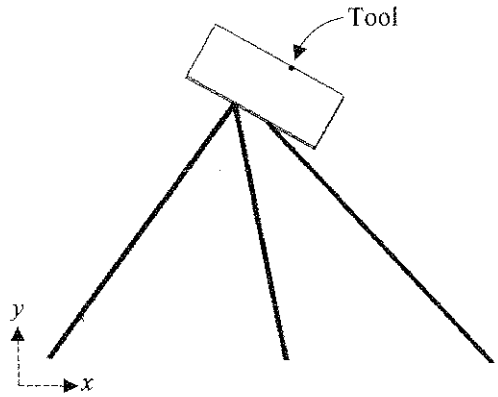

Figure 3.17: Start position of platform for optimum design, x^*.

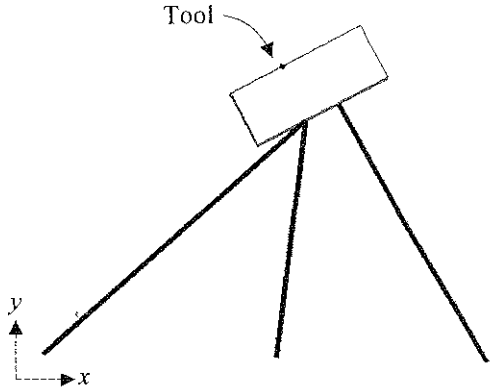

Figure 3.18: End position of platform for optimum design, x^*.

3.2.1.1 Local optima

By using a different initial design of $\boldsymbol{x}_I^0 = (0.7,\ 0.4,\ -1.0)^T$, a local optimum was reached at $\boldsymbol{x}_I^* = (0.750,\ 0.604,\ -0.932)^T$, which is different to the previous local optimum $\boldsymbol{x}^* = (0.123,\ 0.8,\ -0.995)^T$. The value of the cost function at this point is $F_{A.F.D}(\boldsymbol{x}_I^*) = 4.795$, compared to the value of $F_{A.F.D}(\boldsymbol{x}^*) = 4.887$ at the previous local optimum. The values of the cost function at these two design optima are of similar magnitude, which gives the designer of the platform a choice between two very good, but different, feasible designs.

With another initial design of $\boldsymbol{x}_{II}^0 = (0.5,\ 0.8,\ -0.6)^T$ a local optimum at $\boldsymbol{x}_{II}^* = (0.469,\ 0.863,\ -0.837)^T$ is obtained, for which the cost function is $F_{A.F.D}(\boldsymbol{x}_{II}^*) = 8.481$. For the initial design, the platform moves through a singular configuration yielding an extremely

large (~10^6) cost function value. The initial design is therefore in a region of the design space for which the platform assumes singular configurations. The components of the cost function gradient are inaccurately calculted in such regions (due to the fact that the actuator forces are only calculated for the time interval until the platform moves through a singular configuration). Nevertheless the optimization algorithm succeeds in moving away from the undesirable region of the design space, terminating at $\boldsymbol{x}_{II}^{*}$ with a cost function value $F_{A.F.D}(\boldsymbol{x}_{II}^{*})$=8.481, larger than the previous local optima, but dramatically less than the value at $\boldsymbol{x}_{II}^{0}$.

3.2.1.2 Non-differentiable cost function

The $F_{A.F.D}$ and $F_{A.F.S}$ cost functions may be non-differentiable in certain regions. Figure 3.14 shows the plot of $F_{A.F.D}$ vs. x_1 with the other design variables held constant at x_2=0.437, x_3=-0.846, x_4=-0.8, x_5=0.7 and x_6=0.0. For $x_1 \leq 0.3769$, the slope of the cost function takes on the value –10.220 while for $x_1 > 0.3769$ the slope has the value 1.091. The cost function is therefore non-differentiable at x_1=0.3769.

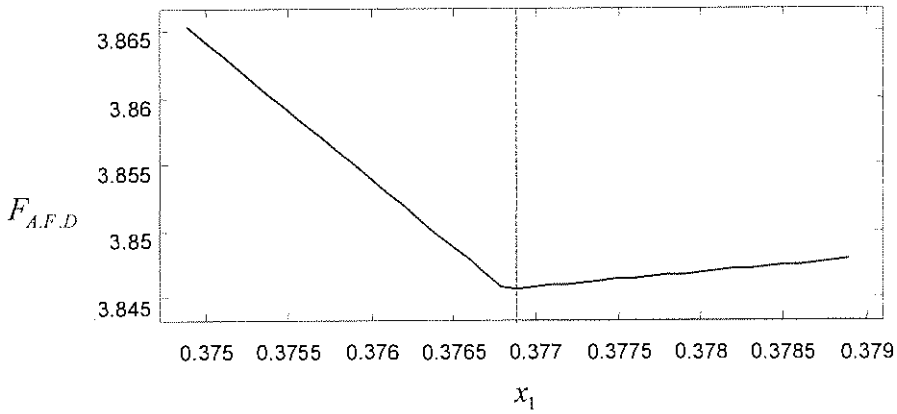

Figure 3.19: Discontinuity in the slope of the cost function $F_{A.F.D}$ as x_1 varies but x_i, i=2, 3, ..., 6, are kept fixed.

Non-differentiable cost functions make it difficult for gradient based optimization algorithms to converge to the optimum, as some components of the gradient vector change abruptly for small changes in the design variables. For Figure 3.19 the x_1 component of the gradient changes from -10.220 to 1.091 as x_1 increases from 0.3768 to 0.3769. Because of the nature of the cost

function defined in (2-4), discontinuities in the gradient are expected at the various local minima that may occur.

3.2.2 Overall discussion of all the results

A summary of the results obtained for different choices and numbers of design variables, different cost functions, tool paths and initial designs, are given in Appendix G. A brief discussion of the overall results follow here.

The $F_{A.F.S}$ cost function gives, as expected, similar results to $F_{A.F.D}$. This makes $F_{A.F.S}$ a preferred cost funcion over $F_{A.F.D}$, as it is computationally far less expensive. The time to determine the optimum design using $F_{A.F.D}$ for five variables is around 1 hour compared to 10 minutes using $F_{A.F.S}$.

For three design variables with tool path B, the value of $F_{A.F.D}$ at $\boldsymbol{x}^*_{Q.I}$=(0.750, 0.653, -0.944)^T, which is the optimum design obtained from $F_{Q.I}$, is 3.788. This value is more than 3.392, which is the value that is obtained from the optimum design $\boldsymbol{x}^*_{A.F.D}$=(0.184, 0.800, -1.019)^T using $F_{A.F.D}$ as a cost function. The value of $F_{Q.I}$ at the optimum design obtained from $F_{A.F.D}$, $\boldsymbol{x}^*_{A.F.D}$=(0.184, 0.800, -1.019)^T, is -0.366, which is considerably more than the minimum value, -0.476 that is found using $F_{Q.I}$. Thus the two criteria used as cost functions are not exactly equivalent, although minimizing the one results in a reduction in the other, and vice versa.

A further reason for the difference in optimum designs using $F_{A.F.D}$ and $F_{Q.I}$, is that $F_{Q.I}$ does not take external forces on the platform into account. However, as already indicated, an optimum design found by using one of the cost functions is a fairly good design under the criterion of the other cost function.

The importance of design optimization can be shown by comparing arbitrary feasible designs (designs where all the design constraints are met) with the optimum designs. An arbitrary feasible design can be generated by taking the cost function as constant and simply satisfying the constraints for a random starting point using *LFOPC*. This will give a feasible design that is not optimized under the criterion of any cost function. Table 3-1 gives arbitrary designs for a design with three design variables and for tool path A. Table 3-2 gives arbitrary designs with five design variables and also for tool path A.

Table 3-1: Cost function values for feasible designs (three design variables, x_i, i=1, 2, 3, tool path A).

Feasible design, $\boldsymbol{x}^{\#}$	$F_{A.F.D}(\boldsymbol{x}^{\#})$	$F_{Q.I}(\boldsymbol{x}^{\#})$
$(0.435, 0.758, -1.086)^T$	5.163	-0.273
$(0.314, 0.297, -0.893)^T$	6.194	-0.188
$(0.323, 0.576 -0.858)^T$	5.590	-0.253

Table 3-2: Cost function values for feasible designs (five design variables, x_i, i=1, 2, ..., 5, tool path A).

Feasible design, $\boldsymbol{x}^{\#}$	$F_{A.F.D}(\boldsymbol{x}^{\#})$	$F_{Q.I}(\boldsymbol{x}^{\#})$
$(0.503, 0.237, -0.928, -1.057, 1.037)^T$	9.631	-0.106
$(0.495, 0.879, -0.732, -0.555, 0.500)^T$	28.064	-0.047
$(0.301, 0.548, -0.937, -0.852, 0.731)^T$	5.402	-0.265

Clearly the $F_{A.F.D}$ cost function values for arbitrary designs with three design variables, are significantly larger than the $F_{A.F.D}$ cost function values for the optimum designs, $\boldsymbol{x}^*_{A.F.D}$, which is around 3.5.

The cost function values for arbitrary designs with five design variables, emphasize the importance of design optimization. The optimum designs with five design variables, $\boldsymbol{x}^*_{A.F.D}$,

have $F_{A.F.D}$ cost function values of around 1.5, compared to 5.4 and higher for the arbitrary feasible designs.

Table 3-3 gives the optimum designs for five variables, x_i, i=1, 2, ..., 5, with no tool force and with 90% of the tool force applied to the platform respectively. These results shows how important it is to model the applied tool force fairly accurately. The optimum design with no applied tool force, x^{**}, is a very poor design for when the tool force is applied. The optimum design with 90% of the tool force applied, gives a similar cost function value to that obtained when the total tool force is applied.

Table 3-3: Cost functions for optimum designs with no applied tool force.

	x^{**}	$F_{A.F.D}(x^{**})$	With tool force: $F_{A.F.D}(x^{**})$
No tool force: x^{**}	$(0.711, 0.310, -0.921, -0.287, 0.100)^T$	1.167	6.138
90% of tool force: x^{**}	$(0.750, 0.832, -0.978, -1.121, 0.101)^T$	1.424	1.634
Total tool force: x^{**}	$(0.750, 0.833, -0.961, -1.148, 0.101)^T$	1.525	1.525

Chapter 4: Optimization using the approximation method: *Dynamic-Q*

The previous chapter discusses the optimum designs for three and for five design variables. The computational time for finding the optimum design with five design variables using the $F_{A.F.D}$ cost function, is around one hour, depending on the initial design. The reason for the long computational time is that for n design variables, n+1 cost function evaluations are required to determine all the components of the cost function gradient at each design point along the *LFOPC* optimization trajectory. With around 700 iteration steps required to find the optimum design the computational process becomes very expensive.

4.1 The Dynamic-Q approach

For larger numbers of design variables an alternative way of finding the optimum design is clearly required that would demand much less iterations. One solution is to use an approximation optimization algorithm, such as *Dynamic-Q*. Appendix F gives the basic *Dynamic-Q* algorithm, which is an approximation method, where a computational expensive function is approximated by a spherical quadratic function. Figure 3.8 shows that the cost function may not only be non-differentiable, but also non-quadratic, which make the optimization problem ill-suited for *Dynamic-Q*, as *Dynamic-Q* is based on the assumption that near the optimum, the cost function can be approximated by a spherical quadric function. In spite of this, the results show that *Dynamic-Q* does remarkably well when applied to the current problems.

Dynamic-Q is used here only for the designs that are optimized with the $F_{A.F.D}$ cost function, and which have five- or six design variables. These design problems justify the use of approximations due to excessive computational times required when direct methods are used.

4.2 Discussion of results

The results obtained using *Dynamic-Q* are also listed in Appendix G. For all the design problems, the move limits are set to (Δ_i=0.2 i=1, 2, ..., 5). The optimum designs obtained using *Dynamic-Q* are similar to the ones obtained using *LFOPC*. For example, an optimum design for tool path A, using *Dynamic-Q* is $\boldsymbol{x}^*$=(0.737, 0.737, -0.916, -1.049, 0.100)^T after 31 iterations with $F_{A.F.D}(\boldsymbol{x}^*) = 1.564$, compared to an optimum design using *LFOPC*, $\boldsymbol{x}^*$=(0.750, 0.833, -0.961, -1.148, 0.101)^T after 727 iterations with $F_{A.F.D}(\boldsymbol{x}^*) = 1.525$. This shows that *Dynamic-Q* can successfully and economically be applied to these design problems, and in particularly so for design problems with five or six design variables.

The cost function convergence histories for the *LFOPC* and *Dynamic-Q* algorithms for the five design variable problem are shown in Figure 4.1 and Figure 4.2 for two different initial designs. They show that the *Dynamic-Q* algorithm gives very good and acceptable designs after considerably fewer iterations than that required by *LFOPC*.

Figure 4.2 show the cost function convergence histories of the respective algorithms, with an initial design for which the platform passes through a singular configuration. For this design problem, *Dynamic-Q* gives a good design much faster than *LFOPC*. Indeed an acceptable and feasible design with low cost function value is reached within less than 20 iterations. This is because the relative large specified move limits of the *Dynamic-Q* algorithm, allow the design,

$\boldsymbol{x}$, to move quickly away from the region of the design space for which the platform moves near a singular configuration.

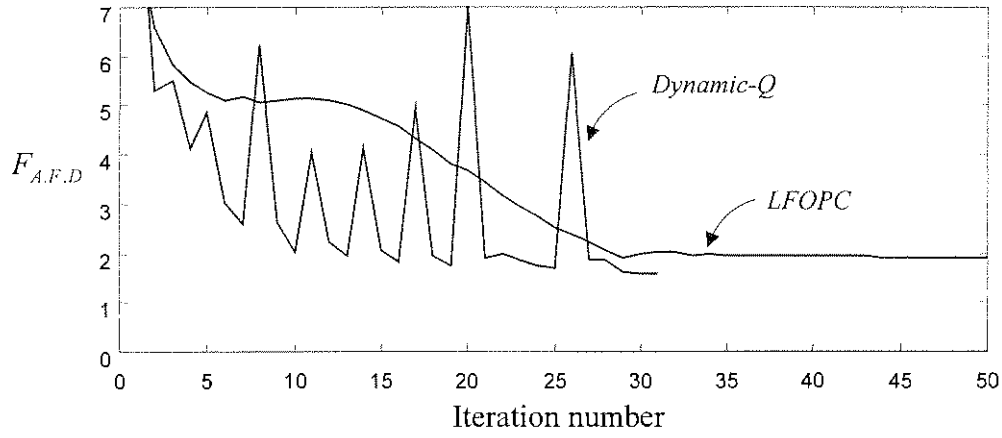

Figure 4.1: Convergence history for *LFOPC* and *Dynamic-Q*, with x^0=(0.4, 0.6, -0.6, -0.8, 0.7)^T. $F_{LFOPC}(x^*)$=1.525 after 727 iterations and $F_{Dyn.-Q}(x^*)$=1.564 after 31 iterations.

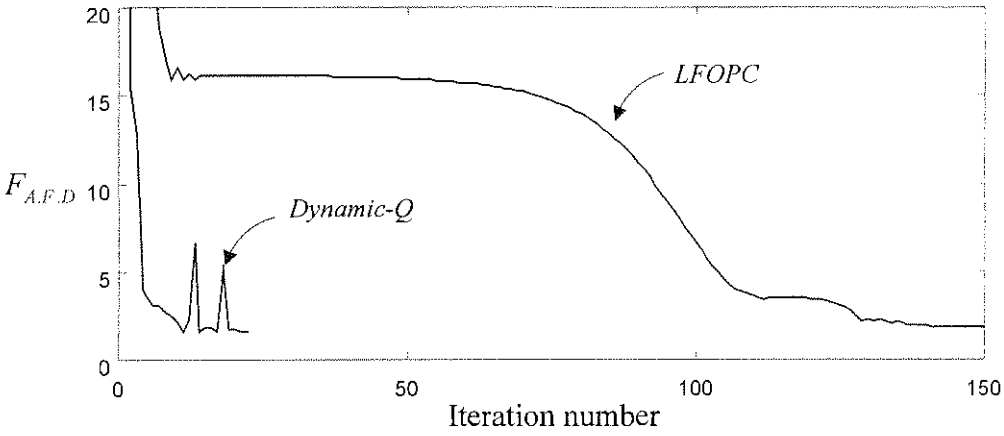

Figure 4.2: Convergence histories of *LFOPC* and *Dynamic-Q*, with x^0=(0.5, 0.8, -0.6, -0.6, 0.5)^T. $F_{LFOPC}(x^*)$=1.552 after 442 iterations and $F_{Dyn.-Q}(x^*)$=1.590 after 22 iterations.

The results for the design problem with six design variables (see Appendix G) show that for the optimum design, the sixth design variable, x_6, is at either of its constraint limits, i.e. there is a local optimum at each of the bounds on the sixth variable. This means that a planar Stewart platform of the type considered here, need not have mechanical allowance for the sixth design variable. However, it is important to check which local optimum, corresponding to x_6=0 or x_6=x_1 respectively, gives the best design.

4.2.1 More general moving platform

The moving platform can also have a more general form than the one used in this study by adding an additional design variable, x_7 as shown in Figure 4.3.

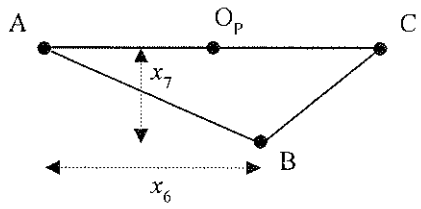

Figure 4.3: A more general moving platform.

The results for the design problem with seven design variables are not listed here, but they show that x_7=0 in the optimum designs. This means that for a practical platform, nothing is gained by adding a feature that will allow for point B to lie away from point A.

Chapter 5: Concluding discussion

This study shows that the optimization methodology presented here, can successfully be applied to the optimal design of a planar Stewart platform required to perform a prescribed task. The mathematical optimization approach yields a good platform design based on the proposed design criteria, whilst at the same time satisfying stipulated design constraints. In particular this method solves the following problems that are inherent to the design of parallel platforms:

1. it finds a design, $\boldsymbol{x}$, for which the given tool path lies inside the workspace,
2. it solves the optimum placement problem of the platform relative to the tool, and
3. it gives a platform design, which has the property that the platform will not assume a near singular configuration as the platform executes its task.

The design methodology described in this study has the important potential application that it can form an integral part of a design software package for planar Stewart platforms to be used in machining operations. This methodology may also be extended to be applicable to the optimal design of spatial Stewart platforms.

5.1 Implications of some specific solutions to the design problems

The solutions to the optimum design problems show that the consideration of the five design variables, x_i, i=1, 2, ..., 5, as defined in this study (see Figure 2.2), are sufficient in determining the optimum design of a practical planar Stewart platform. The sixth design variable (and the seventh as defined in Section 4.2.1) does not affect any further reduction in the cost function, and can be neglected. It is however important to check which of the two extreme designs, corresponding to x_6=0 and x_6=x_1 respectively gives the better design.

Another important implication of the results is that the actuator force cost function, based on the static analysis, $F_{A.F.S}$, gives almost the same results as the computationally more expensive cost function, $F_{A.F.D}$, which is based on dynamic simulation. The cost function, $F_{A.F.S}$, can therefore be used in place of $F_{A.F.D}$ when a platform is to be designed under the criterion of minimization of maximal actuator forces.

This study accentuates the importance of design optimization for planar parallel platforms. Arbitrary feasible designs fare much worse than designs obtained by mathematical programming, when comparing the cost functions of the respective designs. The study also points to the fact that mathematical optimization is probably also very important and a feasible option in the design of other more complicated dynamical systems. It should therefore be applied more commonly in practice as a design tool for engineers.

5.2 Future work

This study serves as a building block in establishing a comprehensive Stewart platform design package. Future work following on this study should be directed at extending the current capabilities of the design software that was developed in this study. The final platform design package should allow the user the choice of other design criteria (such as for example maximizing the manipulator stiffness) and additional design constraints (such as for example maximum actuator velocities and/or accelerations).

The Department of Mechanical Engineering of the University of Pretoria is currently developing such a complete Stewart platform design package. It is envisaged that the software will interface with current existing CAD-software packages in which the tool path is given. The

program will be required to find the optimum platform design that satisfies the criteria specified by the user. Having this information available the operator will be able to make the necessary design settings after which the software system will also control the platform in performing its task.

Appendix A: Tool force

This appendix describes how the tool force is determined for a specific machining operation. It also describes how the tool force is implemented in the dynamic analysis.

A.1 Tool force for machining operation

In this study the only machining operation that is considered is form-milling (Figure A.1) with a ball-nose cutter [25] (Figure A.2).

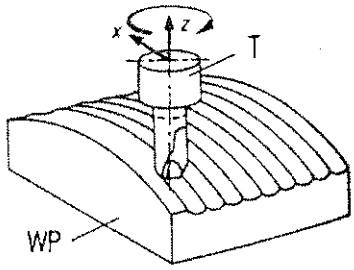

Figure A.1: Form milling.

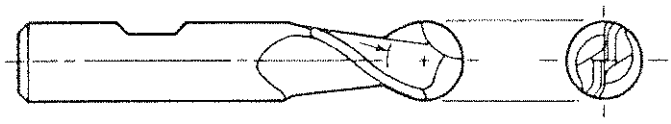

Figure A.2: Ball-nose cutter.

Figure A.3 shows a model for the tool forces on the workpiece [9]. The tool is moving in the x-direction relative to the workpiece. The machining forces on the workpiece, due to one tool tooth can be resolved into an active force, F_a, in the work plane and a passive force, F_p, perpendicular to the work plane. The active force, F_a, can be resolved into a force parallel to the movement of the tool tooth, F_c, and a force perpendicular to F_c, which is F_{cN}. F_c and F_{cN} are forces working in on the workpiece. F_c and F_{cN} are independent of the angle through which the tool has rotated, and is given by Kienzle's machining force equation [9]:

$$F_i = b \cdot k_i h^{1-m_i} \quad \text{where } i = c, cN, p \qquad \textbf{(A-1)}$$

with b the undeformed chip width [mm], k_i the specific cutting force [N.mm^{-2}] for an undeformed chip thickness in the range of [0.1mm, 1.0mm], h the undeformed chip thickness [mm] and m_i the incremental value for the workpiece-cutting material pair.

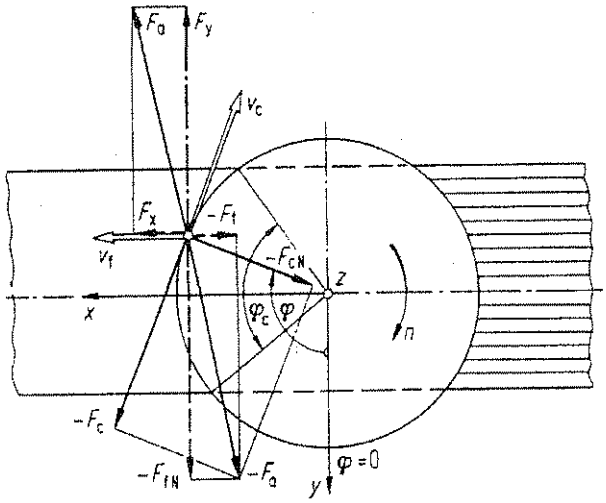

Figure A.3: Model of tool forces.

The k_i and m_i values are empirical, and depend on the workpiece material and tool type. These values are available from tables in reference [9]. The undeformed chip width and chip thickness can be calculated from the feed speed, tool geometry and the tool rotation speed.

The machining forces may vary with time as the tool rotates on the workpiece. For this study however the *maximum* tool force in any given direction at any time is calculated, and is taken as a *constant* tool force.

The machining operation considered is to form-mill an aluminium workpiece with a two teeth ball-nose cutter having a shaft diameter, D, of ½ inch. The tool speed is 2000rpm, the feed speed is 0.25mm/rev, and the depth of the cut is 6.35mm (D/2). The tool forces on the workpiece are then calculated as f_x=1600N, f_y=400N and f_z=1800N, with the directions as shown in Figure A.4.

As is described in Appendix B, the tool velocity is not constant over the tool path, therefore the feed speed also varies over the tool path. But for the sake of simplicity, the feed speed is taken as being constant at 0.25mm/rev, which correspond to the maximum tool velocity that is reached over the tool path.

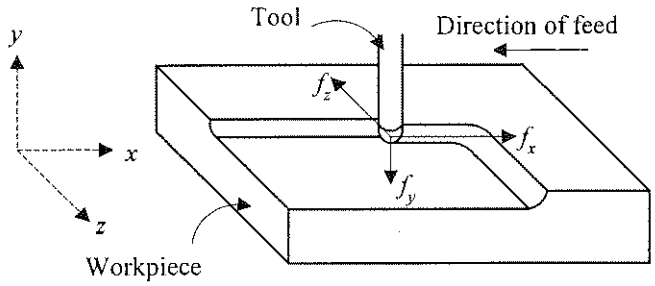

Figure A.4: The machining forces acting in on the workpiece.

A.2 Implementation of tool force for dynamic analysis

The tool forces act in on the workpiece at the contact point between the tool tip and the workpiece (Figure A.5). The tool force components f_x and f_y do not vary with time, and their points of action are fixed at the origin of the global coordinate system OXY, which is chosen to coincide with the tool tip.

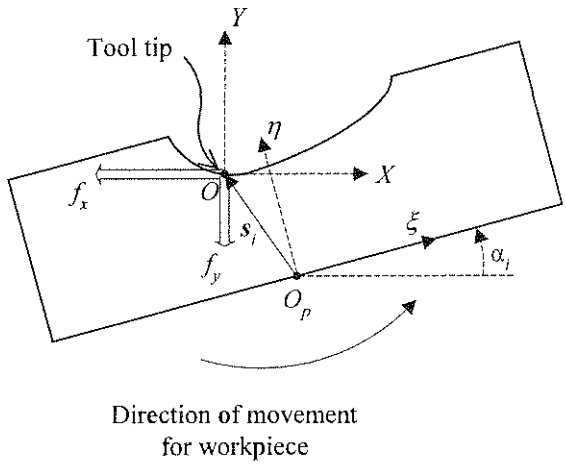

Figure A.5: The tool force components acting in on the workpiece at the tool tip.

For each time instant t_i, i=1, 2, ..., n_p, the tool force components are implemented in the dynamical model as a moment, m_i, and forces, f_x and f_y, acting at position $\boldsymbol{s}_i$ in the local coordinate system $O_P\xi\eta$. The moment m_i is simply found by the cross-product between the vector $\boldsymbol{s}_i$, which is the vector from the platform origin O_P to the tool tip O, and the tool force represented by components f_x and f_y:

$$m_i = \boldsymbol{s}_i \times \begin{bmatrix} f_x \\ f_y \end{bmatrix} \tag{A-2}$$

Appendix B: Determination of actuator leg length drivers for a prescribed tool task path

A parallel platform is designed for the task it should perform. In this study, only machining tasks are considered. The machining task is completely specified by:

1. the tool path that the tool should cut out of the workpiece,
2. the relative angle between the tool axis OY and the tool path, and
3. the tool velocity, i.e. the velocity with which uncut material is fed to the tool.

The tool path needs to be transformed to drivers for the actuators, as these drivers drive the platform to carry out its task. (An actuator driver is a curve specifying the actuator length as a function of time. Each actuator has a unique driver, also sometimes referred to as actuator leg length trajectories).

The actuator drivers for a tool path can be determined by transforming the tool path to a platform path. The platform path is the path that the platform origin traces out as the platform executes its task. The actuator drivers are calculated from the platform path by using the kinematic equations of the platform. The following sections describe how the tool path is specified and how the actuator drivers are calculated from this tool path.

B.1 Machining task

B.1.1 The tool path

The discretised tool is specified by the sequence of vectors $\boldsymbol{s}_i$, i=1,2,...,n_{tp}, in the local coordinate system, $O_P\xi\eta$, which is fixed to the platform, as shown in Figure B.1. A

cubic-spline, $\eta=\Gamma_{tp}(\xi)$, may be constructed from discretised points, giving the η-coordinate of the tool path as a continuous function of the ξ-coordinate. $\eta=\Gamma_{tp}(\xi)$ is created using the n_{tp} points specified, as well as the slope of the tool path at the start and end points of the tool path.

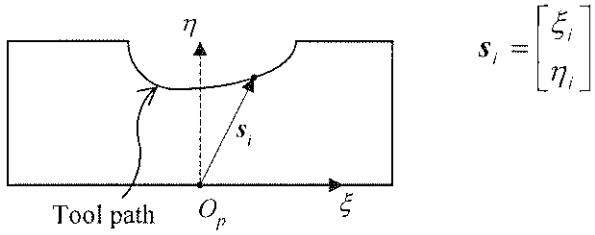

Figure B.1: Tool path in local coordinate system $O_P\xi\eta$.

The actuator drivers are approximated by computing (by means of an appropriate transformation of the local tool path to the global system), the leg lengths that correspond to the n_p discrete points on the local path (n_p may be different from the n_{tp} used in constructing the tool path cubic-spline above).

B.1.2 Relative angle between tool axis *OY* and tool path

Here the global *OX* axis (see Figure 2.1) is chosen to be tangential to the tool path at all times, i.e. the tool axis *OY* is perpendicular to the path. In more general applications of course, the relative angle between the tool axis *OY* and the tool path may be specified as a function of the tool tip position $\boldsymbol{s}$.

B.1.3 Tool velocity

The specification of the platform task is completed by giving the tool velocity. This is the velocity with which uncut material is fed to the tool. It is tangential to the tool path, and is a function of time. In machining applications the maximum tool velocity is usually specified, as well as the maximum magnitude of any acceleration of the workpiece tangential the tool path.

In this study the maximum tool velocity and maximum magnitude of acceleration is v_{max}=0.01 m.s^{-1} and a_{max}=0.005 m.s^{-2} respectively. For the machining task to be completed as quickly as possible, the tool velocity should have a profile as shown in Figure B.2.

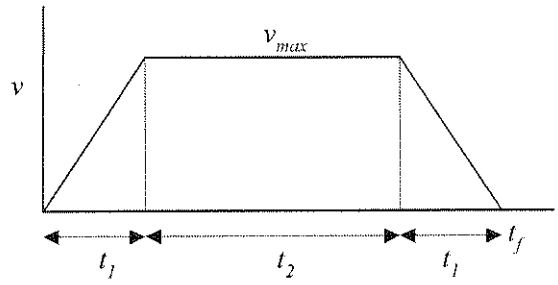

Figure B.2: Velocity profile of tool relative to workpiece.

The time to complete the task, t_f, is calculated using the known relationship:

$$s = ut + \frac{1}{2}at^2 \qquad \textbf{(B-1)}$$

where s is the distance traveled in time t, u is the initial velocity and a is the constant acceleration.

If the length of the tool path is l_{path}, the time to complete the task, t_f, is calculated as follows (refer to Figure B.2):

$$t_1 = \frac{v_{max}}{a_{max}}$$

$$l_{path} = \frac{1}{2}a_{max}t_1^2 + v_{max}t_2 + (v_{max}t_1 - \frac{1}{2}a_{max}t_1^2) \qquad \textbf{(B-2)}$$

$$\Rightarrow t_2$$

$$t_f = t_1 + t_2 + t_1$$

l_{path} is approximated by the Riemann-sum of the line integral, where the tool path is discretised for this purpose by n_{path} points:

$$l_{path} = \sum_{j=1}^{n_{path}-1} \Delta s_j \approx \int_{path} ds \qquad \text{(B-3)}$$

$$\text{with } \Delta s_j = \sqrt{(\xi_{j+1} - \xi_j)^2 + (\eta_{j+1} - \eta_j)^2}$$

With the time t_f known, a cubic-spline of the velocity profile, $v=\Gamma_v(t)$, is generated. This cubic-spline is integrated to give a cubic-spline, $s=\Gamma_s(t)$, giving the travel of the tool along the tool path. The abscissa of the cubic-spline $s=\Gamma_s(t)$, are interchanged to give a cubic-spline $t=\Gamma_t(s)$ from which the time, t, it takes the platform to cover a distance, s, along the tool path from the starting point, may be determined as a function of s.

The time instant, t, at which the tool moves through point i, is calculated by determining the distance, s, the tool has traveled from the starting point to point i. The time, t, is simply found by evaluating $t=\Gamma_t(s)$.

If S denote the set of points on the tool path, such that $\boldsymbol{s}_i \in S$, $i=1, 2, ..., n_p$, and if Ω denote the set of time instances associated with the points $\boldsymbol{s}_i$, such that $t_i \in \Omega$, $i=1, 2, ..., n_p$, then t_i is the time at which the platform passes through point $\boldsymbol{s}_i$. The set of points, S, and the set of associated time instants Ω, completely describes the manipulator task. The set of points S is transformed to give the platform path, which in turn is transformed to give the actuator drivers.

For the two tool paths shown in Figure 2.3 the tool path length, l_{path}, and time to complete the task, t_f, are the same, namely l_{path}=0.01m and t_f=12s (t_1=2s and t_2=8s).

B.2 Platform path

The platform path is the path that the platform origin, O_P, traces in the global coordinate system OXY, as the platform executes its task. It is calculated for each point i by a transformation of the point $\boldsymbol{s}_i$ on the tool path.

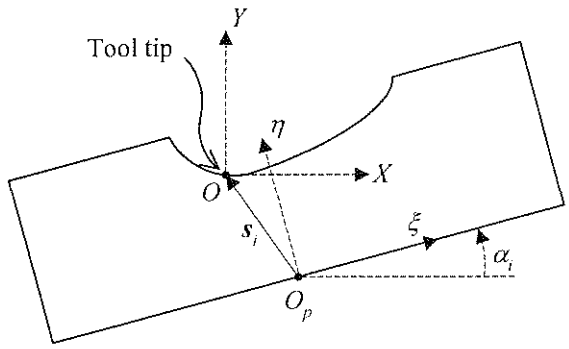

Figure B.3: Tool path with global (OXY) and local ($O_P\xi\eta$) coordinate systems shown.

The vector $\boldsymbol{s}_i$ (Figure B.3) is a vector from the platform origin O_P to the tool tip (situated at the origin O). The vector $-\boldsymbol{s}_i$ gives the coordinates of the platform $(X_P, Y_P)_i$ in the global coordinate system. Vector $\boldsymbol{s}_{i\,O_P\xi\eta}$ given in the local coordinate system $O_P\xi\eta$ can be transformed to the global coordinate system OXY by the transformation:

$$\boldsymbol{s}_{i\,OXY} = A_i \boldsymbol{s}_{i\,O_P\xi\eta}$$
$$\text{with } A_i = \begin{bmatrix} \cos\alpha_i & -\sin\alpha_i \\ \sin\alpha_i & \cos\alpha_i \end{bmatrix} \qquad \textbf{(B-4)}$$

The orientation of the platform at point i is α_i which is found using the fact that the tool axis OX is always tangential to the tool path. The cubic-spline giving the slope of the tool path in the local coordinate system, is given by the derivative of the tool path cubic-spline, $\eta=\Gamma_{tp}(\xi)$, with respect to ξ, i.e. $\dfrac{\partial\Gamma_{tp}(\xi)}{\partial\xi} = \Gamma_{slope}(\xi)$. The platform orientation α_i is then simply found by:

$$\alpha_i = -\text{atan}\left\{\Gamma_{slope}(\xi_i)\right\} \qquad \textbf{(B-5)}$$

The actuator drivers are calculated using the points on the platform path, $(X_P, Y_P)_i$ and the platform orientation, α_i for i=1, 2, ..., n_p.

B.3 Actuator drivers

Here follows a derivation of expressions for the respective lengths l_1, l_2, and l_3, in terms of the global position of the platform origin, (X_P, Y_P), the orientation of the platform α and the design variables x_i, i=1, 2, ..., 6.

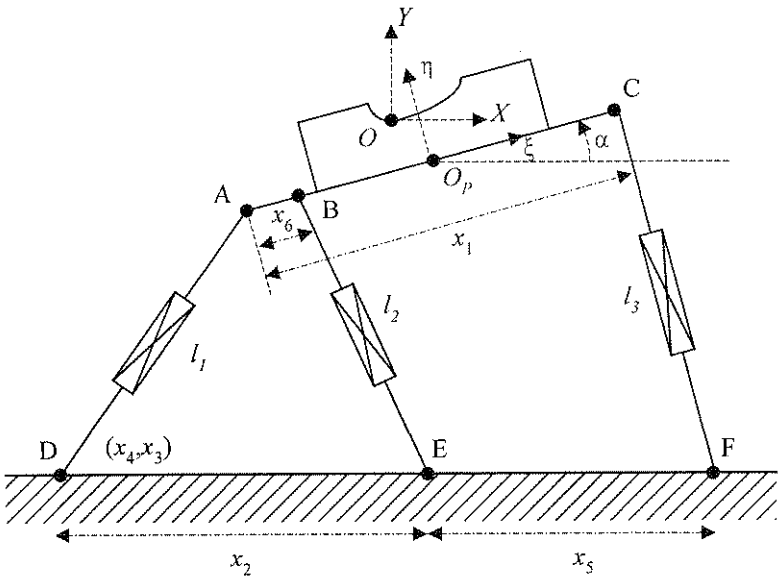

Figure B.4: Design variables of manipulator.

With reference to Figure B.4 the folowing relationships may be derived:

$$
\begin{aligned}
X_A &= -\frac{x_1}{2}\cos\alpha + X_P \\
Y_A &= -\frac{x_1}{2}\sin\alpha + Y_P \qquad & X_D &= x_3 \\
X_B &= X_A + x_6\cos(\alpha) & X_E &= X_D + x_2 = x_3 + x_2 \\
Y_B &= Y_A + x_6\sin(\alpha) & X_F &= X_E + x_5 = x_3 + x_2 + x_5 \\
X_C &= \frac{x_1}{2}\cos\alpha + X_P & Y_D &= Y_E = Y_F = x_4 \\
Y_C &= \frac{x_1}{2}\sin\alpha + Y_P
\end{aligned}
\qquad \text{(B-6)}
$$

$$
\begin{aligned}
l_1 &= \left[(X_A - X_D)^2 + (Y_A - Y_D)^2\right]^{\frac{1}{2}} \\
l_2 &= \left[(X_B - X_E)^2 + (Y_B - Y_E)^2\right]^{\frac{1}{2}} \\
l_3 &= \left[(X_C - X_F)^2 + (Y_C - Y_F)^2\right]^{\frac{1}{2}}
\end{aligned}
\qquad \text{(B-7)}
$$

The actuator drivers are found by finding $l_{j,i}$, j=1, 2, 3, for each point i, i=1, 2, ..., n_p, on the platform path. Each point $(X_P, Y_P)_i$, on the platform path is associated with a time instant t_i from the set Ω, at which the origin of the platform moves through that point. The combination of $l_{j,i}$ and t_i gives the actuator drivers.

Appendix C: The Jacobian matrix and the quality index

The Jacobian matrix, or simply the Jacobian, plays an important role in the kinematic and dynamic analysis of a mechanical system. This matrix appears in the velocity and acceleration equations of a system. The following sub-section describes exactly what the Jacobian is.

The quality index, to be dealt with later, is a dimensionless ratio that measures the "closeness" of a manipulator to a so-called singular configuration. It can be used as a design criterion for manipulators. In the last sub-section the quality index is defined and its use discussed.

C.1 Jacobian

Any set of variables that uniquely describes the position and orientation of all bodies in a system, is called a set of generalized coordinates. A set of generalized coordinates may be represented by the column vector $\boldsymbol{q} = [q_1,\ q_2,\ ...,\ q_{nc}]^T$, where nc is the number of generalized coordinates used to describe the system. The vector of planar Cartesian generalized coordinates for body i is, $\boldsymbol{q}_i = [x, y, \phi]_i^T$, where x and y gives the position of the origin of body i in the global coordinate system, and ϕ gives its orientation. For a planar system with nb bodies, there are $nc = 3 \cdot nb$ planar Cartesian generalized coordinates, and the vector of generalized coordinates is then $\boldsymbol{q} = [\boldsymbol{q}_1^T,\ \boldsymbol{q}_2^T,\ ...,\ \boldsymbol{q}_{nb}^T]^T$.

There are kinematic constraints that apply to a system, such as, for example, a planar revolute joint, which constrains two points on two separate bodies to have common global coordinates x and y. A kinematic constraint on a system can be written in the form of $\Phi^{(K)}(\boldsymbol{q}) = 0$. Note that the kinematic constraints are not explicit functions of time. All the kinematic constraints on the

bodies in a system can be combined by writing them in vector form as $\Phi^{(K)}(\boldsymbol{q}) = \left[\Phi_1^{(K)}(\boldsymbol{q}),\ \Phi_2^{(K)}(\boldsymbol{q}),\ ...,\ \Phi_k^{(K)}(\boldsymbol{q})\right]^T = \mathbf{0}$, where k is the number of kinematic constraints.

There may also be driving constraints which apply to a system. An example of a driving constraint is a planar actuator, which specify the distance between two points on two separate bodies as a function of time. A driving constraint can be written in the form of $\Phi^{(D)}(\boldsymbol{q},t) = 0$. Combining all the driving constraints they can also be written in vector form as $\Phi^{(D)}(\boldsymbol{q},t) = \left[\Phi_1^{(D)}(\boldsymbol{q},t),\ \Phi_2^{(D)}(\boldsymbol{q},t),\ ...,\ \Phi_d^{(D)}(\boldsymbol{q},t)\right]^T = \mathbf{0}$ where d is the number of driving constraints.

The kinematic and driving constraints on a system such as a manipulator can be combined to give:

$$\Phi(\boldsymbol{q},t) = \begin{bmatrix} \Phi^{(K)}(\boldsymbol{q}) \\ \Phi^{(D)}(\boldsymbol{q},t) \end{bmatrix} = \mathbf{0} \qquad \textbf{(C-1)}$$

Differentiating (C-1) by means of the chain rule gives:

$$\begin{aligned} &\dot{\Phi} = \Phi_q \dot{\boldsymbol{q}} + \Phi_t = \mathbf{0} \\ &\Phi_q \dot{\boldsymbol{q}} = -\Phi_t \\ &\text{where } (\Phi_q)_{i,j} = \frac{\partial \Phi_i}{\partial \boldsymbol{q}_j},\ i = 1,\ 2,\ ...,\ k+d;\ j = 1,\ 2,\ ...,\ nc \\ &\qquad\qquad \text{and } k + d = nc \end{aligned} \qquad \textbf{(C-2)}$$

The relationship in (C-2) is used to solve for $\dot{\boldsymbol{q}}$ at discrete instants of time, if Φ_q is nonsingular.

The Φ_q matrix is called the Jacobian. If its determinant is zero, i.e. if $|\Phi_q| = 0$, it is singular and the associated manipulator is in a singular configuration.

C.1.1 Actuator forces

The actuator forces for a planar parallel platform to be *held in a fixed position* can be calculated by means of the Jacobian as follows [18]:

$$\boldsymbol{w} = \Phi_q \boldsymbol{f}$$

$$\text{where } \boldsymbol{w} = \begin{bmatrix} f_x \\ f_y \\ m_z \end{bmatrix}; \text{ and } \boldsymbol{f} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} \qquad \text{(C-3)}$$

where the force applied by the platform is given by $\boldsymbol{w}$, and the actuator forces by $\boldsymbol{f}$. Here f_1, f_2 and f_3 are the actuator forces in actuator 1, 2 and 3 respectively and f_x and f_y are the global components of the resultant force applied by the platform at a point of interest on the platform, and m_z is the resultant moment of the forces about the same point. It also follows that $\boldsymbol{f} = \Phi_q^{-1} \boldsymbol{w}$.

C.2 Quality index

The quality index of a static manipulator configuration was originally defined by Keller and is given in the article of Lee et. al. [18] as:

$$\lambda = \frac{\text{abs}|\Phi_q|}{|\Phi_q|_{\max}} \qquad \text{(C-4)}$$

where $|\Phi_q|_{\max}$ is the maximum determinant of the Jacobian that can be attained for a given manipulator. Lee, et. al. [18] show that for a planar parallel platform, $|\Phi_q|_{\max} = \frac{L_1 + L_2 + L_3}{2}$

where L_1, L_2 and L_3 are the lengths of the sides of the manipulator (Figure C.1).

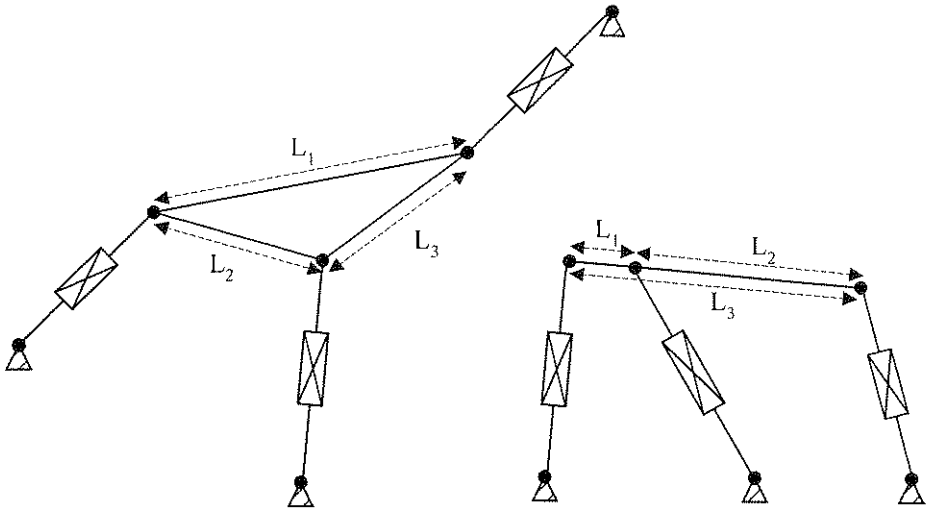

Figure C.1: The lengths of the sides of planar parallel manipulators.

The quality index has the following properties:

1. λ has a minimum value of zero, which is attained in a singular configuration, and λ is therefore very small for near-singular configurations;
2. λ has a maximum value of one;
3. if λ is very small the actuator forces required to hold the manipulator in a static configuration become very large.

Appendix D: Computation of actuator forces through static analysis

The Jacobian can be used to determine the actuator forces for a planar parallel platform, given the external forces that act on the platform. This method gives the static forces in the actuators for a particular platform position and orientation (see paragraph C.1.1 in Appendix C for the calculation). This section describes how the external forces working in on the platform are determined.

There are two types of external forces on the platform that have to be considered in a static analysis. These are the tool force and the gravitational force. The tool force is discussed in Appendix A, where it is shown that the tool force can be applied to the platform as a resultant force and a resultant moment acting at the platform origin. The gravitational force can be applied to the platform in the same manner.

Figure D.1 shows the center of gravity, with the gravitational force, f_g, working through the center of gravity. The gravitational force can be applied as an external force and a moment acting at the platform origin, O_P. The gravitational force has a magnitude of, $|f_g|$, and works in the Y-direction of the global coordinate system OXY. The external moment, m_g, is found by the cross-product, $m_g = \boldsymbol{g} \times \begin{bmatrix} 0 \\ f_g \end{bmatrix}$.

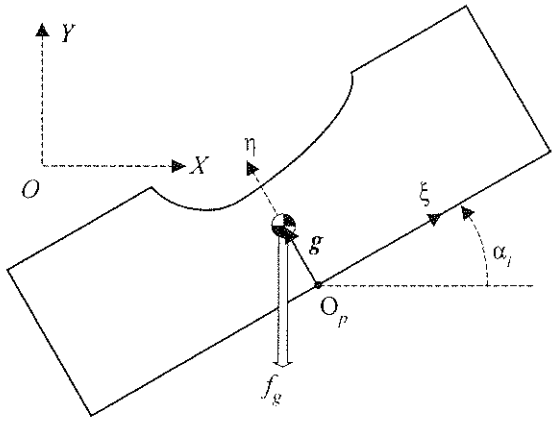

Figure D.1: The gravitation force, f_g, working in on the platform.

D.1 Actuator forces

The resultant external forces that are applied to the platform, are given by f_e. In order for the platform to be in a static position and orientation, the resultant force applied by the platform, w, should balance the resultant external forces acting in on to the platform, f_e, i.e.:

$$w = -f_e \tag{D-1}$$

The resultant external forces, f_e, are applied to some point of interest on the platform (C-3). In this case the point of interest is chosen to coincide with the platform origin, O_p. The resultant external forces, f_e, are therefore given as:

$$f_e = \begin{bmatrix} f_x^{tool} \\ f_y^{tool} \\ m_z^{tool} \end{bmatrix} + \begin{bmatrix} 0 \\ f_g \\ m_g \end{bmatrix} \tag{D-2}$$

and the resultant force applied by the platform are:

$$w = -\left(\begin{bmatrix} f_x^{tool} \\ f_y^{tool} \\ m_z^{tool} \end{bmatrix} + \begin{bmatrix} 0 \\ f_g \\ m_g \end{bmatrix} \right) \tag{D-3}$$

With w known it follows that the actuator forces f may be determined from Equation (C-3):

$$f = \Phi_q^{-1} w \tag{D-4}$$

Appendix E: *LFOPC*

The *LFOPC* (*Leap-Frog Optimization Program for Constrained optimization*) algorithm applies a dynamic trajectory method for unconstrained optimization, originally proposed and developed by Snyman [26, 27], to a penalty function formulation of the constrained problem [28, 32, 35]. The original dynamic trajectory method is based on the physical model of the motion of a particle of unit mass in an n-dimensional conservative force field, where the potential energy of the particle is given by the function to be minimized $F(\boldsymbol{x})$. This method is a proven reliable and robust method.

The *LFOPC* algorithm is highly suited for the type of optimization problems encountered in the engineering field. Optimization difficulties encountered in engineering problems are typically that:

1. the functions (which can be the cost function or the constraint functions) are expensive to evaluate, such as when evaluated through *CFD* or *FEM* simulations,
2. there may be noise present in these functions, that originate from experimental data, or from numerical inaccuracies,
3. the functions may be discontinuous or non-differentiable, as is the case with the actuator force cost function used in this study,
4. the functions may have multiple local minima, as is again the case with the actuator force cost function,
5. there may be regions in the design space where the functions are not defined, and
6. some design problems may have an extremely large number of design variables.

These difficulties limit the application of traditional optimization algorithms to engineering problems. *LFOPC* is a very robust algorithm that has been successfully applied to design problems that possess some of the difficulties stated above [2, 30, 32, 35, 36, 40]. The basics of the leap-frog trajectory method are briefly discussed below.

E.1 Basic dynamic model

Assume a particle of unit mass in a n-dimensional conservative force field with potential energy at $\boldsymbol{x}$ given by $F(\boldsymbol{x})$, then at $\boldsymbol{x}$ the force on the particle is given by:

$$\boldsymbol{a} = \ddot{\boldsymbol{x}} = -\nabla F(\boldsymbol{x}) \tag{E-1}$$

from which is follows that for the time interval $[0,t]$:

$$\begin{aligned} &\frac{1}{2}\|\dot{\boldsymbol{x}}(t)\|^2 - \frac{1}{2}\|\dot{\boldsymbol{x}}(0)\|^2 = F(\boldsymbol{x}(0)) - F(\boldsymbol{x}(t)) \\ &T(t) - T(0) = F(0) - F(t) \\ &\text{or} \\ &F(t) + T(t) = \text{constant} \end{aligned} \tag{E-2}$$

where $F(t)$ and $T(t)$ is the potential energy and kinetic energy of the particle respectively, at time t.

Note that $\Delta F = -\Delta T$, therefore as long as T increases, F decreases. This is the basis of the dynamic algorithm.

E.2 Basic *LFOP* algorithm for unconstrained problems

Given $F(\boldsymbol{x})$ and the starting point $\boldsymbol{x}(0)=\boldsymbol{x}^0$, compute the trajectory by solving the initial value problem (*IVP*):

$$\begin{aligned} \ddot{\boldsymbol{x}}(t) &= -\nabla F(\boldsymbol{x}(t)) \\ \dot{\boldsymbol{x}}(0) &= 0 \\ \boldsymbol{x}(0) &= \boldsymbol{x}^0 \end{aligned} \qquad \text{(E-3)}$$

Monitor $\dot{\boldsymbol{x}}(t) = \boldsymbol{v}(t)$, for while $T(t) = \frac{1}{2}\|\boldsymbol{v}(t)\|^2$ increases, $F(\boldsymbol{x}(t))$ decreases, and the solution $\boldsymbol{x}$ is moving towards a minimum of F. When $\|\boldsymbol{v}(t)\|$ decreases, the solution $\boldsymbol{x}$ is moving "uphill". In this case an *interfering strategy* is applied to extract energy from the particle, in order to increase the likelihood of descent.

In practice the numerical integration of the *IVP* in (E-3) is done by means of the leap-frog method:

Compute for k=0, 1, ... and time step Δt:

$$\begin{aligned} \boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} + \boldsymbol{v}^{(k)}\Delta t \\ \boldsymbol{v}^{(k+1)} &= \boldsymbol{v}^{(k)} + \boldsymbol{a}^{(k+1)}\Delta t \end{aligned} \qquad \text{(E-4)}$$

where

$$\begin{aligned} \boldsymbol{a}^{(k)} &= -\nabla f(\boldsymbol{x}^{(k)}) \\ \boldsymbol{v}^{(0)} &= \frac{1}{2}\boldsymbol{a}^{(0)}\Delta t \end{aligned} \qquad \text{(E-5)}$$

The typical *interfering strategy* referred to above is:

$$\begin{aligned} &\text{If } \left\|\boldsymbol{v}^{(k+1)}\right\| \geq \left\|\boldsymbol{v}^{(k)}\right\| \text{ continue} \\ &\quad \text{else set } \boldsymbol{v}^{(k)} = \frac{\boldsymbol{v}^{(k+1)} + \boldsymbol{v}^{(k)}}{4};\quad \boldsymbol{x}^{(k)} = \frac{\boldsymbol{x}^{(k+1)} + \boldsymbol{x}^{(k)}}{2} \end{aligned} \qquad \text{(E-6)}$$

compute the new $\boldsymbol{v}^{k+1}$ and continue.

Three termination criteria are used:

1. Stop if $\left\|\boldsymbol{x}^{(k)}-\boldsymbol{x}^{(k-1)}\right\|<\varepsilon_x$
2. Stop if $\left\|\boldsymbol{a}^{(k)}\right\|<\varepsilon_g$
3. Stop if the maximum number of iterations is exceeded.

Other heuristics are incorporated, such as specifying a maximum trajectory step size *DELT*, an empirical formula for determining the initial value for Δt, and a scheme to magnify and reduce Δt in order to minimize the number of iterations required for convergence.

E.3 Modification for constraints

Constraints are accommodated in the algorithm by means of a penalty function formulation. This formulation solves the constrained minimization problem stated in (2-1), by the unconstrained minimization of a modified cost function, $P(\boldsymbol{x},\mu)$:

$$P(\boldsymbol{x},\mu)=F(\boldsymbol{x})+\mu\sum_{j=1}^{m}[c_j(\boldsymbol{x})]^2u_j(c_j)+\mu\sum_{k=1}^{p}[h_k(\boldsymbol{x})]^2 \qquad \textbf{(E-7)}$$

where

$$u_j(c_j)=\begin{cases}0 \text{ if } c_j(x)\le 0\\ 1 \text{ if } c_j(x)>0\end{cases} \qquad \textbf{(E-8)}$$

and where $\mu>>0$ is the overall penalty parameter.

The solution to the minimization problem is done in three phases:

Phase 0

Given some $\boldsymbol{x}^0$, apply *LFOP* to $P(\boldsymbol{x},\mu_0)$ with an overall penalty parameter of $\mu=\mu_0=10^2$ to give $\boldsymbol{x}^*(\mu_0)$.

Phase 1

With $\boldsymbol{x}^0 = \boldsymbol{x}^*(\mu_0)$ and $\mu = \mu_1 = 10^4$, apply *LFOP* to $P(\boldsymbol{x}, \mu_1)$ to give $\boldsymbol{x}^*(\mu_1)$. Identify the active constraints $i_a = 1, 2, \ldots, n_a$ where $c_{i_a}(\boldsymbol{x}^*(\mu_1)) > 0$.

Phase 2

With $\boldsymbol{x}^0 = \boldsymbol{x}^*(\mu_1)$ apply *LFOP* to:

$$\text{minimize } P_a(\boldsymbol{x}, \mu_1) = \mu_1 \sum_{k=1}^{p} h_k^2(\boldsymbol{x}) + \mu_1 \sum_{i_a=1}^{n_a} c_{i_a}^2(\boldsymbol{x})$$

to give $\boldsymbol{x}^*$, where $P_a(\boldsymbol{x}, \mu_1)$ is a function of only the equality constraint h_j, $j = 1, 2, \ldots, p$, and the active constraints c_{i_a}, $i_a = 1, 2, \ldots, n_a$.

Appendix F: *Dynamic-Q*

The *Dynamic-Q* algorithm was proposed and developed by Snyman et. al. [34], and further developed by Craig et. al. [4, 5, 7]. In the field of engineering, the functions of an optimization problem are often very expensive to evaluate. These optimization problems can be solved economically by using approximation methods that require less function evaluations than standard gradient based descent algorithms. *Dynamic-Q* is such a method.

F.1 Basic algorithm

Dynamic-Q adopts a successive approximation approach to solve the minimization problem in (2-1). It solves successive sub-problems constructed from the original problem (2-1). These sub-problems are analytically simple and can easily and economically be solved by the dynamic trajectory algorithm *LFOPC*.

F.1.1 Construction of successive sub-problems

A series of successive approximate quadratic sub-problems $P[k]$, k=1, 2, ..., can be constructed as follows:

Suppose $F(\boldsymbol{x}^{(k)})$, $c_j(\boldsymbol{x}^{(k)})$, $\nabla F(\boldsymbol{x}^{(k)})$ and $\nabla c_j(\boldsymbol{x}^{(k)})$ are known at the design point $\boldsymbol{x}^{(k)}$, construct the approximate spherical quadratic function:

$$\tilde{F}(\boldsymbol{x}) = F(\boldsymbol{x}^{(k)}) + \nabla^T F(\boldsymbol{x}^{(k)})(\boldsymbol{x}-\boldsymbol{x}^{(k)}) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}^{(k)})^T P_0^{(k)}(\boldsymbol{x}-\boldsymbol{x}^{(k)}) \qquad \textbf{(F-1)}$$

where $P_0^{(k)} = diag(p_0^{(k)}, p_0^{(k)}, \ldots, p_0^{(k)}) = p_0^{(k)} I$

$p_0^{(1)}$ depends on the specific problem (typically specified as a small positive number or zero), thereafter compute $p_0^{(k)}$ as follows:

$$p_0^{(k)} = \frac{2\{F(x^{(k-1)}) - F(x^{(k)}) - \nabla^T F(x^{(k)})(x^{(k-1)} - x^{(k)})\}}{\left\| x^{(k-1)} - x^{(k)} \right\|^2}; \; k = 2, 3, \ldots \qquad \textbf{(F-2)}$$

Similarly, construct approximations to the constraint functions:

$$\tilde{c}_j(\boldsymbol{x}) = c_j(\boldsymbol{x}^{(k)}) + \nabla^T c_j(\boldsymbol{x}^{(k)})(\boldsymbol{x} - \boldsymbol{x}^{(k)}) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(k)})^T P_j^{(k)} (\boldsymbol{x} - \boldsymbol{x}^{(k)}); \; j = 1, 2, \ldots, m$$
$$\text{where } P_j^{(k)} = diag(p_j^{(k)}, p_j^{(k)}, \ldots, p_j^{(k)}) = p_j^{(k)} I \qquad \textbf{(F-3)}$$

and $p_j^{(k)}$ selected and computed in a similar manner as $p_0^{(k)}$, i.e.:

$$p_j^{(k)} = \frac{2\{c_j(\boldsymbol{x}^{(k-1)}) - c_j(\boldsymbol{x}^{(k)}) - \nabla^T c_j(\boldsymbol{x}^{(k)})(\boldsymbol{x}^{(k-1)} - \boldsymbol{x}^{(k)})\}}{\left\| \boldsymbol{x}^{(k-1)} - \boldsymbol{x}^{(k)} \right\|^2}; \; k = 2, 3, \ldots \qquad \textbf{(F-4)}$$

To avoid too high convexity or concavity, the values of $p_j^{(k)}$ are limited to a range, to force $p_j^{(k)}$ to be small in absolute value.

To control convergence, move limits are also introduced:

$$x_i - x_i^{(k)} - \Delta_i \le 0$$
$$-x_i + x_i^{(k)} - \Delta_i \le 0; \; i = 1, 2, \ldots, n \qquad \textbf{(F-5)}$$

Typically $\Delta_i = \Delta$ for all i.

F.2 Formal *Dynamic-Q* procedure

The *Dynamic-Q* procedure is as follows:

1. Given $\boldsymbol{x}^{(1)}$, start with k=1.

2. Solve sub-problem $P[k]$ using *LFOPC*:

$$\underset{\boldsymbol{x}}{\text{minimize}}\ \widetilde{F}(\boldsymbol{x}) \text{ with } \boldsymbol{x} = [x_1, x_2, ..., x_n]^T;\ x_i \in \Re$$

$$\text{such that } \begin{cases} \widetilde{c}_j(\boldsymbol{x}) \le 0; j = 1,2,..., m \\ x_i - x_i^{(k)} - \Delta_i \le 0 \\ -x_i + x_i^{(k)} - \Delta_i \le 0;\ i = 1, 2, ..., n \end{cases}$$

where $\widetilde{F}(\boldsymbol{x})$ and $\widetilde{c}_j(\boldsymbol{x}^{(k)})$ are computed according to (F-1) and (F-3) to give the solution $\boldsymbol{x}^{*(k)}$.

3. Increment k, i.e. k=k+1, then set $\boldsymbol{x}^{(k)} = \boldsymbol{x}^{*(k-1)}$. Check whether the solution has converged:

   Stop if $\left\| \boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)} \right\| < \varepsilon$,

   Otherwise repeat step 2.

Appendix G: Results

A summary of the results obtained for the design problems are given here. Firstly the results obtained using the direct *LFOPC* algorithm are given, followed by the results computed by the approximation method *Dynamic-Q*. Two different tool paths, A and B, corresponding to the two paths given in Figure 2.3 are considered. Different initial designs, $\boldsymbol{x}^0$, are used and the number of design variables are also varied as indicated below.

G.1 *LFOPC*

The parameter settings for the $F_{A.F.D}$ cost function throughout are ε_x=10^{-5}, ε_g=10^{-5} and the maximum number of allowable steps per phase is 1000. The $F_{A.F.S}$ and $F_{Q.I}$ cost functions are calculated more accurately than $F_{A.F.D}$, so the parameter settings of *LFOPC* when these functions are used are throughout, ε_x=10^{-5}, ε_g=10^{-7} and the maximum number of allowable steps per phase is 2000. For three design variables *DELT*=0.35 throughout and for five design variables *DELT*=0.45 throughout.

G.1.1 Three design variables

Here only three design variables are used, the other design variables are fixed at the values x_4=-0.8, x_5=0.7 and x_6=0.0. For the constraints on the variables (see (2-3)) the following limits are set: $0.1 \le x_1 \le 0.75$, $0.1 \le x_2 \le 0.8$, $x_3 \le 0$.

Table G-1: Violated constraints at x^0 for tool path A.

x^0	Violated constraints
$(0.4, 0.6, -0.6)^T$	$\min[l_2] \geq \underline{l}_2$ $\min[l_3] \geq \underline{l}_3$
$(0.7, 0.4, -1.0)^T$	none
$(0.5, 0.8, -0.6)^T$	$\min[l_2] \geq \underline{l}_2$

Table G-2: Tool path A, three design variables.

$F(x)$ used	x^0	$F(x^0)$	x^*	$F(x^*)$	Active constraints	No. of steps
$F_{A.F.D}$	$(0.4, 0.6, -0.6)^T$	10.107	$(0.123, 0.800, -0.995)^T$	4.887	$\max[l_1] \leq \bar{l}_1$ $x_2 \leq 0.8$	182
	$(0.7, 0.4, -1.0)^T$	5.313	$(0.750, 0.604, -0.932)^T$	4.795	$\min[l_2] \geq \underline{l}_2$ $x_1 \leq 0.75$	84
	$(0.5, 0.8, -0.6)^T$	$4.4 \cdot 10^6$	$(0.469, 0.863, -0.837)^T$	8.481	$\min[l_2] \geq \underline{l}_2$ $x_2 \leq 0.8$	15
$F_{A.F.S}$	$(0.4, 0.6, -0.6)^T$	10.104	$(0.125, 0.800, -0.995)^T$	4.883	$\max[l_1] \leq \bar{l}_1$ $x_2 \leq 0.8$	2275
	$(0.7, 0.4, -1.0)^T$	5.308	$(0.750, 0.604, -0.932)^T$	4.793	$\min[l_2] \geq \underline{l}_2$ $x_1 \leq 0.75$	290
	$(0.5, 0.8, -0.6)^T$	117.16	$(0.750, 0.748, -1.045)^T$	4.814	$x_1 \leq 0.75$	1282
$F_{Q.I}$	$(0.4, 0.6, -0.6)^T$	-0.120	$(0.750, 0.513, -0.925)^T$	-0.327	$\min[l_2] \geq \underline{l}_2$ $x_1 \leq 0.75$	436
	$(0.7, 0.4, -1.0)^T$	-0.287	$(0.750, 0.513, -0.925)^T$	-0.327	$\min[l_2] \geq \underline{l}_2$ $x_1 \leq 0.75$	311
	$(0.5, 0.8, -0.6)^T$	-0.012	$(0.750, 0.513, -0.925)^T$	-0.327	$\min[l_2] \geq \underline{l}_2$ $x_1 \leq 0.75$	350

Table G-3: Violated constraints at x^0 for tool path B.

x^0	Violated constraints
$(0.4, 0.6, -0.6)^T$	$\min[l_2] \geq \underline{l}_2$ $\min[l_3] \geq \underline{l}_3$
$(0.7, 0.4, -1.0)^T$	none
$(0.5, 0.8, -0.6)^T$	$\min[l_2] \geq \underline{l}_2$ $\min[l_3] \geq \underline{l}_3$

Table G-4: Tool path B, three design variables.

$F(x)$ used	x^0	$F(x^0)$	x^*	$F(x^*)$	Active constraints	No. of steps
$F_{A.F.D}$	$(0.4, 0.6, -0.6)^T$	4.332	$(0.184, 0.800, -1.019)^T$	3.392	$\max[l_1] \leq \bar{l}_1$ $x_2 \leq 0.8$	593
	$(0.7, 0.4, -1.0)^T$	4.264	$(0.606, 0.467, -0.864)^T$	3.839	$\min[l_2] \geq \underline{l}_2$ $\min[l_3] \geq \underline{l}_3$	158
	$(0.5, 0.8, -0.6)^T$	5.117	$(0.184, 0.800, -1.019)^T$	3.392	$\max[l_1] \leq \bar{l}_1$ $x_2 \leq 0.8$	316
$F_{A.F.S}$	$(0.4, 0.6, -0.6)^T$	4.327	$(0.187, 0.800, -0.988)^T$	3.384	$x_2 \leq 0.8$	4027#
	$(0.7, 0.4, -1.0)^T$	4.260	$(0.575, 0.472, -0.854)^T$	3.834	$\min[l_2] \geq \underline{l}_2$ $\min[l_3] \geq \underline{l}_3$	1198#
	$(0.5, 0.8, -0.6)^T$	5.114	$(0.190, 0.800, -0.937)^T$	3.389	$x_2 \leq 0.8$	4021#
$F_{Q.I}$	$(0.4, 0.6, -0.6)^T$	-0.321	$(0.750, 0.653, -0.944)^T$	-0.476	$\min[l_2] \geq \underline{l}_2$ $x_1 \leq 0.75$	318
	$(0.7, 0.4, -1.0)^T$	-0.306	$(0.750, 0.653, -0.944)^T$	-0.476	$\min[l_2] \geq \underline{l}_2$ $x_1 \leq 0.75$	308
	$(0.5, 0.8, -0.6)^T$	-0.209	$(0.750, 0.653, -0.944)^T$	-0.476	$\min[l_2] \geq \underline{l}_2$ $x_1 \leq 0.75$	307

Note: # Optimization algorithm terminated on maximum number of steps allowed.

G.1.2 Five design variables

Five design variables are considered here. For the constraints on the variables (see (2-3)) the following limits are set: $0.1 \leq x_1 \leq 0.75$, $0.1 \leq x_2$, $x_3 \leq 0$, $0.1 \leq x_5$, $x_2 + x_5 \leq 1.5$.

Table G-5: Violated constraints at x^0 for tool path A.

x^0	Violated constraints
(0.4, 0.6, -0.6, -0.8, 0.7)^T	$\min[l_2] \geq \underline{l}_2$ $\min[l_3] \geq \underline{l}_3$
(0.7, 0.4, -1.0, -1.1, 0.2)^T	$\max[l_1] \leq \bar{l}_1$ $\max[l_3] \leq \bar{l}_3$
(0.5, 0.8, -0.6, -0.6, 0.5)^T	$\min[l_1] \geq \underline{l}_1$ $\min[l_2] \geq \underline{l}_2$

Table G-6: Tool path A, five design variables.

$F(x)$ used	x^0	$F(x^0)$	x^*	$F(x^*)$	Active constraints	No. of steps
$F_{A.F.D}$	(0.4, 0.6, -0.6, -0.8, 0.7)^T	10.107	(0.750, 0.833, -0.961, -1.148, 0.101)^T	1.525	$\max[l_1] \leq \bar{l}_1$ $\max[l_3] \leq \bar{l}_3$ $x_1 \leq 0.75$	727
	(0.7, 0.4, -1.0, -1.1, 0.2)^T	3.735	(0.750, 0.832, -0.961, -1.147, 0.100)^T	1.524	$\max[l_1] \leq \bar{l}_1$ $\max[l_3] \leq \bar{l}_3$ $x_1 \leq 0.75$ $x_5 \geq 0.1$	751
	(0.5, 0.8, -0.6, -0.6, 0.5)^T	1737.8	(0.750, 0.808, -0.966, -1.141, 0.126)^T	1.552	$\max[l_1] \leq \bar{l}_1$ $\max[l_3] \leq \bar{l}_3$ $x_1 \leq 0.75$	442
$F_{A.F.S}$	(0.4, 0.6, -0.6, -0.8, 0.7)^T	10.104	(0.750, 0.745, -0.952, -1.086, 0.115)^T	1.553	$\max[l_3] \leq \bar{l}_3$ $x_1 \leq 0.75$	4024#
	(0.7, 0.4, -1.0, -1.1, 0.2)^T	3.737	(0.750, 0.725, -0.944, -1.064, 0.101)^T	1.544	$\max[l_3] \leq \bar{l}_3$ $x_1 \leq 0.75$	4048#
	(0.5, 0.8, -0.6, -0.6, 0.5)^T	121.38	(0.750, 0.730, -0.950, -1.075, 0.116)^T	1.558	$\max[l_3] \leq \bar{l}_3$ $x_1 \leq 0.75$	4048#
$F_{Q.I}$	(0.4, 0.6, -0.6, -0.8, 0.7)^T	-0.120	(0.675, 1.165, -0.913, -1.041, 0.100)^T	-0.740	$x_5 \geq 0.1$	4026#
	(0.7, 0.4, -1.0, -1.1, 0.2)^T	-0.086	(0.641, 1.140, -0.902, -1.027, 0.100)^T	-0.736	$\min[l_3] \geq \underline{l}_3$ $x_5 \geq 0.1$	4016#
	(0.5, 0.8, -0.6, -0.6, 0.5)^T	-0.012	(0.571, 1.088, -0.884, -0.996, 0.100)^T	-0.724	$\min[l_3] \geq \underline{l}_3$ $x_5 \geq 0.1$	4057#

Note: # Optimization algorithm terminated on maximum number of steps allowed.

Table G-7: Violated constraints at x^0 for tool path B.

x^0	Violated constraints
(0.4, 0.6, -0.6, -0.8, 0.7)^T	$\min[l_2] \geq \underline{l}_2$ $\min[l_3] \geq \underline{l}_3$
(0.7, 0.4, -1.0, -1.1, 0.2)^T	$\max[l_3] \leq \bar{l}_3$
(0.5, 0.8, -0.6, -0.6, 0.5)^T	$\min[l_1] \geq \underline{l}_1$ $\min[l_2] \geq \underline{l}_2$ $\min[l_3] \geq \underline{l}_3$

Table G-8: Tool path B, five design variables.

$F(x)$ used	x^0	$F(x^0)$	x^*	$F(x^*)$	Active constraints	No. of steps
$F_{A.F.D}$	(0.4, 0.6, -0.6, -0.8, 0.7)^T	4.332	(0.750, 0.737, -1.021, -1.135, 0.337)^T	1.812	$\max[l_1] \leq \bar{l}_1$ $\max[l_3] \leq \bar{l}_3$ $x_1 \leq 0.75$	711
	(0.7, 0.4, -1.0, -1.1, 0.2)^T	4.191	(0.750, 0.737, -1.021, -1.135, 0.337)^T	1.812	$\max[l_1] \leq \bar{l}_1$ $\max[l_3] \leq \bar{l}_3$ $x_1 \leq 0.75$	688
	(0.5, 0.8, -0.6, -0.6, 0.5)^T	5.317	(0.750, 0.736, -1.021, -1.134, 0.336)^T	1.812	$\max[l_1] \leq \bar{l}_1$ $\max[l_3] \leq \bar{l}_3$ $x_1 \leq 0.75$	678
$F_{A.F.S}$	(0.4, 0.6, -0.6, -0.8, 0.7)^T	4.327	(0.750, 0.777, -0.959, -1.088, 0.138)^T	1.652	$\max[l_3] \leq \bar{l}_3$ $x_1 \leq 0.75$	4064#
	(0.7, 0.4, -1.0, -1.1, 0.2)^T	4.193	(0.750, 0.645, -0.971, -1.019, 0.221)^T	1.746	$\max[l_1] \leq \bar{l}_1$ $x_1 \leq 0.75$	4075#
	(0.5, 0.8, -0.6, -0.6, 0.5)^T	5.310	(0.750, 0.911, -0.972, -1.191, 0.129)^T	1.643	$\max[l_1] \leq \bar{l}_1$ $\max[l_3] \leq \bar{l}_3$ $x_1 \leq 0.75$	4079#
$F_{Q.I}$	(0.4, 0.6, -0.6, -0.8, 0.7)^T	-0.321	(0.674, 1.111, -0.884, -0.896, 0.149)^T	-0.800	$\min[l_3] \geq \underline{l}_3$	2001#
	(0.7, 0.4, -1.0, -1.1, 0.2)^T	-0.071	(0.592, 1.258, -0.913, -1.007, 0.100)^T	-0.814	$x_5 \geq 0.1$	4024#
	(0.5, 0.8, -0.6, -0.6, 0.5)^T	-0.209	(0.586, 1.123, -0.866, -0.888, 0.100)^T	-0.803	$\min[l_3] \geq \underline{l}_3$ $x_5 \geq 0.1$	4080#

Note: # Optimization algorithm terminated on maximum number of steps allowed.

G.2 *Dynamic-Q*

The results obtained using the *Dynamic-Q* approximation method for the $F_{A.F.D}$ cost function are given here. In the first sub-section, five design variables are used, and the last sub-section gives the results for six design variables. A move limit of Δ=0.2 is imposed on all design variables. The convergence criteria prescribed throughout on *Dynamic-Q* are:

Stop if $\left\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\right\| < 10^{-2}$.

The parameter settings for the *LFOPC* algorithm used by *Dynamic-Q* are throughout ε_x=10^{-8}, ε_g=10^{-5} with the maximum number of allowable steps per phase, 1000. For five design variables *DELT*=0.45 and for six design variables *DELT*=0.49 throughout.

G.2.1 Five design variables

The design constraints are the same as used for *LFOPC*. The constraints on the variables are set to: $0.1 \le x_1 \le 0.75$, $0.1 \le x_2$, $x_3 \le 0$, $0.1 \le x_5$, $x_2 + x_5 \le 1.5$.The same initial designs were used as for *LFOPC*.

Table G-9: Violated constraints at x^0 for tool path A.

x^0	Violated constraints
(0.4, 0.6, -0.6, -0.8, 0.7)^T	$\min[l_2] \geq \underline{l}_2$ $\min[l_3] \geq \underline{l}_3$
(0.7, 0.4, -1.0, -1.1, 0.2)^T	$\max[l_1] \leq \bar{l}_1$ $\max[l_3] \leq \bar{l}_3$
(0.5, 0.8, -0.6, -0.6, 0.5)^T	$\min[l_1] \geq \underline{l}_1$ $\min[l_2] \geq \underline{l}_2$

Table G-10: Tool path A, five design variables.

x^0	$F(x^0)$	x^*	$F_{A.F.D}(x^*)$	Active constraints	No. of steps
(0.4, 0.6, -0.6, -0.8, 0.7)^T	10.107	(0.737, 0.737, -0.916, -1.049, 0.100)^T	1.564	$\min[l_2] \geq \underline{l}_2$ $x_5 \geq 0.1$	31
(0.7, 0.4, -1.0, -1.1, 0.2)^T	3.735	(0.750, 0.822, -0.924, -1.119, 0.100)^T	1.526	$x_1 \leq 0.75$ $x_5 \geq 0.1$	31
(0.5, 0.8, -0.6, -0.6, 0.5)^T	1737.8	(0.695, 0.796, -0.908, -1.079, 0.100)^T	1.590	$\min[l_2] \geq \underline{l}_2$ $x_5 \geq 0.1$	22

Table G-11: Violated constraints at x^0 for tool path B.

x^0	Violated constraints
$(0.4, 0.6, -0.6, -0.8, 0.7)^T$	$\min[l_2] \geq \underline{l}_2$ $\min[l_3] \geq \underline{l}_3$
$(0.7, 0.4, -1.0, -1.1, 0.2)^T$	$\max[l_3] \leq \bar{l}_3$
$(0.5, 0.8, -0.6, -0.6, 0.5)^T$	$\min[l_1] \geq \underline{l}_1$ $\min[l_2] \geq \underline{l}_2$ $\min[l_3] \geq \underline{l}_3$

Table G-12: Tool path B, five design variables.

x^0	$F(x^0)$	x^*	$F_{A.F.D}(x^*)$	Active constraints	No. of steps
$(0.4, 0.6, -0.6, -0.8, 0.7)^T$	4.332	$(0.675, 0.783, -0.912, -1.041, 0.100)^T$	1.721	$\min[l_2] \geq \underline{l}_2$ $x_5 \geq 0.1$	62
$(0.7, 0.4, -1.0, -1.1, 0.2)^T$	4.191	$(0.662, 0.787, -0.909, -1.038, 0.100)^T$	1.741	$\min[l_2] \geq \underline{l}_2$ $x_5 \geq 0.1$	31
$(0.5, 0.8, -0.6, -0.6, 0.5)^T$	5.317	$(0.750, 0.820, -0.934, -1.107, 0.126)^T$	1.656	$x_1 \leq 0.75$	59

G.2.2 Six design variables

The constraints on the design variables are the same as for five design: $0.1 \le x_1 \le 0.75$, $0.1 \le x_2$, $x_3 \le 0$, $0.1 \le x_5$, $x_2 + x_5 \le 1.5$, but with the additional constraint on the sixth design variable: $0 \le x_6 \le x_1$.

Table G-13: Violated constraints at x^0 for tool path A.

x^0	Violated constraints
$(0.4, 0.6, -0.6, -0.8, 0.7, 0.0)^T$	$\min[l_2] \ge \underline{l}_2$ $\min[l_3] \ge \underline{l}_3$
$(0.7, 0.4, -1.0, -1.1, 0.2, 0.7)^T$	$\max[l_1] \le \bar{l}_1$ $\max[l_2] \le \bar{l}_2$ $\max[l_3] \le \bar{l}_3$
$(0.5, 0.8, -0.6, -0.6, 0.5, 0.2)^T$	$\min[l_1] \ge \underline{l}_1$ $\min[l_2] \ge \underline{l}_2$

Table G-14: Tool path A, six design variables.

x^0	$F(x^0)$	x^*	$F_{A.F.D}(x^*)$	Active constraints	No. of steps
$(0.4, 0.6, -0.6, -0.8, 0.7, 0.0)^T$	10.107	$(0.736, 0.653, -0.884, -0.954, 0.146, 0.049)^T$	1.768	$\min[l_2] \ge \underline{l}_2$	18
$(0.7, 0.4, -1.0, -1.1, 0.2, 0.7)^T$	25.870	$(0.671, 0.342, -1.055, -0.438, 0.905, 0.661)^T$	2.043	none	20
$(0.5, 0.8, -0.6, -0.6, 0.5, 0.2)^T$	173.82	$(0.663, 0.517, -1.027, -0.439, 0.850, 0.655)^T$	2.096	none	11

Table G-15: Violated constraints at x^0 for tool path B.

x^0	Violated constraints
(0.4, 0.6, -0.6, -0.8, 0.7, 0.0)^T	$\min[l_2] \geq \underline{l}_2$ $\min[l_3] \geq \underline{l}_3$
(0.7, 0.4, -1.0, -1.1, 0.2, 0.7)^T	$\max[l_2] \leq \bar{l}_2$ $\max[l_3] \leq \bar{l}_3$
(0.5, 0.8, -0.6, -0.6, 0.5, 0.2)^T	$\min[l_1] \geq \underline{l}_1$ $\min[l_2] \geq \underline{l}_2$ $\min[l_3] \geq \underline{l}_3$

Table G-16: Tool path B, six design variables.

x^0	$F(x^0)$	x^*	$F_{A.F.D}(x^*)$	Active constraints	No. of steps
(0.4, 0.6, -0.6, -0.8, 0.7, 0.0)^T	4.332	(0.750, 0.831, -0.930, -1.131, 0.203, 0.001)^T	1.721	$\min[l_2] \geq \underline{l}_2$ $x_1 \leq 0.75$ $x_6 \geq 0$	58
(0.7, 0.4, -1.0, -1.1, 0.2, 0.7)^T	66.323	(0.744, 0.765, -0.911, -1.065, 0.285, 0.022)^T	1.833	$\min[l_2] \geq \underline{l}_2$	33
(0.5, 0.8, -0.6, -0.6, 0.5, 0.2)^T	223.44	(0.750, 0.873, -0.936, -1.126, -0.100, 0.003)^T	1.665	$\min[l_2] \geq \underline{l}_2$ $x_1 \leq 0.75$ $\max[l_3] \leq \bar{l}_3$	100[#]

Note: [#] The optimization algorithm was manually terminated after 100 iterations. The design with the lowest cost function out of the 100 iterations was chosen as the optimum design. This design was reached after 32 iterations and is given in the table.

References

[1] Abassi W.A., Ridgeway S.C., Adsit P.D., Crane C.D. and Duffy J., "Development of a spatial 6-6 parallel platform for contour milling", *Proceedings of the ASME Manufacturing Engineering Division 1997 International M.E. Congress and Exposition (IMECE)*, Dallas, 1997, p.373-380.

[2] Berner D.F., Snyman J.A., "The influence of joint angle constraints on the optimum design of a planar robotic manipulator following a complicated path", *Computers Math Applic*, 1999, vol.37, p.111-124.

[3] Cauchy A., "Deuxième mémoire sur les polygones et les polyèdres", *Journal de l'Ecole polytechnique*, May 1813, p.87-98.

[4] Craig K.J., de Kock D.J., Snyman J.A., "Using CFD and mathematical optimization to minimize stack pollution", *International Journal for Numerical Methods in Engineering*, 1999, vol.44, p.551-566.

[5] Craig K.J., Venter P., de Kock D.J., Snyman J.A., "Optimization of structured grid spacing parameters for separated flow simulation using mathematical optimization", *Journal of Wind Engineering and Industrial Aerodynamics*, 1999, vol.80, p.221-231.

[6] Datoussaïd S., Verlinden O., Dehombreux P. and Conti C., 1996, "Optimal design of multibody systems", *Proceedings of the 1996 3rd international conference on dynamics and control of structures in space*, 1996, p.507-522.

[7] de Kock D.J., Craig K.J., Snyman J.A., "Using mathematical optimization in CFD analysis of a continuous quenching process", Accepted for publication in the *International Journal for Numerical Methods in Engineering*, 1999.

[8] du Plessis L.J., "An optimization approach to the determination of manipulator workspaces", *Master's thesis in Mechanical Engineering*, University of Pretoria, 1999.

[9] Dubbel H., "Handbook of mechanical engineering", *Springer-Verlag*, London, 1994, p.K43-K47.

[10] Fitzgerald J.M., "Evaluating the Stewart platform for manufacturing", *Robotics today*, 1993, vol.6, no.1, p.1-3.

[11] Gosselin C., " Determination of the workspace of 6-DOF parallel manipulators", *Journal of Mechanisms, Transmissions, and Automation in Design*, 1990, vol.112, no.3, p.331-336.

[12] Gosselin C., Angeles J., "The optimum kinematic design of a planar three-degree-of-freedom parallel manipulator", *Journal of Mechanisms, Transmissions, and Automation in Design*, 1988, vol.110, no.1, p.35-41.

[13] Gough V.E., "Universal tire test machine", *Proceedings, ninth international technical congress F.I.S.I.T.A.*, May 1962, p.117.

[14] Haug E.J., "Computer aided kinematics and dynamics of mechanical systems", *Allen and Bacon*, Boston, 1989.

[15] Haug E.J., Luh C.M., Adkins F.A., Wang J-Y., "Numerical algorithms for mapping boundaries of manipulator workspaces", *ASME Journal of Mechanical Design*, 1996, vol.118, no.2, p.228-234.

[16] Hay A.M, Snyman J.A., "The determination of non-convex workspaces of generally constrained planar Stewart platforms", Accepted for publication in *Computers Math Applic*, 1999.

[17] Ji Z., "Placement analysis for a class of platform manipulators", *Proceedings of the 1995 ASME Design Engineering Technical Conferences*, 1995, DE-vol.82, no.1, p.773-779.

[18] Lee J., Duffy J., Keler M., "The optimum quality index for the stability of in-parallel planar platform devices", *ASME Journal of Mechanical Design*, 1999, vol.121, no.1, p.15-20.

[19] Machinability data center, "Machining data handbook", 2nd edition, *Metcut research associates inc.*, 1972.

[20] Merlet J-P., "Parallel manipulators: state of the art and perspectives", *http://www-sop.inria.fr/saga/personnel/merlet/Etat/etat_de_lart.html*, 1996.

[21] Merlet J-P., "Follow-up on parallel manipulators: a short history", *http://www.robot.ireq.ca/CRR/Archives/1995/0233.html*, 1995.

[22] Merlet J-P., "Workspace-oriented methodology for designing a parallel manipulator", *Proceedings of the 1996 IEEE international conference on robotics and automation*, 1996, p.3726-3731.

[23] Merlet J-P., "Singular configurations of parallel manipulators and Grassmann geometry", *The International Journal of Robotics Research*, 1989, vol.8, p.45-56.

[24] Nikravesh P.E., "Computer-aided analysis of mechanical systems", *Prentice-Hall International*, Englewood Cliffs, 1988.

[25] Oberg E., "Machinery's handbook", Twenty-second edition, *Industrial Press Inc.*, 1989.

[26] Snyman J.A, "A new and dynamic method for unconstrained minimization", *Applied Mathematical Modeling*, 1982, vol.6, p.449-462.

[27] Snyman J.A, "An improved version of the original leap-frog dynamic method for unconstrained minimization", *Applied Mathematical Modeling*, 1983, vol.7, p.216-218.

[28] Snyman J.A, "The LFOPC leap-frog method for constrained optimization", To appear in *Computers Math Applic*, 1999.

[29] Snyman J.A., Berner D.F., "A mathematical optimization methodology for the optimal design of a planar robotic manipulator", *International Journal for Numerical Methods in Engineering*, 1999, vol.44, p.535-550.

[30] Snyman J.A., Berner D.F., "The design of a planar robotic manipulator for optimum performance of prescribed tasks", To appear in *Structural Optimization*, 1999.

[31] Snyman J.A., du Plessis L.J., Duffy J., "An optimization approach to the determination of the boundaries of manipulator workspaces", Accepted for publication *The ASME Journal of Mechanical Design*, 1999.

[32] Snyman J.A., Frangos C., Yavin Y., "Penalty function solutions to optimal control problems with general constraints via a dynamic optimization method", *Computers Math Applic*, 1992, vol.23, p.47-56.

[33] Snyman J.A., Heyns P.S., Vermeulen P.J., "Vibration isolation of a mounted engine through optimization", *Mechanism and Machine Theory*, 1995, vol.30, p.109-118.

[34] Snyman J.A., Stander N., "A new successive approximation method for optimum structural optimization problems", *AIAA Journal*, 1994, vol.32, p.1310-1315.

[35] Snyman J.A., Stander N., Roux W.J., "A dynamic penalty function method for the solution of structural optimization problems", *Appl Math Modelling*, 1994, vol.18, p.453-460.

[36] Snyman J.A., van Tonder F., "Optimum design of a three dimensional robotic manipulator", *Structure Optimization*, 1999, vol.17, p.172-185.

[37] Stamper R.E., Tsai L., Walsh G.C., "Optimization of a three DOF translational platform for well-conditioned workspace", *Proceedings of the 1997 IEEE international conference on robotics and automation*, 1997, vol.4, p.3250-3255.

[38] Stewart D., "A platform with six degrees of freedom", *Proceedings of the institution of mechanical engineers*, 1965, p.371-386.

[39] Takano M., Masaki H., Sasaki K., "Concept of total computer-aided design system of robot manipulators", *Robotics Research: Third International Symposium. MIT Press Series in Artificial Intelligence,* 1985, p.289-296.

[40] Van Wyk A.J., Snyman J.A., Heyns P.S., "Optimization of vibratory conveyor for reduced support reaction forces", *R&D Journal of the SAIMechE*, 1994, vol.10, p.12-17.