UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

Optimization of Machining Parameters with Consideration of Tool Adjustment in Turning Operations

A thesis submitted to
the School of Graduate Studies and Research
in partial fulfilment of the requirements for the
degree of Master of Applied Science in
Mechanical Engineering

By

Mussa I. Mgwatu

Ottawa-Carleton Institute for
Mechanical and Aerospace Engineering
University of Ottawa
Ottawa, Ontario, Canada

Abstract

Tolerancing in design stage and machining parameter optimization in manufacturing stage are related problems which have a great impact on overall production economics of a machining operation. Unfortunately, these problems have been viewed separately causing conflicts between design and manufacturing decisions. While a design decision is to attain a certain part tolerance in the manufacturing stage, the manufacturing decision may be more concerned about economical production and production feasibility.

To avoid such conflicting situations, this thesis presents six optimization models which are developed and solved in two stages in order to integrate tolerancing and machining parameters decisions. One model is developed in the first stage. The first-stage model is related to a design problem and provides better part tolerance to be used in the second stage to incorporate tool adjustment decisions into the optimization of machining parameters. Five models are developed in the second stage and are divided into single-machine and multi-machine turning operation problems. For single-machine turning operations, three models are formulated. The first is a single-product multi-pass model considering pass selection for determining optimum cutting speed, feed rate, depth of cut, dimension deviation and number of passes. The second is a single-product multi-pass model considering inventory and setup costs to find optimum cutting speed, feed rate, depth of cut, dimension deviation and batch size. The third is a multi-product multi-pass model considering inventory and setup costs for selection of optimum cutting speed,

feed rate, depth of cut, dimension deviations and production cycle time. For multi-machine turning operations, two models are formulated. One is a multi-feature multi-pass model with the objective of minimizing total cost for determining optimum workload assignment, cutting speed, feed rate, depth of cut, and feature dimension deviations of machine-pass-feature combinations. The second is a multi-feature multi-pass model with the objective of minimizing cycle time which, once solved, will give optimum workload assignment, cutting speed, feed rate, depth of cut, and feature dimension deviations of machine-pass-feature combinations.

A solution method and illustrative examples are given to test the feasibility of the developed optimization models.

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. M. Liang for his guidance, valuable comments and thoughtful suggestions at each stage of my research work.

My Master's degree program was financially supported by the Canadian Commonwealth Scholarship and Fellowship Plan on behalf of the Government of Canada. Special thanks should go to all staff members of the Canadian Commonwealth Scholarship and Fellowship Plan for their cooperation. More thanks are due to the School of Graduate Studies and Research Scholarship Selection Committee for offering me the University of Ottawa Excellence Scholarship.

I would like to acknowledge the members of my defence committee, Dr. B.S. Dhillon and Dr. J.C. Beddoes for taking their time to examine my thesis.

I am very much grateful to my parents Joha and Iddi for their continued support and helpful advice.

Finally, I have to thank my wife Tunu whose patience and encouragement have contributed to the completion of this research work.

Table of Contents

List of Figures

List of Tables

Nomenclature

i	index for machines, $i,\ldots,m$
j	index for cutting passes, $j,\ldots,J$
k	index for products or features, $k,\ldots,K$
J_k	index for last cutting pass of product or feature k
δ_k	part tolerance for product or feature k in the design phase (mm)
v_j	cutting speed for pass j (m/min)
f_j	feed rate for pass j (mm/rev)
d_j	depth of cut for pass j (mm)
y	dimension deviation in the manufacturing phase (mm)
p_j	$= 1$, if the cutting pass is selected; $= 0$, otherwise
Q	production batch size
v_{kj}	cutting speed for pass j of product k (m/min)
f_{kj}	feed rate for pass j of product k (mm/rev)
d_{kj}	depth of cut for pass j of product k (mm)
y_{kJ_k}	dimension deviation of product k after the last pass in manufacturing phase (mm)
T	cycle time for batch production (min)
X_{ijk}	$= 1$, if pass j of feature k is assigned to machine i; $= 0$, otherwise

v_{ijk}	cutting speed for machine-pass-feature combination i, j, k (m/min)
f_{ijk}	feed rate for machine-pass-feature combination i, j, k (mm/rev)
d_{ijk}	depth of cut for machine-pass-feature combination i, j, k (mm)
$y_{iJ_k k}$	dimension deviation of feature k after the last pass on machine i (mm)
Z	cycle time for each machine (min)
$g_{1k}, g_{2k}, g_{3k}, g_{4k}$	constants for production cost-tolerance function of product or feature k
δ_{lk}	minimum allowed tolerance for product or feature k in design stage (mm)
δ_{uk}	maximum allowed tolerance for product or feature k in design stage (mm)
A	scrap or rework cost ($)
D_0	initial stock diameter of workpiece (mm)
D_{j-1}	diameter of the workpiece before the (j-1)th pass (mm)
D_{k0}	initial stock diameter of product k in a single-machine turning operation (mm)
$D_{k(j-1)}$	diameter of product k before the (j-1)th pass in a single-machine turning operation (mm)
D_{0k}	initial stock diameter of feature k in a multi-machine turning operation (mm)

$D_{(j-1)\,k}$	diameter of feature k before the (j-1)th pass in a multi-machine turning operation (mm)
L	length of the workpiece (mm)
L_k	length of product/feature k (mm)
C_o	operating cost rate ($/min)
α_T, β_T, γ_T, K_T	constants in tool life equation
C_t	cost of tool change per edge ($/edge)
C_a	tool adjustment cost rate ($/min)
t_r	tool replacement time (min)
t_a	tool adjustment time (min)
w	allowed nose wear of a cutting tool (mm)
δ	maximum allowed tolerance in manufacturing stage (mm), which is obtained in design stage.
δ_{kJ_k}	maximum allowed tolerance in manufacturing stage for the last pass of product k (mm)
δ_{iJ_kk}	maximum allowed tolerance in manufacturing stage for the last pass of feature k on machine i (mm)
$v_j^{(l)}$, $v_j^{(u)}$	minimum, maximum cutting speed for pass j (m/min)
$v_{kj}^{(l)}$, $v_{kj}^{(u)}$	minimum, maximum cutting speed for pass j of product k (m/min)
v_{kJ_k}	cutting speed for the last pass of product k (m/min)
$v_{ijk}^{(l)}$, $v_{ijk}^{(u)}$	minimum, maximum cutting speed for pass j of feature k on machine i (m/min)

$v_{iJ_k k}$	cutting speed for the last pass of feature k on machine i (m/min)
$f_j^{(l)}, f_j^{(u)}$	minimum, maximum feed rate for pass j (mm/rev)
$f_{kj}^{(l)}, f_{kj}^{(u)}$	minimum, maximum feed rate for pass j of product k (mm/rev)
f_{kJ_k}	feed rate for last pass of product k (mm/rev)
$f_{ijk}^{(l)}, f_{ijk}^{(u)}$	minimum, maximum feed rate for pass j of feature k on machine i (mm/rev)
$f_{iJ_k k}$	feed rate for the last pass of feature k on machine i (mm/rev)
d_{total}	total amount of depth of cut to be removed (mm)
d_{ktotal}	total amount of depth of cut to be removed for product or feature k (mm)
$d_j^{(u)}$	maximum depth of cut for pass j (mm)
$d_J^{(l)}, d_J^{(u)}$	minimum, maximum depth of cut for the last pass (mm)
$d_{kj}^{(l)}, d_{kj}^{(u)}$	minimum, maximum depth of cut for pass j of product k (mm)
d_{kJ_K}	depth of cut for last pass of product k (mm)
$d_{ijk}^{(u)}$	maximum depth of cut for pass j of feature k on machine i (mm)
$d_{iJ_k k}$	depth of cut for the last pass of feature k on machine i (mm)
$d_{iJ_k k}^{(l)}, d_{iJ_k k}^{(u)}$	minimum, maximum depth of cut for the last pass of feature k on machine i (mm)
β_F, γ_F, K_F	cutting force constants for tool-workpiece combination
$F_c^{(u)}$	maximum cutting force (Kg)
K_P	cutting power constant
η_m	efficiency of a machine tool

$P_c^{(u)}$	maximum cutting power of the motor (KW)
α_S, β_S, γ_S, K_S	surface roughness constants
$S_F^{(u)}$	maximum allowed surface roughness (μm)
$S_{Fk}^{(u)}$	maximum allowed surface roughness for product or feature k (μm)
t_m	machining time (min)
T_L	tool life (min)
λ	demand rate (parts/min)
λ_k	demand rate for product k (parts/min)
P	production rate (parts/min)
P_k	production rate for product k (parts/min)
P_r	minimum required production rate (parts/min)
P_{rk}	minimum required production rate for product k (parts/min)
i_h	inventory holding cost rate
C_s	setup cost per production run ($/setup)
C_{sk}	setup cost per production run for product k ($/setup)

Chapter 1

Introduction

1.1 Overview

Machining is a manufacturing process that refers to the removal of material from a workpiece by means of relative motion between a cutting tool and the workpiece. The material from the workpiece is removed in the form of chips. The main purpose of the machining operation is to change the shape of raw material to a finished component. Five basic machining processes are turning for producing external rotational parts, drilling for producing holes, and milling, planing and shaping for processing prismatic parts.

In traditional machining, operations such as part loading and tool setting were performed manually. The machine flexibility and efficiency were very low. Today, the application of numerical control (NC) technology allows the machine tools to perform operations automatically. The basic components of NC machines are the part program to provide instructions for machining steps, the machine control unit (MCU) to read and interpret the instructions in the part program and control the motions of the machine, and the machine tool to perform the actual machining operation.

Due to the NC technology, the machining conditions, governed by machining parameters such as machining speed, feed rate, and depth of cut can be easily controlled. As a result, both productivity and product quality can be improved with lower cost. However, the success or failure in achieving these goals greatly depends on how a process plan is prepared.

Process planning includes evaluation of product design and selection of work material, machining processes, and machining parameters. Design evaluation identifies geometric features, part dimensions, surface roughness, and tolerance specifications. The choice of work materials should consider the cost, applicability, machinability, and mechanical properties of the materials. Machining processes are selected based on the shape of the component to be produced.

The selection of part tolerance and machining parameters in process planning has a great impact on overall production economics of a machining operation. For example, to produce a part with a very tight tolerance is expensive due to high machining cost. Similarly, the manufacturing cost for a part with a very loose tolerance is high because of the scrap or rework cost. On the other hand, at very high cutting speeds, the total production cost increases due to the increased tool replacement cost caused by frequent tool changes. Also, at very low cutting speeds, the total production cost is high because more machining time is required to produce a part. Therefore, the part tolerance and machining parameters should be optimized in a process plan.

1.2 Objectives

Tolerancing in the design stage has been discussed by many researchers. The optimization of machining parameters in manufacturing stage for both single-machine and multi-machine has also been investigated in the literature. However, tolerancing and machining parameter optimization are mostly addressed separately causing conflicts between design and manufacturing decisions. In the manufacturing stage, tool adjustment analysis is often performed in isolation from the machining parameter optimization process.

The main objectives of this research are therefore:

(1) to integrate tolerancing and machining parameter selection decisions;

(2) to optimize machining parameters and tool adjustment decisions for the single-machine turning process in the following cases.

   (a) single-product, multi-pass considering pass selection,

   (b) single-product, multi-pass considering inventory and setup costs,

   (c) multi-product, multi-pass considering inventory and setup costs; and

(3) to optimize machining parameters and tool adjustment decisions for multi-machine, multi-feature, and multi-pass operation considering different decision objectives.

1.3 Thesis Organization

This thesis is divided into five chapters. Following this introductory chapter, the literature survey on machining parameter optimization, tolerancing, tool life and tool wear, tool adjustment and quality loss analyses are given in Chapter 2. Models for optimizing tolerance, machining parameters and tool adjustments are developed in Chapter 3. These models were developed in two stages. The first stage model is related to a design problem and will provide the optimum tolerance. The second stage models are related to manufacturing problems and will simultaneously provide optimum machining parameters and tool adjustment decisions for both single-machine and multi-machine turning operations. Chapter 4 provides computational results where illustrative examples are given and solved. Chapter 5 concludes the thesis and gives recommendations for future work.

Chapter 2

Literature Survey

2.1 Machining Parameter Optimization

A rich body of literature on machining parameter optimization has been developed since Taylor's pioneer work in metal cutting. Taylor (1907) studied the relationship between tool life and cutting speed and established an empirical equation which is known as Taylor's tool life equation.

Many mathematical models have been formulated for the selection of optimum cutting speed, feed and depth of cut for different machining processes. Problems of single-pass and multi-pass machining operations on single-machine and multi-machine have also been investigated. In the earlier studies, the cutting speed was the only machining parameter to be optimized. The objective criterion was minimum-cost, minimum-time, or maximum-profit cutting speed. Later on, machining parameters such as feed rate and depth of cut were also included in the optimization process. Brown (1962) analyzed the effect of cutting speed and feed for single and double-pass cutting operations using single tool and multiple tools. Walverkar and Lambert (1970) and Petropoulos (1973) performed a more detailed study with consideration of physical

machining restrictions such as the available machine power and surface roughness of the workpiece. They determined the cutting speed and feed with a minimum-production-cost criterion for a single pass turning process using geometric programming without including the depth of cut. Lambert and Walverkar (1978) extended the work of Walverkar and Lambert (1970) and included the depth of cut in a multi-pass machining problem. Recently, Yeo *et al* (1995) reported the determination of cutting speed, feed rate and depth of cut for multi-pass turning operations such as longitudinal turning, facing, taper turning and contour turning to produce a stepped shaft. The aforementioned optimization models dealt with single-machine manufacturing problems. Other researchers such as Agapiou (1992) and Hitomi and Ham (1977) extended single-machine problems to multi-machine flow type manufacturing problems.

In practice, a single pass operation may not necessarily be the most economic choice. Ermer and Kromodihardjo (1981) illustrated that, in some cases, a multi-pass operation may be more economical than a single pass. The results from their work showed that double-pass or triple-pass cutting operations may be more preferred than a single-pass cutting operation when the problem is treated with practical machining constraints such as cutting force and cutting power. Recently, Tan and Creese (1995) discussed a general multi-pass machining model for machining parameter selection in turning. The optimum cutting speed, feed rate, depth of cut, and number of passes are simultaneously obtained in a single stage. Again, they observed that multi-pass turning could be more economic than the single-pass turning.

Machining parameter optimization models can also be developed based on experimental results. Chua *et al* (1993) reported several such models developed by regression analysis of their experimental data.

Multi-criteria approach has also been employed to seek a trade-off between several objectives. Ghiassi *et al* (1984) presented two methods for solving a multiple criteria decision making problem, where surface integrity, tool life, and metal removal rate are maximized while the surface roughness is minimized. The first method provided a set of non-dominated solution graphically, and the second method gave a specific solution from a non-dominated set. Malakooti and Deviprasad (1989) explained and applied an interactive pair-comparison approach to solve a discrete multiple objective problem. They claimed that, their approach can be applied to any multi-criteria problem with either discrete, integer or continuous set of alternatives. Malakooti (1991) developed a dynamic approach to the multiple objective problem which could be used for on-line selection of machining parameters and monitoring of operations. Recently, Mesquita *et al* (1995) described a model and an interactive program system for computer-aided optimization of cutting speed, feed and depth of cut for rough, multi-pass turning operations. Commonly used criteria were minimum unit production cost, unit production time and number of passes.

Most researchers assume that the cutting conditions are deterministic and the characteristics of tool and workpiece remain constant during machining. It has been observed that the cutting conditions are affected by the uncertainty of tool life. Iwata *et al* (1972) showed that the coefficients used in tool life, cutting force, and cutting power

equations are probabilistic in nature. Hati and Rao (1976) formulated deterministic and probabilistic models and confirmed that the indices involved in the extended Taylor's tool life equation are not deterministic for a wide range of cutting conditions and that some of the constraint parameters are also probabilistic. Fenton and Joseph (1979) presented an objective function consisting of a combination of unit production cost, production rate and profit rate for both deterministic tool life and statistical tool life distributions. The statistical tool life distributions were normal distribution, uniform distribution and Weibull distribution. They further challenged that the tool life equation for any set of machining and tool parameters should follow a certain probabilistic density function, and because the objective function depends on the tool life, it should also be a statistical quantity. However, they admitted that it is very difficult to know the true statistical distribution of the tool life. The work by Sekhon (1983) presented a method for simulation of a probabilistic manufacturing system in which the workpiece and tool variability were considered to minimize cost and maximize production rate. The same conclusion was drawn that in most cases both the cutting tool and the workpiece possess characteristics subjected to random variation.

Expert systems in the field of Artificial Intelligence (AI) have emerged to be one of the major topics of discussion in manufacturing for design, planning, scheduling and control. Wang and Wysk (1986) introduced a computerized expert system to generate machining parameters using empirical equations and modified data retrieval approach. The optimum machining data were obtained as soon as a combination of material, tool and operation is entered in the system. Zhou and Wysk (1992) used an expert system for

tool selection, evaluation of machining parameters, and tool replacement.

Fuzzy logic and neural networks have shown a great potential for solving machining process optimization problems. Fuzzy logic which combines fuzzy sets and fuzzy rules has the capability of modelling complex, non-linear programming problems. Trappey *et al* (1988) developed a fuzzy non-linear programming model with a fuzzy goal and fuzzy constraints, and used Kuhn-Tucker conditions to determine the optimal solution of the fuzzy machining economic model. Wang (1992) applied multi-layer backpropagation neural networks for multi-objective optimization of cutting parameters. He further suggested that the extension of his approach could be used for on-line monitoring and adjustment of cutting parameters based on information from sensors.

The cutting force and cutting power can change during a machining operation due to variations in tool property, hardness of material, width and depth of cut or the air gap of the part. Adaptive control (AC) has been proposed to compensate for such variations by adjusting cutting speed and feed during machining operation. AC can be classified into two categories: adaptive control constraint (ACC) and adaptive control optimization (ACO). With ACC, the feed rate, in most studies, is dynamically adjusted to maintain a specified target cutting power or force level. With ACO, the cutting objective, i.e., the ratio of material removal rate and tool wear, is expected to be optimized dynamically. However, due to the difficulty of on-line measurement of tool wear, the application of ACO seems to be unlikely in the near future (Kalpakjian 1995). Therefore, most AC studies focus on ACC (Kalpakjian 1995). As implied by its definition, ACC can only passively, at most, maintain a reference or target power or force level so as to maintain

a feasible or reliable machining process. The machining parameters cannot be optimized by ACC. Furthermore, the target power or force level will be very conservative if the optimization problem is not addressed beforehand. As revealed above, the machining parameter selection and tool adjustment decision in manufacturing stage, and tolerancing and machining parameter decisions between design and manufacturing stages have not yet been integrated, thereby causing possible conflicts in shop floor operations. Moreover, the emerging ACC also demands a higher but yet safer power or force target to follow. This again depends on how the machining parameters can be optimized.

2.2 Tool Life and Tool Wear

Tool life depends on machining parameters. At a higher cutting speed, the tool life is shorter because of high friction between the tool and machined surface. In order to form a basis for optimizing machining conditions, it is necessary to understand the nature of tool wear. For gradual or progressive tool wear, certain regions of tool face and flank are worn. Wear on the tool face is characterized by the formation of a crater, a result from the action of chips flowing along the face. Wear on the flank of the tool is caused by friction between newly machined surface and the contact area on the tool flank. Flank wear has been widely used as a measure of tool life.

Major tool wear mechanisms for crater and flank wear are described by Bhattacharyya and Ham (1969), Kannatey-Asibu (1985), and Zhou and Wysk (1992) as adhesion, abrasion, diffusion, fatigue and plastic deformation. The dominant wear mechanisms and the rate of wear are functions of temperature, stress, tool material,

workpiece material, and other factors, and depend on a given operation and a set of machining conditions. According to Jeang and Yang (1992), and Wang and Zuo (1994), tool wear is a non-decreasing, non-linear process which consists of three distinct stages. In the initial wear stage, the wear rate is relatively high, but only for a short period of time. The second stage is a steady or normal wear period which constitutes a major portion of tool wear. During this period, the rate of tool wear is approximately constant over time. The last stage is the accelerated period, a rapid and exponential tool wear resulting in a tool failure.

Mukherjee and Basu (1967) designed experiments and applied multiple regression analysis to evaluate tool wear and observed the relationship between the flank wear of a single-point turning tool and machining parameters. Their observation concluded that tool wear is more sensitive to cutting speed than feed and depth of cut. Bhattacharyya and Ham (1969) proposed an analytical approach to predict the amount and rate of wear for a tool-work combination when the interaction coefficient and cutting force are known.

Kuljanić (1975) surveyed direct and indirect methods of in-process measurement of flank wear. In direct measurements, devices such as optical sensors, electrical sensors, radioactive sensors, and pneumatic sensors are used to monitor the volumetric loss from the tool. In indirect measurements, equations are established to correlate tool wear and cutting force or torque, vibration, surface roughness of machined part, temperature and thermoelectric effect. El Gomayel and Bregger (1986) adopted the electromagnetic sensor method to measure the tool wear and showed that the increase in workpiece diameter during the turning operation could be used as a reliable measure of tool wear.

A paper by DeVor *et al* (1977) presented a study of tool life and its variation for a finish turning operation using the weighted least-square method. It was noted that the tool wear variation increases significantly as wear level increases and that at higher level of flank wear, the variance of tool life is not homogeneous under different cutting conditions. An energy method was suggested by Usui *et al* (1984) to predict chip formation and cutting forces in turning operations and to calculate the stress and temperature on the wear faces. Wear characteristic equation, calculated stress, and temperature were then used to predict flank and crater wear. Carlsson and Strand (1992) developed a statistical model based on different probability distributions to predict the tool life. Rao (1986) conducted experimental studies for the purpose of wear monitoring during stable turning and suggested to use the wear index as the basis for the tool monitoring. Zhou and Wysk (1992) integrated such tool wear index into a tooling cost recovery model and a probabilistic unit cost model to record tool usage and to determine optimum tool replacement intervals. Other studies for tool replacement (e.g., La Commare *et al* 1983, Galante and Lombardo 1991, and Koulamas 1991) were also carried out based on flank wear.

As shown in the above survey, previous research has been mostly focused on crater and flank wear. Flank wear has been used as the major tool life indicator and used in machining parameter optimization and tool replacement decisions. Another important and yet often overlooked tool wear is tool nose wear, i.e., the wear occurred at the tool tip or nose. Though nose wear may not be used as tool life measure, it does represent the major contributor to the deviation of part dimension. Therefore, both flank wear and

nose wear should be used in optimizing machining operations with flank wear used for tool life measure and nose wear used for tool adjustment. Surprisingly, however, very little work has been done on nose wear. The only available reference is the work by Du *et al* (1993) in the accessible literature. Nevertheless, enough details are not available in their report.

2.3 Tolerancing

It is impossible to produce every finished part which conforms completely to its nominal dimensional specifications. The allowed deviation from a nominal dimensional specification is called tolerance. The purpose of dimensioning and tolerancing is to determine the dimensions and tolerances that are to be produced in the manufacture of a component.

Several tolerancing methods have been proposed. Evans (1974) suggested a statistical dimensioning method and a tolerance simulation method. The statistical dimensioning method uses the knowledge of process distributions to select a tolerance within a predetermined range. The tolerance simulation method treats dimensions and tolerances as random variables and the simulation of the dimensional design of an assembly becomes necessary. Other methods which are used to specify tolerances include restrictive tolerancing method and table-based standard tolerancing method (Lingaiah 1994).

The above mentioned tolerancing methods have been traditionally applied without explicitly considering machining cost and may not provide economical tolerances. To this

end, Speckhart (1972) introduced an exponential cost-tolerance model to explicitly address the relationship between tolerance and machining cost. For the same purpose, Spotts (1973) reported a reciprocal squared model. Along this line, Sutherland and Roth (1975) proposed a reciprocal power model; Chase and Greenwood (1988) further expressed a reciprocal model as the objective function, and worst case and statistical allocation as constraints to assign component tolerance at minimum cost. Ostwald and Huang (1977) developed a discrete-variable linear programming model with zero-one variables to optimize the cost of tolerance. Cheikh and McGoldrick (1988) presented a unified engineering approach to the problem of tolerance determination and tolerance build-up where the product function, process performance and manufacturing cost are all involved. They recognized that the integration of design procedures and manufacturing process is important for higher productivity. More generalized exponential cost-tolerance model was recently introduced by Dong and Soom (1990) for selecting tolerances using the information provided by a CAD system. Zhang and Wang (1993) employed a generalized exponential cost-tolerance model and solved the problem using simulated annealing technique to allocate design and machining tolerances of a shaft and hole assembly.

Dong *et al* (1994) also conducted a study and developed new cost-tolerance models. The new models were classified as hybrid models and polynomial models which present better fits to the empirical cost-tolerance data than previous models. In addition to the manufacturing cost, the recent study by Feng and Kusiak (1995) further considered the cost by quality loss based on Taguchi quality loss function. This approach certainly

better fits today's manufacturing environment as low manufacturing cost is often achieved at the expense of looser tolerances and thus will be penalized due to poor quality.

Tolerance charting is a graphical method whereby the sequence of machining operations, dimensions and tolerances of the workpiece can be evaluated and then compared with the blueprint requirements. Ngoi and Chua (1993) presented a matrix approach of deriving dimensions and tolerances during tolerance charting. Ngoi and Tan (1995) recently proposed a geometric tolerancing method and applied tolerance control charts to solve working dimensions and tolerances using mathematical models.

2.4 Tool Adjustment and Quality Loss

Tool adjustment is aimed at maintaining part dimension within an acceptable dimension deviation range in order to improve part quality and avoid reworking or rejecting a part. Research on this topic is very limited. A few reported studies are briefly reviewed as follows. Quesenberry (1988) studied the adjustment of the process to minimize the expected mean square of deviations of part measurements from nominal value to produce parts of better quality. A two-part compensator was proposed, the first for adjusting the process due to estimated mean of tool wear and the second for centering the process and protecting the process shift due to other causes. Jeang and Yang (1992) proposed an analytical method for optimal initial tool setting and replacement cycle based on tool replacement cost and part quality loss. Narang and Fischer (1993) accounted for tool adjustment based on the process variability of the machine, specified tolerance and tool wear rate. Wang and Zuo (1994) contributed to in-process tool adjustment by formulating

a mathematical model for the selection of optimal number of adjustments and their corresponding intervals.

2.5 Motivation

The above survey has revealed that effort has not been made to integrate tolerancing decisions in design stage and machining parameter decisions in manufacturing stage. Another observation is that the tool adjustment decision has not been incorporated into the machining parameter optimization decisions. As a result, the decisions made in such a way often conflict with each other and are rarely implemented on shop floor.

For these reasons, it is proposed that (a) tolerancing and machining parameter decisions be integrated; and (b) tool adjustment and machining parameter selection problems be simultaneously solved for both single-machine and multi-machine turning processes.

Chapter 3

Tolerancing and Machining Parameter Optimization

In this chapter, the optimization models for tolerancing, tool adjustment and machining parameter selection are developed in two stages. The first stage model is a design related problem which will provide optimum part tolerance. The optimum part tolerance obtained in the first stage will be used as the upper bound of tolerance in the second stage. The second-stage models are manufacturing related problems which are formulated to determine optimum machining parameters and tool adjustment decisions for both single-machine and multi-machine turning operations. For single-machine turning operations, three models are formulated. These models are single-product multi-pass model considering pass selection, single-product multi-pass model considering inventory and setup costs, and multi-product multi-pass model considering inventory and setup costs. For multi-machine turning operations, two models are formulated. These models are multi-feature multi-pass model with the objective of minimizing total cost and multi-feature multi-pass model with the objective of minimizing cycle time.

3.1 Assumptions

The following assumptions are made in developing the optimization models:

(1) The data for tool life, cutting force, cutting power, and surface roughness are deterministic and are known.

(2) In batch production, the demand is known with a constant continuous rate.

(3) In multi-product batch production, each cycle has exactly one setup for each product, and the parts are produced in the same sequence in each production cycle.

(4) In multi-machine turning processes, all machines are identical and are able to perform any cutting pass of a feature.

(5) In multi-machine turning processes, the time required to move the workpiece from one machine to another machine is small and neglected in the formulation of the models.

3.2 Tolerancing in Design Stage

The aim of the first-stage model is to determine part tolerance at minimum total cost in the design stage. The total cost consists of machining cost and quality loss. The output of this model will be used as the input, an upper limit of tolerance, for the second-stage models. A generalized model by Dong and Soom (1990) is adopted to express the machining cost-tolerance relationship. A quality loss function suggested by Taguchi *et al* (1989) is used to describe the quality loss-tolerance relationship. The details are given below.

Machining cost-tolerance function

For a particular machining process, a generalized exponential machining cost-tolerance function is given by Dong and Soom (1990):

$$g(\delta) = g_1 e^{-g_2(\delta - g_3)} + g_4 \qquad (3.1)$$

where $g(\delta)$ is the cost of producing a mechanical feature with a specified tolerance level δ. g_1, g_2, g_3 and g_4 are constants which control the shape and the position of the cost-tolerance curve and are determined by curve-fitting based on experimental data. The machining cost-tolerance function can be represented graphically as shown in Figure 3.1.

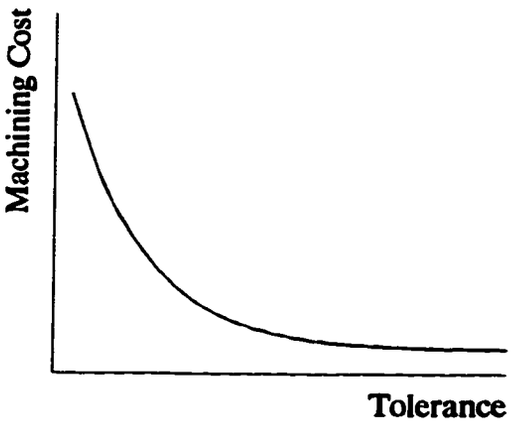

Figure 3.1 Machining cost-tolerance function

Quality loss-tolerance function

A quality loss is defined as the cost incurred when the quality characteristic of the part deviates from its nominal or target value. The amount of quality loss depends on the

amount of deviation. Taguchi *et al* (1989) derived a quality loss function which is expressed as a quadratic function of quality deviation from the nominal or target dimension. The derivation of the quality loss function results in the following expression:

$$L(x) = k(x-m_o)^2 \tag{3.2}$$

where k is a proportionality constant, x quality characteristic of the product, and m_o the quality target value.

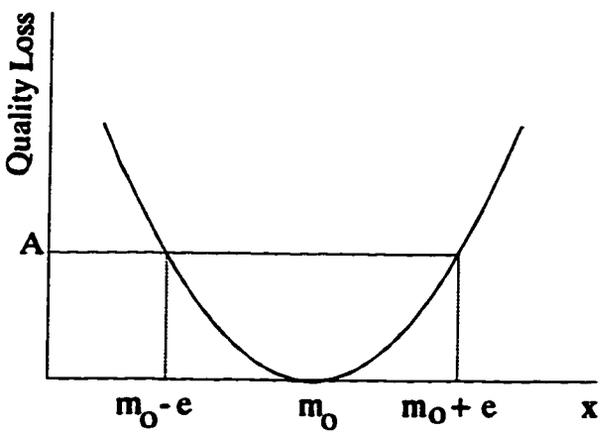

Figure 3.2 Quality loss and deviation from target value

The maximum quality loss is A which represents the cost of rework or cost of scrapped part. The maximum quality loss is reached when the deviation of the functional characteristic of the part from the target value m_o is equal to tolerance limit e (Figure 3.2). Then, from Equation (3.2), we have

$$L(m_o + e) = k(m_o + e - m_o)^2 \tag{3.3}$$

Therefore,

$$A = ke^2 \tag{3.4}$$

and

$$k = \frac{A}{e^2} \tag{3.5}$$

Substituting Equation (3.5) into Equation (3.2), the quality loss function is

$$L(x) = \frac{A}{e^2}(x - m_o)^2 \tag{3.6}$$

In the metal cutting context, the quality loss-tolerance function can be written in the following form:

$$L_q(\delta) = \frac{A}{\delta_u^2}\delta^2 \tag{3.7}$$

where δ is the actual tolerance, and δ_u is the maximum allowed tolerance.

Adding machining cost-tolerance and quality loss-tolerance functions together, the objective function is obtained and the first-stage model can be formulated as follows.

Model 1

$$Min\ G(\delta) = \sum_{k=1}^{K} \left\{ \left[g_{1k} e^{-g_{2k}(\delta_k - g_{3k})} + g_{4k} \right] + \frac{A_k}{\delta_{uk}^2}\delta_k^2 \right\} \tag{3.8}$$

subject to:

$$\delta_{lk} \leq \delta_k \leq \delta_{uk} \tag{3.9}$$

The objective of the model is to find optimal tolerance of different products or features k at minimum total cost in the design stage. Constraint (3.9) defines the region at which the tolerance is technically feasible. Figure 3.3 below shows the effect of tolerance on machining cost, quality loss, and total cost.

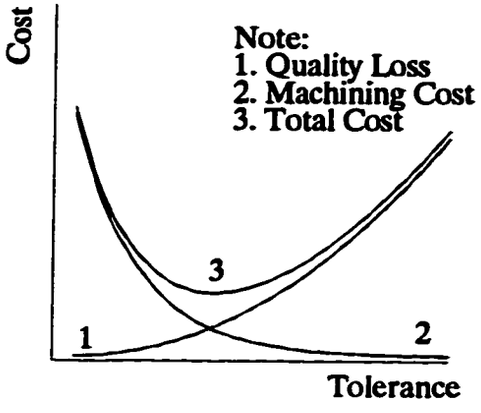

Figure 3.3 Machining cost, quality loss, and total cost

3.3 Optimization of Machining Parameters and Tool Adjustment Decisions

The second-stage models for the optimization of machining parameters and tool adjustment decisions are presented in the following two sections. Section 3.3.1 deals with single-machine multi-pass turning problems and Section 3.3.2 multi-machine multi-pass

turning problems. Three models are developed in Section 3.3.1 while two models are developed in Section 3.3.2. The main cost components to be considered in the second-stage models are discussed below.

Machining cost

Machining cost is the cost incurred during the actual machining process. Machining time decreases with the increase of cutting speed v (m/min) and feed rate f (mm/rev) and is given by:

$$t_m = \frac{\pi D L}{1000 v f} \tag{3.10}$$

where D is the diameter of the workpiece (mm), and L is the length of the workpiece (mm).

Unit machining cost, i.e., machining cost per piece is the product of machining time t_m and operating cost rate C_o. Operating cost includes overhead cost plus labour cost. Therefore, unit machining cost is:

$$C_m = \frac{\pi D L C_o}{1000 v f} \tag{3.11}$$

Tool cost

Tool cost is the cost per cutting edge which depends on tool life T_L (min) and machining parameters. When the tool follows a gradual or progressive wear, Taylor's equation offers a good estimate for tool life. The Taylor's tool-life equation is:

$$v T_L^n = C \tag{3.12}$$

where C and n are constants depending on work material, tool material, tool geometry, and machining conditions. Although the cutting speed is the most significant machining parameter affecting tool life, other parameters such as feed rate f (mm/rev) and depth of cut d (mm) are also important and are included in the extended Taylor's tool-life equation. Wang (1992) expressed the extended Taylor's tool-life equation as:

$$T_L = \frac{K_T}{v^{\alpha_T} f^{\beta_T} d^{\gamma_T}} \tag{3.13}$$

where α_T, β_T, γ_T, and K_T are deterministic constants and tool-work dependent.

If C_t is denoted as tool cost per cutting edge, then tool cost of machining a single workpiece is given by:

$$C_{to} = \frac{t_m}{T_L} C_t \tag{3.14}$$

t_m and T_L are respectively given in Equations (3.10) and (3.13), hence

$$\frac{t_m}{T_L} = \frac{\pi D L}{1000 K_T} v^{\alpha_T - 1} f^{\beta_T - 1} d^{\gamma_T} \tag{3.15}$$

Substituting Equation (3.15) into Equation (3.14), the tool cost per piece is:

$$C_{to} = \frac{\pi D L}{1000 K_T} v^{\alpha_T - 1} f^{\beta_T - 1} d^{\gamma_T} C_t \tag{3.16}$$

Tool replacement cost

The cost for the time required to remove a worn tool from a machine and replace with a new tool or regrind the same one is called tool replacement cost. The relationship between tool replacement time and cutting speed is that when the cutting speed is increased, the tool life is reduced thus increasing total tool replacement time.

Denoting t_r as the time to replace a tool, the tool replacement time distributed to each workpiece is given by:

$$t_c = \frac{t_m}{T_L} t_r \tag{3.17}$$

Tool replacement cost is the product of tool replacement time and operating cost rate, i.e.,

$$C_{tr} = \frac{t_m}{T_L} C_o t_r \tag{3.18}$$

Substituting Equation (3.15) into Equation (3.18), we obtain

$$C_{tr} = \frac{\pi D L}{1000 K_T} v^{\alpha_T - 1} f^{\beta_T - 1} d^{\gamma_T} C_o t_r \tag{3.19}$$

Tool adjustment cost

The gradual or progressive wear of the cutting tool during the machining process will affect part dimensions. Due to tool wear, the dimension of the part at the end of the process may be larger than that at the beginning of the process (Figure 3.4). To maintain better part quality, the tool should be adjusted when the dimension deviation reaches a

preset tolerance limit.

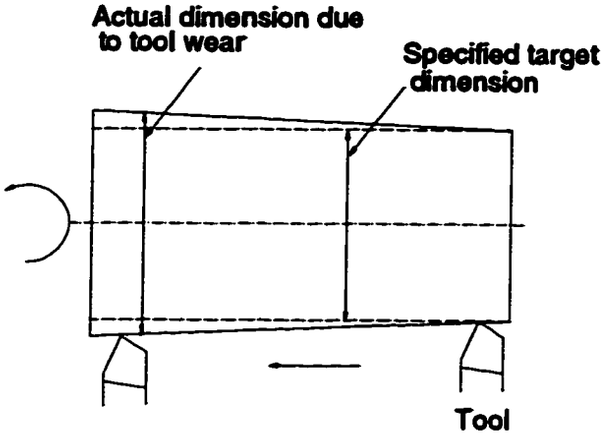

Figure 3.4 Effect of tool wear on dimension

A simple method for performing on-line tool wear monitoring is the deviation probing method (Vajpayee 1995). This method uses probes and sensors to monitor part dimension change. One end of the probe is connected to the machine control unit (MCU), and the other end touches the work with a stylus. The required dimension deviation is included in the part program. Whenever this value reaches the tolerance limit, appropriate compensation of tool offset is carried out during machining. Nevertheless, in most cases, the part dimension change cannot be accurately monitored due to the vibration inherent in the machining processes. The application of this method is thus still very limited.

The optical tool condition sensor is also being investigated for detecting tool wear such as average flank wear and nose wear. The flank wear information will be used for tool replacement whereas nose wear for tool position compensation (Du *et al* 1993).

However, compensating the tool offset during machining requires extra control system implementation and heavy investment which may not be practical in reality (Wang *et al*, 1996). The severe machining conditions caused by cutting fluid, chips, vibrations further hinder the application of this method. In today's manufacturing environment, the dimension measuring and tool adjustment are mostly done off-line. This involves additional machine down time and labour cost. Therefore, a trade-off between the improved quality and increased cost has to be taken into account.

According to Narang and Fischer (1993), the amount of tool wear causing dimension deviation during a pass for a single workpiece is:

$$w_p = \frac{t_m}{T_L} w \tag{3.20}$$

where w is the maximum allowed nose wear of the cutting tool in mm.

The number of tool adjustments during a single pass for a single workpiece is therefore:

$$n_{ad} = \frac{\frac{t_m}{T_L} w}{y} \tag{3.21}$$

Accordingly, when considering off-line tool adjustments, the tool adjustment cost distributed to each workpiece is:

$$C_{ad} = \frac{\frac{t_m}{T_L} w}{y} C_a t_a \tag{3.22}$$

where y is part dimension deviation (mm), C_a is tool adjustment cost rate ($/min), and t_a is tool adjustment time (min).

When Equation (3.15) is substituted into Equation (3.22), we have

$$C_{ad} = \frac{\pi D L}{1000 K_T} v^{a_T - 1} f^{\beta_T - 1} d^{\gamma_T} \frac{w}{y} C_a t_a \tag{3.23}$$

Similarly, the tool adjustment time per piece can be expressed as:

$$t_{ad} = \frac{\pi D L}{1000 K_T} v^{a_T - 1} f^{\beta_T - 1} d^{\gamma_T} \frac{w}{y} t_a \tag{3.24}$$

Quality loss

Referring to Equations (3.7) and (3.21), the quality loss per piece caused by dimension deviation is:

$$L_q = \frac{\frac{t_m}{T_L} w}{y} \left(\frac{A}{\delta^2} y^2 \right) \tag{3.25}$$

where A is defined as the rework or scrap cost, and δ is the maximum allowed tolerance in the manufacturing stage. Note that δ instead of δ_a is used in Equation (3.25) simply because we are now dealing with a manufacturing stage problem and the value of δ is determined by solving Model 1.

In Equation (3.25), $\frac{\frac{t_m}{T_L} w}{y}$ represents the number of adjustments distributed to each workpiece and $\frac{A}{\delta^2} y^2$ is the cumulative quality loss over the time between two

consecutive tool adjustments. Substituting Equation (3.15) into Equation (3.25) results in:

$$L_q = \frac{\pi D L}{1000 K_T} v^{\alpha_T - 1} f^{\beta_T - 1} d^{\gamma_T} \frac{w}{y} \left(\frac{A}{\delta^2} y^2 \right) \tag{3.26}$$

The total unit production cost for a multi-pass turning operation is the sum of machining cost, tool cost, tool replacement cost, tool adjustment cost and quality loss of all passes which is given by:

$$C_u = \sum_{j=1}^{J} [C_{mj} + C_{toj} + C_{trj}] + C_{adJ} + L_{qJ} \tag{3.27}$$

Substituting Equations (3.11), (3.16), (3.19), (3.23) and (3.26) into Equation (3.27), it follows that

$$\begin{aligned} C_u = \sum_{j=1}^{J} \left[\frac{\pi D_{j-1} L C_o}{1000 v_j f_j} + \frac{\pi D_{j-1} L}{1000 K_T} v_j^{\alpha_T - 1} f_j^{\beta_T - 1} d_j^{\gamma_T} (C_t + C_o t_r) \right] \\ + \frac{\pi D_{J-1} L}{1000 K_T} v_J^{\alpha_T - 1} f_J^{\beta_T - 1} d_J^{\gamma_T} \frac{w}{y} \left(C_a t_a + \frac{A}{\delta^2} y^2 \right) \end{aligned} \tag{3.28}$$

where:

$$D_{j-1} = D_0 - 2 \sum_{q=1}^{j-1} d_q \tag{3.29}$$

Note that the tool adjustment cost and quality loss are considered only in the last cutting pass where the final dimension of the part is obtained.

Similarly, the total unit production time is the sum of machining time, replacement time and tool adjustment time as given below:

$$t_u = \sum_{j=1}^{J}\left[\frac{\pi D_{j-1} L}{1000 v_j f_j} + \frac{\pi D_{j-1} L}{1000 K_T} v_j^{\alpha_T - 1} f_j^{\beta_T - 1} d_j^{\gamma_T} t_r\right] + \frac{\pi D_{J-1} L}{1000 K_T} v_J^{\alpha_T - 1} f_J^{\beta_T - 1} d_J^{\gamma_T} \frac{w}{y} t_a \qquad (3.30)$$

The Equations (3.28) and (3.30) are important in the development of models in the following sections and will be frequently used.

3.3.1 Single-machine Turning Processes

In a single-machine turning operation, all cutting operations of a workpiece are performed on a single machine. In this section, three single-machine multi-pass turning problems are formulated and are presented below.

3.3.1.1 Single-product, Multi-pass Model Considering Pass Selection

Traditionally, the depth of cut, and the number of layers or passes needed to machine a stock to the final dimension of a part are either selected by an operator or from handbook. For example, four passes are specified for the part shown in Figure 3.5 with more depth of cut for the first two passes and less depth of cut for the last two. However, the depth of cut and the number of passes so selected are often either too conservative leading to low productivity or too aggressive causing unnecessary tool breakages and part defects. To solve this problem, the pass selection decision should be

incorporated. Keeping this in mind, a model is developed to simultaneously provide optimum cutting speed, feed rate, depth of cut, number of passes, and dimension deviation for tool adjustment. The objective is to minimize the unit production cost. The model is as follows below.

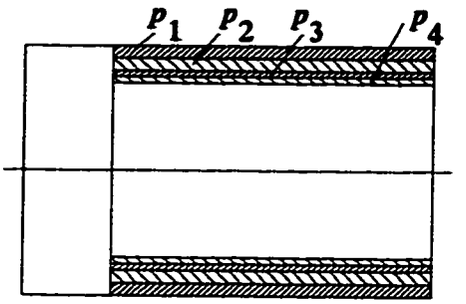

Figure 3.5 An example of a four-pass turning operation

Model 2.1

$$Min\ C_u = \sum_{j=1}^{J} \left[\frac{\pi D_{j-1} L C_o}{1000 v_j f_j} + \frac{\pi D_{j-1} L}{1000 K_T} v_j^{\alpha_T - 1} f_j^{\beta_T - 1} d_j^{\gamma_T} (C_t + C_o t_r) \right] p_j$$

$$+ \frac{\pi D_{J-1} L}{1000 K_T} v_J^{\alpha_T - 1} f_J^{\beta_T - 1} d_J^{\gamma_T} \frac{w}{y} \left(C_a t_a + \frac{A}{\delta^2} y^2 \right) p_J \qquad (3.31)$$

where:

$$D_{j-1} = D_0 - 2 \sum_{q=1}^{j-1} d_q$$

subject to:

$$v_j^{(l)} \le v_j \le v_j^{(u)}, \quad \forall j \tag{3.32}$$

$$f_j^{(l)} \le f_j \le f_j^{(u)}, \quad \forall j \tag{3.33}$$

$$d_j \le d_j^{(u)}, \quad j=1,2,\ldots,J\text{-}1 \tag{3.34}$$

$$d_J^{(l)} \le d_J \le d_J^{(u)} \tag{3.35}$$

$$0 \le y \le \delta \tag{3.36}$$

$$p_J = 1 \tag{3.37}$$

$$\sum_{j=1}^{J} d_j = d_{total} \tag{3.38}$$

$$\sum_{j=1}^{J} d_j p_j = d_{total} \tag{3.39}$$

$$\frac{d_j}{d_{total}} - p_j \le 0, \quad \forall j \tag{3.40}$$

$$0 \le p_j \le 1, \quad j=1, 2,\ldots,J\text{-}1 \tag{3.41}$$

$$K_F f_j^{\beta_F} d_j^{\gamma_F} \le F_c^{(u)}, \quad \forall j \tag{3.42}$$

$$K_P v_j f_j^{\beta_F} d_j^{\gamma_F} \leq P_c^{(u)}, \quad \forall j \tag{3.43}$$

$$K_S v_J^{\alpha_S} f_J^{\beta_S} d_J^{\gamma_S} \leq S_F^{(u)} \tag{3.44}$$

The objective function of Model 2.1 minimizes the total unit production cost. The decision variables are v_j, f_j, d_j, y and p_j which represent the cutting speed, feed rate, depth of cut, dimension deviation for tool adjustment and pass selection, respectively. For a given tool-workpiece material combination, constraints (3.32) through (3.34) give the lower and upper bounds for cutting speed, feed rate, and depth of cut respectively. Constraint (3.35) restricts the depth of cut for the last pass. Constraint (3.36) specifies that the part dimension deviation at which the tool is adjusted should be within the tolerance specified in the design stage. Constraint (3.37) shows that at least one pass is justified and should be performed. If one pass is justified, it should be the last pass since otherwise the tool adjustment cost and quality loss cannot be considered. Constraint (3.38) states that the sum of depth of cut from each pass should be equal to the total depth of cut to be removed from the material. Constraint (3.40) specifies the conditions that if a pass is not selected, the depth of cut should be zero, and if a pass is justified, the depth of cut must be greater than zero. The purpose of constraint (3.41) is to use continuous variables P_j. Constraints (3.38) through (3.41) all together guarantee that p_j will take a binary value 0 or 1 even if this is not explicitly specified. As p_j is a continuous variable, the computational time can be significantly reduced.

The cutting force and cutting power are mostly considered in the selection of cutting speed, feed, and depth of cut. The greater the cutting speed, feed and depth of cut, the greater the deflection of the workpiece and tool, dimensional error, and the higher the power consumption. Therefore associated limits are set such that the cutting force and cutting power do not exceed their maximum allowable values. The limiting cutting force and cutting power are given in constraints (3.42) and (3.43) respectively as suggested by Iwata *et al* (1977). The empirical coefficients β_F, γ_F and K_F represent the factors that can affect the cutting force and depend on tool-work material combination, machining process, workpiece hardness and tool geometry. The empirical coefficient K_P for cutting power is the function of coefficient K_F and mechanical efficiency of the machine tool. The mechanical efficiency η_m indicates the power lost due to friction. The empirical coefficient K_P is expressed by Iwata *et al* (1972) as:

$$K_P = \frac{K_F}{6120\,\eta_m} \tag{3.45}$$

The experimental study of Bhattacharyya *et al* (1970) has shown that the cutting speed, feed and depth of cut can affect the surface roughness of the part. Constraint (3.44) reflects this study and sets a limit for surface roughness in the last pass. α_S, β_S, γ_S and K_S are constants for surface roughness and their values are dependent on the tool-work combination.

3.3.1.2 Single-product, Multi-pass Model Considering Inventory and Setup Costs

Model 2.1 has considered minimization of cost for production of a single workpiece on a single machine. In batch production where a product is manufactured in batches and when the demand is concerned, Model 2.1 is not applicable. A new model is needed to take care of batch production on a single machine such that the decision on batch size can be made. To be feasible, the demand is satisfied only when its rate is lower than the production rate. However, when the production rate is greater than the demand rate, there is a tendency of gradual inventory built-up during the production period. Therefore, the total cost is the sum of the production cost to satisfy the demand, the inventory holding cost which may be in the form of opportunity cost or deterioration cost, and the setup cost for setting the machine for each production batch. Before presenting the model, individual cost components are first calculated as follows.

Production cost

Denoting C_u as the unit production cost and λ the demand rate, the production cost to satisfy the demand is then given by:

$$C_Q = C_u \lambda \tag{3.46}$$

Inventory cost

The production batch size is the number of units produced in each cycle and is expressed as:

$$Q = P t_Q \tag{3.47}$$

Therefore

$$t_Q = \frac{1}{P} Q \tag{3.48}$$

where P is the production rate and t_Q is the time needed to produce a batch.

The unit production time can also be described as the reciprocal of production rate

$$t_u = \frac{1}{P} \tag{3.49}$$

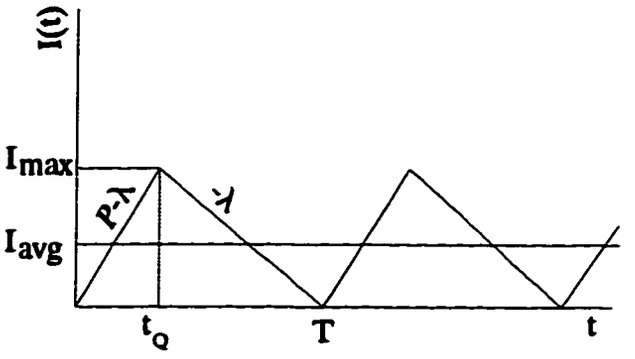

Figure 3.6 Inventory with finite production rate

From Figure 3.6, the maximum inventory is

$$I_{max} = (P - \lambda) t_Q \tag{3.50}$$

and substituting Equation (3.48) into Equation (3.50), we get

$$I_{max} = \left(1 - \frac{\lambda}{P}\right)Q \tag{3.51}$$

The average inventory is thus

$$I_{avg} = \frac{1}{2}\left(1 - \frac{\lambda}{P}\right)Q \tag{3.52}$$

The inventory holding cost per unit per period time is given by:

$$C_h = i_h C_u \tag{3.53}$$

where i_h is the inventory holding cost rate. Therefore, the inventory holding cost is obtained as:

$$I_c = \frac{1}{2}\left(1 - \frac{\lambda}{P}\right)Q\, i_h C_u \tag{3.54}$$

Setup cost

If C_s is the setup cost per production run, the total setup cost over a planning period is given by:

$$S_c = \frac{\lambda}{Q} C_s \tag{3.55}$$

where λ/Q represents the number of production runs during the planning period.

The total cost is the sum of production cost, inventory cost and setup cost, i.e.,

$$T_c = C_u \lambda + \frac{1}{2}\left(1 - \frac{\lambda}{P}\right) i_h C_u Q + \frac{\lambda}{Q} C_s \tag{3.56}$$

From Equation (3.49), we have

$$P = \frac{1}{t_u} \tag{3.57}$$

Substituting Equation (3.30) into Equation (3.57), we get

$$P = \frac{1}{\sum_{j=1}^{J}\left(\frac{\pi D_{j-1} L}{1000 v_j f_j} + \frac{\pi D_{j-1} L}{1000 K_T} v_j^{\alpha_T - 1} f_j^{\beta_T - 1} d_j^{\gamma_T} t_r\right) + \frac{\pi D_{J-1} L}{1000 K_T} v_J^{\alpha_T - 1} f_J^{\beta_T - 1} d_J^{\gamma_T} \frac{w}{y} t_a} \tag{3.58}$$

Similarly, substituting Equations (3.28) and (3.58) into Equation (3.56), the model of minimizing total cost can be formulated as follows.

Model 2.2

$$\begin{aligned} Min\ T_c = & \left\{\sum_{j=1}^{J}\left[\frac{\pi D_{j-1} L C_o}{1000 v_j f_j} + \frac{\pi D_{j-1} L}{1000 K_T} v_j^{\alpha_T - 1} f_j^{\beta_T - 1} d_j^{\gamma_T}(C_t + C_o t_r)\right]\right. \\ & \left. + \frac{\pi D_{J-1} L}{1000 K_T} v_J^{\alpha_T - 1} f_J^{\beta_T - 1} d_J^{\gamma_T} \frac{w}{y}\left(C_a t_a + \frac{A}{\delta^2} y^2\right)\right\}\left\{\lambda\right. \\ & + \frac{i_h Q}{2}\left[1 - \lambda\left(\sum_{j=1}^{J}\left(\frac{\pi D_{j-1} L}{1000 v_j f_j} + \frac{\pi D_{j-1} L}{1000 K_T} v_j^{\alpha_T - 1} f_j^{\beta_T - 1} d_j^{\gamma_T} t_r\right)\right.\right. \\ & \left.\left. + \frac{\pi D_{J-1} L}{1000 K_T} v_J^{\alpha_T - 1} f_J^{\beta_T - 1} d_J^{\gamma_T} \frac{w}{y} t_a\right)\right]\Bigg\} + \frac{\lambda}{Q} C_s \end{aligned} \tag{3.59}$$

subject to:

$$v_j^{(l)} \le v_j \le v_j^{(u)}, \quad \forall j \tag{3.60}$$

$$f_j^{(l)} \le f_j \le f_j^{(u)}, \quad \forall j \tag{3.61}$$

$$d_j^{(l)} \le d_j \le d_j^{(u)}, \quad \forall j \tag{3.62}$$

$$\sum_{j=1}^{J} d_j = d_{total} \tag{3.63}$$

$$0 \le y \le \delta \tag{3.64}$$

$$K_F f_j^{\beta_F} d_j^{\gamma_F} \le F_c^{(u)}, \quad \forall j \tag{3.65}$$

$$K_P v_j f_j^{\beta_F} d_j^{\gamma_F} \le P_c^{(u)}, \quad \forall j \tag{3.66}$$

$$K_S v_J^{\alpha_S} f_J^{\beta_S} d_J^{\gamma_S} \le S_F^{(u)} \tag{3.67}$$

$$\frac{1}{\sum_{j=1}^{J}\left(\frac{\pi D_{j-1} L}{1000 v_j f_j} + \frac{\pi D_{j-1} L}{1000 K_T} v_j^{\alpha_T - 1} f_j^{\beta_T - 1} d_j^{\gamma_T} t_r\right) + \frac{\pi D_{J-1} L}{1000 K_T} v_J^{\alpha_T - 1} f_J^{\beta_T - 1} d_J^{\gamma_T} \frac{w}{y} t_a} \ge P_r$$

$$\text{for } P_r > \lambda \tag{3.68}$$

where:

$$D_{j-1} = D_0 - 2\sum_{q=1}^{j-1} d_q$$

The objective of Model 2.2 is to minimize the total cost consisting of production cost, inventory cost and setup cost, and to obtain optimum cutting speed, feed rate, depth of cut, dimension deviation for tool adjustment, and production batch size. Constraints (3.60), (3.61) and (3.62) specify lower and upper bounds for the cutting speed, feed rate and depth of cut respectively. Constraint (3.63) states that the total depth of cut should be equal to the total material to be removed. Constraint (3.64) says that the dimension deviation at which the tool is adjusted should meet the tolerance specification from the design stage. The restrictions for cutting force and cutting power are respectively presented in constraints (3.65) and (3.66). Constraint (3.67) imposes the limit for surface roughness in the last pass. Constraint (3.68) requires that the production rate should be greater than or equal to the demand rate.

3.3.1.3 Multi-product, Multi-pass Model Considering Inventory and Setup Costs

Model 2.2 is applied only when one type of product is processed on a single NC machine. The model to be developed in this sub-section will be used to solve a multi-product problem with a cyclic production policy. This policy is commonly used when different types of products are produced in batches on a single machine. In this case, all types of products will be processed in turn but only one batch for each product is to be

scheduled in a single cycle. In order to ensure that the machine is able to satisfy the demand of all types of products, the following condition is needed (Nahmias 1993):

$$\sum_{k=1}^{K} \frac{\lambda_k}{P_k} \leq 1 \tag{3.69}$$

where λ_k is the demand rate for product k, (parts/min) and P_k is the production rate for product k, (parts/min).

Let T be the cycle time. The production batch size for product k in each production cycle is given by:

$$Q_k = \lambda_k T \tag{3.70}$$

Equation (3.56) can be extended to multi-product case:

$$T_c = \sum_{k=1}^{K} \left[C_{uk} \lambda_k + \frac{1}{2} \left(1 - \frac{\lambda_k}{P_k} \right) i_h C_{uk} Q_k + \frac{\lambda_k}{Q_k} C_{sk} \right] \tag{3.71}$$

substituting Q_k from Equation (3.70) into Equation (3.71), we have

$$T_c = \sum_{k=1}^{K} \left[C_{uk} \lambda_k + \frac{1}{2} \left(1 - \frac{\lambda_k}{P_k} \right) i_h C_{uk} \lambda_k T + \frac{C_{sk}}{T} \right] \tag{3.72}$$

The production rate of product k is expressed as:

$$P_k = \frac{1}{t_{uk}} \tag{3.73}$$

Referring to Equation (3.28) and Equation (3.30), the unit production cost and production time of product k can respectively be written as:

$$C_{uk} = \sum_{j=1}^{J_k}\left[\frac{\pi D_{k(j-1)} L_k C_o}{1000 v_{kj} f_{kj}} + \frac{\pi D_{k(j-1)} L_k}{1000 K_T} v_{kj}^{\alpha_T - 1} f_{kj}^{\beta_T - 1} d_{kj}^{\gamma_T}\left(C_t + C_o t_r\right)\right]$$

$$+ \frac{\pi D_{k(J_k-1)} L_k}{1000 K_T} v_{kJ_k}^{\alpha_T - 1} f_{kJ_k}^{\beta_T - 1} d_{kJ_k}^{\gamma_T} \frac{w}{y_{kJ_k}}\left(C_a t_a + \frac{A}{\delta_{kJ_k}^2} y_{kJ_k}^2\right) \tag{3.74}$$

and

$$t_{uk} = \sum_{j=1}^{J_k}\left(\frac{\pi D_{k(j-1)} L_k}{1000 v_{kj} f_{kj}} + \frac{\pi D_{k(j-1)} L_k}{1000 K_T} v_{kj}^{\alpha_T - 1} f_{kj}^{\beta_T - 1} d_{kj}^{\gamma_T} t_r\right) + \frac{\pi D_{k(J_k-1)} L_k}{1000 K_T} v_{kJ_k}^{\alpha_T - 1} f_{kJ_k}^{\beta_T - 1} d_{kJ_k}^{\gamma_T} \frac{w}{y_{kJ_k}} t_a \tag{3.75}$$

whereas the production rate of product k is obtained as:

$$P_k = \frac{1}{\sum_{j=1}^{J_k}\left(\frac{\pi D_{k(j-1)} L_k}{1000 v_{kj} f_{kj}} + \frac{\pi D_{k(j-1)} L_k}{1000 K_T} v_{kj}^{\alpha_T - 1} f_{kj}^{\beta_T - 1} d_{kj}^{\gamma_T} t_r\right) + \frac{\pi D_{k(J_k-1)} L_k}{1000 K_T} v_{kJ_k}^{\alpha_T - 1} f_{kJ_k}^{\beta_T - 1} d_{kJ_k}^{\gamma_T} \frac{w}{y_{kJ_k}} t_a} \tag{3.76}$$

where:

$$D_{k(j-1)} = D_{k0} - 2\sum_{q=1}^{j-1} d_{kq}, \quad \forall k \tag{3.77}$$

and D_{k0} is the initial stock diameter of product k (mm).

The objective function is then formed by substituting C_{uk} and P_k into Equation (3.72). Rearranging the terms, the optimization model can thus be presented as follows:

Model 2.3

$$Min\ Tc = \sum_{k=1}^{K}\left[\left\{\sum_{j=1}^{J_k}\left[\frac{\pi D_{k(j-1)}L_k C_o}{1000\, v_{kj} f_{kj}} + \frac{\pi D_{k(j-1)}L_k}{1000\, K_T} v_{kj}^{\alpha_T - 1} f_{kj}^{\beta_T - 1} d_{kj}^{\gamma_T}\left(C_t + C_o t_r\right)\right] + \frac{\pi D_{k(J_k-1)}L_k}{1000\, K_T} v_{kJ_k}^{\alpha_T - 1} f_{kJ_k}^{\beta_T - 1} d_{kJ_k}^{\gamma_T} \frac{w}{y_{kJ_k}}\left(C_a t_a + \frac{A}{\delta_{kJ_k}^2} y_{kJ_k}^2\right)\right\}\left\{\lambda_k + \frac{1}{2} i_h \lambda_k T\left[1 - \lambda_k\left(\sum_{j=1}^{J_k}\left(\frac{\pi D_{k(j-1)}L_k}{1000\, v_{kj} f_{kj}} + \frac{\pi D_{k(j-1)}L_k}{1000 K_T} v_{kj}^{\alpha_T - 1} f_{kj}^{\beta_T - 1} d_{kj}^{\gamma_T} t_r\right) + \frac{\pi D_{k(J_k-1)}L_k}{1000\, K_T} v_{kJ_k}^{\alpha_T - 1} f_{kJ_k}^{\beta_T - 1} d_{kJ_k}^{\gamma_T} \frac{w}{y_{kJ_k}} t_a\right)\right]\right\} + \frac{C_{sk}}{T}\right] \quad (3.78)$$

subject to:

$$v_{kj}^{(l)} \le v_{kj} \le v_{kj}^{(u)}, \quad \forall\, k, j \quad (3.79)$$

$$f_{kj}^{(l)} \le f_{kj} \le f_{kj}^{(u)}, \quad \forall\, k, j \quad (3.80)$$

$$d_{kj}^{(l)} \le d_{kj} \le d_{kj}^{(u)}, \quad \forall\, k, j \quad (3.81)$$

$$\sum_{j=1}^{J_k} d_{kj} = d_{ktotal}, \quad \forall k \tag{3.82}$$

$$0 \le y_{kJ_k} \le \delta_{kJ_k}, \quad \forall k \tag{3.83}$$

$$K_F f_{kj}^{\beta_F} d_{kj}^{\gamma_F} \le F_c^{(u)}, \quad \forall k, j \tag{3.84}$$

$$K_P v_{kj} f_{kj}^{\beta_F} d_{kj}^{\gamma_F} \le P_c^{(u)}, \quad \forall k, j \tag{3.85}$$

$$K_S v_{kJ_k}^{\alpha_S} f_{kJ_k}^{\beta_S} d_{kJ_k}^{\gamma_S} \le S_{Fk}^{(u)}, \quad \forall k \tag{3.86}$$

$$\frac{1}{\sum_{j=1}^{J_k}\left(\frac{\pi D_{k(j-1)} L_k}{1000 v_{kj} f_{kj}} + \frac{\pi D_{k(j-1)} L_k}{1000 K_T} v_{kj}^{\alpha_T - 1} f_{kj}^{\beta_T - 1} d_{kj}^{\gamma_T} t_r\right) + \frac{\pi D_{k(J_k-1)} L_k}{1000 K_T} v_{kJ_k}^{\alpha_T - 1} f_{kJ_k}^{\beta_T - 1} d_{kJ_k}^{\gamma_T} \frac{w}{y_{kJ_k}} t_a} \ge P_{rk}$$

$$\forall k \tag{3.87}$$

where:

$$D_{k(j-1)} = D_{k0} - 2\sum_{q=1}^{j-1} d_{kq}, \quad \forall k$$

The objective of Model 2.3 is to minimize the total cost which consists of production cost, inventory cost and setup cost. The model, once solved, will provide optimum cutting speed, feed rate, depth of cut, dimension deviation and cycle time. Constraints (3.79) through (3.81) specify cutting speed, feed rate and depth of cut lower

and upper bounds respectively. Constraint (3.82) states that the sum of depth of cut from each pass of product k is equal to the total stock of material of product k to be removed. Constraint (3.83) assures that the part dimension deviation in the last pass of product k is within the required tolerance from the design stage. Constraints (3.84) and (3.85) give the limits for cutting force and cutting power respectively. The maximum allowed surface roughness for the last pass of product k is imposed in constraint (3.86). Constraint (3.87) ensures that the production rate should be greater than or equal to the demand rate.

3.3.2 Multi-machine Turning Processes

Generally, a multi-machine manufacturing system involves several machines. The machines may be arranged in a job shop, flow shop or cellular layout. In job shop environment, these machines are grouped together to perform similar operations for different parts. For example, several general-purpose NC lathes may form a turning work center. In flow shop, the machines are arranged together according to the process sequences. In cellular systems, the machines are grouped according to the process combinations that occur in families of parts. The main advantage of a job shop layout is its flexibility where there is less restriction on part movement therefore allowing alternative part routings.

Suppose multi-pass turning operations are performed within a turning work center of a job shop to transform a raw material stock to a finished part as shown in Figure 3.7. Different features are produced by removing unwanted material to obtain a finished part. The problem is to assign the cutting passes of the features to the available machines so

that the total production cost or cycle time can be minimized. In this section, two models associated with different objectives are formulated. One model is formulated with the objective of minimizing the total production cost and the other is formulated with the objective of minimizing the cycle time.

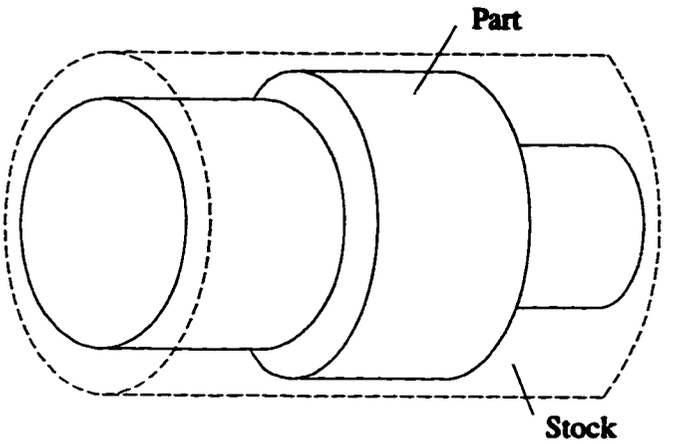

Figure 3.7 Transformation of a stock to a finished part

3.3.2.1 Multi-feature, Multi-pass: Minimizing Total Cost

The model in this section is developed to minimize the total unit production cost by assigning workload to individual machines and properly selecting machining parameters. Solving this model will simultaneously provide optimum workload for each machine, optimum machining parameters and feature dimension deviations for all machine-pass-feature combinations. The problem is formulated as follows.

Model 2.4

$$Min\ C_u = \sum_{i=1}^{m}\sum_{j=1}^{J_k}\sum_{k=1}^{K}\left[\frac{\pi D_{(j-1)k}L_k C_o}{1000 v_{ijk} f_{ijk}} + \frac{\pi D_{(j-1)k}L_k}{1000 K_T} v_{ijk}^{\alpha_T - 1} f_{ijk}^{\beta_T - 1} d_{ijk}^{\gamma_T}(C_t + C_o t_r)\right] X_{ijk}$$

$$+ \sum_{i=1}^{m}\sum_{k=1}^{K}\frac{\pi D_{(J_k-1)k}L_k}{1000 K_T} v_{iJ_k k}^{\alpha_T - 1} f_{iJ_k k}^{\beta_T - 1} d_{iJ_k k}^{\gamma_T}\frac{w}{y_{iJ_k k}}\left(C_a t_a + \frac{A}{\delta_{iJ_k k}^2} y_{iJ_k k}^2\right) X_{iJ_k k} \qquad (3.88)$$

subject to:

$$\sum_{j=1}^{J_k}\sum_{k=1}^{K}\left[\frac{\pi D_{(j-1)k}L_k}{1000 v_{ijk} f_{ijk}} + \frac{\pi D_{(j-1)k}L_k}{1000 K_T} v_{ijk}^{\alpha_T - 1} f_{ijk}^{\beta_T - 1} d_{ijk}^{\gamma_T} t_r\right] X_{ijk}$$

$$+ \sum_{k=1}^{K}\frac{\pi D_{(J_k-1)k}L_k}{1000 K_T} v_{iJ_k k}^{\alpha_T - 1} f_{iJ_k k}^{\beta_T - 1} d_{iJ_k k}^{\gamma_T}\frac{w}{y_{iJ_k k}} t_a X_{iJ_k k} \le Z, \quad \forall i \qquad (3.89)$$

$$\sum_{i=1}^{m}\sum_{j=1}^{J_k} d_{ijk} = d_{k\,total}, \quad \forall k \qquad (3.90)$$

$$\sum_{i=1}^{m}\sum_{j=1}^{J_k} d_{ijk} X_{ijk} = d_{k\,total}, \quad \forall k \qquad (3.91)$$

$$\sum_{i=1}^{m} X_{ijk} = 1, \quad \forall j, k \qquad (3.92)$$

$$X_{iJ_k k} = 1, \quad \forall i; k = i \qquad (3.93)$$

$$v_{ijk}^{(l)} \le v_{ijk} \le v_{ijk}^{(u)}, \quad \forall i, j, k \qquad (3.94)$$

$$f_{ijk}^{(l)} \le f_{ijk} \le f_{ijk}^{(u)}, \quad \forall\, i, j, k \tag{3.95}$$

$$d_{ijk} \le d_{ijk}^{(u)}, \quad \forall\, i, j, k \tag{3.96}$$

$$d_{iJ_k k}^{(l)} \le d_{iJ_k k} \le d_{iJ_k k}^{(u)}, \quad \forall\, i;\ k = i \tag{3.97}$$

$$0 \le y_{iJ_k k} \le \delta_{iJ_k k}, \quad \forall\, i, k \tag{3.98}$$

$$K_F f_{ijk}^{\beta_F} d_{ijk}^{\gamma_F} \le F_c^{(u)}, \quad \forall\, i, j, k \tag{3.99}$$

$$K_P v_{ijk} f_{ijk}^{\beta_F} d_{ijk}^{\gamma_F} \le P_c^{(u)}, \quad \forall\, i, j, k \tag{3.100}$$

$$K_S v_{iJ_k k}^{\alpha_S} f_{iJ_k k}^{\beta_S} d_{iJ_k k}^{\gamma_S} \le S_{Fk}^{(u)}, \quad \forall\, i, k \tag{3.101}$$

$$X_{ijk} \ge 0 \tag{3.102}$$

where:

$$D_{(j-1)k} = D_{0k} - 2\sum_{q=1}^{j-1} d_{iqk}, \quad \forall\, i, k \tag{3.103}$$

and D_{0k} is the original diameter of feature k (mm).

The objective function of Model 2.4 is similar to that of Model 2.1 with some modifications to accommodate several machines and features of a workpiece. The model is solved for optimal workload assignment, cutting speed, feed rate, depth of cut and

dimension deviation for each machine-pass-feature combination. Constraint (3.89) forces the production time of the machines not to exceed the cycle time. If the production time is less than the cycle time, then slack time is allowed. Constraint (3.90) indicates that the sum of depths of cut of a feature should be equal to total stock of material to be removed from that feature. Constraint (3.92) means that each pass of a feature has to be processed by only one machine. Constraints (3.90), (3.91) and (3.92) will jointly guarantee the value of X_{ijk} to be binary, either 0 or 1. Constraint (3.93) ensures that at least one pass, that is, the last pass of each feature must be performed. Constraints (3.94) through (3.96) give the lower and upper bounds for cutting speed, feed rate, depth of cut respectively. Constraint (3.97) restricts the depth of cut for the last pass of each feature to be in the range specified by the lower and upper bounds. Constraint (3.98) assures that the part dimension deviation is within allowed tolerance specifications. The restrictions for cutting force and cutting power are respectively presented in constraints (3.99) and (3.100). Finally, the surface roughness limit for last passes is imposed by constraint (3.101).

3.3.2.2 Multi-feature Multi-pass: Minimizing Cycle Time

In practice, the cycle time may become a more important concern than the production cost. In this case, a model is required to minimize cycle time. The model is formulated in the following.

Model 2.5

$$\textit{Min } Z \tag{3.104}$$

subject to:

$$\sum_{j=1}^{J_k}\sum_{k=1}^{K}\left[\frac{\pi D_{(j-1)k}L_k}{1000\,v_{ijk}f_{ijk}} + \frac{\pi D_{(j-1)k}L_k}{1000\,K_T}v_{ijk}^{\alpha_T-1}f_{ijk}^{\beta_T-1}d_{ijk}^{\gamma_T}t_r\right]X_{ijk}$$

$$+\sum_{k=1}^{K}\frac{\pi D_{(J_k-1)k}L_k}{1000\,K_T}v_{iJ_kk}^{\alpha_T-1}f_{iJ_kk}^{\beta_T-1}d_{iJ_kk}^{\gamma_T}\frac{w}{y_{iJ_kk}}t_a X_{iJ_kk} \le Z, \quad \forall i$$

$$\sum_{i=1}^{m}\sum_{j=1}^{J_k}d_{ijk} = d_{k\,total}, \quad \forall k$$

$$\sum_{i=1}^{m}\sum_{j=1}^{J_k}d_{ijk}X_{ijk} = d_{k\,total}, \quad \forall k$$

$$\sum_{i=1}^{m}X_{ijk} = 1, \quad \forall j, k$$

$$X_{iJ_kk} = 1, \quad \forall i; k = i$$

$$v_{ijk}^{(l)} \le v_{ijk} \le v_{ijk}^{(u)}, \quad \forall i, j, k$$

$$f_{ijk}^{(l)} \le f_{ijk} \le f_{ijk}^{(u)}, \quad \forall i, j, k$$

$$d_{ijk} \le d_{ijk}^{(u)}, \quad \forall i, j, k$$

$$d_{iJ_ik}^{(l)} \le d_{iJ_ik} \le d_{iJ_ik}^{(u)}, \quad \forall\, i;\, k = i$$

$$0 \le y_{iJ_ik} \le \delta_{iJ_ik}, \quad \forall\, i,\, k$$

$$K_F f_{ijk}^{\beta_F} d_{ijk}^{\gamma_F} \le F_c^{(u)}, \quad \forall\, i,\, j,\, k$$

$$K_P v_{ijk} f_{ijk}^{\beta_F} d_{ijk}^{\gamma_F} \le P_c^{(u)}, \quad \forall\, i,\, j,\, k$$

$$K_S v_{iJ_ik}^{\alpha_S} f_{iJ_ik}^{\beta_S} d_{iJ_ik}^{\gamma_S} \le S_{Fk}^{(u)}, \quad \forall\, i,\, k$$

$$X_{ijk} \ge 0$$

where:

$$D_{(j-1)k} = D_{0k} - 2\sum_{q=1}^{j-1} d_{iqk}, \quad \forall\, i,\, k$$

and D_{0k} is the original diameter of feature k (mm).

The main objective of Model 2.5 is to assign work loads to machines in a multi-machine multi-pass machining system with the objective of minimizing the maximum cycle time. All constraints in this model are the same as those used in Model 2.4. Solving Model 2.5 will result in optimum workload assignment, cutting speed, feed rate, depth of cut and dimension deviation for each machine-pass-feature combination, and cycle time.

Chapter 4

Computational Results

The first-stage model and second-stage models are all nonlinear programming problems (NLP). The first-stage model has a nonlinear objective function with inequality linear constraints. All second-stage models consist of nonlinear objective functions with inequality and equality constraints. Some constraints are linear and others are nonlinear. Nonlinear programming problems are difficult to solve even for small-sized problems. Fortunately, powerful nonlinear optimization software package such as LINGO have been developed and are commercially available. LINGO is selected for solving the models because it is equipped with fast internal solver for large-scaled problems and is also able to handle NLPs with both inequality and equality constraints.

To test the feasibility of the models developed in chapter 3, illustrative examples are given and solved in this chapter. Before presenting illustrative examples, the following section gives some guidelines for the selection of input data for the examples.

4.1 Selection of Input Data

4.1.1 Tolerance and Surface Roughness

Machining processes like honing and grinding may produce small tolerances and smooth surfaces while machining processes like sawing and flame cutting will produce large tolerances and rough surfaces. Unnecessary cost can be avoided if the machining process, tolerance and surface finish can be properly selected. It is not advisable to perform an expensive process onto a component whose tolerance and surface finish requirements can be met with a cheaper process. It is equally undesirable to select a tolerance or a surface finish level which cannot be attained by the existing machining processes.

The parameters for a generalized exponential machining cost-tolerance function for tolerance allocation are based on empirical data and are recorded by Dong and Soom (1990). These parameters are workpiece size dependent and cover different surface features such as external rotational features, hole features, plane features and location features. The parameters for external rotational features are given in Table 4.1. The surface roughness values for various machining processes are obtained from Lingaiah (1994).

Table 4.1 Parameters of machining cost-tolerance function for external rotational features

Dimension range (mm)	g_1	g_2	g_3	g_4
$D<40$	3.96	22.05	0.00	0.79
$40 \leq D < 500$	3.96	21.65	0.00	1.04
$D \geq 500$	3.96	21.20	0.00	1.29

4.1.2 Cutting Tool Materials and Tool Wear

The cutting tool must have the following physical and metallurgy properties: high yield strength, high wear resistance, high fracture toughness, high fatigue resistance, high thermal conductivity, high thermal shock resistance and good oxidation resistance. There are various materials for making cutting tools. Some cutting tool materials are carbon steels, high speed steels (HSS), carbides, ceramics and diamonds. Of all the cutting tools, HSS and cemented carbides are commonly used in machining operations because they are stronger than plain carbon steel tools and cheaper than ceramic and diamond tools.

During machining, the tool is subjected to severe mechanical stresses due to cutting forces causing tool wear which in turn affects the tool life, the surface quality and the dimensional accuracy of the workpiece. Different forms of tool wear are shown in Figure 4.1. The tool wear are nose wear, flank wear and crater wear. The average flank wear land width of 0.3 mm is often used as an indicator of the end of the tool life. The nose wear leads to changes to workpiece dimensions and can be used to compensate for dimensional errors (Du *et al* 1993). Therefore, the nose wear is used for tool adjustment rather than the flank wear.

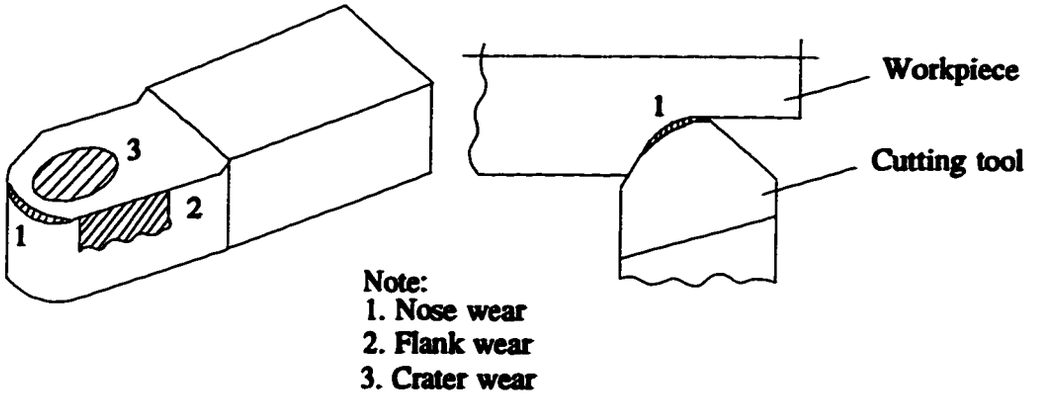

Figure 4.1 Tool wear classification

4.1.3 Cutting Speed, Feed rate and Depth of Cut

The cutting process variables such as cutting speed, feed rate, and depth of cut have direct influence on the cutting force, cutting power, the temperature rise at the chip-tool interface, tool life, type of chip, and dimensional accuracy and surface quality of the workpiece. Owing to the above fact, the process variables are often restricted within lower and upper bounds. Zohdi and Biles (1986) and Carvill (1993) recommended ranges of cutting speed, feed rate and depth of cut for different machining operations in different tool-workpiece material combinations.

4.2 Illustrative Examples

Example for Model 1

Consider a low carbon steel shaft which is to be machined from a rotational stock as shown in Figure 4.2. The length and diameter of the initial stock are respectively 250 mm and 90 mm. The diameter of the shaft is machined to 80 mm at a length of 200 mm by a turning process. The optimal tolerance of the final dimension is determined based on the following data given by Dong and Soom (1990): $g_{11} = 3.96$, $g_{21} = 21.65$, $g_{31} = 0.00$, $g_{41} = 1.04$. The minimum allowed tolerance is $\delta_{l1} = 0.002$ mm and the maximum allowed tolerance is $\delta_{u1} = 0.08$ mm. The maximum quality loss is assumed to be \$ 1.0. Using a LINGO program, the first-stage problem is solved in less than a second. The optimal tolerance is 0.0659 mm.

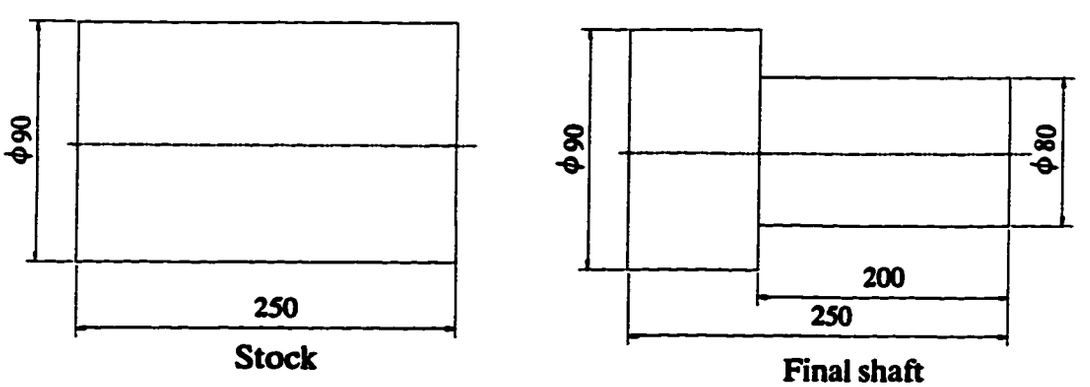

Figure 4.2 A rotational stock and a finished shaft

For all second-stage example problems that follow, the input data are given in Table 4.2. The estimated data based on tool life, surface roughness, cutting force and

cutting power are taken from Wang (1992). Part tolerance limit is 0.0659 mm and is obtained from Model 1. The tool material is cemented carbide and the workpiece material is low carbon steel. Additional data will be given in the following computational examples.

Table 4.2 Input parameters for all examples

Parameters	Values
Operating cost rate C_o	\$ 3.0/min
Tool cost C_t	\$ 5.5/edge
Tool replacement time t_r	0.5 min
Nose wear w	0.1 mm
Tool adjustment cost C_a	\$ 3.0/min
Tool adjustment time t_a	0.2 min
Rework cost A	\$ 1.0
Efficiency of machine tool η_m	0.8
Constants for tool life α_T, β_T,γ_T, K_T	1.70, 1.55, 1.22,1570000
Maximum cutting force $F_c^{(u)}$	20.0 Kg
Constants for cutting force β_F, γ_F, K_F	1.18, 1.26, 1.38
Maximum cutting power $P_c^{(u)}$	2.0 KW
Constants for surface roughness α_S, β_S, γ_S, K_S	-0.25, 0.72, 0.23, 1.17

Example for Model 2.1

As given above, the diameter D_0 of the low carbon steel stock is 90 mm and its length L is 250 mm. The shaft is machined on a stand-alone NC lathe. The total depth of cut is 5 mm and the length is 200 mm. The tolerance limit is $\delta = 0.0659$ mm. The required surface roughness $S_F^{(u)} = 1.6$ μm. A maximum of four cutting passes is considered. Other data are given in Table 4.3.

Table 4.3 Input data for Model 2.1

	Upper and lower bounds						
	Cutting speed (m/min)		Feed rate (mm/rev)		Depth of cut (mm)		
Pass	$v_j^{(l)}$	$v_j^{(u)}$	$f_j^{(l)}$	$f_j^{(u)}$	$d_j^{(u)}$	$d_j^{(l)}$	$d_j^{(u)}$
1	90	120	0.8	2.0	5.0		
2	90	120	0.8	2.0	5.0		
3	90	120	0.8	2.0	5.0		
4	168	210	0.13	0.50		0.3	1.0

A C code (Appendix A) is used to generate the LINGO-format-input for Model 2.1 example. The LINGO input is shown in Appendix B. Using LINGO on a 486 PC, the problem is solved in 3 seconds. The results are summarized in Table 4.4. The results indicate that two passes are selected out of the four passes. The total production cost for the two cutting passes is $ 2.2345 per piece.

Table 4.4 Results for Model 2.1 example

Pass	Variable		Value
	Cutting speed (m/min)	v_2	120.0000
2	Feed rate (mm/rev)	f_2	2.000000
	Depth of cut (mm)	d_2	4.361455
	Cutting speed (m/min)	v_4	210.0000
4	Feed rate (mm/rev)	f_4	0.500000
	Depth of cut (mm)	d_4	0.638545
	Dimension deviation (mm)	y	0.051056

According to Tan and Creese (1995), the maximum depth-of-cut constraint may have great influence on the unit production cost and the selected number of passes. To confirm this argument, additional calculations are performed for different maximum depths of cut of rough cutting and the associated unit production costs and selected number of passes are summarized in Table 4.5. The relationship between the maximum depth of cut and unit production cost is shown in Figure 4.3. In our case, a single pass will not be feasible because it should be a finish cutting which will not satisfy the total depth of cut to be removed. On the other hand, a single pass with very heavy depth of cut may also be infeasible due to excessive cutting force and power which may lead to tool breakage.

Table 4.5 Different maximum depths of cut in rough cutting for Model 2.1

Maximum depth of cut $d_j^{(u)}$ (mm)	Production cost ($/piece)	Number of passes
1.60	3.5468	4
1.67	3.5436	4
2.00	2.9112	3
2.33	2.8854	3
2.67	2.8791	3
3.00	2.8745	3
3.33	2.8703	3
3.67	2.8665	3
4.00	2.2449	2
4.33	2.2354	2
4.67	2.2345	2
5.00	2.2345	2

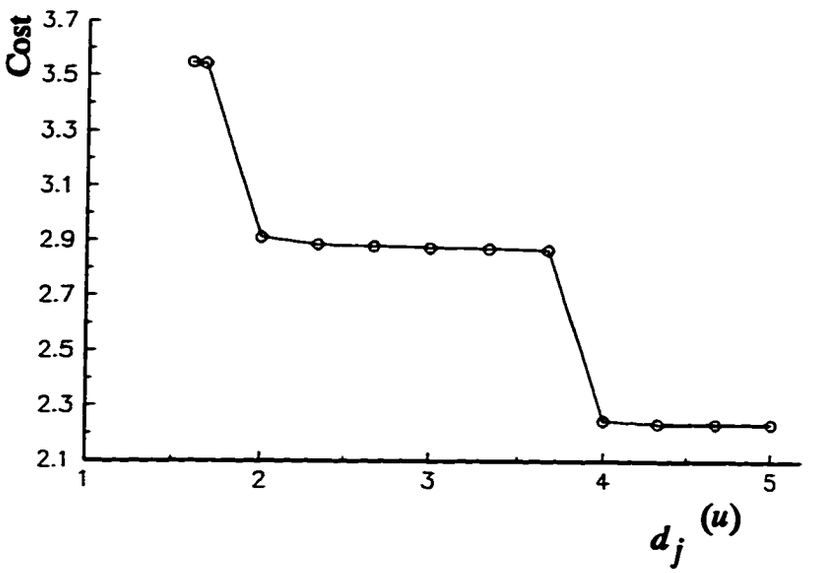

Figure 4.3 Maximum depth of cut versus unit production cost

Example for Model 2.2

The stock and shaft in the previous example are used here. In addition to the geometrical data used in the previous example, the following input data are needed. The maximum allowable tolerance and surface finish are $\delta = 0.0659$ mm and $S_F^{(u)} = 1.6$ μm respectively. The setup cost C_s is \$ 40.0 per setup and the inventory holding rate i_h is 20% per year (3.858×10^{-7} per min). The demand rate λ and the minimum production rate P_r are 0.2 part/min and 0.4 part/min respectively. The maximum allowed number of cutting passes are 4. Other data are given in Table 4.6.

Table 4.6 Input data for Model 2.2

	Upper and lower bounds							
	Cutting speed (m/min)		Feed rate (mm/rev)		Depth of cut (mm)			
Pass	$v_j^{(l)}$	$v_j^{(u)}$	$f_j^{(l)}$	$f_j^{(u)}$	$d_j^{(l)}$	$d_j^{(u)}$	$d_J^{(l)}$	$d_J^{(u)}$
1	90	120	0.8	2.0	1.0	5.0		
2	90	120	0.8	2.0	1.0	5.0		
3	90	120	0.8	2.0	1.0	5.0		
4	168	210	0.13	0.50			0.3	0.9

The C code used to generate the LINGO-input-format for Model 2.2 example is listed in Appendix C, and the LINGO-input format of Model 2.2 is in Appendix D. The problem is solved by LINGO on a 486 PC. The computational time is about 3 seconds. The total cost is $ 0.71045 per min. The optimum production batch size is 1627. The details are shown in Table 4.7.

Table 4.7 Results for Model 2.2 example

Pass	Variable		Value
1	Cutting speed (m/min)	v_1	120.0000
	Feed rate (mm/rev)	f_1	2.000000
	Depth of cut (mm)	d_1	2.700000
2	Cutting speed (m/min)	v_2	120.0000
	Feed rate (mm/rev)	f_2	2.000000
	Depth of cut (mm)	d_2	1.000000
3	Cutting speed (m/min)	v_3	120.0000
	Feed rate (mm/rev)	f_3	2.000000
	Depth of cut (mm)	d_3	1.000000
4	Cutting speed (m/min)	v_4	210.0000
	Feed rate (mm/rev)	f_4	0.500000
	Depth of cut (mm)	d_4	0.300000
	Dimension deviation (mm)	y	0.050692

Table 4.8 Different setup costs for Model 2.2

Setup cost ($/setup)	Batch size	Total cost ($/min)
10	417	0.7097644
20	822	0.710025
30	1225	0.7102428
40	1627	0.7104499
50	2025	0.7106525
60	2422	0.7108529

To gain some insights into the problem, additional computations were carried out. The results are listed in Table 4.8. It is interesting to note that the batch size and setup cost have a linear relationship (Figure 4.4) while other parameters remain unchanged. It is also observed that the total cost is not much affected by the setup cost. This may further reveal the importance of reducing setup cost and estimating the batch size.

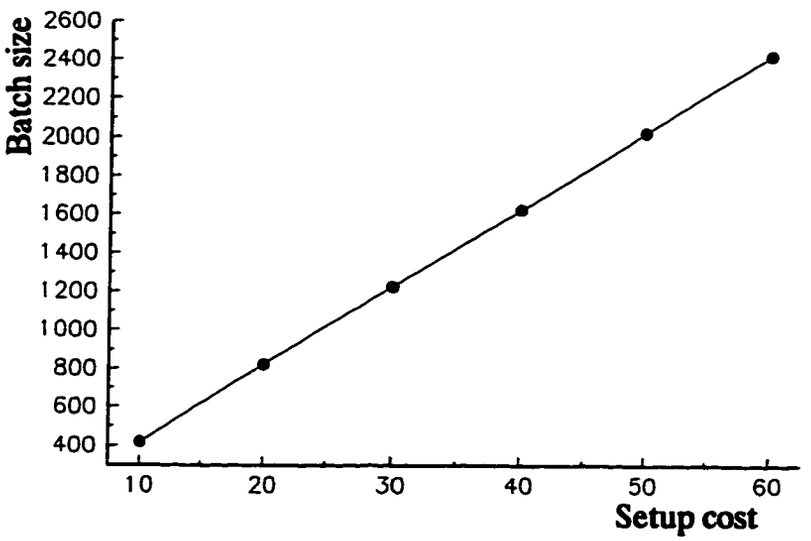

Figure 4.4 Batch size versus setup cost

Example for Model 2.3

In this case, three types of carbon steel shafts are to be produced in batches by a turning process on a stand-alone NC machine. The tolerance limits for the products are set to δ_{kJ_k} =0.0659 mm. The setup cost for each type of product is C_{sk} is \$ 40.0/setup and the inventory holding rate i_k is 20% per year (3.858×10^{-7} per min). More data are given in Table 4.9.

Table 4.9 Input data for Model 2.3

	Product 1	Product 2	Product 3
Initial stock diameter D_{k0} (mm)	120	50	90
Length to be cut L_k (mm)	300	150	200
Demand rate λ_k (parts/min)	0.18	0.12	0.15
Production rate P_{rk} (parts/min)	0.50	0.40	0.45
Surface roughness S_{Fk} (μm)	1.6	0.5	0.8
Total depth of cut d_{ktotal} (mm)	5.0	8.0	10.0

		Upper and lower bounds					
		Cutting speed (m/min)		Feed rate (mm/rev)		Depth of cut (mm)	
Product	Pass	$v_{kj}^{(l)}$	$v_{kj}^{(u)}$	$f_{kj}^{(l)}$	$f_{kj}^{(u)}$	$d_{kj}^{(l)}$	$d_{kj}^{(u)}$
1	1	90	120	0.8	2.0	1.0	5.0
	2	90	120	0.8	2.0	1.0	5.0
	3	168	210	0.13	0.5	0.4	0.9
2	1	90	120	0.8	2.0	1.0	5.0
	2	90	120	0.8	2.0	1.0	5.0
	3	168	210	0.13	0.5	0.3	0.8
3	1	90	120	0.8	2.0	1.0	5.0
	2	90	120	0.8	2.0	1.0	5.0
	3	168	210	0.13	0.5	0.35	0.85

A C code in Appendix E generates a LINGO-input-format for Model 2.3 example. The generated LINGO input is shown in Appendix F. The problem is again solved using

LINGO on a 486 PC. The computational time is 18 seconds. The total cost is $ 1.6142 per min with a production cycle time of 15445.74 min. Table 4.10 shows the output.

Table 4.10 Results for Model 2.3 example

Product	Pass	Variable		Value
1	1	Cutting speed (m/min)	v_{11}	120.0000
		Feed rate (mm/rev)	f_{11}	2.000000
		Depth of cut (mm)	d_{11}	3.600000
	2	Cutting speed (m/min)	v_{12}	120.0000
		Feed rate (mm/rev)	f_{12}	2.000000
		Depth of cut (mm)	d_{12}	1.000000
	3	Cutting speed (m/min)	v_{13}	210.0000
		Feed rate (mm/rev)	f_{13}	0.500000
		Depth of cut (mm)	d_{13}	0.400000
		Dimension deviation (mm)	y_{13}	0.051166
2	1	Cutting speed (m/min)	v_{21}	120.0000
		Feed rate (mm/rev)	f_{21}	2.000000
		Depth of cut (mm)	d_{21}	4.361455
	2	Cutting speed (m/min)	v_{22}	120.0000
		Feed rate (mm/rev)	f_{22}	2.000000
		Depth of cut (mm)	d_{22}	3.338545
	3	Cutting speed (m/min)	v_{23}	210.0000
		Feed rate (mm/rev)	f_{23}	0.500000
		Depth of cut (mm)	d_{23}	0.300000
		Dimension deviation (mm)	y_{23}	0.0510633
3	1	Cutting speed (m/min)	v_{31}	120.0000
		Feed rate (mm/rev)	f_{31}	1.810294
		Depth of cut (mm)	d_{31}	4.788111
	2	Cutting speed (m/min)	v_{32}	120.0000
		Feed rate (mm/rev)	f_{32}	1.999787
		Depth of cut (mm)	d_{32}	4.361889
	3	Cutting speed (m/min)	v_{33}	210.0000
		Feed rate (mm/rev)	f_{33}	0.500000
		Depth of cut (mm)	d_{33}	0.850000
		Dimension deviation (mm)	y_{33}	0.051013

Table 4.11 Different setup costs for Model 2.3

Setup cost ($/setup)	Cycle time (min)	Total cost ($/min)
10	4333.99	1.611269
20	8273.076	1.612344
30	11951.47	1.613317
40	15445.74	1.614216
50	18814.82	1.615056
60	22105.66	1.615848

Additional computation results (Table 4.11) show that the cycle time has a linear relationship with setup cost (Figures 4.5) when other data remain the same. This finding may be used as a guideline to estimate the production cycle time. It is again observed that the total cost is not considerably affected by setup cost.

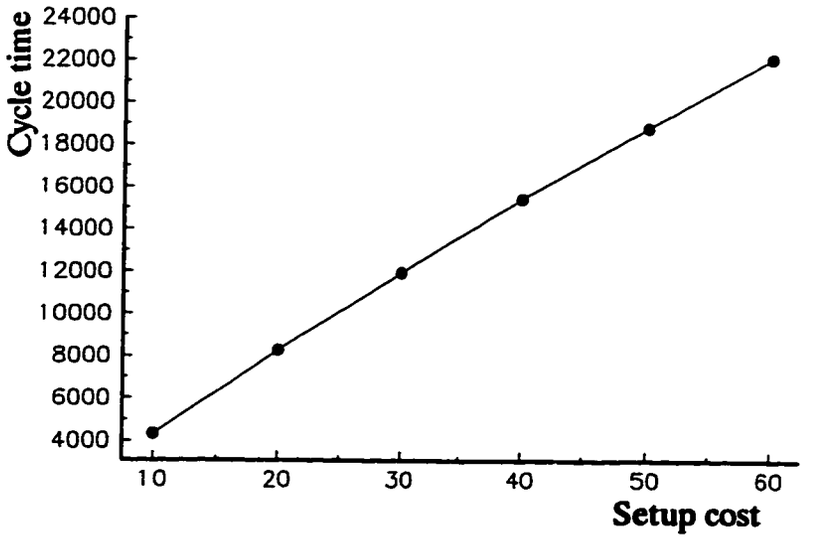

Figure 4.5 Cycle time versus setup cost

Examples for Model 2.4 and Model 2.5

Consider a low carbon steel shaft (Figure 4.6) which is to be processed on three turning machines to produce three different features. Each feature has to undergo three cutting passes. The properties of the machines are the same. The first feature is produced from the initial stock with diameter of 150 mm and length 300 mm where the total depth of cut to be removed is 5.0 mm. From the resulting diameter, the second and third features are produced. The second feature is produced by removing the total depth of cut of 8.0 mm at a length of 100 mm and the third feature is produced by removing the total depth of cut of 10.0 mm at a length 150 mm. The surface roughness and tolerance limits for all the features are 1.6 μm and 0.0659 mm respectively. Other input data are shown in Table 4.12. The data are same for both Model 2.4 and Model 2.5.

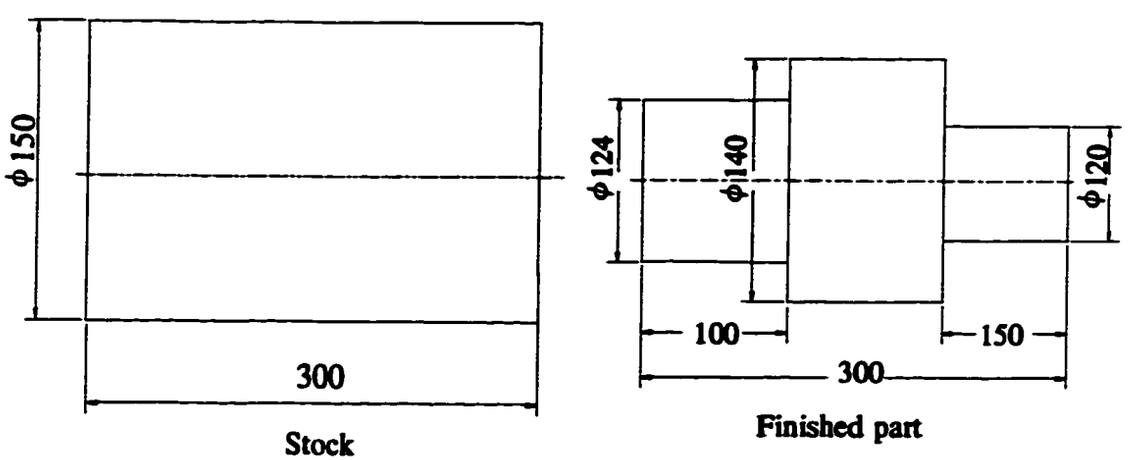

Figure 4.6 A stock and a finished part (for Model 2.4 and Model 2.5)

Table 4.12 Input data for Model 2.4 and Model 2.5

		Upper and lower bounds						
		Cutting speed (m/min)		Feed rate (mm/rev)		Depth of cut (mm)		
Feature	Pass	$v_{ijk}^{(l)}$	$v_{ijk}^{(u)}$	$f_{ijk}^{(l)}$	$f_{ijk}^{(u)}$	$d_{ijk}^{(u)}$	$d_{ij'_k k}^{(l)}$	$d_{ij'_k k}^{(u)}$
	1	90	120	0.8	2.0	5.0		
1	2	90	120	0.8	2.0	5.0		
	3	168	210	0.13	0.5		0.3	1.0
	1	90	120	0.8	2.0	5.0		
2	2	90	120	0.8	2.0	5.0		
	3	168	210	0.13	0.5		0.3	1.0
	1	90	120	0.8	2.0	5.0		
3	2	90	120	0.8	2.0	5.0		
	3	168	210	0.13	0.5		0.3	1.0

The corresponding C program for generating the LINGO model format and the generated format for Model 2.4 are respectively listed in Appendices G and H. The problem is solved in LINGO on a 486 PC. The computational time is 11.53 minutes. The increased computational time is mainly caused by the increased problem size. Workload assignment is shown in Table 4.13. For example, all three passes of feature 1 have been assigned to machine 1, the last pass of feature 2 is allocated to machine 2 and the first pass of feature 2 to machine 3. The optimum machining parameters and feature dimension deviations of all machine-pass-feature combinations are also shown in the Table. The minimum production cost is $ 13.4193 per piece with a cycle time of 3.53 min. The production times of the first, second and third machine are 2.418 min, 0.7 min and 1.247 min respectively. It clearly shows that, with the objective of minimizing production cost, the workloads on the machines are not balanced and slack times are

observed on all the machines. Therefore, if production time or due date is a more concerned, cycle time should be used as the objective. To this end, Model 2.5 will be used.

Table 4.13 Machining parameters and feature dimension deviations for Model 2.4 example

M/c	Pass	Feature	v	f	d	y
	1	1	120.0000	2.000000	4.361455	
1	2	1	120.0000	2.000000	0.3385453	
	3	1	210.0000	0.500000	0.300000	0.05104959
2	3	2	210.0000	0.500000	1.000000	0.05104512
	1	3	120.0000	2.000000	4.361455	
	1	2	120.0000	2.000000	4.361455	
3	2	2	120.0000	2.000000	2.638545	
	2	3	120.0000	1.872690	4.638545	
	3	3	210.0000	0.500000	1.000000	0.05104590

Note: v is cutting speed (m/min), f is feed rate (mm/rev), d is depth of cut (mm) and y is dimension deviation (mm).

The same set of input data are used for Model 2.5. The C program for generating the LINGO input is shown in Appendix I and the resulting LINGO format is listed in Appendix J. It took 2.33 minutes for LINGO to solve the problem on a 486 PC. The summary of optimal workload assignment, machining parameters and feature dimension deviations of all the machine-pass-feature combinations are presented in Table 4.14. The minimum cycle time now is 2.88 min which is significantly shorter than 3.53 min obtained by solving Model 2.4. However, the machines are still not well balanced. Now,

the production times for machine 1, 2, and 3 are 2.88 min, 0.524 min, and 1.395 min respectively. This means that most of workload is assigned to the first machine. To improve the balance, the models need to be refined.

Table 4.14 Machining parameters and feature dimension deviations for Model 2.5 example

M/c	Pass	Feature	v	f	d	y
	1	1	120.0000	2.000000	4.361455	
1	2	1	120.0000	2.000000	0.3385453	
	3	1	210.0000	0.500000	0.300000	0.065900
	1	2	120.0000	2.000000	2.000000	
	2	3	120.0000	2.000000	4.000000	
2	3	2	168.0034	0.500000	1.000000	0.04432998
	2	2	90.00602	1.444168	5.000000	
3	1	3	120.0000	1.728426	5.000000	
	3	3	168.0046	0.500000	1.000000	0.065900

According to Agapiou (1992), the production time equality constraint tends to reduce or eliminate the slack time at all the machines with a remarkable reduction in the cycle time. The production time inequality constraint (3.89) is changed to a production time equality constraint and Model 2.4 and Model 2.5 are solved again. The production time equality constraint specifies that the sum of production time and slack time should be equal to the cycle time. The new results are summarized in Table 4.15 and Table 4.16 respectively. The minimum production cost obtained from Model 2.4 now is $ 13.7745

per piece with a cycle time of 1.91 min. The production times for the first, second and third machine are 1.91 min, 1.17 min and 1.41 min respectively. Although the workloads on machines are not perfectly balanced, the slack and cycle time are reduced significantly. The cycle time is reduced from 3.53 min to 1.91 min which is about 45.9%. However, the production cost is increased from $ 13.4193 per piece to $ 13.7745 per piece at about 2.65%.

The minimum cycle time resulting from Model 2.5 with a production time equality constraint is 1.6265 min. The production time is the same for all machines and is 1.6265 min. This means the workloads on machines are perfectly balanced and the slack times are totally eliminated from all the machines. The cycle time reduces from 2.8824 min to 1.6265 min which is about 43.6%.

Table 4.15 Machining parameters and feature dimension deviations for Model 2.4 example using production time equality constraint.

M/c	Pass	Feature	v	f	d	y
	3	1	210.0000	0.500000	1.000000	0.05104782
1	1	3	120.0000	1.953622	4.458347	
	2	3	120.0000	1.915382	4.541653	
	1	1	120.0000	2.000082	2.000000	
2	1	2	120.0000	2.000000	4.361455	
	3	2	210.0000	0.500000	1.000000	0.05105276
	2	1	120.0000	2.000000	1.999918	
3	2	2	120.0000	2.000000	2.638545	
	3	3	210.0000	0.500000	1.000000	0.05105875

Table 4.16 Machining parameters and feature dimension deviations for Model 2.5 example using production time equality constraint.

M/c	Pass	Feature	v	f	d	y
	3	1	210.0000	0.500000	0.300000	0.065900
1	2	3	120.0000	2.000000	4.000000	
	1	1	120.0000	2.000000	4.361455	
2	1	2	120.0000	1.869154	2.580175	
	3	2	168.0000	0.500000	1.000000	0.065900
	1	3	120.0000	1.728502	5.000000	
	2	1	120.0000	2.000000	0.3385453	
3	2	2	90.00000	1.971799	4.419825	
	3	3	168.0000	0.500000	1.000000	0.065900

Chapter 5

Conclusions and Recommendations

The objectives proposed in Chapter 1 have been achieved. The accomplished work is summarized as follows:

(1) The tolerancing and machining parameters decisions have been integrated. The first-stage model has been formulated as a design problem and solved for optimum part tolerance. The optimum part tolerance obtained in the first stage has been incorporated in the formulation of manufacturing related problems in the second stage.

(2) A single-product, multi-pass model considering pass selection for single-machine turning process is formulated and solved. The model is solved for optimum cutting speed, feed rate, depth of cut, dimension deviation and number of passes. The illustrative example clearly shows that the production cost can be reduced by proper selection of cutting passes.

(3) A single-product multi-pass model considering inventory and setup costs for single-machine turning process is developed and solved to determine optimum cutting speed, feed rate, depth of cut, dimension deviation and production batch

size. Our calculation shows that batch size is linearly related to setup cost. This finding confirms the importance of setup cost reduction and may be used as a guideline in estimating the production batch size.

(4) A multi-product multi-pass model considering inventory and setup costs for single-machine turning process is proposed. The optimum cutting speed, feed rate, depth of cut, dimension deviations and production cycle time can be simultaneously determined for several products by solving this model. It is also observed that the production cycle time is proportional to setup cost.

(5) The formulation of a multi-feature multi-pass model for multi-machine turning processes with an objective of minimizing total cost has been carried out. In addition to optimizing cutting speed, feed rate, depth of cut, and feature dimension deviations, this model also optimizes the workload assignment between machines.

(6) A multi-feature multi-pass model for multi-machine turning processes with an objective of minimizing cycle time is formulated. This model is preferred if production time is an important concern.

The following are some interesting topics for future study:

(1) *Integration of tolerancing and machining decisions for multi-dependent parts.* Model 2.3 has considered multiple parts but these parts are independent. The dimension of the part components in assembly are dependent on each other. The decision of tolerance allocation between the dependent components should be integrated with the machining decisions so that the quality loss and machining cost

can be simultaneously considered.

(2) *Integration of tolerancing and machining decisions for milling operations.* This study has focused on turning operations. In reality, however, most of the parts need milling operations. Simultaneously considering tolerancing and machining decisions for milling operations will be of particular interest.

(3) *Multi-part multi-machine problem.* In Section 3.3.2, the two models for multi-machine operation deal with only a single-part problem. In practice, a multi-machine system is mostly used for a variety of parts. Therefore, the models should be extended to multi-part problems. A foreseeable problem is that the computational time could be too long to implement these models. An efficient solution method is thus desirable.

References

1. Agapiou, J.S., "Optimization of Multistage Machining Systems, Part 1: Mathematical Solution," *Journal of Engineering for Industry*, Vol. 114, No. 4, pp. 524-531, 1992.

2. Agapiou, J.S., "Optimization of Multistage Machining Systems, Part 2: The Algorithm and Applications," *Journal of Engineering for Industry*, Vol. 114, No. 4, pp. 532-538, 1992.

3. Bhattacharyya, A., Farfa-González, R., and Ham, I., " Regression Analysis for Predicting Surface Finish and its Application in the Determination of Optimum Machining Conditions," *Journal of Engineering for Industry*, Vol. 92, No.3, pp. 711-714, 1970.

4. Bhattacharyya, A., and Ham, I., "Analysis of Tool Wear, Part 1: Theoretical Models of Flank Wear," *Journal of Engineering for Industry*, pp. 790-798, 1969.

5. Brown, R.H., "On the Selection of Economical Machining Rates," *International Journal of Production Research*, Vol. 1, No. 2, pp. 1-22, 1962.

6. Carlsson, T.E., and Strand, F., "A Statistical Model for Prediction of Tool Life as a Basis for Economical Optimization of the Cutting Process," *Annals of the CIRP*, Vol. 41. No. 1, 1992.

7. Carvill, J., *Mechanical Engineer's Data Handbook*, Butterworth-Heinemann Ltd., 1993.

8. Chase, K.W., and Greenwood, W.H., "Design Issues in Mechanical Tolerance Analysis," *Manufacturing Review*, Vol. 1, No. 1, pp. 50-59, 1988.

9. Cheikh, A., and McGoldrick, F., "The Influence of Cost, Function and Process Capability on Tolerance," *International Journal of Quality and Reliability Management*, Vol. 5, No. 3, pp. 15-28, 1988.

10. Chua, M.S., Rahman, M., Wong, Y.S., and Loh, H.T., "Determination of Optimal Cutting Conditions Using Design of Experiments and Optimization Techniques," *International Journal of Machine Tools and Manufacture*, Vol. 33, No. 2, pp. 297-305, 1993.

11. DeVor, R.E., Anderson, D.L., and Zdeblick, W.J., "Tool Life Variation and its Influence on the Development of Tool Life Models," *Journal of Engineering for Industry*, Vol. 99, series B, No. 3, pp. 578-584, 1977.

12. Dong, Z., Hu, W., and Xue, D., "New Production Cost-Tolerance Models for Tolerance Synthesis," *Journal of Engineering for Industry*, Vol. 116, No. 2, pp. 199-206, 1994.

13. Dong, Z., and Soom, A., "Automatic Optimal Tolerance Design for Related Dimension Chains," *Manufacturing Review*, Vol. 3, No. 4, pp. 262-271, 1990.

14. Du, R., Zhang, B., Hungerford, W., and Pryor, T., "Tool Condition Monitoring and Compensation in Finish Turning using Optical Sensors," *Sensor Adaptive Machine, Inc. Document*, Windsor, Ontario, 1993.

15. El Gomayel, J.I., and Bregger, K.D., "On-Line Tool Wear Sensing for Turning Operations," *Journal of Engineering for Industry*, Vol. 108, No. 1, pp. 44-47, 1986.

16. Ermer, D.S., and Kromodihardjo, S., "Optimization of Multipass Turning with Constraints," *Journal of Engineering for Industry*, Vol. 103, No. 3, pp. 462-468, 1981.

17. Evans, D.H., "Statistical Tolerancing: The State of the Art," *Journal of Quality Control*, Vol. 6, No. 4, pp 188-195, 1974.

18. Feng, C. -X., and Kusiak, A., "Probabilistic Tolerance Synthesis: A Comparative study," *Proceedings of the 4th Industrial Engineering Research Conference*, pp. 357-366, 1995.

19. Fenton, R.G., and Joseph, N.D., "The Effect of the Statistical Nature of Tool-Life on the Economics of Machining," *International Journal of Machine Tool Design and Research*, Vol. 19, pp. 43-50, 1979.

20. Galante, G., and Lombardo, A., "Tool Replacement with Adaptive Control in a Non-Stationary Non-Periodic Stochastic Process," *International Journal of Production Research*, Vol. 29, No. 11, pp. 2365-2374, 1991.

21. Ghiassi, M., DeVor, R.E., Dessouky, M.I., and Kijowski, B.A., "An Application of Multiple Criteria Decision Making Principles for Planning Machining Operations," *IIE Transactions*, Vol. 16, No. 2, pp. 106-114, 1984.

22. Hati, S.K., and Rao, S.S., "Determination of Optimum Machining Conditions-Deterministic and Probabilistic Approach," *Journal of Engineering for Industry*, Vol. 98, No. 1, pp. 354-359, 1976.

23. Hitomi, K., and Ham, I., "Group Scheduling Technique for Multiproduct, Multistage Manufacturing Systems," *Journal of Engineering for Industry*, Vol. 99, pp. 759-765, 1977.

24. Iwata, K., Murotsu, Y., Iwatsubo, T., and Fujii, S., "A Probabilistic Approach to the Determination of the Optimum Cutting Conditions," *Journal of Engineering for Industry*, Vol. 94, No. 4, pp. 1099-1107, 1972.

25. Iwata, K., Murotsu, Y., and Oba, F., "Optimization of Cutting Conditions for Multi-pass Operations Considering Probabilistic Nature in Machining Process," *Journal of Engineering for Industry*, Vol. 99, series B, No. 1, pp. 210-217, 1977.

26. Jeang, A., and Yang, K., "Optimal Tool Replacement with Nondecreasing Tool Wear," *International Journal of Production Research*, Vol. 30, No. 2, pp. 299-314, 1992.

27. Kalpakjian, S., *Manufacturing Engineering and Technology*, Addison-Wesley Publishing Company, Reading Massachusetts, 1995.

28. Kannatey-Asibu, Jr., E., "A Transport-Diffusion Equation in Metal Cutting and its Application to Analysis of the Rate of Flank Wear," *Journal of Engineering Industry*, Vol. 107, No. 1, pp. 81-89, 1985.

29. Koulamas, C., "Simultaneous Determination of Optimal Machining Conditions and Tool Replacement Policies in Constrained Machining Economics Problems by Geometric Programming," *International Journal of Production Research*, Vol. 29, No. 12, pp. 2407-2421, 1991.

30. Kuljanić, E., "Effect of Stiffness on Tool Wear and New Tool Life Equation," *Journal of Engineering for Industry*, Vol. 97, series B, No. 1, pp. 939-944, 1975.

31. La Commare, U., Noto La Diega, S., and Passannanti, A., "Optimum Tool Replacement Policies with Penalty Cost for Unforeseen Tool Failure," *International Journal of Machine Tool Design and Research*, Vol. 23, No. 4, pp. 237-243, 1983.

32. Lambert, B.K., and Walvekar A.G., "Optimization of Multipass Machining Operations," *International Journal of Production Research*, Vol. 16, No. 4, pp. 259-265, 1978.

33. Lingaiah, K., *Machine Design Data Handbook*, McGraw-Hill, Inc., New York, 1994.

34. Malakooti, B., "An Interactive On-Line Multi-Objective Optimization Approach with Application to Metal Cutting Turning Operation," *International Journal of Production Research*, Vol. 29, No. 3, pp. 575-598, 1991.

35. Malakooti, B., and Deviprasad, J., "An Interactive Multiple Criteria Approach for Parameter Selection in Metal Cutting," *Operations Research*, Vol. 37, No. 5, pp. 805-818, 1989.

36. Mesquita, R., Krasteva, E., and Doytchinov, S., "Computer-Aided Selection of Optimum Machining Parameters in Multipass Turning," *International Journal of Advanced Manufacturing Technology*, Vol. 10, pp. 19-26, 1995.

37. Mukherjee, S.N., and Basu, S.K., "Multiple Regression Analysis in Evaluation of Tool Wear," *International Journal of Machine Tool Design and research*, Vol.7, pp. 15-21, 1967.

38. Nahmias, S., *Production and Operations Analysis*, Second Edition, Richard D. Irwin Inc., Homewood, IL, 1993.

39. Narang, R.V., and Fischer, G.W., "Development of a Framework to Automate Process Planning Functions and to Determine Machining Parameters," *International Journal of Production Research*, Vol. 31, No. 8, pp. 1921-1942, 1993.

40. Ngoi, B.K.A., and Chua, C.K., "A Matrix Approach to Tolerance Charting," *International Journal of Advanced Manufacturing Technology*, Vol. 8, No. 3, pp. 175-181, 1993.

41. Ngoi, B.K.A., and Tan, C.K., "Geometrics in Computer-Aided Tolerance Charting," *International Journal of Production Research*, Vol. 33, No. 3, pp. 835-868, 1995.

42. Ostwald, P.F., and Huang, J., "A Method for Optimal Tolerance Selection," *Journal of Engineering for Industry*, Vol. 99, series B, No. 3, pp. 558-565, 1977.

43. Petropoulos, P.G., "Optimal Selection of Machining Rate Variables by Geometric Programming," *International Journal of Production Research*, Vol. 11, No. 4, pp. 305-314, 1973.

44. Quesenberry, C.P., "An SPC Approach to Compensating a Tool-wear Process," *Journal of Quality Technology*, Vol. 20, No. 4, pp. 220-229, 1988.

45. Rao, S.B., "Tool Wear Monitoring Through the Dynamics of Stable Turning," *Journal of Engineering for Industry*, Vol. 108, No. 3, pp. 183-190, 1986.

46. Sekhon, G.S., "A Simulation Model of Machining Economics Incorporating Stochastic Variability of Work and Tool Properties," *International Journal of Machine Tool Design and Research*, Vol. 23, No. 1, pp. 61-70, 1983.

47. Speckhart, F.H., "Calculation of Tolerance Based on a Minimum Cost Approach," *Journal of Engineering for Industry*, Vol. 94, No. 2, pp. 447-453, 1972.

48. Spotts, M.F., "Allocation of Tolerances to Minimize Cost of Assembly," *Journal of Engineering for Industry*, Vol. 95, No. 3, pp. 762-764, 1973.

49. Sutherland, G.H., and Roth, B., "Mechanism Design: Accounting for Manufacturing Tolerances and Costs in Function Generating Problems," *Journal of Engineering for Industry*, Vol. 98, pp. 283-286, 1975.

50. Taguchi, G., Elsayed, E.A., and Hsiang, T., *Quality Engineering in Production Systems*, McGraw-Hill Book Company, New York, 1989.

51. Tan, F.P., and Creese, R.C., "A Generalized Multi-pass Machining Model for Machining Parameter Selection in Turning," *International Journal of Production Research*, Vol. 33, No. 5, pp. 1467-1487, 1995.

52. Taylor, F.W., "On the Art of Cutting Metals," *Transactions of the American Society of Mechanical Engineers*, Vol. 28, pp. 31-279, 1907.

53. Trappey, J.-F.C., Liu, C.R., and Chang, T.-C., "Fuzzy Nonlinear Programming: Theory and Application in Manufacturing," *International Journal of Production Research*, Vol. 26, No. 5, pp. 975-985, 1988.

54. Usui, E., Shirakashi, T., and Kitagawa, T., "Analytical Prediction of Cutting Tool Wear," *Wear*, Vol. 100, pp. 129-151, 1984.

55. Vajpayee, S.K., *Principles of Computer-Integrated Manufacturing*, Prentice Hall, New Jersey, 1995.

56. Walvekar, A.G., and Lambert, B.K.,"An Application of Geometric Programming to Machining Variable Selection, "*International Journal of Production Research*, Vol. 8, No. 3, pp. 241-245, 1970.

57. Wang, H.-P., and Wysk R.A., "An Expert System for Machining Data Selection," *Computers and Industrial Engineering*, Vol. 10, No. 2, pp. 99-107, 1986.

58. Wang, J., "Multiple-Objective Optimization of Machining Operations Based on Neural Networks," *International Journal of Advanced Manufacturing Technology*, Vol. 7, pp. 1-9, 1992.

59. Wang, D.X., and Zuo, M.J., "Analysis of Machining Tool Adjustment and Replacement," *Proceedings of the 3rd Industrial Engineering Research Conference*, pp. 350-354, 1994.

60. Wang, D.X., Zuo, M.J., Qi, K.Z., and Liang M., "On-line Tool Adjustment With Adaptive Tool Wear Function Identification," *International Journal of Production Research* (accepted), 1996.

61. Yeo, S.H., Rahman, M., and Wong, Y.S., "A Tandem Approach to Selection of Machinability Data," *International Journal of Advanced Manufacturing Technology*, Vol. 10, pp. 79-86, 1995.

62. Zhang, C., and Wang, H.-P., "Integrated Tolerance Optimization with Simulated Annealing," *International Journal of Advanced Manufacturing Technology*, Vol. 8, pp. 167-174, 1993.

63. Zhou, C., and Wysk, R.A., "An Integrated System for Selecting Optimum Cutting Speeds and Tool Replacement Times," *International Journal of Machine Tools and Manufacture*, Vol. 32, No. 5, pp. 695-707, 1992.

64. Zhou, C., and Wysk, R.A., "Tool Status Recording and its Use in Probabilistic Optimization," *Journal of Engineering for Industry*, Vol. 114, No. 4, pp. 494-499, 1992.

65. Zohdi, M.E., and Biles, W.E., "Production Processes and Equipment," *Mechanical Engineer's Handbook*, pp. 865-886, 1986.

Appendix A

C code for generating LINGO-format-input for Model 2.1

```
/*
This program is used to generate the LINGO-format-input model
*/

# include <stdio.h>
# include <conio.h>
# include <stdlib.h>
# include <math.h>

# define pi 3.14159
# define c 6120

/*
Q = number of passes
*/

int k,Q,R;

float alpha,beta,z1,z2,z3,z4,z5,z6,kP,cc;

float  D0=90.0,
       L=200.0,
       co=3.0,
       kT=1570000.00,
       alphaT=1.70,
       betaT=1.55,
       gammaT=1.22,
       ct=5.5,
       tr=0.5,
       w=0.1,
       ca=3.0,
       ta=0.2,
       A=1.0,
       dtotal=5.0,
```

```
        delta=0.0659,
        betaF=1.18,
        gammaF=1.26,
        kF=1.38,
        Fc_u=20.0,
        eta=0.8,
        Pc_u=2.0,
        kS=1.17,
        alphaS=-0.25,
        betaS=0.72,
        gammaS=0.23,
        SF_u=1.6;

static float vj_l[5] = {0.0,90.0,90.0,90.0,168.0};
static float vj_u[5] = {0.0,120.0,120.0,120.0,210.0};
static float fj_l[5] = {0.0,0.8,0.8,0.8,0.13};
static float fj_u[5] = {0.0,2.0,2.0,2.0,0.50};
static float dj_l[5] = {0.0,0.0,0.0,0.0,0.3};
static float dj_u[5] = {0.0,5.0,5.0,5.0,1.0};

FILE *mus;

main()

{

clrscr();

Q=4;
R=1000;
alpha=alphaT-1;
beta=betaT-1;
z1=pi*D0*L*co/R;
z2=pi*D0*L/(R*kT);
z3=pi*L*co/R;
z4=pi*L/kT;
z5=w*ca*ta;
z6=w*A/(delta*delta);
kP=kF/(eta*c);
cc=ct+co*tr;

/*
The following statement specifies the name for
the generated file
```

```
*/

mus = fopen("mod21","w");

/*
The following section generates the objective function
*/

fprintf(mus,"MODEL\n");
fprintf(mus,"min=");

for (k=1;k<=Q-3;k++)
{
    fprintf(mus,"%f*v%d^(-1)*f%d^(-1)*p%d+%f*v%d^%f\n",
            z1,k,k,k,z2,k,alpha);
    fprintf(mus,"*f%d^%f*d%d^%f*%f*p%d+%f*(%f-2*d%d)\n",
            k,beta,k,gammaT,cc,k,z3,D0,k);
}

for (k=2;k<=Q-2;k++)
{
   fprintf(mus,"*v%d^(-1)*f%d^(-1)*p%d",k,k,k);
}
   fprintf(mus,"+%f*10^(-3)",z4);

for (k=1;k<=Q-3;k++)
{
   fprintf(mus,"*(%f-2*d%d)\n",D0,k);
}

for (k=2;k<=Q-2;k++)
{
   fprintf(mus,"*v%d^%f*f%d^%f*d%d^%f*%f*p%d\n",k,alpha,
       k,beta,k,gammaT,cc,k);
}

for (k=1;k<=Q-3;k++)
{
   fprintf(mus,"+%f*(%f-2*d%d",z3,D0,k);
}
for (k=2;k<=Q-2;k++)
{
   fprintf(mus,"-2*d%d)",k);
}
```

```
for (k=3;k<=Q-1;k++)
{
    fprintf(mus,"*v%d^(-1)*f%d^(-1)*p%d+%f*10^(-3)\n",k,k,k,z4);
}

for (k=1;k<=Q-3;k++)
{
    fprintf(mus,"*(%f-2*d%d",D0,k);
}

for (k=2;k<=Q-2;k++)
{
    fprintf(mus,"-2*d%d)",k);
}

for (k=3;k<=Q-1;k++)
{
    fprintf(mus,"*v%d^%f*f%d^%f*d%d^%f*%f*p%d\n",k,alpha,
    k,beta,k,gammaT,cc,k);
}

for (k=1;k<=Q-3;k++)
{
    fprintf(mus,"+%f*(%f-2*d%d",z3,D0,k);
}

for (k=2;k<=Q-2;k++)
{
    fprintf(mus,"-2*d%d",k);
}

for (k=3;k<=Q-1;k++)
{
    fprintf(mus,"-2*d%d)",k);
}

for (k=4;k<=Q;k++)
{
    fprintf(mus,"*v%d^(-1)*f%d^(-1)*p%d+%f*10^(-3)\n",k,k,k,z4);
}

for (k=1;k<=Q-3;k++)
{
    fprintf(mus,"*(%f-2*d%d",D0,k);
```

```
 }

 for (k=2;k<=Q-2;k++)
 {
    fprintf(mus,"-2*d%d",k);
 }

 for (k=3;k<=Q-1;k++)
 {
    fprintf(mus,"-2*d%d)",k);
 }

 for (k=4;k<=Q;k++)
 {
    fprintf(mus,"*v%d^%f\n",k,alpha);
    fprintf(mus,"*f%d^%f*d%d^%f*p%d*(%f+%f/y+%f*y);\n",
    k,beta,k,gammaT,k,cc,z5,z6);
 }

 /*
 This section generates cutting speed, feed rate, cutting
 force and cutting power constraints
 */

 for (k=1;k<=Q;k++)
 {
    fprintf(mus,"v%d>=%f;\n",k,vj_l[k],k);
    fprintf(mus,"v%d<=%f;\n",k,vj_u[k],k);
    fprintf(mus,"f%d>=%f;\n",k,fj_l[k],k);
    fprintf(mus,"f%d<=%f;\n",k,fj_u[k],k);
    fprintf(mus,"%f*f%d^%f*d%d^%f<=%f;\n",kF,k,betaF,
                k,gammaF,Fc_u);
    fprintf(mus,"%f*v%d*f%d^%f*d%d^%f<=%f;\n",kP,k,
                k,betaF,k,gammaF,Pc_u);
 }

 /*
 This section generates depth of cut constraints
 */

 for (k=1;k<=Q-1;k++)
 {
    fprintf(mus,"d%d<=%f;\n",k,dj_u[k]);
 }
```

```
for (k=4;k<=Q;k++)
{
    fprintf(mus,"d%d>=%f;\n",k,dj_l[k]);
    fprintf(mus,"d%d<=%f;\n",k,dj_u[k]);
}

/*
This section generates continuous variables for selecting passes
*/

for (k=1;k<=Q-1;k++)
{
    fprintf(mus,"p%d>=0;\n",k);
    fprintf(mus,"p%d<=1;\n",k);
}

for (k=4;k<=Q;k++)
{
    fprintf(mus,"p%d=1;\n",k);
}

/*
This section generates constraints for sum of depth of cut
*/

for (k=1;k<=Q-1;k++)
{
    fprintf(mus,"d%d+",k);
}

for (k=4;k<=Q;k++)
{
    fprintf(mus,"d%d=%f;\n",k,dtotal);
}

for (k=1;k<=Q-1;k++)
{
    fprintf(mus,"p%d*d%d+",k,k);
}

for (k=4;k<=Q;k++)
{
    fprintf(mus,"p%d*d%d=%f;\n",k,k,dtotal);
}
```

```
for (k=1;k<=Q-1;k++)
{
    fprintf(mus,"d%d/%f-p%d<=%f;\n",k,dtotal,k);
}

/*
This section generates part dimension deviation constraint
*/

    fprintf(mus,"y>=0;\n");
    fprintf(mus,"y<=%f;\n",delta);

/*
This section generates surface finish constraint
*/

for (k=4;k<=Q;k++)
{
    fprintf(mus,"%f*v%d^%f*f%d^%f*d%d^%f<=%f;\n",kS,k,
    alphaS,k,betaS,k,gammaS,SF_u);
}
fprintf(mus,"END\n");
fprintf(mus,"BAT\n");
return 0;
}
```

Appendix B

LINGO-format-input for the example problem of Model 2.1

```
MODEL
min=169.645859*v1^(-1)*f1^(-1)*p1+0.000036*v1^0.700000
*f1^0.550000*d1^1.220000*7.000000*p1+1.884954*(90.000000-2*d1)
*v2^(-1)*f2^(-1)*p2+0.000400*10^(-3)*(90.000000-2*d1)
*v2^0.700000*f2^0.550000*d2^1.220000*7.000000*p2
+1.884954*(90.000000-2*d1-2*d2)*v3^(-1)*f3^(-1)*p3+0.000400*10^(-3)
*(90.000000-2*d1-2*d2)*v3^0.700000*f3^0.550000*d3^1.220000*7.000000*p3
+1.884954*(90.000000-2*d1-2*d2-2*d3)*v4^(-1)*f4^(-1)*p4+0.000400*10^(-3)
*(90.000000-2*d1-2*d2-2*d3)*v4^0.700000
*f4^0.550000*d4^1.220000*p4*(7.000000+0.060000/y+23.026567*y);
v1>=90.000000;
v1<=120.000000;
f1>=0.800000;
f1<=2.000000;
1.380000*f1^1.180000*d1^1.260000<=20.000000;
0.000282*v1*f1^1.180000*d1^1.260000<=2.000000;
v2>=90.000000;
v2<=120.000000;
f2>=0.800000;
f2<=2.000000;
1.380000*f2^1.180000*d2^1.260000<=20.000000;
0.000282*v2*f2^1.180000*d2^1.260000<=2.000000;
v3>=90.000000;
v3<=120.000000;
f3>=0.800000;
f3<=2.000000;
1.380000*f3^1.180000*d3^1.260000<=20.000000;
0.000282*v3*f3^1.180000*d3^1.260000<=2.000000;
v4>=168.000000;
v4<=210.000000;
f4>=0.130000;
f4<=0.500000;
1.380000*f4^1.180000*d4^1.260000<=20.000000;
0.000282*v4*f4^1.180000*d4^1.260000<=2.000000;
d1<=5.000000;
```

```
d2<=5.000000;
d3<=5.000000;
d4>=0.300000;
d4<=1.000000;
p1>=0;
p1<=1;
p2>=0;
p2<=1;
p3>=0;
p3<=1;
p4=1;
d1+d2+d3+d4=5.000000;
p1*d1+p2*d2+p3*d3+p4*d4=5.000000;
d1/5.000000-p1<=0.000000;
d2/5.000000-p2<=0.000000;
d3/5.000000-p3<=0.000000;
y>=0;
y<=0.065900;
1.170000*v4^-0.250000*f4^0.720000*d4^0.230000<=1.600000;
END
BAT
```

Appendix C

C code for generating LINGO-format-input for Model 2.2

```
/*
This program is used to generate the LINGO-format-input model
*/

# include <stdio.h>
# include <conio.h>
# include <stdlib.h>
# include <math.h>

# define pi 3.14159
# define c 6120

/*
Q = number of passes
*/

int k,Q,R;

float alpha,beta,z1,z2,z3,z4,z5,z6,z7,z8,z9,z10,kP,cc,wt;

float  D0=90.0,
       L=200.0,
       co=3.0,
       kT=1570000.00,
       alphaT=1.70,
       betaT=1.55,
       gammaT=1.22,
       ct=5.5,
       tr=0.5,
       w=0.1,
       ca=3.0,
       ta=0.2,
       A=1.0,
       dtotal=5.0,
       delta=0.0659,
```

```
        betaF=1.18,
        gammaF=1.26,
        kF=1.38,
        Fc_u=20.0,
        eta=0.8,
        Pc_u=2.0,
        kS=1.17,
        alphaS=-0.25,
        betaS=0.72,
        gammaS=0.23,
        SF_u=1.6,
        cs=40.0,
        ih=3.858,
        lambda=0.2,
        rate=0.4;
static float vj_l[5] = {0.0,90.0,90.0,90.0,168.0};
static float vj_u[5] = {0.0,120.0,120.0,120.0,210.0};
static float fj_l[5] = {0.0,0.8,0.8,0.8,0.13};
static float fj_u[5] = {0.0,2.0,2.0,2.0,0.5};
static float dj_l[5] = {0.0,1.0,1.0,1.0,0.3};
static float dj_u[5] = {0.0,5.0,5.0,5.0,0.9};

FILE *mus;

main()

{

clrscr();

Q=4;
R=1000;
alpha=alphaT-1;
beta=betaT-1;
z1=pi*D0*L*co/R;
z2=pi*D0*L/(R*kT);
z3=pi*L*co/R;
z4=pi*L/kT;
z5=w*ca*ta;
z6=w*A/(delta*delta);
z7=pi*D0*L/R;
z9=pi*L/R;
kP=kF/(eta*c);
cc=ct+co*tr;
```

```
wt=w*ta;

 /*
The following statement specifies the name for
the generated file
*/

mus = fopen("mod22","w");

/*
The following section generates the objective function
*/

fprintf(mus,"MODEL\n");
fprintf(mus,"min=");

for (k=1;k<=Q-3;k++)
{
    fprintf(mus,"(%f*v%d^(-1)*f%d^(-1)+%f*v%d^%f\n",
            z1,k,k,z2,k,alpha);
    fprintf(mus,"*f%d^%f*d%d^%f*%f+%f*(%f-2*d%d)\n",
            k,beta,k,gammaT,cc,z3,D0,k);
}

for (k=2;k<=Q-2;k++)
{
   fprintf(mus,"*v%d^(-1)*f%d^(-1)",k,k);
}
   fprintf(mus,"+%f*10^(-3)",z4);

for (k=1;k<=Q-3;k++)
{
   fprintf(mus,"*(%f-2*d%d)\n",D0,k);
}

for (k=2;k<=Q-2;k++)
{
   fprintf(mus,"*v%d^%f*f%d^%f*d%d^%f*%f\n",k,alpha,
       k,beta,k,gammaT,cc);
}

for (k=1;k<=Q-3;k++)
{
   fprintf(mus,"+%f*(%f-2*d%d",z3,D0,k);
```

```
}

for (k=2;k<=Q-2;k++)
{
    fprintf(mus,"-2*d%d)",k);
}

for (k=3;k<=Q-1;k++)
{
    fprintf(mus,"*v%d^(-1)*f%d^(-1)+%f*10^(-3)\n",k,k,z4);
}

for (k=1;k<=Q-3;k++)
{
    fprintf(mus,"*(%f-2*d%d",D0,k);
}

for (k=2;k<=Q-2;k++)
{
    fprintf(mus,"-2*d%d)",k);
}

for (k=3;k<=Q-1;k++)
{
    fprintf(mus,"*v%d^%f*f%d^%f*d%d^%f*%f\n",k,alpha,
        k,beta,k,gammaT,cc);
}

for (k=1;k<=Q-3;k++)
{
    fprintf(mus,"+%f*(%f-2*d%d",z3,D0,k);
}

for (k=2;k<=Q-2;k++)
{
    fprintf(mus,"-2*d%d",k);
}

for (k=3;k<=Q-1;k++)
{
    fprintf(mus,"-2*d%d)",k);
}

for (k=4;k<=Q;k++)
```

```
{
    fprintf(mus,"*v%d^(-1)*f%d^(-1)+%f*10^(-3)\n",k,k,z4);
}

for (k=1;k<=Q-3;k++)
{
    fprintf(mus,"*(%f-2*d%d",D0,k);
}

for (k=2;k<=Q-2;k++)
{
    fprintf(mus,"-2*d%d",k);
}

for (k=3;k<=Q-1;k++)
{
    fprintf(mus,"-2*d%d)",k);
}

for (k=4;k<=Q;k++)
{
    fprintf(mus,"*v%d^%f\n",k,alpha);
    fprintf(mus,"*f%d^%f*d%d^%f*(%f+%f/y+%f*y))*(%f+\n",
    k,beta,k,gammaT,cc,z5,z6,lambda);
}

for (k=1;k<=Q-3;k++)
{
    fprintf(mus,"%f*10^(-7)*Qs/2*(1-%f*(%f*v%d^(-1)*f%d^(-1)+%f*v%d^%f\n",
    ih,lambda,z7,k,k,z2,k,alpha);
    fprintf(mus,"*f%d^%f*d%d^%f*%f+%f*(%f-2*d%d)\n",k,beta,
    k,gammaT,tr,z3,D0,k);
}

for (k=2;k<=Q-2;k++)
{
    fprintf(mus,"*v%d^(-1)*f%d^(-1)",k,k);
}
    fprintf(mus,"+%f*10^(-3)",z4);

for (k=1;k<=Q-3;k++)
{
    fprintf(mus,"*(%f-2*d%d)\n",D0,k);
}
```

```
for (k=2;k<=Q-2;k++)
{
    fprintf(mus,"*v%d^%f*f%d^%f*d%d^%f*%f\n",k,alpha,k,beta,
    k,gammaT,tr);
}

for (k=1;k<=Q-3;k++)
{
    fprintf(mus,"+%f*(%f-2*d%d",z9,D0,k);
}

for (k=2;k<=Q-2;k++)
{
    fprintf(mus,"-2*d%d)",k);
}

for (k=3;k<=Q-1;k++)
{
    fprintf(mus,"*v%d^(-1)*f%d^(-1)+%f*10^(-3)\n",k,k,z4);
}

for (k=1;k<=Q-3;k++)
{
    fprintf(mus,"*(%f-2*d%d",D0,k);
}

for (k=2;k<=Q-2;k++)
{
    fprintf(mus,"-2*d%d)",k);
}

for (k=3;k<=Q-1;k++)
{
    fprintf(mus,"*v%d^%f*f%d^%f*d%d^%f*%f\n",k,alpha,k,beta,
    k,gammaT,tr);
}

for (k=1;k<=Q-3;k++)
{
    fprintf(mus,"+%f*(%f-2*d%d",z9,D0,k);
}

for (k=2;k<=Q-2;k++)
{
```

```
    fprintf(mus,"-2*d%d",k);
}

for (k=3;k<=Q-1;k++)
{
    fprintf(mus,"-2*d%d)",k);
}

for (k=4;k<=Q;k++)
{
    fprintf(mus,"*v%d^(-1)*f%d^(-1)+%f*10^(-3)\n",k,k,z4);
}

for (k=1;k<=Q-3;k++)
{
    fprintf(mus,"*(%f-2*d%d",D0,k);
}

for (k=2;k<=Q-2;k++)
{
    fprintf(mus,"-2*d%d",k);
}

for (k=3;k<=Q-1;k++)
{
    fprintf(mus,"-2*d%d)",k);
}

for (k=4;k<=Q;k++)
{
    fprintf(mus,"*v%d^%f\n",k,alpha);
    fprintf(mus,"*f%d^%f*d%d^%f*(%f+%f/y))))+%f*%f/Qs;\n",k,beta,
    k,gammaT,tr,wt,lambda,cs);
}

/*
This section generates cutting speed, feed rate, cutting
force and cutting power constraints
*/

for (k=1;k<=Q;k++)
{
    fprintf(mus,"v%d>=%f;\n",k,vj_l[k]);
    fprintf(mus,"v%d<=%f;\n",k,vj_u[k]);
```

```
    fprintf(mus,"f%d>=%f;\n",k,fj_l[k]);
    fprintf(mus,"f%d<=%f;\n",k,fj_u[k]);
    fprintf(mus,"%f*f%d^%f*d%d^%f<=%f;\n",kF,k,betaF,
            k,gammaF,Fc_u);
    fprintf(mus,"%f*v%d*f%d^%f*d%d^%f<=%f;\n",kP,k,
            k,betaF,k,gammaF,Pc_u);

}

/*
This section generates constraints for depth of cut and sum of depth of cut
*/

for (k=1;k<=Q;k++)
{
    fprintf(mus,"d%d>=%f;\n",k,dj_l[k]);
    fprintf(mus,"d%d<=%f;\n",k,dj_u[k]);
}

for (k=1;k<=Q-1;k++)
{
    fprintf(mus,"d%d+",k);
}

for (k=4;k<=Q;k++)
{
    fprintf(mus,"d%d=%f;\n",k,dtotal);
}

/*
This section generates dimension deviation constraint
*/

    fprintf(mus,"y>=0;\n");
    fprintf(mus,"y<=%f;\n",delta);

/*
This section generates surface finish constraint
*/

for (k=4;k<=Q;k++)
{
    fprintf(mus,"%f*v%d^%f*f%d^%f*d%d^%f<=%f;\n",kS,k,
    alphaS,k,betaS,k,gammaS,SF_u);
```

```
}

/*
This section generates the constraint such that the
demand rate is less than the production rate
*/

for (k=1;k<=Q-3;k++)
{
    fprintf(mus,"1/(%f*v%d^(-1)*f%d^(-1)+%f*v%d^%f\n",z7,k,k,
    z2,k,alpha);
    fprintf(mus,"*f%d^%f*d%d^%f*%f+%f*(%f-2*d%d)\n",k,beta,k,
    gammaT,tr,z3,D0,k);
}
for (k=2;k<=Q-2;k++)
{
    fprintf(mus,"*v%d^(-1)*f%d^(-1)",k,k);
}
    fprintf(mus,"+%f*10^(-3)",z4);

for (k=1;k<=Q-3;k++)
{
    fprintf(mus,"*(%f-2*d%d)\n",D0,k);
}
for (k=2;k<=Q-2;k++)
{
    fprintf(mus,"*v%d^%f*f%d^%f*d%d^%f*%f\n",k,alpha,k,beta,
    k,gammaT,tr);
}

for (k=1;k<=Q-3;k++)
{
    fprintf(mus,"+%f*(%f-2*d%d",z9,D0,k);
}

for (k=2;k<=Q-2;k++)
{
    fprintf(mus,"-2*d%d)",k);
}

for (k=3;k<=Q-1;k++)
{
    fprintf(mus,"*v%d^(-1)*f%d^(-1)+%f*10^(-3)\n",k,k,z4);
}
```

```
for (k=1;k<=Q-3;k++)
{
    fprintf(mus,"*(%f-2*d%d",D0,k);
}

for (k=2;k<=Q-2;k++)
{
    fprintf(mus,"-2*d%d)",k);
}

for (k=3;k<=Q-1;k++)
{
    fprintf(mus,"*v%d^%f*f%d^%f*d%d^%f*%f\n",k,alpha,k,beta,
    k,gammaT,tr);
}
for (k=1;k<=Q-3;k++)
{
    fprintf(mus,"+%f*(%f-2*d%d",z9,D0,k);
}

for (k=2;k<=Q-2;k++)
{
    fprintf(mus,"-2*d%d",k);
}

for (k=3;k<=Q-1;k++)
{
    fprintf(mus,"-2*d%d)",k);
}

for (k=4;k<=Q;k++)
{
    fprintf(mus,"*v%d^(-1)*f%d^(-1)+%f*10^(-3)\n",k,k,z4);
}
for (k=1;k<=Q-3;k++)
{
    fprintf(mus,"*(%f-2*d%d",D0,k);
}

for (k=2;k<=Q-2;k++)
{
    fprintf(mus,"-2*d%d",k);
}
```

```
for (k=3;k<=Q-1;k++)
{
    fprintf(mus,"-2*d%d)",k);
}
for (k=4;k<=Q;k++)
{
    fprintf(mus,"*v%d^%f\n",k,alpha);
    fprintf(mus,"*f%d^%f*d%d^%f*(%f+%f/y))>=%f;\n",k,beta,k,gammaT,
    tr,wt,rate);
}
fprintf(mus,"END\n");
fprintf(mus,"BAT\n");
return 0;
}
```

Appendix D

LINGO-format-input for the example problem of Model 2.2

```
MODEL
min=(169.645859*v1^(-1)*f1^(-1)+0.000036*v1^0.700000
*f1^0.550000*d1^1.220000*7.000000+1.884954*(90.000000-2*d1)
*v2^(-1)*f2^(-1)+0.000400*10^(-3)*(90.000000-2*d1)
*v2^0.700000*f2^0.550000*d2^1.220000*7.000000
+1.884954*(90.000000-2*d1-2*d2)*v3^(-1)*f3^(-1)+0.000400*10^(-3)
*(90.000000-2*d1-2*d2)*v3^0.700000*f3^0.550000*d3^1.220000*7.000000
+1.884954*(90.000000-2*d1-2*d2-2*d3)*v4^(-1)*f4^(-1)+0.000400*10^(-3)
*(90.000000-2*d1-2*d2-2*d3)*v4^0.700000
*f4^0.550000*d4^1.220000*(7.000000+0.060000/y+23.026567*y))*(0.200000+
3.858000*10^(-7)*Qs/2*(1-0.200000*(56.548618*v1^(-1)*f1^(-1)+0.000036*v1^0.700000
*f1^0.550000*d1^1.220000*0.500000+1.884954*(90.000000-2*d1)
*v2^(-1)*f2^(-1)+0.000400*10^(-3)*(90.000000-2*d1)
*v2^0.700000*f2^0.550000*d2^1.220000*0.500000
+0.628318*(90.000000-2*d1-2*d2)*v3^(-1)*f3^(-1)+0.000400*10^(-3)
*(90.000000-2*d1-2*d2)*v3^0.700000*f3^0.550000*d3^1.220000*0.500000
+0.628318*(90.000000-2*d1-2*d2-2*d3)*v4^(-1)*f4^(-1)+0.000400*10^(-3)
*(90.000000-2*d1-2*d2-2*d3)*v4^0.700000
*f4^0.550000*d4^1.220000*(0.500000+0.020000/y))))+0.200000*40.000000/Qs;
v1>=90.000000;
v1<=120.000000;
f1>=0.800000;
f1<=2.000000;
1.380000*f1^1.180000*d1^1.260000<=20.000000;
0.000282*v1*f1^1.180000*d1^1.260000<=2.000000;
v2>=90.000000;
v2<=120.000000;
f2>=0.800000;
f2<=2.000000;
1.380000*f2^1.180000*d2^1.260000<=20.000000;
0.000282*v2*f2^1.180000*d2^1.260000<=2.000000;
v3>=90.000000;
v3<=120.000000;
f3>=0.800000;
f3<=2.000000;
```

```
1.380000*f3^1.180000*d3^1.260000<=20.000000;
0.000282*v3*f3^1.180000*d3^1.260000<=2.000000;
v4>=168.000000;
v4<=210.000000;
f4>=0.130000;
f4<=0.500000;
1.380000*f4^1.180000*d4^1.260000<=20.000000;
0.000282*v4*f4^1.180000*d4^1.260000<=2.000000;
d1>=1.000000;
d1<=5.000000;
d2>=1.000000;
d2<=5.000000;
d3>=1.000000;
d3<=5.000000;
d4>=0.300000;
d4<=0.900000;
d1+d2+d3+d4=5.000000;
y>=0;
y<=0.065900;
1.170000*v4^-0.250000*f4^0.720000*d4^0.230000<=1.600000;
1/(56.548618*v1^(-1)*f1^(-1)+0.000036*v1^0.700000
*f1^0.550000*d1^1.220000*0.500000+1.884954*(90.000000-2*d1)
*v2^(-1)*f2^(-1)+0.000400*10^(-3)*(90.000000-2*d1)
*v2^0.700000*f2^0.550000*d2^1.220000*0.500000
+0.628318*(90.000000-2*d1-2*d2)*v3^(-1)*f3^(-1)+0.000400*10^(-3)
*(90.000000-2*d1-2*d2)*v3^0.700000*f3^0.550000*d3^1.220000*0.500000
+0.628318*(90.000000-2*d1-2*d2-2*d3)*v4^(-1)*f4^(-1)+0.000400*10^(-3)
*(90.000000-2*d1-2*d2-2*d3)*v4^0.700000
*f4^0.550000*d4^1.220000*(0.500000+0.020000/y))>=0.400000;
END
BAT
```

Appendix E

C code for generating LINGO-format-input for Model 2.3

```
/*
This program is used to generate the LINGO-format-input model
*/

# include <stdio.h>
# include <conio.h>
# include <stdlib.h>
# include <math.h>

# define pi 3.14159
# define c 6120

/*
P = number of types of products
Q = number of passes
*/

int i,k,Q,P,R;

float alpha,beta,z1,z2,z3,z4,z5,z6,z7,u1,u2,u3,
      u4,u5,u6,u7,v1,v2,v3,v4,v5,v6,v7,wt,ww,kP,cc;

float  co=3.0,
       kT=1570000.00,
       alphaT=1.70,
       betaT=1.55,
       gammaT=1.22,
       ct=5.5,
       tr=0.5,
       w=0.1,
       ca=3.0,
       ta=0.2,
       A=1.0,
       betaF=1.18,
```

```
        gammaF=1.26,
        kF=1.38,
        Fc_u=20.0,
        eta=0.8,
        Pc_u=2.0,
        kS=1.17,
        alphaS=-0.25,
        betaS=0.72,
        gammaS=0.23,
        cs=40.0,
        ih=3.858;

static float D0[4] = {0.0,120.0,50.0,90.0};
static float L[4] = {0.0,300.0,150.0,200.0};
static float delta[4] = {0.0,0.0659,0.0659,0.0659};
static float lambda[4] = {0.0,0.18,0.12,0.15};
static float rate[4] = {0.0,0.50,0.40,0.45};
static float dtotal[4] = {0.0,5.0,8.0,10.0};
static float SF_u[4] = {0.0,1.6,0.5,0.8};
static float v_l[4] = {0.0,90.0,90.0,168.0};
static float v_u[4] = {0.0,120.0,120.0,210.0};
static float f_l[4] = {0.0,0.8,0.8,0.13};
static float f_u[4] = {0.0,2.0,2.0,0.50};
static float d_l[4][4] = {
                          {0.0,0.0,0.0,0.0},
                          {0.0,1.0,1.0,0.4},
                          {0.0,1.0,1.0,0.3},
                          {0.0,1.0,1.0,0.35}
                        };

static float d_u[4][4] = {
                          {0.0,0.0,0.0,0.0},
                          {0.0,5.0,5.0,0.9},
                          {0.0,5.0,5.0,0.8},
                          {0.0,5.0,5.0,0.85}
                        };

FILE *mus;

main()

{

clrscr();
```

```
P=3;Q=3;
R=1000;
alpha=alphaT-1;
beta=betaT-1;
z1=pi*D0[1]*L[1]*co/R;
z2=pi*D0[1]*L[1]/(R*kT);
z3=pi*L[1]*co/R;
z4=pi*L[1]/kT;
z5=w*A/(delta[1]*delta[1]);
z6=pi*D0[1]*L[1]/R;
z7=pi*L[1]/R;
u1=pi*D0[2]*L[2]*co/R;
u2=pi*D0[2]*L[2]/(R*kT);
u3=pi*L[2]*co/R;
u4=pi*L[2]/kT;
u5=w*A/(delta[2]*delta[2]);
u6=pi*D0[2]*L[2]/R;
u7=pi*L[2]/R;
v1=pi*D0[3]*L[3]*co/R;
v2=pi*D0[3]*L[3]/(R*kT);
v3=pi*L[3]*co/R;
v4=pi*L[3]/kT;
v5=w*A/(delta[3]*delta[3]);
v6=pi*D0[3]*L[3]/R;
v7=pi*L[3]/R;
wt=w*ta;
ww=w*ca*ta;
kP=kF/(eta*c);
cc=ct+co*tr;

 /*
The following statement specifies the name for
the generated file
*/

mus = fopen("mod23","w");

/*
The following section generates the objective function
*/

fprintf(mus,"MODEL\n");
fprintf(mus,"min=");
```

```
 for (i=1;i<=P-2;i++)
 {
     for (k=1;k<=Q-2;k++)
     {
     fprintf(mus,"(%f*v%d%d^(-1)*f%d%d^(-1)+%f*v%d%d^%f\n",
               z1,i,k,i,k,z2,i,k,alpha);
     fprintf(mus,"*f%d%d^%f*d%d%d^%f*%f+%f*(%f-2*d%d%d)\n",
               i,k,beta,i,k,gammaT,cc,z3,D0[i],i,k);
     }
 }

 for (i=1;i<=P-2;i++)
 {
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"*v%d%d^(-1)*f%d%d^(-1)",i,k,i,k);
    }
 }

 for (i=1;i<=P-2;i++)
 {
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"+%f*10^(-3)*(%f-2*d%d%d)\n",z4,D0[i],i,k);
    }
 }
 for (i=1;i<=P-2;i++)
 {
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"*v%d%d^%f*f%d%d^%f*d%d%d^%f*%f\n",i,k,alpha,
        i,k,beta,i,k,gammaT,cc);
    }
 }

 for (i=1;i<=P-2;i++)
 {
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"+%f*(%f-2*d%d%d",z3,D0[i],i,k);
    }
 }

 for (i=1;i<=P-2;i++)
```

```
{
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"-2*d%d%d)",i,k);
    }
}

for (i=1;i<=P-2;i++)
{
    for (k=3;k<=Q;k++)
    {
    fprintf(mus,"*v%d%d^(-1)*f%d%d^(-1)+%f*10^(-3)\n",i,k,i,k,z4);
    }
}

for (i=1;i<=P-2;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"*(%f-2*d%d%d",D0[i],i,k);
    }
}

for (i=1;i<=P-2;i++)
{
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"-2*d%d%d)",i,k);
    }
}

for (i=1;i<=P-2;i++)
{
    for (k=3;k<=Q;k++)
    {
    fprintf(mus,"*v%d%d^%f\n",i,k,alpha);
    fprintf(mus,"*f%d%d^%f*d%d%d^%f*(%f+%f/y%d%d+%f*y%d%d))*(%f+\n",
    i,k,beta,i,k,gammaT,cc,ww,i,k,z5,i,k,lambda[i]);
    }
}

for (i=1;i<=P-2;i++)
{
    for (k=1;k<=Q-2;k++)
```

```
    {
    fprintf(mus,"%f*10^(-7)*%f*T/2*(1-%f*(%f*v%d%d^(-1)*f%d%d^(-1)+%f\n",
    ih,lambda[i],lambda[i],z6,i,k,i,k,z2);
    fprintf(mus,"*v%d%d^%f*f%d%d^%f*d%d%d^%f*%f+%f*(%f-2*d%d%d)\n",
    i,k,alpha,i,k,beta,i,k,gammaT,tr,z3,D0[i],i,k);
    }
}

for (i=1;i<=P-2;i++)
{
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"*v%d%d^(-1)*f%d%d^(-1)+%f*10^(-3)",i,k,i,k,z4);
    }
}

for (i=1;i<=P-2;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"*(%f-2*d%d%d)\n",D0[i],i,k);
    }
}

for (i=1;i<=P-2;i++)
{
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"*v%d%d^%f*f%d%d^%f*d%d%d^%f*%f\n",i,k,alpha,i,k,beta,
    i,k,gammaT,tr);
    }
}

for (i=1;i<=P-2;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"+%f*(%f-2*d%d%d",z7,D0[i],i,k);
    }
}

for (i=1;i<=P-2;i++)
{
    for (k=2;k<=Q-1;k++)
```

```
    {
    fprintf(mus,"-2*d%d%d)",i,k);
    }
}

for (i=1;i<=P-2;i++)
{
    for (k=3;k<=Q;k++)
    {
    fprintf(mus,"*v%d%d^(-1)*f%d%d^(-1)+%f*10^(-3)\n",i,k,i,k,z4);
    }
}

for (i=1;i<=P-2;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"*(%f-2*d%d%d",D0[i],i,k);
    }
}

for (i=1;i<=P-2;i++)
{
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"-2*d%d%d)",i,k);
    }
}

for (i=1;i<=P-2;i++)
{
    for (k=3;k<=Q;k++)
    {
    fprintf(mus,"*v%d%d^%f\n",i,k,alpha);
    fprintf(mus,"*f%d%d^%f*d%d%d^%f*(%f+%f/y%d%d))))+%f/T+\n",
    i,k,beta,i,k,gammaT,tr,wt,i,k,cs);
    }
}

for (i=2;i<=P-1;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"(%f*v%d%d^(-1)*f%d%d^(-1)+%f*v%d%d^%f\n",
```

```
                u1,i,k,i,k,u2,i,k,alpha);
        fprintf(mus,"*f%d%d^%f*d%d%d^%f*%f+%f*(%f-2*d%d%d)\n",
                i,k,beta,i,k,gammaT,cc,u3,D0[i],i,k);
        }
}

for (i=2;i<=P-1;i++)
{

    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"*v%d%d^(-1)*f%d%d^(-1)",i,k,i,k);
    }
}

for (i=2;i<=P-1;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"+%f*10^(-3)*(%f-2*d%d%d)\n",u4,D0[i],i,k);
    }
}
for (i=2;i<=P-1;i++)
{
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"*v%d%d^%f*f%d%d^%f*d%d%d^%f*%f\n",i,k,alpha,
        i,k,beta,i,k,gammaT,cc);
    }
}

for (i=2;i<=P-1;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"+%f*(%f-2*d%d%d",u3,D0[i],i,k);
    }
}

for (i=2;i<=P-1;i++)
{
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"-2*d%d%d)",i,k);
```

```
    }
}

for (i=2;i<=P-1;i++)
{
    for (k=3;k<=Q;k++)
    {
    fprintf(mus,"*v%d%d^(-1)*f%d%d^(-1)+%f*10^(-3)\n",i,k,i,k,u4);
    }
}

for (i=2;i<=P-1;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"*(%f-2*d%d%d",D0[i],i,k);
    }
}

for (i=2;i<=P-1;i++)
{
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"-2*d%d%d)",i,k);
    }
}

for (i=2;i<=P-1;i++)
{
    for (k=3;k<=Q;k++)
    {
    fprintf(mus,"*v%d%d^%f\n",i,k,alpha);
    fprintf(mus,"*f%d%d^%f*d%d%d^%f*(%f+%f/y%d%d+%f*y%d%d))*(%f+\n",
    i,k,beta,i,k,gammaT,cc,ww,i,k,u5,i,k,lambda[i]);
    }
}

for (i=2;i<=P-1;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"%f*10^(-7)*%f*T/2*(1-%f*(%f*v%d%d^(-1)*f%d%d^(-1)+%f\n",
    ih,lambda[i],lambda[i],u6,i,k,i,k,u2);
    fprintf(mus,"*v%d%d^%f*f%d%d^%f*d%d%d^%f*%f+%f*(%f-2*d%d%d)\n",
```

```
        i,k,alpha,i,k,beta,i,k,gammaT,tr,u3,D0[i],i,k);
        }
    }

    for (i=2;i<=P-1;i++)
    {
        for (k=2;k<=Q-1;k++)
        {
        fprintf(mus,"*v%d%d^(-1)*f%d%d^(-1)+%f*10^(-3)",i,k,i,k,u4);
        }
    }

    for (i=2;i<=P-1;i++)
    {
        for (k=1;k<=Q-2;k++)
        {
        fprintf(mus,"*(%f-2*d%d%d)\n",D0[i],i,k);
        }
    }

    for (i=2;i<=P-1;i++)
    {
        for (k=2;k<=Q-1;k++)
        {
        fprintf(mus,"*v%d%d^%f*f%d%d^%f*d%d%d^%f*%f\n",i,k,alpha,i,k,beta,
        i,k,gammaT,tr);
        }
    }

    for (i=2;i<=P-1;i++)
    {
        for (k=1;k<=Q-2;k++)
        {
        fprintf(mus,"+%f*(%f-2*d%d%d",u7,D0[i],i,k);
        }
    }

    for (i=2;i<=P-1;i++)
    {
        for (k=2;k<=Q-1;k++)
        {
        fprintf(mus,"-2*d%d%d)",i,k);
        }
    }
```

```
for (i=2;i<=P-1;i++)
{
    for (k=3;k<=Q;k++)
    {
    fprintf(mus,"*v%d%d^(-1)*f%d%d^(-1)+%f*10^(-3)\n",i,k,i,k,u4);
    }
}

for (i=2;i<=P-1;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"*(%f-2*d%d%d",D0[i],i,k);
    }
}

for (i=2;i<=P-1;i++)
{
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"-2*d%d%d)",i,k);
    }
}

for (i=2;i<=P-1;i++)
{
    for (k=3;k<=Q;k++)
    {
    fprintf(mus,"*v%d%d^%f\n",i,k,alpha);
    fprintf(mus,"*f%d%d^%f*d%d%d^%f*(%f+%f/y%d%d))))+%f/T+\n",i,k,beta,
    i,k,gammaT,tr,wt,i,k,cs);
    }
}

for (i=3;i<=P;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"(%f*v%d%d^(-1)*f%d%d^(-1)+%f*v%d%d^%f\n",
            v1,i,k,i,k,v2,i,k,alpha);
    fprintf(mus,"*f%d%d^%f*d%d%d^%f*%f+%f*(%f-2*d%d%d)\n",
            i,k,beta,i,k,gammaT,cc,v3,D0[i],i,k);
    }
}
```

```
for (i=3;i<=P;i++)
{
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"*v%d%d^(-1)*f%d%d^(-1)",i,k,i,k);
    }
}

for (i=3;i<=P;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"+%f*10^(-3)*(%f-2*d%d%d)\n",v4,D0[i],i,k);
    }
}

for (i=3;i<=P;i++)
{
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"*v%d%d^%f*f%d%d^%f*d%d%d^%f*%f\n",i,k,alpha,
        i,k,beta,i,k,gammaT,cc);
    }
}

for (i=3;i<=P;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"+%f*(%f-2*d%d%d",v3,D0[i],i,k);
    }
}

for (i=3;i<=P;i++)
{
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"-2*d%d%d)",i,k);
    }
}

for (i=3;i<=P;i++)
{
    for (k=3;k<=Q;k++)
```

```
    {
    fprintf(mus,"*v%d%d^(-1)*f%d%d^(-1)+%f*10^(-3)\n",i,k,i,k,v4);
    }
}

for (i=3;i<=P;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"*(%f-2*d%d%d",D0[i],i,k);
    }
}

for (i=3;i<=P;i++)
{
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"-2*d%d%d)",i,k);
    }
}

for (i=3;i<=P;i++)
{
    for (k=3;k<=Q;k++)
    {
    fprintf(mus,"*v%d%d^%f\n",i,k,alpha);
    fprintf(mus,"*f%d%d^%f*d%d%d^%f*(%f+%f/y%d%d+%f*y%d%d))*(%f+\n",
    i,k,beta,i,k,gammaT,cc,ww,i,k,v5,i,k,lambda[i]);
    }
}

for (i=3;i<=P;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"%f*10^(-7)*%f*T/2*(1-%f*(%f*v%d%d^(-1)*f%d%d^(-1)+%f\n",
    ih,lambda[i],lambda[i],v6,i,k,i,k,v2);
    fprintf(mus,"*v%d%d^%f*f%d%d^%f*d%d%d^%f*%f+%f*(%f-2*d%d%d)\n",
    i,k,alpha,i,k,beta,i,k,gammaT,tr,v3,D0[i],i,k);
    }
}

for (i=3;i<=P;i++)
{
```

```
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"*v%d%d^(-1)*f%d%d^(-1)+%f*10^(-3)",i,k,i,k,v4);
    }
}

for (i=3;i<=P;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"*(%f-2*d%d%d)\n",D0[i],i,k);
    }
}

for (i=3;i<=P;i++)
{
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"*v%d%d^%f*f%d%d^%f*d%d%d^%f*%f\n",i,k,alpha,i,k,beta,
    i,k,gammaT,tr);
    }
}

for (i=3;i<=P;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"+%f*(%f-2*d%d%d",v7,D0[i],i,k);
    }
}

for (i=3;i<=P;i++)
{
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"-2*d%d%d)",i,k);
    }
}

for (i=3;i<=P;i++)
{
    for (k=3;k<=Q;k++)
    {
    fprintf(mus,"*v%d%d^(-1)*f%d%d^(-1)+%f*10^(-3)\n",i,k,i,k,v4);
```

```
    }
}

for (i=3;i<=P;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"*(%f-2*d%d%d",D0[i],i,k);
    }
}

for (i=3;i<=P;i++)
{
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"-2*d%d%d)",i,k);
    }
}

for (i=3;i<=P;i++)
{
    for (k=3;k<=Q;k++)
    {
    fprintf(mus,"*v%d%d^%f\n",i,k,alpha);
    fprintf(mus,"*f%d%d^%f*d%d%d^%f*(%f+%f/y%d%d))))+%f/T;\n",i,k,beta,
    i,k,gammaT,tr,wt,i,k,cs);
    }
}

/*
This section generates cutting speed, feed rate, cutting
force and cutting power constraints
*/

for (i=1;i<=P;i++)
{
    for (k=1;k<=Q;k++)
    {
    fprintf(mus,"v%d%d>=%f;\n",i,k,v_l[k]);
    fprintf(mus,"v%d%d<=%f;\n",i,k,v_u[k]);
    fprintf(mus,"f%d%d>=%f;\n",i,k,f_l[k]);
    fprintf(mus,"f%d%d<=%f;\n",i,k,f_u[k]);
    fprintf(mus,"%f*f%d%d^%f*d%d%d^%f<=%f;\n",kF,i,k,betaF,
                i,k,gammaF,Fc_u);
```

```
    fprintf(mus,"%f*10^2*v%d%d*f%d%d^%f*d%d%d^%f<=%f*10^2;\n",kP,i,k,
            i,k,betaF,i,k,gammaF,Pc_u);
    }
}

/*
This section generates depth of cut constraints
*/

for (i=1;i<=P;i++)
{
    for (k=1;k<=Q;k++)
    {
    fprintf(mus,"d%d%d>=%f;\n",i,k,d_l[i][k]);
    fprintf(mus,"d%d%d<=%f;\n",i,k,d_u[i][k]);
    }
}

/*
This section generates the sum of depth of cut constraints
*/

for (i=1;i<=P-2;i++)
{
    for (k=1;k<=Q-1;k++)
    {
    fprintf(mus,"d%d%d+",i,k);
    }
}

for (i=1;i<=P-2;i++)
{
    for (k=3;k<=Q;k++)
    {
    fprintf(mus,"d%d%d=%f;\n",i,k,dtotal[i]);
    }
}

for (i=2;i<=P-1;i++)
{
    for (k=1;k<=Q-1;k++)
    {
    fprintf(mus,"d%d%d+",i,k);
    }
```

```
 }

 for (i=2;i<=P-1;i++)
 {
    for (k=3;k<=Q;k++)
    {
    fprintf(mus,"d%d%d=%f;\n",i,k,dtotal[i]);
    }
 }

 for (i=3;i<=P;i++)
 {
    for (k=1;k<=Q-1;k++)
    {
    fprintf(mus,"d%d%d+",i,k);
    }
 }

 for (i=3;i<=P;i++)
 {
    for (k=3;k<=Q;k++)
    {
    fprintf(mus,"d%d%d=%f;\n",i,k,dtotal[i]);
    }
 }

 /*
 This section generates dimension deviation constraints
 */

 for (i=1;i<=P;i++)
 {
    for (k=3;k<=Q;k++)
    {
    fprintf(mus,"y%d%d>=0;\n",i,k);
    fprintf(mus,"y%d%d<=%f;\n",i,k,delta[i]);
    }
 }


 /*
 This section generates surface finish constraints
 */
```

```
for (i=1;i<=P;i++)
{
    for (k=3;k<=Q;k++)
    {
    fprintf(mus,"%f*v%d%d^%f*f%d%d^%f*d%d%d^%f<=%f;\n",kS,i,k,
    alphaS,i,k,betaS,i,k,gammaS,SF_u[i]);
    }
}

/*
This section generates the constraints such that the
production rates are greater than the demand rates
*/

for (i=1;i<=P-2;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"1/(%f*v%d%d^(-1)*f%d%d^(-1)+%f*v%d%d^%f\n",z6,
    i,k,i,k,z2,i,k,alpha);
    fprintf(mus,"*f%d%d^%f*d%d%d^%f*%f+%f*(%f-2*d%d%d)\n",i,k,
    beta,i,k,gammaT,tr,z3,D0[i],i,k);
    }
}

for (i=1;i<=P-2;i++)
{
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"*v%d%d^(-1)*f%d%d^(-1)+%f*10^(-3)",i,k,i,k,z4);
    }
}

for (i=1;i<=P-2;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"*(%f-2%d%d)\n",D0[i],i,k);
    }
}

for (i=1;i<=P-2;i++)
{
    for (k=2;k<=Q-1;k++)
```

```
    {
    fprintf(mus,"*v%d%d^%f*f%d%d^%f*d%d%d^%f*%f\n",i,k,alpha,
    i,k,beta,i,k,gammaT,tr);
    }
}

for (i=1;i<=P-2;i++)
{
   for (k=1;k<=Q-2;k++)
   {
   fprintf(mus,"+%f*(%f-2*d%d%d",z7,D0[i],i,k);
   }
}

for (i=1;i<=P-2;i++)
{
   for (k=2;k<=Q-1;k++)
   {
   fprintf(mus,"-2*d%d%d)",i,k);
   }
}

for (i=1;i<=P-2;i++)
{
   for (k=3;k<=Q;k++)
   {
   fprintf(mus,"*v%d%d^(-1)*f%d%d^(-1)*%f*10^(-3)\n",i,k,i,k,z4);
   }
}

for (i=1;i<=P-2;i++)
{
   for (k=1;k<=Q-2;k++)
   {
   fprintf(mus,"*(%f-2*d%d%d",D0[i],i,k);
   }
}

for (i=1;i<=P-2;i++)
{
   for (k=2;k<=Q-1;k++)
   {
   fprintf(mus,"-2*d%d%d)",i,k);
   }
```

```
}

for (i=1;i<=P-2;i++)
{
    for (k=3;k<=Q;k++)
    {
    fprintf(mus,"*v%d%d^%f\n",i,k,alpha);
    fprintf(mus,"*f%d%d^%f*d%d%d^%f*(%f+%f/y%d%d))>=%f;\n",i,k,
    beta,i,k,gammaT,tr,wt,i,k,rate[i],i);
    }
}

for (i=2;i<=P-1;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"1/(%f*v%d%d^(-1)*f%d%d^(-1)+%f*v%d%d^%f\n",u6,
    i,k,i,k,u2,i,k,alpha);
    fprintf(mus,"*f%d%d^%f*d%d%d^%f*%f+%f*(%f-2*d%d%d)\n",i,k,
    beta,i,k,gammaT,tr,u3,D0[i],i,k);
    }
}

for (i=2;i<=P-1;i++)
{
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"*v%d%d^(-1)*f%d%d^(-1)+%f*10^(-3)",i,k,i,k,u4);
    }
}

for (i=2;i<=P-1;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"*(%f-2%d%d)\n",D0[i],i,k);
    }
}

for (i=2;i<=P-1;i++)
{
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"*v%d%d^%f*f%d%d^%f*d%d%d^%f*%f\n",i,k,alpha,
```

```
    i,k,beta,i,k,gammaT,tr);
    }
}

for (i=2;i<=P-1;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"+%f*(%f-2*d%d%d",u7,D0[i],i,k);
    }
}

for (i=2;i<=P-1;i++)
{
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"-2*d%d%d)",i,k);
    }
}

for (i=2;i<=P-1;i++)
{
    for (k=3;k<=Q;k++)
    {
    fprintf(mus,"*v%d%d^(-1)*f%d%d^(-1)*%f*10^(-3)\n",i,k,i,k,u4);
    }
}

for (i=2;i<=P-1;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"*(%f-2*d%d%d",D0[i],i,k);
    }
}

for (i=2;i<=P-1;i++)
{
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"-2*d%d%d)",i,k);
    }
}
```

```
for (i=2;i<=P-1;i++)
{
    for (k=3;k<=Q;k++)
    {
    fprintf(mus,"*v%d%d^%f\n",i,k,alpha);
    fprintf(mus,"*f%d%d^%f*d%d%d^%f*(%f+%f/y%d%d))>=%f;\n",i,k,
    beta,i,k,gammaT,tr,wt,i,k,rate[i],i);
    }
}

for (i=3;i<=P;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"1/(%f*v%d%d^(-1)*f%d%d^(-1)+%f*v%d%d^%f\n",v6,
    i,k,i,k,v2,i,k,alpha);
    fprintf(mus,"*f%d%d^%f*d%d%d^%f*%f+%f*(%f-2*d%d%d)\n",i,k,
    beta,i,k,gammaT,tr,v3,D0[i],i,k);
    }
}

for (i=3;i<=P;i++)
{
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"*v%d%d^(-1)*f%d%d^(-1)+%f*10^(-3)",i,k,i,k,v4);
    }
}

for (i=3;i<=P;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"*(%f-2%d%d)\n",D0[i],i,k);
    }
}

for (i=3;i<=P;i++)
{
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"*v%d%d^%f*f%d%d^%f*d%d%d^%f*%f\n",i,k,alpha,
    i,k,beta,i,k,gammaT,tr);
    }
```

```
}

for (i=3;i<=P;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"+%f*(%f-2*d%d%d",v7,D0[i],i,k);
    }
}

for (i=3;i<=P;i++)
{
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"-2*d%d%d)",i,k);
    }
}

for (i=3;i<=P;i++)
{
    for (k=3;k<=Q;k++)
    {
    fprintf(mus,"*v%d%d^(-1)*f%d%d^(-1)*%f*10^(-3)\n",i,k,i,k,v4);
    }
}

for (i=3;i<=P;i++)
{
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"*(%f-2*d%d%d",D0[i],i,k);
    }
}

for (i=3;i<=P;i++)
{
    for (k=2;k<=Q-1;k++)
    {
    fprintf(mus,"-2*d%d%d)",i,k);
    }
}

for (i=3;i<=P;i++)
{
```

```
    for (k=3;k<=Q;k++)
    {
    fprintf(mus,"*v%d%d^%f\n",i,k,alpha);
    fprintf(mus,"*f%d%d^%f*d%d%d^%f*(%f+%f/y%d%d))>=%f;\n",i,k,
    beta,i,k,gammaT,tr,wt,i,k,rate[i],i);
    }
}

fprintf(mus,"END\n");
fprintf(mus,"BAT\n");
return 0;

}
```

Appendix F

LINGO-format-input for the example problem of Model 2.3

```
MODEL
min=(339.291718*v11^(-1)*f11^(-1)+0.000072*v11^0.700000
*f11^0.550000*d11^1.220000*7.000000+2.827431*(120.000000-2*d11)
*v12^(-1)*f12^(-1)+0.000600*10^(-3)*(120.000000-2*d11)
*v12^0.700000*f12^0.550000*d12^1.220000*7.000000
+2.827431*(120.000000-2*d11-2*d12)*v13^(-1)*f13^(-1)+0.000600*10^(-3)
*(120.000000-2*d11-2*d12)*v13^0.700000
*f13^0.550000*d13^1.220000*(7.000000+0.060000/y13+23.026567*y13))*(0.180000+
3.858000*10^(-7)*0.180000*T/2*(1-0.180000*(113.097237*v11^(-1)*f11^(-1)+0.000072
*v11^0.700000*f11^0.550000*d11^1.220000*0.500000+2.827431*(120.000000-2*d11)
*v12^(-1)*f12^(-1)+0.000600*10^(-3)*(120.000000-2*d11)
*v12^0.700000*f12^0.550000*d12^1.220000*0.500000
+0.942477*(120.000000-2*d11-2*d12)*v13^(-1)*f13^(-1)+0.000600*10^(-3)
*(120.000000-2*d11-2*d12)*v13^0.700000
*f13^0.550000*d13^1.220000*(0.500000+0.020000/y13))))+40.000000/T+
(70.685776*v21^(-1)*f21^(-1)+0.000015*v21^0.700000
*f21^0.550000*d21^1.220000*7.000000+1.413715*(50.000000-2*d21)
*v22^(-1)*f22^(-1)+0.000300*10^(-3)*(50.000000-2*d21)
*v22^0.700000*f22^0.550000*d22^1.220000*7.000000
+1.413715*(50.000000-2*d21-2*d22)*v23^(-1)*f23^(-1)+0.000300*10^(-3)
*(50.000000-2*d21-2*d22)*v23^0.700000
*f23^0.550000*d23^1.220000*(7.000000+0.060000/y23+23.026567*y23))*(0.120000+
3.858000*10^(-7)*0.120000*T/2*(1-0.120000*(23.561926*v21^(-1)*f21^(-1)+0.000015
*v21^0.700000*f21^0.550000*d21^1.220000*0.500000+1.413715*(50.000000-2*d21)
*v22^(-1)*f22^(-1)+0.000300*10^(-3)*(50.000000-2*d21)
*v22^0.700000*f22^0.550000*d22^1.220000*0.500000
+0.471238*(50.000000-2*d21-2*d22)*v23^(-1)*f23^(-1)+0.000300*10^(-3)
*(50.000000-2*d21-2*d22)*v23^0.700000
*f23^0.550000*d23^1.220000*(0.500000+0.020000/y23))))+40.000000/T+
(169.645859*v31^(-1)*f31^(-1)+0.000036*v31^0.700000
*f31^0.550000*d31^1.220000*7.000000+1.884954*(90.000000-2*d31)
*v32^(-1)*f32^(-1)+0.000400*10^(-3)*(90.000000-2*d31)
*v32^0.700000*f32^0.550000*d32^1.220000*7.000000
+1.884954*(90.000000-2*d31-2*d32)*v33^(-1)*f33^(-1)+0.000400*10^(-3)
*(90.000000-2*d31-2*d32)*v33^0.700000
```

```
*f33^0.550000*d33^1.220000*(7.000000+0.060000/y33+23.026567*y33))*(0.150000+
3.858000*10^(-7)*0.150000*T/2*(1-0.150000*(56.548618*v31^(-1)*f31^(-1)+0.000036
*v31^0.700000*f31^0.550000*d31^1.220000*0.500000+1.884954*(90.000000-2*d31)
*v32^(-1)*f32^(-1)+0.000400*10^(-3)*(90.000000-2*d31)
*v32^0.700000*f32^0.550000*d32^1.220000*0.500000
+0.628318*(90.000000-2*d31-2*d32)*v33^(-1)*f33^(-1)+0.000400*10^(-3)
*(90.000000-2*d31-2*d32)*v33^0.700000
*f33^0.550000*d33^1.220000*(0.500000+0.020000/y33))))+40.000000/T;
v11>=90.000000;
v11<=120.000000;
f11>=0.800000;
f11<=2.000000;
1.380000*f11^1.180000*d11^1.260000<=20.000000;
0.000282*10^2*v11*f11^1.180000*d11^1.260000<=2.000000*10^2;
v12>=90.000000;
v12<=120.000000;
f12>=0.800000;
f12<=2.000000;
1.380000*f12^1.180000*d12^1.260000<=20.000000;
0.000282*10^2*v12*f12^1.180000*d12^1.260000<=2.000000*10^2;
v13>=168.000000;
v13<=210.000000;
f13>=0.130000;
f13<=0.500000;
1.380000*f13^1.180000*d13^1.260000<=20.000000;
0.000282*10^2*v13*f13^1.180000*d13^1.260000<=2.000000*10^2;
v21>=90.000000;
v21<=120.000000;
f21>=0.800000;
f21<=2.000000;
1.380000*f21^1.180000*d21^1.260000<=20.000000;
0.000282*10^2*v21*f21^1.180000*d21^1.260000<=2.000000*10^2;
v22>=90.000000;
v22<=120.000000;
f22>=0.800000;
f22<=2.000000;
1.380000*f22^1.180000*d22^1.260000<=20.000000;
0.000282*10^2*v22*f22^1.180000*d22^1.260000<=2.000000*10^2;
v23>=168.000000;
v23<=210.000000;
f23>=0.130000;
f23<=0.500000;
1.380000*f23^1.180000*d23^1.260000<=20.000000;
0.000282*10^2*v23*f23^1.180000*d23^1.260000<=2.000000*10^2;
```

```
v31>=90.000000;
v31<=120.000000;
f31>=0.800000;
f31<=2.000000;
1.380000*f31^1.180000*d31^1.260000<=20.000000;
0.000282*10^2*v31*f31^1.180000*d31^1.260000<=2.000000*10^2;
v32>=90.000000;
v32<=120.000000;
f32>=0.800000;
f32<=2.000000;
1.380000*f32^1.180000*d32^1.260000<=20.000000;
0.000282*10^2*v32*f32^1.180000*d32^1.260000<=2.000000*10^2;
v33>=168.000000;
v33<=210.000000;
f33>=0.130000;
f33<=0.500000;
1.380000*f33^1.180000*d33^1.260000<=20.000000;
0.000282*10^2*v33*f33^1.180000*d33^1.260000<=2.000000*10^2;
d11>=1.000000;
d11<=5.000000;
d12>=1.000000;
d12<=5.000000;
d13>=0.400000;
d13<=0.900000;
d21>=1.000000;
d21<=5.000000;
d22>=1.000000;
d22<=5.000000;
d23>=0.300000;
d23<=0.800000;
d31>=1.000000;
d31<=5.000000;
d32>=1.000000;
d32<=5.000000;
d33>=0.350000;
d33<=0.850000;
d11+d12+d13=5.000000;
d21+d22+d23=8.000000;
d31+d32+d33=10.000000;
y13>=0;
y13<=0.065900;
y23>=0;
y23<=0.065900;
y33>=0;
```

```
y33<=0.065900;
1.170000*v13^-0.250000*f13^0.720000*d13^0.230000<=1.600000;
1.170000*v23^-0.250000*f23^0.720000*d23^0.230000<=0.500000;
1.170000*v33^-0.250000*f33^0.720000*d33^0.230000<=0.800000;
1/(113.097237*v11^(-1)*f11^(-1)+0.000072*v11^0.700000
*f11^0.550000*d11^1.220000*0.500000+2.827431*(120.000000-2*d11)
*v12^(-1)*f12^(-1)+0.000600*10^(-3)*(120.000000-211)
*v12^0.700000*f12^0.550000*d12^1.220000*0.500000
+0.942477*(120.000000-2*d11-2*d12)*v13^(-1)*f13^(-1)*0.000600*10^(-3)
*(120.000000-2*d11-2*d12)*v13^0.700000
*f13^0.550000*d13^1.220000*(0.500000+0.020000/y13))>=0.500000;
1/(23.561926*v21^(-1)*f21^(-1)+0.000015*v21^0.700000
*f21^0.550000*d21^1.220000*0.500000+1.413715*(50.000000-2*d21)
*v22^(-1)*f22^(-1)+0.000300*10^(-3)*(50.000000-221)
*v22^0.700000*f22^0.550000*d22^1.220000*0.500000
+0.471238*(50.000000-2*d21-2*d22)*v23^(-1)*f23^(-1)*0.000300*10^(-3)
*(50.000000-2*d21-2*d22)*v23^0.700000
*f23^0.550000*d23^1.220000*(0.500000+0.020000/y23))>=0.400000;
1/(56.548618*v31^(-1)*f31^(-1)+0.000036*v31^0.700000
*f31^0.550000*d31^1.220000*0.500000+1.884954*(90.000000-2*d31)
*v32^(-1)*f32^(-1)+0.000400*10^(-3)*(90.000000-231)
*v32^0.700000*f32^0.550000*d32^1.220000*0.500000
+0.628318*(90.000000-2*d31-2*d32)*v33^(-1)*f33^(-1)*0.000400*10^(-3)
*(90.000000-2*d31-2*d32)*v33^0.700000
*f33^0.550000*d33^1.220000*(0.500000+0.020000/y33))>=0.450000;
END
BAT
```

Appendix G

C code to generate LINGO-format-input for Model 2.4

```
/*
This program is used to generate the LINGO-format-input model
*/

# include <stdio.h>
# include <conio.h>
# include <stdlib.h>
# include <math.h>

# define pi 3.14159
# define c 6120

/*
P = number of machines
Jk = number of passes of feature k
Q = number of features
*/

int i,j,k,P,Jk,Q,R;

float alpha,beta,z1,z2,z3,z4,z5,z6,z7,z8,z9,z10,z11,z12,
      zz1,zz2,zz3,zz4,zz5,zz6,zz7,zz8,zz9,ww,wt,cc,kP;

float  co=3.0,
       kT=1570000.00,
       alphaT=1.70,
       betaT=1.55,
       gammaT=1.22,
       ct=5.5,
       tr=0.5,
       w=0.1,
       ca=3.0,
       ta=0.2,
       A=1.0,
```

```
        betaF=1.18,
        gammaF=1.26,
        kF=1.38,
        Fc_u=20.0,
        eta=0.8,
        Pc_u=2.0,
        kS=1.17,
        alphaS=-0.25,
        betaS=0.72,
        gammaS=0.23;


static float D0[4] = {0.0,150.0,140.0,140.0};
static float L[4] = {0.0,300.0,100.0,150.0};
static float dtotal[4] = {0.0,5.0,8.0,10.0};
static float SF_u[4] = {0.0,1.6,1.6,1.6};
static float delta[4] = {0.0,0.0659,0.0659,0.0659};
static float v_l[4] = {0.0,90.0,90.0,168.0};
static float v_u[4] = {0.0,120.0,120.0,210.0};
static float f_l[4] = {0.0,0.8,0.8,0.13};
static float f_u[4] = {0.0,2.0,2.0,0.50};
static float d_l[4] = {0.0,0.0,0.0,0.3};
static float d_u[4] = {0.0,5.0,5.0,1.0};

FILE *mus;

main()

{

clrscr();

P=3;Jk=3;Q=3;
R=1000;
alpha=alphaT-1;
beta=betaT-1;
z1=pi*D0[1]*L[1]/R;
z2=pi*D0[1]*L[1]/(R*kT);
z3=pi*L[1]/R;
z4=pi*L[1]/kT;
z5=pi*D0[2]*L[2]/R;
z6=pi*D0[2]*L[2]/(R*kT);
z7=pi*L[2]/R;
z8=pi*L[2]/kT;
```

```
z9=pi*D0[3]*L[3]/R;
z10=pi*D0[3]*L[3]/(R*kT);
z11=pi*L[3]/R;
z12=pi*L[3]/kT;
zz1=pi*D0[1]*L[1]*co/R;
zz2=pi*L[1]*co/R;
zz3=pi*D0[2]*L[2]*co/R;
zz4=pi*L[2]*co/R;
zz5=pi*D0[3]*L[3]*co/R;
zz6=pi*L[3]*co/R;
zz7=w*A/(delta[1]*delta[1]);
zz8=w*A/(delta[2]*delta[2]);
zz9=w*A/(delta[3]*delta[3]);
ww=w*ca*ta;
wt=w*ta;
kP=kF/(eta*c);
cc=ct+co*tr;

/*
The following statement specifies the name for
the generated file
*/

mus = fopen("mod24","w");

/*
The following section generates the objective function
*/

fprintf(mus,"MODEL\n");
fprintf(mus,"min=");

for (i=1;i<=P-1;i++)
{
 for (j=1;j<=Jk-2;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"(%f*v%d%d%d^(-1)*f%d%d%d^(-1)+%f*v%d%d%d^%f\n",
        zz1,i,j,k,i,j,k,z2,i,j,k,alpha);
  fprintf(mus,"*f%d%d%d^%f*d%d%d%d^%f*%f)*x%d%d%d+",
        i,j,k,beta,i,j,k,gammaT,cc,i,j,k);
  fprintf(mus,"(%f*(%f-2*d%d%d%d)\n",zz2,D0[k],i,j,k);
  }
```

```
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"*v%d%d%d^(-1)*f%d%d%d^(-1)",i,j,k,i,j,k);
  }
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"+%f*10^(-3)*(%f-2*d%d%d%d)\n",z4,D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"*v%d%d%d^%f*f%d%d%d^%f*d%d%d%d^%f*%f)*x%d%d%d\n",
        i,j,k,alpha,i,j,k,beta,i,j,k,gammaT,cc,i,j,k);
  }
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"+(%f*(%f-2*d%d%d%d",zz2,D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"-2*d%d%d%d)",i,j,k);
  }
 }

 for (j=3;j<=Jk;j++)
```

```
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"*v%d%d%d^(-1)*f%d%d%d^(-1)+%f*10^(-3)\n",
         i,j,k,i,j,k,z4);
  }
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"*(%f-2*d%d%d%d",D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=1;k<=Q-2;k++) {
  fprintf(mus,"-2*d%d%d%d)",i,j,k);
  }
 }

 for (j=3;j<=Jk;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"*v%d%d%d^%f\n",i,j,k,alpha);
  fprintf(mus,"*f%d%d%d^%f*d%d%d%d^%f*(%f+%f/y%d%d%d",
         i,j,k,beta,i,j,k,gammaT,cc,ww,i,j,k);
  fprintf(mus,"+%f*y%d%d%d))*x%d%d%d\n",
         zz7,i,j,k,i,j,k);
  }
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"+(%f*v%d%d%d^(-1)*f%d%d%d^(-1)+%f*v%d%d%d^%f\n",
         zz3,i,j,k,i,j,k,z6,i,j,k,alpha);
  fprintf(mus,"*f%d%d%d^%f*d%d%d%d^%f*%f)*x%d%d%d+",
         i,j,k,beta,i,j,k,gammaT,cc,i,j,k);
  fprintf(mus,"(%f*(%f-2*d%d%d%d)\n",zz4,D0[k],i,j,k);
```

```
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"*v%d%d%d^(-1)*f%d%d%d^(-1)",i,j,k,i,j,k);
  }
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"+%f*10^(-3)*(%f-2*d%d%d%d)\n",z8,D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"*v%d%d%d^%f*f%d%d%d^%f*d%d%d%d^%f*%f)*x%d%d%d\n",
        i,j,k,alpha,i,j,k,beta,i,j,k,gammaT,cc,i,j,k);
  }
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"+(%f*(%f-2*d%d%d%d",zz4,D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"-2*d%d%d%d)",i,j,k);
  }
 }
```

```
for (j=3;j<=Jk;j++)
{
 for (k=2;k<=Q-1;k++)
 {
 fprintf(mus,"*v%d%d%d^(-1)*f%d%d%d^(-1)+%f*10^(-3)\n",
        i,j,k,i,j,k,z8);
 }
}

for (j=1;j<=Jk-2;j++)
{
 for (k=2;k<=Q-1;k++)
 {
 fprintf(mus,"*(%f-2*d%d%d%d",D0[k],i,j,k);
 }
}

for (j=2;j<=Jk-1;j++)
{
 for (k=2;k<=Q-1;k++)
 {
 fprintf(mus,"-2*d%d%d%d)",i,j,k);
 }
}

for (j=3;j<=Jk;j++)
{
 for (k=2;k<=Q-1;k++)
 {
 fprintf(mus,"*v%d%d%d^%f\n",i,j,k,alpha);
 fprintf(mus,"*f%d%d%d^%f*d%d%d%d^%f*(%f+%f/y%d%d%d",
        i,j,k,beta,i,j,k,gammaT,cc,ww,i,j,k);
 fprintf(mus,"+%f*y%d%d%d))*x%d%d%d\n",
        zz8,i,j,k,i,j,k);
 }
}

for (j=1;j<=Jk-2;j++)
{
 for (k=3;k<=Q;k++)
 {
 fprintf(mus,"+(%f*v%d%d%d^(-1)*f%d%d%d^(-1)+%f*v%d%d%d^%f\n",
        zz5,i,j,k,i,j,k,z10,i,j,k,alpha);
 fprintf(mus,"*f%d%d%d^%f*d%d%d%d^%f*%f)*x%d%d%d+",
```

```
        i,j,k,beta,i,j,k,gammaT,cc,i,j,k);
  fprintf(mus,"(%f*(%f-2*d%d%d%d)\n",zz6,D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"*v%d%d%d^(-1)*f%d%d%d^(-1)",i,j,k,i,j,k);
  }
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"+%f*10^(-3)*(%f-2*d%d%d%d)\n",z12,D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"*v%d%d%d^%f*f%d%d%d^%f*d%d%d%d^%f*%f)*x%d%d%d\n",
        i,j,k,alpha,i,j,k,beta,i,j,k,gammaT,cc,i,j,k);
  }
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"+(%f*(%f-2*d%d%d%d",zz6,D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"-2*d%d%d%d)",i,j,k);
  }
```

```
  }

  for (j=3;j<=Jk;j++)
  {
   for (k=3;k<=Q;k++)
   {
   fprintf(mus,"*v%d%d%d^(-1)*f%d%d%d^(-1)+%f*10^(-3)\n",
          i,j,k,i,j,k,z12);
   }
  }

  for (j=1;j<=Jk-2;j++)
  {
   for (k=3;k<=Q;k++)
   {
   fprintf(mus,"*(%f-2*d%d%d%d",D0[k],i,j,k);
   }
  }

  for (j=2;j<=Jk-1;j++)
  {
   for (k=3;k<=Q;k++)
   {
   fprintf(mus,"-2*d%d%d%d)",i,j,k);
   }
  }

  for (j=3;j<=Jk;j++)
  {
   for (k=3;k<=Q;k++)
   {
   fprintf(mus,"*v%d%d%d^%f\n",i,j,k,alpha);
   fprintf(mus,"*f%d%d%d^%f*d%d%d%d^%f*(%f+%f/y%d%d%d",
          i,j,k,beta,i,j,k,gammaT,cc,ww,i,j,k);
   fprintf(mus,"+%f*y%d%d%d))*x%d%d%d+\n",
          zz9,i,j,k,i,j,k);
   }
  }
 }

for (i=3;i<=P;i++)
{
 for (j=1;j<=Jk-2;j++)
 {
```

```
 for (k=1;k<=Q-2;k++)
 {
 fprintf(mus,"(%f*v%d%d%d^(-1)*f%d%d%d^(-1)+%f*v%d%d%d^%f\n",
        zz1,i,j,k,i,j,k,z2,i,j,k,alpha);
 fprintf(mus,"*f%d%d%d^%f*d%d%d%d^%f*%f)*x%d%d%d+",
        i,j,k,beta,i,j,k,gammaT,cc,i,j,k);
 fprintf(mus,"(%f*(%f-2*d%d%d%d)\n",zz2,D0[k],i,j,k);
 }
}

for (j=2;j<=Jk-1;j++)
{
 for (k=1;k<=Q-2;k++)
 {
 fprintf(mus,"*v%d%d%d^(-1)*f%d%d%d^(-1)",i,j,k,i,j,k);
 }
}

for (j=1;j<=Jk-2;j++)
{
 for (k=1;k<=Q-2;k++)
 {
 fprintf(mus,"+%f*10^(-3)*(%f-2*d%d%d%d)\n",z4,D0[k],i,j,k);
 }
}

for (j=2;j<=Jk-1;j++)
{
 for (k=1;k<=Q-2;k++)
 {
 fprintf(mus,"*v%d%d%d^%f*f%d%d%d^%f*d%d%d%d^%f*%f)*x%d%d%d\n",
        i,j,k,alpha,i,j,k,beta,i,j,k,gammaT,cc,i,j,k);
 }
}

for (j=1;j<=Jk-2;j++)
{
 for (k=1;k<=Q-2;k++)
 {
 fprintf(mus,"+(%f*(%f-2*d%d%d%d",zz2,D0[k],i,j,k);
 }
}

for (j=2;j<=Jk-1;j++)
```

```
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"-2*d%d%d%d)",i,j,k);
  }
 }

 for (j=3;j<=Jk;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"*v%d%d%d^(-1)*f%d%d%d^(-1)+%f*10^(-3)\n",
         i,j,k,i,j,k,z4);
  }
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"*(%f-2*d%d%d%d",D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=1;k<=Q-2;k++) {
  fprintf(mus,"-2*d%d%d%d)",i,j,k);
  }
 }

 for (j=3;j<=Jk;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"*v%d%d%d^%f\n",i,j,k,alpha);
  fprintf(mus,"*f%d%d%d^%f*d%d%d%d^%f*(%f+%f/y%d%d%d",
         i,j,k,beta,i,j,k,gammaT,cc,ww,i,j,k);
  fprintf(mus,"+%f*y%d%d%d))*x%d%d%d\n",
         zz7,i,j,k,i,j,k);
  }
 }

 for (j=1;j<=Jk-2;j++)
```

```
{
 for (k=2;k<=Q-1;k++)
 {
 fprintf(mus,"+(%f*v%d%d%d^(-1)*f%d%d%d^(-1)+%f*v%d%d%d^%f\n",
        zz3,i,j,k,i,j,k,z6,i,j,k,alpha);
 fprintf(mus,"*f%d%d%d^%f*d%d%d%d^%f*f)*x%d%d%d+",
        i,j,k,beta,i,j,k,gammaT,cc,i,j,k);
 fprintf(mus,"(%f*(%f-2*d%d%d%d)\n",zz4,D0[k],i,j,k);
 }
}

for (j=2;j<=Jk-1;j++)
{
 for (k=2;k<=Q-1;k++)
 {
 fprintf(mus,"*v%d%d%d^(-1)*f%d%d%d^(-1)",i,j,k,i,j,k);
 }
}

for (j=1;j<=Jk-2;j++)
{
 for (k=2;k<=Q-1;k++)
 {
 fprintf(mus,"+%f*10^(-3)*(%f-2*d%d%d%d)\n",z8,D0[k],i,j,k);
 }
}

for (j=2;j<=Jk-1;j++)
{
 for (k=2;k<=Q-1;k++)
 {
 fprintf(mus,"*v%d%d%d^%f*f%d%d%d^%f*d%d%d%d^%f*f)*x%d%d%d\n",
        i,j,k,alpha,i,j,k,beta,i,j,k,gammaT,cc,i,j,k);
 }
}

for (j=1;j<=Jk-2;j++)
{
 for (k=2;k<=Q-1;k++)
 {
 fprintf(mus,"+(%f*(%f-2*d%d%d%d",zz4,D0[k],i,j,k);
 }
}
```

```
 for (j=2;j<=Jk-1;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"-2*d%d%d%d)",i,j,k);
  }
 }

 for (j=3;j<=Jk;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"*v%d%d%d^(-1)*f%d%d%d^(-1)+%f*10^(-3)\n",
          i,j,k,i,j,k,z8);
  }
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"*(%f-2*d%d%d%d",D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"-2*d%d%d%d)",i,j,k);
  }
 }

 for (j=3;j<=Jk;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"*v%d%d%d^%f\n",i,j,k,alpha);
  fprintf(mus,"*f%d%d%d^%f*d%d%d%d^%f*(%f+%f/y%d%d%d",
          i,j,k,beta,i,j,k,gammaT,cc,ww,i,j,k);
  fprintf(mus,"+%f*y%d%d%d))*x%d%d%d\n",
          zz8,i,j,k,i,j,k);
  }
 }
```

```
for (j=1;j<=Jk-2;j++)
{
 for (k=3;k<=Q;k++)
 {
 fprintf(mus,"+(%f*v%d%d%d^(-1)*f%d%d%d^(-1)+%f*v%d%d%d^%f\n",
        zz5,i,j,k,i,j,k,z10,i,j,k,alpha);
 fprintf(mus,"*f%d%d%d^%f*d%d%d%d^%f*%f)*x%d%d%d+",
        i,j,k,beta,i,j,k,gammaT,cc,i,j,k);
 fprintf(mus,"(%f*(%f-2*d%d%d%d)\n",zz6,D0[k],i,j,k);
 }
}

for (j=2;j<=Jk-1;j++)
{
 for (k=3;k<=Q;k++)
 {
 fprintf(mus,"*v%d%d%d^(-1)*f%d%d%d^(-1)",i,j,k,i,j,k);
 }
}

for (j=1;j<=Jk-2;j++)
{
 for (k=3;k<=Q;k++)
 {
 fprintf(mus,"+%f*10^(-3)*(%f-2*d%d%d%d)\n",z12,D0[k],i,j,k);
 }
}

for (j=2;j<=Jk-1;j++)
{
 for (k=3;k<=Q;k++)
 {
 fprintf(mus,"*v%d%d%d^%f*f%d%d%d^%f*d%d%d%d^%f*%f)*x%d%d%d\n",
        i,j,k,alpha,i,j,k,beta,i,j,k,gammaT,cc,i,j,k);
 }
}

for (j=1;j<=Jk-2;j++)
{
 for (k=3;k<=Q;k++)
 {
 fprintf(mus,"+(%f*(%f-2*d%d%d%d",zz6,D0[k],i,j,k);
 }
}
```

```
for (j=2;j<=Jk-1;j++)
{
 for (k=3;k<=Q;k++)
 {
 fprintf(mus,"-2*d%d%d%d)",i,j,k);
 }
}

for (j=3;j<=Jk;j++)
{
 for (k=3;k<=Q;k++)
 {
 fprintf(mus,"*v%d%d%d^(-1)*f%d%d%d^(-1)+%f*10^(-3)\n",
        i,j,k,i,j,k,z12);
 }
}

for (j=1;j<=Jk-2;j++)
{
 for (k=3;k<=Q;k++)
 {
 fprintf(mus,"*(%f-2*d%d%d%d",D0[k],i,j,k);
 }
}

for (j=2;j<=Jk-1;j++)
{
 for (k=3;k<=Q;k++)
 {
 fprintf(mus,"-2*d%d%d%d)",i,j,k);
 }
}

for (j=3;j<=Jk;j++)
{
 for (k=3;k<=Q;k++)
 {
 fprintf(mus,"*v%d%d%d^%f\n",i,j,k,alpha);
 fprintf(mus,"*f%d%d%d^%f*d%d%d%d^%f*(%f+%f/y%d%d%d",
        i,j,k,beta,i,j,k,gammaT,cc,ww,i,j,k);
 fprintf(mus,"+%f*y%d%d%d))*x%d%d%d;\n",
        zz9,i,j,k,i,j,k);

}
```

```
 }
}

/*
This section generates production time constraints
*/

for (i=1;i<=P;i++)
{
 for (j=1;j<=Jk-2;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"(%f*v%d%d%d^(-1)*f%d%d%d^(-1)+%f*v%d%d%d^%f\n",
         z1,i,j,k,i,j,k,z2,i,j,k,alpha);
  fprintf(mus,"*f%d%d%d^%f*d%d%d%d^%f*%f)*x%d%d%d+",
         i,j,k,beta,i,j,k,gammaT,tr,i,j,k);
  fprintf(mus,"(%f*(%f-2*d%d%d%d)\n",z3,D0[k],i,j,k);
  }
 }

for (j=2;j<=Jk-1;j++)
{
 for (k=1;k<=Q-2;k++)
 {
 fprintf(mus,"*v%d%d%d^(-1)*f%d%d%d^(-1)",i,j,k,i,j,k);
 }
}

for (j=1;j<=Jk-2;j++)
{
 for (k=1;k<=Q-2;k++)
 {
 fprintf(mus,"+%f*10^(-3)*(%f-2*d%d%d%d)\n",z4,D0[k],i,j,k);
 }
}

for (j=2;j<=Jk-1;j++)
{
 for (k=1;k<=Q-2;k++)
 {
 fprintf(mus,"*v%d%d%d^%f*f%d%d%d^%f*d%d%d%d^%f*%f)*x%d%d%d\n",
        i,j,k,alpha,i,j,k,beta,i,j,k,gammaT,tr,i,j,k);
 }
```

```
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"+(%f*(%f-2*d%d%d%d",z3,D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"-2*d%d%d%d)",i,j,k);
  }
 }

 for (j=3;j<=Jk;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"*v%d%d%d^(-1)*f%d%d%d^(-1)+%f*10^(-3)\n",
         i,j,k,i,j,k,z4);
  }
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"*(%f-2*d%d%d%d",D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"-2*d%d%d%d)",i,j,k);
  }
 }

 for (j=3;j<=Jk;j++)
```

```
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"*v%d%d%d^%f\n",i,j,k,alpha);
  fprintf(mus,"*f%d%d%d^%f*d%d%d%d^%f*(%f+%f/y%d%d%d))*x%d%d%d\n",
         i,j,k,beta,i,j,k,gammaT,tr,wt,i,j,k,i,j,k);
  }
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"+(%f*v%d%d%d^(-1)*f%d%d%d^(-1)+%f*v%d%d%d^%f\n",
         z5,i,j,k,i,j,k,z6,i,j,k,alpha);
  fprintf(mus,"*f%d%d%d^%f*d%d%d%d^%f*%f)*x%d%d%d+",
         i,j,k,beta,i,j,k,gammaT,tr,i,j,k);
  fprintf(mus,"(%f*(%f-2*d%d%d%d)\n",z7,D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"*v%d%d%d^(-1)*f%d%d%d^(-1)",i,j,k,i,j,k);
  }
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"+%f*10^(-3)*(%f-2*d%d%d%d)\n",z8,D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"*v%d%d%d^%f*f%d%d%d^%f*d%d%d%d^%f*%f)*x%d%d%d\n",
         i,j,k,alpha,i,j,k,beta,i,j,k,gammaT,tr,i,j,k);
 }
```

```
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"+(%f*(%f-2*d%d%d%d",z7,D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"-2*d%d%d%d)",i,j,k);
  }
 }

 for (j=3;j<=Jk;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"*v%d%d%d^(-1)*f%d%d%d^(-1)+%f*10^(-3)\n",
         i,j,k,i,j,k,z8);
  }
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"*(%f-2*d%d%d%d",D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"-2*d%d%d%d)",i,j,k);
  }
 }

 for (j=3;j<=Jk;j++)
```

```
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"*v%d%d%d^%f\n",i,j,k,alpha);
 fprintf(mus,"*f%d%d%d^%f*d%d%d%d^%f*(%f+%f/y%d%d%d))*x%d%d%d\n",
         i,j,k,beta,i,j,k,gammaT,tr,wt,i,j,k,i,j,k);
  }
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"+(%f*v%d%d%d^(-1)*f%d%d%d^(-1)+%f*v%d%d%d^%f\n",
         z9,i,j,k,i,j,k,z10,i,j,k,alpha);
  fprintf(mus,"*f%d%d%d^%f*d%d%d%d^%f*%f)*x%d%d%d+",
         i,j,k,beta,i,j,k,gammaT,tr,i,j,k);
  fprintf(mus,"(%f*(%f-2*d%d%d%d)\n",z11,D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"*v%d%d%d^(-1)*f%d%d%d^(-1)",i,j,k,i,j,k);
  }
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"+%f*10^(-3)*(%f-2*d%d%d%d)\n",z12,D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"*v%d%d%d^%f*f%d%d%d^%f*d%d%d%d^%f*%f)*x%d%d%d\n",
         i,j,k,alpha,i,j,k,beta,i,j,k,gammaT,tr,i,j,k);
 }
```

```
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"+(%f*(%f-2*d%d%d%d",z11,D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"-2*d%d%d%d)",i,j,k);
  }
 }

 for (j=3;j<=Jk;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"*v%d%d%d^(-1)*f%d%d%d^(-1)+%f*10^(-3)\n",
         i,j,k,i,j,k,z12);
  }
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"*(%f-2*d%d%d%d",D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"-2*d%d%d%d)",i,j,k);
  }
 }

 for (j=3;j<=Jk;j++)
```

```
  {
   for (k=3;k<=Q;k++)
   {
   fprintf(mus,"*v%d%d%d^%f\n",i,j,k,alpha);
   fprintf(mus,"*f%d%d%d^%f*d%d%d%d^%f*(%f+%f/y%d%d%d))",
          i,j,k,beta,i,j,k,gammaT,tr,wt,i,j,k);
   fprintf(mus,"*x%d%d%d<=Z;\n",i,j,k);
   }
  }
 }

/*
This section generates cutting speed, feed rate, cutting
force and cutting power constraints
*/

for (i=1;i<=P;i++)
{
 for (j=1;j<=Jk;j++)
 {
  for (k=1;k<=Q;k++)
  {
  fprintf(mus,"v%d%d%d>=%f;\n",i,j,k,v_l[j]);
  fprintf(mus,"v%d%d%d<=%f;\n",i,j,k,v_u[j]);
  fprintf(mus,"f%d%d%d>=%f;\n",i,j,k,f_l[j]);
  fprintf(mus,"f%d%d%d<=%f;\n",i,j,k,f_u[j]);
  fprintf(mus,"%f*f%d%d%d^%f*d%d%d%d^%f<=%f;\n",kF,i,j,k,betaF,
          i,j,k,gammaF,Fc_u);
  fprintf(mus,"%f*10^2*v%d%d%d*f%d%d%d^%f*d%d%d%d^%f<=%f*10^2;\n",
          kP,i,j,k,i,j,k,betaF,i,j,k,gammaF,Pc_u);
  }
 }
}

/*
This section generates depth of cut constraints
*/

for (i=1;i<=P;i++)
{
 for (j=1;j<=Jk;j++)
 {
  for (k=1;k<=Q;k++)
  {
```

```
   if ((i==k) && (j==Jk))
   fprintf(mus,"d%d%d%d>=%f;\n",i,j,k,d_l[j]);
   fprintf(mus,"d%d%d%d<=%f;\n",i,j,k,d_u[j]);
   }
  }
 }

 /*
 This section generates the sum of depth of cut constraints
 */

 for (i=1;i<=P-1;i++)
 {
  for (j=1;j<=Jk;j++)
  {
   for (k=1;k<=Q-2;k++)
   {
   fprintf(mus,"d%d%d%d+",i,j,k);
   }
  }
 }

 for (i=3;i<=P;i++)
 {
  for (j=1;j<=Jk-1;j++)
  {
   for (k=1;k<=Q-2;k++)
   {
   fprintf(mus,"d%d%d%d+",i,j,k);
   }
  }
 }

 for (i=3;i<=P;i++)
 {
  for (j=3;j<=Jk;j++)
  {
   for (k=1;k<=Q-2;k++)
   {
   fprintf(mus,"d%d%d%d=%f;\n",i,j,k,dtotal[k]);
   }
  }
 }
```

```
 for (i=1;i<=P-1;i++)
 {
  for (j=1;j<=Jk;j++)
  {
   for (k=2;k<=Q-1;k++)
   {
   fprintf(mus,"d%d%d%d+",i,j,k);
   }
  }
 }

 for (i=3;i<=P;i++)
 {
  for (j=1;j<=Jk-1;j++)
  {
   for (k=2;k<=Q-1;k++)
   {
   fprintf(mus,"d%d%d%d+",i,j,k);
   }
  }
 }

 for (i=3;i<=P;i++)
 {
  for (j=3;j<=Jk;j++)
  {
   for (k=2;k<=Q-1;k++)
   {
   fprintf(mus,"d%d%d%d=%f;\n",i,j,k,dtotal[k]);
   }
  }
 }

 for (i=1;i<=P-1;i++)
 {
  for (j=1;j<=Jk;j++)
  {
   for (k=3;k<=Q;k++)
   {
   fprintf(mus,"d%d%d%d+",i,j,k);
   }
  }
 }
```

```
for (i=3;i<=P;i++)
{
 for (j=1;j<=Jk-1;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"d%d%d%d+",i,j,k);
  }
 }
}

for (i=3;i<=P;i++)
{
 for (j=3;j<=Jk;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"d%d%d%d=%f;\n",i,j,k,dtotal[k]);
  }
 }
}

for (i=1;i<=P-1;i++)
{
 for (j=1;j<=Jk;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"d%d%d%d*x%d%d%d+",i,j,k,i,j,k);
  }
 }
}

for (i=3;i<=P;i++)
{
 for (j=1;j<=Jk-1;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"d%d%d%d*x%d%d%d+",i,j,k,i,j,k);
  }
 }
}
```

```
 for (i=3;i<=P;i++)
 {
  for (j=3;j<=Jk;j++)
  {
   for (k=1;k<=Q-2;k++)
   {
   fprintf(mus,"d%d%d%d*x%d%d%d=%f;\n",i,j,k,i,j,k,dtotal[k]);
   }
  }
 }

 for (i=1;i<=P-1;i++)
 {
  for (j=1;j<=Jk;j++)
  {
   for (k=2;k<=Q-1;k++)
   {
   fprintf(mus,"d%d%d%d*x%d%d%d+",i,j,k,i,j,k);
   }
  }
 }

 for (i=3;i<=P;i++)
 {
  for (j=1;j<=Jk-1;j++)
  {
   for (k=2;k<=Q-1;k++)
   {
   fprintf(mus,"d%d%d%d*x%d%d%d+",i,j,k,i,j,k);
   }
  }
 }

 for (i=3;i<=P;i++)
 {
  for (j=3;j<=Jk;j++)
  {
   for (k=2;k<=Q-1;k++)
   {
   fprintf(mus,"d%d%d%d*x%d%d%d=%f;\n",i,j,k,i,j,k,dtotal[k]);
   }
  }
 }
```

```
for (i=1;i<=P-1;i++)
{
 for (j=1;j<=Jk;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"d%d%d%d*x%d%d%d+",i,j,k,i,j,k);
  }
 }
}

for (i=3;i<=P;i++)
{
 for (j=1;j<=Jk-1;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"d%d%d%d*x%d%d%d+",i,j,k,i,j,k);
  }
 }
}

for (i=3;i<=P;i++)
{
 for (j=3;j<=Jk;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"d%d%d%d*x%d%d%d=%f;\n",i,j,k,i,j,k,dtotal[k]);
  }
 }
}


/*
This section generates the constraints such that each
operation is processed by only one machine
*/

for (i=1;i<=P-1;i++)
{
   for (j=1;j<=Jk-2;j++)
   {
       for (k=1;k<=Q-2;k++)
```

```
        {
        fprintf(mus,"x%d%d%d+",i,j,k);
        }
    }
}

for (i=3;i<=P;i++)
{
    for (j=1;j<=Jk-2;j++)
    {
        for (k=1;k<=Q-2;k++)
        {
        fprintf(mus,"x%d%d%d=1;\n",i,j,k);
        }
    }
}

for (i=1;i<=P-1;i++)
{
    for (j=2;j<=Jk-1;j++)
    {
        for (k=1;k<=Q-2;k++)
        {
        fprintf(mus,"x%d%d%d+",i,j,k);
        }
    }
}

for (i=3;i<=P;i++)
{
    for (j=2;j<=Jk-1;j++)
    {
        for (k=1;k<=Q-2;k++)
        {
        fprintf(mus,"x%d%d%d=1;\n",i,j,k);
        }
    }
}

for (i=1;i<=P-1;i++)
{
    for (j=3;j<=Jk;j++)
    {
        for (k=1;k<=Q-2;k++)
```

```
        {
        fprintf(mus,"x%d%d%d+",i,j,k);
        }
    }
}

for (i=3;i<=P;i++)
{
    for (j=3;j<=Jk;j++)
    {
        for (k=1;k<=Q-2;k++)
        {
        fprintf(mus,"x%d%d%d=1;\n",i,j,k);
        }
    }
}

for (i=1;i<=P-1;i++)
{
    for (j=1;j<=Jk-2;j++)
    {
        for (k=2;k<=Q-1;k++)
        {
        fprintf(mus,"x%d%d%d+",i,j,k);
        }
    }
}

for (i=3;i<=P;i++)
{
    for (j=1;j<=Jk-2;j++)
    {
        for (k=2;k<=Q-1;k++)
        {
        fprintf(mus,"x%d%d%d=1;\n",i,j,k);
        }
    }
}

for (i=1;i<=P-1;i++)
{
    for (j=2;j<=Jk-1;j++)
    {
        for (k=2;k<=Q-1;k++)
```

```
        {
        fprintf(mus,"x%d%d%d+",i,j,k);
        }
    }
}

for (i=3;i<=P;i++)
{
    for (j=2;j<=Jk-1;j++)
    {
        for (k=2;k<=Q-1;k++)
        {
        fprintf(mus,"x%d%d%d=1;\n",i,j,k);
        }
    }
}

for (i=1;i<=P-1;i++)
{
    for (j=3;j<=Jk;j++)
    {
        for (k=2;k<=Q-1;k++)
        {
        fprintf(mus,"x%d%d%d+",i,j,k);
        }
    }
}

for (i=3;i<=P;i++)
{
    for (j=3;j<=Jk;j++)
    {
        for (k=2;k<=Q-1;k++)
        {
        fprintf(mus,"x%d%d%d=1;\n",i,j,k);
        }
    }
}

for (i=1;i<=P-1;i++)
{
    for (j=1;j<=Jk-2;j++)
    {
        for (k=3;k<=Q;k++)
```

```
            {
            fprintf(mus,"x%d%d%d+",i,j,k);
            }
        }
    }

    for (i=3;i<=P;i++)
    {
        for (j=1;j<=Jk-2;j++)
        {
            for (k=3;k<=Q;k++)
            {
            fprintf(mus,"x%d%d%d=1;\n",i,j,k);
            }
        }
    }

    for (i=1;i<=P-1;i++)
    {
        for (j=2;j<=Jk-1;j++)
        {
            for (k=3;k<=Q;k++)
            {
            fprintf(mus,"x%d%d%d+",i,j,k);
            }
        }
    }

    for (i=3;i<=P;i++)
    {
        for (j=2;j<=Jk-1;j++)
        {
            for (k=3;k<=Q;k++)
            {
            fprintf(mus,"x%d%d%d=1;\n",i,j,k);
            }
        }
    }

    for (i=1;i<=P-1;i++)
    {
        for (j=3;j<=Jk;j++)
        {
            for (k=3;k<=Q;k++)
```

```
        {
        fprintf(mus,"x%d%d%d+",i,j,k);
        }
    }
}

for (i=3;i<=P;i++)
{
    for (j=3;j<=Jk;j++)
    {
        for (k=3;k<=Q;k++)
        {
        fprintf(mus,"x%d%d%d=1;\n",i,j,k);
        }
    }
}

/*
This section generates the constraints such that the last pass of each feature is
always selected and each machine should perform at least one pass
*/

for (i=1;i<=P;i++)
{
    for (j=1;j<=Jk;j++)
    {
        for (k=1;k<=Q;k++)
        {
        if ((i==k) && (j==Jk))
        fprintf(mus,"x%d%d%d=1;\n",i,j,k);
        }
    }
}

/*
This section generates dimension deviation constraints
*/

for (i=1;i<=P;i++)
{
    for (j=3;j<=Jk;j++)
    {
        for (k=1;k<=Q;k++)
        {
```

```
        fprintf(mus,"y%d%d%d>=0;\n",i,j,k);
        fprintf(mus,"y%d%d%d<=%f;\n",i,j,k,delta[k]);
        }
    }
}

/*
This section generates surface finish constraints
*/

for (i=1;i<=P;i++)
{
 for (j=3;j<=Jk;j++)
 {
  for (k=1;k<=Q;k++)
  {
  fprintf(mus,"%f*v%d%d%d^%f*f%d%d%d^%f*d%d%d%d^%f<=%f;\n",kS,
        i,j,k,alphaS,i,j,k,betaS,i,j,k,gammaS,SF_u[k]);
  }
 }
}

fprintf(mus,"END\n");
fprintf(mus,"BAT\n");
return 0;

}
```

Appendix H

LINGO-format-input for the example problem of Model 2.4

```
MODEL
min=(424.114655*v111^(-1)*f111^(-1)+0.000090*v111^0.700000
*f111^0.550000*d111^1.220000*7.000000)*x111+(2.827431*(150.000000-2*d111)
*v121^(-1)*f121^(-1)+0.000600*10^(-3)*(150.000000-2*d111)
*v121^0.700000*f121^0.550000*d121^1.220000*7.000000)*x121
+(2.827431*(150.000000-2*d111-2*d121)*v131^(-1)*f131^(-1)+0.000600*10^(-3)
*(150.000000-2*d111-2*d121)*v131^0.700000
*f131^0.550000*d131^1.220000*(7.000000+0.060000/y131+23.026567*y131))*x131
+(131.946777*v112^(-1)*f112^(-1)+0.000028*v112^0.700000
*f112^0.550000*d112^1.220000*7.000000)*x112+(0.942477*(140.000000-2*d112)
*v122^(-1)*f122^(-1)+0.000200*10^(-3)*(140.000000-2*d112)
*v122^0.700000*f122^0.550000*d122^1.220000*7.000000)*x122
+(0.942477*(140.000000-2*d112-2*d122)*v132^(-1)*f132^(-1)+0.000200*10^(-3)
*(140.000000-2*d112-2*d122)*v132^0.700000
*f132^0.550000*d132^1.220000*(7.000000+0.060000/y132+23.026567*y132))*x132
+(197.920166*v113^(-1)*f113^(-1)+0.000042*v113^0.700000
*f113^0.550000*d113^1.220000*7.000000)*x113+(1.413715*(140.000000-2*d113)
*v123^(-1)*f123^(-1)+0.000300*10^(-3)*(140.000000-2*d113)
*v123^0.700000*f123^0.550000*d123^1.220000*7.000000)*x123
+(1.413715*(140.000000-2*d113-2*d123)*v133^(-1)*f133^(-1)+0.000300*10^(-3)
*(140.000000-2*d113-2*d123)*v133^0.700000
*f133^0.550000*d133^1.220000*(7.000000+0.060000/y133+23.026567*y133))*x133+
(424.114655*v211^(-1)*f211^(-1)+0.000090*v211^0.700000
*f211^0.550000*d211^1.220000*7.000000)*x211+(2.827431*(150.000000-2*d211)
*v221^(-1)*f221^(-1)+0.000600*10^(-3)*(150.000000-2*d211)
*v221^0.700000*f221^0.550000*d221^1.220000*7.000000)*x221
+(2.827431*(150.000000-2*d211-2*d221)*v231^(-1)*f231^(-1)+0.000600*10^(-3)
*(150.000000-2*d211-2*d221)*v231^0.700000
*f231^0.550000*d231^1.220000*(7.000000+0.060000/y231+23.026567*y231))*x231
+(131.946777*v212^(-1)*f212^(-1)+0.000028*v212^0.700000
*f212^0.550000*d212^1.220000*7.000000)*x212+(0.942477*(140.000000-2*d212)
*v222^(-1)*f222^(-1)+0.000200*10^(-3)*(140.000000-2*d212)
*v222^0.700000*f222^0.550000*d222^1.220000*7.000000)*x222
+(0.942477*(140.000000-2*d212-2*d222)*v232^(-1)*f232^(-1)+0.000200*10^(-3)
```

```
*(140.000000-2*d212-2*d222)*v232^0.700000
*f232^0.550000*d232^1.220000*(7.000000+0.060000/y232+23.026567*y232))*x232
+(197.920166*v213^(-1)*f213^(-1)+0.000042*v213^0.700000
*f213^0.550000*d213^1.220000*7.000000)*x213+(1.413715*(140.000000-2*d213)
*v223^(-1)*f223^(-1)+0.000300*10^(-3)*(140.000000-2*d213)
*v223^0.700000*f223^0.550000*d223^1.220000*7.000000)*x223
+(1.413715*(140.000000-2*d213-2*d223)*v233^(-1)*f233^(-1)+0.000300*10^(-3)
*(140.000000-2*d213-2*d223)*v233^0.700000
*f233^0.550000*d233^1.220000*(7.000000+0.060000/y233+23.026567*y233))*x233+
(424.114655*v311^(-1)*f311^(-1)+0.000090*v311^0.700000
*f311^0.550000*d311^1.220000*7.000000)*x311+(2.827431*(150.000000-2*d311)
*v321^(-1)*f321^(-1)+0.000600*10^(-3)*(150.000000-2*d311)
*v321^0.700000*f321^0.550000*d321^1.220000*7.000000)*x321
+(2.827431*(150.000000-2*d311-2*d321)*v331^(-1)*f331^(-1)+0.000600*10^(-3)
*(150.000000-2*d311-2*d321)*v331^0.700000
*f331^0.550000*d331^1.220000*(7.000000+0.060000/y331+23.026567*y331))*x331
+(131.946777*v312^(-1)*f312^(-1)+0.000028*v312^0.700000
*f312^0.550000*d312^1.220000*7.000000)*x312+(0.942477*(140.000000-2*d312)
*v322^(-1)*f322^(-1)+0.000200*10^(-3)*(140.000000-2*d312)
*v322^0.700000*f322^0.550000*d322^1.220000*7.000000)*x322
+(0.942477*(140.000000-2*d312-2*d322)*v332^(-1)*f332^(-1)+0.000200*10^(-3)
*(140.000000-2*d312-2*d322)*v332^0.700000
*f332^0.550000*d332^1.220000*(7.000000+0.060000/y332+23.026567*y332))*x332
+(197.920166*v313^(-1)*f313^(-1)+0.000042*v313^0.700000
*f313^0.550000*d313^1.220000*7.000000)*x313+(1.413715*(140.000000-2*d313)
*v323^(-1)*f323^(-1)+0.000300*10^(-3)*(140.000000-2*d313)
*v323^0.700000*f323^0.550000*d323^1.220000*7.000000)*x323
+(1.413715*(140.000000-2*d313-2*d323)*v333^(-1)*f333^(-1)+0.000300*10^(-3)
*(140.000000-2*d313-2*d323)*v333^0.700000
*f333^0.550000*d333^1.220000*(7.000000+0.060000/y333+23.026567*y333))*x333;
(141.371552*v111^(-1)*f111^(-1)+0.000090*v111^0.700000
*f111^0.550000*d111^1.220000*0.500000)*x111+(0.942477*(150.000000-2*d111)
*v121^(-1)*f121^(-1)+0.000600*10^(-3)*(150.000000-2*d111)
*v121^0.700000*f121^0.550000*d121^1.220000*0.500000)*x121
+(0.942477*(150.000000-2*d111-2*d121)*v131^(-1)*f131^(-1)+0.000600*10^(-3)
*(150.000000-2*d111-2*d121)*v131^0.700000
*f131^0.550000*d131^1.220000*(0.500000+0.020000/y131))*x131
+(43.982262*v112^(-1)*f112^(-1)+0.000028*v112^0.700000
*f112^0.550000*d112^1.220000*0.500000)*x112+(0.314159*(140.000000-2*d112)
*v122^(-1)*f122^(-1)+0.000200*10^(-3)*(140.000000-2*d112)
*v122^0.700000*f122^0.550000*d122^1.220000*0.500000)*x122
+(0.314159*(140.000000-2*d112-2*d122)*v132^(-1)*f132^(-1)+0.000200*10^(-3)
*(140.000000-2*d112-2*d122)*v132^0.700000
*f132^0.550000*d132^1.220000*(0.500000+0.020000/y132))*x132
```

```
+(65.973389*v113^(-1)*f113^(-1)+0.000042*v113^0.700000
*f113^0.550000*d113^1.220000*0.500000)*x113+(0.471238*(140.000000-2*d113)
*v123^(-1)*f123^(-1)+0.000300*10^(-3)*(140.000000-2*d113)
*v123^0.700000*f123^0.550000*d123^1.220000*0.500000)*x123
+(0.471238*(140.000000-2*d113-2*d123)*v133^(-1)*f133^(-1)+0.000300*10^(-3)
*(140.000000-2*d113-2*d123)*v133^0.700000
*f133^0.550000*d133^1.220000*(0.500000+0.020000/y133))*x133<=Z;
(141.371552*v211^(-1)*f211^(-1)+0.000090*v211^0.700000
*f211^0.550000*d211^1.220000*0.500000)*x211+(0.942477*(150.000000-2*d211)
*v221^(-1)*f221^(-1)+0.000600*10^(-3)*(150.000000-2*d211)
*v221^0.700000*f221^0.550000*d221^1.220000*0.500000)*x221
+(0.942477*(150.000000-2*d211-2*d221)*v231^(-1)*f231^(-1)+0.000600*10^(-3)
*(150.000000-2*d211-2*d221)*v231^0.700000
*f231^0.550000*d231^1.220000*(0.500000+0.020000/y231))*x231
+(43.982262*v212^(-1)*f212^(-1)+0.000028*v212^0.700000
*f212^0.550000*d212^1.220000*0.500000)*x212+(0.314159*(140.000000-2*d212)
*v222^(-1)*f222^(-1)+0.000200*10^(-3)*(140.000000-2*d212)
*v222^0.700000*f222^0.550000*d222^1.220000*0.500000)*x222
+(0.314159*(140.000000-2*d212-2*d222)*v232^(-1)*f232^(-1)+0.000200*10^(-3)
*(140.000000-2*d212-2*d222)*v232^0.700000
*f232^0.550000*d232^1.220000*(0.500000+0.020000/y232))*x232
+(65.973389*v213^(-1)*f213^(-1)+0.000042*v213^0.700000
*f213^0.550000*d213^1.220000*0.500000)*x213+(0.471238*(140.000000-2*d213)
*v223^(-1)*f223^(-1)+0.000300*10^(-3)*(140.000000-2*d213)
*v223^0.700000*f223^0.550000*d223^1.220000*0.500000)*x223
+(0.471238*(140.000000-2*d213-2*d223)*v233^(-1)*f233^(-1)+0.000300*10^(-3)
*(140.000000-2*d213-2*d223)*v233^0.700000
*f233^0.550000*d233^1.220000*(0.500000+0.020000/y233))*x233<=Z;
(141.371552*v311^(-1)*f311^(-1)+0.000090*v311^0.700000
*f311^0.550000*d311^1.220000*0.500000)*x311+(0.942477*(150.000000-2*d311)
*v321^(-1)*f321^(-1)+0.000600*10^(-3)*(150.000000-2*d311)
*v321^0.700000*f321^0.550000*d321^1.220000*0.500000)*x321
+(0.942477*(150.000000-2*d311-2*d321)*v331^(-1)*f331^(-1)+0.000600*10^(-3)
*(150.000000-2*d311-2*d321)*v331^0.700000
*f331^0.550000*d331^1.220000*(0.500000+0.020000/y331))*x331
+(43.982262*v312^(-1)*f312^(-1)+0.000028*v312^0.700000
*f312^0.550000*d312^1.220000*0.500000)*x312+(0.314159*(140.000000-2*d312)
*v322^(-1)*f322^(-1)+0.000200*10^(-3)*(140.000000-2*d312)
*v322^0.700000*f322^0.550000*d322^1.220000*0.500000)*x322
+(0.314159*(140.000000-2*d312-2*d322)*v332^(-1)*f332^(-1)+0.000200*10^(-3)
*(140.000000-2*d312-2*d322)*v332^0.700000
*f332^0.550000*d332^1.220000*(0.500000+0.020000/y332))*x332
+(65.973389*v313^(-1)*f313^(-1)+0.000042*v313^0.700000
*f313^0.550000*d313^1.220000*0.500000)*x313+(0.471238*(140.000000-2*d313)
```

```
*v323^(-1)*f323^(-1)+0.000300*10^(-3)*(140.000000-2*d313)
*v323^0.700000*f323^0.550000*d323^1.220000*0.500000)*x323
+(0.471238*(140.000000-2*d313-2*d323)*v333^(-1)*f333^(-1)+0.000300*10^(-3)
*(140.000000-2*d313-2*d323)*v333^0.700000
*f333^0.550000*d333^1.220000*(0.500000+0.020000/y333))*x333<=Z;
v111>=90.000000;
v111<=120.000000;
f111>=0.800000;
f111<=2.000000;
1.380000*f111^1.180000*d111^1.260000<=20.000000;
0.000282*10^2*v111*f111^1.180000*d111^1.260000<=2.000000*10^2;
v112>=90.000000;
v112<=120.000000;
f112>=0.800000;
f112<=2.000000;
1.380000*f112^1.180000*d112^1.260000<=20.000000;
0.000282*10^2*v112*f112^1.180000*d112^1.260000<=2.000000*10^2;
v113>=90.000000;
v113<=120.000000;
f113>=0.800000;
f113<=2.000000;
1.380000*f113^1.180000*d113^1.260000<=20.000000;
0.000282*10^2*v113*f113^1.180000*d113^1.260000<=2.000000*10^2;
v121>=90.000000;
v121<=120.000000;
f121>=0.800000;
f121<=2.000000;
1.380000*f121^1.180000*d121^1.260000<=20.000000;
0.000282*10^2*v121*f121^1.180000*d121^1.260000<=2.000000*10^2;
v122>=90.000000;
v122<=120.000000;
f122>=0.800000;
f122<=2.000000;
1.380000*f122^1.180000*d122^1.260000<=20.000000;
0.000282*10^2*v122*f122^1.180000*d122^1.260000<=2.000000*10^2;
v123>=90.000000;
v123<=120.000000;
f123>=0.800000;
f123<=2.000000;
1.380000*f123^1.180000*d123^1.260000<=20.000000;
0.000282*10^2*v123*f123^1.180000*d123^1.260000<=2.000000*10^2;
v131>=168.000000;
v131<=210.000000;
f131>=0.130000;
```

f131<=0.500000;
1.380000*f131^1.180000*d131^1.260000<=20.000000;
0.000282*10^2*v131*f131^1.180000*d131^1.260000<=2.000000*10^2;
v132>=168.000000;
v132<=210.000000;
f132>=0.130000;
f132<=0.500000;
1.380000*f132^1.180000*d132^1.260000<=20.000000;
0.000282*10^2*v132*f132^1.180000*d132^1.260000<=2.000000*10^2;
v133>=168.000000;
v133<=210.000000;
f133>=0.130000;
f133<=0.500000;
1.380000*f133^1.180000*d133^1.260000<=20.000000;
0.000282*10^2*v133*f133^1.180000*d133^1.260000<=2.000000*10^2;
v211>=90.000000;
v211<=120.000000;
f211>=0.800000;
f211<=2.000000;
1.380000*f211^1.180000*d211^1.260000<=20.000000;
0.000282*10^2*v211*f211^1.180000*d211^1.260000<=2.000000*10^2;
v212>=90.000000;
v212<=120.000000;
f212>=0.800000;
f212<=2.000000;
1.380000*f212^1.180000*d212^1.260000<=20.000000;
0.000282*10^2*v212*f212^1.180000*d212^1.260000<=2.000000*10^2;
v213>=90.000000;
v213<=120.000000;
f213>=0.800000;
f213<=2.000000;
1.380000*f213^1.180000*d213^1.260000<=20.000000;
0.000282*10^2*v213*f213^1.180000*d213^1.260000<=2.000000*10^2;
v221>=90.000000;
v221<=120.000000;
f221>=0.800000;
f221<=2.000000;
1.380000*f221^1.180000*d221^1.260000<=20.000000;
0.000282*10^2*v221*f221^1.180000*d221^1.260000<=2.000000*10^2;
v222>=90.000000;
v222<=120.000000;
f222>=0.800000;
f222<=2.000000;
1.380000*f222^1.180000*d222^1.260000<=20.000000;

0.000282*10^2*v222*f222^1.180000*d222^1.260000<=2.000000*10^2;
v223>=90.000000;
v223<=120.000000;
f223>=0.800000;
f223<=2.000000;
1.380000*f223^1.180000*d223^1.260000<=20.000000;
0.000282*10^2*v223*f223^1.180000*d223^1.260000<=2.000000*10^2;
v231>=168.000000;
v231<=210.000000;
f231>=0.130000;
f231<=0.500000;
1.380000*f231^1.180000*d231^1.260000<=20.000000;
0.000282*10^2*v231*f231^1.180000*d231^1.260000<=2.000000*10^2;
v232>=168.000000;
v232<=210.000000;
f232>=0.130000;
f232<=0.500000;
1.380000*f232^1.180000*d232^1.260000<=20.000000;
0.000282*10^2*v232*f232^1.180000*d232^1.260000<=2.000000*10^2;
v233>=168.000000;
v233<=210.000000;
f233>=0.130000;
f233<=0.500000;
1.380000*f233^1.180000*d233^1.260000<=20.000000;
0.000282*10^2*v233*f233^1.180000*d233^1.260000<=2.000000*10^2;
v311>=90.000000;
v311<=120.000000;
f311>=0.800000;
f311<=2.000000;
1.380000*f311^1.180000*d311^1.260000<=20.000000;
0.000282*10^2*v311*f311^1.180000*d311^1.260000<=2.000000*10^2;
v312>=90.000000;
v312<=120.000000;
f312>=0.800000;
f312<=2.000000;
1.380000*f312^1.180000*d312^1.260000<=20.000000;
0.000282*10^2*v312*f312^1.180000*d312^1.260000<=2.000000*10^2;
v313>=90.000000;
v313<=120.000000;
f313>=0.800000;
f313<=2.000000;
1.380000*f313^1.180000*d313^1.260000<=20.000000;
0.000282*10^2*v313*f313^1.180000*d313^1.260000<=2.000000*10^2;
v321>=90.000000;

v321 < =120.000000;
f321 > =0.800000;
f321 < =2.000000;
1.380000*f321^1.180000*d321^1.260000 < =20.000000;
0.000282*10^2*v321*f321^1.180000*d321^1.260000 < =2.000000*10^2;
v322 > =90.000000;
v322 < =120.000000;
f322 > =0.800000;
f322 < =2.000000;
1.380000*f322^1.180000*d322^1.260000 < =20.000000;
0.000282*10^2*v322*f322^1.180000*d322^1.260000 < =2.000000*10^2;
v323 > =90.000000;
v323 < =120.000000;
f323 > =0.800000;
f323 < =2.000000;
1.380000*f323^1.180000*d323^1.260000 < =20.000000;
0.000282*10^2*v323*f323^1.180000*d323^1.260000 < =2.000000*10^2;
v331 > =168.000000;
v331 < =210.000000;
f331 > =0.130000;
f331 < =0.500000;
1.380000*f331^1.180000*d331^1.260000 < =20.000000;
0.000282*10^2*v331*f331^1.180000*d331^1.260000 < =2.000000*10^2;
v332 > =168.000000;
v332 < =210.000000;
f332 > =0.130000;
f332 < =0.500000;
1.380000*f332^1.180000*d332^1.260000 < =20.000000;
0.000282*10^2*v332*f332^1.180000*d332^1.260000 < =2.000000*10^2;
v333 > =168.000000;
v333 < =210.000000;
f333 > =0.130000;
f333 < =0.500000;
1.380000*f333^1.180000*d333^1.260000 < =20.000000;
0.000282*10^2*v333*f333^1.180000*d333^1.260000 < =2.000000*10^2;
d111 < =5.000000;
d112 < =5.000000;
d113 < =5.000000;
d121 < =5.000000;
d122 < =5.000000;
d123 < =5.000000;
d131 > =0.300000;
d131 < =1.000000;
d132 < =1.000000;

```
d133<=1.000000;
d211<=5.000000;
d212<=5.000000;
d213<=5.000000;
d221<=5.000000;
d222<=5.000000;
d223<=5.000000;
d231<=1.000000;
d232>=0.300000;
d232<=1.000000;
d233<=1.000000;
d311<=5.000000;
d312<=5.000000;
d313<=5.000000;
d321<=5.000000;
d322<=5.000000;
d323<=5.000000;
d331<=1.000000;
d332<=1.000000;
d333>=0.300000;
d333<=1.000000;
d111+d121+d131+d211+d221+d231+d311+d321+d331=5.000000;
d112+d122+d132+d212+d222+d232+d312+d322+d332=8.000000;
d113+d123+d133+d213+d223+d233+d313+d323+d333=10.000000;
d111*x111+d121*x121+d131*x131+d211*x211+d221*x221+d231*x231+
d311*x311+d321*x321+d331*x331=5.000000;
d112*x112+d122*x122+d132*x132+d212*x212+d222*x222+d232*x232+
d312*x312+d322*x322+d332*x332=8.000000;
d113*x113+d123*x123+d133*x133+d213*x213+d223*x223+d233*x233+
d313*x313+d323*x323+d333*x333=10.000000;
x111+x211+x311=1;
x121+x221+x321=1;
x131+x231+x331=1;
x112+x212+x312=1;
x122+x222+x322=1;
x132+x232+x332=1;
x113+x213+x313=1;
x123+x223+x323=1;
x133+x233+x333=1;
x131=1;
x232=1;
x333=1;
y131>=0;
y131<=0.065900;
```

```
y132 > =0;
y132 < =0.065900;
y133 > =0;
y133 < =0.065900;
y231 > =0;
y231 < =0.065900;
y232 > =0;
y232 < =0.065900;
y233 > =0;
y233 < =0.065900;
y331 > =0;
y331 < =0.065900;
y332 > =0;
y332 < =0.065900;
y333 > =0;
y333 < =0.065900;
1.170000*v131^-0.250000*f131^0.720000*d131^0.230000 < =1.600000;
1.170000*v132^-0.250000*f132^0.720000*d132^0.230000 < =1.600000;
1.170000*v133^-0.250000*f133^0.720000*d133^0.230000 < =1.600000;
1.170000*v231^-0.250000*f231^0.720000*d231^0.230000 < =1.600000;
1.170000*v232^-0.250000*f232^0.720000*d232^0.230000 < =1.600000;
1.170000*v233^-0.250000*f233^0.720000*d233^0.230000 < =1.600000;
1.170000*v331^-0.250000*f331^0.720000*d331^0.230000 < =1.600000;
1.170000*v332^-0.250000*f332^0.720000*d332^0.230000 < =1.600000;
1.170000*v333^-0.250000*f333^0.720000*d333^0.230000 < =1.600000;
END
BAT
```

Appendix I

C code to generate LINGO-format-input for Model 2.5

```
/*
This program is used to generate the LINGO-format-input model
*/

# include <stdio.h>
# include <conio.h>
# include <stdlib.h>
# include <math.h>

# define pi 3.14159
# define c 6120

/*
P = number of machines
Jk = number of passes of feature k
Q = number of features

*/

int i,j,k,P,Jk,Q,R;

float alpha,beta,z1,z2,z3,z4,z5,z6,z7,z8,z9,
      z10,z11,z12,wt,kP;

float  kT=1570000.00,
       alphaT=1.70,
       betaT=1.55,
       gammaT=1.22,
       tr=0.5,
       w=0.1,
       ta=0.2,
       betaF=1.18,
       gammaF=1.26,
       kF=1.38,
```

```
        Fc_u=20.0,
        eta=0.8,
        Pc_u=2.0,
        kS=1.17,
        alphaS=-0.25,
        betaS=0.72,
        gammaS=0.23;

static float D0[4] = {0.0,150.0,140.0,140.0};
static float L[4] = {0.0,300.0,100.0,150.0};
static float dtotal[4] = {0.0,5.0,8.0,10.0};
static float SF_u[4] = {0.0,1.6,1.6,1.6};
static float delta[4] = {0.0,0.0659,0.0659,0.0659};
static float v_l[4] = {0.0,90.0,90.0,168.0};
static float v_u[4] = {0.0,120.0,120.0,210.0};
static float f_l[4] = {0.0,0.8,0.8,0.13};
static float f_u[4] = {0.0,2.0,2.0,0.50};
static float d_l[4] = {0.0,0.0,0.0,0.3};
static float d_u[4] = {0.0,5.0,5.0,1.0};

FILE *mus;

main()

{

clrscr();

P=3;Jk=3;Q=3;
R=1000;
alpha=alphaT-1;
beta=betaT-1;
z1=pi*D0[1]*L[1]/R;
z2=pi*D0[1]*L[1]/(R*kT);
z3=pi*L[1]/R;
z4=pi*L[1]/kT;
z5=pi*D0[2]*L[2]/R;
z6=pi*D0[2]*L[2]/(R*kT);
z7=pi*L[2]/R;
z8=pi*L[2]/kT;
z9=pi*D0[3]*L[3]/R;
z10=pi*D0[3]*L[3]/(R*kT);
z11=pi*L[3]/R;
z12=pi*L[3]/kT;
```

```
wt=w*ta;
kP=kF/(eta*c);

/*
The following statement specifies the name for
the generated file
*/

mus = fopen("mod25","w");

/*
The following section generates the objective function
*/

fprintf(mus,"MODEL\n");
fprintf(mus,"min=Z;\n");

/*
This section generates production time constraints
*/

for (i=1;i<=P;i++)
{
 for (j=1;j<=Jk-2;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"(%f*v%d%d%d^(-1)*f%d%d%d^(-1)+%f*v%d%d%d^%f\n",
        z1,i,j,k,i,j,k,z2,i,j,k,alpha);
  fprintf(mus,"*f%d%d%d^%f*d%d%d%d^%f*%f)*x%d%d%d+",
        i,j,k,beta,i,j,k,gammaT,tr,i,j,k);
  fprintf(mus,"(%f*(%f-2*d%d%d%d)\n",z3,D0[k],i,j,k);
  }
}

for (j=2;j<=Jk-1;j++)
{
 for (k=1;k<=Q-2;k++)
 {
 fprintf(mus,"*v%d%d%d^(-1)*f%d%d%d^(-1)",i,j,k,i,j,k);
 }
}

for (j=1;j<=Jk-2;j++)
```

```
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"+%f*10^(-3)*(%f-2*d%d%d%d)\n",z4,D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"*v%d%d%d^%f*f%d%d%d^%f*d%d%d%d^%f*%f)*x%d%d%d\n",
        i,j,k,alpha,i,j,k,beta,i,j,k,gammaT,tr,i,j,k);
  }
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"+(%f*(%f-2*d%d%d%d",z3,D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"-2*d%d%d%d)",i,j,k);
  }
 }

 for (j=3;j<=Jk;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"*v%d%d%d^(-1)*f%d%d%d^(-1)+%f*10^(-3)\n",
        i,j,k,i,j,k,z4);
  }
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=1;k<=Q-2;k++)
```

```
  {
  fprintf(mus,"*(%f-2*d%d%d%d",D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"-2*d%d%d%d)",i,j,k);
  }
 }

 for (j=3;j<=Jk;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"*v%d%d%d^%f\n",i,j,k,alpha);
  fprintf(mus,"*f%d%d%d^%f*d%d%d%d^%f*(%f+%f/y%d%d%d))*x%d%d%d\n",
        i,j,k,beta,i,j,k,gammaT,tr,wt,i,j,k,i,j,k);
  }
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"+(%f*v%d%d%d^(-1)*f%d%d%d^(-1)+%f*v%d%d%d^%f\n",
        z5,i,j,k,i,j,k,z6,i,j,k,alpha);
  fprintf(mus,"*f%d%d%d^%f*d%d%d%d^%f*%f)*x%d%d%d+",
        i,j,k,beta,i,j,k,gammaT,tr,i,j,k);
  fprintf(mus,"(%f*(%f-2*d%d%d%d)\n",z7,D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"*v%d%d%d^(-1)*f%d%d%d^(-1)",i,j,k,i,j,k);
  }
 }

 for (j=1;j<=Jk-2;j++)
```

```
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"+%f*10^(-3)*(%f-2*d%d%d%d)\n",z8,D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"*v%d%d%d^%f*f%d%d%d^%f*d%d%d%d^%f*%f)*x%d%d%d\n",
         i,j,k,alpha,i,j,k,beta,i,j,k,gammaT,tr,i,j,k);
  }
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"+(%f*(%f-2*d%d%d%d",z7,D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"-2*d%d%d%d)",i,j,k);
  }
 }

 for (j=3;j<=Jk;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"*v%d%d%d^(-1)*f%d%d%d^(-1)+%f*10^(-3)\n",
         i,j,k,i,j,k,z8);
  }
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=2;k<=Q-1;k++)
```

```
  {
  fprintf(mus,"*(%f-2*d%d%d%d",D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"-2*d%d%d%d)",i,j,k);
  }
 }

 for (j=3;j<=Jk;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"*v%d%d%d^%f\n",i,j,k,alpha);
  fprintf(mus,"*f%d%d%d^%f*d%d%d%d^%f*(%f+%f/y%d%d%d))*x%d%d%d\n",
         i,j,k,beta,i,j,k,gammaT,tr,wt,i,j,k,i,j,k);
  }
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"+(%f*v%d%d%d^(-1)*f%d%d%d^(-1)+%f*v%d%d%d^%f\n",
         z9,i,j,k,i,j,k,z10,i,j,k,alpha);
  fprintf(mus,"*f%d%d%d^%f*d%d%d%d^%f*%f)*x%d%d%d+",
         i,j,k,beta,i,j,k,gammaT,tr,i,j,k);
  fprintf(mus,"(%f*(%f-2*d%d%d%d)\n",z11,D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"*v%d%d%d^(-1)*f%d%d%d^(-1)",i,j,k,i,j,k);
  }
 }

 for (j=1;j<=Jk-2;j++)
```

```
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"+%f*10^(-3)*(%f-2*d%d%d%d)\n",z12,D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"*v%d%d%d^%f*f%d%d%d^%f*d%d%d%d^%f*%f)*x%d%d%d\n",
         i,j,k,alpha,i,j,k,beta,i,j,k,gammaT,tr,i,j,k);
  }
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"+(%f*(%f-2*d%d%d%d",z11,D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"-2*d%d%d%d)",i,j,k);
  }
 }

 for (j=3;j<=Jk;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"*v%d%d%d^(-1)*f%d%d%d^(-1)+%f*10^(-3)\n",
         i,j,k,i,j,k,z12);
  }
 }

 for (j=1;j<=Jk-2;j++)
 {
  for (k=3;k<=Q;k++)
```

```
  {
  fprintf(mus,"*(%f-2*d%d%d%d",D0[k],i,j,k);
  }
 }

 for (j=2;j<=Jk-1;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"-2*d%d%d%d)",i,j,k);
  }
 }

 for (j=3;j<=Jk;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"*v%d%d%d^%f\n",i,j,k,alpha);
  fprintf(mus,"*f%d%d%d^%f*d%d%d%d^%f*(%f+%f/y%d%d%d))",
        i,j,k,beta,i,j,k,gammaT,tr,wt,i,j,k,i,j,k,i);
  fprintf(mus,"*x%d%d%d<=Z;\n",i,j,k);
  }
 }
}

/*
This section generates cutting speed, feed rate, cutting
force and cutting power constraints
*/

for (i=1;i<=P;i++)
{
 for (j=1;j<=Jk;j++)
 {
  for (k=1;k<=Q;k++)
  {
  fprintf(mus,"v%d%d%d>=%f;\n",i,j,k,v_l[j]);
  fprintf(mus,"v%d%d%d<=%f;\n",i,j,k,v_u[j]);
  fprintf(mus,"f%d%d%d>=%f;\n",i,j,k,f_l[j]);
  fprintf(mus,"f%d%d%d<=%f;\n",i,j,k,f_u[j]);
  fprintf(mus,"%f*f%d%d%d^%f*d%d%d%d^%f<=%f;\n",kF,i,j,k,betaF,
        i,j,k,gammaF,Fc_u);
  fprintf(mus,"%f*10^2*v%d%d%d*f%d%d%d^%f*d%d%d%d^%f<=%f*10^2;\n",
        kP,i,j,k,i,j,k,betaF,i,j,k,gammaF,Pc_u);
```

```
   }
  }
 }

 /*
 This section generates depth of cut constraints
 */

 for (i=1;i<=P;i++)
 {
  for (j=1;j<=Jk;j++)
  {
   for (k=1;k<=Q;k++)
   {
    if ((i==k) && (j==Jk))
    fprintf(mus,"d%d%d%d>=%f;\n",i,j,k,d_l[j]);
    fprintf(mus,"d%d%d%d<=%f;\n",i,j,k,d_u[j]);
    }
   }
 }

 /*
 This section generates the sum of depth of cut constraints
 */

 for (i=1;i<=P-1;i++)
 {
   for (j=1;j<=Jk;j++)
   {
    for (k=1;k<=Q-2;k++)
    {
    fprintf(mus,"d%d%d%d+",i,j,k);
    }
   }
 }

 for (i=3;i<=P;i++)
 {
  for (j=1;j<=Jk-1;j++)
  {
   for (k=1;k<=Q-2;k++)
   {
   fprintf(mus,"d%d%d%d+",i,j,k);
   }
```

```
 }
}

for (i=3;i<=P;i++)
{
 for (j=3;j<=Jk;j++)
 {
  for (k=1;k<=Q-2;k++)
  {
  fprintf(mus,"d%d%d%d=%f;\n",i,j,k,dtotal[k]);
  }
 }
}

for (i=1;i<=P-1;i++)
{
 for (j=1;j<=Jk;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"d%d%d%d+",i,j,k);
  }
 }
}

for (i=3;i<=P;i++)
{
 for (j=1;j<=Jk-1;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"d%d%d%d+",i,j,k);
  }
 }
}

for (i=3;i<=P;i++)
{
 for (j=3;j<=Jk;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"d%d%d%d=%f;\n",i,j,k,dtotal[k]);
  }
```

```
  }
 }

 for (i=1;i<=P-1;i++)
 {
  for (j=1;j<=Jk;j++)
  {
   for (k=3;k<=Q;k++)
   {
   fprintf(mus,"d%d%d%d+",i,j,k);
   }
  }
 }

 for (i=3;i<=P;i++)
 {
  for (j=1;j<=Jk-1;j++)
  {
   for (k=3;k<=Q;k++)
   {
   fprintf(mus,"d%d%d%d+",i,j,k);
   }
  }
 }

 for (i=3;i<=P;i++)
 {
  for (j=3;j<=Jk;j++)
  {
   for (k=3;k<=Q;k++)
   {
   fprintf(mus,"d%d%d%d=%f;\n",i,j,k,dtotal[k]);
   }
  }
 }

 for (i=1;i<=P-1;i++)
 {
  for (j=1;j<=Jk;j++)
  {
   for (k=1;k<=Q-2;k++)
   {
   fprintf(mus,"d%d%d%d*x%d%d%d+",i,j,k,i,j,k);
   }
```

```
  }
 }

 for (i=3;i<=P;i++)
 {
  for (j=1;j<=Jk-1;j++)
  {
   for (k=1;k<=Q-2;k++)
   {
   fprintf(mus,"d%d%d%d*x%d%d%d+",i,j,k,i,j,k);
   }
  }
 }

 for (i=3;i<=P;i++)
 {
  for (j=3;j<=Jk;j++)
  {
   for (k=1;k<=Q-2;k++)
   {
   fprintf(mus,"d%d%d%d*x%d%d%d=%f;\n",i,j,k,i,j,k,dtotal[k]);
   }
  }
 }

 for (i=1;i<=P-1;i++)
 {
  for (j=1;j<=Jk;j++)
  {
   for (k=2;k<=Q-1;k++)
   {
   fprintf(mus,"d%d%d%d*x%d%d%d+",i,j,k,i,j,k);
   }
  }
 }

 for (i=3;i<=P;i++)
 {
  for (j=1;j<=Jk-1;j++)
  {
   for (k=2;k<=Q-1;k++)
   {
   fprintf(mus,"d%d%d%d*x%d%d%d+",i,j,k,i,j,k);
   }
```

```
 }
}

for (i=3;i<=P;i++)
{
 for (j=3;j<=Jk;j++)
 {
  for (k=2;k<=Q-1;k++)
  {
  fprintf(mus,"d%d%d%d*x%d%d%d=%f;\n",i,j,k,i,j,k,dtotal[k]);
  }
 }
}

for (i=1;i<=P-1;i++)
{
 for (j=1;j<=Jk;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"d%d%d%d*x%d%d%d+",i,j,k,i,j,k);
  }
 }
}

for (i=3;i<=P;i++)
{
 for (j=1;j<=Jk-1;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"d%d%d%d*x%d%d%d+",i,j,k,i,j,k);
  }
 }
}

for (i=3;i<=P;i++)
{
 for (j=3;j<=Jk;j++)
 {
  for (k=3;k<=Q;k++)
  {
  fprintf(mus,"d%d%d%d*x%d%d%d=%f;\n",i,j,k,i,j,k,dtotal[k]);
  }
```

```
  }
 }

 /*
 This section generates the constraints such that each
 operation is processed by only one machine
 */

 for (i=1;i<=P-1;i++)
 {
  for (j=1;j<=Jk-2;j++)
  {
   for (k=1;k<=Q-2;k++)
   {
   fprintf(mus,"x%d%d%d+",i,j,k);
   }
  }
 }

 for (i=3;i<=P;i++)
 {
  for (j=1;j<=Jk-2;j++)
  {
   for (k=1;k<=Q-2;k++)
   {
   fprintf(mus,"x%d%d%d=1;\n",i,j,k);
   }
  }
 }

 for (i=1;i<=P-1;i++)
 {
  for (j=2;j<=Jk-1;j++)
  {
   for (k=1;k<=Q-2;k++)
   {
   fprintf(mus,"x%d%d%d+",i,j,k);
   }
  }
 }

 for (i=3;i<=P;i++)
 {
  for (j=2;j<=Jk-1;j++)
```

```
  {
   for (k=1;k<=Q-2;k++)
   {
   fprintf(mus,"x%d%d%d=1;\n",i,j,k);
   }
  }
 }

 for (i=1;i<=P-1;i++)
 {
  for (j=3;j<=Jk;j++)
  {
   for (k=1;k<=Q-2;k++)
   {
   fprintf(mus,"x%d%d%d+",i,j,k);
   }
  }
 }

 for (i=3;i<=P;i++)
 {
  for (j=3;j<=Jk;j++)
  {
   for (k=1;k<=Q-2;k++)
   {
   fprintf(mus,"x%d%d%d=1;\n",i,j,k);
   }
  }
 }

 for (i=1;i<=P-1;i++)
 {
  for (j=1;j<=Jk-2;j++)
  {
   for (k=2;k<=Q-1;k++)
   {
   fprintf(mus,"x%d%d%d+",i,j,k);
   }
  }
 }

 for (i=3;i<=P;i++)
 {
  for (j=1;j<=Jk-2;j++)
```

```
  {
   for (k=2;k<=Q-1;k++)
   {
   fprintf(mus,"x%d%d%d=1;\n",i,j,k);
   }
  }
 }

 for (i=1;i<=P-1;i++)
 {
  for (j=2;j<=Jk-1;j++)
  {
   for (k=2;k<=Q-1;k++)
   {
   fprintf(mus,"x%d%d%d+",i,j,k);
   }
  }
 }

 for (i=3;i<=P;i++)
 {
  for (j=2;j<=Jk-1;j++)
  {
   for (k=2;k<=Q-1;k++)
   {
   fprintf(mus,"x%d%d%d=1;\n",i,j,k);
   }
  }
 }

 for (i=1;i<=P-1;i++)
 {
  for (j=3;j<=Jk;j++)
  {
   for (k=2;k<=Q-1;k++)
   {
   fprintf(mus,"x%d%d%d+",i,j,k);
   }
  }
 }

 for (i=3;i<=P;i++)
 {
  for (j=3;j<=Jk;j++)
```

```
  {
   for (k=2;k<=Q-1;k++)
   {
   fprintf(mus,"x%d%d%d=1;\n",i,j,k);
   }
  }
 }

 for (i=1;i<=P-1;i++)
 {
  for (j=1;j<=Jk-2;j++)
  {
   for (k=3;k<=Q;k++)
   {
   fprintf(mus,"x%d%d%d+",i,j,k);
   }
  }
 }

 for (i=3;i<=P;i++)
 {
  for (j=1;j<=Jk-2;j++)
  {
   for (k=3;k<=Q;k++)
   {
   fprintf(mus,"x%d%d%d=1;\n",i,j,k);
   }
  }
 }

 for (i=1;i<=P-1;i++)
 {
  for (j=2;j<=Jk-1;j++)
  {
   for (k=3;k<=Q;k++)
   {
   fprintf(mus,"x%d%d%d+",i,j,k);
   }
  }
 }

 for (i=3;i<=P;i++)
 {
  for (j=2;j<=Jk-1;j++)
```

```
  {
   for (k=3;k<=Q;k++)
   {
   fprintf(mus,"x%d%d%d=1;\n",i,j,k);
   }
  }
 }

 for (i=1;i<=P-1;i++)
 {
  for (j=3;j<=Jk;j++)
  {
   for (k=3;k<=Q;k++)
   {
   fprintf(mus,"x%d%d%d+",i,j,k);
   }
  }
 }

 for (i=3;i<=P;i++)
 {
  for (j=3;j<=Jk;j++)
  {
   for (k=3;k<=Q;k++)
   {
   fprintf(mus,"x%d%d%d=1;\n",i,j,k);
   }
  }
 }

 /*
 This section generates the constraints such that the last pass of each feature is
 always selected and each machine should perform at least one pass
 */

 for (i=1;i<=P;i++)
 {
  for (j=1;j<=Jk;j++)
  {
   for (k=1;k<=Q;k++)
   {
   if ((i==k) && (j==Jk))
   fprintf(mus,"x%d%d%d=1;\n",i,j,k);
   }
```

```
  }
 }

 /*
 This section generates dimension deviation constraints
 */

 for (i=1;i<=P;i++)
 {
  for (j=3;j<=Jk;j++)
  {
   for (k=1;k<=Q;k++)
   {
   fprintf(mus,"y%d%d%d>=0;\n",i,j,k);
   fprintf(mus,"y%d%d%d<=%f;\n",i,j,k,delta[k]);
   }
  }
 }

 /*
 This section generates surface finish constraints
 */

 for (i=1;i<=P;i++)
 {
  for (j=3;j<=Jk;j++)
  {
   for (k=1;k<=Q;k++)
   {
   fprintf(mus,"%f*v%d%d%d^%f*f%d%d%d^%f*d%d%d%d^%f<=%f;\n",kS,
         i,j,k,alphaS,i,j,k,betaS,i,j,k,gammaS,SF_u[k]);
   }
  }
 }

 fprintf(mus,"END\n");
 fprintf(mus,"BAT\n");
 return 0;

 }
```

Appendix J

LINGO-format-input for the example problem of Model 2.5

```
MODEL
min=Z;
(141.371552*v111^(-1)*f111^(-1)+0.000090*v111^0.700000
*f111^0.550000*d111^1.220000*0.500000)*x111+(0.942477*(150.000000-2*d111)
*v121^(-1)*f121^(-1)+0.000600*10^(-3)*(150.000000-2*d111)
*v121^0.700000*f121^0.550000*d121^1.220000*0.500000)*x121
+(0.942477*(150.000000-2*d111-2*d121)*v131^(-1)*f131^(-1)+0.000600*10^(-3)
*(150.000000-2*d111-2*d121)*v131^0.700000
*f131^0.550000*d131^1.220000*(0.500000+0.020000/y131))*x131
+(43.982262*v112^(-1)*f112^(-1)+0.000028*v112^0.700000
*f112^0.550000*d112^1.220000*0.500000)*x112+(0.314159*(140.000000-2*d112)
*v122^(-1)*f122^(-1)+0.000200*10^(-3)*(140.000000-2*d112)
*v122^0.700000*f122^0.550000*d122^1.220000*0.500000)*x122
+(0.314159*(140.000000-2*d112-2*d122)*v132^(-1)*f132^(-1)+0.000200*10^(-3)
*(140.000000-2*d112-2*d122)*v132^0.700000
*f132^0.550000*d132^1.220000*(0.500000+0.020000/y132))*x132
+(65.973389*v113^(-1)*f113^(-1)+0.000042*v113^0.700000
*f113^0.550000*d113^1.220000*0.500000)*x113+(0.471238*(140.000000-2*d113)
*v123^(-1)*f123^(-1)+0.000300*10^(-3)*(140.000000-2*d113)
*v123^0.700000*f123^0.550000*d123^1.220000*0.500000)*x123
+(0.471238*(140.000000-2*d113-2*d123)*v133^(-1)*f133^(-1)+0.000300*10^(-3)
*(140.000000-2*d113-2*d123)*v133^0.700000
*f133^0.550000*d133^1.220000*(0.500000+0.020000/y133))*x133<=Z;
(141.371552*v211^(-1)*f211^(-1)+0.000090*v211^0.700000
*f211^0.550000*d211^1.220000*0.500000)*x211+(0.942477*(150.000000-2*d211)
*v221^(-1)*f221^(-1)+0.000600*10^(-3)*(150.000000-2*d211)
*v221^0.700000*f221^0.550000*d221^1.220000*0.500000)*x221
+(0.942477*(150.000000-2*d211-2*d221)*v231^(-1)*f231^(-1)+0.000600*10^(-3)
*(150.000000-2*d211-2*d221)*v231^0.700000
*f231^0.550000*d231^1.220000*(0.500000+0.020000/y231))*x231
+(43.982262*v212^(-1)*f212^(-1)+0.000028*v212^0.700000
*f212^0.550000*d212^1.220000*0.500000)*x212+(0.314159*(140.000000-2*d212)
*v222^(-1)*f222^(-1)+0.000200*10^(-3)*(140.000000-2*d212)
*v222^0.700000*f222^0.550000*d222^1.220000*0.500000)*x222
```

```
+(0.314159*(140.000000-2*d212-2*d222)*v232^(-1)*f232^(-1)+0.000200*10^(-3)
*(140.000000-2*d212-2*d222)*v232^0.700000
*f232^0.550000*d232^1.220000*(0.500000+0.020000/y232))*x232
+(65.973389*v213^(-1)*f213^(-1)+0.000042*v213^0.700000
*f213^0.550000*d213^1.220000*0.500000)*x213+(0.471238*(140.000000-2*d213)
*v223^(-1)*f223^(-1)+0.000300*10^(-3)*(140.000000-2*d213)
*v223^0.700000*f223^0.550000*d223^1.220000*0.500000)*x223
+(0.471238*(140.000000-2*d213-2*d223)*v233^(-1)*f233^(-1)+0.000300*10^(-3)
*(140.000000-2*d213-2*d223)*v233^0.700000
*f233^0.550000*d233^1.220000*(0.500000+0.020000/y233))*x233<=Z;
(141.371552*v311^(-1)*f311^(-1)+0.000090*v311^0.700000
*f311^0.550000*d311^1.220000*0.500000)*x311+(0.942477*(150.000000-2*d311)
*v321^(-1)*f321^(-1)+0.000600*10^(-3)*(150.000000-2*d311)
*v321^0.700000*f321^0.550000*d321^1.220000*0.500000)*x321
+(0.942477*(150.000000-2*d311-2*d321)*v331^(-1)*f331^(-1)+0.000600*10^(-3)
*(150.000000-2*d311-2*d321)*v331^0.700000
*f331^0.550000*d331^1.220000*(0.500000+0.020000/y331))*x331
+(43.982262*v312^(-1)*f312^(-1)+0.000028*v312^0.700000
*f312^0.550000*d312^1.220000*0.500000)*x312+(0.314159*(140.000000-2*d312)
*v322^(-1)*f322^(-1)+0.000200*10^(-3)*(140.000000-2*d312)
*v322^0.700000*f322^0.550000*d322^1.220000*0.500000)*x322
+(0.314159*(140.000000-2*d312-2*d322)*v332^(-1)*f332^(-1)+0.000200*10^(-3)
*(140.000000-2*d312-2*d322)*v332^0.700000
*f332^0.550000*d332^1.220000*(0.500000+0.020000/y332))*x332
+(65.973389*v313^(-1)*f313^(-1)+0.000042*v313^0.700000
*f313^0.550000*d313^1.220000*0.500000)*x313+(0.471238*(140.000000-2*d313)
*v323^(-1)*f323^(-1)+0.000300*10^(-3)*(140.000000-2*d313)
*v323^0.700000*f323^0.550000*d323^1.220000*0.500000)*x323
+(0.471238*(140.000000-2*d313-2*d323)*v333^(-1)*f333^(-1)+0.000300*10^(-3)
*(140.000000-2*d313-2*d323)*v333^0.700000
*f333^0.550000*d333^1.220000*(0.500000+0.020000/y333))*x333<=Z;
v111>=90.000000;
v111<=120.000000;
f111>=0.800000;
f111<=2.000000;
1.380000*f111^1.180000*d111^1.260000<=20.000000;
0.000282*10^2*v111*f111^1.180000*d111^1.260000<=2.000000*10^2;
v112>=90.000000;
v112<=120.000000;
f112>=0.800000;
f112<=2.000000;
1.380000*f112^1.180000*d112^1.260000<=20.000000;
0.000282*10^2*v112*f112^1.180000*d112^1.260000<=2.000000*10^2;
v113>=90.000000;
```

v113<=120.000000;
f113>=0.800000;
f113<=2.000000;
1.380000*f113^1.180000*d113^1.260000<=20.000000;
0.000282*10^2*v113*f113^1.180000*d113^1.260000<=2.000000*10^2;
v121>=90.000000;
v121<=120.000000;
f121>=0.800000;
f121<=2.000000;
1.380000*f121^1.180000*d121^1.260000<=20.000000;
0.000282*10^2*v121*f121^1.180000*d121^1.260000<=2.000000*10^2;
v122>=90.000000;
v122<=120.000000;
f122>=0.800000;
f122<=2.000000;
1.380000*f122^1.180000*d122^1.260000<=20.000000;
0.000282*10^2*v122*f122^1.180000*d122^1.260000<=2.000000*10^2;
v123>=90.000000;
v123<=120.000000;
f123>=0.800000;
f123<=2.000000;
1.380000*f123^1.180000*d123^1.260000<=20.000000;
0.000282*10^2*v123*f123^1.180000*d123^1.260000<=2.000000*10^2;
v131>=168.000000;
v131<=210.000000;
f131>=0.130000;
f131<=0.500000;
1.380000*f131^1.180000*d131^1.260000<=20.000000;
0.000282*10^2*v131*f131^1.180000*d131^1.260000<=2.000000*10^2;
v132>=168.000000;
v132<=210.000000;
f132>=0.130000;
f132<=0.500000;
1.380000*f132^1.180000*d132^1.260000<=20.000000;
0.000282*10^2*v132*f132^1.180000*d132^1.260000<=2.000000*10^2;
v133>=168.000000;
v133<=210.000000;
f133>=0.130000;
f133<=0.500000;
1.380000*f133^1.180000*d133^1.260000<=20.000000;
0.000282*10^2*v133*f133^1.180000*d133^1.260000<=2.000000*10^2;
v211>=90.000000;
v211<=120.000000;
f211>=0.800000;

f211<=2.000000;
1.380000*f211^1.180000*d211^1.260000<=20.000000;
0.000282*10^2*v211*f211^1.180000*d211^1.260000<=2.000000*10^2;
v212>=90.000000;
v212<=120.000000;
f212>=0.800000;
f212<=2.000000;
1.380000*f212^1.180000*d212^1.260000<=20.000000;
0.000282*10^2*v212*f212^1.180000*d212^1.260000<=2.000000*10^2;
v213>=90.000000;
v213<=120.000000;
f213>=0.800000;
f213<=2.000000;
1.380000*f213^1.180000*d213^1.260000<=20.000000;
0.000282*10^2*v213*f213^1.180000*d213^1.260000<=2.000000*10^2;
v221>=90.000000;
v221<=120.000000;
f221>=0.800000;
f221<=2.000000;
1.380000*f221^1.180000*d221^1.260000<=20.000000;
0.000282*10^2*v221*f221^1.180000*d221^1.260000<=2.000000*10^2;
v222>=90.000000;
v222<=120.000000;
f222>=0.800000;
f222<=2.000000;
1.380000*f222^1.180000*d222^1.260000<=20.000000;
0.000282*10^2*v222*f222^1.180000*d222^1.260000<=2.000000*10^2;
v223>=90.000000;
v223<=120.000000;
f223>=0.800000;
f223<=2.000000;
1.380000*f223^1.180000*d223^1.260000<=20.000000;
0.000282*10^2*v223*f223^1.180000*d223^1.260000<=2.000000*10^2;
v231>=168.000000;
v231<=210.000000;
f231>=0.130000;
f231<=0.500000;
1.380000*f231^1.180000*d231^1.260000<=20.000000;
0.000282*10^2*v231*f231^1.180000*d231^1.260000<=2.000000*10^2;
v232>=168.000000;
v232<=210.000000;
f232>=0.130000;
f232<=0.500000;
1.380000*f232^1.180000*d232^1.260000<=20.000000;

```
0.000282*10^2*v232*f232^1.180000*d232^1.260000<=2.000000*10^2;
v233>=168.000000;
v233<=210.000000;
f233>=0.130000;
f233<=0.500000;
1.380000*f233^1.180000*d233^1.260000<=20.000000;
0.000282*10^2*v233*f233^1.180000*d233^1.260000<=2.000000*10^2;
v311>=90.000000;
v311<=120.000000;
f311>=0.800000;
f311<=2.000000;
1.380000*f311^1.180000*d311^1.260000<=20.000000;
0.000282*10^2*v311*f311^1.180000*d311^1.260000<=2.000000*10^2;
v312>=90.000000;
v312<=120.000000;
f312>=0.800000;
f312<=2.000000;
1.380000*f312^1.180000*d312^1.260000<=20.000000;
0.000282*10^2*v312*f312^1.180000*d312^1.260000<=2.000000*10^2;
v313>=90.000000;
v313<=120.000000;
f313>=0.800000;
f313<=2.000000;
1.380000*f313^1.180000*d313^1.260000<=20.000000;
0.000282*10^2*v313*f313^1.180000*d313^1.260000<=2.000000*10^2;
v321>=90.000000;
v321<=120.000000;
f321>=0.800000;
f321<=2.000000;
1.380000*f321^1.180000*d321^1.260000<=20.000000;
0.000282*10^2*v321*f321^1.180000*d321^1.260000<=2.000000*10^2;
v322>=90.000000;
v322<=120.000000;
f322>=0.800000;
f322<=2.000000;
1.380000*f322^1.180000*d322^1.260000<=20.000000;
0.000282*10^2*v322*f322^1.180000*d322^1.260000<=2.000000*10^2;
v323>=90.000000;
v323<=120.000000;
f323>=0.800000;
f323<=2.000000;
1.380000*f323^1.180000*d323^1.260000<=20.000000;
0.000282*10^2*v323*f323^1.180000*d323^1.260000<=2.000000*10^2;
v331>=168.000000;
```

```
v331<=210.000000;
f331>=0.130000;
f331<=0.500000;
1.380000*f331^1.180000*d331^1.260000<=20.000000;
0.000282*10^2*v331*f331^1.180000*d331^1.260000<=2.000000*10^2;
v332>=168.000000;
v332<=210.000000;
f332>=0.130000;
f332<=0.500000;
1.380000*f332^1.180000*d332^1.260000<=20.000000;
0.000282*10^2*v332*f332^1.180000*d332^1.260000<=2.000000*10^2;
v333>=168.000000;
v333<=210.000000;
f333>=0.130000;
f333<=0.500000;
1.380000*f333^1.180000*d333^1.260000<=20.000000;
0.000282*10^2*v333*f333^1.180000*d333^1.260000<=2.000000*10^2;
d111<=5.000000;
d112<=5.000000;
d113<=5.000000;
d121<=5.000000;
d122<=5.000000;
d123<=5.000000;
d131>=0.300000;
d131<=1.000000;
d132<=1.000000;
d133<=1.000000;
d211<=5.000000;
d212<=5.000000;
d213<=5.000000;
d221<=5.000000;
d222<=5.000000;
d223<=5.000000;
d231<=1.000000;
d232>=0.300000;
d232<=1.000000;
d233<=1.000000;
d311<=5.000000;
d312<=5.000000;
d313<=5.000000;
d321<=5.000000;
d322<=5.000000;
d323<=5.000000;
d331<=1.000000;
```

```
d332<=1.000000;
d333>=0.300000;
d333<=1.000000;
d111+d121+d131+d211+d221+d231+d311+d321+d331=5.000000;
d112+d122+d132+d212+d222+d232+d312+d322+d332=8.000000;
d113+d123+d133+d213+d223+d233+d313+d323+d333=10.000000;
d111*x111+d121*x121+d131*x131+d211*x211+d221*x221+d231*x231+
d311*x311+d321*x321+d331*x331=5.000000;
d112*x112+d122*x122+d132*x132+d212*x212+d222*x222+d232*x232+
d312*x312+d322*x322+d332*x332=8.000000;
d113*x113+d123*x123+d133*x133+d213*x213+d223*x223+d233*x233+
d313*x313+d323*x323+d333*x333=10.000000;
x111+x211+x311=1;
x121+x221+x321=1;
x131+x231+x331=1;
x112+x212+x312=1;
x122+x222+x322=1;
x132+x232+x332=1;
x113+x213+x313=1;
x123+x223+x323=1;
x133+x233+x333=1;
x131=1;
x232=1;
x333=1;
y131>=0;
y131<=0.065900;
y132>=0;
y132<=0.065900;
y133>=0;
y133<=0.065900;
y231>=0;
y231<=0.065900;
y232>=0;
y232<=0.065900;
y233>=0;
y233<=0.065900;
y331>=0;
y331<=0.065900;
y332>=0;
y332<=0.065900;
y333>=0;
y333<=0.065900;
1.170000*v131^-0.250000*f131^0.720000*d131^0.230000<=1.600000;
1.170000*v132^-0.250000*f132^0.720000*d132^0.230000<=1.600000;
```

```
1.170000*v133^-0.250000*f133^0.720000*d133^0.230000<=1.600000;
1.170000*v231^-0.250000*f231^0.720000*d231^0.230000<=1.600000;
1.170000*v232^-0.250000*f232^0.720000*d232^0.230000<=1.600000;
1.170000*v233^-0.250000*f233^0.720000*d233^0.230000<=1.600000;
1.170000*v331^-0.250000*f331^0.720000*d331^0.230000<=1.600000;
1.170000*v332^-0.250000*f332^0.720000*d332^0.230000<=1.600000;
1.170000*v333^-0.250000*f333^0.720000*d333^0.230000<=1.600000;
END
BAT
```